Generating Models of a Matched Formula With a Polynomial Delay (Extended Abstract)*

Petr Savický
Institute of Computer Science
The Czech Academy of Sciences
Czech Republic
savicky@cs.cas.cz

Petr Kučera
Department of Theoretical Computer Science
and Mathematical Logic
Faculty of Mathematics and Physics
Charles University, Czech Republic
kucerap@ktiml.mff.cuni.cz

Abstract

A matched formula is a CNF formula whose incidence graph admits a matching which matches a distinct variable to every clause. Such a formula is always satisfiable. Matched formulas are used, for example, in the area of parameterized complexity. We prove that the problem of counting the number of the models (satisfying assignments) of a matched formula is #P-complete. On the other hand, we define a class of formulas generalizing the matched formulas and prove that for a formula in this class one can choose in polynomial time a variable suitable for splitting the tree for the search of the models of the formula. As a consequence, the models of a formula from this class, in particular of any matched formula, can be generated sequentially with a delay polynomial in the size of the input. On the other hand, we prove that this task cannot be performed efficiently for linearly satisfiable formulas, which is a generalization of matched formulas containing the class considered above.

1 Introduction

In this paper, we consider the problem of counting the models (satisfying assignments) and generating subsets of the models of a given formula in conjunctive normal form (CNF). It is well known that #SAT, the problem of counting the models of a general CNF, is #P-complete. The problem of enumerating the models of a general CNF formula is clearly also hard, because checking whether there is at least one satisfying assignment of the formula, the SAT problem, is NP-complete.

In this paper, we mostly deal with the problem of enumerating models of a formula. This problem is important in areas of research and applications, such as unbounded model checking [Kang and Park, 2005; McMillan, 2002] or data mining [Coquery *et al.*, 2012]. The success of modern SAT solvers inspired design of model counting and enumeration algorithms as well, see e.g. [Jabbour *et al.*, 2014; Morgado and Marques-Silva, 2005]. In addition to the basic enumeration problem in which we do not require the models to be generated in any prescribed order, other versions have been considered, e.g. generating models by non-decreasing weight [Creignou *et al.*, 2011].

Another line of research concentrated on studying special classes of boolean formulas for which an enumeration algorithm with guaranteed complexity could be devised. One can easily find an example of a formula for which the set of models is exponentially larger than the size of the formula itself. In such a case it is reasonable to include the size of the output into the bound on the running time of an enumeration algorithm. More specifically we say that an algorithm which enumerates models of a formula runs in *output polynomial time* if its running time can be bounded by a polynomial in two variables, the size of the input (i.e. the input formula φ) and the size of the output (i.e. the number of models of φ). In this paper, we consider more restrictive setting as follows. The algorithm receives as input a formula and generates a sequence of all its models in such a way that the time needed for generating the first model and the time between generating any two consecutive models in the sequence is polynomial in the length of the formula. This type of complexity bound is called a polynomial delay. It should be clear that if we can enumerate models of a formula with a polynomial delay, then we can construct an output polynomial algorithm for this task as well. On the other hand, it can be much harder to get an enumeration algorithm with polynomial delay than an output polynomial algorithm. For an overview of various notions of enumeration complexity see [Johnson *et al.*, 1988].

There are special classes of formulas for which polynomial delay enumeration algorithms have been described, this includes 2-CNF formulas, Horn formulas, generalized satisfiability problems and others, see e.g. [Aceto *et al.*, 2013; Creignou and Hébrard, 1997; Dechter and Itai, 1992; Kavvadias *et al.*, 2000]. In this paper, we describe another class of formulas for which a polynomial delay enumeration algorithm based on backtrack-free search can be described. On the contrary to such algorithms known for 2-CNF or Horn formulas, the splitting variable in each step cannot be chosen arbitrarily, however, the existence of a suitable variable is guaranteed and it can be efficiently identified.

In particular we consider the class of matched formulas introduced by [Franco and Van Gelder, 2003]. Given a CNF formula φ, we consider its *incidence graph* $I(\varphi)$ defined as follows. $I(\varphi)$ is a bipartite graph with one part consisting

*This paper is an extended abstract of an article in the Journal of Artificial Intelligence Research [Savický and Kučera, 2016].

of clauses of φ and the other part containing the variables of φ. An edge $\{x, C\}$ for a variable x and a clause C is in $I(\varphi)$ if x or $\overline{x}$ appears in C. It was observed by Aharoni and Linial [1986] and Tovey [1984] that if $I(\varphi)$ admits a matching (i.e. a set of pairwise disjoint edges) of size m (where m is the number of clauses in φ), then φ is satisfiable. Later the formulas satisfying this condition were called *matched formulas* by Franco and Van Gelder. Since a matching of maximum size in a bipartite graph can be found in polynomial time, one can check efficiently whether a given formula is matched.

