Counting Propositional Models.

Eliezer L. Lozinskii
Institute of Mathematics and Computer Science
The Hebrew University, Jerusalem 91904, Israel

Abstract. Given a propositional DNF or CNF formula F with m clauses on n variables, counting the number of models of F is, at the present state of the art, computationally intractable in a worst case. It is shown that an algorithm based on the Inclusion-Exclusion formula computes the exact number of models of F in average time that under reasonable assumptions is $O(m^c n)$, where $c = O(log\ m)$.

Keywords: Analysis of algorithms, satisfiability, counting models.

1. Introduction.

Consider the following problem: Given a propositional formula F in CNF or DNF, what is the number $\mu(F)$ of its models, that is, assignments of truth values to its variables that satisfy F ? This problem of *counting models* finds numerous applications and attracts attention of many researchers [6, 9]. Counting models of a first-order logic formula G is essential for reasoning with incomplete information [8]. Indeed, if G describes faithfully a real world W, then given a formula ϕ such that neither ϕ nor $\neg\phi$ is a logical consequence of G, a reasonable assumption is that the more models of G assert ϕ the more likely is that ϕ is true in W.

The problem of computing number of models is #P-complete in general [11]. So, to satisfy practical applications one has either to resort to a good approximation of $\mu(F)$, or use an algorithm for computing $\mu(F)$ exactly that is efficient in almost all practical cases, in other words, has a reasonable average time complexity.

Regarding the first alternative, Luby and Velickovic [9] presented a deterministic algorithm for computing number of models of a DNF propositional formula F with m clauses on n variables. Given small numbers $\varepsilon, \delta > 0$, the algorithm computes an estimate Y such that

$$(1-\delta)\ \mu(F) - \varepsilon \ \le\ Y\ \le\ \mu(F) + \varepsilon\ .$$

The running time of the algorithm is estimated by a polynomial in n and m multiplied by

$$n^{O(\frac{1}{\delta}(\log^2 m + \log^2 \frac{1}{\varepsilon})(\log\log m + \log\frac{1}{\varepsilon} + \log\frac{1}{\delta}))}\ .$$

So, the running time of computing the approximation grows fast with its precision.

Linial and Nisan [6] studied the problem of computing the size of a union of a given family of sets $S_1 \cup S_2 \cup \cdots \cup S_m$ by means of the Inclusion-Exclusion formula:

$$|\bigcup_{i=1}^{m} S_i| = \sum_{1\le i\le m} |S_i| - \sum_{1\le i<j\le m} |S_i \cap S_j| + \sum_{1\le i<j<k\le m} |S_i \cap S_j \cap S_k| - \cdots$$

$$+ (-1)^m |S_1 \cap \cdots \cap S_m|\ . \qquad (1)$$

An interesting result of this work (among its other contributions) is that the size of the union can be approximated accurately when the sizes of only a certain part of the set intersections are known. Suppose the intersection sizes are known for all subfamilies containing at most k sets. If $k \ge \Omega(\sqrt{m})$, then the union size can be approximated with a relative error of $O(e^{-2k/\sqrt{m}})$.

As it is shown in [6], this result can be applied to approximating the number of assignments satisfying a DNF formula as the size of the union of assignments satisfying each of its clauses. Thus, only k first terms of the formula (1) are needed for a good approximation, such that $\sqrt{m} \le k \le m$. However, for a high precision k approaches m. For instance, for $m = 100$ and an error of 10^{-3}, $k \approx 34$.

In the sequel we show that for a propositional formula F the size of terms of the Inclusion-Exclusion (1) diminishes to zero so fast that in most practical cases only a small number of set intersections is necessary for computing the *exact* number of models of F, so that under reasonable assumptions, the average running time of computing the exact number of models of a formula with m clauses on n variables is $O(m^c n)$, where $c = O(\log m)$.

