RAIRO
INFORMATIQUE THÉORIQUE

B. APOLLONI
S. DI GREGORIO

A probabilistic analysis of a new satisfiability algorithm

RAIRO – Informatique théorique, tome 16, nº 3 (1982), p. 201-223.
<http://www.numdam.org/item?id=ITA_1982__16_3_201_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 16, n° 3, 1982, p. 201 à 223)

A PROBABILISTIC ANALYSIS OF A NEW SATISFIABILITY ALGORITHM (*)

by B. APOLLONI and S. DI GREGORIO [1]

Communicated by G. AUSIELLO

Résumé. — *Un algorithme pour le problème de la satisfabilité a été développé et une analyse probabilistique montre que, dans l'hypothèse de pleine randomicité sur les variables logiques, un temps polynomial, mais non* « presque partout », *est demandé en moyenne par l'algorithme. Cet algorithme est comparable avec celui qui descend de la méthode de l'arbre de Galil, celui-là étant le plus rapide que nous connaissons, celui que nous proposons est en moyenne encore plus rapide.*

Abstract. — *An algorithm has been developed for the satisfiability problem, where probabilistic analysis shows average polynomial time cost, but not* "almost everywhere", *with respect to input lenght for full randomness in the logical variables. The average running time of this algorithm compares favourably with the Galil tree method which is the fastest yet known.*

1. INTRODUCTION

Satisfiability for conjunctive normal forms is the key problem in studying the complexity of *NP*-complete problems [4, 7, 8, 9]. Of the various algorithms proposed for solving this problem, the enumeration tree procedure devised by Galil [6] is considered the fastest yet known.

Although our algorithm has essentialy the same structure as that of Galil, the particular representation of the clauses not only makes probabilistic analyses more manageable but also brings about improvements.

The main characteristic of both these algorithms is that they can detect and reject in toto some groups of elements in the search space which cannot be solution: the wasteful running time involved in testing elements one by one is generally avoided.

The particular algorithm we propose uses ternary strings to represent both the afore-mentioned groups and the remaining search space. A better sequence of operations takes place and the groups obtained can then be probabilistically treated using the Markov process theory.

(*) Received in May 1981.

[1] Dipartimento di Sistemi, Universita' della Calabria, Arcavacata (Cosenza) Italia.

R.A.I.R.O. Informatique théorique/Theoretical Informatics, 0399-0540/1982/201/$ **5.00**

When the n propositional variables constituting the input clauses are assumed fully random, it is found that the expected running time of our algorithm is a polynomial from the cardinality p of the input clauses indipendent of n. Moreover the distribution law of this running time is such that the probability of the time exceeding a polynomial of the input length decreases, but does not vanish, as n increases.

Given a probability ε arbitrarily small, it is at any rate possible to find a polynomial $g(n, p)$, such that the probability that the running time is greater than this polynomial, is less than ε.

Classes of general boolean formulae requiring a polynomial time for satisfiability have been already identified [12, 14], but only simple cases are evidentiated in our interest field.

The probabilistic results mentioned early indicate that, when our algorithm is used, the bulk of the instances are included in the polynomial classes as n increases.

2. PRELIMINARY DEFINITIONS AND NOTATIONS

The satisfiability problem, which arises in mathematical logic, consists in verifying whether a formula in conjunctive normal form is contradictory.

It is usually defined in a short form [8] as follows:

Let $L=\{x_1, x_2, \ldots, x_n, \bar{x}_1, \ldots \bar{x}_n\}$ be a set of literals; any subset of L which does not contain a complementary pair of literals, is said to be a clause; an instance of the problem is a set of clauses $\{F_1, F_2 \ldots F_p\}$; the property to be verified is the existence of a solution clause F such than $F \cap F_i \neq \emptyset, i=1, 2 \ldots p$.

Example 1: An instance of the problem with solution is given by:

$$\{\{\bar{x}_1, \bar{x}_2, \bar{x}_3, x_6\}, \{x_2, x_3, x_4\}, \{\bar{x}_1, \bar{x}_2, \bar{x}_5\}, \{x_1, x_5, \bar{x}_6\}, \{\bar{x}_2, x_3, x_4\}, \{x_1, \bar{x}_2\}, \{x_2, x_3, \bar{x}_4\}\},$$

with solution:

$$\{x_1, \bar{x}_2, x_3\}.$$

We propose an alternative input representation for the satisfiability problem.

Definition 1: Let $\Sigma=\{0, 1, *\}$ be a set and $\# L=2n$; each clause F is represented by a string $s \in \Sigma^n$ such that:

$$s(j)=1 \Leftrightarrow x_j \in F, \qquad s(j)=0 \Leftrightarrow \bar{x}_j \notin F,$$
$$s(j)=* \Leftrightarrow \bar{x}_j,\ x_j \notin F, \qquad j=1, 2 \ldots n,$$

where $s(j)$ is the j-th element of s.

In our representation an instance of the problem is a set of strings $I=\{s_1, s_2, \ldots, s_p\} \subseteqq \Sigma^n$. A string s_i is satisfied by s if there exists a j such that $1 \leqq j \leqq n$ and $s_i(j)=s(j) \neq *$.

A solution of the instance I is a string s such that all the strings in I are satisfied by s.

Example 2: Example 1 in our representation:

I=000**1, *111**, 00**0*, 1***10, *011**, 10****, *110** with solution string 101***

The definitions and properties given to the strings are analogous to those of the clauses they represent:

DEFINITION 2: Star(s) = number of stars in s:

– s is complete if star(s) = 0, otherwise it is incomplete;

– C_s is the set of complete strings obtained by substituting 0,1 for each star of s in all possible combinations;

– C_s is obviously logically equivalent to s.

PROPOSITION 1: *Two sets* $\{s_1, s_2 \ldots s_m\}$, $\{s'_1, s'_2 \ldots s'_{m'}\}$ *are equivalent iff*:

$$\bigcup_{i=1}^{m} C_{s_i} = \bigcup_{j=1}^{m'} C_{s'_j}.$$

DEFINITION 3: We call T_s the union of all sets of strings equivalent to s. In other words $s' \in T_s$ is either s itself or s with some stars replaced by 0 or 1.

PROPOSITION 2: $C_s \subseteq T_s$, $T_{s'} \subseteq T_s$, $C_{s'} \subseteq C_s$ *with* $s' \in T_s$.

Example 3: *11*01 is equivalent to 011*01, 111*01; note that for $s=\underbrace{** \ldots *}_{n}$ $T_{\bar{s}}=\Sigma^n$.

DEFINITION 4: A function $-:\Sigma \rightarrow \Sigma$ is defined such that $-(0)=1$, $-(1)=0$, $-(*)=*$;

– a string $\bar{s}$ such that $\bar{s}(j)=-s(j)$ and $j=1, 2 \ldots n$ will be called the opposite string to s;

– $\bar{I}=\{\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_p\}$ is called the set of non solutions (*cf.* Prop. 3) of the instance with $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_p$ respectively opposite to $s_1, s_2, \ldots, s_p$ of I.

PROPOSITION 3: $\bar{s}_i \in \bar{I}$, $s' \in T_{\bar{s}_i}$ *cannot be a solution of I by definition* 1. $\bigcup_{i=1}^{p} C_{\bar{s}_i}$ *is therefore the set of all complete strings which are not solutions;*

– $s' \in T_s$ *satisfies* s_i, *if* s *satisfies* s_i;

– $s' \in T_s$ *is a solution if* s *is also a solution.*

DEFINITION 5: The Galil algorithm used in our representation develops a binary tree level by level from the root until a solution node is reached. The required steps are as follows:

(1) the root is labelled with all the strings in I;

(2) each left (right) node at the level ν is labelled with the strings of the parent node with the ν-th element equal to 1 or * (0 or *);

(3) a left (right) node at level ν is a solution node, if it cannot be labelled, i. e. when the parent node lacks strings with the ν-th element equal to 1 or * (0 or *). The path from the root to a node generates a sequence of 0, 1 (go left, go right), which, when padded at the end with enough stars to obtain a string of length n, represents a string, the node-string. The node-string of a solution node is a solution string;

(4) if in a node label at level ν there is a string s such that $s(j)=*$, $j=\nu+1, \ldots, n$, then such a node is a leaf. Nodes at level n must necessarily also be leafs.

