On the Gap between the Complexity of SAT and Minimization for Certain Classes of Boolean Formulas

Ondřej Čepek and **Štefan Gurský**

Abstract

It is a wellknown fact that the satisfiability problem (SAT) for Boolean formulas in a conjunctive normal form (CNF) is NP complete, i.e. Σ_1 complete. It is also known that the decision version of Boolean minimization for CNF inputs is Σ_2 complete. On the other hand there are several subclasses of CNFs (e.g. Horn CNFs) for which SAT is known to be in $P = \Sigma_0$ while the minimization problem is Σ_1 complete. Thus, for both the general case and the above mentioned subclasses the gap between the complexity of SAT and minimization is exactly one level in the polynomial hierarchy. There are also some subclasses (e.g. quadratic CNFs) for which there is no gap because both SAT and minimization are in $P = \Sigma_0$. In this short note we shall systematically study different classes of Boolean functions with respect to the size of the mentioned gap and show that there also exist classes for which the gap between the complexity of SAT and minimization is the maximum possible (two levels in the polynomial hierarchy). To this end we shall recall a recent result showing that the minimization of the so called matched CNFs is Σ_2 complete.

Introduction

In this short note we study two important decision problems: the satisfialbility problem (SAT) for formulas in conjunctive normal form (CNF) and the Boolean minimization problem for CNFs. We focus on the gap between the complexity of these two problems for several classes of CNFs.

It is widely accepted, that SAT is one of the most important problems in computer science, both from the theoretical point of view, and from the point of view of practical applications. Boolean minimization, which is a problem of finding a shortest CNF logically equivalent to the input CNF, is certainly a less known problem but nevertheless an important one too. By "shortest" CNF we mean a CNF which is minimal with respect to a given measure which is usually one of the following two: the number of clauses or the sum of lengths of all clauses (total number of literal occurences). Sometimes the problem is stated for disjunctive normal forms (DNFs), in which case the measures are the number of terms or the sum of term lengths.

It is a wellknown fact that SAT is NP complete, i.e. Σ_1 complete (Cook 1971). It is also known that the decision version of Boolean minimization is Σ_2 complete (Umans 2000). Thus the gap between the complexity of SAT and minimization for general CNFs is exactly one level in the polynomial hierarchy. There are several classes of CNFs for which the complexity of SAT and the complexity of Boolean minimization both drop one level down (i.e. SAT from Σ_1 to $\Sigma_0 = P$ and Boolean minimization form Σ_2 to Σ_1), resulting again in the same gap (one level in the polynomial hierarchy). Examples of such classes are Horn CNFs and several its extensions, such as renamable Horn CNFs (Aspvall 1980) or q-Horn CNFs (Boros, Crama, and Hammer 1990). In fact, as we shall see later in this paper, for any class with polynomial time SAT which satisfies few additional simple properties, the Boolean minimization problem is guaranteed to drop from Σ_2 at least to Σ_1 (possibly even to Σ_0).

Since for both the general case and the above mentioned subclasses (Horn CNFs and its extensions) the gap between the complexity of SAT and the complexity of minimization is exactly one level in the polynomial hierarchy, it is tempting to conjecture, that this is usually (or even always) the case. However, it is not too hard to find exceptions. There are subclasses of CNFs (e.g. monotone CNFs, quadratic CNFs, some restrictions of Horn CNFs) for which there is no gap because both SAT and minimization are in $P = \Sigma_0$. So the next logical conjecture is that the gap is always at most one level in the polynomial hierarchy. However, also this conjecture is false, as we shall prove later in the paper.

After systematically studying several classes of Boolean functions for which the mentioned gap is small (at most one level in the polynomial hierarchy), we proceed to show that there also exist classes for which the gap between the complexity of SAT and the complexity of minimization is the maximum possible, i.e. two levels in the polynomial hierarchy, which can only happen when SAT is in $P = \Sigma_0$ while Boolean minimization is Σ_2 complete. An example of such a class is the class of so called matched CNFs introduced in (Franco and Gelder 2003). The main result presented here is the Σ_2 completeness of Boolean minimization for matched CNFs. The same gap also appears for a generalization of matched CNFs called var-satisfiable CNFs defined in (Szeider 2005).

