Computing \#2SAT and \#2UNSAT via Binary Patterns

Guillermo De Ita Luna\(^1\) and J. Raymundo Marcial-Romero\(^2\)

\(^1\) Faculty of Computer Sciences, BUAP
\(^2\) Facultad de Ingeniería, UAEM
delta@cs.buap.mx, jrmarcialr@uaemex.mx

Abstract. We present some results about the parametric complexity of \#2SAT and \#2UNSAT, which consist on counting the number of models and falsifying assignments, respectively, for two Conjunctive Forms. Firstly, we show some cases where given a formula \(F\), \#2SAT\((F)\) can be bounded above by considering a binary pattern analysis over its set of clauses. Secondly, since \#2SAT\((F) = 2^n - \#2UNSAT\((F)\) we show that, by considering the constrained graph \(G_F\) of \(F\), if \(G_F\) represents an acyclic graph then, \#UNSAT\((F)\) can be computed in polynomial time. To the best of our knowledge, this is the first time where \#2UNSAT is computed through its constrained graph, since the inclusion-exclusion formula has been commonly used for computing it.

Keywords: \#SAT, Binary Patterns, Enumerative Combinatorics.

1 Introduction

There are many real-life problems that can be abstracted as counting combinatorial objects on graphs. For instance, reliability network issues are often equivalent to connected component issues on graphs, e.g. the probability that a graph remains connected, is given by the probabilities of failure over each edge, which is essentially the same as counting the number of ways that the edges could fail without losing connectivity [9].

The problem of counting models for a Boolean formula (\#SAT problem) can be reduced to several problems in approximate reasoning. For example, estimating the degree of belief in propositional theories, generating explanations to propositional queries, repairing inconsistent databases, Bayesian inference and truth maintenance systems [1, 8–10]. The above problems come from several AI applications such as planning, expert systems, approximate reasoning, etc.

The combinatorial problems that we are going to address are the computation of the number of models and falsifying assignment for Boolean formulas in two Conjunctive Forms (2-CF), denoted as \#2SAT and \#2UNSAT respectively, based on the string patterns formed by its set of falsifying assignments. Both problems are classical \#P-complete problems even for the restricted cases of monotone and Horn formulas.
Among the class of \#P-complete problems, \#2SAT is considered a fundamental instance due to both its application in deduction issues and its relevance to establish a boundary between efficient and intractable counting problems.

Given a 2-CF $F$ with $n$ variables and $m$ clauses, it is common to analyze the computational complexity of the algorithms for solving \#2SAT and \#2UNSAT regarding to $n$ or $m$ or any combination of both [10, 11].

In [3], some cases are presented where \#2SAT($F$) is computed in polynomial time considering the graph-topological structure of the constrained graph of $F$. Additionally, in [2] a new way to measure the degree of difficulty for solving \#2SAT is presented. It is shown that there is a threshold, defined by the same number of models, where \#2SAT is computed in polynomial time.

On the other hand, for computing \#UNSAT($F$) several algorithms have been designed as finer or shorter versions of the application of the inclusion-exclusion formula [6] [4, 7]. However, there are not concise algorithms or procedures which establish the cases in which \#UNSAT($F$) can be computed in polynomial time considering its constrained graph.

Since \#SAT($F$) = $2^n - \#UNSAT(F)$, it is the case that analogous results can be proved for both. In the following section, we present some of those results with the intention of bound the values where \#2SAT and \#2UNSAT can be computed in polynomial time.

In this paper, firstly, we show some cases where \#2SAT($F$) can be bounded above by considering a binary pattern analysis over its set of falsifying assignments. Secondly, since \#2SAT($F$) = $2^n - \#2UNSAT(F)$ we show that, by considering the constrained graph $G_F$ of $F$, if $G_F$ represents an acyclic graph, then \#2UNSAT($F$) can be computed in polynomial time. To the best of our knowledge, this is the first time where \#2UNSAT is computed through its constrained graph, since the inclusion-exclusion formula has been commonly used for this task.

2 Preliminaries

Let $X = \{x_1, \ldots, x_n\}$ be a set of $n$ boolean variables. A literal is either a variable $x_i$ or a negated variable $\overline{x_i}$. As usual, for each $x_i \in X$, $x_i^0 = \overline{x_i}$ and $x_i^1 = x_i$.

