A Generalization of the Satisfiability Coding Lemma and Its Applications

Milan Mossé
Department of Philsophy, University of California Berkeley, CA, USA

Harry Sha
Department of Computer Science, University of Toronto, CA

Li-Yang Tan
Department of Computer Science, Stanford University, CA, USA

Abstract

The seminal *Satisfiability Coding Lemma* of Paturi, Pudlák, and Zane is a coding scheme for satisfying assignments of k-CNF formulas. We generalize it to give a coding scheme for *implicants* and use this generalized scheme to establish new structural and algorithmic properties of *prime implicants* of k-CNF formulas.

Our first application is a near-optimal bound of $n \cdot 3^{n(1-\Omega(1/k))}$ on the number of prime implicants of any n-variable k-CNF formula. This resolves an open problem from the Ph.D. thesis of Talebanfard, who proved such a bound for the special case of constant-read k-CNF formulas. Our proof is algorithmic in nature, yielding an algorithm for computing the set of all prime implicants – the *Blake Canonical Form* – of a given k-CNF formula. The problem of computing the Blake Canonical Form of a given function is a classic one, dating back to Quine, and our work gives the first non-trivial algorithm for k-CNF formulas.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Prime Implicants, Satisfiability Coding Lemma, Blake Canonical Form, k-SAT

Digital Object Identifier 10.4230/LIPIcs.SAT.2022.9

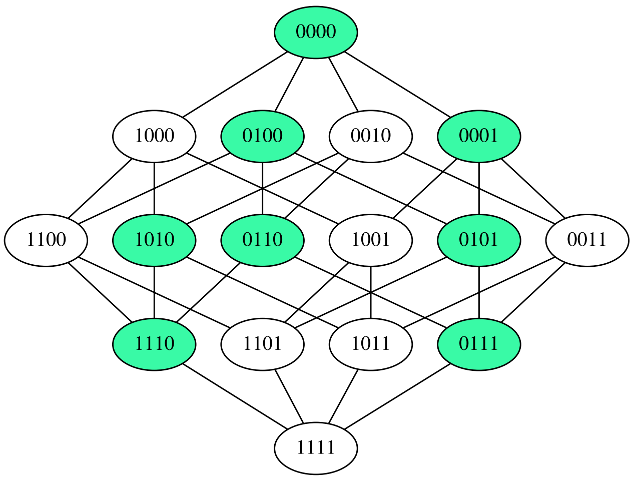
1 Introduction

State-of-the-art algorithms for k-SAT [17, 7] draw on an understanding of the structure of satisfying assignments of k-CNF formulas. In particular, progress in the design of fast k-SAT algorithms has benefited from an understanding of when satisfying assignments are close together and when they are far apart. Roughly speaking, it is useful to understand how many such assignments are "isolated" from others and "how isolated" they are. The following definitions make precise the notion of a satisfying assignment's degree of isolation from other satisfying assignments:

▶ **Definition 1.** *A clause C of a CNF formula is critical for a variable x_i and with respect to a satisfying assignment if flipping the value assigned to x_i makes C false.*

▶ **Definition 2.** *A satisfying assignment of a CNF formula is j-isolated if j variables have critical clauses.*

Consider the hypercube graph with vertex set $\{0,1\}^n$ where there is an edge between u and v if they have Hamming distance 1. If we associate assignments on n variables to elements of $\{0,1\}^n$ in the natural way, a j-isolated satisfying assignment corresponds to a vertex of the hypercube with j neighboring vertices which are not satisfying assignments. For example, in Figure 1, the satisfying assignment 0000 is 2-isolated since it has just 2 non-satisfying neighbors, 1000, and 0010. Furthermore, the clause $C = \neg x_1 \vee x_3 \vee x_4$ is critical for x_1 with respect to 0000 since C is true under the assignment 0000 but not 1000. Similarly, $x_1 \vee x_2 \vee \neg x_3$ is critical for x_3 with respect to 0000.



$$\varphi_{\text{example}} = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4)$$

Figure 1 Satisfying assignments of φ_{example} (drawn in green).

To understand the structure of j-isolated satisfying assignments, one can provide an encoding of them:

▶ **Definition 3.** *A prefix-free encoding of a set S over an alphabet Σ is a map $f : S \to \Sigma^*$ such that for no distinct $x, y \in S$ is $f(x)$ a prefix of $f(y)$.*

▶ **Definition 4.** *A randomized, prefix-free encoding of a set S over an alphabet Σ is a map $F : S \to \Sigma^*$ which is sampled uniformly at random from a family $\mathcal{F}$ of prefix-free encodings $f : S \to \Sigma^*$. The expected encoding length of $s \in S$ is $\mathbb{E}_{f \sim \mathcal{F}}[|f(s)|]$.*

The seminal *Satisfiability Coding Lemma* of Paturi, Pudlák, and Zane [19] gives an encoding scheme for j-isolated satisfying assignments of k-CNF formulas:

▶ **Theorem 5** (Satisfiability Coding Lemma)**.** *There exists a randomized, prefix-free encoding $\mathsf{ENC}^k_{\mathrm{PPZ}}(\varphi, \sigma)$ over the alphabet $\{0, 1\}$, such that any j-isolated satisfying assignment σ of a k-CNF φ has expected encoding length at most $n - j/k$.*

This lemma yields a bound on the number of j-isolated satisfying assignments and an $O(|\varphi|) \cdot 2^{n(1-\Omega(1/k))}$ time algorithm for k-SAT; obtaining an improved runtime of the form $O(|\varphi|) \cdot 2^{n(1-\omega(1/k))}$ remains a central open problem in complexity theory.

1.1 Our main result: A coding lemma for implicants

In this work, we generalize the PPZ Satisfiability Coding Lemma from satisfying assignments to the more general notion of *implicants* and use it to prove new results on the structure of satisfying assignments of k-CNF formulas.

▶ **Definition 6.** *Given a boolean function φ, an implicant of φ is a set of literals I whose conjunction implies φ. That is, every assignment that satisfies all the literals in I also satisfies φ. The size of an implicant is its cardinality.*

It will be convenient for us to view an implicant I as an map $\{x_1, ..., x_n\} \mapsto \{0, 1, *\}$, by defining for $i \in [n]$

$$I[x_i] = \begin{cases} 0 & \text{if } \neg x_i \in I \\ 1 & \text{if } x_i \in I \\ * & \text{otherwise.} \end{cases}$$

For example, every term in a DNF φ is an implicant of φ. As another example, $\{\neg x_1, x_2\} =$ 01**, is an implicant of size two for φ_{example} in Figure 1 since each of 0100, 0101, 0110, 0111 are satisfying.

▶ **Definition 7.** *Variables mapped to* 1 *or* 0 *are fixed by* I*, and variables mapped to* $*$ *are free. When all variables are fixed, the implicant* I *is simply a satisfying assignment; otherwise, we say that* I *is a partial implicant.*

▶ **Definition 8.** *An assignment* J *agreeing with* I *on* I*'s fixed variables is called a completion of* I*.*

Our goal of generalizing the Satisfiability Coding Lemma to implicants motivates the following definitions of critical clauses and j-isolation for implicants:

▶ **Definition 9.** *Fix an implicant* I *of a CNF formula* φ*. A clause* C *of* φ *is* critical *for a variable* x_i *with respect to* I *if* I *maps the literal involving* x_i *in* C *to 1 and every other literal in* C *to 0 or* $*$*. The implicant* I *is* j-isolated *if* j *variables have critical clauses.*

Recall that if I is a satisfying assignment and C is critical for x_i, flipping the value of x_i makes I into a falsifying assignment. On the other hand, if I is a partial implicant and C is critical for x_i, flipping the value of x_i makes *some* completion J of I into a falsifying assignment; indeed, let J be the completion of I that assigns 0 to all literals in C except for the literal containing x_i.