PROPOSITION 4: *The Galil tree has the following properties which can be directly deduced from the above rules and previous definitions:*

– *if s is a node-string of any one node and s' is a node string of its successor then $s' \in T_{s'}$*

– *the strings in the label of a node are all those strings in the input which are not satisfied by its node-string,*

– *a node is a leaf if its node-string s is such that $s \in \bigcup_{i=1}^{p} T_{\bar{s}_i}$, thus neither the node itself nor its successors can be solution nodes.*

Example 4: An example of Galil's algorithm is given in figure 1, the instance of satisfiability problem is the same instance of Examples 1 and 2, the solution string is 101***, the node string of the leaf node is 01**** the opposite of the input string 10****

A "reasonable" measure of the algorithmic complexity of Galil's algorithm is the number of strings in the Galil tree. Of course values of the measurements on both our representation and that of Galil are the same.

3. THE SUBSTRACTION ALGORITHM

Given an instance of the satisfiability problem, suppose that we seek solutions in s, i. e. we seek solutions in the "search space" T_s; if $\bar{s}_k$ is some non-solution and $T_{\bar{s}_k} \cap T_s \neq \emptyset$, then it is possible (according our rules) to break s into the

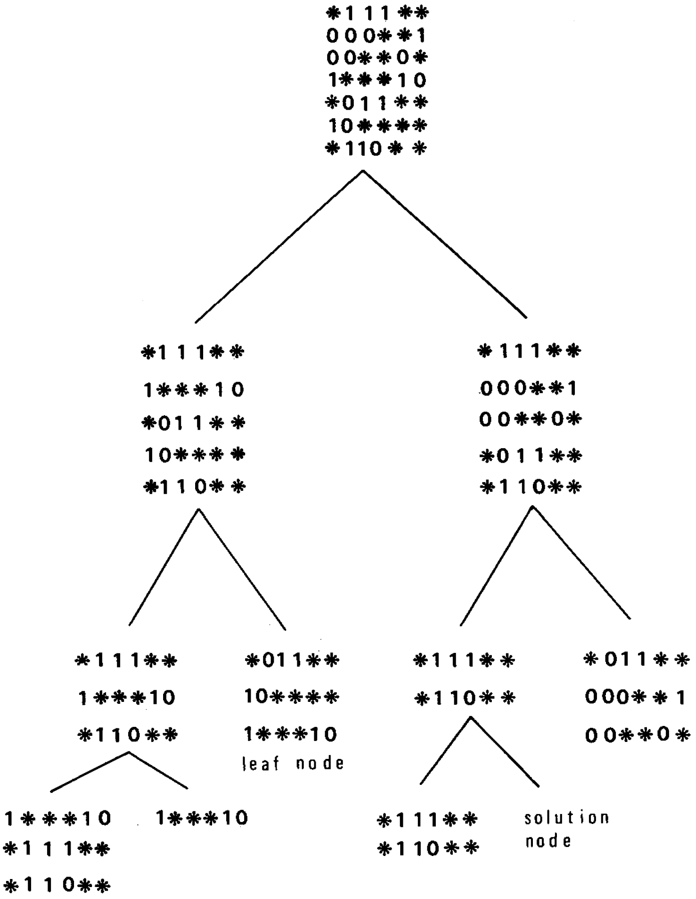


Figure 1. — The Galil tree.

equivalent set $\{s_0, s_1 \dots s_m\}$ such that $T_{s_0}=T_s \cap T_{\bar{s}_k}$ and $\bigcup_{i=1}^{m} T_{s_i} \cap T_{\bar{s}_k}=\emptyset$. A search for a solution will then be continued in $s_1, s_2, \dots, s_m$.

The general intuitive idea of the subtraction algorithm is that a solution is obtained by subtracting the non-solution space from the search space.

Definition 6: comp : $\Sigma^n \times \Sigma^n \to \{\text{true}, \text{false}\}$ is so defined:

$$\text{comp}(s_i, s_k) \text{ is true if}$$

$$s_k(j) \neq *, \qquad s_i(j) \neq * \Rightarrow s_k(j)=s_i(j), \qquad j=1, 2 \dots n.$$

Proposition 5: *If* comp(s_i, s_k) *is true and* $s \in T_{s_i}$ *then* comp(s, s_k) *is true.*

DEFINITION 7: If $\operatorname{comp}(s_i, s_k)$ is true, s^0_{ik} is a string so defined:

$$s^0_{ik}(j) = s_i(j) \qquad \text{if} \quad s_i(j) \neq *,$$

otherwise:

$$s^0_{ik}(j) = s_k(j), \qquad j = 1, 2 \ldots n;$$

$$\operatorname{cost}(s_i, s_k) = \operatorname{star}(s_i) - \operatorname{star}(s^0_{ik}).$$

Example 5: $s_i = {*}{*}11{*}01$, $s_k = {*}1111{*}1$, $\operatorname{comp}(s_i, s_k)$ is true, $s^0_{ik} = {*}111101$.

PROPOSITION 6: $T_{s^0_{ik}} = T_{s_i} \cap T_{s_k} \neq \emptyset$, $C_{s^0_{ik}} = C_{s_i} \cap C_{s_k} \neq \emptyset$ *if* $\operatorname{comp}(s_i, s_k)$ *is true.*

THEOREM 1: *If* $\operatorname{comp}(s_i, s_k)$ *is true and* $m = \operatorname{cost}(s_i, s_k)$, *there is a set:*

$$\{ s^1_{ik}, s^2_{ik} \ldots s^m_{ik} \} \subset T_{s_i}$$

such that:

(1) $C_{s^u_{ik}} \cap C_{s^v_{ik}} = \emptyset$ *where* $u \neq v$ *and* u, v *range* 0 *to* m, *i. e.:*

$$\operatorname{comp}(s^u_{ik}, s^v_{ik}) \text{ is false;}$$

(2) $\bigcup\limits_{w=0}^{m} C_{s^m_{ik}} = C_{s_i}$ *i. e. the set* $\{ s^0_{ik}, s^1_{ik}, \ldots, s^m_{ik} \}$ *is equivalent to* s_i.

Proof: If $m = 0$, then $s^0_{ik} = s_i$ and $s_i \in T_{s_k}$.

If $m > 0$ then $1 \leqq r_1, r_2, \ldots, r_m \leqq m$ are such that $s_i(r_h) = *$:

$$s_k(r_h) = s^0_{ik}(r_h) \neq *, \qquad h = 1, 2, \ldots, m.$$

A specification of $s^1_{ik}, s^2_{ik} \ldots s^m_{ik}$ can then be given by the following rules:

(*a*) $s^u_{ik}(j) = s_i(j)$, $\quad u = 1, 2, \ldots, m, \quad j \neq r_1, r_2, \ldots, r_m$

with j ranging from 1 to n;

(*b*) $s^u_{ik}(r_u) = -s^0_{ik}(r_u) = -s_k(r_u)$, $\quad u = 1, 2 \ldots m$;

(c) $s^u_{ik}(r_t) = s^0_{ik}(r_t) = s_k(r_t)$, $\quad u = 2 \ldots m, \quad t = 1, 2 \ldots u-1$;

(*d*) $s^u_{ik}(r_t) = *$, $\quad u = 1, 2 \ldots m-1, \quad t = u+1, u+2, \ldots, m$.

