Boolean functions with long prime implicants

Ondřej Čepek and **Petr Kučera** and **Stanislav Kuřík**[*]

Departement of Theoretical Computer Science and Mathematical Logic
Faculty of Mathematics and Physics, Charles University in Prague
Malostranské nám. 25, 118 00 Praha 1, Czech Republic

Abstract

In this short note we introduce a class of Boolean functions defined by a minimum length of its prime implicants. We show that given a DNF one can test in polynomial time whether it represents a function from this class. Moreover, in case that the answer is affirmative we present a polynomial time algorithm which outputs a shortest DNF representation of the given function. Therefore the defined class of functions is a new member of a relatively small family of classes for which the Boolean minimization problem can be solved in polynomial time. Finally, we present a generalization of the above class which is still recognizable in polynomial time, and for which the Boolean minimization problem can be approximated within a constatnt factor.

Introduction

One of the most commonly used representations of Boolean functions are DNFs (disjunctive normal forms). For a given function there are typically many different DNFs representing it, which may significantly vary in length. In some applications an important problem is the following: for a given function find a shortest DNF among all of its possible DNF representations. For instance, in artificial intelligence this problem is equivalent to finding a most compact representation of a given knowledge base (Hammer and Kogan 1993; 1994). Such a transformation of a knowledge base accomplishes a knowledge compression, since the actual knowledge does not change, while the size of the representation can be significantly reduced. In general, this problem, known as Boolean minimization (BM), can be stated as follows: given a DNF Φ find a DNF Φ' representing the same function and such that Φ' consists of a minimum possible number of terms.

It is easy to see that BM is NP-hard as it contains the DNF falsifiability problem (FALS) as its special case (a non-falsifiable Boolean function, i.e. a tautology, can be trivially recognized from its shortest DNF representation). In fact, BM was shown to be probably harder than FALS: while FALS is NP-complete (i.e. Σ_1^p-complete) (Cook 1971), BM is Σ_2^p-complete (Umans 2001) (see also the review paper (Umans, Villa, and Sangiovanni-Vincentelli 2006) for related results). BM remains NP-hard even for some classes of Boolean functions for which FALS is solvable in polynomial time. The best known example of such a class are Horn functions (see (Ausiello, D'Atri, and Sacca 1986; Boros and Čepek 1994; Čepek 1995; Hammer and Kogan 1993; Maier 1980) for various BM intractability results).

On the positive side, it is long known that a polynomial time BM algorithm exists for quadratic DNFs. Later, (Hammer and Kogan 1995) introduced two subclasses of Horn functions, acyclic and quasi-acyclic functions, for which BM is solvable in polynomial time as well. This result was then generalized in (Boros et al. 2010), where a class of CQ-Horn functions containing both of the above mentioned subclasses of Horn functions was defined, and a polynomial time BM algorithm for CQ-Horn functions was presented.

The Boolean minimization problem can be also considered in a different context, where the input function is not given by a DNF but instead by a zero-one matrix, where rows correspond to all truepoints and columns to all prime implicants of the function (let us call such a matrix a TPI matrix). An entry of the TPI matrix is one if and only if the corresponding prime implicant covers the corresponding truepoint, i.e if the implicant evaluates to one on the given Boolean vector. Note, that both the number of rows and the number of columns of a TPI matrix can be exponential with respect to the size of a DNF representation of the given function.

Clearly, prime DNF representations of the given function are then in a one-to-one correspondence to those subsets of columns (sets of prime implicants) which cover all truepoints. Finding a minimum (prime) DNF is then equivalent to finding a minimum set cover in an instance where truepoints are the elements of the base set and prime implicates correspond to the system of subsets from which the minimum cover should be selected. Of course, since the set cover problem is known to be NP-hard (Garey and Johnson 1979), this approach does not yield a polynomial time Boolean minimization algorithm even when the input is given by a TPI matrix instead of a DNF. However, when searching for classes of Boolean functions which admit polynomial time BM, one may try to look at those cases where (a) the TPI matrix can be generated from the input DNF in polynomial time, and (b) the resulting set cover problem can be solved in polynomial time.