A clause is a disjunction of different literals (sometimes, we also consider a clause as a set of literals, e.g. $x_1 \lor x_2 = \{x_1, x_2\}$). For $k \in \mathbb{N}$, a $k$-clause is a clause consisting of exactly $k$ literals. A variable $x \in X$ appears in a clause $c$ if either $x$ or $\overline{x}$ is an element of $c$.

A conjunctive form (CF) $F$ is a conjunction of clauses. We say that $F$ is a monotone positive CF if all of its variables appear in unnegated form. A $k$-CF is a CF containing only $k$-clauses.

We use $v(X)$ to represent the variables involved in the object $X$, where $X$ could be a literal, a clause or a CF. For instance, for the clause $c = \{x_1, x_2\}$, $v(c) = \{x_1, x_2\}$. $\text{Lit}(F)$ is the set of literals appearing in $F$, i.e. if $X = v(F)$, then $\text{Lit}(F) = X \cup \overline{X} = \{x_1, x_2, \ldots, x_n, \overline{x_n}\}$. We denote $\{1, 2, \ldots, n\}$ by $[n]$ and the cardinality of a set $A$ by $|A|$. 
An assignment $s$ for $F$ is a boolean function $s : v(F) \rightarrow \{0, 1\}$. An assignment $s$ can also be considered as a set of non complementary pairs of literals, e.g., if $l \in s$, then $\overline{l} \not\in s$, in other words $s$ turns $l$ true and $\overline{l}$ false. Let $c$ be a clause and $s$ an assignment, $c$ is satisfied by $s$ if and only if $c \cap s \neq \emptyset$. On the other hand, if for all $l \in c, \overline{l} \in s$, then $s$ falsifies $c$. If $n = |v(F)|$, then there are $2^n$ possible assignments defined over $v(F)$. Let $S(F)$ be the set of $2^n$ assignments defined over $v(F)$.

Let $F$ be a CF, $F$ is satisfied by an assignment $s$ if each clause in $F$ is satisfied by $s$. $F$ is contradicted by $s$ if any clause in $F$ is falsified by $s$. A model of $F$ is an assignment for $v(F)$ that satisfies $F$. A falsifying assignment of $F$ is an assignment for $v(F)$ that contradicts $F$. The SAT problem consists of determining whether $F$ has a model. SAT($F$) denotes the set of models of $F$, then SAT($F$) $\subseteq S(F)$. Let UNSAT($F$) $= S(F) - SAT(F)$, i.e. UNSAT($F$) is the set of assignments from $S(F)$ that falsify $F$.

The #SAT problem (or #SAT($F$)) consists of counting the number of models of $F$ defined over $v(F)$. #2SAT denotes #SAT for formulas in 2-CF. So 
$\#\text{UNSAT}(F) = 2^n - \#\text{2SAT}(F)$.

A 2-CF $F$ can be represented by an undirected graph, called the constrained graph of $F$, and determined as: $G_F = (V(F), E(F))$, where $V(F) = v(F)$ and $E(F) = \{\{v(x), v(y)\} : \{x, y\} \in F\}$. I.e. the vertices of $G_F$ are the variables of $F$, and for each clause $\{x, y\}$ in $F$ there is an edge $\{v(x), v(y)\} \in E(F)$.

An algorithm to compute #2SAT($F$), consists on determining the set of connected components of its constrained graph $G_F$. It has been proved that the set of connected components of a constrained graph can be determined in linear time with respect to the number of clauses in the formula.

Thus, $\#\text{2SAT}(F) = \#\text{2SAT}(G_F) = \prod_{i=1}^{k} \#\text{2SAT}(G_i)$, where $\{G_1, \ldots, G_k\}$ is the set of connected components of $G_F$ [8].

The set of connected components of $G_F$ conforms a partition of $F$. So, from now on, we will work with a connected component graph. We say that a 2-CF $F$ is a path, a cycle, or a tree if its corresponding constrained graph $G_F$ represents a path, a cycle, or a tree, respectively.

3 Computing #2SAT via Binary Patterns

Let $F = \{C_1, C_2, \ldots, C_m\}$ be a strict 2-CF (each clause has length 2) and let $n = |v(F)|$. The size of $F$ is the sum of the number of clauses and variables, i.e. $n + m$. Let $k$ be a positive integer parameter such that $k < 2^n$. The values of $k$ where #2SAT($F$) $= k$ can be determined in polynomial time have been proved for the following cases.

