The Complexity of Reasoning with FODD and GFODD

Benjamin J. Hescott and **Roni Khardon**
Department of Computer Science, Tufts University
Medford, MA, USA
{hescott|roni}@cs.tufts.edu

Abstract

Recent work introduced Generalized First Order Decision Diagrams (GFODD) as a knowledge representation 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. In this paper, we study the complexity of 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. 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^p_k complete, and equivalence is Π^p_{k+1} complete. For Π_k formulas evaluation is Π^p_k complete, satisfiability is one level higher and is Σ^p_{k+1} complete, and equivalence is Π^p_{k+1} complete.

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 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 single binary predicate and equality (Grädel 2003). Unfortunately, the problem is also undecidable if we restrict attention to satisfiability under finite structures (Fagin 1993; Libkin 2004). Thus, in either case, more refined notions of complexity are not appropriate. 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 the intended applications in AI and it renders the problems decidable allowing for a more detailed complexity analysis.

This paper is motivated by recent work on decision diagrams, known as FODDs and GFODDs (Wang, Joshi, and Khardon 2008; Joshi, Kersting, and Khardon 2011), and the computational questions associated with them. Propositional decision diagrams (Bryant 1986; Bahar et al. 1993) is a successful knowledge representation that captures functions over propositional variables and allows for efficient manipulation and composition of functions. Decision diagrams have been used in various applications in program verification and AI (Bryant 1986; Bahar et al. 1993; Hoey et al. 1999). Motivated by this success, several authors have attempted generalizations to handle relational structure and first order quantification (Groote and Tveretina 2003; Wang, Joshi, and Khardon 2008; Sanner and Boutilier 2009; Joshi, Kersting, and Khardon 2011). The main motivating application has been decision theoretic planning in relational domains (Boutilier, Reiter, and Price 2001; Kersting, Otterlo, and De Raedt 2004; Hölldobler, Karabaev, and Skvortsova 2006; Hölldobler and Skvortsova 2004) where implementations of FODDs and GFODDs have been shown to be useful in supporting symbolic solutions (Joshi, Kersting, and Khardon 2010; Joshi and Khardon 2011).

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 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" (Khardon and Roth 1996; 1997; Khardon, Mannila, and Roth 1999). The paper formalizes the evaluation, satisfiability, and equivalence problems for such diagrams and analyses their complexity. To avoid the 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, is more complex: 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 membership proofs in these results allow for constants in the signature but the hardness results, except for satisfiability for Π_1 formulas, hold 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 (Joshi, Kersting, and Khardon 2010; Joshi and Khardon 2011) and can be used to partly motivate or justify the use of these heuristics. For first order logic, the results give an interesting new avenue for efficient inference when the number of objects is bounded. These issues and further questions for future work are discussed in the concluding section of the paper.

Due to space constraints a significant amount of detail has been left out of proofs and constructions. These are available in the full version of this paper.

Complexity Theory Preliminaries

We assume basic familiarity with complexity theory including the classes P, NP, and co-NP (Homer and Selman 2001; Sipser 2012; Papadimitriou 1994). The polynomial hierarchy is defined from these inductively starting with Σ_1^p =NP, 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 in 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 either of these can serve as the oracle in the definition.

Generalized FODD

We assume familiarity with basic concepts and notation in predicate logic (e.g., (Lloyd 1987; Russell and Norvig 1995; Chang and Keisler 1990)). Decision diagrams are similar to expressions in first order logic. 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, a function with arity ≥ 1). In addition, we assume that the arity of predicates is bounded by some numerical constant. A term is a variable or a 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 is an object in the world of interest and an atom is a property which is either true or false.

First order decision diagrams (FODD) and their generalization (GFODD) were defined by (Wang, Joshi, and Khardon 2008; Joshi, Kersting, and Khardon 2011) inspired by previous work in (Groote and Tveretina 2003). A first order decision diagram 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. In the figures depicting GFODD in this paper, the left-going child denotes the true branch and the right child denotes the false branch. In addition, edge direction in the figures is always from top to bottom. 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. In addition, a GFODD has a list of aggregation operators that are defined below.