Preliminaries

Boolean variables $x_1, x_2, \ldots$ are variables that can get either value true or value false. A *literal* is a variable or its negation. A *clause* is a disjunction of literals and hence the value

of a clause is true, if any of its literals is true. The value of an empty clause is false. A *conjunctive normal form* (CNF) is a conjunction of clauses and it evaluates to true if all its clauses evaluate to true. An empty CNF evaluates to true. The dual objects to clauses and CNFs are terms and DNFs. A *term* is a conjunction of literals and hence the value of a term is false, if any of its literals is false. The value of an empty term is true. A *disjunctive normal form* (DNF) is a disjunction of terms and it evaluates to false if all its terms evaluate to false. An empty DNF evaluates to false.

A *Boolean function* is a mapping from $\{0,1\}^n$ (where n is the number of Boolean variables) to $\{0,1\}$. It is a well known fact, that any Boolean function can be represented by both CNF and DNF, usually in more than one way. Well studied problems associated with CNFs and DNFs are the following:

Problem: SAT
Input: CNF φ
Question: Is φ satisfiable, i.e. is there an assignment of truth values to variables for which φ evaluates to true?

Problem: FALS
Input: DNF φ
Question: Is φ falsifiable, i.e. is there an assignment of truth values to variables for which φ evaluates to false?

These two problems are of course equivalent in the sense, that each of them can be trivially reduced to the other one in polynomial time, since a negation of a CNF is a DNF and vice versa.

Now we are ready to define the decision versions of the Boolean minimization problem. We need just one more definition: two formulas φ and ψ are said to be *equivalent* if they represent the same Boolean function.

Problem: MINCNFCLAUSES
Input: CNF φ and integer k
Question: Is there a CNF ψ equivalent to φ that has at most k clauses?

Problem: MINDNFTERMS
Input: DNF φ and integer k
Question: Is there a DNF ψ equivalent to φ that has at most k terms?

The problems MINCNFCLAUSES and MINDNFTERMS are equivalent, that is each one can be reduced to the other in polynomial time. The reason is again the same as in the case of equivalnce of SAT and FALS, namely the fact that a negation of a CNF is a DNF and vice versa.

Problem: MINCNFLIT
Input: CNF φ and integer k
Question: Is there a CNF ψ equivalent to φ that has at most k occurences of literals?

Problem: MINDNFLIT
Input: DNF φ and integer k
Question: Is there a DNF ψ equivalent to φ that has at most k occurences of literals?

Again, the problems MINCNFLIT and MINDNFLIT are clearly equivalent. It follows, that we can choose either the CNF or the DNF variant of the above minimization problems, find its complexity and then apply the result to the other normal form.

Remark 1. All six decision problems defined above also make sense if the input is restricted to some class C of CNFs or a class D of DNFs. Note, that the reduction arguments used for the complexity equivalence then work only if the negations of CNFs from C constitute the class D and vice versa. Fortunately, this is in fact a quite common phenomenon for many wellknown classes as we shall see in the following section.

The gap for particular classes of CNFs

In this section we shall study the gap between the complexity of SAT and both versions of MinCNF by recalling known results about SAT (or FALS) and MinCNF (or MinDNF) for several classes of formulas. We will mostly try to stick to the CNF versions of the results because SAT is a much more common problem in literature than FALS and the complexity results are usualy proved for SAT not FALS. On the other hand, for some reason the minimization results appear more frequently in the DNF notation, so we will sometimes switch back and forth between the two notations.

General CNFs

The fact that SAT is NP-complete (i.e. Σ_1 complete) for a general CNF input is wellknown. In fact, SAT was the very first problem that was proved to be NP-complete in the ground-breaking paper by S.A.Cook (Cook 1971). The complexity of minimization for general CNF inputs was proved by C.Umans (Umans 2000) almost 30 years later. The proof originally appeared in a DNF version.

