

# A Method for Generating All the Prime Implicants of Binary CNF Formulas

**Yakoub Salhi**

CRIL-CNRS, Université d'Artois,  
F-62307 Lens Cedex, France  
salhi@cril.fr

## Abstract

In this paper, we propose a method for generating all the prime implicants of a binary CNF formula. The main idea consists in using an algorithm for generating the maximal independent sets to compute all the prime implicants that cover a model. Such an approach mainly allows to reduce the number of calls to a SAT oracle. Before describing this algorithm, we propose a preprocessing method for reducing the search space, which is based on the use of the resolution closure. This method consists in exploiting the unit clauses and the equivalences generated in the resolution closure. Finally, we propose a simple approach for dividing the search space, which is based on the use of an algorithm that generates all the prime implicants that contain a given literal.

## Introduction

Developing efficient algorithms for solving the problem of generating all the prime implicants of propositional formulas is an important challenge in computer science. One of the main applications of this problem is its use for minimizing Boolean functions, which consists in reducing the size of a Boolean function to a smaller, simpler and equivalent function (Quine 1952; McCluskey 1956; Quine 1959). Boolean function minimization is used in several computer science tasks that are related to classical propositional logic, such as reducing digital circuit size, which is an important way to improve computation speed (Tison 1967). Generating prime implicants has many other applications in computer science in general, and artificial intelligence in particular, such as knowledge compilation (Darwiche and Marquis 2002), model based diagnosis (de Kleer, Mackworth, and Reiter 1990) and databases (del Val 1994). Different Other applications are provided in (Ginsberg 1989; Acuña et al. 2012; Slavkovik and Ågotnes 2014).

In the literature, several algorithms for generating prime implicants of propositional formulas use branch and bound/backtrack techniques, such as DPLL-like procedures (Castell 1996; Schrag 1996; Ravi and Somenzi 2004). There are also integer linear programming encodings for computing one of the minimum-size prime implicants (e.g. (Pizzuti 1996; Manquinho et al. 1997)). More recently, a SAT-based approach for generating all the prime impli-

cants has been proposed in (Jabbour et al. 2014). This approach consists in providing a SAT encoding of the input formula, so that the models of this encodings represent the prime implicants of the original formula.

We focus in this work on the problem of generating all the prime implicants of a propositional formula in conjunctive normal form (CNF) that does not contain more than two literals per clause (binary CNF formulas). To the best of our knowledge, the particularities of this tractable class in SAT have not been exploited in the context of generating all the prime implicants. However, it is worth noticing that this class has been studied for the problem of counting the number of prime implicants (Boufkhad and Dubois 1999; Talebanfard 2016). One of the main results in (Talebanfard 2016) consists in showing that the maximum number of prime implicants of the binary CNF formulas having  $n$  variables is greater than  $3^{\frac{n}{3}}$ . In a sense, this shows that the problem of generating all the prime implicants of a binary CNF formula is a complex task.

In this paper, we propose a new approach for generating all the prime implicants of a binary CNF formula. We first propose a preprocessing method to reduce the search space by using the resolution closure. This preprocessing method benefits from the fact that computing the resolution closure is a tractable task in the case of the binary CNF formulas. It reduces the search space by exploiting the unit clauses and the equivalences found in the resolution closure. In addition, we provide an algorithm for generating all the prime implicants of a binary CNF formula. The base idea consists in reducing the number of calls to a SAT oracle by using an algorithm for generating the maximal independent sets (MIS). More precisely, the MIS generator is used in our algorithm to get all the prime implicants that cover a model. We consider in this work that generating maximal independent sets is simpler than generating prime implicants, since we do not check the satisfiability. Intuitively, the reason is in the fact that each Boolean interpretation that satisfies a binary CNF formula can be seen as a complement of a particular independent set (without complementary literals) of the graph where the literals represent the vertices and the clauses represent the edges. Furthermore, to avoid all the prime implicants that cover a model in the future search steps of our algorithm (recursive calls), we only add a single clause.

Our last contribution is an approach for dividing the

search space. This approach uses an algorithm for generating all the prime implicants that contain a given literal, which is defined by using our algorithm for generating the prime implicants and by taking into account each case where the considered literal is possibly in a prime implicant. Our approach for dividing the search space is an interesting research path for defining parallel generators of prime implicants.

## Preliminaries

### SAT Problem and Prime Implicant Notion

A CNF formula is a conjunction of clauses where a *clause* is a disjunction of literals. A *literal* is a positive or negated propositional variable. A CNF formula can also be seen as a set of clauses and a clause as a set of literals. Thus, a CNF formula of the form  $(l_1^1 \vee \dots \vee l_{k_1}^1) \wedge \dots \wedge (l_1^n \vee \dots \vee l_{k_n}^n)$  can be also represented in this work by the set of clauses  $\{(l_1^1 \vee \dots \vee l_{k_1}^1), \dots, (l_1^n \vee \dots \vee l_{k_n}^n)\}$  and the set of sets  $\{\{l_1^1, \dots, l_{k_1}^1\}, \dots, \{l_1^n, \dots, l_{k_n}^n\}\}$ .

Given a CNF formula  $\phi$ , we denote by  $Var(\phi)$  (resp.  $Lit(\phi)$ ) the set of propositional variables (resp. literals) occurring in the formula  $\phi$ . A CNF formula is *binary* if each clause contains at most two literals. Given a literal  $l$ , we use  $\bar{l}$  to denote the complementary literal of  $l$ .

A *Boolean interpretation* of a propositional formula  $\phi$  is an assignment that associates truth values in  $\{0, 1\}$  to propositional variables in  $Var(\phi)$ , where 0 stands for false and 1 stands for true. It is extended to propositional formulas as usual. A Boolean interpretation of  $\phi$  is *complete* if it gives a truth value to each variable in  $Var(\phi)$ , otherwise it is said to be *partial*. A *model* of a propositional formula is a complete Boolean interpretation satisfying this formula, i.e., a complete Boolean interpretation making this formula true. Although we define a model as a complete interpretation, it is worth noticing that a partial interpretation can make a formula true. The problem of determining if there exists a model that satisfies a given propositional formula, abbreviated as SAT, is one of the most studied NP-complete problems.

