The Complexity of Reasoning with FODD and GFODD[☆]

Benjamin J. Hescott, Roni Khardon

Department of Computer Science, Tufts University, Medford, MA 02155, USA

Abstract

Recent work introduced Generalized First Order Decision Diagrams (GFODD) as a knowledge representation language that is useful in mechanizing decision theoretic planning in relational domains. GFODDs generalize function-free first order logic and include numerical values and numerical generalizations of existential and universal quantification. Previous work presented heuristic inference algorithms for GFODDs and implemented these heuristics in systems for decision theoretic planning. In this paper, we study the complexity of the computational problems addressed by such implementations. In particular, we study the evaluation problem, the satisfiability problem, and the equivalence problem for GFODDs under the assumption that the size of the intended model is given with the problem, a restriction that guarantees decidability. Our results provide a complete characterization placing these problems within the polynomial hierarchy. The same characterization applies to the corresponding restriction of problems in first order logic, giving an interesting new avenue for efficient inference when the number of objects is bounded. Our results show that for Σ_k formulas, and for corresponding GFODDs, evaluation and satisfiability are Σ_k^P complete, and equivalence is Π_{k+1}^P complete. For Π_k formulas evaluation is Π_k^P complete, satisfiability is one level higher and is Σ_{k+1}^P complete, and equivalence is Π_{k+1}^P complete.

Keywords: Complexity Analysis, Polynomial Hierarchy, Decision Diagrams, First Order Logic

1. Introduction

The complexity of inference in first order logic has been investigated intensively. It is well known that the problem is undecidable, and that this holds

[☆]A preliminary version of this paper has appeared as [10]. This paper includes a broader exposition and a significant amount of additional details in proofs and constructions required to obtain the technical results.

Email addresses: hescott@cs.tufts.edu (Benjamin J. Hescott), roni@cs.tufts.edu (Roni Khardon)

even with strong restrictions on the types and number of predicates allowed in the logical language. For example, the problem is undecidable for quantifier prefix $\forall^2\exists^*$ with a signature having a single binary predicate and equality [8]. Unfortunately, the problem is also undecidable if we restrict attention to satisfiability under finite structures [6, 24]. Thus, in either case, one cannot quantify the relative difficulty of problems without further specialization or assumptions. On the other hand, algorithmic progress in AI has made it possible to reason efficiently in some cases. In this paper we study such problems under the additional restriction that an upper bound on the intended model size is given explicitly. This restriction is natural for many applications, where the number of objects is either known in advance or known to be bounded by some quantity. Since the inference problem is decidable under this restriction, we can provide a more detailed complexity analysis.

This paper is motivated by recent work on decision diagrams, known as FODDs and GFODDs, and the computational questions associated with them. Binary decision diagrams [3, 1] provide a successful representation language capturing functions over propositional variables, that allows for efficient manipulation and composition of functions, and diagrams have been used in various applications in program verification and AI [3, 1, 11]. Motivated by this success, several authors have attempted generalizations to handle relational structure and first order quantification [9, 33, 30, 16]. In particular FODDs [33] and their generalization GFODDs [16] have been introduced and shown to be useful in the context of decision theoretic planning [2, 20, 12, 13] for problems with relational structure [15, 17].

GFODDs can be seen to generalize the function-free portion of first order logic (i.e., signatures with constants but without higher arity functions) to allow for non-binary numerical values generalizing truth values, and for numerical quantifiers generalizing existential and universal quantification in logic. Efficient heuristic inference algorithms for such diagrams have been developed focusing on the finite model case, and using the notion of "reasoning from examples" [22, 23, 21]. This paper analyses the complexity of the evaluation, satisfiability, and equivalence problems for such diagrams, focusing on the GFODD subset with min and max aggregation that are defined in the next section. To avoid undecidability and get a more refined classification of complexity, we study a restricted form of the problem where the finite size of the intended model is given as part of the input to the problem. As we argue below, this is natural and relevant in the applications of GFODDs for solving decision theoretic control problems. The same restrictions can be used for the corresponding (evaluation, satisfiability and equivalence) problems in first order logic, but to our knowledge this has not been studied before. We provide a complete characterization of the complexity showing an interesting structure. Our results are developed for the GFODD representation and require detailed arguments about the graphical representation of formulas in that language. The same lines of argument (with simpler proof details) yield similar results for first order logic. To translate our results to the language of logic, consider the quantifier prefix of a first order logic formula using the standard notation using Σ_k, Π_k to denote alternation

depth of quantifiers in the formula. With this translation, our results show that:

(1) Evaluation over finite structures spans the polynomial hierarchy, that is, evaluation of Σ_k formulas is Σ_k^P complete, and evaluation of Π_k formulas is Π_k^P complete.

(2) Satisfiability, with a given bound on model size, follows a different pattern: satisfiability of Σ_k formulas is Σ_k^P complete, and satisfiability of Π_k formulas is Σ_{k+1}^P complete.

(3) Equivalence, under the set of models bounded by a given size, depends only on quantifier depth: both the equivalence of Σ_k formulas and equivalence of Π_k formulas are Π_{k+1}^P complete.

The positive results allow for constants in the signature but the hardness results, except for satisfiability for Π_1 formulas, hold even without constants. For signatures without constants, satisfiability of Π_1 formulas is in NP; when constants are allowed, it is Σ_2^P complete as in the general template.

These results are useful in that they clearly characterize the complexity of the problems solved heuristically by implementations of GFODD systems [15, 17] and can be used to partly motivate or justify the use of these heuristics. For example, the "model checking reductions" of [16] that simplify the structure of diagrams replace equivalence tests with model evaluation on a "representative" set of models. When this set is chosen heuristically, as in [15], this leads to inference that is correct with respect to these models but otherwise incomplete. Our results show that this indeed leads to a reduction of the complexity of the inference problem, so that the loss in accuracy is traded for improved worst case run time. Importantly, it shows that without compromising correctness, the complexity of equivalence tests that are used to compress the representation will be higher. These issues and further questions for future work are discussed in the concluding section of the paper.

The rest of the paper is organized as follows. The next section defines FODDs and GFODDs and provides a more detailed motivation for the technical questions. Section 3 then develops the results for FODDs. We treat the FODD case separately for three reasons. First, this serves for an easy introduction into the results that avoids some of the more involved arguments that are required for GFODDs. Second, as will become clear, for FODDs we do not need the additional assumption on model size, so that the results are in a sense stronger. Finally, some of the proofs for GFODDs require alternation depth of at least two so that separate proofs are needed for FODDs in any case. Section 4 develops the results for GFODDs. The final section concludes with a discussion and directions for future work.

2. FODDs and GFODDs and their Computational Problems

This section introduces the GFODD representation language and associated computational problems, and explains how they are motivated by prior work on applying GFODDs in decision theoretic planning. We assume familiarity with basic concepts and notation in predicate logic [25, 29, 4] as well as basic notions from complexity theory [14, 32, 26].

Decision diagrams are similar to expressions in first order logic (FOL). They are defined relative to a relational signature, with a finite set of predicates $p_1, p_2, \ldots, p_n$ each with an associated arity (number of arguments), a countable set of variables $x_1, x_2, \ldots$, and a set of constants $c_1, c_2, \ldots, c_m$. We do not allow function symbols other than constants (that is, functions with arity ≥ 1). In addition, we assume that the arity of predicates is bounded by some numerical constant. A term is a variable or constant and an atom is either an equality between two terms or a predicate with an appropriate list of terms as arguments. Intuitively, a term refers to an object in the world of interest and an atom is a property which is either true or false.

To motivate the diagram representation consider first a simpler language of generalized expressions which we illustrate informally by some examples. In FOL we can consider open formulas that have unbound variables. For example, the atom $color(x, y)$ is such a formula and its truth value depends on the assignment of x and y to objects in the world. To simplify the discussion, we assume for this example that arguments are typed and x ranges over "objects" and y over "colors". We can then quantify over these variables to get a sentence which will be evaluated to a truth value in any concrete possible world. For example, we can write $\exists y, \forall x, color(x, y)$ expressing the statement that there is a color associated with all objects. Generalized expressions allow for more general open formulas that evaluate to numerical values. For example, $E_1 =$ [if $color(x, y)$ then 1 else 0] is similar to the logical expression and $E_2 =$ [if $color(x, y)$ then 0.3 else 0.5] returns non binary values. Quantifiers from logic are replaced with aggregation operators that combine numerical values and provide a generalization of the logical constructs. In particular, when the open formula is restricted to values 0 and 1, the operators max and min simulate existential and universal quantification. Thus, $[\max_y, \min_x,$ if $color(x, y)$ then 1 else 0] is equivalent to the logical sentence $\exists y, \forall x, color(x, y)$ given above. But we can allow for other types of aggregations. For example, $[\max_y, \text{sum}_x,$ if $color(x, y)$ then 1 else 0] evaluates to the largest number of objects associated with one color, and the expression $[\text{sum}_x, \min_y,$ if $color(x, y)$ then 0 else 1] evaluates to the number of objects that have no color association. GFODDs are also related to work in statistical relational learning [28, 27, 5]. For example, if the expression E_2 captures probability of ground facts for the predicate $color()$ and the ground facts are mutually independent then $[\text{product}_x, \text{product}_y,$ if $color(x, y)$ then 0.3 else 0.5] captures the joint probability for all facts for $color()$. Of course, the open formulas in logic can include more than one atom and similarly expressions can be more involved. In this manner, a generalized expression represents a function from possible worlds to numerical values. GFODDs capture the same set of functions but provide an alternative representation for the open formulas through directed graphs. GFODDs were introduced together with a set of operations that can be used to manipulate and combine functions and in this way provide a tool for computation with numerical functions over possible worlds. Prior work includes implementation of the FODD fragment where the only aggregation operator allowed is max [17, 15] and more recently implementations for GFODDs

with max and average aggregations [19, 18]. In this paper we investigate several computational questions for GFODDs with min and max aggregation.

2.1. Syntax

First order decision diagrams (FODD) and their generalization (GFODD) were defined by [33, 16] inspired by previous work in [9]. GFODDs are composed of two parts, including the aggregation functions and the open formula portion which is captured by a diagram or graph. The aggregation portion is given by a listing of the variables in the diagram in some explicit order $(w_{i_1}, \ldots, w_{i_m})$ and a corresponding list of length m specifying aggregation over each w_{i_j}. In this paper we restrict aggregation operators for each variable to be min or max. To reflect the structure of GFODDs, and distinguish between aggregation list V and the graph portion of a diagram B, we sometimes denote a GFODD by $\langle V, B\rangle$. For example, in the expression $[\max_y, \min_x, \text{if } color(x, y) \text{ then 1 else 0}]$, V corresponds to $[\max_y, \min_x]$ and B corresponds to $[\text{if } color(x, y) \text{ then 1 else 0}]$. However, when clear from the context we use B as a shorthand for $\langle V, B\rangle$. FODDs are a special case of GFODDs where the aggregation function is max for all variables. Due to associativity and commutativity of max, the aggregation function for FODDs does not need to be represented explicitly.

As in propositional decision diagrams [3, 1], the diagram portion is a rooted acyclic graph with directed edges. Each node in the graph is labeled. A non-leaf node is labeled with an atom from the signature and it has exactly two outgoing edges. The directed edges correspond to the truth values of the node's atom. A leaf is labeled with a non-negative numerical value. We sometimes restrict diagrams to have only binary leaves with values 0 or 1. In this case we can consider the values to be the logical values false and true. An example diagram is shown in Figure 1. In this diagram and all other diagrams in this paper, left going edges denote the true branch out of a node and right going edges represent the false branch.

Similar to the propositional case [3, 1], GFODD syntax is restricted to comply with a predefined total order on atoms. In the propositional case the ordering constraint yields a normal form (a unique minimal representation for each function) which is in turn the main source of efficient reasoning. For GFODDs, a normal form has not been established but the use of ordering makes for more efficient simplification of diagrams. In particular, following [33], we assume a fixed ordering on predicate names, e.g., $p_1 \prec p_2 \prec \ldots \prec p_n$, and a fixed ordering on variable names, e.g., $x_1 \prec x_2 \prec \ldots$ and constants $c_1 \prec c_2 \prec \ldots c_m$ and require that $c_i \prec x_j$ for all i and j. The order is extended to atoms by considering them as lists. That is, $p_i(\ldots) \prec p_j(\ldots)$ if $i < j$ and $p_i(x_{k_1}, \ldots, x_{k_a}) \prec p_i(x_{k'_1}, \ldots, x_{k'_a})$ if $(x_{k_1}, \ldots, x_{k_a}) \prec (x_{k'_1}, \ldots, x_{k'_a})$ in the lexicographic ordering over lists. Node labels in the GFODD must obey this order so that if node a is above node b in the diagram then the labels satisfy $a \prec b$. The example of Figure 1 is ordered with predicate ordering $E \prec$ " = " and lexicographic variable ordering $v_1 \prec v_2 \prec v_3$.

The ordering assumption is helpful when constructing systems using GFODDs because it simplifies the computations. Our complexity results hold in gen-

eral, whether the assumption holds or not, therefore showing that while the assumption is convenient it does not fundamentally change the complexity of the problems. In particular, for the positive results, the algorithms showing membership in various complexity classes hold even in the more general case when the diagrams are not sorted. For the hardness results, the reductions developed hold even in the more restricted case when the diagrams are sorted. A significant amount of details in our analysis is devoted to handling ordering issues in hardness results.

Our complexity analysis will use the following classification of GFODD into subclasses. We say that a GFODD is a max-k-alternating GFODD if its set of aggregation operators has k blocks of aggregation operators, where the first includes max aggregation, the second includes min aggregation, and so on. We similarly define min-k-alternating GFODD where the first block has min aggregation operators. A GFODD has aggregation depth k if it is in one of these two classes. As mentioned above, FODD is the class of max-1-alternating GFODD. We also refer to min-1-alternating GFODD as min GFODD.

2.2. Semantics

Diagrams, like first order formulas, are evaluated in possible worlds that provide an interpretation of their symbols.[1] In particular, a possible world or *Interpretation* , I, specifies a domain of objects, an assignment of each constant in the signature to an object in the domain, and the truth values of predicates over these objects.

The semantics assigns a value, denoted $\mathrm{MAP}_B(I)$, for any diagram B on any interpretation I by considering all possible valuations. A variable valuation ζ is a mapping from the set of variables in B to domain elements in the interpretation I. This mapping assigns each node label to a concrete ("ground") atom in the interpretation and therefore to its truth value and in this way defines a single path from root to leaf. The value of this leaf is the value of the GFODD B under the interpretation, I, with variable valuation ζ and is denoted $\mathrm{MAP}_B(I, \zeta)$. The final value, $\mathrm{MAP}_B(I)$, is defined by aggregating over $\mathrm{MAP}_B(I, \zeta)$. In particular, considering the aggregation order $(w_{i_1}, \ldots, w_{i_m})$ we loop with j taking values from m to 1 aggregating values over w_{i_j} using its aggregation operator. We denote this by $\mathrm{MAP}_B(I) = AG_\zeta \mathrm{MAP}_B(I, \zeta)$ where for the special case of FODDs this yields $\mathrm{MAP}_B(I) = \max_\zeta \mathrm{MAP}_B(I, \zeta)$.

Consider evaluating the FODD example in Figure 1 on interpretation $I = ([1, 2, 3], \{E(1, 3), E(3, 1), E(1, 2), E(2, 1)\})$. Then for $\zeta = \{v_1/1, v_2/2, v_3/3\}$ we have $\mathrm{MAP}_B(I, \zeta) = 0$ but for $\zeta = \{v_1/3, v_2/1, v_3/2\}$ we have $\mathrm{MAP}_B(I, \zeta) = 1$ and therefore $\mathrm{MAP}_B(I) = \max_\zeta \mathrm{MAP}_B(I, \zeta) = 1$.

[1] Possible worlds are known in the literature under various names including *first order structures*, *first order models*, and *interpretations*. In this paper we use the term interpretations.

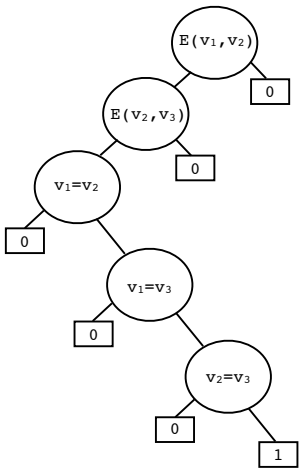

Figure 1: An example FODD. In this and all other diagrams in this paper, left going edges represent the true branch out of a node and right going edges represent the false branch. Interpreting E as an edge relation of a graph, this FODD tests that the graph has a Hamiltonian path of length 3.

2.3. Computations with GFODDs

The GFODD representation language was introduced as a tool for mechanizing and solving decision problems given by structured Markov Decision Processes (MDP), also known as Relational MDP or First Order MDP. A detailed exposition is beyond the scope of this paper (see [33, 16]). This section provides some necessary technical details and some background to motivate the computational problems investigated in the paper. In this context, a planning problem world state can be described using an interpretation providing the objects in the world and the relations among them. An action moves the world from one state to another, where in MDPs this transition is non-deterministic. The so-called Q function $Q(s, a)$ provides a quality estimate of each action a in each state s. Using this function, one can control the MDP by picking $a = \text{argmax}_a Q(s, a)$ in state s. There are several algorithms to calculate such Q functions and previous work has introduced GFODDs as a compact representation for these functions. This is done by implementing a symbolic version of the well known Value Iteration (VI) algorithm, where the symbolic algorithm operates by manipulating GFODDs. Action selection provides our first computational question, that is, evaluating $Q(s, a)$. In our context, this means calculating $\text{MAP}_B(I)$ where I captures s and a and B is the representation of the Q function. The same computational problem occurs in several other steps in the symbolic VI algorithm. We define this problem below as GFODD Evaluation.

Recall that a GFODD represents a function from interpretations to real values. One of the main operations required for the symbolic VI algorithm is combination of such functions. In particular, let f_1 and f_2 be functions represented by two GFODDs, and let $\odot$ be any binary operation over real values

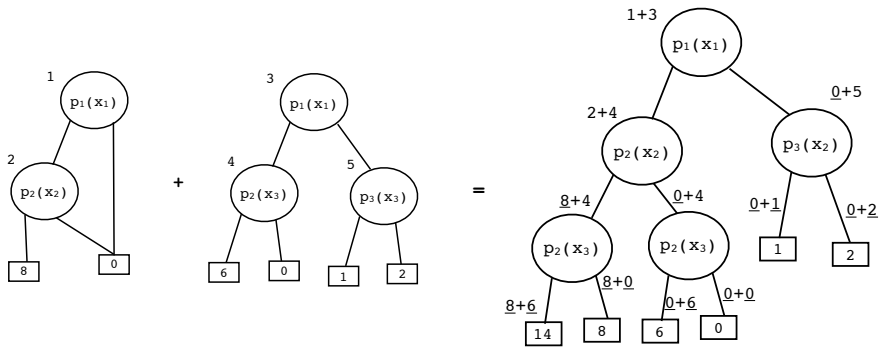

Figure 2: Overview of the Apply procedure of [33] for binary operations over two diagrams. Recall that GFODDs use an ordering over the atoms labeling nodes, so that atoms lower in the ordering are always higher in the diagram. Let p and q be the roots of B_1 and B_2 respectively. The procedure chooses a new root label (the smaller of p and q) and recursively combines the corresponding sub-diagrams, according to the relation between the two labels ($\prec$, $=$, or $\succ$). In this example, we assume predicate ordering $p_1 \prec p_2 \prec p_3$, and parameter ordering $x_1 \prec x_2 \prec x_3$. Note that the two diagrams share the argument x_1 as would be the case with constants but x_2 and x_3 are unique to one diagram. In this example, non-leaf nodes are annotated with numbers and numerical leaves are underlined for identification during the execution trace. For example, the top level call adds the functions corresponding to nodes 1 and 3. Since the label at these nodes, $p_1(x_1)$, is identical, we add the two left children and the two right children respectively. To illustrate another case, consider the node marked $2+4$. Here we pick the smaller label from 2, and then add both left and right child of node 2 to node 4. These calls are performed recursively and dynamic programming is used to avoid repeated recursive calls where such repetitions arise (this does not happen in the current example).