2. Algorithm.

Let F be a propositional formula, and $\mu(F)$ denote the number of its models, that is, assignments of truth values to its variables satisfying F. As it has been mentioned in section 1, one of strong motivations for this work is computing the number of models of a deductive system, such as a deductive database, a logic program, an automated reasoning system. These systems are usually presented in CNF as sets of disjunctions and/or implications [1, 7]. With this in mind, and with no loss of generality, we consider in the sequel CNF formulae, such that a formula F is a set of m *clauses* $\{C_1, C_2, \ldots, C_m\}$ over n variables $\{x_1, x_2, \ldots, x_n\}$. A clause C_i is a disjunction of a set of *literals* $\{L_1, L_2, \ldots, L_{k_i}\}$. Any literal L represents one of the variables x_j, such that $L \in \{x_j, \neg x_j\}$. Let A_i be a set of all assignments falsifying C_i. Then

$$\mu(F) = 2^n - \Big|\bigcup_{C_i \in F} A_i\Big| = 2^n + \sum_{k=1}^{m} (-1)^k \nu_k , \tag{2}$$

where

$$\nu_k = \sum_{1 \le j_1 < j_2 < \ldots < j_k \le m} |A_{j_1} \cap A_{j_2} \cap \cdots \cap A_{j_k}|. \tag{3}$$

Let $|C_i|$ denote the *length* of a clause C_i, that is, the number of its literals. Then there are exactly $2^{n - |C_i|}$ assignments falsifying C_i, so,

$$|A_i| = 2^{n - |C_i|} .$$

Let σ be a collection of clauses $\{C_{i_1}, C_{i_2}, \ldots, C_{i_k}\} \subseteq F$. Each clause is a set of literals, hence a union of clauses can be viewed as another clause, so denote

$$U(\sigma) = \bigcup_{C_i \in \sigma} C_i .$$

The intersection $\bigcap_{C_j \in \sigma} A_j$ is a set of assignments such that each one of them falsifies all clauses of σ, and hence falsifies $U(\sigma)$ as well. If $U(\sigma)$ is a tautology containing a pair of opposite literals $\{L, \neg L\}$, then $U(\sigma)$ cannot be falsified, and so, $\bigcap_{C_j \in \sigma} A_j = \varnothing$. This implies the following lemma.

Lemma 1. For a set of clauses $\sigma \subseteq F$,

$$|\bigcap_{C_j \in \sigma} A_j| = \begin{cases} 0 & \textit{if } U(\sigma) \textit{ is a tautology} \\ 2^{n-|U(\sigma)|} & \textit{otherwise.} \end{cases} \quad \blacksquare \tag{4}$$

If $U(\sigma)$ is a tautology, then for all supersets σ' of σ, $U(\sigma')$ is a tautology, and $|\bigcap_{C_j \in \sigma'} A_j| = 0$. So, no superset of σ contributes data to expressions (2), (3). Therefore, if for some t, $v_t = 0$, then $v_k = 0$ for all $k = t+1, \ldots, m$. Hence only $t-1$ first terms of the sum of (2) are needed for computing $\mu(F)$.

Let σ_k denote a collection of exactly k clauses of F. Then (3) can be rewritten as

$$v_k = \sum_{U(\sigma_k) \textit{ is not a tautology}} (2^{n-|U(\sigma_k)|}) . \tag{5}$$

The following algorithm implements the abovementioned observations. A k-th *generation* G_k is a collection of all subsets σ_k of F containing k clauses, such that $U(\sigma_k)$ is not a tautology (call such a σ_k an *active subset*). Let $\sigma_k = \{C_{i_1}, C_{i_2}, \ldots, C_{i_k}\}$ be ordered such that $i_1 < i_2 < \cdots < i_k$. We call i_k the *top* of σ_k (denote $top(\sigma_k) = i_k$). For $j = 1, 2, \ldots, m$, C_j denote the original clauses of F.

Algorithm 1. (Given a propositional CNF formula F with m clauses on n variables, returns the number $\mu(F)$ of assignments satisfying F);

(1) $k := 1$; /* The first generation consists of singleton sets of clauses */
(2) $G_1 := \{ \{C\} \mid C \in F \}$;
(3) $v_1 := \sum_{\{C_i\} \in G_1} 2^{n-|C_i|}$;
(4) $acc := v_1$; /* acc serves as an accumulator of v_k's */
(5) While $G_k \neq \varnothing$ do
    begin /* Compute a new generation */
(6) $k := k + 1$;
(7) $G_k := \varnothing$;
(8) $v_k := 0$;
(9)   For all $\sigma_{k-1} \in G_{k-1}$ do
(10)   For j from $top(\sigma_{k-1}) + 1$ to m do
             begin /* Extend subsets */
(11) $\sigma_k := \sigma_{k-1} \cup \{C_j\}$;
(12)   If $U(\sigma_k)$ is not a tautology then
                 begin /* Include σ_k in G_k */
(13) $G_k := G_k \cup \{\sigma_k\}$;
(14) $v_k := v_k + 2^{n-|U(\sigma_k)|}$;
                 end
             end
(15) $acc := acc + (-1)^{k-1} v_k$;
    end
(16) $\mu(F) := 2^n - acc$. $\blacksquare$

