On the Fixed-Parameter Tractability of the Equivalence Test of Monotone Normal Forms

Matthias Hagen [1]

Friedrich-Schiller-Universität Jena, Institut für Informatik, D–07737 Jena

Abstract

We consider the problem Monet—given two monotone formulas φ in DNF and ψ in CNF, decide whether they are equivalent. While Monet is probably not coNP-hard, it is a long standing open question whether it has a polynomial time algorithm and thus belongs to P. In this paper we examine the parameterized complexity of Monet. We show that Monet is in FPT by giving fixed-parameter algorithms for different parameters.

Key words: Analysis of algorithms, Computational complexity, Equivalence test, Fixed-parameter tractability, Monotone normal forms

1 Introduction

The problem Monet—Mo(notone) n(ormal form) e(quivalence) t(est)—asks for the equivalence of two monotone formulas φ in DNF and ψ in CNF. Algorithms solving the computational variant Monet′—given a monotone DNF, compute the equivalent CNF—can be easily transformed to solve Monet and vice versa. Monet and Monet′ are equivalent in the sense of solvability in appropriate terms of polynomial time [1]. Furthermore, Monet′ is polynomially equivalent to the fundamental problems of dualizing monotone CNFs and the transversal hypergraph generation. Hence, Monet and Monet′ have many applications in such different fields like artificial intelligence and logic [6,7], computational biology [3], database theory [18], data mining and machine learning [12], mobile communication systems [22], distributed systems [10], and graph theory [13,16]. The currently best known algorithms for Monet

Email address: hagen@cs.uni-jena.de (Matthias Hagen).

[1] Supported in part by a Landesgraduiertenstipendium Thüringen.

run in quasi-polynomial time or use $O(\log^2 n)$ nondeterministic bits [8,9,14]. Thus, on the one hand, Monet is probably not coNP-complete, but on the other hand a polynomial time algorithm is not yet known. Actually there are polynomial time algorithms for many special classes—e.g. when φ is a k-DNF, 2-monotonic, μ-equivalent, or acyclic [2,5,6]—but the complexity of the general problem Monet is a long standing open question [20].

In this paper we analyze the parameterized complexity of Monet. We show that Monet is in FPT for the parameters number v of variables in φ and ψ, number m of monomials in φ, and a parameter q describing the variable frequencies in φ.

2 Preliminaries

Two Boolean formulas are *equivalent* if they have the same truth table. *Monotone formulas* are Boolean formulas with $\wedge$ and $\vee$ as only connectives. No negation signs are allowed. A monomial (resp. clause) is the conjunction (disjunction) of variables. We often refer to monomials or clauses simply as *terms*. A monotone *DNF* (resp. *CNF*) is the disjunction (conjunction) of monomials (clauses). A monotone normal form ϱ is said to be *irredundant* if there are no two terms in ϱ such that one is contained in the other. Since the irredundant DNF and CNF of monotone formulas are unique [21] and can be obtained from respective redundant normal forms in quadratic time, we only concentrate on irredundant inputs yielding the following formal definition.

Monet:	*instance:*	irredundant, monotone formulas φ in DNF and ψ in CNF with variable set V
	question:	are φ and ψ equivalent?

Note that irredundant formulas with different sets of variables cannot be equivalent. As this can be tested in quadratic time, we assume that φ and ψ contain the same variables. The *size* of the Monet-instance (φ, ψ) is the number of variable occurrences in φ and ψ. An *assignment* for φ and ψ is a subset $\mathcal{A} \subseteq V$. Thereby, the notion is that variable x is set to true iff $x \in \mathcal{A}$. This means that the powerset $\mathcal{P}(V)$ can also be seen as the set of all assignments for φ and ψ. In the same way we consider the monomials of φ and the clauses of ψ to be sets of variables. Hence, they can also be viewed as assignments.

In this paper we analyze versions of Monet that have some parameters as input in addition to the DNF φ and the CNF ψ. Briefly, a parameterized problem with parameter k is *fixed-parameter tractable* if it can be solved by an algorithm running in time $O(f(k) \cdot poly(n))$, where f is a function depending on k only, n is the size of the input, and $poly(n)$ is any polynomial in n. The class FPT contains all fixed-parameter tractable problems. For a more general survey on fixed-parameter tractability we refer to the monograph of Niedermeier [19].

3 Results

3.1 Number v of Variables as Parameter

A first super-naïve fixed-parameter tractability result for the number v of variables is at hand by simply checking all of the possible 2^v assignments for an instance (φ, ψ) of size n. This yields an $O(2^v \cdot n)$-time algorithm for MONET. To considerably improve this time bound, we use the notion of the maximum latency introduced by Makino and Ibaraki [17].

