Computing #2SAT and #2UNSAT by Binary Patterns

Guillermo De Ita Luna¹ and J. Raymundo Marcial-Romero²

¹ Faculty of Computer Sciences, BUAP

² Facultad de Ingeniería, UAEM

deita@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 (2-CF's). 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 #UNSAT(F).

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 [1,8,9]. 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 #UNSAT respectively, based on the string patterns formed by its set of clauses. Both problems are classical #P-complete problem 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 where $\#SAT(F)$ is computed in polynomial time considering graph-topological structures are presented. 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 $\#SAT$ and $\#UNSAT$ 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 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 this task.

2 Preliminaries

Let $X = \{x_1, \dots, x_n\}$ 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. $x_1 \vee x_2 = \{x_1, x_2\}$). 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 $c = \{\overline{x_1}, x_2\}$, $v(c) = \{x_1, x_2\}$. $Lit(F)$ is the set of literals appearing in F , i.e. if $X = v(F)$, then $Lit(F) = X \cup \overline{X} = \{x_1, \overline{x_1}, \dots, x_n, \overline{x_n}\}$. We denote $\{1, 2, \dots, n\}$ by $\llbracket n \rrbracket$ 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 a set of non complementary pairs of literals, e.g., if $l \in s$, then $\bar{l} \notin s$, in other words s turns l *true* and $\bar{l}$ *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 $l \in c, \bar{l} \in s$, 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. $\text{SAT}(F)$ denotes the set of models of F , then $\text{SAT}(F) \subseteq S(F)$. Let $\text{UNSAT}(F) = S(F) - \text{SAT}(F)$, i.e. $\text{UNSAT}(F)$ 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 $\#2\text{UNSAT}(F) = 2^n - \#2\text{SAT}(F)$.

A 2-CF formula F can be represented by an undirected graph. Let F be a 2-CF, the *constrained graph* of F is the undirected graph $G_F = (V(F), E(F))$, 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 $\#2\text{SAT}(F)$, 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,

$\#2\text{SAT}(F) = \#2\text{SAT}(G_F) = \prod_{i=1}^k \#2\text{SAT}(G_i)$, where $\{G_1, \dots, G_k\}$ 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 by Binary Patterns

Let $F = \{C_1, C_2, \dots, C_m\}$ 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 $\#2\text{SAT}(F) = k$ can be determined in polynomial time have been proved for the following cases.

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

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

So, the hard cases to answer whether $\#2SAT(F) = k$ appear 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 $F = \{C_1, C_2, \dots, C_m\}$ be a 2-CF and $n = |v(F)|$. If $m > 1$ then $\#2SAT(F) < 2^n - 2^{n-2}$ (which is analogous to say that $\#2UNSAT$ has at least 2^{n-2} elements).*

Proof. Let $C = (l_i, l_j)$ be a clause and s an assignment that falsify C . As we assume that C is not a tautology then $v(l_i) \neq v(l_j)$, and as s falsifies C then $\bar{l}_i \in C$ and $\bar{l}_j \in C$. 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 $2^n - 2^{n-2}$.

So, given a 2-CF F , if we assume $\#2SAT(F) = k$ such that $k > p(n)$ for a polynomial $p(n)$, then $k < 2^n - 2^{n-2}$.

Corollary 1. *Let $F = \{C_1, C_2, \dots, C_m\}$ 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 that $k > 2^n - m(2^{n-2})$ holds.

Let $F = \{C_1, C_2, \dots, C_m\}$ be a 2-CF and $n = |v(F)|$. Assume an enumeration over the variables of $v(F)$, e.g. $x_1, x_2, \dots, x_n$. 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 $\bar{x}_j \in C_i$ 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 $*$ to represent the elements that can take any truth value in the set A_i , e.g. if $F = \{C_1, C_2, \dots, C_m\}$ is a 2-CF, $n = |v(F)|$, $C_1 = \{x_1, x_2\}$ and $C_2 = \{x_2, x_3\}$ then we will write $A_1 = 00 * \dots *$ and $A_2 = *00 * \dots *$. 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 falsified 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 $\#UNSAT(C_i \cup C_j) = \#UNSAT(C_i) + \#UNSAT(C_j) - \#UNSAT(C_i \cap C_j)$. The following lemmas show when the number of models can be reduced.

Lemma 2. *Let F be a 2-CF, $n = |v(F)|$. 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. $A_i \cap A_j = * \dots * 0 * \dots * 0 * \dots * 0 * \dots *$) 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 are falsified.

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

Lemma 3. *Let F be a 2-CF, $n = |v(F)|$. 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 $F = \{C_1, C_2, \dots, C_m\}$ be a 2-CF and $n = |v(F)|$. The hard cases to answer whether $\#SAT(F) = k$, are given when $m > n$.*

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

4 Polynomial Time Procedures for #2UNSAT

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

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

For computing $\#UNSAT(F)$ 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 $\#UNSAT(F)$ can be computed in polynomial time considering its constrained graph.