We give an encoding ENC^k which generalizes the encoding $\mathsf{ENC}^k_{\mathrm{PPZ}}$ from j-isolated assignments to all j-isolated implicants:

▶ **Theorem 10** (Implicant Coding Lemma)**.** *There exists a randomized, prefix-free encoding* $\mathsf{ENC}^k(\varphi, I)$ *over the alphabet* $\{0, 1, *\}$*, such that any* j*-isolated implicant* I *of a* k*-CNF* φ *has expected encoding length at most* $n - j/k$*.*

We use Theorem 10 to study prime implicants of CNF formulas.

1.2 Applications of Theorem 10: Prime Implicants of *k*-CNF formulas

▶ **Definition 11.** *A prime implicant is an implicant* I*, such that no strict subset of* I *is an implicant.*

To study prime implicants, we exploit a simple fact relating j-isolated implicants and size-j prime implicants, which is shown in the appendix:

▶ **Fact 12.** *Fix a* k*-CNF* φ*. A size-*j *prime implicant of* φ *is* j*-isolated, but the converse is not generally true.*

Prime implicants reveal information about the structure of the satisfying solutions of Boolean formulas. Indeed, if we associate assignments on n variables to vertices of the hypercube graph Q_n, then the prime implicants of a Boolean formula φ correspond to subcubes of solutions not contained within a larger subcube. More specifically, a size-j implicant corresponds to a $(n-j)$-cube of satisfying assignments. As an example, in Figure 1, $\{x_1, x_3, \neg x_4\} = 1{*}10$ is a prime implicant of size 3, however, the implicant $\{\neg x_1, x_2, x_3\} = 011{*}$ is not prime, since the subset $\{\neg x_1, x_2\} = 01{*}{*}$ is also an implicant.

We apply our implicant coding lemma to understand two basic questions regarding prime implicants, the first structural and the second algorithmic:

- First, how many prime implicants can a function have?
- Second, how quickly can one enumerate all prime implicants of a given function?

We study these questions as they relate to k-CNFs, giving near-optimal answers in both cases, and extending our results to m-clause CNFs and monotone k-CNFs.

1.2.1 Background and context

Historically, both questions have received a great deal of interest. Concerning the first question, extremal bounds for the class of all n-variable functions are fairly well-understood. Let $\#\text{PrimeImplicants}(\varphi)$ denote the number of prime implicants of a function φ. A classic result of Chandra and Markowsky shows a universal bound of $\#\text{PrimeImplicants}(\varphi) \leq O(3^n/\sqrt{n})$ for every n-variable boolean function φ [2]. Even earlier work of Dunham and Fridshal provides a near-matching lower bound, constructing an n-variable function that has $\Omega(3^n/n)$ prime implicants [5].

Turning to the second question, the algorithmic problem of obtaining the disjunction of all prime implicants of a given function φ, called the *Blake Canonical Form* (BCF) of φ, is also well-studied. Unlike a minimal equivalent DNF, the BCF of a formula is unique. Umans showed obtaining a minimal DNF to be Σ_2-complete [28], and a classic and well-known approach to obtaining a minimal DNF, the *Quine-McClusky algorithm* [14, 20, 22], proceeds by generating and then paring down the BCF. This algorithm obtains the BCF by converting the arbitrary formula to a CNF, drawing up a table of satisfying assignments, and finding prime implicants by iterated consensus. Alternative methods of obtaining the BCF include a reduction to integer or 0-1 programming [12, 16] and the use of SAT-solvers, directly or through a generalization [4, 9, 11]. While some of these approaches have been studied empirically, none offers any runtime guarantees better than the trivial bound of $O(|\varphi| \cdot 3^n)$ associated with an exhaustive search.

Prime implicants and the BCF have recently been of interest in the field of explainable AI, an important area of research that seeks to understand and interpret the decisions of complex machine learning models. Let $f : \{0,1\}^n \to \{0,1\}$ be a binary classifier with n Boolean features, let $x \in \{0,1\}^n$ be an input to the classifier, and let $f(x)$ be the decision of f on x. Then, if $f(x) = 1$, a prime implicant of f that has x as a completion is viewed as an explanation of f's "decision" to label x with 1; similarly, if $f(x) = 0$, a prime implicant of $\neg f$ that has x as a completion is viewed an explanation of f's "decision" to label x with 0 ([25], definition 5). An implicant I is viewed as a *sufficient* explanation for the model's decision, since toggling the features not contained in I does not change the model's decision. Prime implicants are then the *simplest* sufficient explanations, because no prime implicant can be made shorter. Darwiche and Hirth build on this work to define the notion of a *necessary* reason (i.e. explanation) for a decision, as well as a notion of *decision bias* [3]. Especially relevant to this paper's discussion is the same author's definition of a *complete* reason (i.e. explanation) for the decision made by a classifier f; Darwiche and Hirth define this to be the disjunction of all simplest sufficient explanations (prime implicants), i.e. the BCF. Further work in explainable AI computes these explanations (prime implicants) for certain machine learning models such as Neural Networks [9], Bayesian Networks [25], Boosted Trees [10], and monotone machine learning models [13].

1.2.2 Our results

We focus on understanding these questions in the case where φ is represented as a k-CNF formula, and show that our implicant coding lemma gives near-optimal answers to both. The problem of bounding the number of prime implicants for k-CNF formulas was first considered in the Ph.D. thesis of Talebanfard [26], who gave a non-trivial upper bound for *read-r* k-CNF formulas, i.e. those in which each variable occurs in at most r clauses:

▶ **Theorem 13** (Talebanfard [26]). *Let φ be an n-variable read-r k-CNF formula. Then*

$$\#\text{PrimeImplicants}(\varphi) \leq 3^{n(1-1/(rk))}.$$

To prove Theorem 13, Talebanfard [26] himself generalized the Satisfiability Coding Lemma to (d,k)-CSPs, constraint satisfaction problems in which variables can take d values and each constraint has at most k variables. Given a read-r k-CNF formula, Talebanfard generates a $(3, rk)$-CSP with isolated solutions that correspond exactly to prime implicants in the original formula. For constant-read k-CNFs, this gives a bound of $3^{n(1-\Omega(1/k))}$. In the conclusion of his thesis, Talebanfard asked whether this bound in fact holds for all k-CNF formulas, with no restriction on read. In [27], Talebanfard remarked that to answer this question, one would need to develop a generalization of the Satisfiability Coding Lemma that "can treat isolated solutions and prime implicants in general as the same objects."