Similar to the propositional case (Bryant 1986; Bahar et al. 1993), 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 FODDs and GFODDs, a normal form has not been established, but the use of ordering makes for more efficient simplification of diagrams. In particular, following (Wang, Joshi, and Khardon 2008), 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 $(x_{k_1}, \ldots, x_{k_a}) \prec (x_{k'_1}, \ldots, x_{k'_a})$. 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 $a \prec b$. We assume that all diagrams are legally ordered in this way.

The ordering assumption is helpful in that it simplifies the computations. We note, however, that for this paper the assumption makes for more complex analysis because we must show that hardness results hold even for this restricted case.

The semantics of GFODDs assigns a value $\text{MAP}_B(I)$ for any diagram B and interpretation I, where I is a "possible world" specifying a domain of objects and the truth values of predicates over these objects. The "multiple-paths" semantics defines this value 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 sets 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 $\text{MAP}_B(I, \zeta)$.

Since different ζ will generate different values

$\text{MAP}_B(I,\zeta)$ we need to aggregate these to yield a single value for I. FODDs are defined by using $\max$ aggregation, that is, $\text{MAP}_B(I) = \max_\zeta \text{MAP}_B(I,\zeta)$.

GFODDS allow for other forms of aggregation of the values over $\text{MAP}_B(I,\zeta)$. In particular, let the variables in B be given in some arbitrary order $w_{i_1},\ldots,w_{i_m}$. Then B is associated with another list of length m specifying aggregation over each w_{i_j} in that order. The definition by (Joshi, Kersting, and Khardon 2011) allows for various aggregation operators (for example, min, max, sum, average) but here we use a more restricted set allowing each variable to be associated with either $\max$ or $\min$ aggregation. The semantics is defined as follows. First we calculate $\text{MAP}_B(I,\zeta)$ for all ζ. Then we loop with j taking values from m to 1 aggregating values over w_{i_j} using its aggregation operator.

When leaves are in $\{0,1\}$ $\max$ and $\min$ aggregation correspond to existential and universal quantifiers in logic with the formula given in quantified normal form, that is, with all quantifiers at the front. For example, the expression $[\max_{w_1}\min_{w_2}$ if $p(w_1,w_2)$ then 1 else $0]$, that can be captured with a diagram with one internal node, is the same as the logical formula $\exists w_1, \forall w_2, p(w_1,w_2)$.

We say that a GFODD is $\max$-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.

The following result, mirroring the case in logic, will allow us to simplify some of the arguments by borrowing results from their "complements". 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 the max leaf value in B be of value $\leq M$. 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. The following result can be shown by induction over aggregation steps:

Theorem 1 *Let B be a GFODD with min and max aggregation and maximum leaf value $\leq M$, and let $B' = complement(B)$. For any interpretation I, $MAP_B(I) = M - MAP_{B'}(I)$.*

For diagrams with binary leaves this yields $\text{MAP}_B(I) = 1 - \text{MAP}_{B'}(I)$, i.e., negation. Since diagrams A, B satisfy $A = B$ iff $\text{complement}(A) = \text{complement}(B)$ this shows that the complexity of equivalence of $\max$-k-alternating GFODDs is the same as that of $\min$-k-alternating GFODDs.

Computational Problems

As mentioned in the introduction, to avoid undecidability we restrict the computational problems so that the size of interpretations is given as part of the input. 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 GFODD implementations (Joshi, Kersting, and Khardon 2010; Joshi and Khardon 2011; Joshi et al. 2013).

We can now define the computational problems of interest. The simplest problem requires us to evaluate a diagram on a given interpretation.

Definition 2 (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.*

To calculate $\text{MAP}_B(I)$ we can appeal to a GFODD Evaluation oracle multiple times, once for each leaf value as V, and return the highest achievable result. The second problem most naturally applies for diagrams with binary leaves.