(e.g., addition). The combination operation returns a GFODD representing a function f_3 such that for all I we have $f_3(I) = f_1(I) \odot f_2(I)$. That is, f_3 is a symbolic representation of the pointwise operation over function values of f_1 and f_2. Note that since f_1 and f_2 are closed expressions we can standardize apart their variables before taking this operation.

Figure 2 shows how to combine the diagram portions (i.e., the open expressions) in a semantically coherent manner using the Apply procedure of [33]. The following theorem identifies conditions for correctness of Apply when used with closed expressions. We say that a binary operation $\odot$ is safe with respect to aggregation operator agg if it distributes with respect to it, that is $b \odot agg\{a_1, a_2, \ldots, a_n\} = agg\{(a_1 \odot b), (a_2 \odot b), \ldots, (a_n \odot b)\}$. A list of safe pairs of binary operations and aggregation operators was provided by [16]. For the arguments of this paper we recall that the binary operations $+$ and $\wedge$ are safe with respect to max and min aggregation. For example $5 + \max\{1, 2, 3, 4\} = \max\{6, 7, 8, 9\}$. With this definition we have:

Theorem 1 (see Theorem 4 of [16]). *Let $B_1 = \langle V_1, D_1 \rangle$ and $B_2 = \langle V_2, D_2 \rangle$ be GFODDs that do not share any variables and assume that op_c is safe with respect to all operators in V_1 and V_2. Let $D = apply(B_1, B_2, op_c)$. Let V be any*

permutation of the list of variable in V_1 and V_2 so long as the relative order of operators in V_1 and V_2 remains unchanged, and let $B = \langle V, D \rangle$. Then for any interpretation I, $MAP_B(I) = MAP_{B_1}(I)\ op_c\ MAP_{B_2}(I)$.

Therefore, when adding (or taking the logical-and of) functions represented by diagrams that are standardized apart we can use the Apply procedure on the graphical representations of these functions, and at the same time we have some flexibility in putting together their list of aggregation functions. This will be useful in our reductions.

The Apply procedure can introduce redundancy into diagrams. By this we mean that a simpler syntactic form, often a sub-diagram, can represent the same function. To illustrate, consider the diagrams of Figure 2 as FODDs (i.e., with max aggregation) and consider edge marked by $\underline{0} + 4$. It is easy to see that this edge can be redirected to a leaf with value zero without changing $\text{MAP}_B(I)$ for any I. This is true because if we can reach the leaf with value 6 using some valuation then we can also reach the leaf with value 14 using another valuation because x_2 is not constrained. Therefore, the max aggregation will always ignore valuations reaching value 6. It is also easy to see that the edge marked $\underline{8} + 4$ can be redirected to value 14 without changing $\text{MAP}_B(I)$. Simplification[2] of diagrams by removing unnecessary portions is crucial for efficiency of GFODD implementations and a significant amount of previous work was devoted to mechanizing this process. Note that it is most natural to keep the aggregation portion fixed and simply manipulate the diagram portion. In this paper we abstract this process as testing for GFODD Equivalence, that is, testing whether the diagram is equivalent to a second simpler one. Motivated by the focus in the implementations on algorithms that remove one edge at a time, as illustrated in the example, we also formalize this special case.

2.4. Complexity Theory Notation

Recall that the polynomial hierarchy is defined from P, NP, and co-NP using an inductive construction with reference to computation with oracles [14, 32, 26]. In particular we have that Σ_1^P =NP, and Π_1^P =co-NP. An algorithm is in the class A^B if it uses computation in A with a polynomial number of calls to an oracle for a problem in class B. Then we have $\Sigma_{k+1}^P = \text{NP}^{\Sigma_k^P}$, and $\Pi_{k+1}^P = \text{co-NP}^{\Sigma_k^P}$. A problem is in Σ_k^P iff its complement is in Π_k^P and thus (since the oracle always answers deterministically and correctly) either of these can serve as the oracle in the definition.

2.5. Computational Problems

Before defining the computational problems we must define the representation of inputs. We assume that GFODDs are given using a list of aggregation

[2]Simplification was called *reduction* by [33]; to avoid confusion with the standard complexity theory meaning of the term reduction we use the term simplification instead.

operators and associated variables and a labelled graph representation of the diagram. This is clearly polynomially related to the number of variables and number of nodes in the GFODD. Some of our problems require interpretations as input. Here we assume a finite domain so as to avoid issues of representing the interpretation. Thus an interpretation is given as a list of objects serving as domain elements, a list specifying the mapping of constants to objects, and the extension of each predicate on these objects. Given that the signature is fixed and the arity of each predicate is constant, this implies that the size of I is polynomially related to the number of objects in I. As illustrated in the example of Figure 1, a graph $G = (V, E)$ can be seen as an interpretation with domain V and with one predicate formed by the edge relation.

We can now define the computational problems of interest. We separate the definitions for FODDs and GFODDs because for GFODDs the unrestricted problems are undecidable and they require further refinement. The simplest problem requires us to evaluate a diagram on a given interpretation.

Definition 2 (FODD Evaluation). *Given diagram B, interpretation I with finite domain, and value $V \geq 0$: return Yes iff $MAP_B(I) \geq V$. In the special case when the leaves are restricted to $\{0, 1\}$ and $V = 1$ this can be seen as a returning Yes iff $MAP_B(I)$ is true.*

To calculate $\text{MAP}_B(I)$ we can "run" a procedure for FODD Evaluation multiple times, once for each leaf value as V, and return the highest achievable result. Thus, if FODD Evaluation is in complexity class A, we can calculate the function value in P^A. This fact is used several times in our constructions.

Since diagrams generalize FOL it is natural to investigate satisfiability:

Definition 3 (FODD Satisfiability). *Given diagram B with leaves in $\{0, 1\}$: return Yes iff there is some I such that $MAP_B(I)$ is true.*

When B has more than two values in its leaves the satisfiability problem becomes:

Definition 4 (FODD Value). *Given diagram B and value $V \geq 0$: return Yes iff there is some I such that $MAP_B(I) = V$.*

Notice that FODD Value requires that V is achievable but no value larger than V is achievable on the same I and, as the proofs below show, the extra requirement makes the problem harder. On the other hand, if we replace equality with $\geq V$ in FODD Value, the problem is equivalent to FODD Satisfiability because we can simply replace leaf values in the diagram with 0,1 according to whether they are $\geq V$.

Finally, as motivated above, we investigate the simplification problem and its special case with single edge removal.

Definition 5 (FODD Equivalence). *Given diagrams B_1 and B_2: return Yes iff $MAP_{B_1}(I) = MAP_{B_2}(I)$ for all I.*

Definition 6 (FODD Edge Removal). *Given diagrams B_1 and B_2, where B_2 can be obtained from B_1 by redirecting one edge to a zero valued leaf: return Yes iff $MAP_{B_1}(I) = MAP_{B_2}(I)$ for all I.*

Given the discussion above, GFODDs with binary leaves can be seen to capture the function free fragment of first order logic with equality. It is well known that satisfiability and therefore also equivalence of expressions in this fragment of first order logic is not decidable. In fact, the problem is undecidable even for very restricted forms of quantifier alternation (see survey and discussion in [8]). For example, the problem is undecidable for quantifier prefix $\forall^2\exists^*$ with a single binary predicate and equality. The problem is also undecidable if we restrict attention to satisfiability under finite structures. Therefore, without further restrictions, we cannot expect much by way of classification of the complexity of the problems stated above for GFODDs.

We therefore restrict the problems so that the size of interpretations is given as part of the input. The restriction ensures that the problems are decidable and reveals the structure promised above. There are two motivations for using such a restriction. The first is that in some applications we might know in advance that the number of relevant objects is bounded by some large constant. For example, the main application of GFODDs to date has been for solving decision theoretic planning problems; in this context the number of objects in an instance (e.g., the number of trucks or packages in a logistics transportation problem) might be bounded by some known quantity. The second is that our results show that even under such strong conditions the computational problems are hard, providing some justification for the heuristic approaches used in FODD and GFODD implementations [15, 17, 18].

Definition 7 (GFODD Model Evaluation). *Given diagram B, interpretation I with finite domain, and value $V \geq 0$: return Yes iff $MAP_B(I) \geq V$. Note that when the leaves are restricted to $\{0, 1\}$ and $V = 1$ this can be seen as a returning Yes iff $MAP_B(I)$ is true.*

Definition 8 (GFODD Satisfiability). *Given diagram B with leaves in $\{0, 1\}$ and integer N in unary: return Yes iff there is some I, with at most N objects, such that $MAP_B(I)$ is true.*

Definition 9 (GFODD Value). *Given diagram B, integer N in unary and value $V \geq 0$: return Yes iff there is some I, with at most N objects, such that $MAP_B(I) = V$.*

Definition 10 (GFODD Equivalence). *Given diagrams B_1 and B_2 (with the same aggregation functions) and integer N in unary: return Yes iff for all I with at most N objects, $MAP_{B_1}(I) = MAP_{B_2}(I)$.*

Definition 11 (GFODD Edge Removal). *Given diagrams B_1 and B_2 (with the same aggregation functions), where B_2 can be obtained from B_1 by redirecting one edge to a zero valued leaf, and given integer N in unary: return Yes iff for all I with at most N objects, $MAP_{B_1}(I) = MAP_{B_2}(I)$.*

Since we are assuming a fixed arity k, the assumption that N is in unary is convenient because it implies that the size of an intended interpretation I is polynomial in N. Therefore, an algorithm for these problems can explicitly represent an interpretation of the required size and test it. Our hardness results use N which is at most linear in the size of the corresponding diagram B.

3. The Complexity of Reasoning with FODD

In this section we develop the complexity results for the special case of FODDs. Evaluation of FODDs is essentially the same as evaluation of conjunctive queries in databases and can be analyzed similarly. We include the argument here for completeness.

Theorem 12. *FODD Evaluation is NP-complete.*

Proof. Membership in NP is shown by the algorithm that guesses a valuation ζ, calculates $\text{MAP}_B(I, \zeta)$ and returns Yes iff the leaf reached has value $\geq V$. Yes is returned iff some valuation yields a value $\geq V$ as needed.

For hardness we reduce the directed Hamiltonian path to this problem. As illustrated in Figure 1, given the number of nodes in a graph we can represent a generic Hamiltonian path verifier as a FODD B. To do this we simply produce a left going path $E(x_1, x_2), E(x_2, x_3), \ldots, E(x_{n-1}, x_n)$ which verifies existence of the edges, followed by equality tests to verify that all nodes are distinct. All "failure exits" on this path go to 0 and the success exit of the last test yields 1. Call this diagram B. This diagram is ordered with $E \prec$ " = " and lexicographic ordering over arguments. Now, given any input G for Hamiltonian path, we represent it as an interpretation I and produce $(B, I, 1)$ as the input for FODD Evaluation. Clearly, G has a Hamiltonian path iff $\text{MAP}_B(I) = 1$. □

The other results for FODDs rely on the existence of small models:

Lemma 13. *For any FODD B with k variables and constants, if $MAP_B(I) = V$ for some I then there is an interpretation I' with at most k objects such that $MAP_B(I') = V$.*

Proof. Let I be as in the statement. Then there is a valuation ζ such that ζ reaches a leaf valued V in B. Let I' be an interpretation including the objects that are used in the path traversed by ζ where the truth value of any predicate over arguments from these objects agrees with I. We have that I' has at most k objects, ζ is a suitable valuation for I' and $\text{MAP}_B(I', \zeta) = V$. In addition, no other valuation ζ' leads to a value larger than V because, if it did, the same value would be achievable in I. Hence, $\text{MAP}_B(I') = \text{MAP}_B(I', \zeta) = V$. □

Theorem 14. *FODD Satisfiability is NP-Complete.*

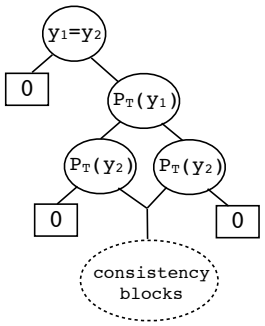

Figure 3: The gadget verifying that $P_T()$ simulates Boolean values.

Proof. For membership we can guess an interpretation I, which by the previous lemma can be small, and guess a valuation ζ for that interpretation. We return Yes if and only if $\text{MAP}_B(I, \zeta) = 1$.

We show hardness with a reduction from 3SAT. Let f be an arbitrary 3CNF formula. We create a new FODD variable for each literal occurrence in the CNF so that $v_{(i,j)}$ corresponds to the jth literal in the ith clause.

Our FODD has three portions connected in a chain. The first portion checks that the predicate $P_T()$ in the interpretation can be used to simulate Boolean assignments. To achieve this, we first ensure that the interpretation has at least two different objects, referred to by variables y_1 and y_2. We then use a small block that ensures that the truth value of $P_T(y_1)$ is not equal to $P_T(y_2)$. As a result $P_T(y_1)$ and $P_T(y_2)$ correspond to true and false logical values. This is shown in Figure 3.

The second portion ensures that if $v_{(i,j)}$ and $v_{(i',j')}$ correspond to the same Boolean variable then they map to the same object. For every variable x_i we create a shadow FODD variable w_i and equate it to all the $v_{(i',j')}$ that correspond to x_i. We call this sequence of equalities a consistency block. For example, consider the CNF

$$(x_1 \vee \overline{x_2} \vee x_4) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee \overline{x_4})$$

where the corresponding FODD variables are

$$v_{(1,1)}, v_{(1,2)}, v_{(1,3)}, \quad v_{(2,1)}, v_{(2,2)}, v_{(2,3)}, \quad v_{(3,1)}, v_{(3,2)}, v_{(3,3)}.$$

The first block, corresponding to x_1, ensures that $w_1, v_{(1,1)}, v_{(2,1)}, v_{(3,1)}$ are all assigned the same value. In addition to testing that the values are equivalent the block tests that each variable gets bound to the same object as y_1 or y_2. The only possible way to not get a 0 in these blocks is to ensure that each variable in the block has the same value and that it is equal to either y_1 or y_2. Figure 4 shows the consistency blocks for our example.

The third portion tracks the structure of f to guarantee the same truth value in the FODD. To follow the structure of f, we build a block for each clause and

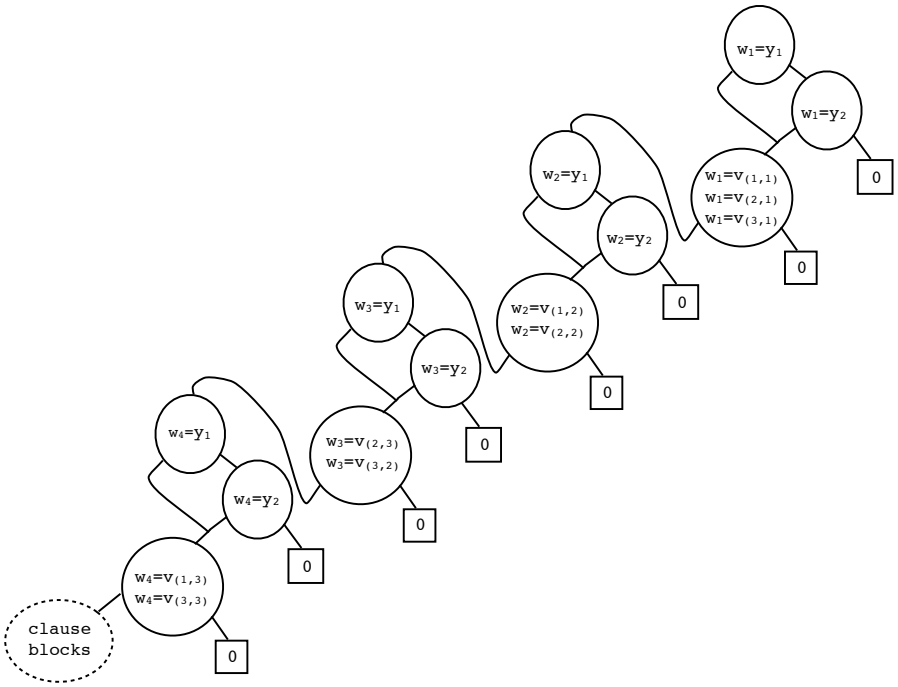

Figure 4: The gadget verifying variable consistency in the max FODD for the formula $(x_1 \vee \overline{x_2} \vee x_4) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee \overline{x_4})$. Nodes that have multiple equalities are a shorthand for a sequence of equality nodes where non-equality on any of the tests leads to the "false exit", which is a 0 leaf in this diagram.

chain these blocks together. Each block has 3 nodes corresponding to the 3 literals in the clause. In particular, if the jth literal in the ith clause is positive the true edge (literal satisfied; call this success) continues to the next clause, and the false edge (literal failed) continues to the next literal. For a negative literal the true and false directions are swapped. The fail exit of the 3rd literal is attached to 0. Clause blocks have one entry and one exit and they are chained together. The success exit of the last clause is connected to the leaf 1. The only way to reach a value of 1 is if every clause block was satisfied by the valuations to $v_{(i,j)}$. Figure 5 illustrates the clause blocks for our example.

Each of the portions, including the clause blocks, has one entry and one exit and we chain them together to get the diagram B. For a valuation to be mapped to 1 it must succeed in all three portions. We claim that f is satisfied if and only if there is some interpretation I such that $MAP_B(I) = 1$.