[*]The first two authors gratefully acknowledge a support by the Czech Science Foundation (grant P202/10/1188)

One such case arises when each prime implicant of the input DNF covers at most two truepoints, or in other words when each prime implicant has length at least $n-1$, where n is the number of variables. We shall call a class of such DNFs $LPI(1)$. In such a situation the set cover problem on the TPI matrix reduces to an edge cover problem on an undirected graph, which is solvable in polynomial time (Norman and Rabin 1959), and so condition (b) holds. We show in this paper, that also (a) is true by proving that a simple variant of the consensus method can be used to generate the TPI matrix in polynomial time with respect to the length of any input DNF. The same procedure also tests for any input DNF whether all its prime implicants are long enough, i.e. the recognition problem for the class $LPI(1)$ is solvable in polynomial time as well. In fact we show a more general result. We define a class $LPI(k)$ as the class of those DNFs which have all prime implicants of length at least $n-k$, and we prove that for any constant k both the recognition problem for $LPI(k)$ and the problem of generating the TPI matrix are solvable in polynomial time with respect to the length of the input DNF. Finally, we show that for $LPI(k)$ (where $k > 1$ is a constant) a known approximation algorithm for the set cover problem with a bounded set size (Duh and Fürer 1997) can be turned into an approximation algorithm for BM achieving a constant approximation ratio.

Definitions and results

A *Boolean function* f on n propositional variables $x_1, \ldots, x_n$ is a mapping $f : \{0,1\}^n \mapsto \{0,1\}$. Propositional variables $x_1, \ldots, x_n$ and their negations $\overline{x}_1, \ldots, \overline{x}_n$ are called *literals* (*positive* and *negative* literals, respectively). A conjunction of literals

$$T = \bigwedge_{i \in I} x_i \wedge \bigwedge_{j \in J} \overline{x}_j \qquad (1)$$

is called a *term*, if every propositional variable appears in it at most once, i.e. if $I \cap J = \emptyset$. We often do not write the conjunction operator explicitely, e.g. instead of $x \wedge y \wedge \overline{z}$ we use $xy\overline{z}$. For two Boolean functions f and g we write $f \leq g$ if

$$\begin{aligned} &\forall (x_1, \ldots, x_n) \in \{0,1\}^n \\ &[f(x_1, \ldots, x_n) = 1 \Longrightarrow g(x_1, \ldots, x_n) = 1]. \end{aligned} \qquad (2)$$

Since each term is in itself a Boolean function, formula (2) also defines the meaning of inequalities $T_1 \leq T_2$, $T_1 \leq f$, and $f \leq T_1$, where T_1, T_2 are terms and f is a Boolean function.

We say that a term T_1 absorbs another term T_2 if $T_2 \leq T_1$ (i.e. literals in T_1 form a subset of literals in T_2, e.g. term $\overline{x}z$ absorbs term $\overline{x}\overline{y}z$). A term T is called an implicant of a function f if $T \leq f$. An implicant T is called prime, if there is no distinct implicate T' absorbing T, or in other words, an implicant of a function is prime if dropping any literal from it produces a term which is not an implicate of that function. The set of all prime implicants of function f will be denoted by $PI(f)$.

It is a well-known fact that every Boolean function f can be represented by a disjunction of terms (see e.g. (Genesereth and Nilsson 1987)). Such an expression is called a *disjunctiove normal form* (or DNF) of the Boolean function f. It should be noted that a given Boolean function may have many DNF representations. If two distinct DNFs, say Φ_1 and Φ_2 represent the same function, we say that they are *equivalent*, and denote this fact by $\Phi_1 \equiv \Phi_2$. A DNF Φ representing function f is called *prime* if each term of Φ is a prime implicant of function f. The unique DNF consisting of all prime implicants of function f, i.e. all implicants in $PI(f)$ is called the *canonical DNF* of f. We shall often treat a DNF as a set of its terms.

Boolean minimization problem is defined as follows. Given DNF Φ representing a function f, find a shortest equivalent DNF which also represents f. We can measure length of Φ by various measures, however in this paper we shall consider only the number of terms in Φ and by "shortest" we shall always mean "with the least number of terms".

Now we are ready to define the central notion of this paper. We shall say, that Boolean function f on n variables belongs to class *LPI*(k), where *LPI* stands for *long prime implicants*, if every prime implicant of f contains at least $n-k$ literals.

Consensus method and LPI(k)

In this section we shall recall a well know consensus procedure (which is called resolution method in case of CNFs) and we shall investigate its behaviour on a DNF belonging to class LPI(k).

