On the Complexity and Inapproximability of Shortest Implicant Problems

Christopher Umans*

Computer Science Division
U.C. Berkeley
Berkeley, CA 94720-1776
umans@cs.berkeley.edu

Abstract. We investigate the complexity and approximability of a basic optimization problem in the second level of the Polynomial Hierarchy, that of finding shortest implicants. We show that the DNF variant of this problem is complete for a complexity class in the second level of the hierarchy utilizing $\log^2 n$-limited nondeterminism. We obtain inapproximability results for the DNF and formula variants of the shortest implicant problem that show that trivial approximation algorithms are optimal for these problems, up to lower order terms. It is hoped that these results will be useful in studying the complexity and approximability of circuit minimization problems, which have close connections to implicant problems.

1 Introduction

Circuit minimization problems have been extensively studied since the 1950's as important practical problems in logic synthesis, and, since the early 1970's, as natural problems in the second level of the Polynomial Hierarchy. These problems are widely believed to be Σ_2^p-complete; however, they have defied precise complexity classification until very recently, when the Minimum Equivalent DNF problem was shown to be Σ_2^p-complete [11], verifying a conjecture of Stockmeyer [9]. Still, very little has been proved about the complexity of the other major circuit minimization problems: the only non-trivial hardness result known for the Minimum Equivalent Expression problem is $\mathrm{P}_{||}^{\mathrm{NP}}$-hardness [5], and nothing better than trivial coNP-hardness is know for the Minimum Equivalent Circuit problem. The precise complexity is also unknown for many other variants of these problems, such as the "Π_2^p" versions ("Is C a minimal circuit?"). And, despite the continuing importance of these problems in logic synthesis and the significant effort spent designing heuristic and approximate solutions, no formal results (positive or negative) on the approximability of circuit minimization problems are known. This is partly attributable to the dearth of other well-studied optimization problems in the second level of the hierarchy.

* Supported in part by an NSF Graduate Research Fellowship and NSF grant CCR-9626361

As a first step towards improving the current state of knowledge with respect to the complexity and approximability of circuit minimization problems, and in order to expand the available "tools" with which to work in the second level of the hierarchy, we focus in this paper on a basic optimization problem that is closely connected to circuit minimization, namely, *finding shortest implicants*.

The connection between implicant finding and circuit minimization is not surprising; since the introduction of the now-classic Quine-McCluskey algorithm [8, 6], most heuristic and exact methods for DNF minimization have been based on explicitly or implicitly generating the prime implicants of the input function (see [2, 10]), and manipulating implicants also plays an important role in other circuit minimization problems [3]. Implicant problems have already proven their usefulness in the context of complexity results for circuit minimization — in [11], a variant of the shortest implicant problem is a critical intermediate step in the reduction showing that the Minimum Equivalent DNF problem is Σ_2^p-complete.

Our results. In this paper, we settle the complexity of the DNF variant of the shortest implicant problem, proving it complete for a class in the second level of the hierarchy utilizing limited non-determinism. Previous work [11] placed the problem in a similar (larger) class but did not establish the hardness result. We also note that the complement of this problem, properly interpreted, is a "robust" version of CNF SAT, and that other "log-robust" variants of NP-complete problems are complete for this class.

We also obtain optimal inapproximability results for the DNF and formula variants of the shortest implicant problem. For the formula version, this improves the inapproximability factor from $n^{1/2-\epsilon}$ (in [11]) to $n^{1-\epsilon}$. The main technical aspect of the proof involves the fairly delicate construction of formulae whose shortest implicants and implicates are almost as large as the formula itself. These results indicate that the best approximation algorithms for these problems are trivial ones – for the DNF variant, the algorithm returns the shortest term of the input DNF; for the formula variant, the algorithm returns any satisfying assignment. In other words, no non-trivial exploitation of the special structure of formulae or DNF formulae aids in the approximation of the shortest implicant problem.