Consider first the case where f is satisfiable. We introduce the interpretation I that has two objects, a and b, where $P_T(a) =$ true, and $P_T(b) =$ false. Let v be a satisfying assignment for f and let $\zeta(v)$ be a valuation for B on I where $y_1 = a, y_2 = b$ and if v maps x_i to 1 then w_i and its block are mapped to a and otherwise the block is mapped to b. Here, $\zeta(v)$ succeeds in all blocks, implying

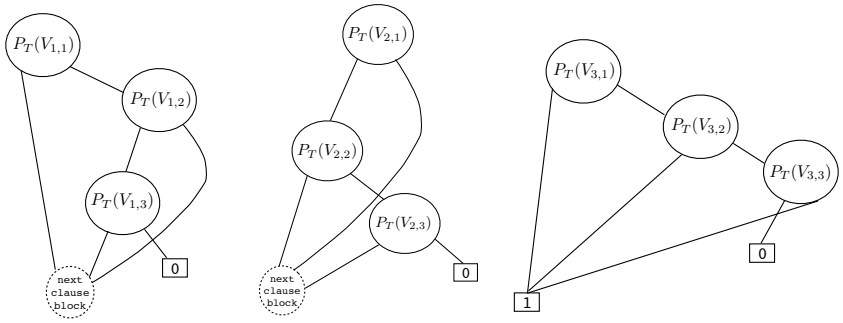

Figure 5: The clause blocks for $(x_1 \vee \overline{x_2} \vee x_4) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee \overline{x_4})$.

that $MAP_B(I, \zeta(v)) = 1$ and therefore $MAP_B(I) = 1$.

Consider next the case where $MAP_B(I) = 1$ for some I and let ζ be such that $MAP_B(I, \zeta) = 1$. Then we claim that ζ identifies a satisfying assignment. First, since ζ succeeds in the first block we identify two objects that correspond via $P_T()$ to truth values. Without loss of generality assume that $P_T(y_1)$ is true. Then success in the second portion implies that we can identify an assignment to the Boolean variables, if the ith block is assigned to y_1 we let $x_i = 1$ and otherwise $x_i = 0$. Finally, success in the third portion implies that the clauses in f are satisfied by the assignment to the x_i's. This completes the correctness proof.

Finally we address node ordering in the diagram. The only violation of ordering is the use of $P_T()$ in the first block. Otherwise, we have all equalities above $P_T()$, variable ordering $y_i \prec w_j \prec V_{a,b}$, and lexicographic ordering within a group. Now because our diagram forms one chain of blocks leading to a single sink leaf with value 1 we can move the three $P_T()$ nodes to the bottom of the diagram in Figure 4. This does not change the map value for any valuation and thus does not affect correctness. We therefore conclude that B is consistently sorted and f is satisfiable iff $\text{MAP}_B(I) = 1$ for some I. □

This proof illustrates the differences in arguments needed for FODDs and GFODDs vs. First Order Logic. For the latter, we can use the sentence $\exists v, (p_{x_1}(v) \vee \overline{p_{x_2}(v)} \vee p_{x_4}(v)) \wedge (\overline{p_{x_1}(v)} \ldots)$ to show the hardness result, and the proof is therefore considerably simpler. However, this cannot be easily represented as a FODD because the literals appearing in the clauses will violate predicate order and, if we try to reorder the nodes from a naive FODD encoding, the result might be exponentially larger. An alternative formulation can use $\exists x_1 \ldots (p(x_1) \vee \overline{p(x_2)} \vee p(x_4)) \wedge (\overline{p(x_1)} \ldots$ to avoid the problem with predicate order. However, similar ordering issues now arise for the arguments. Our reduction introduces additional variables as well as the variable consistency gadget to get around these issues. The same structure of reduction from 3SAT instances and their QBF generalizations will be used in the results for GFODD.

Theorem 15. *FODD Equivalence and FODD Edge Removal are* Π_2^P*-complete.*

Proof. Since Edge Removal is a special case of Equivalence it suffices to show membership for Equivalence and hardness for Edge Removal. The hardness result is given in two stages; we first present a reduction which does not respect the constraint on node ordering and Edge Removal structure, and then show how to fix the construction to respect these restrictions.

Membership in Π_2^P*.* First observe that, by Lemma 13, if the diagrams are not equivalent then there is a small interpretation that serves as a witness for the difference. Using this fact, we can show that non-equivalence is in Σ_2^P. Given B_1, B_2 we guess an interpretation I of the appropriate size, and then appeal to an oracle for FODD Evaluation to calculate $\text{MAP}_{B_1}(I)$ and $\text{MAP}_{B_2}(I)$. Using these values we return Yes or No accordingly. To calculate the map values, let B be one of these diagrams, and let the leaf values of the diagram be $v_1, v_2, \ldots, v_k$. We make k calls to FODD Evaluation with (B, I, v_i) as input. $\text{MAP}_B(I)$ is the largest value on which the oracle returns Yes. If a witness I for non-equivalence exists then this process can discover it and say No, and otherwise it will always say Yes. Therefore non-equivalence is in Σ_2^P, and equivalence is in Π_2^P.

Reduction basics. To show hardness, consider the problem of deciding arrowing from the Ramsey theory of graphs [31]. Given two graphs G_1, G_2 we say that G_1 includes an *embedding* of G_2 if there is a 1-1 mapping g from nodes of G_2 to nodes of G_1, such that for every edge (v_1, v_2) of G_2, the edge $(g(v_1), g(v_2))$ is in G_1. We say that G_1 includes an *isomorphic embedding* of G_2 if, in addition, g satisfies that for every edge (v_1, v_2) not in G_2, the edge $(g(v_1), g(v_2))$ is not in G_1.

We say that F arrows (G, H), denoted $F \rightarrow (G, H)$, if for every 2-color edge-coloring of F into colors red and blue, the red subgraph of F includes an embedding of G or the blue subgraph of F includes an embedding of H.

The arrowing problem is as follows: Given 3 graphs F, G, H as input, return Yes iff F arrows (G, H). This problem was shown to be Π_2^P-complete by [31]. We reduce this problem to FODD equivalence. The signature includes equality and two arity-2 predicates E_F and E_C, where E_F captures the edge relation of the main graph F and E_C captures a coloring of all possible edges such that when $E_C(x_i, x_j)$ is true the edge is colored red and when it is false the edge is colored blue.

The main construction. To transform arrowing into an instance of FODD equivalence we build two FODDs with binary leaves. The first FODD is satisfied iff I includes an isomorphic embedding of F in its edge relation E_F. The second FODD is satisfied iff the same condition holds *and* the coloring defined by E_C has a red embedding of G or a blue embedding of H. Note that, due to the 1-1 requirement, I must have at least as many objects as there are nodes in F. We illustrate the construction using the example input in Figure 6. Here the input graphs F, G, H are a positive instance of arrowing.

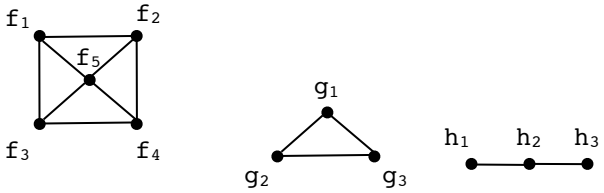

Figure 6: Three input graphs to the Arrowing problem F, G, H. This is a positive instance as (one can verify that) every two coloring of F has either a red triangle (G) or two blue edges connected by a single vertex (H).

To build a FODD which verifies that I has an isomorphic embedding of F, we map each node to a variable in the FODD and test that each node has its correct neighbors. We first build a "node mapping" gadget that makes sure that each variable in the FODD is mapped to a different object in the interpretation. This is done by following a path of $\binom{V}{2}$ inequalities, where off-path edges go to 0 and the final exit continues to the next portion. This gadget, for our example graph F with 5 nodes, is shown in Figure 7. To test isomorphism to F we test the neighbors of each node in sequence to verify that edges exist iff they are in F. The FODD fragment in Figure 8 shows how this can be tested for vertex f_1 in the example. If the edge is present in the graph we continue left (using the `true` branch) to the next neighbor and if the edge is not in F we continue to the right child (the `false` branch). Edges off this path are directed to the zero leaf. The endpoint of the path will connect to the next portion of the FODD. This construction can be done for each node and the fragments can be connected together to yield the F verifier. This is illustrated in Figure 9. Finally, the diagram B_1 is built by connecting the F verifier at the bottom of node mapping gadget, and replacing the bottom node of the F verifier with a leaf valued 1. We refer to this diagram as the "complete F verifier" below. This construction can be done in polynomial time for any graph F. It should be clear form the construction that $\mathrm{MAP}_{B_1}(I) = 1$ iff I includes an isomorphic embedding of F in its edge relation E_F. In addition, the verifier diagram is ordered where we have " $=$ " $\prec E_F$, and where variables are ordered lexicographically.

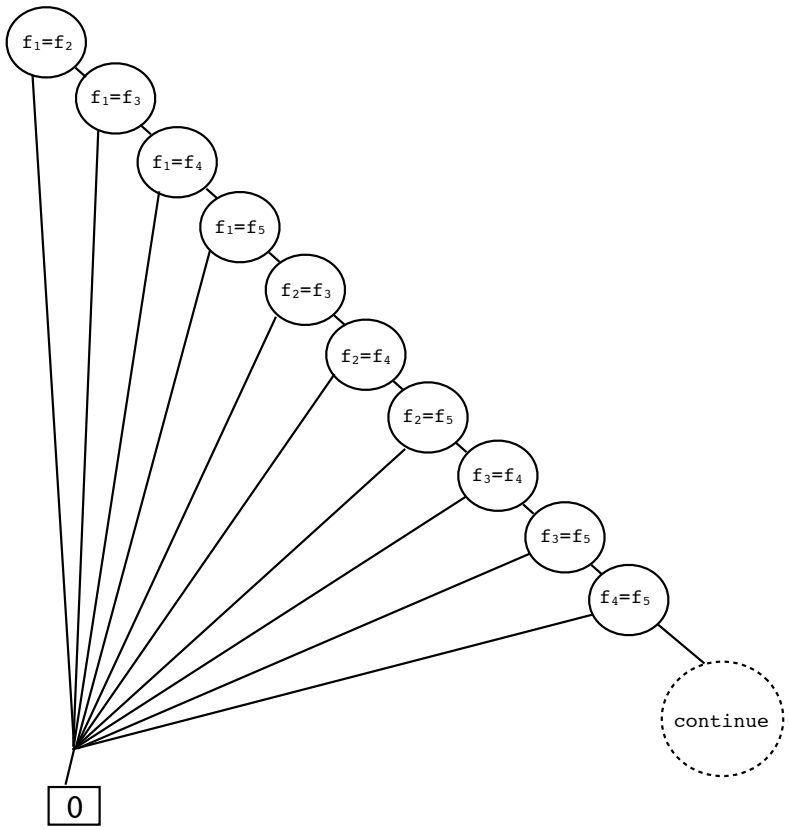

Figure 7: A FODD verifying that variables f_i are mapped to distinct objects in I.

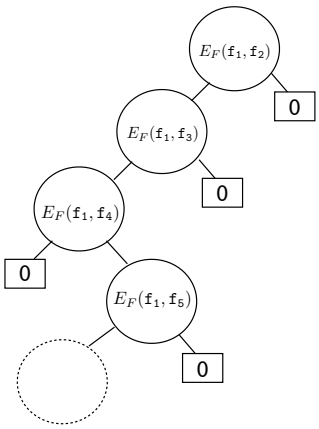

Figure 8: A fragment of a FODD verifying the graph F. Here every neighbor of f_1 is tested. Since f_1 is connected to f_2, f_3 we expect the corresponding atoms to be true and thus continue on the left branch; since it is not connected to vertex f_4, the gadget continues on the false side; f_5 is connected and we continue to the left again.

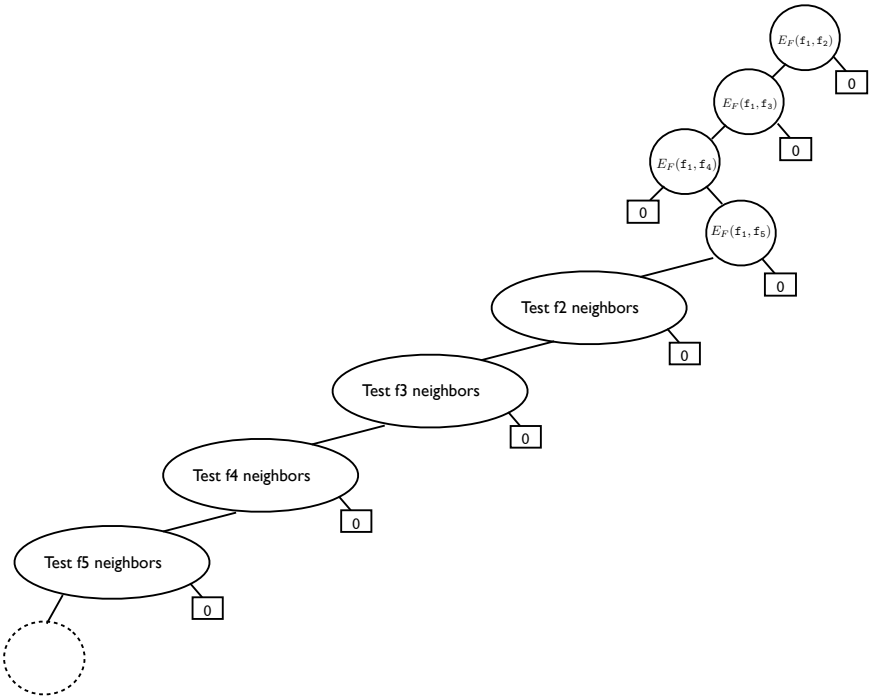

Figure 9: The FODD verifying the structure of the graph F.

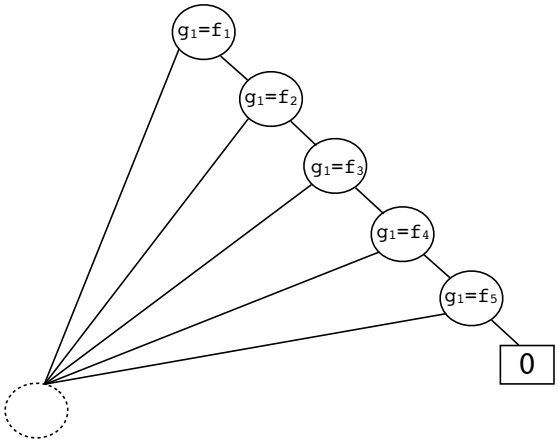

Figure 10: A FODD verifying the mapping of vertex g_1 to some vertex in F.

The second diagram B_2 includes the complete F verifier and additional FODD fragments that are described next to capture the conditions on G and H respectively. In order to verify the embedding of colored subgraph G we first define a node mapping capturing the mapping of G nodes into F nodes, and then verify that the required edges exist and that they have the correct color. The FODD fragment in Figure 10 shows how we can select a node mapping for vertex g_1. This fragment returns 0 unless g_1 is mapped to one of the nodes in F that are identified in the B_1 portion. As depicted in Figure 11, this can be repeated for all the nodes in G, verifying that each node in G is mapped to a node in F. Next we need to verify that the mapping is one to one. This can be done by using a path of inequalities between the variables referring to nodes of G. This FODD fragment is given in Figure 12. For correctness, we need to chain the two tests together, but this will violate node ordering. We therefore interleave the tests putting the uniqueness equality tests for a variable exactly after the equalities selecting its value. This change is possible because each such block has exactly one exit point. The resulting diagram, for our running example, is shown in Figure 13.

To complete the embedding test, we need to check that the edges are preserved and that they have the correct color. We do this by first checking that the corresponding edges in G are in F. We can do this using a left going path testing each edge in turn, where we test both $E_F(g_i, g_j)$ and $E_F(g_j, g_i)$ to account for the fact that the graph is undirected.[3] This is illustrated on the left

[3]The test of both directions of the edge is not necessary, because a different portion of the diagram already verifies that the embedding of F is undirected, but we include it here to simplify the argument.

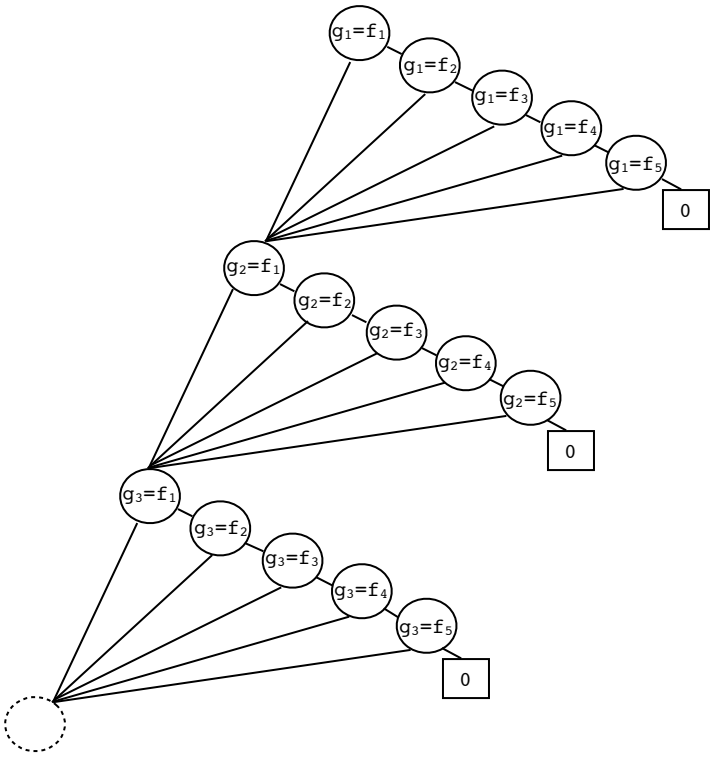

Figure 11: The complete FODD verifying the vertices of G are mapped to vertices in F.

hand side of Figure 14. Note that, because we are testing for an embedding (i.e., not for an isomorphic embedding) we test only for the edges in G and do not need to verify nonexistence of the edges not in G (it just happens here that G is a clique so this is not visible in the example). The same FODD structure is repeated with predicate E_C replacing E_F to verify that the edges of G are colored red, as shown on the right of Figure 14.

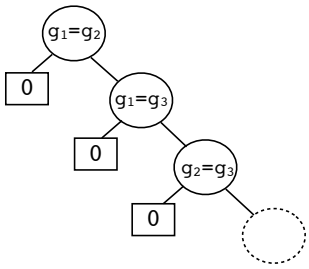

Figure 12: A FODD verifying the mapping is one to one.

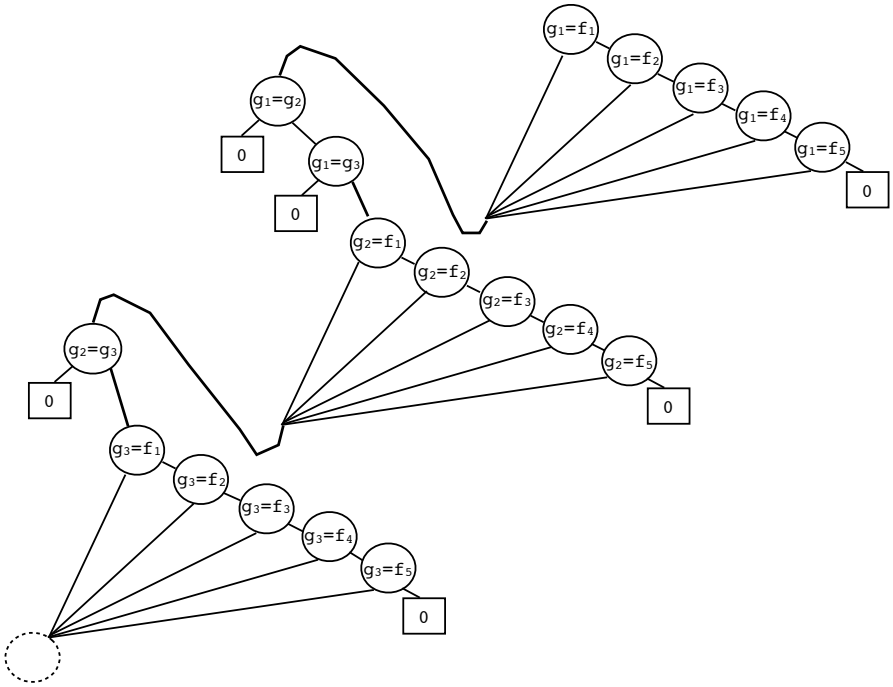

Figure 13: Node mapping construction for G reordered to comply with sorted order.