Assertion 1 can be proved for $u = 0$ or $v = 0$ by considering property (*b*) and that $s^0_{ik}(r_u) \neq *$, $u = 1, 2 \ldots m$. It can be proved for $u \neq 0$ and $v \neq 0$ by considering properties (*b*) and (*c*): suppose $u < v$ then:

$$s^v_{ik}(r_u) = s^0_{ik}(r_u) \qquad \text{and} \qquad s^u_{ik}(r_u) = -s^0_{ik}(r_u),$$

thus $\operatorname{comp}(s^u_{ik}, s^v_{ik})$ is false since $s^0_{ik}(r_u) \neq *$.

For assertion 2 let $s_c \in C_{s_i}$ be considered: if $s_c(r_h) = s^0_{ik}(r_h)$ and $h = 1, 2, \ldots, m$, then $s_c \in C_{s^0_{ik}}$, otherwise there is $1 \leqq h' \leqq m$ such that $s_c(r_{h'}) \neq s^0_{ik}(r_{h'})$ and $s_c(r_{h<h'}) = s^0_{ik}(r_h)$, in which case $s_c \in C_{s^{h'}_{ik}}$ by properties (*a*), (*b*), (*c*), (*d*). □

Example 6:

$$\begin{array}{ll}
s_i = ***0*0 & r_1 = 1, \quad r_2 = 2, \quad r_3 = 3 \\
s_k = 000**0 & \\
s_{ik}^0 = 0000*0 & s_{ik}^1 = 1**0*0 \\
 & s_{ik}^2 = 01*0*0 \\
 & s_{ik}^3 = 0010*0 \\
s_{c_1} = 000010 \in C_{s_{ik}^0} & s_{c_2} = 011000 \in C_{s_{ik}^2}
\end{array}$$

Theorem 2 follows directly from Theorem 1 and Definition 7:

THEOREM 2:

$$C_{s_k} \cup \left(\bigcup_{\omega=1}^{m} C_{s_{ik}^w} \right) = C_{s_i} \cup C_{s_k},$$

$$C_{s_k} \cap \left(\bigcup_{w=1}^{m} C_{s_{ik}^w} \right) = \emptyset.$$

DEFINITION 8: The substitution of s_i by $s_{ik}^1, s_{ik}^2 \ldots s_{ik}^m$ is called the subtraction of s_k from s_i involving the positions $r_1, r_2 \ldots r_m$: $\text{sub}(s_i, s_k) = \{ s_{ik}^1, s_{ik}^2 \ldots s_{ik}^m \}$.

Note that when $\text{comp}(s_i, s_k)$ is false then $\text{sub}(s_i, s_k) = s_i$ and the subtraction is said not effective; if $s_{ik}^0 = s_i$ $\text{sub}(s_i, s_k) = \emptyset$ and the subtraction is said total.

LEMMA 1: *If* $\text{comp}(s_i, s_k)$ *is true*, $\text{sub}(s_i, s_k)$ *in* $r_1, r_2 \ldots r_m = \{ s_{ik}^1, s_{ik}^2 \ldots s_{ik}^m \}$ *and* $s_i \in T_{s_j}$, *then* $\text{comp}(s_j, s_k)$ *is true*,

$$\text{sub}(s_j, s_k) \textit{ in } \quad r'_1, r'_2 \ldots r'_m = \{ s_{jk}^1, s_{jk}^2 \ldots s_{jk}^m \}$$

and:

$$\{ r_1, r_2 \ldots r_m \} \subseteqq \{ r'_1, r'_2 \ldots r'_m \}.$$

PROPOSITION 7: *Supposing* $\text{comp}(s_i, s_k)$ *to be true and* $s_i(q) = *$, $s_k(q) \neq *$ $1 \leqq q \leqq n$, *we can obtain* $\text{sub}(s_i, s_k) = s_{ik}^1, s_{ik}^2 \ldots s_{ik}^m$ *with* $s_{ik}^w(q) \neq *$, $w = 1, 2 \ldots m$ *if* $r_1 = q$.

LEMMA 2: *Given an instance of the satisfiability problem* $I = \{ s_1, s_2 \ldots s_p \}$ *a string is a solution if* $C_s \cap \left(\bigcup_{i=1}^{p} C_{\bar{s}_i} \right) = \emptyset$ *i. e.* $\text{comp}(s, s_i)$ *is false and* $i = 1, 2 \ldots u$.

Proof: By Proposition 3. □

DEFINITION 9: If $s \in \text{sub}(s', s'')$, then s is a son of s' and s' is the father of s. The definitions of successor, ancestor and brother are immediately obvious.

DEFINITION 10: Let s be a string and $T=\{s_1, s_2 \ldots s_q\}$ a set of strings, then $G^{\nu}_{s,T}$ is defined as follows:

$$G^0_{s,T}=\{s\},$$

$$G^{\nu}_{s,T}=\bigcup_{i=1}^{\#G^{\nu-1}_{s,T}} \mathrm{sub}(s_{(\nu-1)_i}, s_{r_i}),$$

where:

$$s_{(\nu-1)_i}\in G^{\nu-1}_{s,T}, \qquad 1\leqq r_i\leqq q,$$

must be such that comp $(s_{(\nu-1)_i}, s_{r_i})$ is true; if such r_i doesn't exist, then it is indifferent which r_i must be taken.

$$C_{G^{\nu}_{s,T}}=\bigcup_{i=1}^{\#G^{\nu}_{s,T}} C_{s_{\nu_i}}.$$

Note that in this definition particular strings to be substracted are not specified as long an effective substraction can be be made.

LEMMA 3: $C_{G^q_{s,T}}\cap\left(\bigcup_{i=1}^{q} C_{s_i}\right)=\emptyset$ *where* $T=\{s_1, s_2\ldots s_q\}$.

Proof: Suppose that $s'\in C_{G^q_{s,T}}$, $s_t\in T$ and comp (s', s) is true, considering that s' was obtained after q subtractions and that no subtraction of s_t was performed, then a string $s_{t'}\neq s_t$ in T was subtracted twice.

The second substraction of $s_{t'}$ was not effective against the definition of $G^{\nu}_{s,T}$. □

Lemma 2 and 3 enables us to develope an algorithm which has been intuitively sketched at beginning of section 3. Let S be the set of all the complete solutions and $\overline{S}=\bigcup_{i=1}^{p} C_{\overline{s}_i}$ the set of all the complete strings which are not solutions for some instance of the satisfiability problem $I=\{s_1, s_2, \ldots, s_p\}$, then $C_{\hat{s}}=S\cup\overline{S}$ where $\hat{s}$ is the string with all stars. If $\overline{s}_1, \overline{s}_2\ldots\overline{s}_p$ are subtracted from $\hat{s}$ and $G^0_{\hat{s},\overline{I}}, G^1_{\hat{s},\overline{I}}\ldots$ are generated, in the positive case we obtain a string s such that comp$(s, \overline{s}_i)$ is false and $i=1, 2, \ldots, p$, i. e. s is a solution by Lemma 2; in the negative case $G^p_{\hat{s},\overline{I}}=\emptyset$ is obtained since $C_{\hat{s}}=\overline{S}$ and no solution can be found.

THEOREM 3: *The following subtraction algorithm holds for the satisfiability problem:*

1. $\nu=0$;

2. *generate* $G^{\nu}_{\hat{s},\bar{I}}$, *such that between possible subtractions from* $s_{(\nu-1)_i} \in G^{\nu-1}_{\hat{s},\bar{I}}$ $i=1, 2, \ldots, \#G^{\nu-1}_{\hat{s},\bar{I}}$ *priority is given in order:* (1) to total subtractions, (2) *to subtractions involving only a position;*

3. *if there is* $s_{\nu_i} \in G^{\nu}_{\hat{s},\bar{I}}$ *such that* comp $(s_{\nu_i}, \bar{s}_j)$ *is false and* $j=1, 2, \ldots, p$, *then* s_{ν_i} *is a solution string. Stop;*

4. *if* $G^{\nu}_{\hat{s},\bar{I}} = \emptyset$, *there is no solution string. Stop;*

5. $\nu \leftarrow \nu+1$, *go to step* 2.