For a monotone formula ϱ we denote by $T(\varrho)$ (resp. $F(\varrho)$) the set of assignments that satisfy ϱ (do not satisfy ϱ). We say that a MONET-instance (φ, ψ) is *well-formed* if φ and ψ are not empty but $T(\varphi) \cap F(\psi)$ is. Testing whether a given MONET-instance is well-formed can be accomplished in polynomial time. The first condition is obviously trivial and the second is equivalent to testing the validity of $\varphi \to \psi$, which is an easy quadratic time procedure [4].

Definition 3.1 (Maximum Latency) *Let (φ, ψ) be a well-formed* MONET-*instance. By U we denote the set of assignments that are neither in $T(\varphi)$ nor in $F(\psi)$, i.e., $U = F(\varphi) \cap T(\psi)$. The latency of (φ, ψ) is defined as*

$$\lambda(\varphi, \psi) = \min\{|\mathcal{A}_U \triangle t| : \mathcal{A}_U \in U,\ t \text{ is a term of } \varphi \text{ or } \psi\},$$

where $\triangle$ denotes the symmetric difference. For well-formed MONET-*instances with v variables the maximum latency is defined as*

$$\Lambda(v) = \max\{\, \lambda(\varphi, \psi) \;:\; \varphi \text{ and } \psi \text{ have } v \text{ variables}\}.$$

Makino and Ibaraki proved the following tight bound on the maximum latency.

Proposition 3.2 ([17]) $\Lambda(v) = \lfloor v/4 \rfloor + 1$.

Finding an assignment $\mathcal{A}_U \in U$ that does not satisfy φ but satisfies ψ is equivalent to prove $(\varphi, \psi) \notin$ MONET. Hence, with Proposition 3.2 a well-formed MONET-instance (φ, ψ) can be tested for equivalence by checking all the assignments that differ in at most $\lfloor v/4 \rfloor + 1$ variables from any term of φ and ψ. We will use this idea in an algorithm that has a better running time than the first super-naïve approach. For the analysis we will need the following combinatorial observation.

Lemma 3.3 *Let $0 < \varepsilon < \frac{1}{2}$. Then we have*

$$\sum_{i=1}^{\lfloor \varepsilon k \rfloor} \binom{k}{i} \in O\left(\left[\left(\frac{1}{\varepsilon}\right)^{\varepsilon} \cdot \left(\frac{1}{1-\varepsilon}\right)^{1-\varepsilon} \right]^k \cdot \frac{1}{\sqrt{k}} \right).$$

PROOF. It is well-known (see, e. g., [11]) that asymptotically $\sum_{i=0}^{\lfloor \varepsilon k \rfloor} \binom{k}{i} = 2^{k \cdot h(\varepsilon) - \frac{1}{2}\log k + O(1)}$, where $h(\varepsilon) = -\varepsilon \log \varepsilon - (1-\varepsilon)\log(1-\varepsilon)$ is the entropy function. Expanding the asymptotic equation yields the lemma. □

Theorem 3.4 *Let (φ, ψ) be a* Monet*-instance of size n having v variables. Then $(\varphi, \psi) \in$* Monet *can be decided in time*

$$O\left(\left[\left(\frac{1}{\frac{1}{4}+\frac{1}{v}}\right)^{\frac{1}{4}+\frac{1}{v}} \left(\frac{1}{\frac{3}{4}-\frac{1}{v}}\right)^{\frac{3}{4}-\frac{1}{v}} \right]^v \cdot \frac{1}{\sqrt{v}} \cdot n^2 \right).$$

PROOF. If the instance is not well-formed, it is rejected in quadratic time. Otherwise, for $v \leq 4$, we check all the at most 16 assignments in a brute-force manner in constant time.

For $v \geq 5$, we check all assignments that differ from any term of φ or ψ in at most $\Lambda(v)$ variables. This means that we check the $(\lfloor v/4 \rfloor + 1)$-neighborhoods of the terms of φ and ψ, which suffices as follows from Proposition 3.2. For each such assignment $\mathcal{A}$ we test in $O(n)$ time whether φ and ψ get the same value. There are at most n terms in φ and ψ. Hence, the running time of an algorithm checking all the necessary assignments can be bounded by $O(s \cdot n^2)$, where s denotes the number of assignments in a $\Lambda(v)$-neighborhood. We have $s = \sum_{i=1}^{\lfloor \varepsilon v \rfloor} \binom{v}{i}$, where $\varepsilon = \frac{1}{4} + \frac{1}{v}$. For $v \geq 5$ we have $\varepsilon < \frac{1}{2}$. Hence, the estimation of Lemma 3.3 can be applied and the theorem follows. □