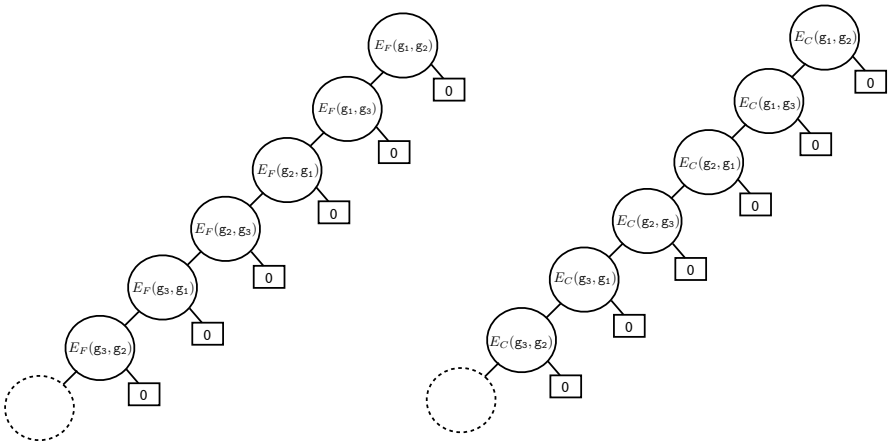

Figure 14: A FODD verifying that the edges in G are in F.

A similar construction with node mapping, edge verifier, and color verifier can be used for H. The node mapping construction is identical. Figure 15 shows the edge and color verifiers. The only difference in construction is that the color verifier tests that the edge is not in E_C to capture the color blue and therefore has a mirror structure to the one verifying the E_F edges. Note that in this case H is not a complete graph and we are indeed only testing for the edges in H. This construction can be done in polynomial time for any G and H.

Finally we connect the three portions together to obtain B_2 as follows. The final output of the complete F verifier is connected to the root of the G verifier. The final output of the G verifier is connected to 1. The zero leaf of the G verifier is removed and instead connected to the root of the H verifier. The final output of the H verifier is connected to 1. Therefore, there are exactly two edges leading to the 1 leaf in this diagram, corresponding to the positive outputs of the G and H verifiers. Figure 16 shows an overview of the two FODDs, B_1 and B_2, generated by the reduction.

The diagrams B_1 and B_2 are not consistent with any sorting order over node labels, and thus we need to modify them to get a consistent ordering. We show below how this can be done with only a linear growth in the size of the diagrams and without changing the semantics of B_1 and B_2. Before presenting this transformation we show that $F \to (G, H)$ iff B_1 and B_2 are equivalent.

Correctness of the construction. Consider the case when $F \to (G, H)$, that is, for every 2-color edge-coloring of F there is a red G or a blue H. We show that the two FODDs are equivalent by way of contradiction. Assume that B_1 and B_2 are not equivalent and let I be any witness to this fact. Now, $\text{MAP}_{B_1}(I) = 0$ implies $\text{MAP}_{B_2}(I) = 0$ because the only paths to 1 in B_2 go through a copy of B_1. Therefore, for the assumed witness I, it must be the case that $\text{MAP}_{B_1}(I) = 1$ and $\text{MAP}_{B_2}(I) = 0$.

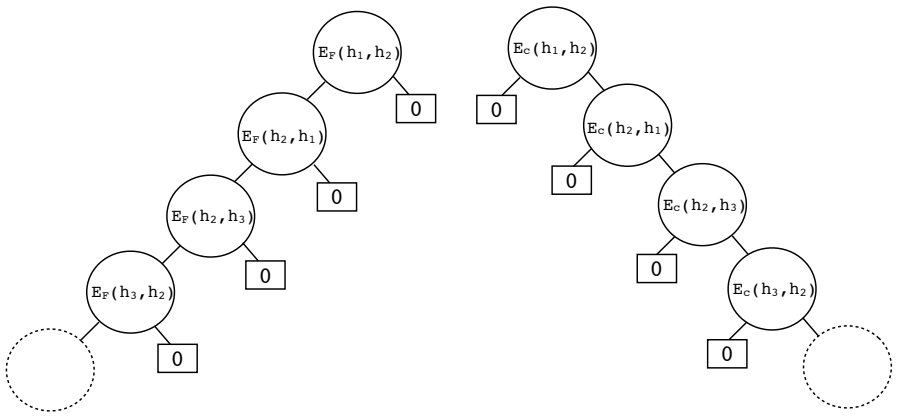

Figure 15: Left: a FODD verifying that the edges in H are in F. Right: a FODD verifying that the color of H is blue (i.e., its edges are not in E_C).

By construction, $\text{MAP}_{B_1}(I) = 1$ implies that I has an isomorphic embedding of F. Because $F \rightarrow (G, H)$, any coloring of that embedding, including the coloring captured by E_C in I, has a red G or a blue H. Assume that the embedding in I has a red G. Then we can construct the appropriate node mapping in a valuation ζ to show that $\text{MAP}_{B_2}(I, \zeta) = 1$, contradicting the assumption. The same argument handles the case when the embedding has a blue H.

Consider the case when F does not arrow (G, H). Then there is a valid 2-color edge-coloring of F which does not have a red G and does not have a blue H. Construct the corresponding interpretation I that represents F and this edge-coloring. We claim that $\text{MAP}_{B_1}(I) = 1$ and $\text{MAP}_{B_2}(I) = 0$. The fact $\text{MAP}_{B_1}(I) = 1$ follows by mapping the nodes in F to the variables that represent them. Now if $\text{MAP}_{B_2}(I) = 1$ then $\text{MAP}_{B_2}(I, \zeta) = 1$ for some ζ and we can trace the path that ζ traverses in B_2. This path together with ζ can be used to identify either a red G or a blue H in I and therefore in the corresponding coloring of F. This contradicts the assumption that the coloring is a witness for non-arrowing.

Fixing the construction to handle ordering and edge removal special case. We next consider the node ordering in B_1 and B_2. The diagram B_1 is sorted, where predicate order puts equalities above E_F and arguments are lexicographically ordered. For B_2 we consider the sub-block structure of the construction. Expanding each of the sub-blocks of F, G, H in Figure 16 we observe that B_2 has the structure shown in Figure 17. We further observe that each block is internally sorted, but blocks of equalities E_F and E_C are interleaved. By analyzing this structure we see that the blocks can be reordered at the cost of duplicating some portions yielding the structure in Figure 18. It is easy to see that B_2 is satisfied in I if and only if the reordered diagram is satisfied in I. The diagrams yield the same value for any valuation ζ which does not exit to 0 due to bad

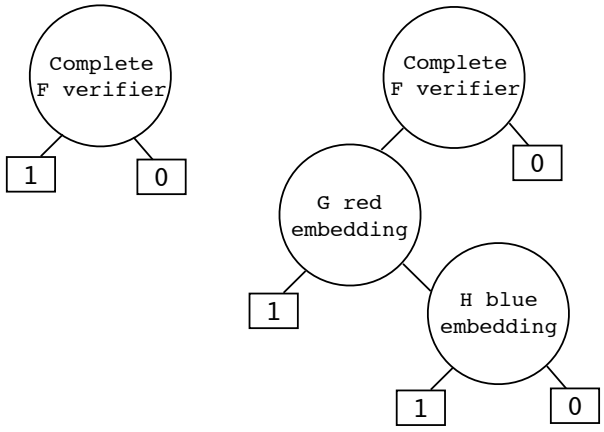

Figure 16: High level structure of the two FODDs created from the graphs F, G, H.

node mapping for G or H. Thus the original version might yield 1 (e.g., through G path) when the reordered diagram yields 0 on such a valuation (e.g., via the H equalities). But in such a case there is another valuation that is identical to ζ except that it modifies the bad node mapping (the H equalities) and that yields 1 for the new diagram. The final diagram is consistent with predicate ordering " $=$ " $\prec E_F \prec E_C$ and variable ordering where $f_i \prec g_j \prec h_k$ for all i, j, k.

Finally, we further change B_1 by adding the equality blocks of G and H to the construction, so that the modified B_1 is as shown in Figure 19. Using the same argument as in B_2 one can see that this does not change the semantics of B_1. Moreover, with this change B_1 can be obtained from B_2 by one edge removal (of the edge below the F verifier in B_2) so that the reduction holds for this more restricted case. □

As mentioned above, FODD Value is defined similarly to FODD Satisfiability but requires more stringent conditions. The next result shows that this difference is important and FODD Value is one level higher in the hierarchy.

Theorem 16. *FODD Value is Σ_2^P-complete.*

Proof. The algorithm showing membership is as follows. We first observe that by Lemma 13 we can restrict our attention to small interpretations. Given input B and V we guess an interpretation I of the appropriate size. We then make two calls to an oracle for FODD Evaluation. Let V' be either the least leaf value greater than V or any value greater than the max leaf if V is the maximum. We query the oracle for FODD Evaluation on (B, I, V) and (B, I, V') and return Yes iff the oracle returns Yes on the first and No on the second. The algorithm returns Yes iff there is an interpretation I with value V.

For hardness we present a reduction from non-Equivalence of FODDs with binary leaves, which was shown to be Σ_2^P-hard in Theorem 15. We are given $B_1 =$

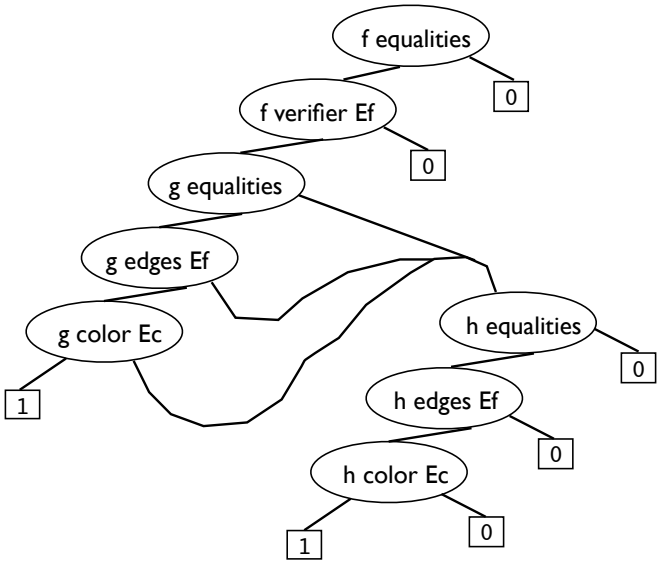

Figure 17: Expanded view of the structure of B_2.

$\max_{x_1} B_1(x_1)$ and $B_2 = \max_{x_2} B_2(x_2)$ as input for FODD non-Equivalence where B_1 and B_2 are standardized apart so that x_1, x_2 stand for disjoint sets of variables. We construct the diagram $B = \max_{x_1} \max_{x_2} B(x_1, x_2)$ where $B(x_1, x_2) = B_1(x_1) + B_2(x_2)$ can be calculated directly on the graph representation of B_1 and B_2 using the Apply procedure of [33] (see Figure 2). Because x_1 and x_2 are disjoint, the diagram B has the following behavior for any interpretation I: if $\text{MAP}_{B_1}(I) = 1$ and $\text{MAP}_{B_2}(I) = 1$ then $\text{MAP}_B(I) = 2$; otherwise if exactly one of them evaluates to 1 then $\text{MAP}_B(I) = 1$; and otherwise $\text{MAP}_B(I) = 0$. We produce $(B, V = 1)$ as input for FODD Value.

Now, if B_1 and B_2 are not equivalent then there is an interpretation such that their maps are different, and without loss of generality we may assume $\text{MAP}_{B_1}(I) = 1$ and $\text{MAP}_{B_2}(I) = 0$. As argued above in this case $\text{MAP}_B(I) = 1$ as needed. For the other direction let I be such that $\text{MAP}_B(I) = 1$. Then, again using the argument above, we have $\text{MAP}_{B_1}(I) = 1$ and $\text{MAP}_{B_2}(I) = 0$ or vice versa and the diagrams are not equivalent. □

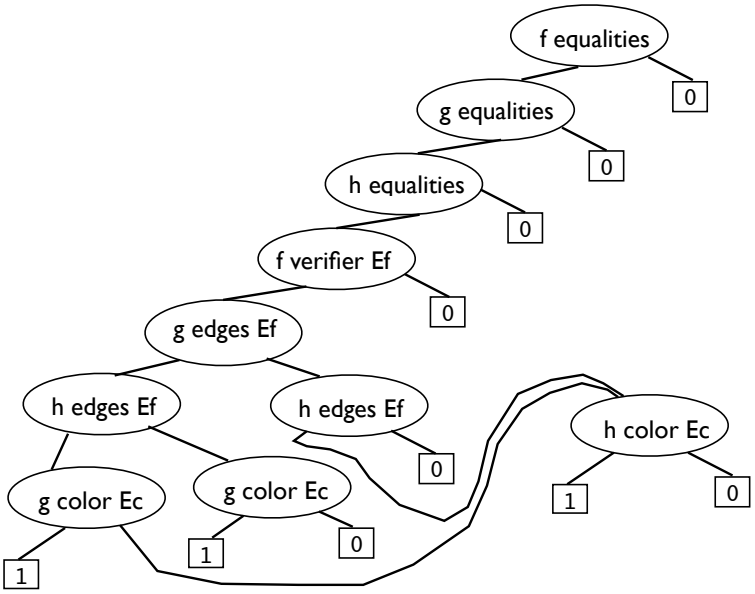

Figure 18: Expanded view of the reordered structure of B_2.

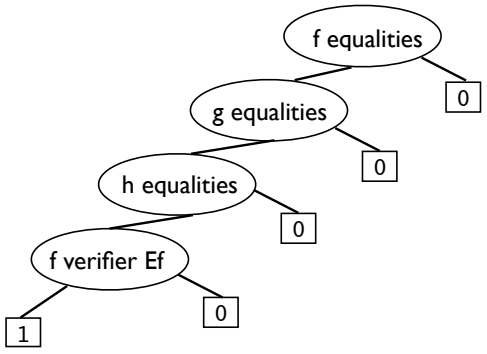

Figure 19: Expanded view of the modified version of B_1.

4. The Complexity of Reasoning with GFODD

In this section we analyze the computational problems for GFODD. We start with some observations on a notion of "complement" for GFODDs. Let B be a GFODD associated with the ordered list of variables $(w_{i_1}, \ldots, w_{i_m})$, and aggregation list $(A_1, \ldots, A_m)$ where each A_i is min or max. Let $B' = \text{complement}(B)$ (with respect to maximum value M) be the diagram corresponding to B where we change leaf values and aggregation operators as follows: Let M be any value greater or equal to the max leaf value in B. Any leaf value v is replaced with $M - v$. Each aggregation operator A_i is replaced with A'_i where where if A_i is min then A'_i is max and vice versa.

Theorem 17. *Let B be a GFODD with min and max aggregation and maximum leaf value $\leq M$, and let $B' = \text{complement}(B)$. For any interpretation I, $MAP_B(I) = M - MAP_{B'}(I)$.*

Proof. By the construction of B', for any valuation ζ, we have that $\text{MAP}_B(I, \zeta) = M - \text{MAP}_{B'}(I, \zeta)$. Considering the aggregation process, note that $A_i \text{ MAP}_B(\ldots w_i) = A_i \ [M - \text{MAP}_{B'}(\ldots w_i)] = M - A'_i \text{ MAP}_{B'}(\ldots w_i)$. Now using this fact, we can argue by induction backward from the innermost (rightmost) aggregation that for any prefix of variables $P = w_{i_1}, \ldots, w_{i_p}$, valuation ζ_p for these variables, and remaining variables $R = w_{i_{p+1}}, \ldots, w_{i_m}$, we have $A_R \text{ MAP}_B(I, (P = \zeta_p; R)) = M - A'_R \text{ MAP}_{B'}(I, (P = \zeta_p; R))$. When the prefix is empty we get the statement of the theorem. □

Notice that for diagrams with binary leaves this yields $\text{MAP}_B(I) = 1 - \text{MAP}_{B'}(I)$, that is, negation. As immediate applications we get:

Corollary 18. *The complexity of GFODD Equivalence for* min-k*-alternating GFODD is the same as the complexity of GFODD Equivalence for* max-k*-alternating GFODD.*

Proof. By Theorem 17, two min diagrams B_1, B_2 are equivalent if and only if their complements B'_1, B'_2 are equivalent where we can use the maximum among the leaf values of the two diagrams as M. □

Corollary 19. *The equivalence problem for* min*-GFODD is Π_2^P-complete.*

We can now turn to analysis of the computational problems. Evaluation is similar to the FODD case but the hardness proof is more involved due to the interaction between quantifier order and node ordering in the diagram.

Theorem 20. *GFODD Evaluation for* max-k*-alternating GFODDs is Σ_k^P- complete. GFODD Evaluation for* min-k*-alternating GFODDs is Π_k^P-complete.*

Proof. We prove membership by induction on k. Since the inductive step includes diagrams that do not satisfy the sorting order we show that the claim holds in this more general case. Consider the input (B, I, V). For the base case, $k = 1$, we guess a valuation ζ, calculate $v = \text{MAP}_B(I, \zeta)$, and return Yes iff

$v \geq V$. In the max case, if the true value is at least V then we say Yes for some ζ, and if the true value is less than V then $\text{MAP}_B(I, \zeta) < v$ for all ζ and therefore we always say No. Thus the problem is in NP. In the min case, if the true value is at least V then all ζ yield Yes, and if the true value is less than V then some ζ yields No. Thus the problem is in co-NP.

For the inductive step assume that the claim holds for $k-1$ and consider the input (B, I, V) with an interpretation I, value bound V and a *max-k*-alternating diagram $B = \max_{\mathbf{w}_1} \min_{\mathbf{w}_2} \ldots Q^A_{\mathbf{w}_k}\ B(\mathbf{w}_1, \ldots, \mathbf{w}_k)$ where in order to simplify the notation each $\mathbf{w}_i$ may be a single variable or a set of variables and we use the boldface notation to denote this fact.