Note that the specification of priority in step 2 *for minimizing the cost of the algorithm is trivial.*

Example 7: We apply the subtraction algorithm for that instance of Example 4 where the Galil algorithm was used.

$$\begin{array}{ll} \bar{s}_1 = 111{*}{*}0 & \bar{s}_5 = {*}001{*}{*} \\ \bar{s}_2 = 11{*}{*}1{*} & \bar{s}_6 = {*}100{*}{*} \\ \bar{s}_3 = {*}000{*}{*}. & \bar{s}_7 = 0{*}{*}{*}01 \\ \bar{s}_4 = 01{*}{*}{*}{*} & \end{array}$$

$$G_{\hat{s},\bar{I}} = \{ {*}{*}{*}{*}{*}{*} \}$$

$$G^{1}_{\hat{s},\bar{I}} = \{ 0{*}{*}{*}{*}{*},\ 10{*}{*}{*}{*},\ 11{*}{*}0{*} \}$$

because sub (******, 11**1*) is performed:

$$\begin{array}{c} {*}{*}{*}{*}{*}{*} \\ \underline{11{*}{*}1{*}} \\ 0{*}{*}{*}{*}{*} \\ 10{*}{*}{*}{*} \\ 11{*}{*}0{*} \end{array}$$

$$G^{2}_{\hat{s},\bar{I}} = \{ 00{*}{*}{*}{*},\ 101{*}{*}{*},\ 1001{*}{*},\ 110{*}0{*},\ 111{*}01 \}$$

because sub (0*****, 01****),

sub(10****, *000**), sub (11**0*, 111**0) are performed:

0*****	10****	11**0*
01****	*000**	111**0
00****	101***	110*0*
	1001**	111*01

101*** is the solution string.

Analogously to the Galil algorithm a "reasonable" measure of algorithmic complexity in the subtraction algorithm is $\sum_{i=1}^{p'} \# G^{i}_{\hat{s},\bar{I}}$, where p' is such that there is $s \in G^{p'}_{\hat{s},\bar{I}}$ solution string, or $G^{p'}_{\hat{s},\bar{I}} = \emptyset$.

In order to compare our algorithm with that of Galil one we must specify which subtraction has to be performed and which order must be given to position $r_1, r_2, \ldots, r_m$ involved in the subtraction.

Although such a specification brings about an increase in the cost of the subtraction algorithm it is essential so that comparison can be made.

DEFINITION 11: For each possible subtraction we put $r_1 < r_2 \ldots < r_m$, and we chose from the various subtractions possible that which has a minimum value r_1, $r_2, \ldots, r_m$ when the lexicographic order for such sequences is considered.

LEMMA 3: *Such specifications enable two goals to be obtained in order:*

(1) *total subtractions are performed, when possible;*

(2) *strings in* $G^{\nu}_{\hat{s},\bar{I}}$ *have the* $\nu -$ *th element different from* $*$, *if possible. When for* $s \in G^{\nu}_{\hat{s},\bar{I}}$ $s(\nu) = *$, *then for* s' *successor of* s *in* $G^{\nu' > \nu}_{\hat{s},\bar{I}}$, $s'(\nu) = *$, i.e. *there is no subtraction for the sons of* s *in the position* ν.

Proof: The demonstration of property 1 is trivial. Suppose, for demonstrating property 2, that for a string $s_a \in G^{\nu-1}_{\hat{s},\bar{I}}$ there is no subtraction involving the position ν, then the same holds for all the successors of s_a by Lemma 1. This property guarantees that if a subtraction is performed for a string $s \in G^{\nu-1}_{\hat{s},\bar{I}}$, then the first position involved in this subtraction is $r_1 \geqq \nu$, if $r_1 = \nu$ then all the strings obtained by subtracting s have, by Prop. 7, a different symbol from $*$ in position ν. □

A binary tree — the subtraction tree — will now be developed from the root, level by level and from left to right. Its nodes at level ν will be labelled by the elements of $G^{\nu}_{\hat{s},\bar{I}}$ with the following rules, until a solution node is found:

(1) the root is labelled with the element of $G^{0}_{\hat{s},\bar{I}}$;

(2) at each level each left (right) node is labelled with strings obtained by subtraction from strings of parent node and having at the ν-th position 0 or * (1). Note that in the Galil algorithm we have the contrary rule, because opposite strings are used.

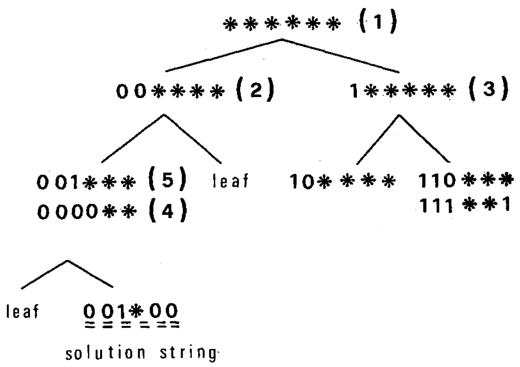
If a node is labelled by a string which cannot be subtracted, it is a solution node and the string s is a solution string.

Example 8: The subtraction tree is built for the same instance of Examples 4 and 7. The numbers in parentheses identify subtractions strings carried out below the tree.

It will be shown in the appendix that, in the worst case, the number of strings present in a subtraction tree is of the same order of the magnitude as the number of the strings in the Galil tree for the same instance of the satisfiability problem.

Fundamentally, in both trees string multiplication occurs in correspondence with the input strings stars in the Galil tree and with the stars of $G^{\nu}_{\hat{s}, \bar{I}}$ strings in our case.

Except for particular distribution of the input strings stars, the Galil tree has a far wider span than that of the subtraction tree.



(1)	(2)	(3)	(4)	(5)
******	00****	1*****	0000**	001***
01****	*001**	111**0	*000**	0***01
1*****	001***	10****	∅	001*1*
00****	0000**	110***		001*00
		111**1		

Figure 2. — The subtraction tree.

4. PROBABILISTIC ANALYSIS OF THE SUBTRACTION ALGORITHM

In this section the probabilistie analysis of the variable $L_{s_0,\,T}\# \bigcup\limits_{j=1}^{p} G^i_{s_0,\,T}$ will be performed, where s_0 is a generic string such that $u_1=n$-star (s), n being the string length and $0\leqq u_1\leqq n$, $T=\{s_1, s_2\ldots s_p\}$, such that star $(s_1)=$star $(s_2)\ldots=$star (s_p) and $u_2=n$-star (s) and $0\geqq u_2\geqq n$.

This analysis can, whithout any loss of generality, be applied to our algorithm in the regular case of 3-satisfiability problem [4, 8, 9, 7].

In this section we shall be concerned with evaluating the expected value $E[L_{s_0,\,T}]$, the variance $V[L_{s_0,\,T}]$ and the limiting distribution law of $L_{s_0,\,T}$, where mathematical expectation is performed with respect to the statistical population of strings in T, for given values both of the string length and $\# T$ (respectively n and p). The results obtained hold when T is assumed full random, i. e. when the positions of the $n-u_2$ stars in each string of T are a random sample from the n variables positions assumed to be equiprobable, and the values of the u_2 literals are a randon sample from a Bernoulli population with parameter $P_B=0{,}5$.

LEMMA 4: *If $s_\nu\in G^\nu_{s_0,\,T}$, then star $(s)\leqq n-\nu-u_1$.*

THEOREM 4: When full randomness is assumed, there exist a polynomial g such that $E[L_{s_0,\,T}]\leqq g(p)$ and a polynomial g′ such that $V[L_{s_0,\,T}]<g'(p, n)$.