Practical significance of complexity results in the second level of the hierarchy. In the more common case of NP-complete problems, a completeness result removes the likelihood of an efficient exact solution and justifies a concentration of effort on approximate or heuristic solutions. It might be argued that for circuit minimization and shortest implicant problems, for which coNP-hardness can be trivially proved, more precise complexity characterizations are of little more than academic interest to practitioners. However, one quickly realizes that most reasonable methods for approximately or heuristically solving these problems requires the exact solution of TAUTOLOGY as a "subroutine." Indeed, at least one major software suite for logic minimization (ESPRESSO) utilizes just such a TAUTOLOGY subroutine [3], which of course has been tuned to work reasonably quickly on most real-world instances. In this context an "efficient" algorithm is an algorithm in P^{NP}, and a hardness result requires placing the problem in a complexity class above NP or coNP. It is therefore appropriate to investigate the complexity of these problems *relative to an NP oracle*, which requires precisely the sort of characterizations in the second level of the hierarchy that we provide in this paper.

2 Preliminaries

Let f be a Boolean function. An *implicant* of f is a conjunction that implies f; an *implicate* of f is a disjunction that is implied by f. For each implicant C, there is a natural associated partial assignment that sets the variable x_i to 1 if the literal x_i appears in conjunction C and to 0 if the literal $\overline{x_i}$ appears in C. When this partial assignment is applied to f, we get a function f' that is a tautology; in this way, we can think of an implicant of f as a partial assignment that forces f to one. Similarly, an implicate of f can be thought of as a partial assignment that forces f to zero.

If C is an implicant (resp. implicate) of f, then by $\mathrm{vars}(C)$ we mean the set of variables appearing in conjunction (resp. disjunction) C. The *length* of C is simply $|\mathrm{vars}(C)|$. We adopt the following shorthand: $SI(f)$ denotes the length of the shortest implicant of f; $SC(f)$ denotes the length of the shortest implicate (sometimes called clause) of f; $S(f)$ is defined to be $\min\{SI(f), SC(f)\}$. In other words, $SI(f)$ is the smallest number of variables that must be fixed in order to force f to one; $SC(f)$ is the smallest number of variables that must be fixed to force f to zero, and $S(f)$ is the smallest number of variables that must be fixed to force f to a constant.

If ϕ is a formula, then $|\phi|$ is the *size* of ϕ, defined to be the number of occurrences of literals in ϕ, and $L(f)$ denotes the formula complexity of Boolean function f, i.e. the size of the smallest formula that computes f.

The two variants of the shortest implicant problem that we deal with in this paper are:

SHORTEST IMPLICANT[DNF]. Given a DNF formula ϕ and an integer k, is there an implicant of ϕ that contains k or fewer literals?

SHORTEST IMPLICANT[FORMULA]. Given a $(\wedge, \vee, \neg)$-formula ϕ and an integer k, is there an implicant of ϕ that contains k or fewer literals?

A technique used throughout this paper involves substituting formulae for individual variables. This can be thought of as "weighting" the variables. The following lemma (proof omitted) describes how these substitutions affect the length of the shortest implicants and implicates of a formula:

Lemma 1. *Let g and $f(x_1, x_2, \ldots, x_n)$ be Boolean functions, let $V \subseteq \{x_1, x_2, \ldots, x_n\}$ and let $f' = f(\phi_1, \phi_2, \ldots \phi_n)$, where ϕ_i is an independent copy of g if $x_i \in V$ and ϕ_i is just x_i otherwise. Then the following hold:*

1. *$SI(f') \geq \min_C\{|vars(C) \setminus V| + S(g)|vars(C) \cap V|\}$, where C ranges over all implicants of f.*

2. if C is an implicant of f and $vars(C) \cap V = \emptyset$ then C is an implicant of f'.

By symmetry, the lemma also holds if "implicant" is replaced by "implicate" and "$SI(f')$" is replaced by "$SC(f')$."

3 Complexity of finding shortest implicants of a DNF

We first restate a positive result from [11]:

Lemma 2 ([11]). *Let $\phi = t_1 \vee t_2 \vee \ldots \vee t_n$ be a DNF formula. Then any shortest implicant C of ϕ may be obtained from some term t_i with at most $\log n$ additions of literals and at most $\log n$ deletions of literals (C is "$\log n$-close" to t_i).*

The class required to capture the complexity of SHORTEST IMPLICANT[DNF] we call GC($\log^2 n$, coNP), after the "*g*uess then *c*heck" framework of Cai and Chen [1] (however they are predominantly concerned with classes in this framework with far weaker verifiers). The class GC($\log^2 n$, coNP) can be described as all languages L expressible as:

$$L = \{I : \exists X \text{ such that } R(I, X)\},$$

where $|X| \leq O(\log^2 |I|)$ and R is a coNP predicate. Notice that the usual logical characterization of Σ_2^p is identical except that $|X|$ is bounded only by some polynomial in $|I|$.