Theorem 1 ((Umans 2000))**.** The problems MINDNFTERMS and MINDNFLIT are both Σ_2 complete.

Corollary 1. Problems MINCNFCLAUSES and MINCNFLIT are also Σ_2 complete.

Thus the gap between the complexity of SAT and the complexity of minimization for general CNF inputs is exactly one level in the polynomial hierarchy. Let us now turn our attention to several classes of CNFs for which this gap vanishes.

Monotone CNFs

A CNF is *positive* if it contains only positive literals and it is *negative* if it contains only negative literals. A CNF is *monotone* if it is positive or negative. Similar definitions apply to DNFs. Note, that a negation of a positive CNF is a negative DNF and a negation of a negative CNF is a positive DNF, so a negation of a monotone CNF is always a monotone DNF and vice versa. Thus the correspondence between the complexity of CNF and DNF problems described in Remark 1 works here.

Monotone CNFs are of course always satisfiable, so the SAT problem is trivial. It is easy to see that also the minimization problems MINCNFCLAUSES and MINCNFLIT are easy. Any monotone CNF explicitely contains the canonical CNF of the represented function which is the unique minimal CNF with respect to both the number of clauses and the number of literals. To obtain the canonical CNF from a monotone input, it therefore suffices to check for every clause whether it is absorbed by some other clause which is obviously possible to do in polynomial time with respect to the length of the input CNF. In fact, a stronger result holds: it was proved in (Goldsmith, Hagen, and Mundhenk 2008) that the monotone CNF minimization problem is in Logspace (the result was shown for DNFs).

Quadratic CNFs

A CNF is *quadratic* if every clause in it contains at most two literals and a DNF is quadratic if every term in it contains at most two literals. Clearly, a negation of a quadratic CNF is a quadratic DNF and vice versa, so Remark 1 applies again. Both SAT and minimization problems are again solvable in polynomial time for quadratic CNFs (in fact in linear time with respect to the length of the input CNF), although the problems are not as trivial as in the monotone case. Both linear time algorithms are based on some preprocessing after which the core of the Boolean problem is transformed into a directed graph problem. The SAT algorithm (known as a 2-SAT algorithm in the literature (Aspvall, Plass, and Tarjan 1979)) detects the strongly connected components of the constructed directed graph and from these components either detects unsatisfiability or constructs a satisfying assignment. The minimization algorithm (which constructs a CNF with both the minimum number of clauses and the minimum number of literals) is based on finding a transitive reduction of the underlying directed graph. This linear time algorithm is considered a folklore and the authors of this note would be grateful for any citations pointing to the original publication of this result.

Restrictions of Horn CNFs

The last classes that we study in this note where both SAT and MINCNF problems are solvable in polynomial time are several subclasses of Horn CNFs. A CNF is *Horn* if every clause in it contains at most one positive literal. A DNF is Horn if every term in it contains at most one negative literal. Hence, a negation of a Horn CNF is a Horn DNF and vice versa, so these definitions fit Remark 1 perfectly (and this carries over to the subclasses defined below), and thus we can restrict our attention to CNFs only.

It is a wellknown fact that SAT for Horn CNFs is solvable by unit propagation and several linear time implementations of this algorithm exist (Dowling and Gallier 1984; Itai and Makowsky 1987; Minoux 1988). On the other hand, the minimization problem for Horn CNFs is NP-hard as we shall see in the following subsection. However, there are several nested subclasses of Horn CNFs which admit polynomial time minimization. The definitions of these subclasses rest on the notion of the CNF digraph.

Definition 1 (CNF digraph)**.** For Horn CNF φ, the *CNF digraph* G_φ is a directed graph, where vertices are the Boolean variables and every clause C in φ with exactly one positive literal (called a *head* of C) generates directed edges from variables corresponding to negative literals in C (called *subgoals* of C) to the head of C. Note that clauses having only negative literals generate no edges in G_φ.