Table 1 contains the running time stated in Theorem 3.4 for special v in a more readable format. Note that an estimation for $v \to \infty$ yields a lower bound of

$$\Omega(1.7547^v \cdot \frac{1}{\sqrt{v}} \cdot n^2).$$

v		running time
$\geq$	5	$O(1.991^v \cdot \frac{1}{\sqrt{v}} \cdot n^2)$
$\geq$	10	$O(1.911^v \cdot \frac{1}{\sqrt{v}} \cdot n^2)$
$\geq$	20	$O(1.843^v \cdot \frac{1}{\sqrt{v}} \cdot n^2)$
$\geq$	50	$O(1.792^v \cdot \frac{1}{\sqrt{v}} \cdot n^2)$
$\geq$	100	$O(1.774^v \cdot \frac{1}{\sqrt{v}} \cdot n^2)$
$\geq$	1000	$O(1.757^v \cdot \frac{1}{\sqrt{v}} \cdot n^2)$

Table 1
Running time from Theorem 3.4 for special values of v

3.2 Number m of Monomials as Parameter

We show that Monet is fixed-parameter tractable with the number m of monomials in φ as parameter.

Theorem 3.5 *Let (φ, ψ) be a* Monet*-instance of size n with m monomials in φ. Then $(\varphi, \psi) \in$* Monet *can be decided in time $O(2^{m(m-\log m+4)}m^3 + n^2)$.*

PROOF. Note that m monomials can split the set of variables into at most 2^m classes of variables that appear in exactly the same monomials (actually there are at most $2^m - 1$ classes but this would only complicate the below estimations). Choosing representatives for each class and replacing variables by them yields a modified DNF φ' with m monomials and at most 2^m variables. Hence, the irredundant, equivalent CNF ψ' of φ' cannot have clauses that contain more than m variables. We compute ψ' by adapting the KS-algorithm of Kavvadias and Stavropoulos [15] for hypergraph transversal generation. The main idea of the KS-algorithm is to process a depth-first search in a search tree that is built as follows. The root of the tree corresponds to a monomial of φ'. If the subset of variables on the path from the root to the current node (this subset forms a clause candidate) does not intersect all monomials of φ', the KS-algorithm expands it by picking a monomial that is not yet intersected and generating edges for each so-called *appropriate* variable in this monomial. Briefly, a variable is appropriate if adding it to the current candidate set does not result in a set where another variable could be left out and still all monomials except the last one are intersected. Checking a monomial for appropriate vertices can be done in time $O((2^m \cdot m)^2)$ since a monomial contains at most 2^m variables, the current clause candidate set has size at most m, and the size of φ' is bounded by $2^m \cdot m$. Expanding only by appropriate variables ensures that generated clauses are minimal and that no repetitions occur [15].

If all monomials are covered, the KS-algorithm puts out the clause and starts backtracking. In the worst case the search tree that is traversed by the KS-algorithm contains a node corresponding to each variable subset of size at most m (written on the paths from the root to the nodes). There are $\sum_{i=0}^{m} \binom{2^m}{i} \leq m \cdot \binom{2^m}{m}$ such subsets. Using Stirling's formula and the fact that $e < 4$ we get $\binom{2^m}{m} \leq 2^{m(m-\log m+2)}$. Since in the worst case for each node the appropriate variables have to be determined, the KS-algorithm needs $O(2^{m(m-\log m+4)}m^3)$ time to compute the CNF ψ'.

From ψ' we compute a CNF ψ'' without representatives. This is done by systematically processing the representatives one after the other. Let y be the currently processed representative that stands for the variables $x_{i_1}, \ldots, x_{i_k}$. For each occurrence of y the respective clause is copied k times and in the j-th copy we replace y by x_{i_j}. Since ψ' is irredundant, all the intermediate results of the computation and ψ'' are irredundant. Thus, we can immediately reject whenever an intermediate result gets larger than ψ. Hence, the time needed to compute ψ'' is $O(n)$ as n is an upper bound on the size of ψ. When there is no representative left we have to check whether ψ'' and ψ are identical. This can be accomplished in time $O(n^2)$. □

Note that we have the same result with the number of clauses of ψ as parameter since we could simply exchange the roles of DNF and CNF.

3.3 Variable Frequencies as Parameter

For a MONET instance (φ, ψ) we denote by q the largest number of monomials over all variables x that do not include x, i.e.,

$$q = \max_{x \in V} \{ |\{\mu : x \notin \mu, \text{ where } \mu \text{ is a monomial of } \varphi\}| \}.$$

We show that MONET is fixed-parameter tractable with q as parameter.

Theorem 3.6 *Let (φ, ψ) be a MONET-instance of size n and q as defined above. Then $(\varphi, \psi) \in$ MONET can be decided in time $O(2^{q(q-\log q+4)}q^3 n + n^3)$.*

PROOF. For an irredundant monotone normal form ϱ we denote by $\varrho^{x=0}$ (resp. $\varrho^{x=1}$) the normal form that is obtained by setting x to `false` (`true`) and removing redundant terms. These irredundant forms $\varrho^{x=0}$ and $\varrho^{x=1}$ can be easily computed in quadratic time.