To estimate the complexity of Algorithm 1 we note (see loop 5) that for every non-empty generation G_k, loop 9 is run $|G_k|$ times. Within loop 9, loop 10 is run at most $m-1$ times, and its running time is dominated by that of line 12, where checking tautology of $U(\sigma_k)$ is $O(n)$. So, the running time T of Algorithm 1 is

$$T = O(Nmn), \tag{6}$$

where N is the total number of active subsets of F in all generations produced by the algorithm.

In a worst case $N = 2^m$, that accounts for the high complexity of the counting problem [11]. However, since only active sets of clauses take part in the generating process, in most cases the number of sets constituting k-th generation G_k diminishes very fast with k (as shown in the next section) decreasing significantly the average running time of the algorithm.

3. Average running time.

Let F be a set of m clauses of length l on n variables. Assume that the variables and literals are uniformly distributed in the clauses, such that any variable appears in a clause C with the same probability $\frac{l}{n}$, and if x appears in C , then literals $x, \neg x$ are equiprobable. Let p_k denote the probability that a set of k clauses of F is an active one.

Lemma 2. Any set of k clauses $\sigma_k \subseteq F$ is active with the probability

$$p_k \le \left(1 - \frac{l^2}{2n^2}\right)^{\frac{n}{2}k(k-1)}.$$

Proof. By induction on k .

Base. Each original clause of F is not a tautology with probability $p_1 \le 1$, so the Lemma holds for $k = 1$.

Let us compute p_2 for a set $S = \{C, C'\}$ of any two clauses of F . Let X, Y denote sets of variables appearing in C, C' respectively, such that $X = \{x_i \mid i = 1, \dots, l\}$, $Y = \{y_i \mid i = 1, \dots, l\}$, and $v(L)$ be the variable appearing in the literal L (*represented* by L). If there are literals $L \in C$ and $L' \in C'$ such that $v(L) = v(L') = y$ and $L = \neg L'$, then we say that y *causes tautology* of S making S inactive. Let $q(x)$ denote the probability that a variable x causes tautology of S which means that x appears in both clauses C and C' (with the probability $\frac{l^2}{n^2}$) and is represented by literals L in C and L' in C' such that $L = \neg L'$ (the latter happens with probability 0.5). Hence, $q(x) = \frac{l^2}{2n^2}$. The set S is active iff none of the n variables $x_1, x_2, \dots, x_n$ appearing in F causes tautology of S , and none of C, C' is a tautology. Hence,

$$p_2 \le \prod_{i=1}^{n}(1 - q(x_i)) = \left(1 - \frac{l^2}{2n^2}\right)^n.$$

So, the Lemma holds for $k = 2$ as well.

Induction step. Suppose that the Lemma holds for some k , such that $2 \le k < m$, and consider a subset σ_{k+1} of F containing $k+1$ clauses, $\sigma_{k+1} = \sigma_k \cup \{C'\}$. The set σ_{k+1} is an active one iff σ_k and $\{C, C'\}$ are active for all $C \in \sigma_k$. Hence

$$p_{k+1} = p_k \cdot {p_2}^k \le \left(1 - \frac{l^2}{2n^2}\right)^{\frac{n}{2}k(k-1)+nk} = \left(1 - \frac{l^2}{2n^2}\right)^{\frac{n}{2}k(k+1)}.$$