If $k = 0$ or $k = 1$ the Transitive Closure procedure presented in [5] can be applied. Such procedure has a linear time complexity on the size of the 2-CF.

If $k$ is upper bounded by a polynomial function on $n$, e.g. $k \leq p(n)$, then in [2], an exact algorithm was shown for determining when #SAT($F$) $\leq p(n)$. Such algorithm has a polynomial time complexity on the size of $F$. 

So, the hard cases to answer whether \#2SAT\((F) = k\) are when the parameter \(k\) is a value higher than \(p(n)\). The following results show how the parameter \(k\) can also be upper bounded.

**Lemma 1.** Let \(F = \{C_1, \ldots, C_m\}\) be a 2-CF and \(n = |v(F)|\). \#2SAT\((F) < 2^n - 2^{n-2}\) (which is analogous to say that \#2UNSAT has at least \(2^{n-2}\) elements).

**Proof.** Let \(C = \{l_i, l_j\}\) be a clause of \(F\) and \(s\) a falsifying assignment of \(C\). As we assume that \(C\) is not a tautology then \(v(l_i) \neq v(l_j)\), and as \(s\) falsifies \(C\) then \(l_i \in s\) and \(l_j \in s\). So two of the \(n\) positions in the assignment have fixed values, and there are \(n - 2\) different variables that can be assigned any truth value. That means that there are \(2^{n-2}\) possible assignments that falsify \(C\). Hence, from the \(2^n\) assignments, \(2^{n-2}\) are falsified by \(C\). Thus, \#2SAT\((F)\) is not bigger than \(2^n - 2^{n-2}\).

**Corollary 1.** Let \(F = \{C_1, C_2, \ldots, C_m\}\) be a 2-CF. The number \(A\) of assignments of \(F\) which falsify \(F\) is bounded by \(2^{n-2} \leq A \leq m(2^{n-2})\).

**Proof.** If each of the \(m\) clauses have literals coming from different variables, then it is a fact that \(m(2^{n-2})\) assignments are not models. On the other hand, if the literals of the \(m\) clauses come from the same variables, then it is a fact that at least \(2^{n-2}\) assignments are not models of \(F\).

So, given a 2-CF \(F\), if we assume \#2SAT\((F) = k\) such that \(k > p(n)\) for a polynomial \(p(n)\), then \(2^n - m(2^{n-2}) < k < 2^n - 2^{n-2}\). And in order to know if \#2SAT\((F) = k\) we can test first if \(k > 2^n - m(2^{n-2})\) holds.

Let \(F = \{C_1, C_2, \ldots, C_m\}\) be a 2-CF and \(n = |v(F)|\). Assume an enumeration over the variables of \(v(F)\), e.g. \(x_1, x_2, \ldots, x_n\). For each \(C_i = \{x_j, x_k\}\), let \(A_i\) be a set of binary strings such that the length of each string is \(n\). The values at the \(j\)-th and \(k\)-th positions of each string, \(1 \leq j, k \leq n\) represent the truth value of \(x_j\) and \(x_k\) that falsifies \(C_i\). E.g., if \(x_j \in C_i\) then the \(j\)-th element of \(A_i\) is set to 0. On the other hand, if \(x_j \notin C_i\) then the \(j\)-th element of \(A_i\) is set to 1. The same argument applies to \(x_k\). It is easy to show that if \(C_i = \{x_j, x_k\}\), then \(x_j\) and \(x_k\) have the same values in each string of \(A_i\).

**Example 1.** Let \(F = \{C_1, C_2\}\) be a 2-CF and \(|v(F)| = 3\). If \(C_1 = \{x_1, x_2\}\) and \(C_2 = \{x_2, x_3\}\) then \(A_1 = \{000, 001\}\) and \(A_2 = \{000, 100\}\).

We will use the symbol \(\ast\) to represent the elements that can take any truth value in the set \(A_i\), e.g. if \(F = \{C_1, \ldots, C_m\}\) is a 2-CF, \(n = |v(F)|\), \(C_1 = \{x_1, x_2\}\) and \(C_2 = \{x_2, x_3\}\) then we will write \(A_1 = 00\ast \ast \ast \ast\) and \(A_2 = \ast 00\ast \ast \ast \ast\). This abuse of notation will allow us to present a concise and clear representation in the rest of the paper, for considering the string \(A_i\) as a binary pattern that represents the falsifying assignments of the clause \(C_i\).