Horn CNF φ is *acyclic* if G_φ is acyclic (or equivalently if every strong component of G_φ is a singleton). It is *quasi-acyclic* if every directed edge inside a strong component of G_φ comes from a quadratic prime implicate of φ. In such a case all variables inside a strong component of G_φ are logically equivalent thanks to a cycle of implications (corresponding to the cycle of directed edges), and thus we can view quasi-acyclic CNFs as acyclic ones where the same variable may have several names. Finally, Horn CNF φ is *CQ-Horn* (short for "component-wise quadratic Horn") if every prime implicate C of φ has at most one subgoal of C in the same strong component of G_φ as the head of C. Clearly, acyclic Horn CNFs form a strict subset of quasi-acyclic Horn CNFs, which in turn form a strict subset of CQ-Horn CNFs. Polynomial time algorithms solving both MINCNFCLAUSES and MINCNFLIT problems appeared in (Hammer and Kogan 1992) for acyclic Horn CNFs, in (Hammer and Kogan 1995) for quasi-acyclic Horn CNFs, and in (Boros et al. 2009) for CQ-Horn CNFs. The first algorithm is quite easy, it suffices to transform the input CNF into an irredundant and prime one to obtain the unique minimum CNF. The second algorithm uses some of the tricks known from the minimization of quadratic CNFs (in particular the algorithm for a transitive reduction of a directed graph), while the third algorithm is quite complex.

Horn CNFs and their generalizations

As stated in the previous subsection, SAT for Horn CNFs is solvable in linear time and thus it is in $P = \Sigma_0$. On the other hand, both MINCNF problems are long known to be NP-complete (Σ_1-complete). The complexity of the MINCNFCLAUSES problem for Horn CNFs was first addressed in (Ausiello, D'Atri, and Saccá 1986) where its NP-hardness was established. However, the paper does not use Boolean terminology; the result is stated for directed hypergraphs, which are in some sense isomorphic objects to Horn CNFs. The same result was later independently proved in (Hammer and Kogan 1993), this time using Boolean terms. First complexity results for the MINCNFLIT problem for Horn CNFs are even older than the clause minimization ones. The first NP-hardness proof for the problem appeared in a paper dealing with minimum covers in a relational database (Maier 1980). Although strictly speaking the measure defined in (Maier 1980) is slightly different from the sum of clause lengths, the proof can be easily modified to work also for this measure. A simpler proof (this time really using the number of literals as the minimality measure) then appeared in (Hammer and Kogan 1993). Recently, a proof that works for both measures simultaneously and moreover strengthens the result from general Horn CNFs to cubic Horn CNFs (every clause has at most three

literals) appeared in (Boros, Čepek, and Kučera 2013).

We see, that the gap between the complexity of SAT and minimization for Horn CNFs (and even for cubic Horn CNFs) is one level in the polynomial hierarchy. An interesting question is, whether this gap grows larger for generalizations of Horn CNFs which maintain polynomial time SAT, i.e. whether the complexity of minimization grows for such classes. Good candidates to look at are the classes of renamable Horn CNFs (Aspvall 1980) and q-Horn CNFs (Boros, Crama, and Hammer 1990; Boros, Hammer, and Sun 1994). A CNF is renamable Horn if it can be turned into a Horn CNF by complementing some subsets of variables (negative literals are switched to positive and vice versa for the variables in the selected subset). The class of q-Horn CNFs generalizes both renamable Horn and quadratic CNFs. For both classes it is not difficult to prove that the complexity of both MINCNF problems stays in $NP = \Sigma_1$, so the gap stays the same. The principal reason for this fact is that both classes are closed under partial assignment. In fact, we can make a much more general observation here. Let X be a class of CNFs satisfying the following three properties:

- Satisfiability: Given an arbitrary CNF $\varphi \in X$ it is possible to decide in polynomial time with respect to the size of φ whether φ is satisfiable.
- Partial assignment: Given an arbitrary CNF $\varphi \in X$, if ψ is produced from φ by fixing some variables to 0 or 1 and substituting these values into φ, then $\psi \in X$.
- Prime representations: Given an arbitrary CNF $\varphi \in X$, if φ represents a function f then all prime CNF representations of f belong to X.