Proof: Let us consider a string s_j with star $(s_j)\leqq n-u_1-\nu$, which itself verified hypothesis of full randomness. If $s_i\in T$, and we assume $s_j\in G^\nu_{s_0,\,T}$, the probability that comp (s_i, s_j) is true and that the cost $(s_i, s_j)=m$ is given by:

$$P(m, j, i\mid \nu, s_j, s_i)=P\{[\text{comp}(s_j, s_i)]\cap[\text{cost}(s_j, s_i)=m]\mid s_j\in G^\nu_{s_0,\,T},\ s_i\in T\}$$
$$\leqq\left[\binom{n-\nu-u_1}{m}\binom{\nu+u_1}{u_2-m}\Big/\binom{n}{u_2}\right]2^{-(u_2-m)},\quad (1)$$

where the term in square parentheses in (1) refers to the probability that m stars of s_j match with m nonstars of s_i and the right hand term is the probability that the remaining nonstars of s_i are equal to the corresponding elements of s_j. Equality in (1) is obtained when plus $(s_j)=n-\nu-u_1$. But since not all the strings s_j thus defined belong to $G^\nu_{s_0,\,T}$, it is necessary, when computing the reduction cost inducted by s_i, to consider the joint event that $s_j\in G^\nu_{s_0,\,T}$, comp (s_j, s_i) is true and cost $(s_j, s_i)=m$.

The resulting probability is conditioned only by ν, by the membership of s_i to T and by the particular string s_0 and is given by:

$$\begin{aligned} P(m, j, i, s_j \mid \nu, s_i, s_0) = P\{ & [\text{comp}(s_j, s_i) \text{ is true}] \\ & \cap [\text{cost}(s_j, s_i) = m] \cap \exists s_j \mid \\ & \textit{plus}\ (s_j) \leqq n - \nu - u_1,\ s_i \in T,\ s_0 \} \\ & < \left[\binom{n-\nu-u_1}{m} \binom{\nu+u_1}{u_2-m} \bigg/ \binom{n}{u_2} \right] 2^{-\nu}. \qquad (2) \end{aligned}$$

Taking into account the fact that the elements in $G^{\nu}_{s_0, T}$ are obtained by ν successive subtractions of elements in T starting from s_0 and that the hypothesis of full randomness holds, so we have that:

$$\# C_{G^0_{s_0, T}} > \# C_{G^1_{s_0, T}} > \ldots \# C_{G^{\nu}_{s_0, T}} > \ldots$$

until effective subtraction can be performed. On an average

$$\# C_{G^{\nu}_{s_0, T}} = \frac{1}{2} \# C_{G^{\nu-1}_{s_0, T}}.$$

This explains why the probability that two corresponding nonstars of s_i and s_j are equal is less than 1/2.

To be precise, if we partition the sample space in the subregion such that $s_j \in G^{\nu}_{s_0, T}$ and its complementary, by using the Bayes theorem [5] probability (2) can be written as:

$$\left. \begin{aligned} P(m, j.\ i, s_j \mid \nu, s_i, s_0) &= P(m, j, i \mid \nu, s_1, s_j, s_0)\, P(s_j \in G^{\nu}_{s_0, T}) \\ P(m, j, i \mid \nu, s_i, s_j, s_0) &= P(m, j, i \mid \nu, s_i, s_j) / P(s_0) \\ P(s_j \in G^{\nu}_{s_0, T}) &= 2^{-(\nu+u_1)} \\ P(s_0) &= 2^{-u_1}. \end{aligned} \right\} \qquad (3)$$

So (2) follows from (3).

The subtraction algorithm seeks to identify a string $s_i \in T(\# T = p)$, such that comp (s_i, s_i) is true where $s_j \in G^{\nu}_{s_0, T}$. Considering those subtraction terms connected with the joint probability that more than one string in T is available for the subtraction of $s_j \in G^{\nu}_{s_0, T}$ and the fact that at the level ν, at least ν strings in T have been already used, we can write:

$$\begin{aligned} P(m, j, s_j \mid \nu, T, s_0) = P(\exists s \in T : & [\text{comp}(s_j, s) \text{ is true}] \\ & \cap [\text{cost}(s_j, s) = m] \mid \text{plus}(s_j) \leqq n,\ \nu,\ u_1,\ T,\ s_0) \\ & < p \,.\, P(m, i, j, s_j \mid \nu, s_i, s_0). \qquad (4) \end{aligned}$$

Therefore:

$$
\begin{aligned}
E[\operatorname{cost}(s_j,\ s_i \mid s_j \in G^{\nu-1}_{s_0,\,T})] &= \sum_{t=1}^{u_2} t\,.\,P(t,\ j,\ s_j \mid \nu,\ T,\ s_0)\\
&< p \sum_{t=1}^{u_2} t\,.\,P(t,\ i,\ j,\ s_j \mid \nu,\ s_i,\ s_0)\\
&< (p/2^{\nu}) \sum_{t=1}^{u_2} t\,.\,P(t,\ j,\ i \mid \nu,\ s_j,\ s_i) < u_2 p/2^{\nu}. \qquad (5)
\end{aligned}
$$

From stochastic branching-process theory (3, 11, 15) we have:

$$
E[\#\ G^{\nu}_{s_0,\,T}] = \prod_{i=1}^{\nu} E\ [\operatorname{cost}(s_j,\ s_i \mid s_j \in G^{i-1}_{s_0,\,T})] < (u_2 p)^{\nu}/2^{\nu(\nu+1)/2}, \qquad (6)
$$

$$
\begin{aligned}
V[\# G^{\nu}_{s_0,\,T}] &= (\tilde{\sigma}^2_{\nu}/(E[\operatorname{cost}(s_0,\ s_i)]^2)(E[\# G^{\nu}_{s_0,\,T}])^2\\
&\quad + E[\# G^{\nu}_{s_0,\,T}]\left(1 - \frac{E[\operatorname{cost}(s_j,\ s_i \mid s_j \in G^{\nu}_{s_0,\,T})]}{E[\operatorname{cost}(\mathrm{s}_0,\ s_i)]}\right)\\
&< ((u_2/2)^2/(E[\operatorname{cost}(s_0,\ s_i)])^2)(E[\# G^{\nu}_{s_0,\,T}])^2 + E[\# G^{\nu}_{s_0,\,T}]), \qquad (7)
\end{aligned}
$$

where $\tilde{\sigma}^2$ ranges in the interval $V[\operatorname{cost}(s_0,\ s_i)]$, $V[\operatorname{cost}(s_j,\ s_i \mid s_j \in G^{\nu}_{s_0,\,T})]$, and this quantity, decreasing with ν increasing, is less than the square of the half range of cos $t(s_j,\ s_i)$, i. e. $(u_2/2)^2$.

Therefore the expected value and the variance of $L_{s_0,\,T}$ are less than following functions:

$$
E[L_{s_0,\,T}] < u_2 \frac{u_2^{2\,lg_2(u_2 p)+1} - 1}{u_2 - 1} + 1 = u_2 \frac{u_2 (u_2 p^2)^{lg_2(u_2)} - 1}{u_2 - 1} + 1, \qquad (8)
$$

$$
\begin{aligned}
V[L_{s_0,\,T}] < \Bigg[\bar{\tilde{\sigma}}^2 (E[\operatorname{cost}(s_0,\ s_i')]^2)\Bigg(&\frac{u_2^2 (u_2 p)^{4\,lg_2(u_2)} - 1}{u_2 - 1}\\
&+ 2 + u_2 \frac{u_2 (u_2 p)^{2\,lg_2(u_2)} - 1}{u_2 - 1}\Bigg)\Bigg] \times (1 + D(n,\ u_2,\ p)). \qquad (9)
\end{aligned}
$$

where $\bar{\tilde{\sigma}}^2$ is a weighted mean of $\tilde{\sigma}_{\nu}$, D is a polynomial function of p, u_2 and n, which decreases with n and converges to 0 as n tends to infinity. In fact two steps may be required to obtain the sum of the expected values of $L_{s_0,\,T}$ in (8), which can be partitioned into the sum of terms greater than 1/2 and the sum of those remaining.