Definition 3 (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.*

Finally, diagrams allow for redundancies in the representation. It is therefore crucial for applications of FODDs and GFODDs that diagrams can be compressed into a form which is equivalent semantically but is smaller syntactically. One way to view such transformations, which has been productive in practice (Wang, Joshi, and Khardon 2008; Joshi, Kersting, and Khardon 2011), is to check whether an edge can be "removed" in the sense that instead of directing it to a sub-diagram we can direct it to a zero leaf and in this way potentially eliminate the sub-diagram rooted at the child. To perform this check we can produce the alternative diagram and test whether they are semantically equivalent. This is abstracted here as a comparison between arbitrary diagrams. Our hardness results, though, apply even to the special case of testing removal of a single edge.

Definition 4 (GFODD Equivalence) *Given diagrams B_1 and B_2 and integer N in unary: return Yes iff for all I with at most N objects, $MAP_{B_1}(I) = MAP_{B_2}(I)$.*

The assumption that N is in unary is convenient but not essential as our constructions will involve interpretations where the number of objects is linear in the size of B (bounded by the number of variables in B).

Complexity Results

Theorem 5 *GFODD Evaluation for $\max$-k-alternating GFODDs is Σ^p_k-complete. GFODD Evaluation for $\min$-k-alternating GFODDs is Π^p_k-complete.*

Proof Sketch. We prove membership by induction on k. Consider the input (B,I,V) for a max-k-alternating diagram B. We guess a tuple of objects i to bind the first block

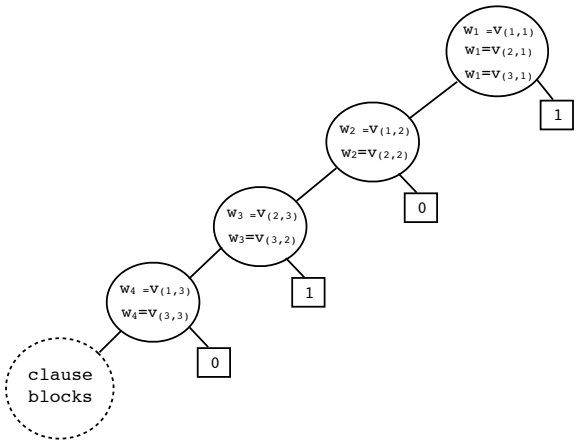

Figure 1: Example of variable consistency blocks for reduction from QBF to GFODD Evaluation for formula $\forall x_1 \exists x_2 \forall x_3 \exists x_4 (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee \neg x_4)$.

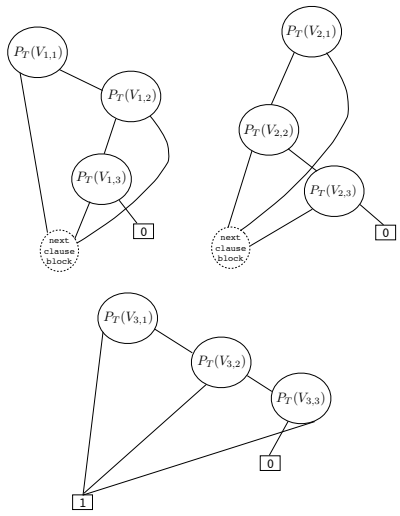

Figure 2: The clause blocks.

of max variables, substitute it into B to yield B' (removing the first block of variables), appeal to a Π^p_{k-1} oracle to solve GFODD Evaluation on (B', I, V), and return the same answer. This yields a correct algorithm in $\text{NP}^{\Sigma^p_{k-1}}$. The argument for the other aggregation prefix is symmetric.

To show hardness we give a reduction from QBF. Given a quantified 3CNF boolean formula we transform this into a GFODD B and interpretation I^* with two objects, and where truth value for the only predicate is $P_T(a) =$ true, and $P_T(b) =$ false. The construction guarantees that B evaluates to a value of 1 in I^* if and only if the quantified boolean formula is satisfied, therefore implying the theorem.