Theorem 1. SHORTEST IMPLICANT[DNF] *is GC($\log^2 n$, coNP)-complete.*

Proof. Membership in the class follows from Lemma 2. Hardness is via a generic reduction. Let L be a language in GC($\log^2 n$, coNP), and let I be an instance. Let ϕ be a 3-DNF obtained from $R(I, X)$ via a Cook reduction. Formula ϕ consists of Boolean variables $x_1, x_2 \ldots x_k$ that constitute X (with $k = c \log^2 |I|$ for some constant c), and additional Boolean variables $y_1, y_2, \ldots y_m$. Using standard methods, we can ensure that no term of ϕ contains more than one x variable. Instance I is in L if and only if there exists an implicant of ϕ consisting of exactly the variables $x_1, x_2, \ldots x_k$.

We think of the variables $x_1, x_2 \ldots x_k$ as being grouped into $r = \log |I|$ blocks of $s = k/r$ variables each. We group the terms of ϕ similarly, with ϕ_i containing the disjunction of exactly those terms of ϕ in which some variable in the i^{th} group appears (i.e. ϕ_1 contains those terms of ϕ that include one of $x_1, x_2, \ldots x_s$; ϕ_2 contains terms that include one of $x_{s+1}, x_{s+2}, \ldots x_{2s}$; etc...). Some terms in ϕ include no x variables at all, and we let ϕ_0 contain the disjunction of these terms. Because no term of ϕ contains more than one x variable, $\phi_0, \phi_1, \phi_2, \ldots, \phi_r$ partition the terms of ϕ, and of course $\phi \equiv \phi_0 \vee \phi_1 \vee \phi_2 \vee \cdots \vee \phi_r$.

Within each group of x variables, there are $t = 2^s = |I|^c$ possible partial assignments to those x variables. Denote by ϕ_{ij} the formula ϕ_i in which the j^{th} partial assignment is applied to the x variables in the formula, for $1 \leq i \leq r$ and $1 \leq j \leq t$.

Finally, we define the following DNF formula f involving new variables a_{ij} and z_i:

$$\begin{array}{cccccccc} f = & a_{11}\overline{z_1} & \vee & a_{12}\overline{z_1} & \vee \ldots \vee & a_{1t}\overline{z_1} \\ \vee & z_1 a_{21}\overline{z_2} & \vee & z_1 a_{22}\overline{z_2} & \vee \ldots \vee & z_1 a_{2t}\overline{z_2} \\ & \vdots & & \vdots & \vdots & \vdots \\ \vee & z_{r-2}a_{(r-1)1}\overline{z_{r-1}} & \vee & z_{r-2}a_{(r-1)2}\overline{z_{r-1}} & \vee \ldots \vee & z_{r-2}a_{(r-1)t}\overline{z_{r-1}} \\ \vee & z_{r-1}a_{r1} & \vee & z_{r-1}a_{r2} & \vee \ldots \vee & z_{r-1}a_{rt} \end{array}$$

It is straightforward to verify that every implicant of f that consists of *only* the a variables must include at least one a_{ij} variable for each $i \in \{1 \ldots r\}$, and that every such conjunction is an implicant of f.

We are now one step away from the final formula. First we define ϕ' as follows:

$$\phi' = f \wedge \left(\phi_0 \vee \bigvee_{i=1}^{r} \bigvee_{j=1}^{t} (a_{ij} \wedge \phi_{ij}) \right)$$

When multiplied out into DNF, ϕ' is a 7-DNF. Finally, define ϕ'' to be ϕ' in which all variables except for the a variables have been substituted for by parity on $r+1$ variables. Notice that the resulting formula is still only polynomially large when multiplied out into DNF, because the DNF representations of the parity formulas each have $2^{r+1}/2 = |I|$ terms, and there are a constant number (at most 7) of substitutions per term of ϕ'.

We claim that ϕ'' has an implicant of length at most r if and only if $I \in L$.