In [3], considering the graph-topological structure of F , $\#SAT(F)$ is computed in polynomial time for some classes of formulas. Since $\#SAT(F) = 2^n - \#UNSAT(F)$, it is the case that analogous results can be proved for $\#UNSAT(F)$.

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

If G_F is a path, then $F = \{C_1, C_2, \dots, C_m\} = \{\{x_1^{\epsilon_1}, x_2^{\delta_1}\}, \{x_2^{\epsilon_1}, x_3^{\delta_1}\}, \dots, \{x_m^{\epsilon_m}, x_{m+1}^{\delta_m}\}\}$, where $\delta_i, \epsilon_i \in \{0, 1\}$, $i \in \llbracket m \rrbracket$. Let f_i be a family of clauses of a formula F , built as follows: $f_1 = \emptyset$; $f_i = \{C_j\}_{j < i}$, $i \in \llbracket m \rrbracket$. Notice that $n = |v(F)| = m + 1$, $f_i \subset f_{i+1}$, $i \in \llbracket m-1 \rrbracket$. Let $UNSAT(f_i) = \{s : s \text{ falsifies } f_i\}$, $A_i = \{s \in UNSAT(f_i) : x_i \in s\}$, $B_i = \{s \in UNSAT(f_i) : \bar{x}_i \in s\}$. Let $\alpha_i = |A_i|$; $\beta_i = |B_i|$ and $\mu_i = |UNSAT(f_i)| = \alpha_i + \beta_i$.

The first pair (α_1, β_1) is $(0, 0)$ since there is not logical value for x_1 such that f_1 is falsified. We compute (α_i, β_i) associated with each variable x_i , $i = 2, \dots, n$, according to the signs: ϵ_i, δ_i of the literals in the clause C_j , $j = 1, \dots, m$.

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

Proof. Assume that (α_j, β_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 $C_i = \{\bar{x}_{i-1}, \bar{x}_i\}$. 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 μ_{i-1} already counted. However, that $C_i \cap C_{i-1} = \{\bar{x}_{i-1}\}$ (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 α_{i-1} . So to compute α_i to the μ_{i-1} falsifying assignments, 2^{i-2} falsifying assignments have to be added and α_{i-1} subtracted. That $\beta_i = \mu_{i-1}$ 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 $(\epsilon_i, \delta_i) = (0, 1)$ then $(\alpha_i, \beta_i) = (\mu_{i-1}, \mu_{i-1} + (2^{i-2} - \alpha_{i-1}))$.*

Proof. That $(\epsilon_i, \delta_i) = (0, 1)$ means the clause $C_i = \{\bar{x}_{i-1}, x_i\}$. That $\alpha_i = \mu_i$ follows by the fact that $x_i = 1$ does not falsify C_i . 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 μ_{i-1} already counted. However, that $C_i \cap C_{i-1} = \{\bar{x}_{i-1}\}$ means the cases were $x_{i-1} = 1$ have already been counted and then have to be removed (lemma 2). This value is given by α_{i-1} . So to compute β_i to the μ_{i-1} falsifying assignments, 2^{i-2} falsifying assignments have to be added and α_{i-1} subtracted.

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

Proof. Similar to lemma 4.

Lemma 7. Let F be a 2-CF where G_F represents a path. If $(\epsilon_i, \delta_i) = (1, 1)$ then $(\alpha_i, \beta_i) = (\mu_{i-1}, \mu_{i-1} + (2^{i-2} - \beta_{i-1}))$.

Proof. Similar to lemma 5.

Fig. 1. Counting falsified assignments for a monotone positive path

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

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

Proof. Derived from the previous lemmas.

As $F = f_m$ then $\#UNSAT(F) = \mu_m = \alpha_m + \beta_m$. 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 (α_n, β_n) holds that $\#UNSAT(F) = \mu_n = \alpha_n + \beta_n$.

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 $(\alpha_i, \beta_i), i = 1, \dots, n$ can be related to the Fibonacci numbers as follows.

Lemma 8. Let $F = \{C_1, C_2, \dots, C_m\}$ be a monotone positive 2-CF. The recurrence for computing the pair $(\alpha_i, \beta_i), i = 2, 3, \dots, m$ 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 $i - 1$.

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

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

$$\#UNSAT(F) = \sum_{i=1}^n 2^{n-i} \cdot F_{i-1}$$

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

Unfolding the sum until the first values (α_1, β_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: $\#SAT(F) + \#UNSAT(F) = 2^n$ and $\#SAT(F) = F_{n+2}$.

Definition 2. The constrained graph G_F of a 2-CF formula $F = \{C_1, C_2, \dots, C_m\}$ 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.