For convenience purposes, we represent from now on the Boolean interpretations as sets of literals. More precisely, the set of literals  $\{p_1, \dots, p_m, \neg q_1, \dots, \neg q_n\}$  represent the Boolean interpretation that associates 1 to the variables  $p_1, \dots, p_m$  and 0 to  $q_1, \dots, q_n$ . Note that to be a Boolean interpretation a set of literals does not have to contain a literal and its negation (complementary literals).

A (partial) Boolean interpretation that satisfies a propositional formula is called an *implicant*. Moreover, an implicant  $m$  of a formula  $\phi$  is called a *prime implicant* (in short PI) of this formula if for all literal  $l \in m$ , the Boolean interpretation  $m \setminus \{l\}$  does not satisfy  $\phi$ .

Let us now describe notations that are used in the sequel. Given a Boolean interpretation  $m$ , we use  $\bar{m}$  to denote the clause  $\bigvee_{l \in m} \bar{l}$ . Moreover, given a CNF formula  $\phi$  and a partial Boolean interpretation  $m$ , we use  $\phi|_m$  to denote the CNF formula obtained from  $\phi$  by removing all the clauses containing literals in  $m$  and removing all the complement literals of those in  $m$ . More precisely,  $\phi|_m = \{c \in \phi \mid \forall l \in m, l \notin c \text{ and } \bar{l} \notin c\} \cup \{c \setminus \bar{l} \mid c \in \phi, l \in m, l \notin c \text{ and } \bar{l} \in c\}$ .

In other words,  $\phi|_m$  is the result of assigning truth according to the interpretation  $m$ . Furthermore, given a CNF formula  $\phi$  and a literal  $l$  in  $\phi$ , we use  $\phi^{\uparrow l}$  to denote the formula obtained from  $\phi$  by removing the literal  $l$ , i.e.,  $\phi^{\uparrow l} = \{c \setminus \{l\} \mid c \in \phi\}$ .

A tautological clause is a clause containing two complementary literals (e.g.  $p \vee \neg p$ ). The tautological clauses can be removed from a CNF formula without changing its truth value. From now on, we only consider the binary CNF formulas that do not contain any tautological clause.

### Maximal Independent Sets

We here describe the problem of generating the maximal independent sets of an undirected graph. The algorithms solving this problem are used in our method for generating the prime implicants.

Given an undirected graph  $G = (V, E)$  where  $V$  is the set of its vertices and  $E$  is the set of its edges, an *independent set* of  $G$  is a subset  $S$  of  $V$  such that no two of which are adjacent, i.e., for all  $v, v' \in S$  with  $v \neq v'$ ,  $\{v, v'\} \notin E$ . Moreover,  $S$  is a *maximal independent set* if it is not a proper subset of any other independent set, i.e.,  $S \not\subset S'$  for every independent set  $S'$  of  $G$ . The problem of maximal independent set enumeration consists in finding all the maximal independent sets of a given graph.

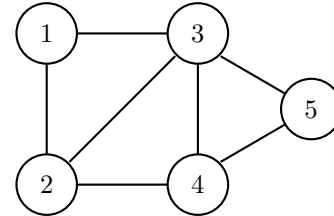


Figure 1: An undirected graph

Consider, for instance, the graph described in Figure 1. The set of vertices  $\{1, 4\}$  is a maximal independent set since the vertices 2 and 3 are adjacent to 1 and 5 is adjacent to 4, and 4 and 1 are not adjacent. The sets  $\{1, 5\}$ ,  $\{2, 5\}$  and  $\{3\}$  are the other maximal independent sets.

There are several interesting methods in the literature for generating the maximal independent sets of a graph (e.g. (Tsukiyama et al. 1977; Johnson, Papadimitriou, and Yannakakis 1988; Boros et al. 2000)). In this work, we use an oracle for generating the maximal independent sets in our algorithm for generating the prime implicants of a binary CNF formula.

### A Preprocessing to Reduce the Search Space

In this section, we propose a preprocessing method for reducing the search space in computing the prime implicants of a binary CNF formula. This method is based on computing the resolution closure for obtaining unit clauses, also called backbone, and equivalences between literals. Let us recall that a unit clause is a clause that contain a single literal. From the computational complexity point of view, our preprocessing method can be performed in polynomial time,

since computing the resolution closure of any binary CNF formula is a polynomial task.

The resolution rule is defined as follows:

$$\frac{l \vee c_1 \quad \bar{l} \vee c_2}{c_1 \vee c_2} [Res]$$

The clause  $c_1 \vee c_2$  is called a *resolvent* of the clauses  $l \vee c_1$  and  $\bar{l} \vee c_2$ . We use  $c_1, c_2 \vdash_{Res} c$  to denote the fact that  $c$  is a resolvent of the clauses  $c_1$  and  $c_2$ .

**Definition 1 (Resolution Closure).** Let  $\phi$  be a CNF formula and  $n \in \mathbb{N}$ . Then, we define:

$$\begin{aligned} \mathcal{C}^0(\phi) &= \phi \\ \mathcal{C}^1(\phi) &= \phi \cup \{c \mid \exists c_1, c_2 \in \phi : c_1, c_2 \vdash_{Res} c\} \\ \mathcal{C}^{n+1}(\phi) &= \mathcal{C}^1(\mathcal{C}^n(\phi)) \end{aligned}$$

We use  $\mathcal{C}(\alpha)$  to denote the resolution closure  $\bigcup_n \mathcal{C}^n(\alpha)$ .

For instance, consider the binary CNF formula  $\phi = \{(p \vee q), (p \vee \neg q), (\neg p \vee r)\}$ . We have  $\mathcal{C}^1(\alpha) = \{(p \vee q), (p \vee \neg q), (\neg p \vee r), p, q \vee r, \neg q \vee r\}$ . Then,  $\mathcal{C}(\alpha) = \mathcal{C}^2(\alpha) = \mathcal{C}^1(\alpha) \cup \{r, p \vee r\}$  holds.