This is precisely what our implicant coding lemma achieves, and thus our first main application is an affirmative answer to Talebanfard's question:

▶ **Theorem 14.** *There is a universal constant $c > 0$ such that for all n-variable k-CNF formulas φ,*

$$\#\text{PrimeImplicants}(\varphi) \leq n \cdot 3^{n(1-c/k)}.$$

Theorem 14 is near-optimal, as [26] also shows the existence of a k-CNF formula with $3^{n(1-O(\log k)/k)}$ prime implicants. (The construction of this formula is based on the lower bound construction of [5] discussed above.)

While Talebanfard showed Theorem 13 by generalizing the Satisfiability Coding Lemma to (d,k)-CSPs and mapping prime implicants of an input formula φ to isolated solutions of a corresponding (d,k)-CSP, we show that the Satisfiability Coding Lemma has a simple and direct generalization to j-isolated partial implicants, namely Theorem 10, from which the above result follows directly. Our approach is more straightforward than the one taken by Talebanfard. Indeed, there is no need to generate an auxiliary formula; while Talebanfard generates a CSP and bounds the isolated solutions of the CSP, we simply bound the j-isolated implicants of φ directly and observe that this is also a bound on the size-j prime implicants of φ. Moreover, unlike Talebanfard's generalized encoding, our encoding operates directly on prime implicants, and as a result, our proof of Theorem 14 is algorithmic in nature; we are able to leverage known de-randomization techniques to give an algorithm for enumerating the prime implicants of a given k-CNF formula, an algorithm which is the first to provide a non-trivial performance guarantee:

▶ **Theorem 15.** *There is a universal constant $c > 0$ and a deterministic algorithm which, given as input an n-variable k-CNF formula φ, computes the Blake Canonical Form of φ in time*

$$O(|\varphi| \cdot n^{2k+2} \cdot 3^{n(1-c/k)}).$$

The runtime of this algorithm is similarly near-optimal, given the aforementioned existence of a k-CNF formula with $3^{n(1-O(\log k)/k)}$ prime implicants.

Extensions: *m*-clause CNFs and monotone *k*-CNFs. We also give extensions of our results to m-clause CNFs and monotone k-CNFs (those in which no variable is negated). For m-clause CNFs, we give a similar bound and algorithm, where $\log m$ takes the place of k in the above results. For monotone k-CNFs, we obtain better bounds on the number of prime implicants and the runtime of our algorithm for k-CNFs, replacing the 3 in the base of the exponent by 2. This yields an upper-bound on the minimum DNF size of monotone k-CNF formulas, and we find that when $k = o(n/\log(n))$, for large n our bound is tighter than the best-known bound of $2^{n(1-1/(100k))}$, due to Miltersen, Radhakrishnan and Wegner [15]. Notably, the latter bound is shown using Håstad's switching lemma [23], which, like PPZ's Satisfiability Coding Lemma, underlies a number of results in complexity theory. By extending the Satisfiability Coding Lemma to give an encoding for partial implicants, we allow comparison between the Satisfiability Coding Lemma and Håstad's switching lemma, showing that the Satisfiability Coding Lemma is "more powerful" in this context, in the sense that its extension yields a tighter bound.

1.3 Overview of Proofs

We now give a prose overview of the ideas underlying our results. We begin with our encoding ENC^k for implicants, which generalizes the Satisfiability Coding Lemma's encoding $\mathsf{ENC}^k_{\mathrm{PPZ}}$ for satisfying assignments. For context and background, we give a brief description of the latter. At a high level, the encoding $\mathsf{ENC}^k_{\mathrm{PPZ}}$ iterates through the variables in uniform random order, writing down their values in the encoding and plugging in $I[x_i]$ for the i-th variable x_i in φ on iteration i. The formula φ is simplified with each iteration: clauses containing 1 (i.e. satisfied clauses) and literals assigned 0 (i.e. falsified literals) are deleted. The important exception is when x_i appears as a unit clause – a clause comprising either x_i or its negation – appears in φ. The simple but key observation is that the value for x_i does not have to be included in the encoding in this case since the decoder will be able to infer how x_i must be assigned to satisfy the unit clause. This saves precious space.

In more detail, if a variable x_i with a critical clause C comes after all variables in C in the random ordering, we are guaranteed that C simplifies to a unit clause comprising only the variable x_i in this process. The probability that a variable x_i comes after all variables in a clause of length k is $1/k$. With j isolated variables, at least j/k of them are eventually unit clauses in expectation, leading to the expected encoding length of $n - j/k$.

Our encoding ENC^k proceeds similarly, except now over the alphabet $\{0, 1, *\}$, and when $I[x_i] = *$, we *delete* literals x_i and $\neg x_i$ from φ and append $*$ to the encoding. The analysis of our encoding relies on our generalized notion of a critical clause being defined specifically so as to provide an analogous guarantee: a variable x_i with a critical clause C has probability at least $1/k$ of coming after all variables in C, in which case it need not be recorded.

Since every size-j prime implicants is j-isolated, our encoding algorithm gives rise to a bound on the total number of prime implicants of a k-CNF formula. We first bound the number of size-j prime implicants, $\#\mathrm{PrimeImplicants}_j(\varphi)$, for each j by $3^{n(1-c/k)}$ for some universal constant c, and then sum over $j \in [n]$. Our encoding algorithm also gives rise to an algorithm for generating the Blake Canonical Form. An inefficient strategy is to decode every ternary string of length at most $n - j/k$ using every permutation of variables, since every prime implicant is encoded as some such string for some such permutation. We obtain our near-optimal algorithm by showing that our guarantees on encoding length only requires k-wise independent permutations rather than fully random ones.

2 Implicant Coding Lemma

In this section, we prove Theorem 10. We first state our encoding and decoding algorithms in prose, following up with formal presentations and a proof of the lemma.

The encoder $\mathsf{ENC}^k_\pi(\varphi, I)$ is parameterized by a random permutation π, which it uses to relabel the variables in φ and I. We initialize the encoding to be the empty string and then, for each $i \in [n]$, do the following. If x_i is in some unit clause of φ, do nothing. Otherwise, append the value to which x_i is mapped to the encoding. We then replace φ with $\varphi(i, I)$, the result of substituting $I[x_i]$ for x_i in φ. Thus if x_i is free, all literals containing it are deleted, and if x_i is fixed, all literals which contain x_i and are mapped to 0 are deleted, and all other literals containing x_i are mapped to 1, so that their clauses are satisfied and thus deleted. Formally:

Algorithm 1 Encoding algorithm $\mathsf{ENC}^k_\pi(\varphi, I)$ for k-CNFs.

```
relabel $\varphi$ and $I$ according to $\pi$
initialize an empty string $y$
$\varphi_0 = \varphi$
for $i$ in 1,...,$n$ do
    if $x_i$ not in a unit clause in $\varphi_{i-1}$ then
        append $I[x_i]$ to $y$
    end
    $\varphi_i = \varphi_{i-1}(i, I)$
end
return $y$
```

The decoder $\mathrm{DEC}^k_\pi(\varphi, y)$ again relabels φ according to π, and then for each $i \in [n]$ does the following. If there is no unit clause, simply read the value for x_i off of the encoding y. Otherwise, if there is a unit clause containing a literal x_i or $\neg x_i$, the decoder sets the variable x_i to satisfy the clause instead of reading another value from the encoding. In both cases, the decoder assigns the variable x_i and simplifies the formula φ before proceeding to the next iteration. Formally:

Algorithm 2 Decoding algorithm $\mathrm{DEC}^k_\pi(\varphi, y)$ for k-CNFs.

```
relabel $\varphi$ according to $\pi$
$\psi_0 = \varphi$
$J$ = empty implicant
for $i$ in 1,...,$n$ do
    if $x_i$ not a unit clause of $\psi_{i-1}$ then
        $J[x_i] =$ read next symbol of $y$
    else
        $J[x_i] = 1$ if unit clause is $x_i$ and 0 if the unit clause is $\neg x_i$
    end
    $\psi_i = \psi_{i-1}(i, J)$
end
Relabel $J$ according to $\pi^{-1}$
return $J$
```

We now prove Theorem 10:

Proof.