Lemma 9. Let $F = \{C_1, C_2, \dots, C_m\}$ be a 2-CF where G_F represents a tree. The recurrence which compute (α_p, β_p) where there are j clauses such that $x_p = \bigcap C_j$ is given by:

$$\begin{aligned} \alpha_p &= \alpha_i \cdot 2^{j-1} + \alpha_j \cdot 2^{i-1} - \alpha_i \cdot \alpha_j \\ \beta_p &= \beta_i \cdot 2^{j-1} + \beta_j \cdot 2^{i-1} - \beta_i \cdot \beta_j \end{aligned} \quad (2)$$

Proof. When two different branches, one of size i and the other of size j meet in the same father node, then the two pairs (α_i, β_i) and (α_j, β_j) are associated at the same variable x_p (the father node) and then the charge (α_p, β_p) associated to x_p is updated.

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. $\#2UNSAT(F)$ is computed in polynomial time on the number of nodes in G_F .

Proof. A postorder traversal of the tree computes $\#UNSAT(F)$ while the recurrences (1) and (2) are applied to computes $\#2UNSAT(F)$.

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 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 value of the 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 $2^{n-1} - 1$.*

Proof. If G_F is a minimum dependent tree means that F is of the form $F_n = \{C_1, \dots, C_{n-1}\} = \{(x_1, x_2), (x_1, x_3), \dots, (x_1, x_n)\}$. As all clauses are dependent pairwise, they get for any pair of clauses $C_i, C_j \in F_n$, a less value for $\#UNSAT(C_i \cup C_j)$ (by lemma 2 and 3). The computation of $\#UNSAT(F_n)$ 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 $C_i, i = 1, \dots, n-1$ in an independent way to the other clauses, the charge associated also with x_1 will be $(\alpha_1, \beta_1) = (0, 1)$. As C_1 and C_2 are two branches with the same father (x_1), the subtree has two leaves (x_2 and x_3). According to (2), the new (α_1, β_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 α_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 β_1 for the clause $C_i \in F_n$ can be deduced from the value of β'_1 associated to the clause C_{i-1} in the following way: $\beta_1 = 2 \cdot \beta'_1 + 1$. Then if there are $n-1$ clauses, the final value for β_1 will be $\beta_1 = 2^{n-2} \cdot 1 + 2^{n-3} + \dots + 2^0 = \sum_{i=0}^{n-2} 2^i = 2^{n-1} - 1$.

Thus, the number of falsified 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 $\alpha_1 = 2^{n-1} - 1$. Then, the value $2^{n-1} - 1$ is a lower bound for the value $\#UNSAT(F)$, $n = |v(F)|$.

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

Fig. 2. Counting unsatisfying assignments on a tree

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). 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. As $\# \text{SAT}(F) = 2^n - \# \text{UNSAT}(F)$, then $\# \text{UNSAT}(F)$ will be an upper bound for k .

References

1. Dahllöf, V., Jonsson, P., Wahlström, M.: Counting models for 2SAT and 3SAT formulae. *Theoretical Computer Sciences* 332(1-3), 265–291 (2005)
2. De Ita, G., Marcial-Romero, R., Hernández, J.A.: A threshold for a Polynomial Solution of $\# 2 \text{SAT}$. *Fundamenta Informaticae* 113(1), 63–77 (2011)
3. De Ita, G., Bello, P., Contreras, M.: New Polynomial Classes for $\# 2 \text{SAT}$ Established Via Graph-Topological Structure. *Engineering Letters* 15(2), 250–258 (2007), <http://www.engineeringletters.com>
4. Olivier, D.: Counting the number of solutions for instances of satisfiability. *Theor. Comp. Sc.* 81, 49–64 (1991)
5. Gusfield, D., Pitt, L.: A bounded approximation for the minimum cost 2-Sat problem. *Algorithmica* 8, 103–117 (1992)
6. Linial, N., Nisan, N.: Approximate inclusion-exclusion. *Combinatorica* 10(4), 349–365 (1990)
7. Lozinskii, E.: Counting propositional models. *Inf. Proc. Letters* 41, 327–332 (1992)
8. Roth, D.: On the hardness of approximate reasoning. *Artificial Intelligence* 82, 273–302 (1996)
9. Vadhan Salil, P.: The Complexity of Counting in Sparse, Regular, and Planar Graphs. *SIAM Journal on Computing* 31(2), 398–427 (2001)
10. Wahlström, M.: A Tighter Bound for Counting Max-Weight Solutions to 2SAT Instances. In: Grohe, M., Niedermeier, R. (eds.) *IWPEC 2008*. LNCS, vol. 5018, pp. 202–213. Springer, Heidelberg (2008)
11. Zhou, J., Yin, M., Zhou, C.: New Worts-Case Upper Bound for $\# 2 \text{-SAT}$ and $\# 3 \text{-SAT}$ with the Number of Clauses as the Parameter. In: *Proc. of the AAAI*, pp. 217–222 (2010)