Definition 1 (Consensus) *We say, that terms T_1 and T_2 have a conflict in variable x, if x appears positively in one of them and negatively in the other. If T_1 and T_2 have exactly one conflict, then we say, that they have a* consensus*. In this case we can write $T_1 = x\tilde{T_1}$ and $T_2 = \overline{x}\tilde{T_2}$ for some propositional variable x and some terms $\tilde{T_1}$ and $\tilde{T_2}$ which have no conflict. The terms T_1 and T_2 are called parent terms and the conjunction* $\textsc{Cons}(T_1, T_2) = \tilde{T_1}\tilde{T_2}$ *is called the* consensus *of T_1 and T_2.*

The following is an easy lemma (Büning and Letterman 1999; Quine 1955) and it lies at the basis of consensus procedure, also called Quine's procedure.

Lemma 2 *Let T_1 and T_2 be two implicans of a Boolean function f which have a consensus. Then* $\textsc{Cons}(C_1, C_2)$ *is also an implicant of f.*

The consensus procedure (Büning and Letterman 1999; Quine 1955) consists of repeating two basic operations, removing absorbed terms and adding a non-absorbed consensus of two terms. When none of these operations would modify the current DNF, the procedure stops, and at that time, the current DNF consists of all prime implicants of the given Boolean function.

```
CONSENSUS PROCEDURE($\Phi$)
Input: DNF $\Phi$, representing Boolean function $f$
Output: Canonical DNF of $f$

while one of the following conditions applies do
    if there exist two terms $C$ and $D$ in $\Phi$ such that
        $C$ absorbs $D$ then
    $\Phi \leftarrow \Phi \setminus \{D\}$
  if there exist two terms $C$ and $D$ in $\Phi$
    with consensus $X = \textsc{Cons}(C, D)$
    which is not absorbed by any other term in $\Phi$ then
    $\Phi \leftarrow \Phi \vee \{\textsc{Cons}(C, D)\}$
end while
return $\Phi$
```

It can be shown (Büning and Letterman 1999; Quine 1955), that this algorithm always finishes and that it correctly returns canonical DNF of function f given by input DNF Φ. We shall show, that if every implicant of f has length at least $n-k$, i.e. if f belongs to class LPI(k), then the Consensus Procedure runs in polynomial time if k is a fixed constant. Note, that this is not true in general as there is a DNF Φ with m terms such that the number of prime implicant of the function it represents is exponential in m, an example of such a formula can be found in (Crama and Hammer 2008). We shall show that this situation cannot occur if the input DNF represents a function from LPI(k). First we shall bound number of prime implicants by a polynomial in the number of truepoints.

Lemma 3 *Let f be an n-variable Boolean function which has m true points. Then in every DNF representation of f, the maximum number of distinct terms of length $d \leq n$ is*

$$\frac{2m}{2^{n-d}}\binom{\lceil \log_2 m \rceil}{n-d}$$

Proof. In the proof we shall use geometric interpretation of a subcube corresponding to function f. As we already know, every term of length n is uniquely determined by a single Boolean point (the only one it covers or vice versa, the only one which makes it equal 1) or a vertex in the Boolean hypercube, every term of length $n-1$ is determined by a unique pair of Boolean points (an edge in the Boolean hypercube), every term of length $n-2$ is uniquely determined by a foursome of Boolean points (a 2-dimensional cube) and so on. Following this pattern, the number of distinct terms of length $d \leq n$ in any DNF representation of f is bounded by the number of $(n-d)$-dimensional subcubes in the maximal possible subcube of an n-dimensional Boolean hypercube consisting of m vertices. An n-dimensional hypercube consists of 2^n vertices and thus the maximum dimension of a subcube made of m vertices is $\lceil \log_2 m \rceil$. Notice, that in an r-dimensional hypercube there is $\binom{r}{r-k} = \binom{r}{k}$ ways to pick $(r-k)$ coordinates and 2^{r-k} ways to fix these coordinates to a given vector. Thus an r-dimensional hypercube contains

$$\binom{r}{k}2^{r-k}$$

k-dimensional subcubes. Now if we set $r = \lceil \log_2 n \rceil$ and $k = n-d$, the number of distinct terms of length d in any DNF representation of an n-variable Boolean function which has m true points is bounded from above by

$$2^{\lceil \log_2 m \rceil - (n-d)}\binom{\lceil \log_2 m \rceil}{n-d} \leq \frac{2m}{2^{n-d}}\binom{\lceil \log_2 m \rceil}{n-d}$$

where the inequality follows from a simple fact that $\lceil \log_2 m \rceil \leq 1 + \log_2 m$ □

Now we are ready to show, that the number of prime implicants of a function f from LPI(k) represented by a DNF with r terms is polynomial in r although it may be exponential in k.