Prefix-free. To show this, we will show that the decoding algorithm inverts the encoding algorithm. That is, $\mathrm{DEC}_\pi^k(\varphi, \mathsf{ENC}_\pi^k(\varphi, I)) = I$ for all implicants I of φ. Fix such an implicant I and its encoding $y = \mathsf{ENC}_\pi^k(\varphi, I)$. It suffices to show by induction that the following hold at the beginning of iteration i.

1. The restriction of I to the first $i-1$ variables is precisely the value of J constructed by $\mathrm{DEC}_\pi^k(\varphi, y)$ so far.
2. The encoder and decoder have inspected the same number of characters in the encoding, moving from left to right.
3. $\varphi_{i-1} = \psi_{i-1}$.

The base case holds trivially. Assuming the claims hold at the start of iteration i, we will prove the claims hold at the start of iteration $i+1$. Consider the ith iteration. By the inductive hypothesis $\varphi_{i-1} = \psi_{i-1}$, so both the encoding and decoding algorithms move into the same case, depending on the presence of a unit clause.

In the if case, the encoder adds the encoding precisely the value used by the decoder, by the inductive hypothesis on the second claim. This establishes the first claim. Both the encoder and decoder use one character of the encoding, so the second claim holds as well. The third claim then follows from the first claim and the fact that $\varphi_{i-1} = \psi_{i-1}$.

In the else case, since I is an implicant, only one of x_i and $\neg x_i$ can be a unit clause in $\psi_{i-1} = \varphi_{i-1}$. Therefore this uniquely (and correctly) determines the value used by the decoder, establishing the first claim. Neither the encoder nor the decoder uses a character of the encoding, so the second claim holds as well. Finally, the third claim follows from the first claim and the inductive hypothesis on the third claim.

Therefore at the end of iteration i, or the beginning of iteration $i+1$, each claim holds, completing the induction.

Encoding length. It suffices to show that for any j-isolated implicant I, the expected encoding length of I under ENC_π^k over uniformly random permutations π is at most $n - j/k$. For then, to establish Theorem 10, one defines ENC^k to sample π uniformly at random and run ENC_π^k.

Let I be any j-isolated implicant of φ, and π be a uniformly random permutation on $[n]$. If in the ordering after application of π, a variable x_i comes after all other variables appearing in its critical clause, then on iteration i, this clause remains unsatisfied and is a unit clause containing x_i. Thus ENC_π^k passes over the variable x_i, saving space by not adding $I[x_i]$ to the encoding. The probability that x_i comes after all other variables in its critical clause is at least $1/k$, there being at most k variables in the clause and π being uniform. Since I is j-isolated, there are at least j critical clauses, so that by linearity of expectation, at least j/k fixed variables are skipped, giving an encoding of expected length at most $n - j/k$. ◀

It will serve later discussion to have a bound on the runtime of these algorithms:

▶ **Lemma 16.** *ENC_π^k and DEC_π^k each run in time $O(|\varphi| \cdot n)$.*

Proof. We consider only ENC_π^k, as the proof for DEC_π^k is the same. It suffices to show that each of the iterations i for $i \in [n]$ takes time $O(|\varphi|)$. On iteration i, we attempt to determine whether the variable x_i appears in a unit clause of φ_{i-1} and update the formula. Each of these can be done with a linear scan through the formula, requiring time $|\varphi_{i-1}| \le |\varphi|$. ◀

3 Bounding the Number of Prime Implicants for k-CNFs

Let φ be any k-CNF. In this section, we use our encoding lemma to give a new upper bound on $\#\text{PrimeImplicants}(\varphi)$. Our bound implies that for all n and all $k \geq 2$,

$$\#\text{PrimeImplicants}(\varphi) \leq n \cdot 3^{n(1-.358/k)}.$$

Our bound requires the following key fact, the proof of which is deferred to the appendix.

▶ **Fact 17.** *If* ENC *is a prefix-free encoding of S into strings formed from the alphabet {0,1,*}, and* ENC *has average code length ℓ, then $|S| \leq 3^{\ell}$.*

Since size-j prime implicants are j-isolated, together with Theorem 10, the above fact gives a bound on $\#\text{PrimeImplicants}_j(\varphi)$, the number of size-$j$ prime implicants of φ:

▶ **Lemma 18.** *Let φ be a k-CNF. $\#\text{PrimeImplicants}_j(\varphi) \leq 3^{n-j/k}$.*

Proof. By Theorem 10, the average description length of a j-isolated implicant (over the random choice in permutation) is at most $n - j/k$. This is also true when the average is taken over all j-isolated implicants and all permutations. Hence, there exists a permutation π such that the average description length under the coding ENC_π^k is at most $n - j/k$ when the average is taken over all j-isolated implicants. Then as ENC_π^k is prefix-free, from Fact 17, the number of j-isolated implicants, and hence prime implicants of size j, is at most $3^{n-j/k}$. ◀

We now bound the total number of prime implicants for a k-CNF φ. Our bound invokes the 3-ary entropy function $H_3(x) = x\log_3(2) - x\log_3(x) - (1-x)\log_3(1-x)$:

▶ **Theorem 14.** *Let φ be a k-CNF. Let $c \in [0, 2/3]$ satisfy $H_3(c) \leq 1 - c/k$. Then*

$$\#\text{PrimeImplicants}(\varphi) \leq [n(1-c)+1] \cdot 3^{n(1-c/k)}.$$

In particular, for all $k \geq 2$, we can take $c = 0.358$.

Proof of Theorem 14. Let φ be a k-CNF. If $j \geq cn$, by Lemma 18,

$$\#\text{PrimeImplicants}_j(\varphi) \leq 3^{n-j/k} \leq 3^{n(1-c/k)}.$$

Thus

$$\sum_{j \geq nc} \#\text{PrimeImplicants}_j(\varphi) \leq n(1-c) \cdot 3^{n(1-c/k)}. \tag{1}$$

For $j < nc$, we observe that $\#\text{PrimeImplicants}_j(\varphi)$ is at most the number of ways of choosing locations for j fixed variables and then assigning them, so that

$$\sum_{j<nc} \#\text{PrimeImplicants}_j(\varphi) \leq \sum_{j<nc} \binom{n}{j} \cdot 2^j \leq 3^{n \cdot H_3(c)}, \tag{2}$$

the final inequality following from a familiar bound on the volume of a Hamming ball of radius nc (see, for example, Proposition 3.3.3 in [6]). Together, inequalities (1) and (2) imply the first result in the statement of the theorem, given our choice of c.