It is easy to see that one can compute the resolution closure for any binary CNF formula in polynomial time. Indeed, the number of all possible binary clauses is quadratic in the number of the literals of the input binary CNF formula.

**Proposition 1.** Let  $\phi$  be a satisfiable binary CNF formula. Then, we have  $PIes(\phi) = \{m' \cup m \mid m' \in PIes(\phi|_m)\}$  where  $m = Lit(\phi) \cap \mathcal{C}(\phi)$ .

*Proof.* This property comes from the fact that the literals in  $m = Lit(\phi) \cap \mathcal{C}(\phi)$  are logical consequences of the formula  $\phi$ . It means that  $m \subseteq m'$  for every implicant  $m'$  of the formula  $\phi$ .  $\square$

The previous proposition shows that computing the resolution closure allows us to reduce the search space for computing the prime implicants. Indeed, it shows that the primes implicants of any satisfiable binary CNF formula  $\phi$  can be directly obtained from the prime implicants of the formula  $\phi|_{Lit(\phi) \cap \mathcal{C}(\phi)}$ .

**Definition 2.** Let  $\phi$  be a binary CNF formula. Then, we define the equivalence relation  $\leftrightarrow_\phi$  over the literals of  $\phi$  as follows:  $l \leftrightarrow_\phi l'$  iff  $\{\bar{l} \vee l', l \vee \bar{l}'\} \subseteq \mathcal{C}(\phi)$ . Clearly,  $\leftrightarrow_\phi$  is an equivalence relation since the equivalence connective  $\leftrightarrow$  is reflexive, symmetric and transitive ( $\psi_1 \leftrightarrow \psi_2 \equiv (\psi_1 \rightarrow \psi_2) \wedge (\psi_2 \rightarrow \psi_1)$ ).

In other words, we have  $l \leftrightarrow_\phi l'$  if and only if  $l \leftrightarrow l'$  is a logical consequence of  $\phi$ .

Given a binary CNF formula  $\phi$ , we use  $\mathcal{EC}_\phi$  to denote the set of the equivalence classes of  $\leftrightarrow_\phi$ . In addition, we use  $[l]_\phi^{\leftrightarrow_\phi}$  to denote the equivalence class of the literal  $l$ , i.e.,  $[l]_\phi^{\leftrightarrow_\phi} = \{l' \in Lit(\phi) \mid l \leftrightarrow_\phi l'\}$ . Moreover, given a set of literals  $\{l_1, \dots, l_k\}$  and a literal  $l$ , we use  $\phi[l/l_1, \dots, l_k]$  to denote the formula obtained from  $\phi$  by replacing each occurrence of  $l_i$  (resp.  $\bar{l}_i$ ) with  $l$  (resp.  $\bar{l}$ ) for every  $i \in 1..k$ .

**Proposition 2.** Let  $\phi$  be a binary CNF formula,  $l \in Lit(\phi)$  and  $[l]_\phi^{\leftrightarrow_\phi} = \{l_1, \dots, l_k\}$ . Then,  $PIes(\phi) = \{m \cup \{\bar{l}_1, \dots, \bar{l}_k\} \mid m \in PIes(\psi) : l \notin m\} \cup \{m \cup [l]_\phi^{\leftrightarrow_\phi} \mid m \in PIes(\psi) : \bar{l} \notin m\}$  where  $\psi = \phi[l/l_1, \dots, l_k]$ .

*Proof.*

*Part  $\subseteq$ .* Let  $m$  be a prime implicant of  $\phi$ . Knowing that all the literals in  $[l]_\phi^{\leftrightarrow_\phi}$  are equivalent, we get  $[l]_\phi^{\leftrightarrow_\phi} \subseteq m$  or  $\{\bar{l}_1, \dots, \bar{l}_k\} \subseteq m$ , since otherwise  $m$  does not satisfy  $\phi$ . We here consider the case  $[l]_\phi^{\leftrightarrow_\phi} \subseteq m$ , the other case being similar. Using the fact that  $[l]_\phi^{\leftrightarrow_\phi} \subseteq m$ , we obtain that one of the two interpretations  $m \setminus [l]_\phi^{\leftrightarrow_\phi} \cup \{l\}$  and  $m \setminus [l]_\phi^{\leftrightarrow_\phi}$  is a prime implicant of  $\phi[l/l_1, \dots, l_k]$ . Thus,  $m \in \{m \cup [l]_\phi^{\leftrightarrow_\phi} \mid m \in PIes(\phi[l/l_1, \dots, l_k]) : \bar{l} \notin m\}$  holds.

*Part  $\supseteq$ .* We only consider the case of the elements of the set  $S = \{m \cup \{\bar{l}_1, \dots, \bar{l}_k\} \mid m \in PIes(\psi) : l \notin m\}$ , the other case being similar. Let  $m$  be a Boolean interpretation in  $S$ . Then,  $m \setminus \{\bar{l}_1, \dots, \bar{l}_k\}$  is a prime implicant of  $\psi$ . Knowing that the elements of  $[l]_\phi^{\leftrightarrow_\phi}$  are equivalent, we obtain that  $m \cup \{\bar{l}_1, \dots, \bar{l}_k\}$  is a prime implicant of  $\phi$  if it satisfies this formula. Using the fact that  $l \notin m \setminus \{\bar{l}_1, \dots, \bar{l}_k\}$ , we deduce that  $m$  is a prime implicant of  $\phi$ .  $\square$

Therefore, given binary CNF formula  $\phi$ , Proposition 1 and Proposition 2 shows that the set of prime implicants  $PIes(\phi)$  can be easily computed from  $PIes(\phi|_m[l_1/l_1^1, \dots, l_{k_1}^1] \dots [l_n/l_1^n, \dots, l_{k_n}^n])$  where  $m = Lit(\phi) \cap \mathcal{C}(\phi)$ ,  $\mathcal{EC}_\phi = \{[l_1]_\phi, \dots, [l_n]_\phi\}$ , and  $[l_i]_\phi = \{l_1^i, \dots, l_{k_i}^i\}$  for every  $i \in 1..n$ .