Classes of CNFs satisfying all of the above properties are called *tractable* (sometimes also polynomial time recognition of CNFs from X is required for X to be called tractable, see e.g. (Čepek, Kučera, and Savický 2012)). Note, that Horn, renamable Horn, and q-Horn CNFs are all tractable classes of CNFs. It is easy to see, that given a CNF φ from a tractable class, we can decide in polynomial time whether a given clause C is an implicate of φ by substituting the appropriate values (which make C zero) into φ and testing the satisfiablility of the resulting formula (which is guaranteed to be in X due to the second property). This property of tractable classes has two important consequences (Čepek, Kučera, and Savický 2012).

Lemma 1. Let φ and ψ be two CNFs from a tractable class. Then it is possible to test in polynomial time whether φ and ψ represent the same Boolean function (are logically equivalent) or not.

Proof. It suffices to test for each clause C in φ whether it is an implicate of ψ and for each clause C in ψ whether it is an implicate of φ. The two CNFs are logically equivalent if and only if none of these tests fails. □

Lemma 2. Let X be a tractable class of CNFs. Then both MINCNFCLAUSES and MINCNFLIT problems for CNFs from X belong to $NP = \Sigma_1$.

Proof. Let φ, k be a positive instance of MINCNFCLAUSES (or MINCNFLIT respectively), where $\varphi \in X$. Let f be the Boolean function represented by φ. Then a prime CNF ψ, which represents f and consists of at most k clauses (literals) is a polynomial size certificate for φ, k being a positive instance. Note that we may assume that ψ is a prime representation since the existence of any CNF representing f and consisting of at most k clauses (literals) clearly implies the existence of a prime CNF with the same property. Moreover, the tractability of X (namely the third property) implies $\psi \in X$. The fact that ψ is a polynomially verifiable certificate now follows from the fact that both φ and ψ belong to the tractable class X, and hence using Lemma 1 we get, that it is possible to test in polynomial time that they both represent the same function f. □

The above Lemma shows that if we want to find a class of CNFs where the complexity gap between SAT and minimization is larger than one level in the polynomial hierarchy we have to search among classes where SAT is in P but the class is not tractable. Therefore one of the other two properties establishing tractability (other than the polynomiality of SAT) must be violated. As we shall see in the next section, a good candidate is the second property, i.e. the class we are looking for should not be closed under partial assignment.

Matched formulas

In this section we study the class of matched CNFs and show that while SAT is in P for this class (which is a trivial observation) both minimization problems for this class are Σ_2 complete. Let us start with the definition of the class.

Definition 2 (Incidence graph)**.** For CNF φ, the *incidence graph* G is a bipartite graph, where vertices of one partity are clauses of φ and vertices of the other partity are its variables. A clause is connected to variables that it contains (that is clause C is connected to variable x if and only if it contains either literal x or literal $\overline{x}$).

Definition 3 (Matched CNF)**.** CNF is called *matched* if its incidence graph has matching that covers all clauses. That is every clause can be assigned its own variable.

Matched DNFs can be defined in an analogous way (terms take the role of clauses). Note that obviously a negation of a matched CNF is a matched DNF and vice versa, so Remark 1 applies again for matched formulas. The following simple observation is immediately obvious.

Observation 1 ((Franco and Gelder 2003))**.** Every matched CNF is satisfiable.

Proof. For clause C take its matched variable x. If C contains literal x, set x to true, othervise set it to false. This way all clauses can be satisfied. □

It is easy to see, that the class of matched CNFs is not closed under partial assignment. Consider for example a CNF

$$(x \vee y \vee z) \wedge (x \vee \overline{y} \vee z) \wedge (x \vee y \vee \overline{z})$$

which is clearly matched, but after the partial assignment $x \leftarrow 0$ the resulting CNF

$$(y \vee z) \wedge (\overline{y} \vee z) \wedge (y \vee \overline{z})$$

is not matched. Thus the class of matched formulas is not tractable (despite having trivially solvable SAT), and hence is a good candidate for a large gap between the complexity of SAT and minimization. So let us now look at the minimization problems for matched formulas. Due to Remark 1 we can use either the CNF or the DNF versions of the problems establishing the complexity for both. We will follow the paper (Umans 2000) which uses DNF notation and use the DNF versions.