To see that the second statement in the theorem follows from the first, we observe that our choice of c is increasing in k, so that the desired bound holds for c with $H_3(c) \leq 1 - c/2$, and this is satisfied for $c = .358$. ◀

4 Obtaining Prime Implicants of k-CNFs

This section gives a deterministic algorithm for obtaining all the prime implicants of a k-CNF. The proof proceeds by a now-familiar method of de-randomization that can be applied to randomized algorithms which, like the encoding algorithm given in Section 2, depend only on k-wise (not fully) independent random choices; the method is discussed for example in Noga Alon and Joel Spencer's *The Probabilistic Method* [1], and it is used in Daniel Rolf's deterministic algorithm for unique k-SAT [24] and in Paturi, Pudlák, and Zane's deterministic algorithm for k-SAT [18].

We begin by noting that one can quickly check whether a given implicant is prime. The proof of this fact is deferred to the appendix:

▶ **Fact 19.** *Fix a k-CNF φ. One can confirm in time $O(|\varphi| \cdot n)$ that a partial assignment I is a prime implicant.*

We now give an algorithm for finding all size-j prime implicants. At a high level, the idea is to try to decode every string s in $\{0, 1, *\}^{n-j/k+1}$ using every permutation π. This works because for each size-j prime implicant I, there is some permutation π such that ENC_π^k maps I to a prefix of some such s, and the encoding is prefix-free. The only problem is that there are many permutations π; to make the algorithm faster, we will pick a smaller set of permutations π that gives the same guarantee. We include in the appendix a proof of the following standard fact:

▶ **Fact 20.** *Fix k and a prime power N. There exist random variables $X_1, ..., X_N$ uniformly distributed over N which are k-wise independent, defined over a probability space of size N^k.*

▶ **Lemma 21.** *There exists a set Π of permutations of size $O(n^{2k})$ such that for any size-j prime implicant I, the expected encoding length of ENC_π^k over $\pi \in \Pi$ is at most $n - j/k + 1$.*

Proof. Let N be the smallest power of 2 greater than n^2. Use Fact 20 to obtain $X_1, ..., X_N$ that are k-wise independent and uniformly distributed over $[N]$. Keep only $\mathbf{X} = (X_1, ..., X_n)$ and discard the rest.

Define a permutation $\pi_{\mathbf{X}} : [n] \to [n]$ such that $\pi_{\mathbf{X}}(i)$ maps i to X_i's rank in $\{X_1, ..., X_n\}$, where ties are broken by the subscripts of the variables:

$$\pi_{\mathbf{X}}(i) = |\{X_j : X_j < X_i\}| + |\{X_j : X_j = X_i \text{ and } j \leq i\}|.$$

Without loss of generality, suppose x_1 is a critical variable with a critical clause containing the variables $x_1, x_2, ..., x_k$. Note that x_1 appears last in this clause with respect to $\pi_{\mathbf{X}}$ if $X_1, ..., X_k$ are distinct and $X_1 > X_i$ for $i \in [k]$. Thus the likelihood that x_1 comes last in the critical clause according to $\pi_{\mathbf{X}}$ is at least

$$\mathbb{P}[X_1, ..., X_k \text{ distinct}] \cdot \mathbb{P}[X_1 > X_2, ..., X_k | X_1, ..., X_k \text{ distinct}].$$

Since $X_1, ..., X_N$ are k-wise independent and uniformly distributed over $[N]$, it follows that $(X_1, ..., X_k)$ is distributed uniformly over $[N]^k$. Thus, $\Pr[X_i = X_j] = 1/N$ for distinct $i, j \in [k]$. By a union bound over all pairs, we have $\Pr[X_1, ..., X_k \text{ are all distinct}] \geq 1 - \frac{k^2}{N}$. Furthermore, conditioned on $X_1, ..., X_k$ being distinct, the probability that $X_1 > X_2, ..., X_k$ is $1/k$. Thus the likelihood that x_1 comes last in the critical clause is at least $\frac{1}{k}(1 - \frac{k^2}{N})$. By linearity of expectation, the expected coding length is then at most

$$n - \frac{j}{k}\left(1 - \frac{k^2}{N}\right) \leq n - \frac{j}{k} + \frac{jk}{n^2} \leq n - \frac{j}{k} + 1.$$

Let Π contain all possible $\pi_{\mathbf{X}}$, of which there are $N^k = O(n^{2k})$. ◀

We now consider the faster version of the naive algorithm described above and bound its runtime:

Algorithm 3 Find size-j prime implicants of a k-CNF.

```
PI = $\emptyset$
$\Pi$ = the permutations given by Lemma 21
for $\pi \in \Pi$ and $s \in \{0, 1, *\}^{n-j/k+1}$ do
    $I = \mathrm{DEC}^k_\pi(s)$
    if $I$ is a prime implicant then
        add $I$ to the set PI
    end
end
Return PI
```

▶ **Lemma 22.** *The above algorithm runs in time $O(|\varphi| \cdot n^{2k+1} \cdot 3^{n-j/k})$.*

Proof. By Lemmas 16 and 19, we repeat $O(|\varphi| \cdot n)$ operations for $O(n^{2k} \cdot 3^{n-j/k})$ iterations. ◀

We can now show Theorem 15:

▶ **Theorem 15.** *Let φ be a k-CNF. Then for $H_3(c) \leq 1 - c/k$, we deterministically obtain the BCF in time*

$$O(|\varphi| \cdot n^{2k+2} \cdot 3^{n(1-c/k)}).$$

Proof of Theorem 15. It suffices to show that one can find all size-j prime implicants in time $O(|\varphi| \cdot n^{2k+1} \cdot 3^{n(1-c/k)})$, since summing over $j \in [n]$ then gives the desired result. As in the proof of Theorem 14, we consider two cases. If $j \geq nc$, we use Algorithm 3 and apply Lemma 22. Otherwise, $j < nc$. Then we enumerate over all partial implicants with at most j fixed coordinates, and, using Fact 19, check if each is a prime implicant. To do this for all $j < nc$ takes time

$$O(|\varphi| \cdot n) \cdot \sum_{j<nc} \binom{n}{j} \cdot 2^j \leq O(|\varphi| \cdot n) \cdot 3^{n(1-c/k)},$$

by the same reasoning used in the proof of Theorem 14. ◀

5 Extensions

In this section, we lay out extensions of our results to m-clause and monotone CNFs.

5.1 *m*-clause CNFs

We give a coding lemma for m-clause CNFs using a width-reduction technique of Hirahara [8]

▶ **Lemma 23.** *There exists a randomized, prefix-free encoding $\mathsf{ENC}^m(\varphi, I)$ over the alphabet $\{0, 1, *\}$, such that any j-isolated implicant I of an m-clause formula φ has expected encoding length at most $n - j/k + 1$, where $k = \frac{2+\log_3(m)}{1-\log_3(2)}$.*

As before, we first describe each algorithm in prose, following up with a more formal presentation. Fix an m-clause CNF φ, and j-isolated implicant I. Set $k = \frac{2+\log_3(m)}{1-\log_3(2)}$. For a clause C, denote by C^k the truncation of C to the first k variables in C (with respect to the ordering induced by π) and denote by φ^k the conjunction of C^k for all clauses C in φ. Notice that φ^k is a k-CNF.

