A Linear Time Algorithm for Counting #2SAT on Series-Parallel Formulas

Marco A. López-Medina[1], J. Raymundo Marcial-Romero[1](✉), Guillermo De Ita-Luna[2], and José A. Hernández[1]

[1] Facultad de Ingeniería, UAEMex, Toluca, Mexico
mlopezm158@alumno.uaemex.mx, jrmarcialr@uaemex.mx
[2] Facultad de Ciencias de la Computación, BUAP, Puebla, Mexico
deita@cs.buap.mx

Abstract. An $O(m + n)$ time algorithm is presented for counting the number of models of a two Conjunctive Normal Form Boolean Formula whose constrained graph is represented by a Series-Parallel graph, where n is the number of variables and m is the number of clauses. To the best of our knowledge, no linear time algorithm has been developed for counting in this kind of formulas.

Keywords: #SAT · #2SAT · Complexity theory · Graph theory

1 Introduction

The decision problem $SAT(F)$, where F is a Boolean formula, consists in determining whether F has a model, that is, an assignment to the variables of F such that when evaluated with respect to classical Boolean logic it returns true as a result. If F is in two Conjunctive Normal Form (2-CNF) then $SAT(F)$ can be solved in polynomial time, however if F is in k-CNF, $k > 2$, then $SAT(F)$ is an NP-Complete problem. The counting version consists on determining the number of models of F denoted as $\#SAT(F)$. $\#SAT(F)$ belongs to class #P-Complete even when F is in 2-CNF, the latter denoted as $\#2SAT$ [1].

Although the $\#2SAT$ problem is #P-Complete, there are instances that can be solved in polynomial time [2,3]. For example, if the graph representation of the formula is acyclic, then the number of models can be computed in lineal time. Currently, the algorithms that are used to solve the problem for any formula F in 2-CNF, decompose F into sub-formulas until there are base cases in which it can be counted efficiently. The algorithm with the best time complexity so far was developed by Wahlström [4] which is given by $O(1.2377^n)$ where n represents the number of variables in the formula. The Wahlström algorithm uses the number of times a variable appears in the formula (be it the variable or its negation) as the criterion for choosing it. The two criteria for stopping the algorithm are when $F = \emptyset$ or when $\emptyset \in F$.

On series-parallel graphs the closest work is related to recognizing when a graph is series-parallel and some decision problem as we briefly describe.

© Springer Nature Switzerland AG 2020
L. Martínez-Villaseñor et al. (Eds.): MICAI 2020, LNAI 12468, pp. 437–447, 2020.
https://doi.org/10.1007/978-3-030-60884-2_33

Schoenmakers [5] develops a linear-time algorithm to recognize series-parallel graphs, computing a *source-sink* representation of this graph from a breath-first spanning tree. The complexity of constructing this representation is $O(n\sqrt{n})$. Eppstein [6] gives an algorithm to recognize directed and undirected series-parallel graphs, based on a characterization of their ear decompositions. Takamizawa [7] shows that if a graph is restricted to the series-parallel class, then there exists linear-time algorithms for decision problems and combinatorial problems as: minimum vertex cover, maximum independent vertex set, maximum (induced) line-subgraph, maximum edge (vertex) deletion respect to property "without cycles (or paths) of specified length n or any length $\leq n$", maximum outerplanar (induced) subgraph, minimum feedback vertex set, maximum ladder (induced) subgraph, minimum path cover, maximum matching and maximum disjoint triangle.

In this paper we present an algorithm to count model on a special class of formulas in linear time, the so called, series-parallel formulas [8,9] which to the best of our knowledge has not been tackle previously to count models [10].

2 Preliminaries

2.1 Conjunctive Normal Form

Let $X = \{x_1, ..., x_n\}$ be a set of n Boolean variables (that is, they can only take two possible values 1 or 0). A literal is a variable x_i, denoted in this paper as x_i^1 or the denied variable $\neg x_i$ denoted in this paper as x_i^0. A clause is a disjunction of different literals. A Boolean formula F in conjunctive normal form is a conjunction of clauses.