It is known [4] that for any two pair of clauses \(C_i\) and \(C_j\), it holds that \#\text{UNSAT}(C_i \cup C_j) = #\text{UNSAT}(C_i) + #\text{UNSAT}(C_j) - #\text{UNSAT}(C_i \cap C_j).\) The following lemmas show when the number of models can be reduced.
Lemma 2. Let $F$ be a 2-CF, $n = |\nu(F)|$. If $C_i \in F$ and $C_j \in F$, $i \neq j$ have not complementary pairs of literals and they share a literal (e.g. $C_i \cap C_j \neq \emptyset$), then there are exactly $2^{n-1} - 2^{n-3}$ assignments from $S(F)$ falsifying $C_i \cup C_j$.

Proof. Since $C_i \cap C_j \neq \emptyset$ the elements $A_i$ and $A_j$ have a same value in the common literal (e.g. $A_i \cap A_j = \ast \ldots \ast 0 \ast \ldots \ast 0 \ast \ldots \ast$) which represent $2^{n-3}$ assignments. That means that $2^{n-2} + 2^{n-2} - 2^{n-3} = 2^{n-1} - 2^{n-3}$ assignments from $S(F)$ are falsified.

Example 2. If $C_1 = \{x_1, x_2\}$ and $C_2 = \{x_2, x_3\}$ then $A_1 = 00 \ast \ldots \ast$ and $A_2 = \ast 00 \ast \ldots \ast$. As $C_1 \cap C_2 = \{x_2\}$, the common pattern assignment $000 \ast \ldots \ast$ falsifies both clauses and there are exactly $2^{n-2} + 2^{n-2} - 2^{n-3} = 2^{n-1} - 2^{n-3}$ assignments from $S(F)$ are falsified.

Lemma 3. Let $F$ be a 2-CF, $n = |\nu(F)|$. If $C_i \in F$ and $C_j \in F$, $i \neq j$ contain complementary literals, that is $x_k \in C_i$ and $\overline{x_k} \in C_j$, the unsatisfied set of assignments $A_i$ and $A_j$ form a disjoint set of assignments. Consequently, both clauses suppress exactly $2^{n-2} + 2^{n-2} = 2^{n-1}$ assignments from $S(F)$.

Definition 1. [4] If two clauses of $F$ have at least one complementary literal, it is said that they have the independence property. Otherwise, we say that both clauses are dependent.

Theorem 1. Let $F = \{C_1, C_2, \ldots, C_m\}$ be a 2-CF, $n = |\nu(F)|, m \geq 2$. The hard cases to answer whether $\#\text{SAT}(F) = k$, are given when $m > n$.

Proof. By the previous lemmas, the clauses in $F$ suppress between $2^n - m(2^{n-2})$ and $2^{n-1} - 2^{n-3}$ assignments from $S(F)$. If $m \leq n$, then almost all exact procedures can compute $\#\text{SAT}(F)$ in polynomial time [3]. Thus, the hard cases for answer whether $\#\text{SAT}(F) = k$ have to consider that $m > n$.

4 Polynomial Time Procedures for $\#2\text{UNSAT}$

Given a 2-CF $F = \{C_1, \ldots, C_m\}$, let $A_i$ be the set of assignments from $S(F)$ falsifying $C_i$. The number of unsatisfied assignments ($\#\text{UNSAT}(F)$) can be counted by the inclusion-exclusion formula, (e.g. $\#\text{UNSAT}(F) = \big| \bigcup_{i=1}^{m} A_i \big|$), in the following way:

$$\big| \bigcup_{i=1}^{m} A_i \big| = \sum_{i=1}^{m} |A_i| - \sum_{i<j} |A_i \cap A_j| + \sum_{i<j<k} |A_i \cap A_j \cap A_k| + \ldots + (-1)^{m-1} \big| \bigcap_{i=1}^{m} A_i \big|$$

For computing $\#\text{UNSAT}(F)$ several algorithms have been designed as finer versions of the application of the inclusion-exclusion formula [4, 7]. However, there are not concise algorithms which establish when $\#\text{UNSAT}(F)$ can be computed in polynomial time considering its constrained graph.
In [3], considering the graph-topological structure of $F$, $\#\text{SAT}(F)$ is computed in polynomial time for some classes of formulas. Since $\#\text{SAT}(F) = 2^n - \#\text{UNSAT}(F)$, it is the case that analogous results can be proved for $\#\text{UNSAT}(F)$.