The encoding algorithm, ENC_π^m, is a recursive procedure. In the base case, I is an implicant of φ^k. We then set the encoding to be $1 \circ \mathsf{ENC}_\pi^k(\varphi^k, I)$ where the 1 is used to mark the case split. We will argue that I must be at least j-isolated with respect to φ^k and thus we can use the ENC_π^k to guarantee an expected encoding length of at most $n - j/k$

In the recursive case, I is not an implicant of φ^k. Here we select some clause C_i^k such that I does not assign any literal in C_i^k to 1. We then communicate the case split, the (index of) the selected clause, and the assignments I makes to the variables in the selected clause. Formally, we append to our encoding $0 \circ \langle i \rangle_3 \circ \langle C_i^k, I \rangle_3$, where $\langle C_i^k, I \rangle_3$ is some canonical, ternary representation of C_i^k, with $I[x_i]$ substituted for each variable x_i in C_i^k. Note that our choice in C_i^k ensures that I assigns all the literals in C_i^k to 0 or $*$ which requires at most $k \log_3(2)$ ternary digits to represent. Finally, we simplify the formula by assigning all the variables in C_i^k according to I (as is done in ENC^k) and call the encoding algorithm recursively on the simplified formula and the remaining partial assignment. Letting S denote the variables appearing in C_i^k, we let $I|_S$ denote the restriction of I to the set S, $I|_{\bar{S}}$ denote the restriction of I to the variables not in S, and $\varphi(I|_S)$ denote the simplified formula.

Algorithm 4 Encoding algorithm $\mathsf{ENC}_\pi^m(\varphi, I)$ for m-clause CNFs.

```
initialize an empty string $y$
let $C_1, ..., C_m$ enumerate the clauses in $\varphi$
if $I$ is an implicant of $\varphi^k$ then
    append $1 \circ \mathsf{ENC}_\pi^k(\varphi^k, I)$ to $y$
    break
else
    fix some truncated clause $C_i^k$ such that $I$ does not assign any variable in $C_i^k$ to 1.
    let $S$ contain the variables appearing in $C_i^k$
    append $0 \circ i \circ \langle C_i^k, I \rangle_3 \circ \mathsf{ENC}_\pi^m(\varphi(I|_S), I|_{\bar{S}})$ to $y$
```

The decoding algorithm DEC_π^m inverts the encoding as follows. It reads the first bit to determine which case split was taken during the encoding. If it was 1, then simply return $\mathsf{DEC}_\pi^k(\varphi^k, y)$. If it was 0, read i and I's assignments to C_i^k from the next $\log_3(m)$ and $k \log_3(2)$ bits respectively. From this, we can determine I's assignments to the k variables in C_i^k. Then, we simplify φ using these assignments and make a recursive call on the simplified formula and the remainder of the encoding.

Algorithm 5 Decoding algorithm $\mathsf{DEC}_\pi^m(\varphi, y)$ for m-clause CNFs.

```
initialize an empty implicant $I$
case = read first bit of $y$
if $case = 1$ then
    return $\mathsf{DEC}_\pi^k(\varphi^k, y)$
else
    $i$ = read the next $\log_3(m)$ symbols of $y$
    fill $I$ by the partial assignment defined by the next $k \log_3(2)$ symbols
    recursively call $\mathsf{DEC}_\pi^m(\varphi(I), y)$
```

Proof of Lemma 23. First we will show the following.

▷ Claim 24 (Base case). If I is an implicant of φ^k, then it must be j'-isolated for $j' \geq j$

▷ Claim 25 (Recursive case). If I is not an implicant of φ^k, there is some clause C_i^k with exactly k literals such that if S is the set of variables appearing in C_i^k, $I|_{\bar{S}}$ is a j'-isolated implicant of $\varphi(I|_S)$ for $j' \geq j - k$.

For Claim 24, let C be a critical clause for x_i in φ with respect to I. Then I maps every literal to 0 or $*$ except the literal in C involving x_i. Since I is still an implicant of φ^k, C^k must contain this literal. Furthermore, since literals may only be deleted from C to get C^k, the literal involving x_i remains the only literal assigned 1 by I. Thus, C^k is again critical for x_i in φ^k with respect to I. Since critical clauses remain critical, at least j variables have critical clauses in φ^k and thus I is a j'-isolated implicant of φ^k for $j' \geq j$.

For Claim 25, let C_i^k be a clause such that I does not assign any variable in C_i^k to 1. Since I is an implicant of φ, there was a literal in C_i that was assigned 1, which is now missing. Thus, C_i originally had more than k literals and the truncation C_i^k has exactly k literals. Let S be the variables appearing in C_i^k. Let C be a critical clause for some variable $x_j \notin S$ of φ. Since x_j is not assigned in $\varphi(I|_S)$, and the rest of the variables are assigned consistently with I, C remains critical for x_j in $\varphi(I|_S)$. Since $|S| = k$, and there were j variables with critical clauses in φ, there are at least $j - k$ variables with critical clauses in $\varphi(I|_S)$ with respect to $I|_{\bar{S}}$.

This concludes the proof of the two claims. We will now show that ENC_π^m is prefix-free and has the desired expected encoding length.

Prefix-free. We will show that $\mathsf{DEC}_\pi^m(\varphi, \mathsf{ENC}_\pi^m(\varphi, I)) = I$ for all implicants I of φ. The recursive case is inverted by DEC_π^m since the ordering of the clauses and variables are fixed. Then, ENC_π^m eventually reaches the base case since k variables are removed at every recursive call and I remains an implicant of φ at the start of each call to ENC_π^m step by Claim 25. The base case follows from the result on ENC_π^k.

Encoding length. We claim by induction on the number of recursive calls that the length of the encoding is in expectation (with respect to π) at most $n - j/k + 1$. For the base case, by Claim 24, we have that I is a j'-isolated implicant of φ^k where $j' \geq j$, so the encoding length is 1 more than the expected encoding length of $\mathsf{ENC}_\pi^k(\varphi^k, I)$, which by Theorem 10 is at most $n - j'/k \leq n - j/k$, as required.

If there is a recursive call, by Claim 25, we have that $\varphi(I|_S)$ is a formula with $n - k$ variables, and $I|_{\bar{S}}$ is a j'-isolated implicant of $\varphi(I|_S)$ with $j' \geq j - k$. Then by the inductive hypothesis, the average encoding length of $I|_{\bar{S}}$ is at most $n - k - (j - k)/k + 1$. Thus, the average encoding length of I in this case is at most

$$\begin{aligned}
&1 + \log_3(m) + k\log_3(2) + (n-k) - (j-k)/k + 1 \\
&= 1 + \log_3(m) + k\log_3(2) + n - k - j/k + 2 \\
&= n - j/k + 1,
\end{aligned}$$

where the last equality follows from our choice of k, which gives

$$\log_3(m) + k\log_3(2) - k + 2 = 0. \quad ◀$$

Let $\#\text{PrimeImplicants}_j(\varphi)$ denote the number of size-j prime implicants. As in the k-CNF case, we get a bound from the encoding lemma, together with Fact 17:

▶ **Lemma 26.** *For an m-clause CNF, we have $\#\mathrm{PrimeImplicants}_j(\varphi) \leq 3^{n-j/k+1}$, where $k = \frac{2+\log_3(m)}{1-\log_3(2)}$.*

We then repeat the analysis in the proof of Theorem 14:

▶ **Theorem 27.** *Let φ be an m-clause CNF, $k = \frac{2+\log_3(m)}{1-\log_3(2)}$ and $H_3(c) = 1 - c/k$. For m-clause CNFs with $m \geq 3$,*

$$\#\mathrm{PrimeImplicants}(\varphi) \leq [n(1-c)+1] \cdot 3^{n(1-c/k)+1}.$$

Replacing the decoder DEC_π^k with DEC_π^m in the k-CNF construction gives immediately:

▶ **Theorem 28.** *Let φ be an m-clause CNF. Let $k = \frac{2+\log_3(m)}{1-\log_3(2)}$, and $H_3(c) = 1 - c/k$. We deterministically obtain the BCF in time*

$$O(|\varphi| \cdot n^{2\cdot k+2} \cdot 3^{n(1-c/k)+1}).$$

5.2 Monotone *k*-CNFs

We get a tighter bound and a faster algorithm in the monotone case, using the following fact, the proof of which is deferred to the appendix.