Let $v(Y)$ be the set of variables involved in the object Y, where Y can be a literal, a clause or a Boolean formula. For example, for the clause $c = \{x_1^1 \vee x_2^0\}$, $v(c) = \{x_1, x_2\}$.

An assignment s in F is a Boolean function $s : v(F) \to \{0, 1\}$. s is defined as:

$$s(x^0) = 1 \text{ if } s(x^1) = 0, \text{ otherwise }, s(x^0) = 0.$$

The assignment can be extended to conjunctions and disjuntions as follows:

- $s(x \wedge y) = 1$ if $s(x) = s(y) = 1$, otherwise, $s(x \wedge y) = 0$
- $s(x \vee y) = 0$ if $s(x) = s(y) = 0$, otherwise, $s(x \vee y) = 1$

Let F be a Boolean formula in CNF, it is said that s satisfies F, if for each clause c in F, it holds $s(c) = 1$. On the other hand, it is said that F is contradicted by s ($s \not\models F$), if there is at least one clause c of F such that $s(c = 0)$. Thus a model of F is an assignment that satisfies F.

Given a formula F in CNF, SAT is to determine whether F has a model, while $\#SAT$ is to count the number of models that F. On the other hand, $\#2SAT$ denotes $\#SAT$ for formulas in 2-CNF.

2.2 The Restricted Graph of a 2-CNF

There are some graphical representations of a Conjunctive Normal Form, in this case the signed primary graph (restricted graph) [11] will be used.

Let F be a 2-CNF, its restricted graph is denoted by $G_F = (V(F), E(F))$ where the vertices of the graph are the variables $V(F) = v(F)$ and $E(F) = \{\{x_i^\epsilon, x_j^\gamma\} \mid \{x_i^\epsilon \vee x_j^\gamma\} \in F\}$, that is, for each clause $\{x_i^\epsilon \vee x_j^\gamma\} \in F$ there is an edge $\{x_i^\epsilon, x_j^\gamma\} \in E(F)$. For $x \in V(F)$, $\delta(x)$ denotes its degree, that is the number of incident edges in x. Each edge $c = \{x_i^\epsilon, x_j^\gamma\} \in E(F)$ has associated a pair (ϵ, γ), which represent whether the variables x_i or x_j appear negated or not. For example, the clause $(x_1^0 \vee x_2^1)$ has associated the pair (0, 1) meaning that in the clause, x_i appears negated and x_2 not.

2.3 Methods Already Reported to Count in #2SAT

The basic idea considered in related papers to count models on a restricted graph G consists on computing a tuple (α_i, β_i) over each vertex $x_i^{\epsilon_i}$ where α_i represents the number of times that $x_i^{\epsilon_i}$ appears positive in the models of G and β_i the number of times $x_i^{\epsilon_i}$ appears negative in the models of G. For example a clause with a simple vertex $\{x_i^{\epsilon_i}\}$ has associated the tuple $(1, 1)$ Given and edge(clause) $e = \{x_i^{\epsilon_i}, x_j^{\gamma_i}\}$ if the counting begins at $x_i^{\epsilon_i}$ the tuples $(1, 1)$, $(2, 1)$ are associated to $x_i^{\epsilon_i}$ and $x_j^{\gamma_i}$ respectively however if the counting begins at $x_j^{\gamma_i}$ the tuples are associated inversely. The models are the sum of the last two elements of the tuple.