To construct B we create a GFODD variable $v_{(i,j)}$ corresponding to the jth literal of the ith clause in the QBF. In addition, for each QBF variable x_i we create a "shadow variable" w_i. We group w_i and the set of $v_{(i_1,i_2)}$ that refer to x_i or $\neg x_i$ in the QBF into the set $\mathbf{w}_i$. The set $\mathbf{w}_i$ has aggregation corresponding to the quantification of x_i (max for $\exists$ and min for $\forall$). Using these variables, we build FODD fragments we call variable consistency blocks. 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 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 $\forall x_1 \exists x_2 \forall x_3 \exists x_4 (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee \neg x_4)$. Figure 1 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 leaf value is 1. The consistency blocks are chained in the order of the quantification of the QBF. This chaining order guarantees that any unintended valuation (that does not assign a block $\mathbf{w}_i$ consistently) exits on the first block from the left which is violated, and is assigned the default value for this block.

Once every consistency has been checked we continue to the clause blocks (see Figure 2) that mimic the structure of the QBF clauses. This yields the diagram B where we set the aggregation function to be $Q^A_1 \mathbf{w}_1, Q^A_2 \mathbf{w}_2, \ldots, Q^A_m \mathbf{w}_m$ where Q^A_i and $\mathbf{w}_i$ are as above. The detailed proof uses the block ordering, and induction over aggregation order from m to 1, to show that unintended valuations do not affect $\text{MAP}_B(I)$ and that this implies the claim above. □

In the proof above, one might be tempted to use just the clause blocks and explicitly unify the GFODD variables that correspond to the same literal thus avoiding the need for variable consistency blocks. Such a strategy works for the corresponding proof for first order logic. This is not possible for GFODDs, however, because unifying the variables will violate ordering constraints. If we keep the variables unified and try to reorder nodes in the diagram the result might grow exponentially in size. The variable consistency blocks allow us to get around this problem.

Theorem 6 *GFODD Satisfiability for* $\max$*-k-alternating GFODDs is Σ^p_k-complete.*

Proof Sketch. For membership we guess I (due to the bound on arity, this requires polynomial space in the number of objects) and the binding of the first max block, substitute it into the input diagram, and appeal to an oracle for GFODD Evaluation for the resulting depth $k-1$ diagram. The hardness argument is similar to the proof for GFODD evaluation but instead of specifying I^* in the reduction we add a gadget to B that forces any satisfying interpretation to be isomorphic to I^*. Since I^* exists, we get that B is satisfiable iff the QBF is true. □

Theorem 7 *GFODD Equivalence for diagrams with aggregation depth k is Π^p_{k+1}-complete.*

Proof Sketch. As noted above from the properties of complement(B), it suffices to show that the theorem

holds for max-k-alternating GFODDs. For membership we show that non-equivalence 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 use multiple calls 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. Clearly if a witness for non-equivalence 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 $\text{NP}^{\Sigma^p_k}$.

Hardness requires a significant extension of the proof technique from above. In particular, we start with a Π_k QBF, and "peel off" the first block of quantifiers, yielding diagrams in max-$(k-1)$-alternating GFODD for an equivalence test. To simplify the notation it is convenient to group adjacent variables having the same quantifiers into groups so that the QBF has the form $\forall \mathbf{x}_1, Q_2\mathbf{x}_2 \ldots Q_k\mathbf{x}_k f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$ where $\mathbf{x}_i$ refers to a set of variables. We define a notion of "legal interpretation" which must embed 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 $B = B_1 \wedge B_2$ (that can be computed without increasing quantifier depth; see Theorem 4 of (Joshi, Kersting, and Khardon 2011)) such that: (C1) for all I, $\text{MAP}_{B_1}(I) = 1$ if and only if I is legal, and (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, assuming that the $\wedge$ operation is correct, 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 also $\text{MAP}_B(I) = 1$. Thus B and B_1 are equivalent. The other direction (when the QBF is not satisfied) is argued in a similar manner.