Given a general formula φ, we can measure how far it is from being matched by considering its maximum deficiency $\delta^*(\varphi)$, the number of clauses which remain unmatched in a maximum matching of $I(\varphi)$. A formula φ is thus matched iff $\delta^*(\varphi) = 0$. A weaker notion of deficiency $\delta(\varphi) = m - n$, where m is the number of clauses and n the number of the variables in φ, is also often being considered.

Matched formulas play a significant role in the theory of satisfiability solving. Since their introduction matched formulas have been considered as a base class in parameterized algorithms for satisfiability, see e.g. the book of [Flum and Grohe, 2006] for an overview of parameterized algorithms theory. In particular, [Fleischner *et al.*, 2002] show that satisfiability of formulas with maximum deficiency bounded by a constant k can be decided in time $O(\|\varphi\|n^{O(k)})$ where $\|\varphi\|$ is the length of the input formula φ and n denotes the number of its variables. This result was later improved by [Szeider, 2003] to an algorithm for satisfiability parameterized with maximum deficiency of a formula with complexity $O(2^k n^3)$. Parameterization based on backdoor sets with respect to matched formulas were considered by Szeider [2007].

Since all matched formulas are trivially satisfiable, we ask a stronger question: How hard is it to count or enumerate the models of a matched formula? We prove that counting the models of a matched formula is a #P-complete problem, and turn our attention to generating models of a matched formula. The main result of the paper is an algorithm which generates models of a matched formula with a polynomial delay. The algorithm starts by constructing a splitting tree whose nodes correspond to either a matched or an unsatisfiable formula. In some cases this strategy is not sufficient since some nodes of the tree cannot be split in this way. We prove that such a node corresponds to a formula which can be satisfied by iterated elimination of pure literals. Formulas with this property will be called pure literal satisfiable. These formulas were studied by [Kullmann, 2000] as a subclass of linearly satisfiable formulas. If a node with a pure literal satisfiable formula is reached, the algorithm switches to a simpler strategy. We prove that the models of a pure literal satisfiable formula can be generated with a delay linear in the length of the formula. On the other hand, the #SAT problem for pure literal satisfiable formulas is #P-complete, because this problem is #P-complete for monotone 2CNFs [Valiant, 1979], which are pure literal satisfiable.

Several generalizations of matched formulas have also been considered in the literature. Kullmann [2000] generalized matched formulas into the class of linearly satisfiable formulas. Autarkies based on matchings were studied by Kullmann [2003]. Szeider [2005] considered another generalization of matched formulas, the classes of biclique satisfiable and var-satisfiable formulas. Unfortunately, for both biclique satisfiable and var-satisfiable formulas it is hard to check if a formula falls into one of these classes.

We show in the paper that the algorithm for efficient generating models of a matched formula does not transfer to the class of linearly satisfiable formulas by demonstrating that it is not possible to generate models of a linearly satisfiable formula with a polynomial delay unless P=NP.

2 Definitions

In this section, we give the necessary definitions and summarize the results we use in the paper.

A *Boolean function on n variables* is a mapping $f : \{0,1\}^n \to \{0,1\}$. A *literal* is either a variable (e.g. x), called *positive literal*, or its negation (e.g. $\overline{x}$), called *negative literal*. A *clause* is a disjunction of a set of literals, which contains at most one literal for each variable. Formula φ is in *conjunctive normal form (CNF)* or, equivalently, φ is a CNF formula, if it is a conjuction of clauses. We shall exclusively work with CNF formulas, in particular by "formula" we always mean a "CNF formula". The size of a formula φ is the number of the clauses in φ and will be denoted as $|\varphi|$. The length of a formula φ is the total number of occurrences of literals in φ, i.e. the sum of the sizes of the clauses in φ, and will be denoted as $\|\varphi\|$. Given a variable x and a value $a \in \{0,1\}$, $\varphi[x = a]$ denotes a formula originating from φ by substituting x with value a and the obvious simplifications consisting in removing falsified literals and satisfied clauses. Given a partial assignment $x_1 = a_1, \ldots, x_k = a_k$, then $\varphi[x_1 = a_1, \ldots, x_k = a_k]$ denotes the formula originating from φ by substituting the variables according to the specified partial assignment and the obvious simplifications. Similarly, if $l_1, \ldots, l_k$ is a sequence of literals, then $\varphi[l_1, \ldots, l_k]$ denotes the formula obtained from φ using the partial assignment defined so that the literals $l_1, \ldots, l_k$ are satisfied. We say that a literal l is *pure* in a CNF formula, if it occurs in the formula and the negated literal $\bar{l}$ does not. A literal is *irrelevant* in a formula, if neither the literal nor its negation occurs in the formula. A variable is *pure*, if it appears only positively, or only negatively in φ, i.e. it appears in a literal which is pure in φ.