There are reported methods to count models in some graphical representations of a 2-CNF formula F [12], here we stated the methods needed in the paper:

- If the graph represents a path e.g. a formula of the form $P_n = \{\{x_1^{\epsilon_1}, x_2^{\gamma_1}\}, \{x_2^{\epsilon_2}, x_3^{\gamma_2}\}, \cdots \{x_{n-1}^{\epsilon_{n-1}}, x_n^{\gamma_{n+1}}\}\}$ of n vertices, the number of models is given by the sum of the elements of the pair (α_n, β_n). where $(\alpha_1, \beta_1) = (1, 1)$ and the tuple for the other vertices is computed according to the next recurrence.

$$(\alpha_i, \beta_i) = \begin{cases} (\alpha_{i-1} + \beta_{i-1}, \alpha_{i-1}) \ if \ (\epsilon_{i-1}, \gamma_{i-1}) = (1, 1) \\ (\alpha_{i-1}, \alpha_{i-1} + \beta_{i-1}) \ if \ (\epsilon_{i-1}, \gamma_{i-1}) = (1, 0) \\ (\alpha_{i-1} + \beta_{i-1}, \beta_{i-1}) \ if \ (\epsilon_{i-1}, \gamma_{i-1}) = (0, 1) \\ (\beta_{i-1}, \alpha_{i-1} + \beta_{i-1}) \ if \ (\epsilon_{i-1}, \gamma_{i-1}) = (0, 0) \end{cases} \quad (1)$$

- Let $\{x_i^{\epsilon_1}, x_j^{\gamma_1}\}$ and $\{x_i^{\epsilon_2}, x_j^{\gamma_2}\}$ be two parallel clauses in a formula F, the number of models for this clauses is given by:

$$(\alpha_j, \beta_j) = \begin{cases} (\alpha_i, \alpha_i) \quad if \ (\epsilon_1, \gamma_1) = (1, 1) \ and \ (\epsilon_2, \gamma_2) = (1, 0) \\ (\alpha_i + \beta_i, 0) \ if \ (\epsilon_1, \gamma_1) = (1, 1) \ and \ (\epsilon_2, \gamma_2) = (0, 1) \\ (\beta_i, \alpha_i) \quad if \ (\epsilon_1, \gamma_1) = (1, 1) \ and \ (\epsilon_2, \gamma_2) = (0, 0) \\ (\alpha_i, \beta_i) \quad if \ (\epsilon_1, \gamma_1) = (1, 0) \ and \ (\epsilon_2, \gamma_2) = (0, 1) \\ (0, \alpha_i + \beta_i) \ if \ (\epsilon_1, \gamma_1) = (1, 0) \ and \ (\epsilon_2, \gamma_2) = (0, 0) \\ (\beta_i, \beta_i) \quad if \ (\epsilon_1, \gamma_1) = (0, 1) \ and \ (\epsilon_2, \gamma_2) = (0, 0) \end{cases} \quad (2)$$

– Let $\{x_i^{\epsilon_1}, x_j^{\gamma_1}\}$, $\{x_i^{\epsilon_2}, x_j^{\gamma_2}\}$ and $\{x_i^{\epsilon_3}, x_j^{\gamma_3}\}$ be three parallel clauses in a formula F, the number of models for this clauses is given by:

$$(\alpha_j, \beta_j) = \begin{cases} (0, \alpha_i) & if\ (\epsilon_1, \gamma_1) = (1,1)\ and\ (\epsilon_2, \gamma_2) = (1,0)\ and\ (\epsilon_3, \gamma_3) = (0,0) \\ (\alpha_i + \beta_i, 0) & if\ (\epsilon_1, \gamma_1) = (1,1)\ and\ (\epsilon_2, \gamma_2) = (1,0)\ and\ (\epsilon_3, \gamma_3) = (0,1) \\ (\beta_i, \alpha_i) & if\ (\epsilon_1, \gamma_1) = (1,1)\ and\ (\epsilon_2, \gamma_2) = (0,1)\ and\ (\epsilon_3, \gamma_3) = (0,0) \\ (\alpha_i, \beta_i) & if\ (\epsilon_1, \gamma_1) = (0,1)\ and\ (\epsilon_2, \gamma_2) = (1,0)\ and\ (\epsilon_3, \gamma_3) = (0,0) \end{cases} \quad (3)$$

3 Counting Separately on Series and Parallel Formulas

In this paper instead of associating a tuple to each vertex, as previously explained, we associate a triple to each edge as described below.