To describe B_1, B_2 we start with a simplified construction and then fix some of the details. The set of predicates includes $P_T()$ which is as before 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 $P_T(a) \neq P_T(b)$ (as in I^*) and in addition for each $P_{x_i}()$ we have $P_{x_i}(a) = P_{x_i}(b)$. That is, as above, 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, $\forall x_1 \exists x_2 \forall x_3 \exists x_4 (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee \neg x_4)$, the first block includes only the variable x_1 and the following interpretation is legal: $I = \{[a, b], P_T(a) = \text{true}, P_T(b) = \text{false}, P_{x_1}(a) = P_{x_1}(b) = \text{false}\}$. The diagram B_1 has three portions. The first verifies that I has two objects, the second verifies that $P_T()$ behaves as stated, and

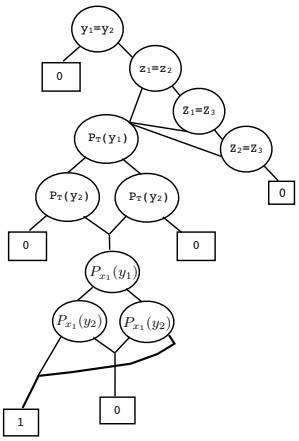

Figure 3: B_1 diagram for Equivalence reduction.

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 3 and the aggregation function is $\max_{y_1,y_2}, \min_{z_1,z_2,z_3}$.

The diagram B_2 is similar to B from the Evaluation proof and includes variable consistency blocks followed by the clause blocks. The only difference is that we need 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 max aggregation and accordingly replace the default value on that block to 0. In addition, the clause blocks in this case 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 and use $P_T(V_{(i_1,i_2)})$ otherwise. The detailed proof shows that C1 and C2 hold and that this implies the theorem.

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 in B_2 determined by the appearance of literals in the QBF. To fix this we simulate 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. For example, $P_T(v_{1,2})$ will be replaced with $q(v_{1,2}, T)$. In addition we force $q()$ to be symmetric so that for any X and Y the truth value of $q(X, Y)$ is the same as the truth value of $q(Y, X)$. In this way we have freedom to use either $q(X, Y)$ or $q(Y, X)$ as the node label which provides sufficient flexibility to handle the order issues. The reformulation of the clause blocks in our example is shown in Figure 4. To implement this idea the full proof uses gadgets to identify the object T and the objects x_i from the first block, to make sure that they are unique and to make sure that bindings are consistent.

This proof holds for k'-alternating GFODDs where $k' \geq 2$. We therefore require a separate proof for equivalence of FODDs where $k' = 1$. The full paper provides a reduction from the problem of deciding arrowing from the Ramsey theory of graphs (Schaefer 2001). □

Theorem 8 *GFODD Satisfiability for* min-k*-alternating GFODDs (where* $k \geq 2$*) is* Σ^p_{k+1}*-complete.*

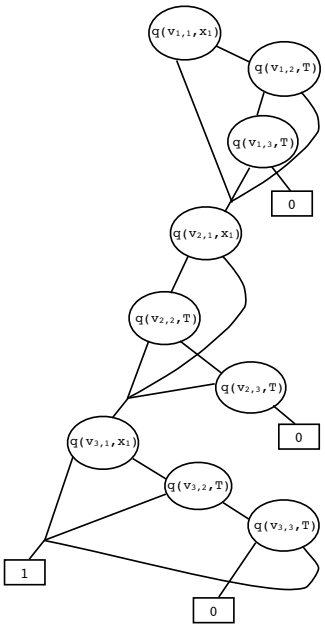

Figure 4: Sorted clause blocks using binary predicate.

Proof Sketch. For membership we guess I and appeal to a depth k GFODD Evaluation oracle. The hardness result adapts the construction of the equivalence proof. Here we start with a Σ_k formula, and "peeling off" one block of quantifiers get a $\min$-$(k-1)$-alternating GFODD. We output the diagram corresponding to B for satisfiability. Properties C1 and C2 remain exactly as in the previous proof, but because of the shift in quantifier ordering we get hardness for satisfiability, i.e., in a single model, rather than the requirement to be satisfied in all legal models in the previous proof. □

All the hardness proofs given above 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 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 $\text{MAP}_B(I)$ in polynomial time. On the other hand, if we allow constants in the signature we can show that the problem follows the same scheme as above:

Theorem 9 *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. Although turning these variables to constants is similar to pulling the $\max$ portion to the head of the formula one can verify that the arguments above regarding valuations and values hold yielding the same hardness result. □

Conclusions

We explored the complexity of computations using FODD and GFODD, providing a classification placing these 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 clearly characterize the complexity of the problems solved heuristically by implementations of GFODD systems (Joshi, Kersting, and Khardon 2010; Joshi and Khardon 2011) and can be used to partly motivate or justify the use of these heuristics. For example, the "model checking reductions" of (Joshi, Kersting, and Khardon 2010) replace equivalence tests with model evaluation on a "representative" set of models, and (Joshi, Kersting, and Khardon 2010) choose this set heuristically leading to inference that is correct with respect to these models but otherwise incomplete. Our results show that this indeed leads to a reduction in complexity so that the compromise in correctness is traded for improved worst case run time. As mentioned in the introduction, the proofs in the paper can be used (with simplified details) to show the same complexity results for the corresponding problems in first order logic. To our knowledge the complexity questions with an explicit bound on model size have not been previously studied in this context. Yet they can be useful where such a bound can be given in a practical setting. In such cases existing optimized QBF algorithms can be used for inference in this restricted form of first order logic.

There are two 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 GFODDs. The work of (Joshi et al. 2013) 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. Related to this, the second question is to identify efficient model evaluation algorithms for GFODDs. The work of (Joshi et al. 2013) 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 tree width" and the same is true for the variable elimination algorithm. 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 RK was on sabbatical leave at Trinity College Dublin.

References

Bahar, R.; Frohm, E.; Gaona, C.; Hachtel, G.; Macii, E.; Pardo, A.; and Somenzi, F. 1993. Algebraic decision diagrams and their applications. In *IEEE /ACM ICCAD*, 188–191.

Boutilier, C.; Reiter, R.; and Price, B. 2001. Symbolic dynamic programming for first-order MDPs. In *Proc. of IJCAI*, 690–700.

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

Chang, C., and Keisler, J. 1990. *Model Theory*. Amsterdam, Holland: Elsevier.

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

Grädel, E. 2003. 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*.

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

Hoey, J.; St-Aubin, R.; Hu, A.; and Boutilier, C. 1999. Spudd: Stochastic planning using decision diagrams. In *Proceedings of UAI*, 279–288.

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

Hölldobler, S.; Karabaev, E.; and Skvortsova, O. 2006. FluCaP: a heuristic search planner for first-order MDPs. *JAIR* 27:419–439.

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

Joshi, S., and Khardon, R. 2011. Probabilistic relational planning with first order decision diagrams. *JAIR* 231–266.

Joshi, S.; Khardon, R.; Raghavan, A.; Tadepalli, P.; and Fern, A. 2013. Solving relational mdps with exogenous events and additive rewards. In *ECML*.

Joshi, S.; Kersting, K.; and Khardon, R. 2010. Self-taught decision theoretic planning with first order decision diagrams. In *Proc. of ICAPS*, 89–96.

Joshi, S.; Kersting, K.; and Khardon, R. 2011. Decision theoretic planning with generalized first order decision diagrams. *AIJ* 175:2198–2222.

Kersting, K.; Otterlo, M. V.; and De Raedt, L. 2004. Bellman goes relational. In *Proc. of ICML*.

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

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

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

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

Lloyd, J. 1987. *Foundations of Logic Programming*. Springer Verlag. Second Edition.

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

Russell, S., and Norvig, P. 1995. *Artificial Intelligence: a modern approach*. Prentice Hall.

Sanner, S., and Boutilier, C. 2009. Practical solution techniques for first order MDPs. *AIJ* 173:748–788.

Schaefer, M. 2001. Graph ramsey theory and the polynomial hierarchy. *J. Comput. Syst. Sci.* 62(2):290–322.

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

Wang, C.; Joshi, S.; and Khardon, R. 2008. First order decision diagrams for relational MDPs. *JAIR* 31:431–472.