Let $F$ be a 2-CF and $n=|v(F)|$, for each variable $x \in v(F)$, a pair $(\alpha_x, \beta_x)$ called the charge, is used to indicate the number of logical values: 'true' and 'false' respectively, that $x$ takes when $\#\text{UNSAT}(F)$ is being computed.

If $G_F$ is a path, then $F = \{C_1, C_2, \ldots, C_m\} = \{\{x_1, x_2\}, \{x_2'\}, \ldots, \{x_m', x_{m+1}\}\}$, where $\delta_i, \epsilon_i \in \{0,1\}, i \in [m]$. Let $f_i$ be a family of clauses of the formula $F$, built as follows: $f_1 = 0; f_i = \{C_j\}_{j<i}, i \in [m]$. Notice that $n = |v(F)| = m + 1, f_i \subset f_{i+1}, i \in [m-1]$. Let $\text{UNSAT}(f_i) = \{s : s \text{ falsifies } f_i\}$,

$$A_i = \{s \in \text{UNSAT}(f_i) : x_i \in s\}, B_i = \{s \in \text{UNSAT}(f_i) : \overline{x}_i \in s\}.$$

Let $\alpha_i = |A_i|$, $\beta_i = |B_i|$ and $\mu_i = |\text{UNSAT}(f_i)| = \alpha_i + \beta_i$.

The first pair $(\alpha_1, \beta_1)$ is $(0,0)$ since there is no logical value for $x_1$ such that $f_1$ is falsified. We compute $(\alpha_i, \beta_i)$ associated with each variable $x_i, i = 2, \ldots, n$, according to the signs: $\epsilon_i, \delta_i$ of the literals in the clause $C_j, j = 1, \ldots, m$.

**Lemma 4.** Let $F$ be a 2-CF where $G_F$ represents a path. If $(\epsilon_1, \delta_1) = (0,0)$ then $(\alpha_1, \beta_1) = (\mu_{i-1} + (2^{i-2} - \alpha_{i-1}), \mu_{i-1})$.

**Proof.** Assume that $(\alpha_j, \beta_j)$ are correct for $0 \leq j < i$. We show that $(\alpha_i, \beta_i) = (\mu_{i-1} + (2^{i-2} - \alpha_{i-1}), \mu_{i-1})$. That $(\epsilon_i, \delta_i) = (0,0)$ means the clause $C_i = \{\overline{x}_i, x_i\}$. The truth values $x_{i-1} = 1$ and $x_i = 1$ falsify $C_i$. By lemma 1, $C_i$ falsifies $2^{i-2}$ assignments in addition to the $\mu_{i-1}$ already counted. However, that $C_i \cap C_{i-1} = \{\overline{x}_i\}$ (given that $G_F$ is a path) means the cases were $x_{i-1} = 1$ have already been counted and in consequence have to be removed (lemma 2). This value is given by $\alpha_{i-1}$. So to compute $\alpha_i$ to the $\mu_{i-1}$ falsifying assignments, $2^{i-2}$ falsifying assignments have to be added and $\alpha_{i-1}$ subtracted. That $\beta_i = \mu_i$ follows by the fact that $x_i = 0$ does not falsify $C_i$.

**Lemma 5.** Let $F$ be a 2-CF where $G_F$ represents a path. If $(\epsilon_1, \delta_1) = (0,1)$ then $(\alpha_1, \beta_1) = (\mu_{i-1}, \mu_{i-1} + (2^{i-2} - \alpha_{i-1}))$.

**Proof.** That $(\epsilon_1, \delta_1) = (0,1)$ means the clause $C_1 = \{\overline{x}_1, x_1\}$. That $\alpha_1 = \mu_1$ follows by the fact that $x_1 = 1$ does not falsify $C_1$. The truth values $x_{i-1} = 1$ and $x_i = 0$ falsify $C_i$. By lemma 1, $C_i$ falsifies $2^{i-2}$ assignments in addition to the $\mu_{i-1}$ already counted. However, that $C_i \cap C_{i-1} = \{\overline{x}_i\}$ means the cases were $x_{i-1} = 1$ have already been counted and thus have to be removed (lemma 2). This value is given by $\alpha_{i-1}$. So to compute $\beta_i$ to the $\mu_{i-1}$ falsifying assignments, $2^{i-2}$ falsifying assignments have to be added and $\alpha_{i-1}$ subtracted.