▶ **Fact 29.** *If φ is monotone, every prime implicant assigns every variable to either 1 or $*$.*

It follows from this fact that 0 never appears in any encoding of a prime implicant in the monotone case. We thus achieve a shorter encoding length, using the following analog of Fact 17 shown by Paturi, Pudlák and Zane [19]:

▶ **Fact 30.** *If ENC is a prefix-free encoding of S into strings formed from the alphabet {1,*}, and ENC has average code length ℓ, then $|S| \leq 2^\ell$.*

Applying this to the analysis of our encoding scheme, we get the following.

▶ **Corollary 31.** *For a monotone k-CNF, the number of size-j prime implicants is at most $2^{n-j/k}$.*

▶ **Theorem 32.** *If φ is a monotone k-CNF and $c \in (0, 1/2]$ is such that $H_2(c) \leq 1 - c/k$, then*

$$\#\mathrm{PrimeImplicants}(\varphi) \leq n \cdot 2^{n(1-c/k)}.$$

In particular, for all $k \geq 2$, one can take $c = .282$.

Proof. We will show that $\#\mathrm{PrimeImplicants}_j(\varphi) \leq 2^{n(1-c/k)}$ for each j. If $j \geq cn$, then Corollary 31 gives

$$\#\mathrm{PrimeImplicants}_j(\varphi) \leq 2^{n(1-c/k)}.$$

Otherwise, $j < cn$. In this case we use the fact that there are at most $\binom{n}{j}$ prime implicants; prime implicants for monotone formulas contain only 1 and $*$, we only need to pick j out of n variables to assign 1. Using an entropy bound on the binomial coefficient,

$$\#\mathrm{PrimeImplicants}_j(\varphi) \leq \binom{n}{j} \leq 2^{H_2(c)\cdot n}.$$

By our choice in c, $\#\mathrm{PrimeImplicants}_j(\varphi) \leq 2^{n(1-c/k)}$. ◀

Denote by dnfsize(φ) the size of the shortest DNF equivalent to φ. We now show that Theorem 32 also gives a bound on the dnfsize for monotone k-CNFs. To that end, we recall that an *essential* prime implicant is one with a completion which is the completion of no other prime implicant, and we use an observation due to Quine [21], the proof of which is deferred to the appendix.

▶ **Fact 33.** *If φ is a monotone formula, then:*
1. *All prime implicants of φ are essential.*
2. *The Blake Canonical Form of φ is the shortest DNF equivalent to φ*

Combining Fact 33 and Theorem 32 yields:

▶ **Corollary 34.** *If φ is a monotone k-CNF, then* $\mathrm{dnfsize}(\varphi) \leq n \cdot 2^{n(1-c/k)}$*, for $c = 0.282$.*

Thus when $k = o(n/\log(n))$, for large enough n the factor of n is negligible, and the above bound is tighter than the best-known bound of $2^{n(1-1/(100k))}$, shown by Miltersen, Radhakrishnan and Wegner [15]. We note that one can also obtain Corollary 31 from the Satisfiability Coding Lemma; since every prime implicant is essential and so maps to some unique satisfying assignment, the satisfying assignments can be partitioned by their j-sensitivity and then bounded using the lemma. However, unlike the Satisfiability Coding Lemma, our construction gives rise to an algorithm for obtaining the minimal DNF of a monotone k-CNF. Indeed, the construction used in the general k-CNF case and the analysis in Theorem 32 yield:

▶ **Theorem 35.** *Let φ be a monotone k-CNF. Then for $c \in (0, 1/2]$ such that $H_2(c) = 1 - c/k$, we deterministically obtain the BCF (or, equivalently, the minimal DNF) in time*

$$O(|\varphi| \cdot n^{2k+1} \cdot 2^{n(1-c/k)}).$$

In particular, for all $k \geq 2$, one can take $c = .282$.

References

1 Noga Alon and Joel H Spencer. *The probabilistic method.* John Wiley & Sons, 2004.

2 Ashok K Chandra and George Markowsky. On the number of prime implicants. *Discrete Mathematics*, 24(1):7–11, 1978.

3 Adnan Darwiche and Auguste Hirth. On the reasons behind decisions. In Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, editors, *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 712–720. IOS Press, 2020. doi:10.3233/FAIA200158.

4 David Déharbe, Pascal Fontaine, Daniel Le Berre, and Bertrand Mazure. Computing prime implicants. In *2013 Formal Methods in Computer-Aided Design*, pages 46–52. IEEE, 2013.

5 B Dunham and R Fridshal. The problem of simplifying logical expressions. *The Journal of Symbolic Logic*, 24(1):17–19, 1959.

6 Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. *Essential Coding Theory.* University at Buffalo, 2012.

7 Thomas Dueholm Hansen, Haim Kaplan, Or Zamir, and Uri Zwick. Faster k-sat algorithms using biased-PPSZ. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 578–589, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316359.

8 Shuichi Hirahara. A duality between depth-three formulas and approximation by depth-two. *arXiv preprint*, 2017. arXiv:1705.03588.

9 Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. Abduction-based explanations for machine learning models. In Pascal Van Hentenryck and Zhi-Hua Zhou, editors, *Proceedings of AAAI19-Thirty-Third AAAI conference on Artificial Intelligence*, number 1 in Proceedings of the AAAI Conference on Artificial Intelligence, pages 1511–1519, United States of America, 2019. Association for the Advancement of Artificial Intelligence (AAAI). AAAI Conference on Artificial Intelligence 2019, AAAI 2019 ; Conference date: 27-01-2019 Through 01-02-2019. doi:10.1609/aaai.v33i01.33011511.

10 Alexey Ignatiev, Nina Narodytska, and João Marques-Silva. On validating, repairing and refining heuristic ML explanations. *CoRR*, abs/1907.02509, 2019. arXiv:1907.02509.

11 Said Jabbour, Joao Marques-Silva, Lakhdar Sais, and Yakoub Salhi. Enumerating prime implicants of propositional formulas in conjunctive normal form. In *European Workshop on Logics in Artificial Intelligence*, pages 152–165. Springer, 2014.

12 Vasco M Manquinho, Paulo F Flores, João P Marques Silva, and Arlindo L Oliveira. Prime implicant computation using satisfiability algorithms. In *Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence*, pages 232–239. IEEE, 1997.