Note that testing $(\varphi, \psi) \in$ MONET is equivalent to testing $(\varphi^{x=0}, \psi^{x=0}) \in$ MONET and $(\varphi^{x=1}, \psi^{x=1}) \in$ MONET for any variable x from φ. Our fixed-parameter algorithm exactly processes these tests. Note that setting $x = 0$ yields a DNF $\varphi^{x=0}$ with at most q monomials. Hence, we can apply Theorem 3.5 and decide $(\varphi^{x=0}, \psi^{x=0}) \in$ MONET in time $O(2^{q(q-\log q+4)}q^3 + n^2)$.

For the second test we recursively call the algorithm with $(\varphi^{x=1}, \psi^{x=1})$ as input. Note that $\varphi^{x=1}$ contains at most $n-1$ variables. If $\varphi^{x=1}$ does not contain any variable, the equivalence test is trivial. Hence, there are at most $n-1$ recursive calls which results in an overall running time of $O(2^{q(q-\log q+4)}q^3 n + n^3)$. $\square$

Note that again the roles of DNF and CNF may be exchanged to get the statement for variable frequencies in ψ as well. Furthermore, the algorithm runs in polynomial time if q is a constant. This yields a new polynomial time special case of MONET.

4 Conclusion

We have shown MONET to be fixed-parameter tractable for the parameters number v of variables in φ and ψ, number m of monomials in φ, and a parameter q describing the variable frequencies in φ. Obvious open questions are to further improve the running times. Especially interesting would be algorithms with running times that are subexponential in the parameters.

Furthermore of interest are parameterized results for other parameters like the size of a largest monomial.

Acknowledgments I thank Martin Mundhenk and the anonymous referee for their valuable comments and suggestions that helped to improve the paper.

References

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

[2] E. Boros, P. L. Hammer, T. Ibaraki, and K. Kawakami. Polynomial-time recognition of 2-monotonic positive Boolean functions given by an oracle. *SIAM Journal on Computing*, 26(1):93–109, 1997.

[3] P. Damaschke. Parameterized enumeration, transversals, and imperfect phylogeny reconstruction. *Theoretical Computer Science*, 351(3):337–350, 2006.

[4] J. P. Delgrande and A. Gupta. The complexity of minimum partial truth assignments and implication in negation-free formulae. *Annals of Mathematics and Artificial Intelligence*, 18(1):51–67, 1996.

[5] T. Eiter. Exact transversal hypergraphs and application to Boolean μ-functions. *Journal of Symbolic Computation*, 17(3):215–225, 1994.

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

[7] T. Eiter and G. Gottlob. Hypergraph transversal computation and related problems in logic and AI. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *Logics in Artificial Intelligence, European Conference, JELIA 2002, Cosenza, Italy, September, 23-26, Proceedings*, volume 2424 of *Lecture Notes in Computer Science*, pages 549–564. Springer, 2002.

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

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

[10] H. Garcia-Molina and D. Barbará. How to assign votes in a distributed system. *Journal of the ACM*, 32(4):841–860, 1985.

[11] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, 1994.

[12] D. Gunopulos, R. Khardon, H. Mannila, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12-14, 1997, Tucson, Arizona*, pages 209–216. ACM Press, 1997.

[13] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.

[14] D. J. Kavvadias and E. C. Stavropoulos. Monotone Boolean dualization is in $\mathsf{coNP}[\log^2 n]$. *Information Processing Letters*, 85(1):1–6, 2003.

[15] D. J. Kavvadias and E. C. Stavropoulos. An efficient algorithm for the transversal hypergraph generation. *Journal of Graph Algorithms and Applications*, 9(2):239–264, 2005.

[16] E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan. Generating all maximal independent sets: NP-hardness and polynomial-time algorithms. *SIAM Journal on Computing*, 9(3):558–565, 1980.

[17] K. Makino and T. Ibaraki. The maximum latency and identification of positive Boolean functions. *SIAM Journal on Computing*, 26(5):1363–1383, 1997.

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

[19] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006.

[20] C. H. Papadimitriou. NP-completeness: A retrospective. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, 7-11 July 1997, Proceedings*, volume 1256 of *Lecture Notes in Computer Science*, pages 2–6. Springer, 1997.

[21] W. Quine. Two theorems about truth functions. *Boletin de la Sociedad Matemática Mexicana*, 10:64–70, 1953.

[22] S. Sarkar and K. N. Sivarajan. Hypergraph models for cellular mobile communication systems. *IEEE Transactions on Vehicular Technology*, 47(2):460–471, 1998.