**Lemma 6.** Let $F$ be a 2-CF where $G_F$ represents a path. If $(\epsilon_1, \delta_1) = (1,0)$ then $(\alpha_1, \beta_1) = (\mu_{i-1} + (2^{i-2} - \beta_{i-1}), \mu_{i-1})$.

**Proof.** Similar to lemma 4.

**Lemma 7.** Let $F$ be a 2-CF where $G_F$ represents a path. If $(\epsilon_1, \delta_1) = (1,1)$ then $(\alpha_1, \beta_1) = (\mu_{i-1}, \mu_{i-1} + (2^{i-2} - \beta_{i-1}))$. 
Proof. Similar to lemma 5.

**Theorem 2.** Let $F = \{C_1, C_2, \ldots, C_m\}$ be a 2-CF where $G_F$ represents a path. The recurrence which compute $\#\text{UNSAT}(F)$ is given by:

$$
\begin{align*}
(\alpha_i, \beta_i) &= \begin{cases} \\
(\mu_{i-1} + (2i-2 - \alpha_{i-1}), \mu_{i-1}) & \text{if } (\epsilon_i, \delta_i) = (0, 0) \\
(\mu_{i-1} + (2i-2 - \alpha_{i-1}), \mu_{i-1}) & \text{if } (\epsilon_i, \delta_i) = (1, 1) \\
(\mu_{i-1} + (2i-2 - \beta_{i-1})), \mu_{i-1}) & \text{if } (\epsilon_i, \delta_i) = (1, 0) \\
(\mu_{i-1} + (2i-2 - \beta_{i-1})), \mu_{i-1}) & \text{if } (\epsilon_i, \delta_i) = (0, 1) \\
\end{cases}
\end{align*}
$$

**Proof.** Derived from the previous lemmas.

As $F = f_m$ then $\#\text{UNSAT}(F) = \mu_m = \alpha_m + \beta_m$. We denote with $\rightarrow$ the application of one of the four rules in the recurrence of theorem 2. We can apply in linear time $O(n)$ the recurrence equations of theorem 2 while we are visiting in depth-first search each node of the path $G_F$. And the last pair $(\alpha_n, \beta_n)$ holds that $\#\text{UNSAT}(F) = \mu_n = \alpha_n + \beta_n$.

**Example 3.** Figure 1 shows the application of theorem 2 for the formula $F = \{(x_1, x_2), (x_2, x_3), (x_3, x_4), (x_4, x_5)\}$. For such case all the variables are positive.

When $F$ is a monotone positive 2-CF, for example figure 1, the values in the pairs $(\alpha, \beta)$, $i = 1, \ldots, n$ can be related to the Fibonacci numbers as follows.

**Lemma 8.** Let $F = \{C_1, C_2, \ldots, C_m\}$ be a monotone positive 2-CF. The recurrence for computing the pair $(\alpha_i, \beta_i)$, $i = 2, 3, \ldots, m$ is given by:

$$
\alpha_i = \alpha_{i-1} + \beta_{i-1}; \quad \beta_i = \alpha_{i-1} + \beta_{i-1} + F_{i-1}
$$

where $F_{i-1}$ is the $(i-1)$-th Fibonacci number.

**Proof.** We have to show that $F_{i-1} = 2^{i-2} - \beta_{i-1}$. If $i = 2$, $F_2 = 2^0 = 0 = 1$ holds. Assume it holds for $j < i - 1$. We show that it holds for $i - 1$.

$$
\begin{align*}
F_{i-1} &= F_{i-2} + F_{i-3} \\
&= 2^{i-3} - \beta_{i-2} + 2^{i-4} - \beta_{i-3} \\
&= 2^{i-3} + 2^{i-4} - (\beta_{i-2} + \beta_{i-3}) \\
&= 2^{i-3} + 2^{i-4} - (\beta_{i-2} + \beta_{i-3}) \\
&= 2^{i-3} + 2^{i-4} - \beta_{i-1} = 2^{i-2} - \beta_{i-1}
\end{align*}
$$

**Corollary 2.** Let $F = \{C_1, C_2, \ldots, C_m\}$ be a positive monotone 2-CF.

$$
\#\text{UNSAT}(F) = \sum_{i=1}^{n} 2^{n-i} \cdot F_{i-1}
$$
Proof. \( \#\text{UNSAT}(F) = \alpha_n + \beta_n = \alpha_{n-1} + \beta_{n-1} + \alpha_{n-1} + \beta_{n-1} + F_{n-1} = 2(\alpha_{n-1} + \beta_{n-1}) + F_{n-1} = 2(\#\text{UNSAT}(F_{n-1})) + F_{n-1} \).