The quantity $(u_2,\ p)^{\nu}/2^{\nu(\nu+1)/2}$ *decreases as* ν *increases. Moreover if* $\nu > \nu_0 = 2\,lg_2(u_2 p) + 1$, then $E[\# G^{\nu}_{s_0,\,T}] < 1/2$. Thus, at the end of the first step, the sum may be bounded by $u_2(u_2^{lg_2(u_2p)+1}-1)/u_2-1)$, where this term corresponds to the hypothesis that at each reduction the maximum cost u_2 has been obtained. In the second step, since the serial addend decreases by a factor less than $1/2$ a term less than 1 is added.

The second inequality follows from the consideration that in the expression $V[L_{s_0, T}]$ all those terms relative to the covariance of pairs $\# G_{s_0, T}^{\nu_1}$, $\# G_{s_0, T}^{\nu_2}$ can be expressed in function of the term relative to the variance of $\# G_{s_0, T}^{\nu}$:

$$2\sum_{\nu=1}^{n-u_2}\sum_{t=\nu+1}^{n-u_2} \operatorname{cov}(\# G_{s_0, T}^{\nu}, \# G_{s_0, T}^{t}) < D(n, u_2, p)\sum_{\nu=1}^{n-u_2} V[\# G_{s_0, T}^{\nu}]. \qquad (10)$$

In fact from linear process theory [1], we have:

$$\operatorname{cov}(\# G_{s_0, T}^{\nu}, \# G_{s_0, T}^{t}) = V[\# G_{s_0, T}^{\nu}]\prod_{i=\nu}^{t} E[\operatorname{cost}(s_j, s_i \mid s_j \in G_{s_0, T}^{i})].$$

Thus, with a reasoning similar to that used in proving (8), we have:

$$\sum_{k=\nu+1}^{n-u_2} cov(\# G_{s_0, T}^{\nu}, \# G_{s_0, T}^{t}) < \rho(\nu) . V[\# G_{s_0, T}^{\nu}],$$

$$\rho(\nu) = \begin{cases} 1 + u_2 \dfrac{(u_2 p)^{21g(u_2)} u_2^{2-\nu} - 1}{u_2 - 1} & \text{if} \quad \nu < \nu_0 = 21g_2(u_2 p) + 2, \\ (1/2)^{(\nu-\nu_0+1)(\nu-\nu_0+2)/2} & otherwise. \end{cases} \qquad (11)$$

Since as ν increases, $V[\# G_{s_0, P}^{\nu}]$ increases and the percentage of terms in the sum in the right hand of (11) whose $\nu > \nu_0$ increases correspondingly, from (11) we obtain (9). □

Because of the intriguing expression of $P(m, j, s_j \mid \nu, t, s_0)$ it is not possible to take an analytical form of the distribution law of $L_{s_0, T}$.

Moreover, due to the correlations between the $\# G_{s_0, T}^{\nu}$ the central limit theorem cannot be exploited. It is however possible to guess at the form this distribution will take since the empty cells theory [11] gives Gaussian — and Poisson — like distributions of $G_{s_0, T}^{\nu}$ for large n-small ν, and large n-large ν, respectively; so we must therefore consider a resultant sum distribution obtained by the addition of a gaussian and many correlated Poisson laws.

From the multiplicative process theory [13] we have that as m tends to infinity the probability function $P(m) = P\{L_{s_0, T} = m\}$, for p finite approachs zero faster than exponentially, since we can write:

$$P(m) \leqslant K_0 K_1^{-1} m^{-1/2}, \qquad K_1 > 1.$$

Thus the product $m\,P(m)$ has the least weight on the value of $E[L_{s_0,T}]$ when n is large.

An analysis of the behaviour of $V[L_{s_0,T}]$ gives interesting informations concerning the frequency of cases where the subtraction algorithm runs in a non polynomial time as can be seen from the following theorem:

THEOREM 5: *The probability that the subtraction algorithm runs in a non polynomial time decreases as n increases, but is never* 0.

Proof: We can write (9) as:

$$V[L_{s_0,T}]=K_1(K_2/K_3)(1+D).E[L_{s_0,T}]^2,$$

where K_2, $K_3>1$ multiply respectively $E\,[L_{s_0,T}]$ and $V\,[L_{s_0,T}]$ in order to transform inequalities (8) and (9) into equalities. By construction rules the ratio K_2^2/K_3 tends to 1 very fast for large n. K_1 takes into account the multiplicative factor $[\tilde{\sigma}^2/(E[\mathrm{cost}(s_0,s_i')]^2 u_2)]$ and the additive factor:

$$u_2\frac{u_2(u_2 p)^{2\lg_2 u_2}-1}{u_2-1}$$

Like K_1 itself, the first factor decreases with n.

Therefore the ratio between the standard deviation and the mean of $L_{s_0,T}$ ensures that the probability of $L_{s_0,T}$ being greater than $h\times E[L_{s_0,T}]$ for h large, is decreasing with n increasing.

Since the mean value of the run time is polynomial in the input length, the first part of theorem is proved.

Moreover, for any n, it is always possible to find a worst case running in a non polynomial time, given by the following properties of T:

$$s_0(j)\neq * \quad\Rightarrow\quad s_1(j)=s_2(j)=\ldots s_p(j)=*,$$

$$s_i(j)\neq * \quad\Rightarrow\quad s_0(j)=s_1(j)=\ldots s_{i-1}(j)=s_{i+1}(j)\ldots s_p(j)=*, \qquad 1\leqq j\leqq n.$$

Therefore the above-mentioned probability is never 0. □

At any rate since K_1, K_2, K_3 are bounded by polinomials in n and p, the following theorem holds:

THEOREM 6: *For each* $\varepsilon>0$ *it is possible to find a polinomial* $g(p,n)$ *such that:*

$$P(L_{s_0,T}\geqq g(p,n))<\varepsilon. \tag{12}$$

Proof: By Tchebichev's inequality:

$$P(|L_{s_0,T}-E[L_{s_0,T}]|>K.g'(p, n).E[L_{s_0,T}])\leqq 1/K^2,$$

where $g'(p, n)=(K_1\times K_2^2/K_3)^{1/2}$ is bounded by a polinomial in p and n.

Thus (12) holds for:

$$g(p, n)=(g'(p, n)\sqrt{1/\varepsilon}+1).E[L_{s_0,T}]). \quad \square$$

THEOREM 7: *$L_{s_0,T}$ is not "almost everywhere polinomial" (after Karp's definition* [10]) *in p and n.*

Proof: The proof follows from the last part of proof of Theorem 5. □

The following theorem can be used to make probabilistic comparisons between the proposed algorithm and the Galil one:

THEOREM 8: *The expected value of the running time of the Galil algorithm, in the hypotesis of full randomness, is polynomial in p and n and is greater than the expected value of the subtraction algorithm.*

Proof: In accordance with the comparison between the two algorithms performed in section 3 we assume that the subtraction tree and the Galil tree have essentially the same complexity. So we will compare the times of the subtraction tree with those of the subtraction algorithm, i. e. with the algorithm obtained from the subtraction tree when the constraints on the subtraction order in definition [11] are removed.

The removal of such constraints allows to recognize a leaf or a solution node as soon as the subset of strings labelling them is generated, while no effective variation occurs in the generation of the other nodes.

Let $P^0(\nu)$ be the probability that no string is generated from a string of $G_{s_0,T}^{\nu}$ and $P^{0\prime}(\nu)$ (an apex occurs in symbols referring to the subtraction tree) the probability that a leaf or a solution node can be found at level ν of the subtraction tree, we have:

$$P^{0\prime}(\nu)=(1/(n-\nu))P^0(\nu), \tag{13}$$

where, for semplicity's sake we put $u_1=0$.