## An Algorithm for Generating Pies

In this section, we propose a recursive algorithm that allows to compute all the prime implicants of a binary CNF formula. The base idea consists in using an algorithm for generating the maximal independent sets (MISes) for computing all the prime implicants that cover a model which is found using a SAT oracle.

**Algorithm 1** A recursive algorithm for generating all the prime implicants of a binary CNF formula

---

```

1: procedure PRIMEIMPS( $\phi, \mathcal{M}$ )  $\triangleright \mathcal{M}$  is a set of Boolean
   interpretations
2:   ( $s, m$ )  $\leftarrow$  SAT( $\phi \wedge \bigwedge_{m \in \mathcal{M}} \bar{m}$ )
3:   if  $s$  then
4:      $u \leftarrow$  UNILIT( $\phi, m$ )
5:      $\psi \leftarrow$  REST( $\phi, u$ )
6:      $L \leftarrow$  MISSES(GRAPH( $\psi$ ))
7:     return  $\{u \cup (Lit(\psi) \setminus s) \mid s \in L\} \cup$ 
      PRIMIMPS( $\phi, \mathcal{M} \cup \{u\}$ )
8:   else
9:     return  $\emptyset$ 

```

---

Our recursive algorithm PRIMIMPS for computing the prime implicants of a binary CNF formula is described in

Algorithm 1. In the following we describe the different instructions of this algorithm. In Line 2,  $\text{SAT}(\cdot)$  represents a query to a SAT oracle where  $s$  is the decision result (true or false) and  $m$  is a model of the input formula if it is satisfiable. Furthermore, given a CNF formula  $\phi$  and a model  $m$  of  $\phi$ , we use  $UL_\phi^m$  to denote the set of literals  $\{l \in m \mid \exists c \in \phi, m \cap c = \{l\}\}$ . It is worth noticing that a literal  $l \in m$  is in  $UL_\phi^m$  if and only if  $m \setminus \{l\}$  does not satisfy  $\phi$ . Thus, it is clear that every prime implicant that covers the model  $m$  contains all the literals in  $UL_\phi^m$ . In the context of our algorithm, the procedure  $\text{UNILIT}(\phi, m)$  in Algorithm 1 (Line 4) is used to compute the set of literals  $UL_\phi^m$ . The procedure  $\text{REST}(\phi, u)$  in Line 5 returns the set of clauses that are not satisfied by  $u$ . We focus in our algorithm on the clauses in  $\text{REST}(\phi, u)$  because we exploit the fact that the literals in  $UL_\phi^m$  are in the current prime implicants. The procedure  $\text{GRAPH}(\psi)$  in Line 6 transforms simply a set of binary clauses  $\psi$  to the undirected graph  $G = (\text{Lit}(\psi), \psi)$ . It is important to note that  $\psi$  does not contain any complementary literals, since the definition of  $UL_\phi^m$ . The procedure  $\text{MISES}(G)$  in the same line returns all the maximal independent sets of  $G$ .

In Line 7, as shown in Theorem 6, we return the set  $\{u \cup (\text{Lit}(\psi) \setminus s) \mid s \in L\}$  since it contains all the prime implicants that include the set of literals  $u$  ( $UL_\phi^m$ ). The recursive call  $\text{PRIMIMPS}(\phi, \mathcal{M} \cup \{u\})$  allows us to find all the prime implicants that do not include  $u$  and any element of  $\mathcal{M}$ . In Line 9,  $\text{PRIMIMPS}$  returns the empty set since the unsatisfiability of  $\phi \wedge \bigwedge_{m \in \mathcal{M}} \bar{m}$  implies that it remains no model, and consequently no prime implicant, which satisfies  $\phi$  and does not include any set of literals in  $\mathcal{M}$ .

In order to show the soundness of our algorithm, we need to show that  $\text{PRIMIMPS}(\phi, \emptyset)$  returns all the prime implicants of  $\phi$ , for every binary CNF formula  $\phi$ . Using the fact that we add  $UL_\phi^m$  to  $\mathcal{M}$  in the recursive call (Line 7) and the fact that the satisfaction of the negations of the models in  $\mathcal{M}$  (Line 2) is required, we obtain that  $\text{PRIMIMPS}(\phi, \mathcal{M})$  terminates for every binary CNF formula  $\phi$  and subset of Boolean interpretations  $\mathcal{M}$ .

**Proposition 3.** *Let  $\phi$  be a binary CNF formula and  $m$  a model of  $\phi$ . Then,  $UL_\phi^m \cup m'$  is a prime implicant of  $\phi$  for every  $m' \subseteq m$  prime implicant of  $\bigwedge_{\substack{c \in \phi \\ UL_\phi^m \neq c}} c$ .*

*Proof.* In this proof we use  $\psi_1$  and  $\psi_2$  to denote the formulas  $\psi_1 \equiv \bigwedge_{\substack{c \in \phi \\ UL_\phi^m \neq c}} c$  and  $\psi_2 \equiv \bigwedge_{\substack{c \in \phi \\ UL_\phi^m \neq c}} c$  respectively.

Assume that there exists  $m' \subseteq m$  prime implicant of  $\psi_2$  s.t.  $UL_\phi^m \cup m'$  is not a prime implicant of  $\phi$ . Then, there exists  $l \in UL_\phi^m \cup m'$  such that  $(UL_\phi^m \cup m') \setminus \{l\}$  satisfies  $\phi$ . Using the definition of  $UL_\phi^m$ , we know that  $l \notin UL_\phi^m$  since otherwise  $UL_\phi^m \cup m' \setminus \{l\}$  does not satisfy  $\psi_1$ , and consequently it does not satisfy  $\phi$ . Thus,  $l \in m'$  holds. Then, there exists a clause  $c = l \vee l' \in \psi_2$  s.t.  $l' \notin m'$  since  $m'$

is a prime implicant of  $\psi_2$ . Knowing that  $UL_\phi^m \cap m' = \emptyset$  since  $\psi_2$  does not contain any literal in  $UL_\phi^m$ , we obtain that  $(UL_\phi^m \cup m') \setminus \{l\}$  does not satisfy  $c$ , and consequently, we get a contradiction. Therefore,  $UL_\phi^m \cup m'$  is a prime implicant of the formula  $\phi$ .  $\square$

Let us recall that a literal in a CNF formula is a *pure literal* if its complement does not occur in this formula. Clearly, given a CNF formula  $\phi$  and a pure literal  $l$ ,  $\phi$  is satisfiable if and only if  $\phi_l$  is satisfiable.