Let φ be a formula defining a Boolean function f on n variables. An assignment of values $v \in \{0,1\}^n$ is a *model* of φ (also a *satisfying assignment* of φ), if it satisfies f, i.e. if $f(v) = 1$. The set of models of φ is denoted as $T(\varphi)$. The models in $T(\varphi)$ are defined on the variables which have an occurrence in φ. A *partial assignment* assigns values only to a subset of the variables.

Note that an empty clause does not admit a satisfying assignment and an empty CNF is satisfied by any assignment.

3 Efficient Splitting Tree Algorithm

The idea of the algorithm is to construct a decision tree for the function represented by a given satisfiable CNF, such that every subtree larger than a single leaf contains a 1-leaf. Consider a decision tree of a function represented by a CNF φ.

Each leaf node in this tree labeled with 1 represents a set of models of φ, more precisely, a leaf in depth d represents 2^{n-d} models of φ. Moreover, different leaves of the tree represent disjoint sets of models. Given a decision tree for the function represented by φ, we can, by traversing it, generate all models of φ in time proportional to its size. This process leads to a large delay between generating successive models, if the tree contains large subtrees with only 0-leaves. The following condition on a class of formulas describes a situation when this can be avoided.

Definition 3.1. Let U be a class of formulas, let $\varphi \in U$ and let x be a variable with an occurrence in φ. We say that x is a *splitting variable* for φ relative to U, if for every $a \in \{0, 1\}$, such that $\varphi[x = a]$ is satisfiable, we have $\varphi[x = a] \in U$. A class of formulas U has the *splitting property*, if every formula in U containing a variable contains a splitting variable relative to U.

We shall associate a splitting problem with a class of formulas U having splitting property.

Definition 3.2. Let U be a class of formulas with the splitting property. The *splitting problem* relative to U is the following problem: Given a formula $\varphi \in U$, find a splitting variable for φ relative to U and the results of satisfiability tests for the formulas $\varphi[x = 0]$ and $\varphi[x = 1]$.

Note that the complexity of the splitting problem relative to U is also an upper bound on the time of a satisfiability test for formulas in U. This is because a formula φ is satisfiable, if and only if for any variable x we have that at least one of the formulas $\varphi[x = 0]$ and $\varphi[x = 1]$ is satisfiable. The result of these satisfiability checks for a splitting variable x is a required part of a solution to the splitting problem.

Theorem 3.3. *If a class of formulas U has the splitting property and the splitting problem relative to U can be solved in time $c(\varphi)$, where $c(\varphi) \geq \|\varphi\|$ for each formula $\varphi \in U$, then the models of a formula $\varphi \in U$ on n variables can be generated with a delay $O(n \cdot c(\varphi))$.*

Remark 3.4. If φ contains a unit clause and U is closed under unit propagation, then a variable x contained in a unit clause is a splitting variable which can be identified efficiently. The reason is that if φ is known to be satisfiable and one of the formulas $\varphi[x = a]$ contains an empty clause, then the other is satisfiable.

Remark 3.5. It is not hard to observe that if a class U satisfies that

1. the satisfiability of formulas in U can be tested in polynomial time, and
2. U is closed under partial assignments,

then the splitting problem relative to U has polynomial complexity. All classes of generalized satisfiability problem described by [Creignou and Hébrard, 1997] have this property. In addition to other classes, consider, for instance, Horn formulas, SLUR formulas, 2CNFs, q-Horn formulas, etc. As a corollary of Theorem 3.3, it is possible to generate the models of formulas in these classes with a polynomial delay.