($\Leftarrow$) If $I \in L$, then there exists an implicant of ϕ consisting of exactly the x variables. Let A be the corresponding partial assignment. By Lemma 1 it is sufficient to exhibit an implicant of ϕ' of length at most r consisting of only the a variables. The implicant of ϕ' consists of a_{ij} for each $1 \leq i \leq r$, where the partial assignment A when restricted to the i^{th} group of x variables is the j^{th} possible partial assignment. This is an implicant of f by our previous claim, and it is an implicant of the second half of ϕ' because its corresponding partial assignment preserves at least those terms in ϕ_A.

($\Rightarrow$) If there exists an implicant of ϕ'' of length at most r, then there must exist an implicant C of ϕ' of length at most r that consists of only the a variables. If not, then $SI(\phi'') > r$ by Lemma 1, a contradiction. Implicant C must be an implicant of f and so by our previous claim it must include at least one a_{ij} for each $i \in \{1 \ldots r\}$. Therefore, it includes *exactly* one a_{ij} for each such i, which implies that there is a partial assignment to ϕ consisting of exactly the x variables: for each $i \in \{1 \ldots r\}$, the i^{th} group of x variables takes on the j^{th} possible value, where a_{ij} is the single variable from among $\{a_{i1}, a_{i2}, \ldots a_{it}\}$ that appears in C. □

More on the class GC($\log^2 n$, coNP). The complement of GC($\log^2 n$, coNP) also contains some interesting natural problems, notably:

ROBUST CNF-SAT Given a CNF formula ϕ and an integer k, is ϕ_ρ satisfiable for all restrictions ρ that set at most k variables to constants?

which is simply the complement of the CNF version of SHORTEST IMPLICANT[DNF] (which asks for the shortest implic*ate* of a CNF formula). By the previous theorem, it is complete for the complement of $\mathrm{GC}(\log^2 n, \mathrm{coNP})$. The "robust" versions of some other NP-complete problems are also complete for this class, although the $\log n$ restriction appears explicitly in the problem statement. For example, one can show that LOG-ROBUST DOMINATING SET (the problem of determining whether a graph has a small dominating set even after the deletion of at most $\log n$ edges) is complete for the complement of $\mathrm{GC}(\log^2 n, \mathrm{coNP})$.

The class $\mathrm{GC}(\log^2 n, \mathrm{coNP})$ appears to be only slightly larger than coNP. It is unlikely to include P^{NP} or even NP itself; intuitively, it does not possess enough "existential nondeterminism" to contain NP. This contrasts with the class $\mathrm{NP}[\log^2 n]^{\mathrm{NP}}$ [1] which *does* contain P^{NP}, and was the previous best upper bound on the complexity of SHORTEST IMPLICANT[DNF] [11]. Clarifying the relationship between $\mathrm{GC}(\log^2 n, \mathrm{coNP})$ is an interesting open quesion.

4 Inapproximability of shortest implicant problems

In this section we consider the approximability of SHORTEST IMPLICANT[DNF]and SHORTEST IMPLICANT[FORMULA].

SHORTEST IMPLICANT[DNF]. By the positive result in section 3, the algorithm that returns the shortest term in the input DNF formula is an approximation algorithm that is guaranteed to return an implicant that is within a $\log m$ *additive* factor of optimal, where m is the number of terms in the input formula. In the next theorem we show that this algorithm is essentially optimal (up to a small constant factor):

Theorem 2. SHORTEST IMPLICANT[DNF] *is coNP-hard to approximate to within an* $(1/4 - \epsilon)\log m$ *additive factor, where* m *is the number of terms in the input formula, for any constant* $\epsilon > 0$.

Proof. Let ϕ be an instance of 3-DNF TAUTOLOGY. We obtain ϕ' by "weighting" each variable by substituting parity on $\log m$ variables (via Lemma 1). The new formula has $O(m^4)$ terms.

If ϕ is a positive instance (i.e. it has an implicant of length zero), then ϕ' has a shortest implicant of length zero. The largest implicant returned by a $(1/4 - \epsilon)\log m$ additive approximation algorithm has length at most $(1/4 - \epsilon)\log m^4$. Otherwise, the shortest implicant of ϕ has length at least one, and so the shortest implicant of ϕ' has length at least $\log m$.