Now for each tuple i of domain objects in I (which is appropriate for the number of variables in $\mathbf{w}_1$) let diagram B' be $B' = \min_{\mathbf{w}_2} \ldots Q^A_{\mathbf{w}_k} B(\mathbf{w}_1 = i, \ldots, \mathbf{w}_k)$. Clearly B' is appropriate for evaluation on I and by the inductive hypothesis we can appeal to a Π^P_{k-1} oracle to solve GFODD Evaluation on (B', I, V). Our algorithm guesses a value i, calculates B', appeals to the oracle, and returns the same answer. Now, if the true value is $< V$ then by definition any call to the oracle would yield No and we correctly answer No. If the true value is $\geq V$ then for some i the oracle would return Yes. Therefore we nondeterministically return Yes and our algorithm is in $\text{NP}^{\Sigma^P_{k-1}}$. The argument for the other aggregation prefix is symmetric and argued in the same manner yielding an algorithm in $\text{co-NP}^{\Sigma^P_{k-1}}$.

To show hardness we give a reduction from QBF_k. Given a quantified 3CNF Boolean formula we transform this into a GFODD B and interpretation I such that the following claim holds:

Claim 1: B evaluates to 1 in I if and only if the quantified Boolean formula is satisfied.

This claim establishes the theorem. The reduction uses a similar structure to the one used for FODD satisfiability with two main differences. First because here we consider evaluation and we can control I we do not need to test for an embedding of a Boolean predicate in I, that is, the first portion in that construction is not needed. On the other hand the construction and proof are more involved because of the alternation of quantifiers.

The interpretation I has two objects, a and b, where $P_T(a)$ =true, and $P_T(b)$ =false. Namely, $I = \{[a, b], P_T(a) = \text{true}, \text{and } P_T(b) = \text{false}\}$.

Let the QBF formula be $Q_1 x_1 Q_2 x_2 \ldots Q_m x_m f$ where Q is a quantifier $\forall$ or $\exists$ and the quantifiers come in k alternating blocks. As above, we start the construction by creating a set of "shadow variables" corresponding to each QBF variable x_i. The corresponding GFODD variables include w_i and the set of $v_{(a,b)}$ that refer to x_i or $\overline{x_i}$ in the QBF. We define $\mathbf{w}_i$ to be the set of variables in the block corresponding to x_i and associate these variables with an aggregation operator Q^A_i where if Q_i is a $\exists$ then Q^A_i is max and if Q_i is a $\forall$ then Q^A_i is min. Using these variables, we build GFODD fragments we call variable consistency blocks. For each x_i, this gadget ensures that if two literals in the QBF refer to the same variable then the corresponding variables in the GFODD will have the

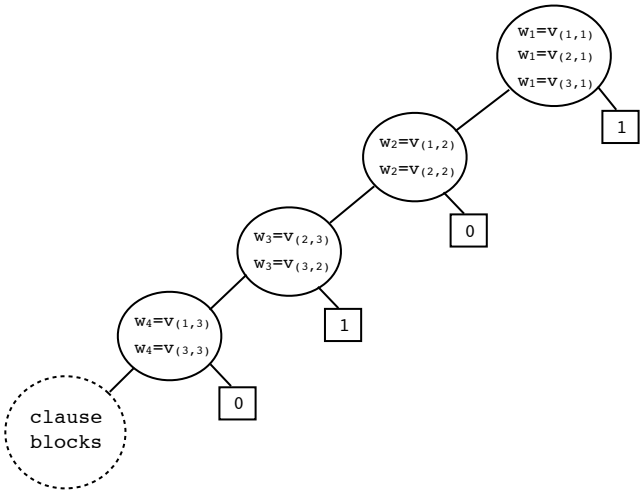

Figure 20: Example of variable consistency blocks for reduction from QBF to GFODD Evaluation for the formula $\forall x_1 \exists x_2 \forall x_3 \exists x_4 (x_1 \vee \overline{x_2} \vee x_4) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee \overline{x_4})$.

same value. If this holds then a valuation goes through the block and continues to the next block. Otherwise, it exits to a default value, where for max blocks the default value is 0, and for min blocks the default value is 1.

Consider the expression

$$\forall x_1 \exists x_2 \forall x_3 \exists x_4 (x_1 \vee \overline{x_2} \vee x_4) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee \overline{x_4}),$$

which has the same clauses as in the previous proof but where we have changed the quantification. Figure 20 shows the variable consistency blocks for this example. Since, $v_{(1,1)}$, $v_{(2,1)}$, and $v_{(3,1)}$ refer to x_1 we need to ensure that when they are evaluated they are evaluated consistently and this is done by the first block. Because x_1 is a $\forall$ variable the default output value is 1. The consistency blocks are chained *in the same order as in the quantification of the QBF*. Once every consistency block has been checked, we continue to the clause blocks whose construction is exactly the same as in the previous proof (see Figure 5). This yields the diagram B where we set the aggregation function to be $Q_1^A \mathbf{w}_1, Q_2^A \mathbf{w}_2, \ldots, Q_m^A \mathbf{w}_m$. Note that if the QBF has k alternating blocks of quantifiers then B has aggregation depth k. The output of the reduction is the pair (B, I). The diagram is ordered with " $=$ " $\prec P_T$ and variables ordered lexicographically.

We next show that Claim 1 holds. We start by showing a correspondence between assignments to the Boolean formula f and object assignments from B to I. Let v be a Boolean assignment. If v assigns x_i to 1 then $\zeta(v)$ maps the entire $\mathbf{w}_i$ block to a. Otherwise $\zeta(v)$ maps the block to b. It is then easy to see that for all v, $\zeta(v)$ satisfies the consistency blocks and $f(v) = 1$ if and only if $\mathrm{MAP}_B(I, \zeta(v)) = 1$. This, however, does not complete the proof

because $\text{MAP}_B(I)$ must also consider valuations ζ that do not arise as maps of assignments v.

We divide the set of valuations to the GFODD into two groups. The first group of legal valuations, called *Group 1* below, is the set of valuations that is consistent with some v.

The second group, *Group 2*, includes valuations that do not arise as $\zeta(v)$ and therefore they violate at least one of the consistency blocks. Let ζ be such a valuation and let Q_j^A be the first block from the left whose constraint is violated. By the construction of B, in particular the order of equality blocks along paths in the GFODD, we have that the evaluation of the diagram on ζ "exits" to a default value on the first violation. Therefore, if Q_j is a $\forall$ then $\text{MAP}_B(I, \zeta) = 1$ and if Q_j is a $\exists$ then $\text{MAP}_B(I, \zeta) = 0$.

We can now show the correspondence in truth values. Consider any partition of the blocks $1, \ldots, m$ into a prefix $1, \ldots, j$ and remainder $(j+1), \ldots, m$, and any Boolean assignment v to the prefix blocks. We claim that for all such partitions

$$Q_{j+1}x_{j+1}, \ldots, Q_m x_m, \quad f\left((x_1, \ldots, x_j) = v, (x_{j+1}, \ldots, x_m)\right) = \\ Q_{j+1}^A \mathbf{w}_{j+1}, \ldots, Q_m^A \mathbf{w}_m, \quad \text{MAP}_B(I, [(\mathbf{w}_1, \ldots, \mathbf{w}_j) = \zeta(v), (\mathbf{w}_{j+1}, \ldots, \mathbf{w}_m)]).$$

Note that when $j = 0$, that is, the prefix is empty, the claim implies that $\text{MAP}_B(I)$ is equal to $Q_1 x_1, \ldots, Q_m x_m, f$, completing the proof. We prove the claim by induction, backwards from m to 0. For the base case, $j = m$, and the second part is empty. The claim then follows because the prefix includes all variables and there is a 1-1 correspondence in truth values for substitutions in group 1.

For the inductive step, the valuation v covers the first $j-1$ blocks. Note that, by the inductive assumption, for any group 1 substitution v_j for x_j and corresponding, $\zeta(v_j)$ for $\mathbf{w}_j$,

$$Q_{j+1}x_{j+1}, \ldots, Q_m x_m, \quad f\left((x_1, \ldots, x_{j-1}) = v, (x_j = v_j), (x_{j+1}, \ldots, x_m)\right) = \\ Q_{j+1}^A \mathbf{w}_{j+1}, \ldots, Q_m^A \mathbf{w}_m, \\ \text{MAP}_B(I, [(\mathbf{w}_1, \ldots, \mathbf{w}_{j-1}) = \zeta(v), (\mathbf{w}_j = \zeta(v_j)), (\mathbf{w}_{j+1}, \ldots, \mathbf{w}_m)]).$$

On the other hand, for any group 2 substitution ζ_j for $\mathbf{w}_j$ and any values for $(\mathbf{w}_{j+1}, \ldots, \mathbf{w}_m)$ we have that the leftmost block whose constraint is violated for the corresponding combined ζ is block j and therefore $\text{MAP}_B(I, [(\mathbf{w}_2, \ldots, \mathbf{w}_{j-1}) = \zeta(v), (\mathbf{w}_j = \zeta_j), (\mathbf{w}_{j+1}, \ldots, \mathbf{w}_m)])$ gets the default value for that block. Therefore, the aggregation over the jth block is determined by group 1 valuations, which are in turn identical to the QBF value and

$$Q_j x_j, \ldots, Q_m x_m, \quad f(x_1, \ldots, x_{j-1}) = v, (x_j, \ldots, x_m)) = \\ Q_j^A \mathbf{w}_j, \ldots, Q_m \mathbf{w}_m, \quad \text{MAP}_B(I, [(\mathbf{w}_2, \ldots, \mathbf{w}_{j-1}) = \zeta(v), (\mathbf{w}_j, \ldots, \mathbf{w}_m)])$$

as required. □

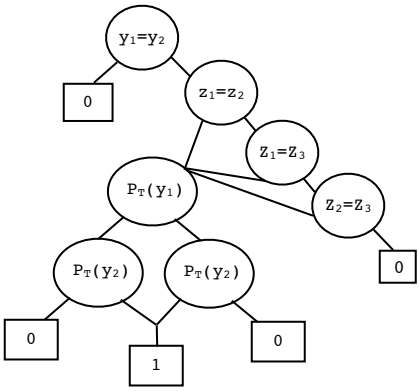

Figure 21: The B_1 diagram for GFODD Satisfiability reduction. The aggregation function is $\max_{y_1} \max_{y_2} \min_{z_1} \min_{z_2} \min_{z_3}$.

It turns out that the complexity of satisfiability is different for min and max diagrams, and their analysis requires different proofs. We therefore start with max-k-alternating GFODDs. The case of min-k-alternating GFODDs is analyzed after the analysis of equivalence because it is using similar techniques.

Theorem 21. *GFODD Satisfiability for* max-k*-alternating GFODDs (where* $k \geq 2$*) is* Σ_k^P*-complete.*

Proof. We first show membership. Let B be a GFODD with aggregation $\max \mathbf{w}_1, \min \mathbf{w}_2, \ldots, Q_k^A \mathbf{w}_k$. Our algorithm nondeterministically chooses an interpretation I and a tuple of values for $\mathbf{w}_1$, from the domain of objects for I. Let i refer to this tuple of objects. We create a new GFODD, $B' = \min \mathbf{w}_2 \ldots Q_k^A B(\mathbf{w}_1 = i, \ldots, w_k)$ and appeal to a Π_{k-1}^P oracle to solve GFODD evaluation on (B', I). If the oracle query returns 1 then we accept and otherwise we reject. The result is clearly correct using an algorithm in $NP^{\Sigma_{k-1}^P}$.

The hardness argument is similar to the proof for GFODD evaluation. The main extension is that in the current proof we verify that any satisfying I embeds the interpretation from the previous proof. The reduction gets a QBF_k formula, $Q_1 x_1 Q_2 x_2 \ldots Q_m x_m f$, with Q_i either a $\forall$ or $\exists$ quantifier. We first construct two diagrams B_1 and B_2, where B_2, the QBF validation diagram, is exactly as in the proof of Theorem 20, that is, it includes consistency blocks followed by clause blocks. The diagram B_1 has two portions. The first verifies that I has at least two objects and the truth values of $P_T()$ on these objects are different. The second portion verifies that I has at most two objects. This is implemented using min variables such that if we identify three distinct objects we set the value to 0. The two portions are put together so as to respect predicate order, and the final diagram B_1 is shown in Figure 21. The aggregation function for B_1 is $\max_{y_1} \max_{y_2} \min_{z_1} \min_{z_2} \min_{z_3}$.

Let $I^* = \{[a, b], P_T(a) = \text{true}, \text{and } P_T(b) = \text{false}\}$ be the intended interpretation. We have the following two claims:
(C1) for all I, $\text{MAP}_{B_1}(I) = 1$ if and only if I is isomorphic to I^*.
(C2) if I is isomorphic to I^* then $\text{MAP}_{B_2}(I) = 1$ if and only if $(Q_1x_1Q_2x_2 \ldots Q_mx_mf) = 1$.

C2 is exactly the same as Claim 1 in the proof of Theorem 20. For C1, given I which is isomorphic to I^*, the valuation of y_1, y_2 to a, b and any valuation to the z's yields a map value of 1. Therefore, considering the aggregation order we see that for $(y_1, y_2) = (a, b)$ in B_1 the minimum over z yields 1, and then the maximum over y's is 1. For the other direction, we need to consider interpretations not isomorphic to I^*. If I has only one object then its map is 0 for all valuations, and therefore the aggregated value is 0. If I has at least 3 objects then for any fixed valuation for y the minimum over z is 0, implying that the maximum over y also yields 0 and $\text{MAP}_{B_1}(I) = 0$. Finally consider any I with two objects where $P_T()$ has the same truth value on the two objects. In this case the map is 0 for any valuation and thus the final map value is 0. We have therefore shown that C1 holds.

For our reduction, we produce $B = \mathit{Apply}(B_1, B_2, \wedge)$ where for the aggregation we make use of Theorem 1 and interleave the aggregation functions of B_1 and B_2 so that B has at most k alternations of quantifiers. This is always possible because the QBF starts with a $\exists$ quantifier and $k \geq 2$.

By the claims C1 and C2 and Theorem 1 we get that $\text{MAP}_B(I) = 1$ if and only if I is isomorphic to I^* and $(Q_1x_1Q_2x_2 \ldots Q_mx_mf) = 1$. Therefore, the QBF is true if and only if there exists an interpretation I (which must be isomorphic to I^*) that satisfies B. □

Equivalence is one level higher in the hierarchy; using a reduction from QBF we show how to "peel off" one level of quantifiers and push that into the "existential quantification" over interpretations that potentially witness non-equivalence.

Theorem 22. *GFODD Equivalence and GFODD Edge Removal for diagrams with aggregation depth k (where $k \geq 2$) are Π^P_{k+1}-complete.*

Proof. By Corollary 18, it suffices to show that this holds for max-k-alternating GFODDs. Since Edge Removal is a special case of Equivalence it suffices to show membership for Equivalence and hardness for Edge Removal.

To show membership we show that the complement, nonequivalence, is in Σ^P_{k+1}. Given two max-k-alternating GFODDs B_1 and B_2 as input, we guess an interpretation I of the appropriate size, and then appeal to an oracle for GFODD Evaluation to calculate $\text{MAP}_{B_1}(I)$ and $\text{MAP}_{B_2}(I)$. Using these values we return Yes or No accordingly. To calculate the map values, let B be one of these diagrams, and let the leaf values of the diagram be $v_1, v_2, \ldots, v_\ell$. We make ℓ calls to GFODD Evaluation with (B, I, v_i) as input. Each call requires an oracle in Σ^P_k and $\text{MAP}_B(I)$ is the largest value on which the oracle returns Yes. Clearly if a witness for nonequivalence exists then this process can discover it and

say Yes (per non-equivalence), and otherwise it will always say No. Therefore non-equivalence is in $\text{NP}^{\Sigma_k^P}$, that is Σ_{k+1}^P and equivalence is in Π_{k+1}^P.

We reduce QBF satisfiability with $k \geq 3$ alternations of quantifiers to equivalence of max-$(k-1)$-alternating GFODDs. The reduction is conceptually similar to the one from the previous theorem but the details are more involved. In particular, here we assume a QBF whose first quantifier is $\forall$, that is, $\forall x_1, Q_2 x_2 \ldots Q_m x_m f(x_1, x_2, \ldots, x_m)$ where this form has k blocks of quantifiers. To simplify the notation it is convenient to collect adjacent variables having the same quantifiers into groups so that the QBF has the form $\forall \mathbf{x}_1 \ldots Q_k \mathbf{x}_k f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$ where $\mathbf{x}_i$ refers to a set of variables.

We next define a notion of "legal interpretations" for our diagrams. A legal interpretation embeds the binary interpretation I^* from the previous proof and in addition includes a truth setting for all the variables in the first $\forall$ block of the QBF. The reduction constructs diagrams B_1, B_2, and $B = \mathit{Apply}(B_1, B_2, \wedge)$ such that the following claims hold:
(C1) for all I, $\text{MAP}_{B_1}(I) = 1$ if and only if I is legal.
(C2) if I is legal and it embeds the substitution $\mathbf{x}_1 = \alpha$ then $\text{MAP}_{B_2}(I) = 1$ if and only if $Q_2 \mathbf{x}_2 \ldots Q_k \mathbf{x}_k f((\mathbf{x}_1 = \alpha), \mathbf{x}_2, \ldots, \mathbf{x}_k) = 1$.

We then output the diagrams (B_1, B) for GFODD equivalence. Note that, by C1 and Theorem 1, for non-legal interpretations we have $\text{MAP}_{B_1}(I) = \text{MAP}_B(I) = 0$ and therefore if the diagrams are not equivalent it must be because of legal interpretations. Now, if the QBF is satisfied then, by definition, for all $\mathbf{x}_1 = \alpha$ we have that $Q_2 \mathbf{x}_2 \ldots Q_k \mathbf{x}_k f((\mathbf{x}_1 = \alpha), \mathbf{x}_2, \ldots, \mathbf{x}_k) = 1$. Therefore, by C2, for all legal I, $\text{MAP}_{B_2}(I) = 1$ and by C1 and the construction $\text{MAP}_B(I) = 1$. Thus B and B_1 are equivalent.

On the other hand, if the QBF is not satisfied then there is a substitution $\mathbf{x}_1 = \alpha$ where $Q_2 \mathbf{x}_2 \ldots Q_k \mathbf{x}_k f((\mathbf{x}_1 = \alpha), \mathbf{x}_2, \ldots, \mathbf{x}_k) = 0$. Therefore, by C2, for the corresponding interpretation I', $\text{MAP}_{B_2}(I') = 0$ and by Theorem 1 we also have $\text{MAP}_B(I') = 0$. But by C1, $\text{MAP}_{B_1}(I') = 1$ and therefore B_1 and B are not equivalent.

We now proceed with the reduction, starting first with a simplified construction ignoring ordering of node labels and edge removal structure, and then elaborating to enforce these constraints. The set of predicates includes $P_T()$ and for every QBF variable x_i in the first $\forall$ block we use a predicates $P_{x_i}()$. Notice that each x_i is a member of $\mathbf{x}_1$ (the first $\forall$ group) where the typeface distinguishes the individual variables in the first block, from blocks of variables. In the simplified construction, a legal interpretation has exactly two objects, say a and b, where $P_T(a) \neq P_T(b)$ and where for each $P_{x_i}()$ we have $P_{x_i}(a) = P_{x_i}(b)$. In other words, the assignment of an object to v in $P_T(v)$ simulates an assignment to Boolean values, but the truth value of $P_{x_i}(v)$ is the same regardless of which object is assigned to v.