Lemma 4 *Let $f \in$ LPI(k) and ϕ be an arbitrary DNF representation of f with r terms. Then the number of distinct implicants of f is at most $2(2k + 2\lceil \log_2 r \rceil)^k rk$.*

Proof. Let us suppose we are given DNF ϕ representing n-variable Boolean function $f \in$ LPI(k). We will denote by r_i the number of terms of length i present in ϕ. By assumption, $r_i = 0$ for $i = 0, 1, \ldots, n-k-1$. Because every term in ϕ of length d determines exactly 2^{n-d} true points of f, it is clear that f has at most

$$\begin{aligned} & 2^k r_{n-k} + 2^{k-1} r_{n-(k-1)} + \cdots + 2r_{n-1} + r_n \\ \leq\ & 2^k (r_{n-k} + r_{n-(k-1)} + \cdots + r_{n-1} + r_n) \\ =\ & 2^k r \end{aligned}$$

true points. From Lemma 3 we get that there may be at most

$$2 \cdot 2^{k+d-n} r \binom{k + \lceil \log_2 r \rceil}{n-d}$$

implicants of length d, for $n-k \leq d \leq n$. In total this gives the following bound on the number of implicants f can have:

$$\begin{aligned} |PI(f)| &\leq \sum_{d=n-k}^{n} 2 \cdot 2^{k+d-n} r \binom{k + \lceil \log_2 r \rceil}{n-d} \\ &\leq 2 \cdot 2^k r \sum_{d=n-k}^{n} \binom{k + \lceil \log_2 r \rceil}{n-d} \\ &\leq 2(2k + 2\lceil \log_2 r \rceil)^k rk \end{aligned}$$

which is obviously polynomial in r. □

As an immediate corollary of Lemma 4 we get, that the Consensus Procedure, if implemented properly, runs in polynomial time on a DNF which represents a function from LPI(k). .

Theorem 5 *Let k be a fixed constant. Then the Consensus Procedure can be implemented to run in polynomial time on DNFs representing functions from class LPI(k).*

Proof. Let Φ denote a DNF which consists of r terms and represents a function from LPI(k). Let Φ be the input of the Consensus Procedure, and let us consider a variant of the Consensus Procedure which runs in two phases. In the first phase, it computes the resolution closure of Φ, i.e. it determines all implicants, which can be derived by consensuses from Φ, and in the second phase, all non-prime implicants are removed from the list.

The first phase can be implemented using a list L of terms, which at the beginning is initialized with terms from Φ, then in a cycle procedure traverses L from the beginning and when considering a term T in the list, it considers every term S which are before T in L. If T and S have consensus $X = \textsc{Cons}$(T, S), which is not yet present in L, then

X is appended at the end of the list. When all terms from the list have been processed, then the first phase stops. It is not hard to observe, that after the first phase all implicants which can be derived by consensuses from Φ are present in list L, moreover running time of this phase is bounded by polynom in the final size of L, which in case of Φ being in LPI(k) is at most $2(2k + 2\lceil\log_2 r\rceil)^k rk$ by Lemma 4.

In the second phase the procedure traverses the list an checks for absorptions, which can be easily done in polynomial time in the length of L.

Together we get, that the running time is polynomial in $2(2k + 2\lceil\log_2 r\rceil)^k rk$, which is polynomial in r if k is a fixed constant. □

Note, that the implementation of the Consensus Procedure described in the proof of Theorem 5 is certainly not the best one, but it is sufficient to make a polynomial running time estimate, which is what we need. It is now obvious, how the Consensus Procedure can be used to check, whether an arbitrary DNF Φ belongs to class LPI(k). We reject the input as soon as an implicant which is shorter than $n-k$ is generated. If no such implicant occurs during the Consensus Procedure on Φ, we can answer that Φ indeed represents a function which belongs to class LPI(k). Moreover, in such a case all prime implicants of Φ were generated. Since also the number of truepoints of function f represented by Φ (assuming that Φ has r terms of length at least $n-k$) is bounded from above by $r2^k$, it is obvious that the TPI matrix of f can be generated from Φ in polynomial time.