**Proposition 4.** *Let  $\phi$  be a binary CNF s.t.  $\forall c \in \phi, |c| = 2$  and  $\text{Lit}(\phi)$  is a model of  $\phi$ , i.e.  $\phi$  is a binary CNF formula that contains only pure literals. Then,  $\text{PIEs}(\phi) = \{\text{Lit}(\phi) \setminus s \mid s \in \text{MISES}(G)\}$  holds where  $G = (\text{Lit}(\phi), \phi)$ .*

*Proof.*

*Part  $\subseteq$ .* Let  $m$  be a prime implicant of  $\phi$ . Then, we know that, for all  $l \vee l' \in \phi, \{l, l'\} \not\subseteq \text{Lit}(\phi) \setminus m$  holds since  $m$  satisfies  $\phi$ . Thus,  $\text{Lit}(\phi) \setminus m$  is an independent set of  $G$ . Moreover, we have for all  $l \in m$ , there exists a clause  $c \equiv l \vee l'$  s.t.  $l' \notin m$ . Thus, for all  $l \in m, (\text{Lit}(\phi) \setminus m) \cup \{l\}$  is not an independent set. As a consequence,  $\text{Lit}(\phi) \setminus m$  is a maximal independent set of  $G$ .

*Part  $\supseteq$ .* Let  $s$  be a maximal independent set of  $G$ . Then for all  $l \vee l' \in \phi, \{l, l'\} \not\subseteq s$ . Thus,  $\text{Lit}(\phi) \setminus s$  satisfies  $\phi$ . Assume that there exists  $l \in \text{Lit}(\phi)$  s.t.  $l \notin s$  and  $(\text{Lit}(\phi) \setminus s) \setminus \{l\}$  satisfies  $\phi$ . Then, there exists a clause  $c = l \vee l'$  s.t.  $l' \notin s$ , and consequently,  $\{l, l'\} \subseteq s$  holds. Thus, we get a contradiction. Therefore,  $\text{Lit}(\phi) \setminus s$  is a prime implicant of the formula  $\phi$ .  $\square$

The two previous propositions are used to show that the output of  $\text{PRIMIMPS}(\phi, \emptyset)$  is a set of prime implicants of  $\phi$ . Indeed, the first proposition show that, given a binary CNF formula  $\phi$  and a model  $m$  of  $\phi$ , all the interpretations of the form  $UL_\phi^m \cup m'$ , where  $m'$  are prime implicants of the part of  $\phi$  which is not satisfied by  $UL_\phi^m$  ( $\psi$ ), are prime implicants of  $\phi$ . Then, the second proposition shows that the prime implicants of the part of  $\phi$  which is not satisfied by  $UL_\phi^m$  are the complements of the maximal independent sets of the graph corresponding to this part. As a consequence, the set  $\{u \cup (\text{Lit}(\psi) \setminus s) \mid s \in L\}$ , which is returned in Line 7, contains only prime implicants of  $\phi$ .

**Proposition 5.** *Let  $\phi$  be a binary CNF formula and  $m$  a model of  $\phi$ . Then,  $\bigcap_{\substack{m' \in \text{PIes}(\phi) \\ m' \subseteq m}} m' = UL_\phi^m$  holds.*

*Proof.* We use  $S$  to denote the set  $\bigcap_{\substack{m' \in \text{PIes}(\phi) \\ m' \subseteq m}} m'$ .

*Part  $\subseteq$ .* Assume that there exists  $l \in S$  s.t.  $l \notin UL_\phi^m$ . Then, for all clause  $c = l \vee l'$  containing  $l$  in  $\phi, \{l, l'\} \subseteq m$  holds. Thus,  $m \setminus \{l\}$  satisfies  $\phi$ , which means that there exists a prime implicant in  $S$  that does not contain  $l$ . Therefore, we get a contradiction, and consequently  $S \subseteq UL_\phi^m$  holds.

*Part  $\supseteq$ .* It is a direct consequence of the fact that  $m \setminus \{l\}$  does not satisfy  $\phi$  for every  $l \in UL_\phi^m$ .  $\square$

The following theorem shows that the set of Boolean interpretations  $\{u \cup (Lit(\psi) \setminus s) \mid s \in L\}$  in Line 7 of Algorithm 1 corresponds to the set of all the prime implicants that cover the model found in Line 2, i.e., using Proposition 5, all the prime implicants including  $UL_\phi^m$  where  $m$  is the model of  $\phi$  found in Line 2.

**Theorem 6.** *Let  $\phi$  be a binary CNF formula and  $m$  a model of  $\phi$ . Then, for all  $m'$  a partial Boolean interpretation of  $\phi$  s.t.  $m' \subseteq m$ ,  $m'$  is a prime implicant of  $\phi$  iff  $m' = UL_\phi^m \uplus m''$  where  $\uplus$  stands for disjoint union,  $m'' \in \{Lit(\psi) \setminus s \mid s \in MISes(G)\}$  with  $\psi \equiv \bigwedge_{\substack{c \in \phi \\ UL_\phi^m \neq c}} c$  and*

$$G = (Lit(\psi), \psi).$$

*Proof.*

*Part  $\Rightarrow$ .* Using Proposition 5,  $UL_\phi^m \subseteq m'$  for every prime implicant  $m'$  that covers  $m$ . Given a prime implicant  $m'$  of  $\phi$  covering  $m$ , we have to show the property  $m' \setminus UL_\phi^m \in \{Lit(\psi) \setminus s \mid s \in MISes(G)\}$ , which is equivalent to show that  $s = Lit(\psi) \setminus (m' \setminus UL_\phi^m)$  is a maximal independent set. Assume that there exists  $l \notin s$  ( $l \in m' \setminus UL_\phi^m$ ) s.t.  $s \cup \{l\}$  is an independent set. It means that  $c \cap s = \emptyset$  for every clause  $c \in \phi$  with  $l \in c$ . Then,  $(m' \setminus UL_\phi^m) \setminus \{l\}$  satisfies  $\psi$ . Using Proposition 3, we know that there exists  $m^3 \subseteq m' \setminus \{l\}$  s.t.  $UL_\phi^m \uplus m^3 \subset m'$  is a prime implicant of  $\phi$ . Thus, we get a contradiction since  $m'$  is a prime implicant of  $\phi$ .