Thus, the Lemma holds for all k . ■

Let $\bar{N}_k, \bar{N}$ denote, respectively, the expected number of active sets of clauses constituting a k-th generation G_k , and the expected total number of active sets produced by Algorithm 1. Then

$$\bar{N}_k \le \binom{m}{k}\left(1 - \frac{l^2}{2n^2}\right)^{\frac{n}{2}k(k-1)}, \qquad \bar{N} = \sum_{k=1}^{m} \bar{N}_k ,$$

and the expected running time of Algorithm 1 is (by (6))

$$\bar{T} = O(\bar{N}mn). \tag{7}$$

Let us consider the proportion

$$\alpha_k = \frac{\bar{N}_{k+1}}{\bar{N}_k} = \frac{m-k}{k+1}\left(1 - \frac{l^2}{2n^2}\right)^{nk}. \tag{8}$$

To estimate $\bar{N}$ let us present it in the form

$$\bar{N} = \sum_{i=1}^{b} \bar{N}_i + \sum_{i=b+1}^{m} \bar{N}_i, \tag{9}$$

and find an integer b such that

$$\alpha_k \le 1 \quad \text{for all} \quad b \le k \le m. \tag{10}$$

Since α_k is a monotonously decreasing function of k (cf. (8)), the condition (10) is satisfied by $\alpha_b = 1$, that is:

$$\frac{m-b}{b+1}\left(1 - \frac{l^2}{2n^2}\right)^{nb} = 1;$$

$$\ln\frac{m-b}{b+1} + nb \cdot \ln\left(1 - \frac{l^2}{2n^2}\right) = 0;$$

$$b = \frac{\ln\dfrac{m-b}{b+1}}{n \cdot \ln\dfrac{2n^2}{2n^2 - l^2}}. \tag{11}$$

Since $b \ge 1$ and $\ln\dfrac{2n^2}{2n^2-l^2} = \ln\left(1 + \dfrac{l^2}{2n^2-l^2}\right) > \dfrac{l^2}{2n^2-l^2} - \dfrac{l^4}{2\,(2n^2-l^2)^2}$, we get from (11)

$$b < \frac{\ln m}{n}\left(\frac{2n^2}{l^2} - \frac{3}{8}\right). \tag{12}$$

Now (9) and (10) yield

$$\sum_{i=b+1}^{m} \bar{N}_i \le (m-b)\,\bar{N}_b; \qquad \bar{N} \le \sum_{i=1}^{b} \bar{N}_i + (m-b)\,\bar{N}_b = O\,(\,m^{b+1}\,). \tag{13}$$

(7), (12) and (13) imply the following theorem.

Theorem 1. Given a CNF propositional formula F with m clauses of length l on n variables, and with uniform distribution of variables and literals among the clauses, the average time of computing the exact number of models of F by Algorithm 1 is

$$\bar{T} = O(m^{b+2} \cdot n)\,, \quad \text{where } b < \frac{\ln m}{n}\left(\frac{2n^2}{l^2} - \frac{3}{8}\right). \quad \blacksquare$$

This result as well as Algorithm 1 (the latter with a slightest modification) applies also to DNF formulae.

4. Non-uniform distributions.

A uniform probability distribution of variables and literals among clauses is commonly assumed for a probabilistic analysis of algorithms handling CNF or DNF formulae [2-5, 10].

However, in real cases distribution of variables and literals among the clauses of a formula F may not be uniform, so it is worthwhile to find out to what extent this affects the estimation of the average complexity of Algorithm 1 presented in Section 3.

Let p_k denote the probability that any set of k clauses, $\sigma_k \subseteq F$, is an active one. Since σ_k is active iff every its subset consisting of two clauses is active, we get

$$p_k = p_2^{\frac{k(k-1)}{2}}.$$

So, the larger is p_2, the more subsets of F must be examined by Algorithm 1 (Section 2) slowing it down. In the extreme, if $p_2 = 1$ then every subset of F is active, and so $\bar{N}$ reaches its maximum value of 2^m. Such a worst distribution corresponds, in particular, to the case where all literals of F are pure.