The expected value of the running time at level ν is greater therefore in the Galil algorithm than in the subtraction one.

Moreover, if we cut the sum of the expected values of $G_{s_0,T}^{\nu}$ as in the proof of (8) the sum of the terms from $\nu>\nu_0$ to n is less than $u_2^{\nu_0}\nu_0\,lg_2(n)$. In fact, since for such values of ν, $E[\# G_{s_0,T}^{\nu})]$ is less than 1/2, we can disregard $P\{m'=2\}$

and $P\{m'=3\}$ and put $P\{m'=1\}=1-(1-P\{m=1\})/(n-\nu)<1-1/(n-\nu)$. So $E[\# G^{\nu}_{s_0, T}]<u_2^{\nu_0}\nu_0/(n-\nu)$.

Therefore, assigning to the reduction at level $\nu>\nu_0$ the cost u_2 we have:

$$E[L'_{s_0, T}]<u_2 \frac{u_2(u_2 p)^{2\, lg_2(u_2)}}{u_2-1}(1+(2\, lg_2(u_2 p)+1)\, lg_2(n)).$$

and so:

$$E[L'_{s_0, T}]<E[L_{s_0, T}](1+(2 lg_2(u_2 p)+1)\, lg_2(n)), \tag{14}$$

where the second inequality results from the construction of (8).

CONCLUSIONS

An exact algorithm, in accordance with the probabilistic criteria of computational complexity outlined by Karp [10], was developed for the satisfiability problem, where the expected value of the running time for a fully random input is a polynomial of input length.

In the probabilistic frame, the problem of having non-polynomial running time is of no particular significance if the probability of its occurring cannot be specified.

The understanding of exponential cases from a deterministic viewpoint constitutes an ever-challenging topic. In our opinion such cases only occur when it is impossible to find an appropriate representation of binary subtrees by means of bounded strings of symbols.

Our key objective then is to represent some combinatorial schemes on logical variables by special operational symbols and design a formal algebra for these symbols.

Such an approach enables the structure of such trees to be put into focus and some of their special characteristics identified such as invariances in the order of the propositional variables, exponential growth sources, and so on.

We started with simple representation of combinatorial schemes, but more complex ones are under study, our aim being to identify the relationship between the characteristics of the input clauses and the structure of the corresponding tree, so that the probability of non-polynomial cases occurring is further minimized.

APPENDIX

LEMMA 5: *If a string s labels a node of the subtraction tree, and s' is its node string, then* comp (s, s') *is true.*

Proof: By construction rules. □

LEMMA 6: *Given an instance of the satisfiability problem, a node in the subtraction tree is a leaf, when the corresponding node of the Galil tree is also a leaf.*

Proof: Let such a node be at level v, s_t is a string in its label and s_n its node string. By Lemma 5 comp (s_t, s_n) is true, by Property 4 we have $\bar{s}_u \in \bar{I}$, such that $s_n \in T_{\bar{s}_u}$, then by lemma 1 comp $(s_t, \bar{s}_u)$ is true. By Property 2 of Lemma 3, sub $(s_t, \bar{s}_i)$ involves positions $r_1, r_2 \ldots r_m > \mathrm{v}$ and being $\bar{s}_i(\mathrm{v}' > \mathrm{v}) = *$ (by definition $s_n(\mathrm{v}' > \mathrm{v}) = *$ and $s_n \in T_{\bar{s}_n}$), the subtraction is total. □

LEMMA 7: *Given an instance of the satisfiability problem, a node in the subtraction tree is a solution node, if the corresponding node of the Galil tree is a solution node.*

Proof: Let such a node be at the level v, s_t a string in its label and s_n, its node string. By Lemma 5 comp (s_t, s_n) is true and by Proposition 3 s_{tn}^0 is too a solution.

Suppose that s_t is a not solution string, in this case there is $\bar{s}_i \in \bar{I}$ such that comp $(s_t, \bar{s}_i)$ is true.

Consider that by Lemma 3 (*see* the proof) sub $(s_t, \bar{s}_i)$ involves only positions $r_1, r_2, \ldots, r_m > \mathrm{v}$, for position $1 \leqq k \leqq \mathrm{v}$ $s_t(k) = *$ implies $\bar{s}_t(k) = *$, $s_t(k) \neq *$ and $\bar{s}_i(k) \neq *$ imply $s_t(k) = \bar{s}_i(k) = s_{tn}^0(k)$; by definition of s_n, $s_n(j) = *$ $\mathrm{v} < j \leqq n$, then $s_t(j) = s_{tn}^0(j)$.

We thus obtain so the contradictory result that comp $(s_{tn}^0, \bar{s}_i)$ is true and s_{tn}^0 is not a solution string; then s_t is a solution. □

THEOREM 9: *Given an instance of the satisfiability problem, the subtraction tree is a subtree of the Galil tree.*

Proof: By Lemmata 6 and 7 and the general rules for building both trees. □

Note that the number of strings labelling a Galil tree node is less than or equal to p. Though a limit for the number of strings labelling nodes in the subtraction tree cannot be given, we will later prove that relationships exist which bind this number of strings in the Galil tree.

In order to simplify proofs, the more regular case of the 3-satisfiability is now considered without loss of generality.

Let s be a string at the level ν' of a subtraction tree, successors of s at level ν'' can be exponential in $\nu''-\nu'$ only when certain types of subtractions must be exclusively used in order to generate successors of s.

According to the results of Section 4, such cases are exceedingly rare.

Definition 12: The following properties identify the six fundamental types of generic subtraction sub $(s_i, s_k)\in G^{\nu}_{s,\bar{l}}$ for the 3-satisfiability problem:

(1) # sub $(s_i, s_k)=3$ and $s^1_{ik}(\nu)=s^2_{ik}(\nu)=s^3_{ik}(\nu)$;

(2) # sub $(s_i, s_k)=3$ and $s^1_{ik}(\nu)\neq s^2_{ik}(\nu)$ and/or $s^1_{ik}(\nu)\neq s^3_{ik}(\nu)$ (note that by Lemma 3, $s^1_{ik}(\nu)=*\Rightarrow s^2_{ik}(\nu)=s^3_{ik}(\nu)=s^1_{ik}(\nu)$);

(3) # sub $(s_i, s_k)=2$ and $s^1_{ik}(\nu)=s^2_{ik}(\nu)$;

(4) # sub $(s_i, s_k)=2$ and $s^1_{ik}(\nu)\neq s^2_{ik}(\nu)$.

(5) #sub $(s_i, s_k)=1$;

(6) # sub $(s_i, s_k)=0$.

Example 9: In figure 3 examples of types 1, 2 and 3 are given:

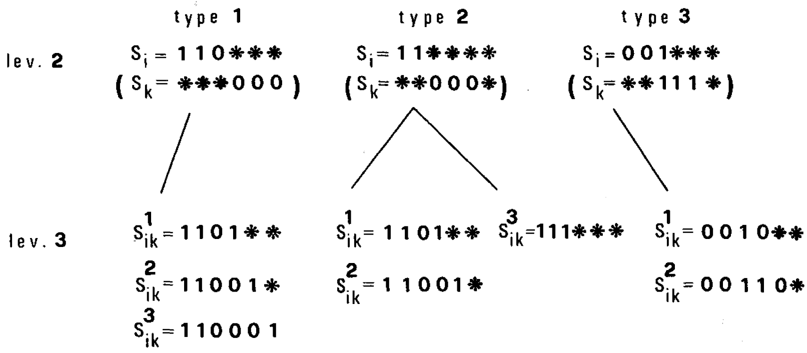


Figure 3. — Examples of subtraction types 1, 2 and 3.

We will now consider only those cases where all the successors of string s at the level ν' are generated up to level ν'' by the same type of subtraction.