Unfolding the sum until the first values \((\alpha_1, \beta_1)\), we obtain the following result.

**Corollary 3.** \( \sum_{i=1}^{n} 2^{n-i} \cdot F_{i-1} + F_{n+2} = 2^n \).

**Proof.** Use the facts: \( \#\text{2SAT}(F) + \#\text{2UNSAT}(F) = 2^n \) and \( \#\text{2SAT}(F) = F + 2 \).

**Definition 2.** The constrained graph \( G_F \) of a 2-CF formula \( F = \{C_1, \ldots, C_m\} \) is called a tree if the following holds:

1. There exist \( j \) clauses \( 1 < j < n \) such that \( \bigcap C_j \neq \emptyset \).
2. For any \( F' \subset F \) whose constrained graph is a path, there is not transitive clauses from \( F' \) in \( F \), and \( F \) has not cycles.

By definition a tree consists of paths which intersect at some point, with the exception that they can not form cycles. Given a tree \( G_F \), we call leaf-edges to the edges with one endpoint in a leaf node of \( T \). All the leaves in a tree have associated the initial charge \((0, 0)\) since just one variable (each leaf node) can not falsify its related clause (the leaf-edge).

**Lemma 9.** Let \( F = \{C_1, \ldots, C_m\} \) be a 2-CF where \( G_F \) is a tree. For each father node \( x_p \) with two branches, one of size \( i \) (it has \( i \) clauses), and the other of size \( j \) \((j \) clauses are involved), the recurrence for updating the charge associated to the node \( x_p; (\alpha_p, \beta_p) \), is given as:

\[
\begin{align*}
\alpha_p &= \alpha_i \cdot 2^{j-1} + \alpha_j \cdot 2^{i-1} - \alpha_i \cdot \alpha_j \\
\beta_p &= \beta_i \cdot 2^{j-1} + \beta_j \cdot 2^{i-1} - \beta_i \cdot \beta_j
\end{align*}
\]  

**Proof.** When two different branches, one of size \( i \) and the other of size \( j \) meet in the same variable \( x_p \) (the father node), \((\alpha_i, \beta_i)\) denotes the number of falsifying assignments for the branch \( i \) then it updates considering all the assignments for the other branch \((2^{i-1} \) assignments). A similar argument is given when the pair \((\alpha_j, \beta_j)\) is considered. Finally, we have to subtract the common assignments in the variable \( x_p; \alpha_i \cdot \alpha_j \) for the value \( \alpha_p \) and \( \beta_i \cdot \beta_j \) for \( \beta_p \).

It is obvious that acyclic graphs can be represented by trees. So, we have the following theorem.

**Theorem 3.** Let \( F \) be a 2-CF where \( G_F \) is a tree. \( \#\text{2UNSAT}(F) \) is computed in polynomial time on the number of nodes in \( G_F \).

**Proof.** A postorder traversal of the tree computes \( \#\text{UNSAT}(F) \) while the recurrences (1) and (2) are applied, computes \( \#\text{2UNSAT}(F) \).

Further results can be presented if a especial class of formulas is considered.
Definition 3. Let $F$ be a 2-CF, if the constrained graph of $F$ is a tree of height one and all its clauses are dependent pairwise (as shown in Figure 2) then the constrained graph is called a minimum dependent tree.

If the formula $F$ is monotone positive (or monotone negative) and its constrained graph is a minimum dependent tree, we can further bound the number of falsifying assignments of $F$.

Lemma 10. Let $F$ be a monotone positive 2-CF. If $G_F$ is a minimum dependent tree then its number of falsifying assignments is $2^n - 1$.