The main result of this paper is that the splitting problem relative to a slight generalization of matched formulas also has polynomial complexity although the class of matched formulas is not closed under partial assignments.

4 Pure Literal Satisfiable Formulas

Before considering matched formulas, let us look at the class of formulas which are satisfiable by iterated elimination of pure literals which we call pure literal satisfiable.

Definition 4.1. A *pure literal sequence* for a formula φ is a consistent sequence of literals $(l_1, \ldots, l_k)$, such that for every $i = 1, \ldots, k$, the literal l_i is either pure or irrelevant in the formula $\varphi[l_1, \ldots, l_{i-1}]$.

If L is a pure literal sequence for φ, the formula $\varphi[L]$ will be called the *reduced* formula corresponding to φ and L. If $\varphi[L]$ does not contain a pure literal, L will be called a *maximal* pure literal sequence for φ.

Definition 4.2. A formula φ is *pure literal satisfiable*, if there is a pure literal sequence L for φ, such that the reduced formula $\varphi[L]$ is empty or, equivalently, the literals in L correspond to a satisfying assignment of φ.

A maximal pure literal sequence for a CNF formula can be found in polynomial time by a simple greedy algorithm.

Lemma 4.3. *A maximal pure literal sequence L for a CNF formula φ can be constructed in time $O(\|\varphi\|)$.*

It turns out that $\varphi[L]$ is the same for every maximal pure literal sequence L.

Lemma 4.4. *Let φ be a CNF formula and let L be a maximal pure literal sequence for φ.*

1. *The formula $\varphi[L]$ is uniquely determined by φ.*
2. *The formula φ is pure literal satisfiable, if and only if $\varphi[L]$ is empty.*

Let us note that pure literal satisfiable formulas are not closed under partial assignments. Therefore pure literal satisfiable formulas do not satisfy the second property required in Remark 3.5 and we have to put more effort into showing that pure literal satisfiable formulas have the splitting property and that the splitting problem relative to pure literal satisfiable formulas has polynomial complexity.

If we take the literals of a maximal pure literal sequence for a formula φ in reversed order, we get the order of variables suitable for selecting a splitting variable. It follows that we can solve the splitting problem for a pure literal satisfiable formula in polynomial time.

Theorem 4.5. *The splitting problem relative to pure literal satisfiable formulas can be solved in time $O(\|\varphi\|)$ where φ is the input pure literal satisfiable formula. Moreover, the set $T(\varphi)$ of the models of a pure literal satisfiable formula φ can be generated with a delay of $O(\|\varphi\|)$.*

5 Matched Formulas

In this section we concentrate on matched formulas. In particular we show that the problem of determining the number of models of a matched formula φ, i.e. the size $|T(\varphi)|$, is as hard as a general #SAT problem.

Theorem 5.1. *The problem of determining $|T(\varphi)|$ given a matched formula φ is #P-complete.*

Our goal is to show that we can generate the models of a matched formula with a polynomial delay. Theorem 3.3 cannot be used for this directly, since the class of the matched formulas does not have the splitting property. For example, the monotone formula $(x \vee y)(x \vee z)(y \vee z)$ is matched, but has no splitting variable relative to the class of matched formulas. We thus have to consider a richer class of formulas. The class we consider generalizes matched and pure literal satisfiable formulas as follows.

Definition 5.2. A formula φ is called *pure literal matched*, if $\varphi[L]$ is matched for a maximal pure literal sequence L.

Elimination of a pure literal preserves the property of being matched. Hence, a matched formula is pure literal matched. Clearly, every pure literal satisfiable formula is pure literal matched, since its reduced formula is empty and, hence, matched.

The basic idea of an efficient splitting algorithm for the pure literal matched formulas is presented in the following theorem.

Theorem 5.3. *Let φ be a matched formula. If for every variable x, which has an occurence in φ, there is $a \in \{0,1\}$, such that $\varphi[x=a]$ is not matched, then φ is pure literal satisfiable.*

Assume that φ is a pure literal matched formula. If φ is actually pure literal satisfiable, then the models of φ can be generated with polynomial delay based on Theorem 4.5. Otherwise consider a maximal pure literal sequence L and let $\psi = \varphi[L]$, which is, by the assumption, a matched formula. Since L is maximal, ψ does not contain a pure literal and since φ is not pure literal satisfiable, ψ is nonempty. Hence, by Theorem 5.3, there is a variable x of ψ, such that $\psi[x=0]$ and $\psi[x=1]$ are both matched. Such x is a splitting variable for φ relative to the class of pure literal matched formulas because L can be used as a pure literal sequence for $\varphi[x=a]$ for any value $a \in \{0,1\}$. We thus get the following theorem.