3.1 Directional Element

For a clause $e = \{x_i^{\epsilon_1}, x_j^{\gamma_1}\}$, a triple $Q_e = (x_i, x_j, C_{x_i x_j})$ is associated, where the first two elements are the variables associated to the literals, $C_{x_i x_j}$ is a quadruple which represents the models for the possible assignments of the literals x_i and x_j. Initially, the value of the quadruple for $C_{x_i x_j}$ has three non-zero elements and one zero element which represents the three models that a clause has associated, as represented on Eq. 4.

$$C_{x_i x_j} = \begin{cases} (1,1,1,0) & if\ \ (\epsilon_1, \gamma_1) = (1,1) \\ (1,0,1,1) & if\ \ (\epsilon_1, \gamma_1) = (1,0) \\ (1,1,0,1) & if\ \ (\epsilon_1, \gamma_1) = (0,1) \\ (0,1,1,1) & if\ \ (\epsilon_1, \gamma_1) = (0,0) \end{cases} \quad (4)$$

Each $C_{x_i x_j}$ is called a directional element. For example for a clause $\{x_i^1, x_j^0\}$, $C_{x_i x_j} = (1,0,1,1)$.

3.2 Extended the Counting

Now we present counting methods on clauses of two kinds, paths, which later on will represent series graphs and parallel clauses. Our method consists on contracting edges until a single edge with two vertices is left.

Series counting

Let $e_1 = \{x_i^{\epsilon_i}, x_j^{\gamma_i}\}$ and $e_2 = \{x_j^{\epsilon_j}, x_k^{\gamma_j}\}$ be two clauses such that $i \neq k \neq j$ and whose triples are $Q_{e_1} = \{x_i, x_j, C_{x_i x_j}\}$ and $Q_{e_2} = \{x_j, x_k, C_{x_j x_k}\}$. As can be noticed, these two clauses form a path, since they are joined by x_j. We *contract* e_1 and e_2 into a single clause, lets call it e_{1-2}, its new triple is given by $Q_{e_{1-2}} = \{x_i, x_k, C_{x_i x_k}\}$ where each element of $C_{x_i x_k}$ is computed as:

$$\pi_1(C_{x_i x_k}) = \pi_1(C_{x_j x_k})\pi_1(C_{x_i x_j}) + \pi_2(C_{x_j x_k})\pi_3(C_{x_i x_j}) \quad (5)$$

$$\pi_2(C_{x_i x_k}) = \pi_1(C_{x_j x_k})\pi_2(C_{x_i x_j}) + \pi_2(C_{x_j x_k})\pi_4(C_{x_i x_j}) \tag{6}$$

$$\pi_3(C_{x_i x_k}) = \pi_3(C_{x_j x_k})\pi_1(C_{x_i x_j}) + \pi_4(C_{x_j x_k})\pi_3(C_{x_i x_j}) \tag{7}$$

$$\pi_4(C_{x_i x_k}) = \pi_4(C_{x_j x_k})\pi_2(C_{x_i x_j}) + \pi_4(C_{x_j x_k})\pi_4(C_{x_i x_j}) \tag{8}$$

where $\pi_l(C_{x_i x_j})$ is the projection function on l-th element of the quadruple.