Theorem 2. The problems MINDNFTERM and MINDNFLIT with inputs restricted to matched DNFs are both Σ_2 complete.

Sketch of proof. The proofs of both these statements follow very closely the original proofs of C. Umans for general CNFs with certain ammendments. Those proofs for general CNFs are quite long and can be found in full in (Umans 2000). The original proof for MINDNFLIT can be found in (Umans 1998), the proof for MINDNFTERM can be found in (Umans 1999). Full version of both proofs with ammendments that make the proofs work for matched formulas can be found in (Gurský 2010). Because of space limitations, we will only show here what changes have to be made to modify the proofs for general CNFs to work also for inputs restricted to matched CNFs.

In both proofs an instance of a certain Σ_2 complete problem is reduced to an instance of the minimization problem at hand. We present here only the formula in the resulting instance of the minimization problem and the way in which it can be transformed into a matched formula without destroying the properties of the formula that make the reduction work.

Let us first look at MINDNFLIT. In his proof Umans reduced an instance of a Σ_2 complete problem SHORTEST IMPLICANT CORE (proof of Σ_2 completness provided therein) to instance of MINDNFLIT. The formula in the resulting instance of MINDNFLIT is in the form $\varphi'' = t_l w_1 w_2 w_3 \dots w_{m'} \vee \bigvee_{i=1}^{m} s'_i$ where s'_i is in the form $s_i w_1 w_2 \dots w_{i-1} w_{i+1} \dots w_{m'}$ with w_i being variable for all i and t_l and s_i are terms for all i. An important fact is that in this formula we can find a matching that matches each of s'_i terms with variable w_{i+1} (the last one with w_1) and the first term can be matched with any of the variables in t_l (it is not empty). Therefore this formula is matched and problem MINDNFLIT is Σ_2 complete when restricted to matched DNFs.

In the proof for MINDNFTERM, C. Umans reduces again problem SHORTEST IMPLICANT CORE (albeit a version somewhat different from the first one) to an instance of MINDNFTERM. The resulting instance is in this case a formula that consists of two parts. The first part are terms in the form $s'_i = s_i z_1 z_2 \dots z_{i-1} z_{i+1} \dots z_m$ with s_i being term and z_i being variable for each i. Each s'_i can be then matched with z_{i+1}. The second part of the formula has terms in the form $u_{i,j} = (p_i)\bar{x_j} z_1 z_2 \dots z_m$, where all parts are variables. However, we can change p_i from being a variable to being a term of parity function on new set of variables $a_1, a_2 \dots$ such that there is enough of a-variables to provide matching for all terms. The parity is used since it cannot be shortened in any way. The formula length grows only polynomially (each term gets a polynomial amount of new variables) and the resulting formula is again matched. Thus MINDNFLIT is also Σ_2 complete when restricted to matched DNFs. □

Corollary 2. Problems MINCNFCLAUSE and MINCNFLIT restricted to matched CNFs are also Σ_2 complete.

Conclusion

In this short note we have looked at several classes of Boolean formulas with respect to the complexity ot the satisfiability problem and the minimizations problems. We have seen that for all classes where both the complexity of SAT and the complexity of minimization are known from the literature, the gap between these two complexities is at most one level in the polynomial hierarchy. The main result of this note shows that the class of matched formulas has the same complexity of minimization as a general formulas, i.e. the minimization problem is Σ_2 complete. This result is somewhat surprising, since the matched formulas have a very simple structure and are always satisfiable, which means that the SAT problem is trivial for them. This altogether implies that the gap between the complexity of SAT and the complexity of minimization for the class of matched formulas is two levels in the polynomial hierarchy, which is of course the maximum possible gap.