Clearly, for any $\epsilon > 0$, $(1/4 - \epsilon)\log m^4 = (1 - 4\epsilon)\log m < \log m$, and so the largest answer the approximation algorithm might give on a positive instance is strictly smaller than the smallest answer it might give on a negative instance. □

[1] The class $\mathrm{NP}[\log^2 n]$ is a subclass of NP in which the usual NDTM is permitted only $O(\log^2 n)$ nondeterministic steps. It is studied in [4, 7].

SHORTEST IMPLICANT[FORMULA]. We now give an optimal inapproximability result for SHORTEST IMPLICANT[FORMULA] (up to lower order terms), which implies that the best approximation algorithm simply returns a single satisfying assignment if there is one (achieving a factor n approximation). In [11] the following theorem is proved; we sketch the relevant aspects of the proof:

Theorem 3 ([11]). SHORTEST IMPLICANT[FORMULA] *is Σ_2^p-hard to approximate to within an $n^{1/2-\epsilon}$ factor, for any constant $\epsilon > 0$.*

Proof. (sketch) We begin with an instance of QSAT_2 given by $\exists X_1 \forall X_2 \phi$, where $X_1 = \{x_1, x_2, \ldots x_m\}$ and X_2 are the sets of existential and universal variables, respectively, and ϕ is a DNF formula involving the variables in X_1 and X_2. We define the formula ϕ' using additional variables $w_1, w_2, \ldots w_m$ as follows:

$$\phi' = \phi \wedge \bigwedge_{i=1}^{m} \left(x_i \overline{w_i} \vee \overline{x_i} w_i \right) .$$

It is easy to show that if the instance of QSAT_2 is a positive instance, then ϕ' has an implicant of length $2m$ consisting of exactly the x and w variables; if the instance of QSAT_2 is a negative instance, then all implicants of ϕ' have length at least $2m+1$, and include all of the x and w variables and at least one variable from X_2.

We define ϕ'' to be ϕ' in which each variable in X_2 has been weighted by a parity formula on N^c variables (as described in Lemma 1), where $N = |\phi'|$ and c is a constant to be determined later. Then the largest answer an $n^{1/2-\epsilon}$-approximate algorithm might give on input ϕ'' if the instance of QSAT_2 is a positive instance is $2m \cdot O(N^{(2c+1)(1/2-\epsilon)})$ (since in this case there exists an implicant of ϕ'' of length $2m$, and $|\phi''| \leq O(N^{(2c+1)})$ because each substituted parity formula has quadratic size). The smallest answer an $n^{1/2-\epsilon}$ algorithm might give on input ϕ'' if the instance of QSAT_2 is a negative instance is $2m + N^c$, by Lemma 1 and our above observation about the structure of the implicants of ϕ'' in this case.

Finally, for sufficiently large N and $c \geq (3/4)\epsilon^{-1}$, we have

$$2m \cdot O(N^{(2c+1)(1/2-\epsilon)}) < 2m + N^c,$$

and hence an $n^{1/2-\epsilon}$-approximate algorithm for SHORTEST IMPLICANT[FORMULA] can distinguish between positive and negative instances of QSAT_2. □

Notice that the $1/2$ in the approximation factor comes from the fact that the parity formulae have quadratic size. By Krapchenko's lower bound (cf. [12]), smaller parity formulae are not possible. However, there may exist smaller formulae that retain the critical property of the parity formulae used in the proof: namely, they have no "short" implicants or implicates. To make this formal, we need a definition: if f is a Boolean formula, let $R(f) = \log L(f) / \log S(f)$. Notice that $R(\text{parity}) = 2$, and that $R(f) \geq 1$ for any formula f.

If there exists an efficiently constructible formula family $F = (f_i)$ for which $R(f_i) < 2$ for all i, then we can use that family in place of the parity formulae in the above proof to obtain improved inapproximability to within an $n^{1/R(F)-\epsilon}$ factor. By the next lemma, it suffices to find a individual formula f instead of a family. The proof follows easily from Lemma 1.