Lemma 1. *Let F be a formula representing a restricted path graph $P_n = \{\{x_1^{\epsilon_1}, x_2^{\gamma_1}\}, \{x_2^{\epsilon_2}, x_3^{\gamma_2}\} \cdots \{x_{n-1}^{\epsilon_{n-1}}, x_n^{\gamma_{n-1}}\}\}$, if the contraction rule is applied from $\{x_1^{\epsilon_1}, x_2^{\gamma_1}\}$ to $\{x_{n-1}^{\epsilon_{n-1}}, x_n^{\gamma_{n-1}}\}$ until a triple $(x_1, x_n, C_{x_1 x_n})$ is obtained then*

$$\#2SAT(F) = \pi_1(C_{x_1 x_n}) + \pi_2(C_{x_1 x_n}) + \pi_3(C_{x_1 x_n}) + \pi_4(C_{x_1 x_n})$$

Proof. By induction comparing it with the well known result of Eq. 1. The base case when there is a simple clause $\{x_1^{\epsilon}, x_2^{\gamma}\}$, the number of models corresponds with that computed with Eq. 1 as shown in Figs. 1 and 2. In both cases the sum of their tuples is three.

Fig. 1. Path P_2 using Eq. 1

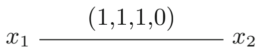

Fig. 2. Path P_2 using serial counting

On a path P_3, with edges $\{\{x_1^{\epsilon_i}, x_2^{\gamma_i}\}, \{x_2^{\epsilon_j}, x_3^{\gamma_j}\}\}$ the number of models is either 5 if $(\gamma_1 = \epsilon_2)$ or 4 if $(\gamma_1 \neq \epsilon_2)$ (Fig. 3).

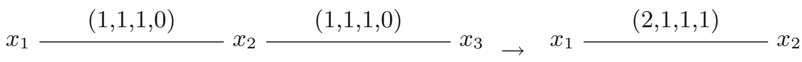

Fig. 3. Path P_2 using serial counting and contraction on x_1

The inductive step is a two case analysis over the values of ϵ and γ.

Example, let $F = \{\{x_1^1, x_2^1\}, \{x_2^1, x_3^0\}, \{x_3^0, x_4^1\}, \{x_4^0, x_5^0\}\}$ be a formula whose restricted graph represents a path. The triples for each clause are: $\{x_1, x_2, (1,1,1,0)\}, \{x_2, x_3, (1,0,1,1)\}, \{x_3, x_4, (1,1,0,1)\}, \{x_4, x_5, (0,1,1,1)\}$.

To make the notation more amenable we use the following representation for the triples.

$$x_1 \xrightarrow{\quad(1,1,1,0)\quad} x_2 \xrightarrow{\quad(1,0,1,1)\quad} x_3 \xrightarrow{\quad(1,1,0,1)\quad} x_4 \xrightarrow{\quad(0,1,1,1)\quad} x_5$$

First we need to contract $\{x_1^1, x_2^1\}$ and $\{x_2, x_3^0\}$ to create a new edge from x_1 to x_3. Using the serial composition equations (6–9) we obtain the new quadruple.

$$x_1 \xrightarrow{\quad(1,1,2,1)\quad} x_3 \xrightarrow{\quad(1,1,0,1)\quad} x_4 \xrightarrow{\quad(0,1,1,1)\quad} x_5$$

Now contracting edges on x_3 we get:

$$x_1 \xrightarrow{\quad(3,2,2,1)\quad} x_4 \xrightarrow{\quad(0,1,1,1)\quad} x_5$$

And finally contracting edges on x_4 we get:

$$x_1 \xrightarrow{\quad(2,1,5,3)\quad} x_5$$

The number of models of the initial formula is obtained by adding the four elements of the quadruple, in this case is 11.

Parallel Counting

Given a clause $\{x_i^{\epsilon_i}, x_j^{\gamma_i}\}$, there are at most four possible parallel clauses of it, including itself: $\{x_i^{\epsilon_i}, x_j^{\gamma_i}\}$, $\{x_i^{\epsilon_i}, x_j^{1-\gamma_i}\}$, $\{x_i^{1-\epsilon_i}, x_j^{\gamma_i}\}$ and the clause $\{x_i^{1-\epsilon_i}, x_j^{1-\gamma_i}\}$. In a formula, at most three of them can be present, otherwise the formula does not have models. In fact, Eqs. 2 and 3 present already known methods for counting models in those classes of parallel clauses.