In our example QBF $\forall x_1 \exists x_2 \forall x_3 \exists x_4 (x_1 \vee \overline{x_2} \vee x_4) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee \overline{x_4})$ the first block includes only the variable x_1 and the following interpretation is legal: $I = \{[a, b], P_T(a) = \text{true}, P_T(b) = \text{false}, P_{x_1}(a) = P_{x_1}(b) = \text{false}\}$.

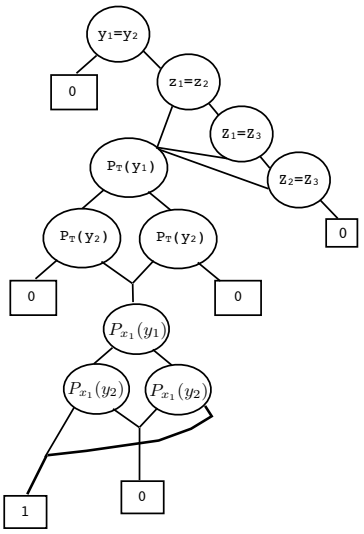

Figure 22: The B_1 diagram for GFODD Equivalence reduction.

The diagram B_1 has three portions where the first two are exactly as in the previous proof, thus verifying that I has two objects and that $P_T()$ behaves as stated. The third portion verifies that each $P_{x_i}()$ behaves as stated, where we use a sequence of blocks, one for each $P_{x_i}()$. The combined diagram B_1 for our example is shown in Figure 22 and the aggregation function is $\max_{y_1,y_2}, \min_{z_1,z_2,z_3}$.

To see that C1 holds consider all possible cases for non-legal interpretations. If I has at most one object the map is 0 for all valuations and thus the aggregation is 0. If I has at least 3 objects, then for any values for y_1, y_2 the min aggregation over z yields 0, and therefore the map is 0. If I has 2 objects but it violates the condition on P_T or P_{x_i} then again the map is 0 for any valuation and the aggregation is 0. On the other hand, if I is legal, then the z block never yields 0 and the correct mapping to y_1, y_2 yields 1. Therefore the aggregation is 1.

The diagram B_2 is constructed by modifying B_2 from the proof of Theorem 20. The first modification is to handle the first $\forall$ block differently. As it turns out, all we need to do is replace the min aggregation for the $\mathbf{w}_1$ block with maximum aggregation and accordingly replace the default value on that block with 0. To avoid confusion we recall that the current proof uses a slightly different notation from the previous one. In the current proof $\mathbf{x}_i$ is a set of variables from the QBF and therefore $\mathbf{w}_i$ is a set of blocks of variables, all of which have the same aggregation function. The modified variable consistency diagram is shown in Figure 23. The clause blocks have the same structure as in the previous construction but use $P_{x_i}(V_{(i_1,i_2)})$ when x_i is a $\forall$ variable from the first block

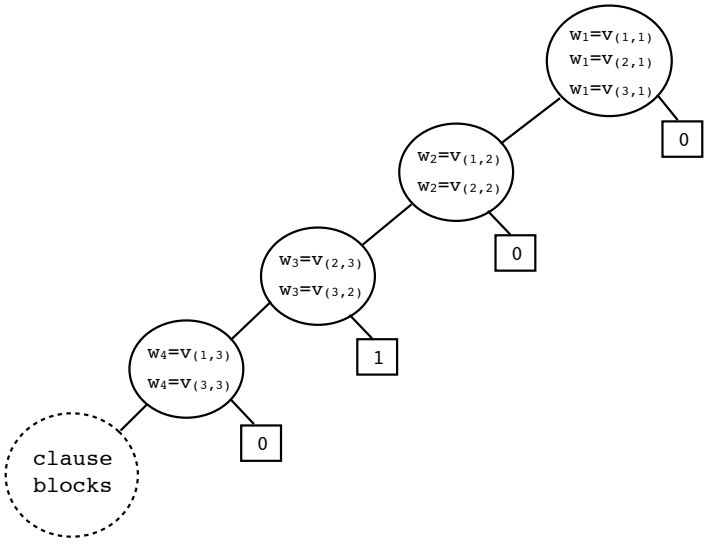

Figure 23: The block consistency diagram for the GFODD equivalence reduction. The only difference from Figure 20 is the default value on the $\mathbf{w}_1$ block.

and use $P_T(V_{(i_1,i_2)})$ otherwise. This is shown in Figure 24. B_2 includes the variable consistency blocks followed by the clause blocks. Note that the new clause blocks are not sorted in any consistent order because the predicates $P_{x_i}()$ and $P_T()$ appear in an arbitrary ordering determined by the appearance of literals in the QBF. Other than this violation, all other portions of the diagrams described are sorted where the predicate order has $= \prec P_T \prec P_{x_i}$ and where variables w_i are before $v_{(i_1,i_2)}$ and variables within group are sorted lexicographically. The combined aggregation function is $\max_{\mathbf{w}_1}, \max_{\mathbf{w}_2}, \min_{\mathbf{w}_3}, \ldots, Q^A_{\mathbf{w}_k}$.

We next show that claim C2 holds, which will complete the proof of the simplified construction. Consider any legal I, let the corresponding truth values for variables in $\mathbf{x}_1$ be denoted α, and consider valuations for the QBF extending $\mathbf{x}_1 = \alpha$. In particular, consider any valuation v to the remaining variables in the QBF and the induced substitution to the GFODD variables $\zeta(v)$ that is easily identified from the construction. Add any consistent group assignment to $\mathbf{w}_1$ (that is, we assign a or b to each subgroup of variables in that group) to $\zeta(v)$ to get $\hat{\zeta}(v)$. By the construction of B_2 we have that $f([\mathbf{x}_1 = \alpha, (\mathbf{x}_2, \ldots, \mathbf{x}_k) = v]) = \mathrm{MAP}_{B_2}(I, \hat{\zeta}(v))$. To see this note that there are no quantifiers in this expression, there is a 1-1 correspondence between the valuations of $\mathbf{x}_2, \ldots, \mathbf{x}_k$ and $\mathbf{w}_2, \ldots, \mathbf{w}_k$, and that as long as the assignment to the $\mathbf{w}_1$ block is group consistent it does not affect the value returned. We call this set of valuations, that arise as translations of substitutions for QBF variables, *Group 1*.

The second group, *Group 2*, includes valuations that do not arise as $\hat{\zeta}(v)$ and therefore they violate at least one of the consistency blocks. Let ζ be such a valuation and let Q^A_j be the first block from the left whose constraint is violated.

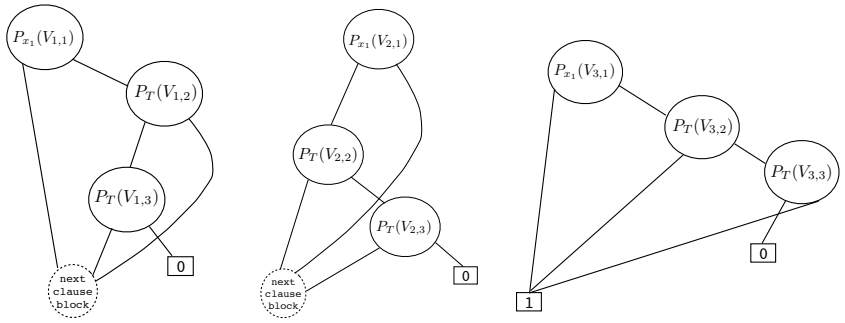

Figure 24: The clause blocks for the GFODD equivalence reduction. For all occurrences of the universal variables of the first block (x_1 in this example) we have replaced P_T with P_{x_i}.

By the construction of B_2, in particular the order of equality blocks along paths in the GFODD, we have that the evaluation of the diagram on ζ "exits" to a default value on the first violation. Therefore, if $j = 1$, that is the violation is in the block of $\mathbf{w}_1$, $\text{MAP}_B(I, \zeta) = 0$ and for $j \geq 2$ if Q_j is a $\forall$ then $\text{MAP}_B(I, \zeta) = 1$ and if Q_j is a $\exists$ then $\text{MAP}_B(I, \zeta) = 0$.

We can now show the correspondence in truth values. Consider any partition of the blocks $2, \ldots, k$ into a prefix $2, \ldots, j$ and remainder $(j+1), \ldots, k$, and any valuation v to the prefix blocks. We claim that for all such partitions

$$Q_{j+1}\mathbf{x}_{j+1}, \ldots, Q_k\mathbf{x}_k, f(\mathbf{x}_1 = \alpha, (\mathbf{x}_2, \ldots, \mathbf{x}_j) = v, \mathbf{x}_{j+1}, \ldots, \mathbf{x}_k) = Q_{j+1}\mathbf{w}_{j+1}, \ldots, Q_k\mathbf{w}_k, \text{MAP}_{B_2}(I, [\mathbf{w}_1 = a, (\mathbf{w}_2, \ldots, \mathbf{w}_j) = \zeta(v), (\mathbf{w}_{j+1}, \ldots, \mathbf{w}_k)]).$$

Note that when $j = 1$, that is, the prefix is empty, this yields that $Q_2\mathbf{x}_2, \ldots, Q_k\mathbf{x}_k, f(\mathbf{x}_1 = \alpha, \mathbf{x}_2, \ldots, \mathbf{x}_k)$ is equal to $Q_2^A\mathbf{w}_2, \ldots, Q_k^A\mathbf{w}_k, \text{MAP}_{B_2}(I, [\mathbf{w}_1 = a, \mathbf{w}_2, \ldots, \mathbf{w}_k)])$. Now because the default value for violations of $\mathbf{w}_1$ is 0 and because the aggregation for $\mathbf{w}_1$ is max the latter expression is equal to $Q_1^A\mathbf{w}_1, \ldots, Q_k^A\mathbf{w}_k, \text{MAP}_{B_2}(I, [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k)])$. This means that $\text{MAP}_{B_2}(I)$ is equal to $Q_2\mathbf{x}_2, \ldots, Q_k\mathbf{x}_k, f(\mathbf{x}_1 = \alpha, \mathbf{x}_2, \ldots, \mathbf{x}_k)$. Therefore, the claim implies C2.

We prove the claim by induction, backwards from k to 1. For the base case, $j = k$, and the second part is empty. The claim then follows because the prefix includes all variables and there is a 1-1 correspondence in truth values for substitutions in group 1.

For the inductive step, the valuation v covers the first $j-1$ blocks. Note that, by the inductive assumption, for any group 1 substitution v_j for $\mathbf{x}_j$ and corresponding, $\zeta(v_j)$ for $\mathbf{w}_j$, $Q_{j+1}\mathbf{x}_{j+1}, \ldots, Q_k\mathbf{x}_k$, $f(\mathbf{x}_1 = \alpha, (\mathbf{x}_2, \ldots, \mathbf{x}_{j-1}) = v, (\mathbf{x}_j = v_j), \mathbf{x}_{j+1}, \ldots, \mathbf{x}_k) = Q_{j+1}^A\mathbf{w}_{j+1}, \ldots, Q_k^A\mathbf{w}_k, \text{MAP}_{B_2}(I, [\mathbf{w}_1 = a, (\mathbf{w}_2, \ldots, \mathbf{w}_{j-1}) = \zeta(v), (\mathbf{w}_j = \zeta(v_j)), (\mathbf{w}_{j+1}, \ldots, \mathbf{w}_k)])$. On the other hand, for any group 2 substitution ζ_j for $\mathbf{w}_j$ and any values for $(\mathbf{w}_{j+1}, \ldots, \mathbf{w}_k)$ we have that the violating block for the corresponding combined ζ is block j and therefore $\text{MAP}_{B_2}(I, [\mathbf{w}_1 = a, (\mathbf{w}_2, \ldots, \mathbf{w}_{j-1}) = \zeta(v), (\mathbf{w}_j = \zeta_j), (\mathbf{w}_{j+1}, \ldots, \mathbf{w}_k)])$ gets the

default value for that block. Therefore, the map is determined by group 1 valuations, which are in turn identical to the QBF value and $Q_j\mathbf{x}_j, \ldots, Q_k\mathbf{x}_k$, $f(\mathbf{x}_1 = \alpha, (\mathbf{x}_2, \ldots, \mathbf{x}_{j-1}) = v, (\mathbf{x}_j, \ldots, \mathbf{x}_k)) = Q_j^A\mathbf{w}_j, \ldots, Q_k^A\mathbf{w}_k, \mathrm{MAP}_{B_2}(I, [\mathbf{w}_1 = a, (\mathbf{w}_2, \ldots, \mathbf{w}_{j-1}) = \zeta(v)\ , (\mathbf{w}_j, \ldots, \mathbf{w}_k)])$ as required. Therefore, the claim on the correspondence of values holds. This completes the proof of C2.

Extending the reduction to handle ordering and edge removal special case. The main idea in the extended construction is to replace the unary predicates P_T and P_{x_i} with one binary predicate $q(\cdot, \cdot)$ where the "second argument" in $q()$ serves to identify the corresponding predicate and hence its truth value. In addition we force $q()$ to be symmetric so that for any A and B the truth value of $q(A, B)$ is the same as the truth value of $q(B, A)$. In this way we have freedom to use either $q(A, B)$ or $q(B, A)$ as the node label which provides sufficient flexibility to handle the ordering issues. To implement this idea we need a few additional constructions.

Let the set of variables in the first $\forall$ block of the QBF be $x_1, x_2, \ldots, x_\ell$. The set of predicates in the extended reduction includes unary predicates $T(), y_1(), y_2(), x_1(), x_2(), \ldots, x_\ell()$, and one binary predicate $q(\cdot, \cdot)$. A legal interpretation includes exactly $\ell + 3$ objects which are uniquely identified by the unary predicates. We therefore slightly abuse notation and use the same symbols for the objects and predicates. In particular, the atoms $y_1(y_1), y_2(y_2)$, $T(T)$, and $x_1(x_1), x_2(x_2), \ldots, x_\ell(x_\ell)$ are true in the interpretation and only these atoms are true for these unary predicates (e.g., $x_1(T)$ is false). The truth values of $q()$ reflect the simulation of $P_T()$ and $P_{x_i}()$ in addition to being symmetric. Thus the truth values of $q(y_1, T)$ and $q(T, y_1)$ are the same and they are the negation of the truth values of $q(y_2, T)$ and $q(T, y_2)$. For all i, the truth values of $q(y_1, x_i)$, $q(x_i, y_1)$, $q(y_2, x_i)$ and $q(x_i, y_2)$ are the same. The truth values of other instances of $q()$, for example, $q(x_2, T)$, can be set arbitrarily. For example, the following interpretation is legal when $\ell = 1$: $I = \{[a, b, c, d], y_1(a), y_2(b), T(c), x_1(d), q(c, a) = q(a, c) = \text{true}, q(c, b) = q(b, c) = \text{false}, q(d, a) = q(a, d) = q(d, b) = q(b, d) = \text{false}, q(\cdot, \cdot) = \text{false}\}$ where $q(\cdot, \cdot)$ refers to any instance not explicitly mentioned in the list.

We next define the diagram B_1 that is satisfied only in legal interpretations. We enforce exactly $\ell + 3$ objects using two complementary parts. The first includes $\binom{\ell+4}{2}$ inequalities on a new set of $\ell + 4$ variables $z_1, \ldots, z_{\ell+4}$ with min aggregation. If we identify $\ell + 4$ distinct objects we set the value to 0. This is shown in Figure 25.

To enforce at least $\ell+3$ objects and identify them we use the following gadget. For each of the unary predicates we have a diagram identifying its object and testing its uniqueness where we use both max and min variables. This is shown for the predicate $T()$ in Figure 26. The node $T(T)$ with max variable T identifies the object T. The nodes $T(r_1), T(r_2)$ with min variables r_1, r_2 make sure that T holds for at most one object. We chain the diagrams together as shown in Figure 27 where the variables r_1, r_2 are shared among all unary predicates. The corresponding aggregation function is $\max_{T, y_1, y_2, x_1, \ldots, x_\ell}, \min_{r_1, r_2}$. This diagram associates each of the $\ell + 3$ objects with one of the unary predicates and in this

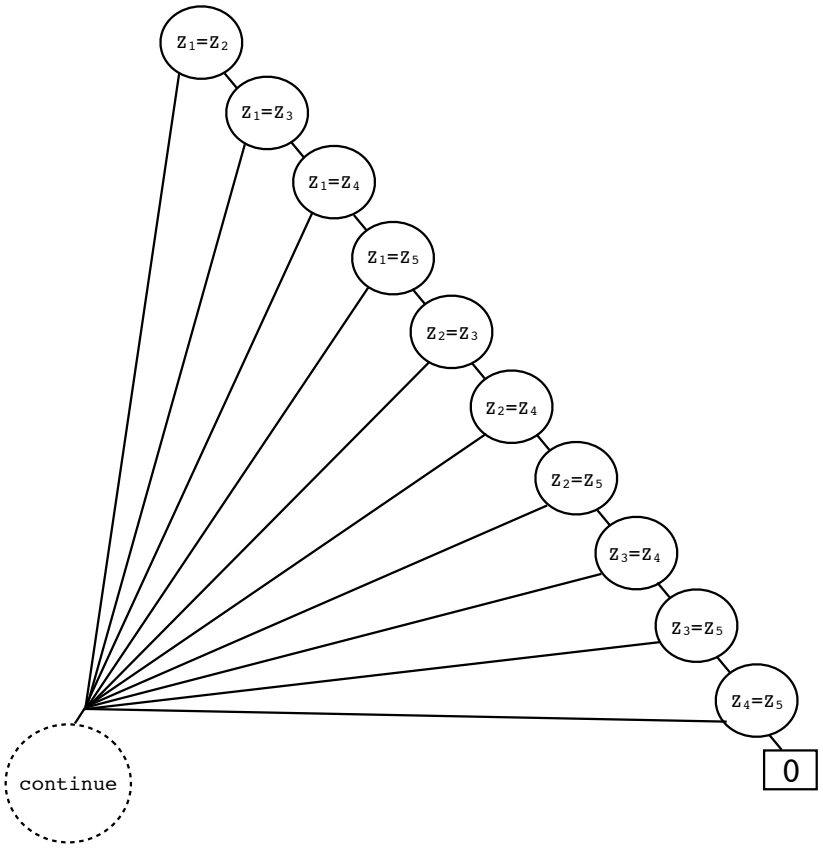

Figure 25: GFODD equivalence reduction. The diagram verifying that the interpretation has less than $\ell + 4$ objects. Here $\ell = 1$ and $\ell + 4 = 5$.

way provides a reference to specific objects in the interpretation.

The symmetry gadget for $q()$ is shown in Figure 28 where the variables m_1, m_2 are min variables. If an input interpretation has two objects A, B where $q(A, B)$ has a truth value different than $q(B, A)$ then minimum aggregation will map the interpretation to 0.

The truth value gadget for the simulation of P_T is shown in Figure 29 and the truth value gadget for the simulation of P_{x_i} is shown in Figure 30. These diagram fragments refer to variables in other portions and they will be connected and aggregated together.