Lemma 3 (Composition Lemma). *Let $f(x_1, x_2, \ldots, x_n)$ be a Boolean formula, and let $f' = f(f_1, f_2, \ldots, f_n)$, where $f_1, f_2, \ldots, f_n$ are independent copies of f. Then $R(f') \leq R(f)$.*

The following theorem is proved in the appendix:

Theorem 4. *There exists a formula f for which $R(f) \leq 1 + \delta$ for any $\delta > 0$.*

From this we have the following inapproximability result for SHORTEST IMPLICANT[FORMULA]:

Theorem 5. SHORTEST IMPLICANT[FORMULA] *is Σ_2^p-hard to approximate to within an $n^{1-\epsilon}$ factor, for any constant $\epsilon > 0$.*

Proof. Initially, the reduction exhaustively searches for the smallest formula f for which $R(f) \leq 1+\delta$ (where δ depends on ϵ). The time required for the search is just a constant overhead in this context. Once such an f is located (its existence is guaranteed by Theorem 4) we compose it with itself repeatedly to obtain a formula whose shortest implicants and implicates are at least the desired length, and substitute it as we did with the parity formula in the proof of Theorem 3. □

5 Conclusions and open problems

As noted in the introduction, natural optimization problems that are complete for the second level of the Polynomial Hierarchy are quite rare. In this paper we have not only added SHORTEST IMPLICANT[FORMULA] to this category, but also have established optimal inapproximability results for it. We have also given perhaps the first example of a natural problem that is complete for a class of limited non-determinism in the second level of the hierarchy, demonstrating the usefulness of limited non-determinism for classifying problems even at this level of hardness. Together with the results in [11], the complexity and inapproximability of shortest implicant problems are now well-understood, and it is hoped that their close relation to circuit minimization problems can now be exploited to prove completeness and formal approximability results for these important real-world problems.

In addition to the major open problems concerning circuit minimization problems, there are several smaller questions raised by this paper: (1) are there other natural complete problems for $GC(\log^2 n, \text{coNP})$ or its complement, preferably ones that do not include an explicit mention of $\log n$ in their definition? (2) what is the relation of $GC(\log^2 n, \text{coNP})$ to surrounding complexity classes (conditional on reasonable complexity-theoretic assumptions)? (3) are there other applications (perhaps in formula complexity) of the "restriction-resistant" formulae constructed in the appendix?

References

[1] Liming Cai and Jianer Chen. On the amount of nondeterminism and the power of verifying. *SIAM Journal on Computing*, 26(3):733–750, 1997.

[2] Olivier Coudert. Two-level logic minimization: an overview. *Integration, the VLSI journal*, 17(2):97–140, 1994.

[3] Srinivas Devadas, Adhijit Ghosh, and Kurt Keutzer. *Logic Synthesis*. McGraw-Hill, Inc., 1994.

[4] Josep Diaz and Jacobo Toran. Classes of bounded nondeterminism. *Mathematical Systems Theory*, 23(1):21–32, 1990.

[5] Edith Hemaspaandra and Gerd Wechsung. The minimization problem for boolean formulas. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, pages 575–84, 1997.

[6] E. J. McCluskey. Minimization of boolean functions. *Bell Systems Technical Journal*, 35(5):1417–44, 1956.

[7] Christos H. Papadimitriou and Mihalis Yannakakis. On limited nondeterminism and the complexity of the V–C dimension. *Journal of Computer and System Sciences*, 53(2):161–70, 1996.

[8] W. V. Quine. The problem of simplifying truth functions. *American Mathematics Monthly*, 59(8):521–531, 1952.

[9] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[10] Tadeusz Strzemecki. Polynomial-time algorithms for generation of prime implicants. *Journal of Complexity*, 8(1):37–63, 1992.

[11] Christopher Umans. The minimum equivalent DNF problem and shortest implicants. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS '98)*, pages 556–563, 1998.

[12] Ingo Wegener. *The complexity of Boolean functions*. Teubner, 1987.

A Appendix: Existence of a formula f with $R(f) \leq 1 + \epsilon$

In this appendix we prove Theorem 4. We first note that relatively simple formulae will not suffice:

Theorem 6. *Let f be a monotone or read-once formula. Then $R(f) \geq 2$.*

Proof. (sketch) We show that the product $SI(f)SC(f)$ is a formal complexity measure (cf. [12]), and hence a lower bound for $L(f)$, by a simple induction on the size of the formula. Then $S(f) = \min\{SI(f), SC(f)\} \leq (SI(f)SC(f))^{1/2} \leq L(f)^{1/2}$, so $R(f) \geq 2$. $\square$

We now proceed with the proof of Theorem 4. The following is the key lemma:

Lemma 4. *If there exists a formula g with $R(g) = c$ then there exists a formula g' with $R(g') \leq 2 - 1/c + \epsilon$ for any $\epsilon > 0$.*

Proof. First, we use the Composition Lemma to obtain a sequence of formulae $g_1, g_2, g_3, \ldots$ for which $R(g_i) \leq c$ for all i: let g_1 be g composed with itself so that $a = S(g_1)$ is sufficiently large (this requirement will be made more precise below), and define g_n for $n > 1$ to be g_1 composed with itself $n-1$ times, so $S(g_n) \geq a^n$. Let m be the smallest even integer such that $m \geq a^c$.