In this section we extend the contraction method for parallel clauses which again works on clauses instead of vertices. In our method since a clause is represented by a triple $(x_i, x_j, C_{x_i x_j})$, there may be a finite number of parallel clauses of the previous form with different $C_{x_i x_j}$ components, these clauses may come from a serial reduction in a serial-parallel formula.

Let $e_1 \ldots e_h$ be parallel clauses on two vertices with triples $Q_{e_l} = (x_i, x_j, C_{x_i y_j}) : 1 \leq l \leq h$. A *merge* of them can be done leaving the result lets say in e_1. The *merge* is accomplished with the following equations:

$$\pi_k(C_{x_i x_j}) = \prod_{l=0}^{h} \pi_k(C_{x_i x_j}) \tag{9}$$

where $k = 1, 2, 3, 4$.

Lemma 2. *Let F be a formula representing a restricted graph of two or three parallel clauses e.g. $F = \{\{x_1^{\epsilon_1}, x_2^{\gamma_1}\}, \{x_1^{\epsilon_2}, x_2^{\gamma_2}\}\}$ or the formula is $F = \{\{x_1^{\epsilon_1}, x_2^{\gamma_1}\}, \{x_1^{\epsilon_2}, x_2^{\gamma_2}\}, \{x_1^{\epsilon_3}, x_2^{\gamma_3}\}\}$, if the merging rule is applied until a single triple is left $(x_1, x_2, C_{x_1x_2}\}$ then*

$$\#2SAT(F) = \pi_1(C_{x_1x_2}) + \pi_2(C_{x_1x_2}) + \pi_3(C_{x_1x_2}) + \pi_4(C_{x_1x_2})$$

Proof. A simple case analysis is done comparing the results with the well known results of Eqs. 2 and 3. The result for two parallel clauses is shown in Fig. 4 and their merge in Fig. 5.

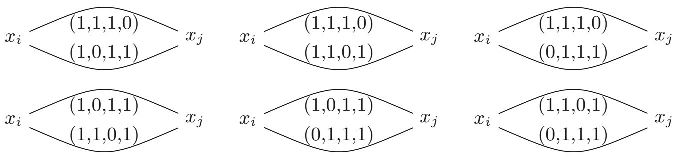

Fig. 4. Parallel cases of two clauses

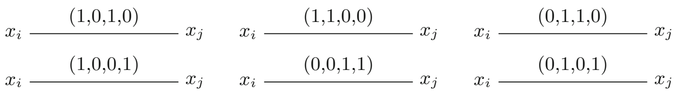

Fig. 5. Applying merging rule on cases of two clauses

From Eq. 2 the pair (α_i, β_i) obtained on cases of two parallel clauses $(x_i^{\epsilon_1}, x_j^{\gamma_1})$ and $(x_i^{\epsilon_2}, x_j^{\gamma_2})$ are $(2, 0)$ if $(\epsilon_1, \gamma_1) = (1, 1)$ and $(\epsilon_2, \gamma_2) = (0, 1)$, it is $(0, 2)$ if $(\epsilon_1, \gamma_1) = (1, 0) and (\epsilon_2, \gamma_2) = (0, 0)$, and $(1, 1)$ on the remaining cases. Fot three clauses the analysis is similar.

For example given parallel clauses $e_1 = \{x_i^1, x_j^1\}$, $e_2 = \{x_i^1, x_j^0\}$, with elements $C_{x_ix_j} = (1, 1, 1, 0)$ of e_1 and $C_{x_ix_j} = (1, 0, 1, 1)$ of e_2. They can be merged obtaining the updated element $C_{x_ix_j} = (1, 0, 1, 0)$ as graphically show in Figs. 6 and 7.