*Part  $\Leftarrow$ .* Using the definition of  $UL_\phi^m$ , we know that  $m \setminus UL_\phi^m = Lit(\psi)$ . Using Proposition 4,  $PIes(\psi) = \{Lit(\psi) \setminus s \mid s \in MISes(G)\}$  holds. Using Proposition 3,  $m = UL_\phi^m \uplus m''$  is a prime implicant of  $\phi$  for every  $m'' \in \{Lit(\psi) \setminus s \mid s \in MISes(G)\}$ .  $\square$

Using the fact that the call of  $PRIMIMPS(\phi, \emptyset)$  performs an exhaustive search in the sense that it considers all the models of  $\phi$  (see the recursive call in Line 7), we obtain that Theorem 6 implies that  $PRIMIMPS(\phi, \emptyset)$  returns all the prime implicants of  $\phi$ .

Compared to the SAT-based approach proposed in (Jabbour et al. 2014), our algorithm uses a smaller number of calls to a SAT oracle. Indeed, Jabbour et al's approach uses a call to a SAT oracle for each prime implicant. There are cases where this approach uses an exponential number of calls to a SAT oracle, while our algorithm uses a single call. Consider, for instance, the simple binary CNF formula  $\phi = (p_1 \vee p'_1) \wedge \dots \wedge (p_n \vee p'_n)$  such that  $p_1, p'_1, \dots, p_n, p'_n$  are all distinct propositional variables, and the model  $m = \{p_1, p'_1, \dots, p_n, p'_n\}$  of  $\phi$ . Clearly, there are  $2^n$  prime implicants that cover the model  $m$ . Thus, Jabbour et al's approach uses  $2^n$  to find these prime implicants. However, after finding the model  $m$ , our algorithm does not use any other call to a SAT oracle. In fact, we use in this case a generator  $MISes$  to avoid calling a SAT oracle.

## On Dividing the Search Space

In this section, we propose an algorithm that generates all the prime implicants that contain a given literal. Such an al-

gorithm can be used for defining parallel algorithms by dividing the search space using guiding paths.

Given a binary CNF formula  $\phi$  and a literal  $l$  occurring in  $\phi$ ,  $FPIes(\phi, l)$  denote the set of all the prime implicants of  $\phi$  that contain  $l$ .

---

### Algorithm 2 A variant of PrimeImps

---

```

1: procedure VPRIMEIMPS( $\phi, \mathcal{M}$ )
2:    $(s, m) \leftarrow \text{SAT}(\phi \wedge \bigwedge_{m \in \mathcal{M}} \bar{m})$ 
3:   if  $s$  then
4:      $u \leftarrow \text{UNILIT}(\phi, m)$ 
5:      $\psi \leftarrow \text{REST}(\phi, u)$ 
6:      $L \leftarrow \text{MISes}(\text{GRAPH}(\psi))$ 
7:      $(\mathcal{R}', \mathcal{M}') \leftarrow \text{VPRIMEIMPS}(\phi, \mathcal{M} \cup \{u\})$ 
8:     return  $\{u \cup (Lit(\psi) \setminus s) \mid s \in L\} \cup \mathcal{R}', \mathcal{M}'$ 
9:   else
10:    return  $(\emptyset, \mathcal{M})$ 

```

---

**Proposition 7.** *Let  $\phi$  be a binary CNF formula and  $l$  a literal in  $\phi$ . Then,  $PIes(\phi) = FPIes(\phi, l) \uplus PIes(\phi^{\uparrow l})$  holds.*

*Proof.*

*Part  $\subseteq$ .* Let  $m$  be a prime implicant of  $\phi$ . If  $m$  contains  $l$ , then  $m$  belongs to  $FPIes(\phi, l)$ . Otherwise,  $m$  includes  $\mathcal{A}(\phi, l)$  since  $m$  satisfies the clauses containing  $l$ . Thus,  $m$  is a prime implicant of  $\phi^{\uparrow l}$ .

*Part  $\supseteq$ .* We have only to show that  $PIes(\phi^{\uparrow l}) \subseteq PIes(\phi)$  since  $FPIes(\phi, l) \subseteq PIes(\phi)$ . Assume that there exists  $m \in PIes(\phi^{\uparrow l})$  s.t.  $m \notin PIes(\phi)$ . Then, there exists  $l_0 \in m$  such that  $m \setminus \{l_0\}$  satisfies  $\phi$ . Clearly,  $m$  does not contain  $l$  and includes  $\mathcal{A}(\phi, l)$  since  $l$  does not occur in the elements of  $PIes(\phi^{\uparrow l})$ . Using the fact that  $l \notin m$ ,  $l_0$  does not belong to  $\mathcal{A}(\phi, l)$ . Thus, the clauses containing  $l_0$  in  $\phi^{\uparrow l}$  are those containing  $l_0$  in  $\phi$ . Using the fact that  $m$  is a prime implicant of  $\phi^{\uparrow l}$ , we know that  $m \setminus \{l_0\}$  does not satisfy  $\phi$  and we get a contradiction.