Theorem 5.4. *The splitting problem relative to pure literal matched formulas can be solved in time $O(n \cdot \|\varphi\|)$ where φ is the input formula on n variables.*

Similarly as the class of matched formulas, also the class of pure literal matched formulas is closed under unit propagation and, hence, Remark 3.4 applies to these classes.

As a corollary of Theorem 5.4 and the general bound from Theorem 3.3, we get the main result of the paper.

Corollary 5.5. *Models of a pure literal matched formula φ, in particular of any matched formula φ, on n variables can be generated with a delay $O(n^2 \cdot \|\varphi\|)$.*

6 Linearly Satisfiable Formulas

In this section we consider the class of linearly satisfiable formulas. By results of [Kullmann, 2000], this class generalizes both the matched formulas and the pure literal satisfiable formulas and, by combining the proofs, also the class of pure literal matched formulas. In this section, we show that it is not possible to generate models of linearly satisfiable formulas with a polynomial delay unless P=NP.

Theorem 6.1. *It is an NP-complete problem to determine, whether a general linearly satisfiable formula has at least 2 models.*

Moreover the following is an example of a linearly satisfiable formula which has no splitting variable.

Example 6.2. *Denote $E = \{a \in \{0,1\}^4 \mid 2 \leq a_1 + a_2 + a_3 + a_4 \leq 3\}$ and for every Boolean variable x, let $x^1 = x$ and $x^0 = \overline{x}$. The formula*

$$\beta(x_1, x_2, x_3, x_4) = \bigwedge_{a \in E} \bigvee_{i=1}^{4} x_i^{a_i}$$

is linearly satisfiable, but has no splitting variable relative to the class of linearly satisfiable formulas.

7 Conclusion and Directions for Further Research

In the paper, we have shown that it is possible to generate the models of a matched formula φ on n variables with delay $O(n^2 \cdot \|\varphi\|)$. As a byproduct we have shown that the models of a pure literal satisfiable formula φ (i.e. a formula satisfiable by iterated pure literal elimination) can be generated with delay $O(\|\varphi\|)$. We have also shown that this result cannot be generalized for the class of linearly satisfiable formulas since it is not possible to generate models of linearly satisfiable formulas with a polynomial delay unless P=NP.

The algorithms described in the paper for the cases of pure literal satisfiable and pure literal matched formulas can be used in a general algorithm for model enumeration which is based on splitting tree. This, in turn, is any DPLL based enumeration algorithm. To this end, a similar approach to the one described by Stefan Szeider [2003] can be used. Together with a formula φ we would keep a maximum matching M of $I(\varphi)$. This maximum matching can then be maintained through the reduction and assignment steps performed in the enumeration algorithm. Once the algorithm arrives at a matched formula, it can select splitting variables in the way we have described in this paper which has a guaranteed polynomial delay.

Recall that Szeider [2003] introduced an algorithm for satisfiability parameterized with maximum deficiency of a formula. This algorithm is based on a DPLL search procedure where between decision steps nontrivial reductions are performed. It would be interesting to know whether the algorithm described by Szeider [2003] could be modified into an algorithm for generating models of a general CNF formula so that the delay is parameterized with maximum deficiency.

Acknowledgments

Petr Savický was supported by CE-ITI and GAČR under the grant number GBP202/12/G061 and by the institutional research plan RVO:67985807. Petr Kučera was supported by the Czech Science Foundation (grant GA15-15511S).

References

[Aceto *et al.*, 2013] Luca Aceto, Dario Monica, Anna Ingólfsdóttir, Angelo Montanari, and Guido Sciavicco.

Logic for Programming, Artificial Intelligence, and Reasoning: 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings, chapter An Algorithm for Enumerating Maximal Models of Horn Theories with an Application to Modal Logics, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[Aharoni and Linial, 1986] Ron Aharoni and Nathan Linial. Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. *Journal of Combinatorial Theory, Series A*, 43(2):196 – 204, 1986.

[Coquery *et al.*, 2012] Emmanuel Coquery, Said Jabbour, Lakhdar Sais, Yakoub Salhi, et al. A SAT-based approach for discovering frequent, closed and maximal patterns in a sequence. In *Proceedings of ECAI*, 2012.