Corollary 6 *Let k be a fixed constant and Φ an arbitrary DNF. We can check in polynomial time whether Φ represents a function from LPI(k), and if so, we can generate the TPI matrix of the function in polynomial time as well.*

Minimization of LPI(k)

It is a long known fact, that finding a shortest (with respect to the number of terms) representation of a Boolean function given by the set of its truepoints, can be viewed as a hyperedge covering problem in a hypergraph. In case of a general hypergraph this is equal to SET COVER which is a well-known NP-complete problem, but in case of a graph, it corresponds to an edge covering problem, which can be solved in polynomial time by matching techniques. In this section we shall describe, how this fact can be used to a polynomial minimization of an LPI(1) function and how SET COVER approximation algorithms can be used to approximate a shortest representation in case of a general LPI(k) class.

We shall start by associating a hypergraph to Boolean function f, whic has a set of truepoints $T(f)$. We shall denote this hypergraph $\mathcal{H}_f = (V_f, \mathcal{E}_f)$. Its vertices will consist of the truepoints, i.e. $V_f = T(f)$, its hyperedges will correspond to sets of truepoints satisfied by the same prime implicant of f. In particular, let S be a prime implicant of f and let us denote

$$E_S = \{v \in T(f) \mid S(v) = 1\},$$

now the set of hyperedges is defined as

$$\mathcal{E}_f = \{E_S \mid S \in PI(f)\}.$$

It can be observed, that the number of terms in a shortest DNF representantion of f is equal to the minimum number of hyperedges which cover all vertices, see e.g. (Berge 1985). In case of general hypergraph, this problem is the same as a well known SET COVER problem. In case $\mathcal{H}_f$ is actually a graph, this problem corresponds to edge cover problem in a graph, which can be solved in polynomial time using matching techniques (Norman and Rabin 1959). This is the case of functions in LPI(1) and thus we get the following theorem.

Theorem 7 *Let Φ be a DNF representing a function f which belongs to LPI(1). Then the shortest DNF representation of f can be found in polynomial time.*

Proof. By Corollary 6 the TPI matrix of f which is equal to the incidence matrix of $\mathcal{H}_f$ can be found in polynomial time. In fact in this case the polynomial bounding the running time will have a low degree as $k = 1$. There are two kinds of prime implicants of f, let I_0 denote the set of prime implicants of f of length n and let I_1 denoted the set of prime implicants of f of length $n-1$. Obviously, for a prime implicant $S \in I_0$ then there exists a uniquely defined truepoint v, such that $S(v) = 1$, and for any other prime implicant $S' \neq S$ of f we have $S'(v) = 0$. It follows, that S is actually an essential implicant and that it has to be present in any DNF representation of f, or in other words, I_0 corresponds to singleton edges which must be in every hyperedge cover of $\mathcal{H}_f$. Each prime implicant $S \in I_1$has exactly two truepoints v on which it is satisfied and thus in this case E_S is an edge and if we exclude singleton hyperedges, $\mathcal{H}'_f = (T(f), \mathcal{E}' = \{E_S \mid S \in I_1\})$ is a graph. Its edge cover can be found in polynomial time (Norman and Rabin 1959), let us denote it by $F \subseteq \mathcal{E}'$. Now let us denote the set of corresponding implicants by $I_F \subseteq I_1$. Then $I_0 \cup I_F$ forms a shortest DNF representation of f. □

If we consider LPI(k) for $k > 1$ we can use the same approach, but in this case we get an instance of SET COVER, which is NP-complete and remains so even if the size of largest set is bounded by 3 (Garey and Johnson 1979). So there is little hope for a polynomial time optimization algorithm like for $k = 1$.