Finally, it is clear that  $FPIes(\phi, l) \cap PIes(\phi^{\uparrow l}) = \emptyset$  since all the elements of  $PIes(\phi^{\uparrow l})$  does not contain  $l$ .  $\square$

Proposition 7 describes a simple approach for dividing recursively the search space. This proposition shows the interest of having a procedure that allows to enumerate all the prime implicants that contain a given literal. In particular, note that if  $l$  is a unit clause in  $\phi$ , then  $PIes(\phi) = FPIes(\phi, l)$  holds since  $PIes(\phi^{\uparrow l}) = \emptyset$ .

Our algorithm FPI for computing the prime implicants of a binary CNF formula that contain a given literal is described in Algorithm 3. In Line 2, the formula  $\psi$  is obtained from  $\phi$  by removing all the clauses containing  $l$  and removing the occurrences of  $\bar{l}$ . This comes from the fact that the aim is to generate the prime implicants containing the literal  $l$ . If the literal  $l$  is a unit clause in  $\phi$  (Line 3), then  $l$  is clearly in all the prime implicants of  $\phi$  (Line 4). Otherwise, the for-loop allows us to consider all the possible cases where the literal  $l$  is forced to be in a prime implicant. In this context,  $\mathcal{A}(\phi, l)$  corresponds to the set of the literals that occurs in the clauses containing  $l$ , i.e.,  $\mathcal{A}(\phi, l) = \{l' \in Lit(\phi) \mid \exists l \vee l' \in \phi\}$ . Moreover, we use in Line 9 a simple variant of our algorithm  $PRIMIMPS$ , called  $VPRIMIMPS$ , which is described

---

**Algorithm 3** An algorithm for generating all the prime implicants containing a given literal

---

```

1: procedure FPI( $\phi, l$ )
2:    $\psi \leftarrow \phi|_l$ 
3:   if  $l \in \phi$  then
4:     return  $\{m \cup \{l\} \mid m \in \text{PRIMIMPS}(\psi, \emptyset)\}$ 
5:   else
6:      $\mathcal{M} \leftarrow \emptyset$ 
7:      $\mathcal{R} \leftarrow \emptyset$ 
8:     for  $l' \in \mathcal{A}(\phi, l)$  do
9:        $(\mathcal{R}', \mathcal{M}') \leftarrow \text{VPRIMIMPS}(\psi^{\uparrow l'}, \mathcal{M})$ 
10:       $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}'$ 
11:       $\mathcal{R} \leftarrow \mathcal{R} \cup \{m \cup \{l\} \mid m \in \mathcal{R}'\}$ 
12:   return  $\mathcal{R}$ 

```

---

in Algorithm 2. The main difference in this variant consists in returning besides the prime implicants the partial interpretations that allow to avoid them. The base idea in the for-loop of FPI consists in considering the case of each clause of the form  $l \vee l'$  by removing the literal  $l'$  to force  $l$  to be in the computed prime implicants. Furthermore, the partial interpretations returned with the prime implicants in VPRIMIMPS are used to do not compute any prime implicant more than one time.

The following theorem shows the soundness of the algorithm FPI.

**Theorem 8.** *Let  $\phi$  be a binary CNF formula and  $l$  a literal in  $\phi$  s.t.  $l$  is not a unit clause in  $\phi$ . Then, for all Boolean interpretation  $m$  of  $\phi$  containing  $l$ ,  $m$  is a prime implicant of  $\phi$  iff there exists  $l' \in \mathcal{A}(\phi, l)$  such that  $m \setminus \{l\}$  is a prime implicant of  $(\phi|_l)^{\uparrow l'}$ .*

*Proof.*

*Part  $\Rightarrow$ .* Let  $m$  be a prime implicant of  $\phi$  s.t.  $l \in m$ . Then, there exists  $l' \in \mathcal{A}(\phi)$  s.t.  $l' \notin m$  since  $l \in m$ . As a consequence,  $m \setminus \{l\}$  satisfies  $(\phi|_l)^{\uparrow l'}$ . Assume now that  $m \setminus \{l\}$  is not a prime implicant of  $(\phi|_l)^{\uparrow l'}$ . Then, there exists  $l_0 \in m \setminus \{l\}$  s.t.  $l'' \in (m \setminus \{l\}) \setminus \{l_0\}$  for every clause  $l_0 \vee l''$  in  $(\phi|_l)^{\uparrow l'}$ . Clearly, the clauses containing  $l_0$  in  $\phi$  are those containing  $l_0$  in  $(\phi|_l)^{\uparrow l'}$ . Thus,  $m \setminus \{l_0\}$  satisfies  $\phi$  and we get a contradiction.

*Part  $\Leftarrow$ .* Assume that there exists  $l' \in \mathcal{A}(\phi, l)$  and a prime implicant  $m$  of  $(\phi|_l)^{\uparrow l'}$  such that  $m \cup \{l\}$  is not a prime implicant of  $\phi$ . Clearly,  $m \cup \{l\}$  satisfies  $\phi$ , since  $l$  satisfies all the clauses containing  $l$  and  $m$  satisfies the other clauses. Then, there exists  $l_0 \in m \cup \{l\}$  such that  $(m \cup \{l\}) \setminus \{l_0\}$  satisfies  $\phi$ . Knowing that  $l'$  is not a literal of  $(\phi|_l)^{\uparrow l'}$  ( $l' \notin m$ ),  $l_0 \neq l'$  holds. Moreover, using the fact that  $\phi$  contains the clause  $l \vee l'$  and  $l' \notin m$ ,  $l_0 \neq l$  holds. Furthermore, if  $l_0$  is a unit clause in  $(\phi|_l)^{\uparrow l'}$  then  $l_0$  is also a unit clause in  $\phi$  or the clause  $l_0 \vee l'$  belongs to  $\phi$ , and as a consequence, we get a contradiction since  $l' \notin m$ . Then, there exists a clause  $c = l_0 \vee l''$  in  $(\phi|_l)^{\uparrow l'}$  and  $\phi$  such that  $l'' \notin m \cup \{l\}$  since  $m$  is a prime implicant of  $(\phi|_l)^{\uparrow l'}$  and  $l_0 \in m$ . Thus, using the fact that  $(m \cup \{l\}) \setminus \{l_0\}$  does not satisfy  $c$ , we get a

contradiction. Therefore,  $m \cup \{l\}$  is a prime implicant of the formula  $\phi$ .  $\square$