[Creignou and Hébrard, 1997] Nadia Creignou and J-J Hébrard. On generating all solutions of generalized satisfiability problems. *Informatique théorique et applications*, 31(6):499–511, 1997.

[Creignou *et al.*, 2011] Nadia Creignou, Frédéric Olive, and Johannes Schmidt. *Theory and Applications of Satisfiability Testing - SAT 2011: 14th International Conference, SAT 2011, Ann Arbor, MI, USA, June 19-22, 2011. Proceedings*, chapter Enumerating All Solutions of a Boolean CSP by Non-decreasing Weight, pages 120–133. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[Dechter and Itai, 1992] Rina Dechter and Alon Itai. Finding all solutions if you can find one. In *AAAI-92 Workshop on Tractable Reasoning*, pages 35–39, 1992.

[Fleischner *et al.*, 2002] Herbert Fleischner, Oliver Kullmann, and Stefan Szeider. Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. *Theoretical Computer Science*, 289(1):503 – 516, 2002.

[Flum and Grohe, 2006] Jörg Flum and Martin Grohe. *Parameterized complexity theory*, volume 3 of *Texts in Theoretical Computer Science. An EATCS Series*. Springer-Verlag Berlin Heidelberg, 1st edition, 2006.

[Franco and Van Gelder, 2003] John Franco and Allen Van Gelder. A perspective on certain polynomial-time solvable classes of satisfiability. *Discrete Appl. Math.*, 125(2-3):177–214, 2003.

[Jabbour *et al.*, 2014] Said Jabbour, Jerry Lonlac, Lakhdar Sais, and Yakoub Salhi. Extending modern SAT solvers for models enumeration. In *IEEE 15th International Conference on Information Reuse and Integration (IRI), 2014*, pages 803–810. IEEE, 2014.

[Johnson *et al.*, 1988] David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119 – 123, 1988.

[Kang and Park, 2005] Hyeong-Ju Kang and In-Cheol Park. SAT-based unbounded symbolic model checking. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(2):129–140, Feb 2005.

[Kavvadias *et al.*, 2000] Dimitris J. Kavvadias, Martha Sideri, and Elias C. Stavropoulos. Generating all maximal models of a Boolean expression. *Information Processing Letters*, 74(3–4):157–162, 2000.

[Kullmann, 2000] Oliver Kullmann. Investigations on autark assignments. *Discrete Applied Mathematics*, 107(1–3):99 – 137, 2000.

[Kullmann, 2003] Oliver Kullmann. Lean clause-sets: generalizations of minimally unsatisfiable clause-sets. *Discrete Applied Mathematics*, 130(2):209 – 249, 2003. The Renesse Issue on Satisfiability.

[McMillan, 2002] Ken L. McMillan. *Computer Aided Verification: 14th International Conference, CAV 2002 Copenhagen, Denmark, July 27–31, 2002 Proceedings*, chapter Applying SAT Methods in Unbounded Symbolic Model Checking, pages 250–264. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[Morgado and Marques-Silva, 2005] A Morgado and J Marques-Silva. Algorithms for propositional model enumeration and counting. Technical report, Instituto de Engenharia de Sistemas e Computadores, Investigação e Desenvolvimento, Lisboa, February 2005.

[Savický and Kučera, 2016] Petr Savický and Petr Kučera. Generating models of a matched formula with a polynomial delay. *Journal of Artificial Intelligence Research*, 56:379–402, 2016.

[Szeider, 2003] Stefan Szeider. Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. In Tandy Warnow and Binhai Zhu, editors, *Computing and Combinatorics*, volume 2697 of *Lecture Notes in Computer Science*, pages 548–558. Springer Berlin Heidelberg, 2003.

[Szeider, 2005] Stefan Szeider. Generalizations of matched CNF formulas. *Annals of Mathematics and Artificial Intelligence*, 43(1-4):223–238, 2005.

[Szeider, 2007] Stefan Szeider. Matched formulas and backdoor sets. In João Marques-Silva and Karem A. Sakallah, editors, *Theory and Applications of Satisfiability Testing – SAT 2007*, volume 4501 of *Lecture Notes in Computer Science*, pages 94–99. Springer Berlin Heidelberg, 2007.

[Tovey, 1984] Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85 – 89, 1984.

[Valiant, 1979] L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189 – 201, 1979.