Next, define recursively a sequence of formula $f_0, f_1, f_2, \ldots$ built from $g_1, g_2, \ldots$. In the following, superscripts identify multiple independent copies of each formula, and "maj" is a formula that is true if at least $m/2$ of its m arguments are true, where m is even.

$$f_0 = x \qquad f_n = \bigwedge_{i=1}^{m} \left(g_n^{(i)} \neg \mathrm{maj}(g_n^{(1)}, \ldots g_n^{(m)}) \vee f_{n-1}^{(i)} \right)$$

For $n > 0$, f_n is true iff for at least $(m/2+1)$ distinct values of i, $g_n^{(i)}$ is zero and $f_{n-1}^{(i)}$ is one, and for all other values of i, either $g_n^{(i)}$ is one or $f_{n-1}^{(i)}$ is one. We now determine bounds on $SC(f_n)$, $SI(f_n)$ and $L(f_n)$:

- For $SC(f_n)$: an implicate of f_n $(n > 0)$ must either (1) force $g_n^{(i)}$ to zero and $f_{n-1}^{(i)}$ to zero for some i, or (2) for at least $m/2$ values of i force $g_n^{(i)}$ to one or $f_{n-1}^{(i)}$ to zero. In the first case, the implicate has length at least $a^n + SC(f_{n-1})$; in the second, the implicate has length at least $(m/2)\min\{a^n, SC(f_{n-1})\}$. We know that $SC(f_0) = 1$, and by induction, $SC(f_n) \geq a^n$; here we use the fact that we can select a large enough so that $m/2 \geq a$.
- For $SI(f_n)$: an implicant of f_n $(n > 0)$ must force at least $m/2+1$ of the f_{n-1} to one. Again by induction we have that $SI(f_n) \geq (m/2+1)^n \geq (m/2)^n$. As an upper bound, notice that forcing all of the f_{n-1} to one forces the whole formula to one, so for $n > 0$, $SI(f_n) \leq mSI(f_{n-1})$. We know that $SI(f_0) = 1$, so by induction, $SI(f_n) \leq m^n$.
- For $L(f_n)$: we have the following recurrence: $L(f_0) = 1$ and $L(f_n) \leq p(m)a^{cn} + mL(f_{n-1}) \leq p(m)m^n + mL(f_{n-1})$, where $p(m) \leq O(m^d)$ is a fixed polynomial in m accounting for all of the occurrences of copies of g_n. It is easy to verify the upper bound $L(f_n) \leq m^n + p(m)nm^n \leq O(nm^{n+d})$.

Now, we take g'_n to be the disjunction of $k = \lceil SI(f_n)/SC(f_n) \rceil$ independent copies of f_n. This has the effect of multiplying the length of the shortest implicate by k, so that $S(g'_n) = SI(f_n)$. We have:

$$R(g'_n) \leq \frac{\log\left(O(nm^{n+d})\lceil \frac{m^n}{a^n} \rceil\right)}{\log((m/2)^n)} \leq \frac{\log\left(O(nm^d(m^2/a)^n)\right)}{\log((m/2)^n)}$$

As n approaches infinity, this last expression approaches $\log(m^2/a)/\log(m/2)$. By choosing a sufficiently large initially, this value can be made arbitrarily close to $2-1/c$. Then taking n sufficiently large yields a formula $g' = g'_n$ with $R(g') \leq 2 - 1/c + \epsilon$ for any $\epsilon > 0$.

One detail remains: the final step in the construction makes sense only if $SI(f_n) \geq SC(f_n)$; if this inequality does not hold, then we observe that f_n itself satisfies the lemma. □

Now we apply this lemma repeatedly, starting with parity (for which $c = 2$), to show that for any $\delta > 0$, there exists an f with $R(f) \leq 1 + \delta$, thus proving Theorem 4.