Another reason for a large p_2 can be a certain correlation between occurrences of literals in the clauses of F or non-uniformity of their distribution. In the latter case, let $r(y)$ denote the probability that a variable y occurs in a clause C, and if this happens, then literals $y, \neg y$ appear in C with probability $\rho(y), \rho(\neg y)$, respectively. Consider clauses C, C' of F. A variable y causes tautology of $\{C, C'\}$ (with probability $q(y)$) iff y occurs in both clauses, but is represented in them by opposite literals, so

$$q(y) = 2\,r(y)^2\,\rho(y)\,\rho(\neg y).$$

Hence any set of two clauses of F is active with probability

$$p_2 = \prod_{i=1}^{n}(1 - q(y_i)).$$

Following the line of Section 3, we get:

$$\bar{N}_k = \binom{m}{k} p_2^{\frac{k(k-1)}{2}} ;$$

$$\alpha_k = \frac{\bar{N}_{k+1}}{\bar{N}_k} = \frac{m-k}{k+1} p_2^k \leq 1 \quad \text{for all} \quad b \leq k \leq m ;$$

$$\alpha_b = \frac{m-b}{b+1} p_2^b = 1 ;$$

$$b = \frac{\log \dfrac{m-b}{b+1}}{\log(1 / p_2)} < \frac{\log m}{\log(1 / p_2)} . \tag{14}$$

Thus, for any distribution of parameters determining the structure of a formula F, the following holds.

Theorem 2. Given a CNF propositional formula F with m clauses on n variables, let p_2 be the probability that union of any two clauses of F is not a tautology. Then the average time of computing the exact number of models of F by Algorithm 1 is $O(m^{b+2}n)$, where $b < \log m / \log(1 / p_2)$. ■

5. Experiments.

Sections 3, 4 provide upper bounds for the average running time of Algorithm 1. To assess its actual performance, Algorithm 1 has been programmed in *C* and run on a *GOULD POWER NODE* computer under *UNIX (UTX 32.2.1a)*. Formulae F have been generated randomly as sets of m clauses each containing l literals over n variables. For each clause, l variables have been chosen equiprobably, and if x has been chosen, then the clause contained x or $\neg x$ with the same conditional probability 0.5.

Parameters of F varied in the following ranges: $10 \leq n \leq 30$, $10 \leq m \leq 100$, $3 \leq l \leq 27$. For each combination of the parameters 100 random instances of F have been processed, and numerous measurements performed. In particular, the following has been measured over all 100 instances of a given tuple (n, m, l): the average number $\bar{N}$ of subsets of a formula F in all generations produced for F by Algorithm 1, and the maximum number g_{max} of generations. The experimental data for $\bar{N}$ were approximated by polynomials: $\bar{N} = O(m^4)$ for fixed n and l, $\bar{N} = O(n^4)$ for fixed m and l. The longer are the clauses, l, the faster runs the algorithm.

Acknowledgments.

Many thanks to Nathan Linial and Michael Ben-Or for insightful discussions and bringing the paper by Luby and Velickovic [9] to my attention. I am grateful to the anonymous referee for most apt and stimulating comments.

References.

1. *Advances in Data Base Theory,* H. Gallaire, J. Minker and J. M. Nicolas (Eds.), Plenum Press, v.1, 1981, v.2, 1984.
2. J. Franco, On the probabilistic performance of algorithms for the satisfiability problem. *Information Processing Letters,* v.23, 1986, 103-106.
3. J. Franco and M. Paull, Probabilistic analysis of the Davis-Putnam procedure for solving the satisfiability problem. *Discrete Appl. Math.,* v.5, 1983, 77-87.
4. A. Goldberg, Average case complexity of the satisfiability problem. *Proc. 4th Workshop on Automated Deduction,* Austin, Texas, 1979, 1-6.
5. A. Goldberg, P. Purdom, and C. Brown, Average time analysis of simplified Davis-Putnam procedures. *Information Processing Letters,* v.15, no.2, 1982, 72-75.
6. N. Linial and N. Nisan, Approximate inclusion-exclusion. *Combinatorica,* v.10, no.4, 1990, 349-365.
7. J. Lloyd, *Foundations of Logic Programming.* Springer-Verlag, 1984.
8. E. L. Lozinskii, Answering atomic queries in indefinite deductive databases. *Intl. J. of Intelligent Systems,* v.4, no.4, 1989, 403-430.
9. M. Luby and B. Velickovic, On deterministic approximation of DNF. *Proceedings of STOC'91,* New Orleans, May 1991.
10. P. Purdom and C. Brown, The pure literal rule and polynomial average time. *SIAM J. Comput.,* v.14, 1985, 943-953.
11. L. G. Valiant, The complexity of computing the permanent. *Theoretical Computer Science,* v.8, 1979, 189-201.