However, for the d-SET COVER problem, a restriction of the SET COVER problem where every set has size at most d, (Duh and Fürer 1997) proposed an approximation algorithm based on a semi-local optimization technique, leading to the approximation ratio of $\mathcal{H}_d - \frac{1}{2}$ where

$$\mathcal{H}_d = \sum_{i=1}^{d} \frac{1}{i}$$

is the dth Harmonic number. Since it is a well-known fact that the values of the sequence $\mathcal{H}_d - \ln(d)$ decrease monotonically towards the limit

$$\lim_{d \to \infty} (\mathcal{H}_d - \ln(d)) = \gamma$$

(where $\gamma = 0.577...$ is the Euler-Mascheroni constant) and every set in the instance of the SET COVER which arises from minimizing an LPI(k) function has size at most $d = 2^k$, we get an approximation algorithm with the approximation ratio $ratio(k)$ such that

$$\lim_{k \to \infty} (ratio(k) - k) = c$$

where $c = \gamma - \frac{1}{2}$. This means that the approximation ratio $ratio(k)$ for BM in the class LPI(k) asymptotically behaves like k. For instance for $k = 2$ we get

$$ratio(2) = \mathcal{H}_4 - \frac{1}{2} = \frac{19}{12} < 2.$$

The approximation factor for the d-Set Cover problem was improved by increasing the additive constant by Levin (Levin 2007) and most recently by Athanassopoulos, Caragiannis and Kaklamanis (Athanassopoulos, Caragiannis, and Kaklamanis 2009).

References

Athanassopoulos, S.; Caragiannis, I.; and Kaklamanis, C. 2009. Analysis of approximation algorithms for k-set cover using factor-revealing linear programs. *Theory of Computing Systems* 45:555–576. 10.1007/s00224-008-9112-3.

Ausiello, G.; D'Atri, A.; and Sacca, D. 1986. Minimal representation of directed hypergraphs. *SIAM Journal on Computing* 15(2):418–431.

Berge, C. 1985. *Graphs and Hypergraphs*. Elsevier Science Ltd.

Boros, E., and Čepek, O. 1994. On the complexity of horn minimization. Technical Report 1-94, RUTCOR Research Report RRR, Rutgers University, New Brunswick, NJ.

Boros, E.; Čepek, O.; Kogan, A.; and Kučera, P. 2010. Exclusive and essential sets of implicates of boolean functions. *Discrete Applied Mathematics* 158(2):81 – 96.

Büning, H. K., and Letterman, T. 1999. *Propositional Logic: Deduction and Algorithms*. New York, NY, USA: Cambridge University Press.

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. New York, NY, USA: ACM.

Crama, Y., and Hammer, P. L. 2008. Boolean functions. Theory, Algorithms and Applications (draft).

Duh, R.-c., and Fürer, M. 1997. Approximation of k-set cover by semi-local optimization. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, 256–264. New York, NY, USA: ACM.

Garey, M., and Johnson, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W.H. Freeman and Company.

Genesereth, M., and Nilsson, N. 1987. *Logical Foundations of Artificial Intelligence*. Los Altos, CA: Morgan Kaufmann.

Hammer, P., and Kogan, A. 1993. Optimal compression of propositional horn knowledge bases: Complexity and approximation. *Artificial Intelligence* 64:131 – 145.

Hammer, P. L., and Kogan, A. 1994. Knowledge compression - logic minimization for expert systems. In *Proceedings of IISF/ACM Japan International Symposium*, 306–312. Tokyo: World Scientific, Singapore.

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

Levin, A. 2007. Approximating the unweighted k-set cover problem: Greedy meets local search. In Erlebach, T., and Kaklamanis, C., eds., *Approximation and Online Algorithms*, volume 4368 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. 290–301.

Maier, D. 1980. Minimal covers in the relational database model. *Journal of the ACM* 27:664 – 674.

Norman, R. Z., and Rabin, M. O. 1959. An algorithm for a minimum cover of a graph. *Proc. Amer. Math. Soc.* 10:315–319.

Quine, W. 1955. A way to simplify truth functions. *Amer.Math.Monthly* 62:627–631.

Umans, C.; Villa, T.; and Sangiovanni-Vincentelli, A. L. 2006. Complexity of two-level logic minimization. *IEEE Trans. on CAD of Integrated Circuits and Systems* 25(7):1230–1246.

Umans, C. 2001. The minimum equivalent dnf problem and shortest implicants. *J. Comput. Syst. Sci.* 63(4):597–611.

Čepek, O. 1995. *Structural Properties and Minimization of Horn Boolean Functions*. Ph.D. dissertation, Rutgers University, New Brunswick, NJ, October 1995.