Acknowledgments

The first author gratefully acknowledges a support by the Czech Science Foundation (grant P202/10/1188).

The second author gratefully acknowledges the support of the Charles University Grant Agency (grant No. 1390213).

References

Aspvall, B.; Plass, M. F.; and Tarjan, R. E. 1979. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters* 8(3):121 – 123.

Aspvall, B. 1980. Recognizing disguised NR(1) instances of the satisfiability problem. *Journal of Algorithms* 1(1):97–103.

Ausiello, G.; D'Atri, A.; and Saccá, D. 1986. Minimal representation of directed hypergraphs. *SIAM J. Comput.* 15(2):418–431.

Boros, E.; Čepek, O.; Kogan, A.; and Kučera, P. 2009. A subclass of Horn CNFs optimally compressible in polynomial time. *Annals of Mathematics and Artificial Intelligence* 57(3-4):249–291.

Boros, E.; Crama, Y.; and Hammer, P. L. 1990. Polynomial-time inference of all valid implications for horn and related formulae. *Annals of Mathematics and Artificial Intelligence* 1:21 – 32.

Boros, E.; Hammer, P. L.; and Sun, X. 1994. Recognition of q-Horn formulae in linear time. *Discrete Applied Mathematics* 55(1):1 – 13.

Boros, E.; Čepek, O.; and Kučera, P. 2013. A decomposition method for CNF minimality proofs. *Theoretical Computer Science*. Available online 23 September 2013.

Čepek, O.; Kučera, P.; and Savický, P. 2012. Boolean functions with a simple certificate for CNF complexity. *Discrete Applied Mathematics* 160(45):365 – 382.

Cook, S. A. 1971. The complexity of theorem-proving procedures. In *STOC'71: Proceedings of the third annual ACM Symposium on Theory of Computing*, 151–158.

Dowling, W. F., and Gallier, J. H. 1984. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *The Journal of Logic Programming* 1(3):267 – 284.

Franco, J., and Gelder, A. V. 2003. A perspective on certain polynomial-time solvable classes of satisfiability. *Discrete Applied Mathematics* 125(23):177 – 214.

Goldsmith, J.; Hagen, M.; and Mundhenk, M. 2008. Complexity of DNF minimization and isomorphism testing for monotone formulas. *Inf. Comput.* 206(6):760–775.

Gurský, Š. 2010. Časová složitost minimalizace booleovských funkcí. Master's thesis, Charles University, Faculty of Mathematics and Physics, Prague.

Hammer, P. L., and Kogan, A. 1992. Horn functions and their DNFs. *Information Processing Letters* 44(1):23 – 29.

Hammer, P. L., and Kogan, A. 1993. Optimal compression of propositional Horn knowledge bases: complexity and approximation. *Artif. Intell.* 64(1):131–145.

Hammer, P., and Kogan, A. 1995. Quasi-acyclic propositional Horn knowledge bases: optimal compression. *Knowledge and Data Engineering, IEEE Transactions on* 7(5):751–762.

Itai, A., and Makowsky, J. 1987. Unification as a complexity measure for logic programming. *The Journal of Logic Programming* 4(2):105 – 117.

Maier, D. 1980. Minimum covers in relational database model. *J. ACM* 27(4):664–674.

Minoux, M. 1988. Ltur: a simplified linear-time unit resolution algorithm for Horn formulae and computer implementation. *Information Processing Letters* 29(1):1 – 12.

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

Umans, C. 1998. The minimum equivalent DNF problem and shortest implicants. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, 556–563.

Umans, C. 1999. Hardness of approximating Σ_2^p minimization problems. In *in Proc. 40th IEEE Symp. on Foundations of Computer Science*, 465–474.

Umans, C. 2000. *Approximability and completeness in the polynomial hierarchy*. Ph.D. Dissertation, University of California, Berkeley. Chair-Papadimitriou, Christos H.