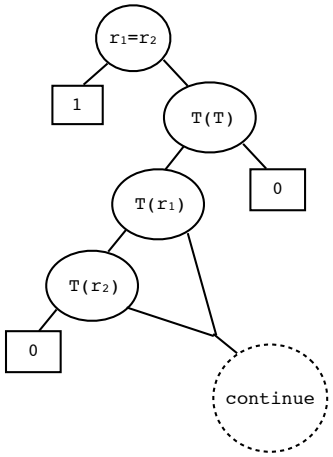

Figure 26: GFODD equivalence reduction. The diagram for the unary predicate $T()$ identifies one object for which $T()$ holds, and makes sure that $T()$ does not hold for two objects.

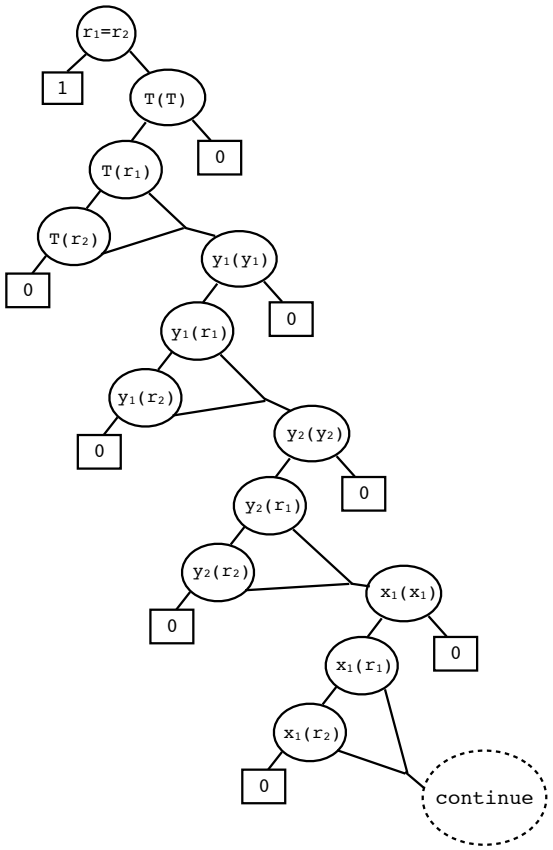

Figure 27: GFODD equivalence reduction. The diagram verifying that each unary predicate corresponds to exactly one object in the interpretation. The corresponding aggregation function is $\max_{T,y_1,y_2,x_1,\ldots,x_\ell} \min_{r_1,r_2}$.

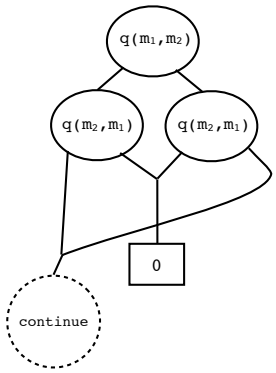

Figure 28: GFODD equivalence reduction. Diagram fragment verifying that $q()$ is symmetric. The corresponding aggregation function is $\min_{m_1,m_2}$.

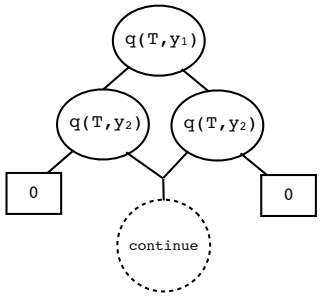

Figure 29: GFODD equivalence reduction. Diagram fragment verifying that $q()$ simulates truth values of $P_T()$ from the simple construction.

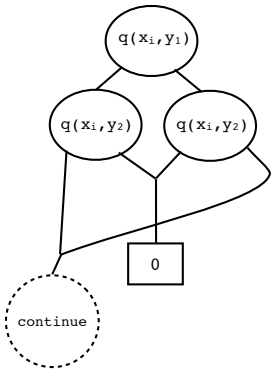

Figure 30: GFODD equivalence reduction. Diagram fragment verifying that $q()$ simulates truth values of $P_{x_i}()$ from the simple construction.

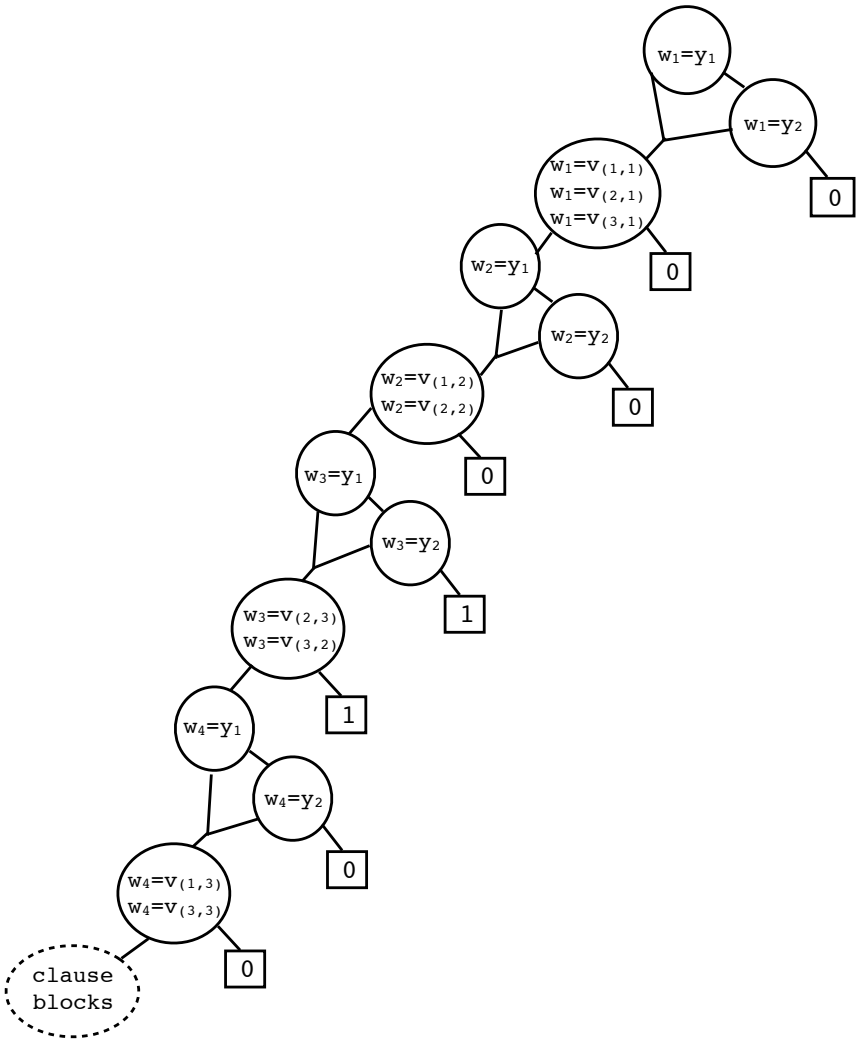

Figure 31: The diagram checking block consistency for the ordered version of the GFODD equivalence reduction.

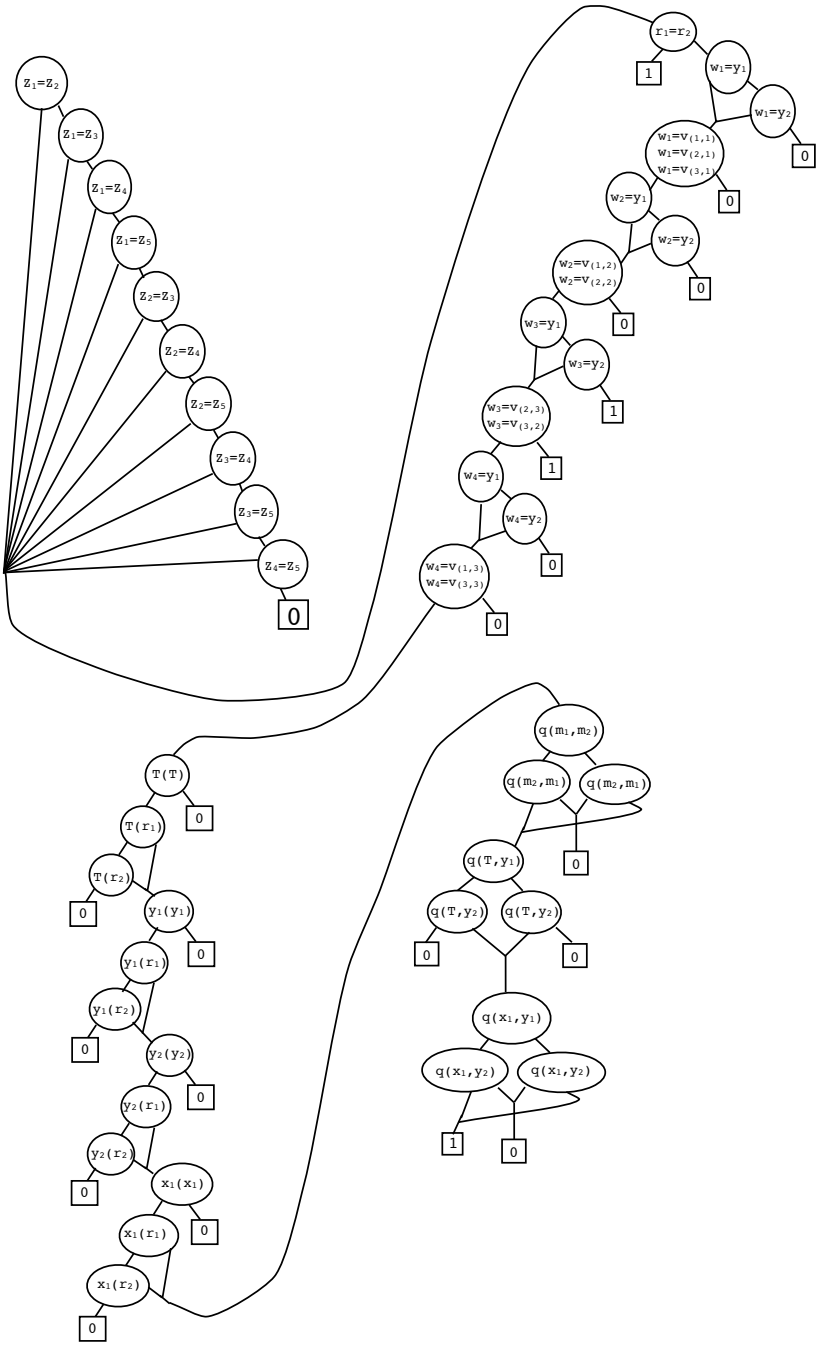

Figure 32: The final version of the diagram B_1 for the ordered version of the GFODD equivalence reduction. The complete aggregation function is $\max_{T,y_1,y_2,x_1,\ldots,x_\ell} \max_{\mathbf{w}_1,\mathbf{w}_2} \min_{m_1,m_2} \min_{r_1,r_2} \min_{z_1,\ldots,z_{\ell+4}} \min_{\mathbf{w}_3} \cdots Q^A_{\mathbf{w}_k}$.

Finally, we add a component that is not needed for verifying that I is legal but will be useful later when we include the clauses. In particular, we include a variable consistency block which is similar to the one in the simple construction (see Figure 23) but where we force each subgroup in $\mathbf{w}_i$ to bind to the same object as y_1 or y_2. This is shown in Figure 31.

We chain the diagrams together as shown in Figure 32 to get B_1 where we have moved the node labelled $r_1 = r_2$ to be above the block consistency gadget. Note that the diagram is sorted where for predicate order we have " $=$ " $\prec T \prec y_1 \prec y_2 \prec x_i \prec q$ and for variables we have $z_i, T, y_i, x_i \prec r_j \prec w_l$, and $y_i \prec v_{(i_1,i_2)}$, and $m_i \prec T \prec x_j$. The complete aggregation function is

$$\max_{T,y_1,y_2,x_1,\ldots,x_\ell} \max_{\mathbf{w}_1,\mathbf{w}_2} \min_{m_1,m_2} \min_{r_1,r_2} \min_{z_1,\ldots,z_{\ell+4}} \min_{\mathbf{w}_3} \ldots Q^A_{\mathbf{w}_k}.$$

We next show that the claim C1 holds for the extended reduction.[4] We first consider all possible cases for illegal interpretations.

- If the interpretation has $\geq \ell+4$ objects then the top portion of the diagram yields 0 for some valuation of z's. Consider any valuation ζ_p to the prefix of variables $T, y_1, y_2, x_1, \ldots, x_\ell$, $\mathbf{w}_1, \mathbf{w}_2$, m_1, m_2, r_1, r_2, and any ζ which is an extension of this valuation, has the violating combination for $z_1, \ldots, z_{\ell+4}$ and any valuation for the other variables. We have $\text{MAP}_{B_1}(I, \zeta) = 0$. Therefore, all aggregations from $\mathbf{w}_k$ to $\mathbf{w}_3$ yield a value of 0 for this prefix. Now continuing backwards, the minimization over z yields a value of 0 for ζ_p, and this holds for any ζ_p. Therefore, all remaining aggregations yield 0 and the final value is 0.

- Next, consider the case where the interpretation has $\leq \ell + 3$ objects but one of the unary predicates is always false (i.e., it does not "pick" any object). The situation is similar to the previous case, but here we get a value of 1 for ζ where $r_1 = r_2$ or where there is a block violation for some $\mathbf{w}_i$ block with min aggregation.

  Consider any valuation ζ_p to the prefix of variables $T, y_1, y_2, x_1, \ldots, x_\ell$, $\mathbf{w}_1, \mathbf{w}_2$, m_1, m_2, which is block consistent on $\mathbf{w}_1, \mathbf{w}_2$ and any ζ which is an extension of this valuation, has $r_1 \neq r_2$, and any valuation for the other variables. We have two cases: if ζ is in group 1 the map is 0 because the unary predicate test fails, and if ζ is in group 2 the map is the default value of the first violated block $\mathbf{w}_i$. Now, because there is at least one group 1 valuation, we can argue inductively backwards from k that all aggregations from k to 3 yield 0, and the same holds for the minimization over z. Next, because the value is 0 for $r_1 \neq r_2$, the minimization over r yields 0 for ζ_p. Considering any variants of ζ_p which are not block

[4]Note that we can remove the variable consistency block which complicates the argument (and does not test anything per legality of I) and still maintain correctness of C1. But including it here simplifies the argument for diagram B below and thus simplifies the overall proof.

consistent on $\mathbf{w}_1, \mathbf{w}_2$ but otherwise identical we see that their value is also 0. As a result all remaining aggregations yield 0.

- Next consider the case where the previous two conditions are satisfied but where one of the unary predicates holds for two or more objects. The argument is identical to the previous case, except that ζ has the violating pair for r_1, r_2 (instead of any $r_1 \neq r_2$).

- Next consider any interpretation that has exactly $\ell + 3$ objects and where the unary predicates identify the objects corresponding to $T, y_1, y_2, x_1, \ldots, x_\ell$ but where $q()$ is not symmetric. In this case we consider a valuation ζ_p to the prefix of variables $T, y_1, y_2, x_1, \ldots, x_\ell$, $\mathbf{w}_1, \mathbf{w}_2$, and extensions ζ that have the violating pair for m_1, m_2. As above, starting with ζ that are block consistent on $\mathbf{w}_1, \mathbf{w}_2$ and have $r_1 \neq r_2$ we can argue that the aggregations down to m yield 0, and as a result that the aggregation over m yields 0. As ζ_p is arbitrary, it follows that the final value is 0.

- The only remaining cases are interpretations that are illegal only because the $q()$ simulation of $P_T()$ or $P_{x_i}()$ is not as required. In this case, the same argument as in the 2nd item in this list shows that the value is 0.

Therefore, if I is illegal then $\text{MAP}_{B_1}(I) = 0$. Consider next any legal interpretation and the intended valuation ζ_p to $T, y_1, y_2, x_1, \ldots, x_\ell$. For any group 1 extension of this valuation and any valuation of the variables m, z, r the diagram yields 1. Therefore, we can argue inductively that all $\mathbf{w}_i$ aggregations down to $\mathbf{w}_3$ yield 1, and the min aggregations over m, z, r yield 1 for ζ_p. Therefore, the max aggregation over $T, y_1, y_2, x_1, \ldots, x_\ell$ yields 1, and $\text{MAP}_{B_1}(I) = 1$. This completes the proof of C1.

The diagram B is obtained by adding the clause blocks below the $q()$ tests of B_1. The clause blocks have the same structure as above but they use $q()$ instead of $P_T()$ and $P_{x_i}()$. This is shown in Figure 33. The final diagram for B is shown in Figure 34 and it has the same aggregation function as B_1. Note that the diagram is also sorted using the same order as in B_1 with the addition that $x_i \prec V_{(i_1,i_2)}$. We next show that:
(C2') $\text{MAP}_B(I) = 1$ if and only if I is legal and $Q_2\mathbf{x}_2 \ldots Q_k\mathbf{x}_k f((\mathbf{x}_1 = \alpha), \mathbf{x}_2, \ldots, \mathbf{x}_k) = 1$ where I embeds the substitution $\mathbf{x}_1 = \alpha$.

The property C2' is slightly different from C2 above, in that it argues directly over B and includes the condition on legal interpretations. Nonetheless, the same argument from the simple case can be used to show that C1 and C2' imply that B_1 and B are equivalent iff the QBF is satisfied.

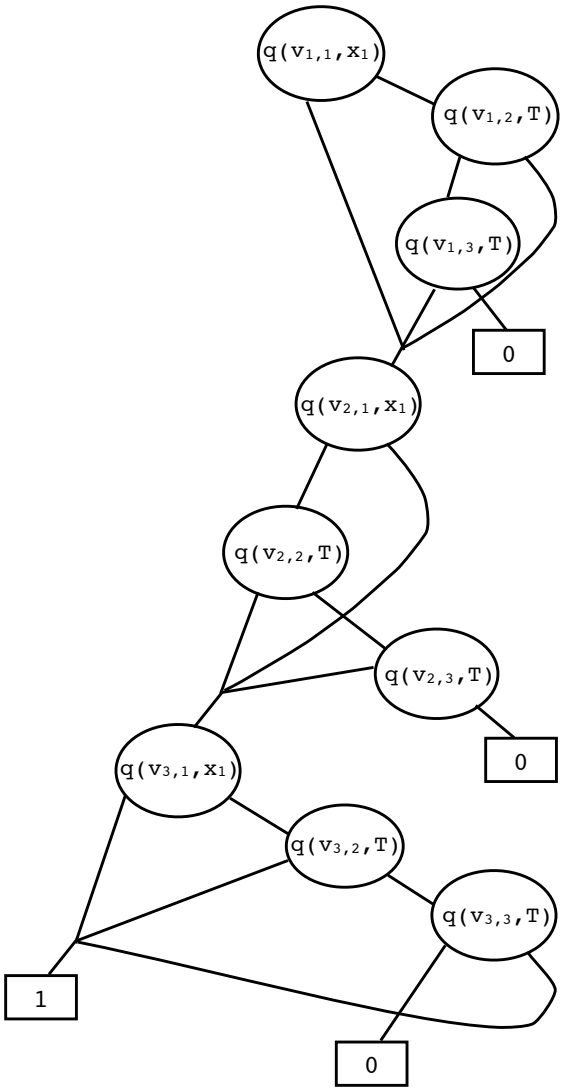

Figure 33: GFODD equivalence reduction. Diagrams capturing the clause blocks.