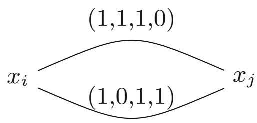

Fig. 6. Graphical representation of the quadruples for e_1, e_2

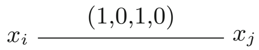

Fig. 7. Merging e_1 and e_2

4 Counting on Series-Parallel Formulas

A graph represents a Series-Parallel formula if it can be built from a single edge and the following two operations:

1. Series construction: subdividing an edge in the graph.
2. Parallel construction: duplicating an edge in the graph.

Another characterization of a series-parallel graph is that it do not contain a subdivision of K_4 (complete graph of four vertices). The first characterization implies that a series-parallel graph always has a vertex of degree two and further more given a series-parallel graph we can always deconstruct it using the inverse of the previous described operations. This section presents the algorithm used for counting models on Series-Parallel graphs. Algorithm 1 computes the number of models of a series-parallel formula, it consists of two main steps: serial deconstruction (contraction) and parallel deconstruction (merging).

Algorithm 1. Procedure that computes $\#2SAT(F)$ when F represents a series-parallel graph

procedure #2SAT(F)
Compute the incident matrix of F to get the degree of each vertex
while $|F > 1|$\{There are more than one clause\} **do**
  for each vertex x_i of degree 2 **do**
    Apply the series contraction rule to the two clauses x_i belongs to
  end for
  for each pair of parallel clause **do**
    Apply the parallel merging rule to those clauses
  end for
end while
Let $e = \{x_i, x_j\}$ be the left clause and its triple $Q_e = (x_i, x_j, C_{x_i x_j})$
return $\pi_1(C_{x_i x_j}) + \pi_2(C_{x_i x_j}) + \pi_3(C_{x_i x_j}) + \pi_4(C_{x_i x_j})$.

Theorem 1. *If F is a series-parallel formula then Algorithm 1 computes $\#2SAT(F)$.*

Proof. A consequence of Lemmas 1 and 2.

4.1 Running Example

We now present an example in order to explain our algorithm, let

$$F = \{\{x_1^1, x_2^1\}, \{x_2^1, x_3^1\}, \{x_3^0, x_{10}^0\}, \{x_1^1, x_4^1\}, \{x_4^1, x_5^0\}, \{x_5^0, x_{10}^1\},\\ \{x_1^1, x_6^1\}, \{x_6^0, x_7^1\}, \{x_7^1, x_{10}^0\}, \{x_1^1, x_8^1\}, \{x_8^0, x_9^0\}, \{x_9^1, x_{10}^1\}\}$$

be a formula whose restricted graph represents a series-parallel graph as shown in Fig. 8. The triples for each clause are:

$$x_1, x_2, (1,1,1,0), x_2, x_3, (1,1,1,0), x_3, x_{10}, (0,1,1,1), x_1, x_4, (1,1,1,0),\\ x_4, x_5, (1,0,1,1), x_5, x_{10}, (1,1,0,1), x_1, x_6, (1,1,1,0), x_6, x_7, (1,1,0,1),\\ x_7, x_{10}, (1,0,1,1), x_1, x_8, (1,1,1,0), x_8, x_9, (0,1,1,1), x_9, x_{10}, (1,1,1,0), .$$

We use the previous graphical representation to make the reductions application clear:

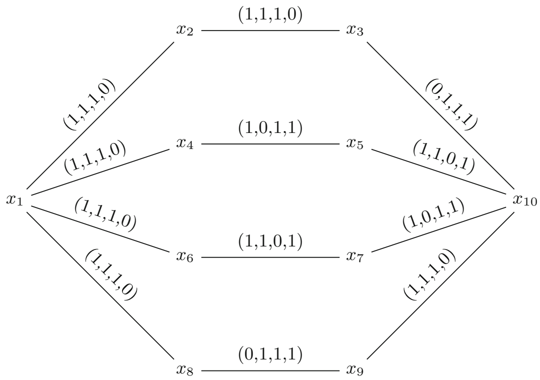

Fig. 8. Representation of a series-parallel formula

First we need to contract on every vertex x_i where $\delta(x_i) = 2$. Then we obtain a new set of edges.