Where further comparison analysis is concerned no loss of generality occurs since (1) the successors of two strings labelling the same node are mutually independent, (2) numerical camparison is made between the s-successors and those strings labelling the Galil tree, whose opposite strings are subtracted during a successor generation in the subtraction tree; nodes labelled by such strings in the Galil tree are strictly correlated to nodes labelled and s-successors in the subtraction tree.

THEOREM 10: *Let an instance of* 3-*satisfiability problem be such that the subtractions to be performed for* $s \in G^{\nu'}_{\hat{s}, \bar{I}}$ *and all its successors up to level* $\nu'' > \nu'$ *must be exclusively of type* 1 (2, 3, 4); *let* $\bar{I}' \subseteqq \bar{I}$ *be the set of strings subtracted by s and successors,* $I' \subseteqq I$ *the set of strings whose opposite strings are in* $\bar{I}'$ *and* T', *the subtree of the subtraction tree having root* n_r *labbelled s; then* (1) *the number of strings labelling nodes of* T' *is exponential in* $\nu'' - \nu'$ *in the non-restrictive hypotesis that only s labels the root of such a subtree;* (2) *in the Galil tree the number of nodes labelled by strings in* I' *is exponential in* $\nu'' - \nu'$ *at least.*

Proof for type 1: (1) All the successors of s at level ν of the subtraction tree $\nu' < \nu \leqq \nu''$ label an unique node and their cardinality is $3^{\nu - \nu'}$, thus those strings labelling nodes in T' up to level $\nu'' - \nu'$ are $(3^{\nu'' - \nu'} - 1)/2$. (2) $\# \bar{I}'$ has the least value $\nu'' - \nu'$ when only one string of $\bar{I}$ is subtracted at each level of $\bar{T}'$. $\bar{s} \in \bar{I}'$ iff $\bar{s}(i) = *$, $i = 1, 2, \ldots, \nu''$, otherwise subtractions of type 1 up to level ν'' could take no place; then strings of I' label all the nodes of the Galil tree from the root to level ν'' and the nodes of the subtree T'_G having root node corresponding to n_r up to level $\nu'' + 3(\nu'' - \nu')$ at least, if we consider the extreme case where the first positions of non-star simbols in strings of I' are $\nu'' + 1$, $\nu'' + 2 \ldots \nu'' + (\nu'' - \nu')$: this implies both that there is no leaf in T'_G and that there are $2^{\nu'' - \nu' + 1} - 1$ nodes labelled by strings of I'. If it is assumed that leaves are to be found in T'_G before level $\nu'' - \nu'$, then the Galil tree proves much more extended with a number of nodes in T'_G greater than $2^{\nu'' - \nu' + 1} - 1$.

Proof for type 2: (1) The subtree T' is complete up to level ν'' and is labelled for each level by the strings in $G^{0}_{\hat{s}, \bar{I}}$, $G^{1}_{\hat{s}, \bar{I}}$, $\ldots$, $G^{\nu''}_{\hat{s}, \bar{I}}$ in this order. Considering that $\# G^{\nu}_{\hat{s}, \bar{I}} = 3^{\nu}$ and that onlv subtractions of types 2 are performed, there are $(3^{\nu'' - \nu' + 1} - 1)/2$ strings labelling nodes in T'. (2) By Theorem 9 the subtree T'_G of the Galil tree corresponding to T' is complete, then the number of nodes is $2^{\nu'' - \nu' + 1} - 1$ with each node labelled by strings of I' in accordance with the rules for its construction.

Proof for type 3: (1) All the successors of s at level ν of the subtraction tree $\nu' \leqq \nu \leqq \nu''$ label an unique node and their cardinality is $2^{\nu'' - \nu'}$, thus the number of strings labelling nodes in T' up to level $\nu'' - \nu'$ is $2^{\nu'' - \nu' + 1} - 1$. (2) Consider that s was obtained after ν' subtractions and that, in order to have a sequence of $\nu'' - \nu'$ subtractions of types 3 starting from s, such subtractions must involve nositions greater than ν''.

The Galil tree is extended by Theorem 9 up to level ν'' at least; consider the path from the root to the node with node-string s_n at level ν'' such that comp (s, s_n) is true (we can find more such nodes); the brother nodes of nodes in the path identified by s_n are roots of subtrees which are labelled by strings in I' and are equal to those having root at the brother nodes, except for cases when either the roots or their successors are leaves.

Suppose that there is no exponential growth of nodes in no subtree having root at the brother nodes of nodes in s_n, a set $I'' \subseteqq I$ at least of ν'' strings must exist, which is the minimum number of strings allowed if exponential growth in $\nu'' - \nu'$ is to be avoided with a sufficient number of leaves; but there is a contradiction here since the set $\bar{I}''$ of strings opposite to strings of I'' can be subtracted by ancestors of s and strings of $\bar{I}''$ have lexicographic order priority, thus subtraction of type 3 cannot be performed. There are at most ν' such strings and a number of nodes exponential in $\nu'' - \nu'$ are labelled by strings in I'. □

Proof for type 4: Trivial.

Theorems 9, 10 show that the subtraction tree algorithm is no less efficient than the Galil algorithm, if we think that a generic case can be broken in more considered cases.

It is easy to construct very particular instances of the satisfiability problem where the Galil algorithm runs exponentially and the subtraction algorithm runs polynomially, consider e. g. Galil trees where all the leaves are formed after level $\lfloor n/2 \rfloor$; for some such case the subtraction tree is such that positions greater than $\lfloor n/2 \rfloor$ are involved early in the subtraction and a solution string is immediatily obtained.

REFERENCES

1. K. J. ASTROM, *Introduction to Stochastic Control Theory*, New York, London, Academic Press, 1970.
2. D. E. BARTON and F. N. DAVID, *Combinatorial Chance*, London, Griffin, 1972.
3. A. T. BHARUKA-REID, *Elements of the Theory of Markov Processes and their Applications*, London, McGraw-Hill, 1960.
4. S. A. COOK, *The Complexity of Theorem-Proving Procedures*, Proc. third A.C.M. Symposium on Theory of Computing, 1971, pp. 151-158.
5. H. KRAMER, *Mathematical Methods of Statistics*, Princeton, Princeton University Press, 1945.
6. Z. GALIL, *On Enumeration Procedures for Theorem Proving and for Integer Programming*, Automata Languages and Programming Third International Colloquim, S. MICHAELSON and R. MILNER, Eds., Edinburg University Press, 1976, pp. 355-381.
7. M. R. GAREY and D. S. JOHNSON, *Computers and Intractability*, San Francisco, W. H. Freeman and C., 1979.

8. R. M. KARP, *Reducibility among Combinatorial Problems, in Complexity of Conputer Computations*, R. E. MILLER and J. W. THATCHER, Eds., New York, Plenum Press, 1972, pp. 85-104.

9. R. M. KARP, *On the Computational Complexity of Combinatorial Problems*, Networks, Vol. 5, 1974, pp. 45-68.

10. R. M. KARP, *The Probabilistic Analysis of some Combinatorial Search Algorithm*, Memorandum No. ERL-M581, University of California, Berkeley, 1976.

11. V. F. KOLCHIN, B. A. SEVASTIYANOV and V. P. CHISTIANOV, *Random Allocations*, NewYork, John Wiley, 1978.

12. H. L. LEWIS, *Satisfiability Problems for Propositional Calculi*, Math. System Theory, Vol. 13, 1979, pp. 45-53.

13. R. OTTER, *The Multiplicative Process*, Ann. Math. Statist., Vol. 20, 1949, pp. 206-224.

14. T. J. SHAEFFER, *The Complexitv of Statisfiability Problems*, X Sym. on Theory of Computing, 1978, pp. 216-226.

15. D. L. SNYDER *Random Point Processes*, New York, John Wiley, 1975.