Proof. If $G_F$ is a minimum dependent tree means that $F$ is of the form $F_n = \{C_1, \ldots, C_{n-1}\} = \{(x_1, x_2), (x_1, x_3), \ldots, (x_1, x_n)\}$. As all clauses are dependent pairwise, they get for any pair of clauses $C_i, C_j \in F_n$, a less value for $\#\text{UNSAT}(C_i \cup C_j)$ (by lemma 2 and 3). The computation of $\#\text{UNSAT}(F_n)$ can be done in incremental way, in the following way. Considering only $C_1$, this is falsified by the pattern $A_1 = 00$ and the charge of $x_1$ is $(\alpha_1, \beta_1) = (0, 1)$. Similarly, considering each clause $C_i, i = 2, \ldots, n - 1$ in an independent way to the other clauses, it determines a charge for $x_1$ of $(\alpha_1, \beta_1) = (0, 1)$. As $C_1$ and $C_2$ are two branches with the same father node ($x_1$), the subtree has two leaves ($x_2$ and $x_3$). According to (2), the new $(\alpha_1, \beta_1)$ is updated as: $\alpha_1 = 0$ and $\beta_1 = 3$, because we have two subtrees, one for $C_1$ and one for $C_2$, both of size 2. When the third clause $C_3$ is considered, we have the charge for $x_1$ as (0,3) for the subtree of size 3 (clauses $C_1$ and $C_2$), while (0,1) is associated to the subtree of size 2 (clause $C_3$), and according to equation (2), we update the charge for $x_1$ as: $\alpha_1 = 0$ and $\beta_1 = 7$. Notice that $\alpha_1$ is always 0 since it is not possible to assign $x_1 = 1$ in order to falsify any clause in $F_n$. While the value for $\beta_1$ for the clause $C_i \in F_n$ can be deduced from the value of $\beta_1$ associated to the clause $C_{i-1}$ in the following way: $\beta_1 = 2 \cdot \beta_i + 1$. Then if there are $n - 1$ clauses, the final value for $\beta_1$ will be $\beta_1 = 2^{n-2} \cdot 1 + 2^{n-3} + \ldots + 2^0 = \sum_{i=0}^{n-2} 2^i = 2^n - 1$.

Thus, the number of falsifying assignments for this class of formulas is always the half minus 1 of the total of assignments. Notice that similar value is obtained for a monotone negative formula $F_n$, although now $\beta_1 = 0$ and $\alpha_1 = 2^n - 1$. Then, the value $2^n - 1$ is a lower bound for the value $\#\text{UNSAT}(F), n = |\nu(F)|$.

As $\#\text{UNSAT}(F_n) + \#\text{SAT}(F_n) = 2^n$, the formulas where $\#\text{UNSAT}(F_n) < \#\text{SAT}(F_n)$ are those whose constrained graph are minimum dependent trees. For any other Boolean formula $F, n = |\nu(F)|$, we will have that $\#\text{UNSAT}(F) \geq$
2^{n-1}. Remember that the hard cases to answer whether \( \#\text{SAT}(F) = k \), is when \( k > p(n) \) for a polynomial \( p(n) \) where \( n = |v(F)| \). Discarding the minimum dependent trees, we have that \( 2^n = \#\text{UNSAT}(F_n) + \#\text{SAT}(F_n) \geq 2^{n-1} + k \) then the value \( k \) is upper bounded by \( 2^n - 2^{n-1} = 2^{n-1} \). So, the hard cases for answer whether \( \#\text{SAT}(F) = k \), are those where \( p(n) < k < 2^{n-1} \).

5 Conclusions

Given a 2-CF \( F \) with \( n \) variables, \( m \) clauses, and a positive integer parameter \( k \), the question: Is \( \#\text{SAT}(F) = k? \), can be answered in an efficient way for values of \( k \); \( k \leq p(n) \) where \( p(n) \) is a polynomial on \( n \), and when \( k \geq 2^{n-1} \).

When \( k \) is outside of those interval values, is convenient to analyze the topology of \( G_F \) (the constrained graph of \( F \)). As \( \#\text{SAT}(F) = 2^n - \#\text{UNSAT}(F) \), then \( \#\text{UNSAT}(F) \) will be an upper bound for \( k \). And if \( G_F \) represents an acyclic graph, then \( \#\text{UNSAT}(F) \) can be computed in polynomial time. To the best of our knowledge, this is the first time where \( \#2\text{UNSAT} \) is computed through its constrained graph.

References