Then we can use the merging of parallel clauses between x_1 and x_{10} (Fig. 10).

The number of models of the initial formula is obtained by adding the four elements of the quadruple, in this case is 38.

5 Complexity Analysis

Although Algorithm1 has a *while* and inside two *for* instructions, in each step either a contraction or a merging operation is applied, hence a clause is removed in each step. So $m-1$ steps are needed to reduce the graph until a single clause is obtained. Each reduction computes a quadruple so $4(m-1)$ operations are needed. The computation of the incident matrix takes $m+n$ steps, so in total $5m+n$ operations are required which represents a procedure of order $O(m+n)$ time complexity.

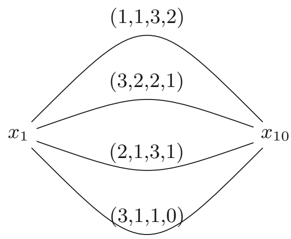

Fig. 9. Result of contracting on x_i where $\delta(x_i) = 2$

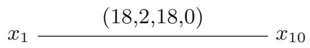

Fig. 10. Result of merging the parallel clauses of Fig. 9

6 Conclusions

In this paper we present an algorithm to compute the $\#2SAT(F)$ when F is a formula whose restricted graph represents a series-parallel graph. We show that our algorithm is correct and its time complexity is linear with respect to the number of variables and clauses of the input formula.

References

1. Winkler, P., Brifhtwell, G.: Counting linear extensions. Order, **8**(e), 225–242 (1991)
2. López-Medina, M.A., Marcial-Romero, J.R., De Ita Luna, G., Montes-Venegas, H.A., Alejo, R.: A linear time algorithm for solving #2SAT on cactus formulas. CoRR, ams/1702.08581 (2017)
3. López, M.A., Marcial-Romero, J.R., De Ita, G., Moyao, Y.: A linear time algorithm for computing #2SAT for outerplanar 2-CNF formulas. In: Martínez-Trinidad, J.F., Carrasco-Ochoa, J.A., Olvera-López, J.A., Sarkar, S. (eds.) MCPR 2018. LNCS, vol. 10880, pp. 72–81. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92198-3_8
4. 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). https://doi.org/10.1007/978-3-540-79723-4_19
5. Schoenmakers, L.A.M.: A new algorithm for the recognition of series parallel graphs. CWI (Centre for Mathematics and Computer Science) (1995)
6. Eppstein, D.: Parallel recognition of series-parallel graphs. Inf. Comput. **98**, 41–55 (1992)
7. Takamizawa, K., Nishizeki, T., Saito, N.: Linear-time computability of combinatorial problems on series-parallel graphs. J. Assoc. Comput. Mach. **29**(3), 623–641 (1982)
8. Gross, J.L., Yellen, J., Zhang, P.: Handbook of Graph Theory. Chapman & Hall/CRC, New York (2013)

9. Dieter, J.: Graphs, Networks and Algorithms., 4th edn. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-32278-5
10. Jakoby, A., Liśkiewicz, M., Reischuk, R.: Space efficient algorithms for directed series-parallel graphs. J. Algorithms **60**(2), 85–114 (2006)
11. Szeider, S.: On fixed-parameter tractable parameterizations of SAT. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 188–202. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24605-3_15
12. Marcial-Romero, J.R., De Ita Luna, G., Hernández, J.A., Valdovinos, R.M.: A parametric polynomial deterministic algorithm for #2SAT. In: Sidorov, G., Galicia-Haro, S.N. (eds.) MICAI 2015. LNCS (LNAI), vol. 9413, pp. 202–213. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27060-9_16