13 Joao Marques-Silva, Thomas Gerspacher, Martin C Cooper, Alexey Ignatiev, and Nina Narodytska. Explanations for monotonic classifiers. In *International Conference on Machine Learning*, pages 7469–7479. PMLR, 2021.

14 Edward J McCluskey. Minimization of boolean functions. *The Bell System Technical Journal*, 35(6):1417–1444, 1956.

15 Peter Bro Miltersen, Jaikumar Radhakrishnan, and Ingo Wegener. On converting CNF to DNF. *Theoretical computer science*, 347(1-2):325–335, 2005.

16 Luigi Palopoli, Fiora Pirri, and Clara Pizzuti. Algorithms for selective enumeration of prime implicants. *Artificial Intelligence*, 111(1-2):41–72, 1999.

17 Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-sat. *J. ACM*, 52(3):337–364, May 2005. doi:10.1145/1066100.1066101.

18 Ramamohan Paturi, Pavel Pudlák, Michael E Saks, and Francis Zane. An improved exponential-time algorithm for k-sat. *Journal of the ACM (JACM)*, 52(3):337–364, 2005.

19 Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 566–574. IEEE, 1997.

20 Willard V Quine. The problem of simplifying truth functions. *The American mathematical monthly*, 59(8):521–531, 1952.

21 Willard V Quine. Two theorems about truth-functions. *Journal of Symbolic Logic*, 1954.

22 Willard V Quine. A way to simplify truth functions. *The American mathematical monthly*, 62(9):627–631, 1955.

23 Alexander A Razborov. Bounded arithmetic and lower bounds in boolean complexity. In *Feasible Mathematics II*, pages 344–386. Springer, 1995.

24 Daniel Rolf. Derandomization of ppsz for unique-k-sat. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 216–225. Springer, 2005.

25 Andy Shih, Arthur Choi, and Adnan Darwiche. A symbolic approach to explaining bayesian network classifiers. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 5103–5111. AAAI Press, 2018.

26 Navid Talebanfard. *On the Combinatorics of SAT and the Complexity of Planar Problems*. PhD thesis, Department Office Computer Science, Aarhus University, 2014.

27 Navid Talebanfard. On the structure and the number of prime implicants of 2-CNFs. *Discrete Applied Mathematics*, 200:1–4, 2016.

28 Christopher Umans. The minimum equivalent DNF problem and shortest implicants. *Journal of Computer and System Sciences*, 63(4):597–611, 2001.

A Proofs of Facts

In this appendix, we include proofs of the facts used in the paper.

Proof of Fact 12. To see that I is j-isolated, note that each of variables x_i for which $I[x_i] \in \{0,1\}$ must have some critical clause, since otherwise all clauses in which x_i appears are twice satisfied, so that the set of literals $I \setminus \{x_i\}$ is an implicant, contradicting the primality of I. To see that the converse is not generally true, consider $J = \{x_i\} \cup I$ for any size-j prime implicant I and any $x_i \notin I$. Then J is j'-isolated for some $j' \leq j$, and J is not a prime implicant, and in particular is not a size-j' prime implicant. ◀

Proof of Fact 17. Let ℓ_I denote the length of $\mathsf{ENC}(I)$ for $I \in S$. Then $\ell = \sum_{I\in S} \ell_I/|S|$ is the average encoding length. Since ENC is prefix-free, the events E_I of rolling a three-sided die and stopping upon generating I are disjoint, and their probabilities sum to 1: $\sum_{I\in S} 3^{-\ell_I} \leq 1$. We wish to show that $\ell \geq \log_3 |S|$:

$$\begin{aligned}\ell - \log_3 |S| &= \sum_{I\in S} \frac{1}{|S|}(\ell_I - \log_3 |S|) = -\sum_{I\in S} \frac{1}{|S|}(\log_3 3^{-\ell_I} + \log_3 |S|) \\ &= -\sum_{I\in S} \frac{1}{|S|} \log_3(|S| 3^{-\ell_I}) \geq -\log_3 \left(\sum_{I\in S} 3^{-\ell_I}\right) \geq 0,\end{aligned}$$

where the penultimate inequality follows from concavity of log. ◀

Proof of Fact 19. We wish to check whether a partial assignment I is a prime implicant of φ. First, compute $\varphi(I)$, the result of replacing each variable x_i in φ with $I[x_i]$. Then, iterate over all clauses in $\varphi(I)$. If any clause in $\varphi(I)$ is all 0's, reject I, which is not a satisfying assignment; otherwise, delete all 0's appearing in the clause. If a clause contains only one literal which is assigned to 1, this is a critical clause for the variable x_i mentioned in that literal; delete this clause (which is satisfied) entirely, and add x_i to a set S.

After these deletions, we are left with a CNF formula $\tilde{\varphi}$. Because all variables appearing in $\tilde{\varphi}$ are free and we have deleted clauses from φ satisfied by I and the literals which are assigned 0 by I, it follows that I is an implicant if and only if $\tilde{\varphi}$ is a tautology. Now, $\tilde{\varphi}$ is a tautotology if and only if each of its disjunctive clauses is a tautology, and one can easily check whether any such clause C is a tautology: simply ensure that for each literal ℓ mentioned in the disjunction C, some literal equivalent to its negation of ℓ is also mentioned in C. Thus one can check whether I is an implicant. To check whether I is a prime implicant, simply confirm that S contains all variables fixed by I; if not, reject I, which fixes variables that do not have critical clauses.

In total this takes time $O(|\varphi| + n \cdot \log n)$, where the linear term corresponds to computation of $\varphi(I)$ and iteration over all clauses, and the remaining term corresponds to operations on S. We will only be needing the looser bound of $O(|\varphi| \cdot n)$. ◀

Proof of Fact 20. Let $\mathbf{F}$ be a finite field of N elements. Sample uniform $c_0, \ldots c_{k-1} \in \mathbf{F}$. Let $X_\alpha = \sum_i c_i \alpha^i$. There are N^k choices of coefficients, and k-wise independence follows by Lagrange interpolation. ◀

Proof of Fact 29. Let I be an implicant of φ. It suffices to show that if $\neg x_i \in I$ for some i, the assignment $I' = I \setminus \{\neg x_i\}$ is a smaller implicant of φ. Indeed, let J be some total assignment agreeing with I' on its fixed variables. If $J_i = 0$, then J agrees with I well, and so satisfies φ. Otherwise, $J_i = 1$, then $J \oplus i$ satisfies I, and hence φ, and $J \oplus i \leq J$, so by monotonicity, J again satisfies φ, as required. ◀

Proof of Fact 33. Let φ be any monotone formula. We show the first statement, as the second follows immediately. Let I be some prime implicant of φ, viewed as a set of un-negated literals. Let σ be the total assignment that assigns everything in I to 1 and every other variable 0. We claim that no other prime implicant covers σ. Indeed, let J be a prime implicant that covers σ; we will show that $J \subseteq I$. If not, there is some variable i such that $x_i \in J$, and $x_i \notin I$, but then J does not cover σ because σ is the ith variable assigned 0. Thus we have that $J \subseteq I$, as I was prime, it must be the case that $J = I$, which shows that the only prime implicant that covers σ is I. ◀