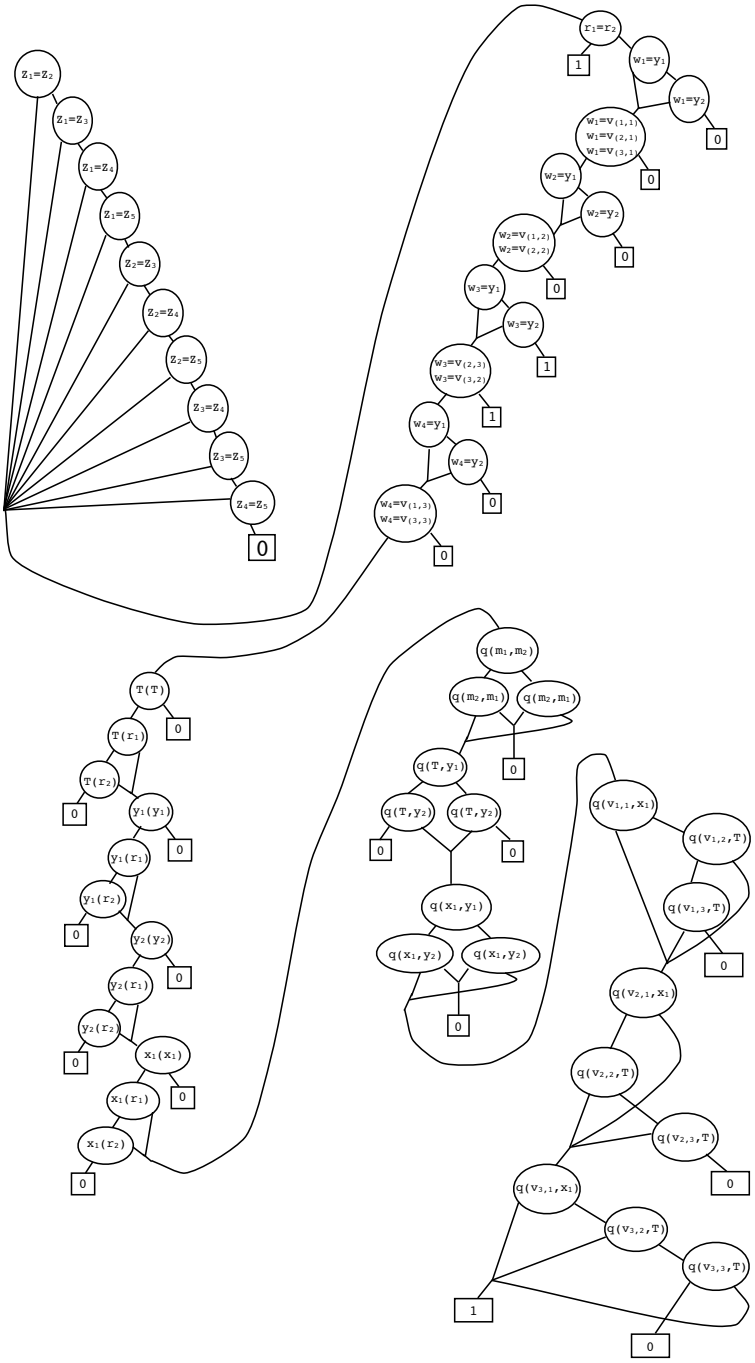

Figure 34: The final version of the diagram B for the ordered version of the GFODD equivalence reduction. The complete aggregation function is $\max_{T,y_1,y_2,x_1,\ldots,x_\ell} \max_{\mathbf{w}_1,\mathbf{w}_2} \min_{m_1,m_2} \min_{r_1,r_2} \min_{z_1,\ldots,z_{\ell+4}} \min_{\mathbf{w}_3} \ldots Q^A_{\mathbf{w}_k}$.

To prove the claim first note that by the construction B adds more tests on the path to a 1 leaf of B_1 and does not add any other paths to a value of 1. Therefore, for any I and any ζ, if $\text{MAP}_B(I,\zeta) = 1$ then $\text{MAP}_{B_1}(I,\zeta) = 1$ and as a result if $\text{MAP}_B(I) = 1$ then $\text{MAP}_{B_1}(I) = 1$. Therefore, by C1, if $\text{MAP}_B(I,\zeta) = 1$ then I is legal. Next, consider any legal I and any unintended valuation ζ_p for the prefix $T, y_1, y_2, x_1, \ldots, x_\ell$. Any valuation extending this prefix with a block consistent assignment for all $\mathbf{w}_i$ and with $r_1 \neq r_2$ will yield a zero. Therefore, the minimization over r_1, r_2 will yield 0 for this prefix, and so will the minimization over m_1, m_2 and maximization over $\mathbf{w}_1, \mathbf{w}_2$. We conclude that the aggregated value for this prefix is 0. Therefore, if $\text{MAP}_B(I) = 1$ and thus the max aggregation over these prefix variables yields 1, it must be through the intended valuation ζ_p for the prefix $T, y_1, y_2, x_1, \ldots, x_\ell$.

However, when $T, y_1, y_2, x_1, \ldots, x_\ell$ are fixed to their intended values, the portions of the diagram testing for $\leq q + 3$ objects, the uniqueness of special objects $T, y_1, y_2, x_1, \ldots, x_k$, the symmetry of $q()$ and its simulation of $P_T()$ and P_{x_i} do not affect the final value in the sense that a valuation reaching them always continues to the next block. Therefore, if we restrict attention to such valuations we can shrink B by removing these blocks and still obtain the same aggregated value. Now we observe that there is a 1-1 correspondence between valuations and values of the resulting B to valuations and values of B_2 in the simple construction (where we extend the notion of block consistent to enforce that $\mathbf{w}_i$ bind to y_1, y_2). Therefore, the claim holds by C2 of the simple construction. □

The result for GFODD Value is similar to the FODD case.

Theorem 23. *GFODD Value for diagrams with aggregation depth k (where $k \geq 2$) is Σ^P_{k+1}-complete.*

Proof. The proof of Theorem 16 goes through almost directly and requires only a slight wording variation. For membership we get the bound on interpretation size by the assumption on the input; then the algorithm is the same.

For the reduction, we use Theorem 1 to calculate $B = \mathit{Apply}(B_1, B_2, +)$. As stated in that theorem, we can mesh together the aggregation lists of B_1 and B_2 interleaving the max and min blocks from each diagram without increasing quantifier depth and the diagram B has the same quantifier prefix and depth as those of B_1 and B_2. □

Unlike max-k-alternating GFODDs, for min diagrams the search for a satisfying interpretation cannot be absorbed into the first aggregation operator. This fact pushes the problem one level higher in the hierarchy.

Theorem 24. *GFODD Satisfiability for* min-k*-alternating GFODDs (where $k \geq 2$) is Σ^P_{k+1}-complete.*

Proof Sketch. For membership, we guess an interpretation I of the appropriate size, and then appeal to a Σ^P_k oracle to solve GFODD evaluation for (B, I). This yields an algorithm in $NP^{\Sigma^P_k}$.

The hardness result uses a slight modification of the equivalence proof, which we sketch next. One can verify that all the details of the modification go through to establish the result.

In particular, we reduce QBF satisfiability with $k \geq 3$ alternations of quantifiers to satisfiability of min-$(k-1)$-alternating GFODDs. Here we assume a QBF whose first quantifier is $\exists$, that is has the form $\exists_1 \mathbf{x}_1 \ldots Q_k \mathbf{x}_k f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$ where $\mathbf{x}_i$ refers to a set of variables. We build B_1, B_2 and B exactly as in Theorem 22 with one exception: the leaf values on the diagram that checks for block consistency are flipped from the previous construction (because the corresponding aggregation operators are switched). The reduction still provides B_1, B_2, and $B = \mathit{Apply}(B_1, B_2, \wedge)$ such that the following claims hold:
(C1) for all I, $\mathrm{MAP}_{B_1}(I) = 1$ if and only if I is legal.
(C2) if I is legal and it embeds the substitution $\mathbf{x}_1 = \alpha$ then $\mathrm{MAP}_{B_2}(I) = 1$ if and only if $Q_2 \mathbf{x}_2 \ldots Q_k \mathbf{x}_k f((\mathbf{x}_1 = \alpha), \mathbf{x}_2, \ldots, \mathbf{x}_k) = 1$.

We then output the diagram B for GFODD satisfiability. Now, if the QBF is satisfied then there exists a value α such that for $\mathbf{x}_1 = \alpha$ we have that $Q_2 \mathbf{x}_2 \ldots Q_k \mathbf{x}_k f((\mathbf{x}_1 = \alpha), \mathbf{x}_2, \ldots, \mathbf{x}_k) = 1$. Therefore, by C2, for the legal I that embeds α, $\mathrm{MAP}_{B_2}(I) = 1$, and by Theorem 1 we have that $\mathrm{MAP}_B(I) = 1$. On the other hand, if the QBF is not satisfied then for all substitutions $\mathbf{x}_1 = \alpha$ we have $Q_2 \mathbf{x}_2 \ldots Q_k \mathbf{x}_k f((\mathbf{x}_1 = \alpha), \mathbf{x}_2, \ldots, \mathbf{x}_k) = 0$. Therefore, by C2, all legal I (and any α they embed) $\mathrm{MAP}_{B_2}(I) = 0$ and by Theorem 1 we also have $\mathrm{MAP}_B(I) = 0$. By C1, $\mathrm{MAP}_B(I) = 0$ for non-legal interpretations. Therefore, B is not satisfiable.

The construction to handle ordering and edge removal structure can be similarly modified for the current proof. □

Up to this point, all hardness proof in the paper use a signature without any constants, i.e., we use equality and unary and binary predicates. For min-GFODDs (the case $k = 1$) the use of constants affects the complexity of the satisfiability problem. In particular, for a signature without constants, if a min-GFODD is satisfied by interpretation I, then it is satisfied by the sub-interpretation of I with just one object (any object in I will do). Moreover, given diagram B and a specific I with one object, model evaluation is in P because there is only one valuation to consider. Therefore, in this case satisfiability is in NP: we can guess the interpretation (i.e. truth values of predicates) and evaluate $\mathrm{MAP}_B(I)$ in polynomial time. On the other hand, if we allow constants in the signature the problem follows the same scheme as above and is Σ_2^P-complete.

Theorem 25. *GFODD Satisfiability for* min-*GFODDs is* Σ_2^P*-complete.*

Proof Sketch. Membership is as in the general case. For hardness, we use the construction in the reduction of the previous proof which yields a GFODD with aggregation $\min^* \max^*$ (i.e., the portion starting with $\mathbf{w}_3$ does not exist) where the max variables are $T, y_1, y_2, x_1, \ldots, x_\ell$. We then turn these variables into constants and remove the max aggregation to yield a min GFODD. One can verify that the proof of Theorem 24 goes through with very minor changes. □

5. Discussion

In this paper we explored the complexity of computations using FODD and GFODD with min and max aggregation, providing a classification placing them within the polynomial hierarchy, where, roughly speaking, equivalence is one level higher in the hierarchy than evaluation and satisfiability. These results are useful in that they characterize the complexity of the problems solved heuristically by implementations of GFODD systems [15, 17, 18] and can be used to partly motivate or justify the use of these heuristics. For example, the "model checking reductions" of [15] replace equivalence tests with model evaluation on a "representative" set of models, and choosing this set heuristically leads to inference that is correct with respect to these models but otherwise incomplete. Our results here show that this indeed leads to reduction of the complexity of the inference problem so that the loss in accuracy is traded for improved worst case run time. As mentioned above, the proofs in the paper can be used (in simpler form) to show the same complexity results for the corresponding problems in first order logic. To our knowledge the corresponding complexity questions for first order logic, with an explicit bound on model size, have not been studied before. Yet they can be useful in many contexts where such a bound can be given in a practical setting. For example, in such cases existing optimized QBF algorithms can be used for inference in this restricted form of first order logic.

There are several important directions for further investigation. The first involves using a richer set of aggregation operators. In particular the definition of GFODDs allows for any function to aggregate values, and functions such as sum, product, and average are both natural and useful for modeling and solving Markov Decision Processes, which have been the main application for FODDs. The work of [18] extends the model checking reductions to GFODDs with average aggregation. Clarifying the complexity of these problems and identifying the best algorithms for them is an important effort for the efficiency of such systems. In this context, it is also interesting to clarify the relationship to query languages in databases that allow for similar aggregations and to formulations of logic with counting that have been developed in the context of database theory [24].

Considering this wider family of GFODDs also raises new computational questions beyond the ones explored in this paper. One such question arises from the connection to statistical relational models and specifically to lifted inference in such models (see e.g. [28, 27, 5]). In particular, consider Markov Logic Networks (MLN) [28] that can be seen to define a distribution over interpretations through a log linear probability model, where features of this model are defined by simple first order formula templates. It is easy to show that such templates and their weights can be encoded using a GFODD with product aggregation, and that these can be combined using a variant of the Apply procedure. The main computational question in this context has been to calculate the probability of a query given the MLN model, and the number of objects n in the domain. Let $\mathcal{I}$ be the set of interpretations with n objects over the relevant

signature. In our case, this question translates to calculating

$$\sum_{I \in \mathcal{I}} \text{MAP}_B(I)$$

for an appropriate B that combines the query and the MLN model. This is closely related to the approaches that solve this problem via lifted weighted model counting [5]. A similarly interesting question would require us to calculate the best I for a particular B

$$\text{argmax}_{I \in \mathcal{I}} \text{MAP}_B(I).$$

In this case, if B captures say profit of some organization, then the computation optimizes the setting so as to maximize profit. Thirdly, we have defined a logic-inspired language but did not define or study any notion of implication. A natural notion of implication with numerical values, related to the one used by [7], is majorization:

$$B_1 \models B_2 \quad \Leftrightarrow \quad \forall_{I \in \mathcal{I}} \text{MAP}_{B_1}(I) \leq \text{MAP}_{B_2}(I).$$

Efficient algorithms and complexity analysis for these new questions will expand the scope and applicability of GFODDs.

Finally, efficient algorithms for model evaluation play an important role in GFODD implementations. The work of [18] provides a generic algorithm inspired by the variable elimination algorithm from probabilistic inference. Several application areas, including databases, AI, and probabilistic inference have shown that more efficient algorithms are possible when the input formula or graph have certain structural properties such as low graph width. We therefore conjecture that similar notions can be developed to provide more efficient evaluation for GFODDs with some structural properties. Coupled with model checking reductions, this can lead to realizations of GFODD systems that combine high expressive power going beyond first order logic with efficient algorithms.

Acknowledgments

This work is partly supported by NSF grant IIS 0964457. Some of this work was done when Roni Khardon was on sabbatical leave at Trinity College Dublin.

References

[1] R. Bahar, E. Frohm, C. Gaona, G. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic decision diagrams and their applications. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 188–191, 1993.

[2] C. Boutilier, R. Reiter, and B. Price. Symbolic dynamic programming for first-order MDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 690–700, 2001.

[3] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[4] C. Chang and J. Keisler. *Model Theory*. Elsevier, Amsterdam, Holland, 1990.

[5] G. Van den Broeck, N. Taghipour, W. Meert, J. Davis, and L. De Raedt. Lifted probabilistic inference by first-order knowledge compilation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2178–2185, 2011.

[6] R. Fagin. Finite-model theory - a personal perspective. *Theoretical Computer Science*, 116(1&2):3–31, 1993.

[7] G. J. Gordon, S. A. Hong, and M. Dudík. First-order mixed integer linear programming. In *Proceedings of the Workshop on Uncertainty in Artificial Intelligence*, pages 213–222, 2009.

[8] E. Grädel. Decidable fragments of first-order and fixed-point logic. From prefix-vocabulary classes to guarded logics. In *Proceedings of Kalmár Workshop on Logic and Computer Science, Szeged*, 2003.

[9] J. Groote and O. Tveretina. Binary decision diagrams for first order predicate logic. *Journal of Logic and Algebraic Programming*, 57:1–22, 2003.

[10] B. J. Hescott and R. Khardon. The complexity of reasoning with FODD and GFODD. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1056–1062, 2014.

[11] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the Workshop on Uncertainty in Artificial Intelligence*, pages 279–288, 1999.

[12] S. Hölldobler, E. Karabaev, and O. Skvortsova. FluCaP: a heuristic search planner for first-order MDPs. *Journal of Artificial Intelligence Research*, 27:419–439, 2006.

[13] S. Hölldobler and O. Skvortsova. A logic-based approach to dynamic programming. In *AAAI-04 workshop on learning and planning in Markov Processes – advances and challenges*, 2004.

[14] S. Homer and A. L. Selman. *Computabilty and Complexity Theory*. Springer-Verlag, New York, 2001.

[15] S. Joshi, K. Kersting, and R. Khardon. Self-taught decision theoretic planning with first order decision diagrams. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 89–96, 2010.

[16] S. Joshi, K. Kersting, and R. Khardon. Decision theoretic planning with generalized first order decision diagrams. *Artificial Intelligence*, 175:2198–2222, 2011.

[17] S. Joshi and R. Khardon. Probabilistic relational planning with first order decision diagrams. *Journal of Artificial Intelligence Research*, pages 231–266, 2011.

[18] S. Joshi, R. Khardon, A. Raghavan, P. Tadepalli, and A. Fern. Solving relational MDPs with exogenous events and additive rewards. In *Proceedings of the European Conference on Machine Learning*, pages 178–193, 2013.

[19] S. Joshi, P. W. Schermerhorn, R. Khardon, and M. Scheutz. Abstract planning for reactive robots. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 4379–4384, 2012.

[20] K. Kersting, M. Van Otterlo, and L. De Raedt. Bellman goes relational. In *Proceedings of the International Conference on Machine Learning*, pages 465–472, 2004.

[21] R. Khardon, H. Mannila, and D. Roth. Reasoning with examples: Propositional formulae and database dependencies. *Acta Informatica*, pages 267–286, 1999.

[22] R. Khardon and D. Roth. Reasoning with models. *Artificial Intelligence*, 87:187–213, 1996.

[23] R. Khardon and D. Roth. Learning to reason. *Journal of the ACM*, 44(5):697–725, 1997.

[24] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[25] J.W. Lloyd. *Foundations of Logic Programming*. Springer Verlag, 1987. Second Edition.

[26] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[27] L. De Raedt, A. Kimmig, and H. Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2462–2467, 2007.

[28] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[29] S. Russell and P. Norvig. *Artificial Intelligence: a modern approach (3rd Edition)*. Prentice Hall, 2001.

[30] S. Sanner and C. Boutilier. Practical solution techniques for first order MDPs. *Artificial Intelligence*, 173:748–788, 2009.

[31] M. Schaefer. Graph Ramsey theory and the polynomial hierarchy. *Journal of Computers and System Sciences*, 62(2):290–322, 2001.

[32] M. Sipser. *Introduction to the Theory of Computation*. Thomson South-Western, 3rd edition, 2012.

[33] C. Wang, S. Joshi, and R. Khardon. First order decision diagrams for relational MDPs. *Journal of Artificial Intelligence Research*, 31:431–472, 2008.