## Related Works

As mentioned above, a SAT-based encoding for enumerating all the prime implicants of a CNF formula has been proposed in (Jabbour et al. 2014). It consists in encoding the input CNF formula as a new one, so that the models of the encoding represent all the prime implicants of the original formula. Compared to this encoding, our approach provides several advantages for the binary CNF formulas. First, we propose a tractable preprocessing method that reduces the search space. Second, the size of the SAT-encoding of (Jabbour et al. 2014) is much greater than the original formula, while in our approach we consider the original formula. Third, we use a single call to a SAT oracle for all the prime implicants that cover a found model of the original formula, contrary to the SAT-based approach of (Jabbour et al. 2014) where a call to a SAT oracle is used for each prime implicant. Indeed, a MIS generator is used in our approach instead of calls to a SAT oracle to found all the prime implicants that cover a model.

It is worth noticing that the notion of maximal independent set has been used in the context of counting the maximum number of prime implicants of the binary CNF formulas in (Talebanfar 2016). Indeed, the author has used the Theorem of Moon and Moser (Moon and Moser 1965) determining the maximum number of maximal independent sets to propose an upper bound of the number of prime implicants. There is no approach described in this work that uses a generator of MISes for enumerating all the prime implicants of a binary CNF formula.

## Conclusion and Perspectives

In this paper, we proposed an approach for generating all the prime implicants of a binary CNF formula. First, we have presented a preprocessing method for reducing the search space, which uses the unit clauses and the equivalences obtained in the resolution closure. Then, we have described a recursive algorithm for generating the prime implicants of a binary CNF formula that combines the use of a SAT oracle with a generator of maximal independent sets. The latter is used in our algorithm for enumerating all the prime implicants that cover a model found using a SAT oracle. We also proposed a simple approach for dividing the search space that uses an algorithm for generating all the prime implicants that contains a given literal. This algorithm is defined by using our algorithm for generating the prime implicants of a binary CNF formula and by considering each case where the considered literal is possibly in a prime implicant.

A future work, we intend to implement and evaluate the proposed method for generating the prime implicants. We also plan to develop a parallel generator of prime implicants following the presented approach for dividing the search space.

## References

- Acuña, V.; Milreu, P. V.; Cottret, L.; Marchetti-Spaccamela, A.; Stougie, L.; and Sagot, M.-F. 2012. Algorithms and complexity of enumerating minimal precursor sets in genome-wide metabolic networks. *Bioinformatics* 28(19):2474–2483.
- Boros, E.; Elbassioni, K. M.; Gurvich, V.; and Khachiyan, L. 2000. An efficient incremental algorithm for generating all maximal independent sets in hypergraphs of bounded dimension. *Parallel Processing Letters* 10(4):253–266.
- Boufkhad, Y., and Dubois, O. 1999. Length of prime implicants and number of solutions of random CNF formulae. *Theor. Comput. Sci.* 215(1-2):1–30.
- Castell, T. 1996. Computation of prime implicates and prime implicants by a variant of the davis and putnam procedure. In *Eighth International Conference on Tools with Artificial Intelligence, ICTAI '96*, 428–429.
- Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *J. Artif. Intell. Res.* 17:229–264.
- de Kleer, J.; Mackworth, A. K.; and Reiter, R. 1990. Characterizing diagnoses. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI'90)*, 324–330.
- del Val, A. 1994. Tractable databases: How to make propositional unit resolution complete through compilation. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, 551–561.
- Ginsberg, M. L. 1989. A circumscriptive theorem prover. *Artif. Intell.* 39(2):209–230.
- Jabbour, S.; Marques-Silva, J.; Sais, L.; and Salhi, Y. 2014. Enumerating prime implicants of propositional formulae in conjunctive normal form. In *Logics in Artificial Intelligence - 14th European Conference, JELIA 2014*, 152–165.
- Johnson, D. S.; Papadimitriou, C. H.; and Yannakakis, M. 1988. On generating all maximal independent sets. *Inf. Process. Lett.* 27(3):119–123.
- Manquinho, V. M.; Flores, P. F.; Silva, J. P. M.; and Oliveira, A. L. 1997. Prime implicant computation using satisfiability algorithms. In *9th International Conference on Tools with Artificial Intelligence, ICTAI '97*, 232–239.
- McCluskey, E. J. 1956. Minimization of boolean functions. *Bell System Technical Journal* 35(6):1417–1444.
- Moon, J., and Moser, L. 1965. On cliques in graphs. *Israel Journal of Mathematics* 3:23–28.
- Pizzuti, C. 1996. Computing prime implicants by integer programming. In *Eighth International Conference on Tools with Artificial Intelligence, ICTAI '96*, 332–336.
- Quine, W. V. 1952. The problem of simplifying truth functions. *The American Mathematical Monthly* 59(8):521–531.
- Quine, W. V. 1959. On cores and prime implicants of truth functions. *The American Mathematical Monthly* 66(9):755–760.
- Ravi, K., and Somenzi, F. 2004. Minimal assignments for bounded model checking. In *Tools and Algorithms for the Construction and Analysis of Systems, 10th International Conference, TACAS 2004*, 31–45.
- Schrag, R. 1996. Compilation for critically constrained knowledge bases. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96*, 510–515.
- Slavkovik, M., and Ågotnes, T. 2014. A judgment set similarity measure based on prime implicants. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems, AAMAS '14*, 1573–1574.
- Talebanfard, N. 2016. On the structure and the number of prime implicants of 2-CNFs. *Discrete Appl. Math.* 200(C):1–4.
- Tison, P. 1967. Generalization of consensus theory and application to the minimization of boolean functions. *IEEE Transactions on Electronic Computers* EC-16(4):446–456.
- Tsukiyama, S.; Ide, M.; Ariyoshi, H.; and Shirakawa, I. 1977. A new algorithm for generating all the maximal independent sets. *SIAM J. Comput.* 6(3):505–517.