American Journal of Mathematical and Computer Modelling
2026, Vol. 11, No. 1, pp. 39–54
https://doi.org/10.11648/j.ajmcm.20261101.14

Research Article

On Important Applications of Special Set Decomposition

Stepan Margaryan*

Mkhitar Sebastatsi Educational Complex, Yerevan, Armenia

Abstract

This paper is devoted to the study of complexity of finding a special covering for a set, as well as to obtaining some important applications of special decomposition. We formulate the problem of existence of a special covering as a decision problem. To determine the complexity class in which this problem is located, we study the relationship between this problem and the Boolean satisfiability problem, treating them as formal languages. We prove that these problems are polynomially equivalent, which means that the problem of existence of a special covering for a set is an NP-complete problem. In this article we also introduce a new concept 'Replaceable Subsets'. The properties of such subsets are used to fill in the missing elements needed to obtain a special set covering. It is proved that when searching for missing elements to fill a special covering, the order in which these elements are considered does not matter. This result is of great importance in the search for satisfiability of Boolean functions.

Keywords

Special Decomposition, Boolean Satisfiability, Equivalence of Problems

1. Introduction

In [13] the concept of a special decomposition of a set and the concept of special covering for a set under such a decomposition are introduced. These concepts have proven to be flexible and easy to use. They can be used in the field of Boolean functions to study important problems, including the satisfiability problem.

In a general sense, the relationship between the complexity classes of various decision problems [2, 4, 11, 12] still remains open. However, with the formation of the concept of NP-complete problem, many fundamental studies with significant results appear from time to time. In particular, various complexity classes based on time and space complexity, as well as the relations between them, are widely studied in [2, 3, 6, 15, 22].

A well-known theoretical study is Schaefer's result [20], where specific classes of Boolean formulas with certain restrictions are considered. If the formula belongs to one of these classes, then it is decidable in polynomial time. In fact, Schaefer's dichotomy theorem shows that, depending on the restrictions in a propositional formula, the problem either belongs to the class P or is NP-complete [2, 4]. Also, if $P \neq NP$, then the formula is in P, if it satisfies certain conditions. In fact, Schaefer considers only the complexity classes P and NP.

Later, Schaefer's result was refined and extended [1]. It is shown that there is also another classification when considering another approach, since there are different problems in P with completely different complexity, which leads to the existence of different complexity classes. As for Schaefer's classification, unfortunately, some problems arise in determining whether a formula belongs to a given class. It is not always

*Correspondence: Stepan Margaryan (Stepan-Margar@mskh.am)

Received: 13 January 2026; **Accepted:** 26 January 2026; **Published:** 6 February 2026

Copyright: © The Author(s), 2026. Published by Science Publishing Group. This is an **Open Access** article, distributed under the terms of the Creative Commons Attribution 4.0 License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

"easy" to check whether a given formula belongs to a given class. A recognition criteria related to this problem is formulated in [8]. In addition, the issues of approximation of a function to Schaefer's classes by fixing some variables of the function are investigated in [17]. Also, an estimate of the complexity of such an algorithm is obtained.

An important result was also obtained in [7]: it turns out that a number of *NP*-complete problems remain *NP*-complete even with a significant restriction on their scope.

We assume that if there exist conditions that completely determine the complexity class of a problem, then the justification of these conditions as applied to a specific problem will most likely be of an algorithmic nature.

In this article we investigate the complexity of finding a special covering for a set. By formulating this problem and the Boolean satisfiability problem as formal languages [2], we prove that these problems are polynomially reducible [4, 11, 12] to each other. This means that using the special decompositions, we obtain results for Boolean functions.

One of the important contributions of this article is connected with exploring some of the challenges that arise when searching for a special covering for a set, or equivalently, when searching for satisfiability of a Boolean function.

These results we obtain by introducing new concept of 'Replaceable Subsets'.

The essence of this concept is by means of permutation of components of some ordered pairs of the special decomposition, collect all elements in some of the domains of the decomposition. For each element, the corresponding ordered pairs are considered separately.

We prove that when searching for missing elements to fill a special covering for the set, the order in which these elements are considered does not matter.

In addition, another result was obtained concerning Boolean functions. It is proven that any I-transformation of a special decomposition of the set of clauses of a function partitions this set into two satisfiable subsets. These results are of great importance in the search for satisfiability of Boolean functions represented in conjunctive normal form.

Another result that may have significant implications for Boolean functions as well. We prove that any Boolean function represented in conjunctive normal form is satisfiable if and only if it can be transformed, according to certain rules, into a function in which every clause contains a negative literal or every clause contains positive literal. These clauses have some similarity with Horn clauses [5, 9], but they are not always the same.

Note also, that the terminology used in relation to the well-known concepts of set theory corresponds to the terminology in [10, 16, 19, 21].

2. Basic Concepts

Let's recall some definitions, terms and notations from [13], that we will rely on. The sets we will consider are assumed to be ordered unless otherwise stated.

Let the nonempty set $S = \{e_1, e_2,..., e_m\}$ be given. Let also for some natural number n,

$$(M_1^0, M_1^1), (M_2^0, M_2^1),..., (M_n^0, M_n^1)$$

be n arbitrary ordered pairs of arbitrary subsets of the set S.

We use the notation d_nS for an arbitrarily ordered set of these ordered pairs:

$$d_nS = \{(M_1^0, M_1^1), (M_2^0, M_2^1),..., (M_n^0, M_n^1)\}.$$

The Boolean values α_i and $\bar{\alpha}_i$, for $i \in [1, n]$, are used to denote superscripts of subsets:

$$\bar{\alpha}_i = 0 \text{ if } \alpha_i = 1, \text{ and } \bar{\alpha}_i = 1 \text{ if } \alpha_i = 0.$$

The tuple of superscripts $(\alpha_1,..., \alpha_n)$ of subsets is also called a Boolean tuple.

Definition 2.1. The set d_nS will be called a special decomposition of the set S, if

1) $\forall i \in [1, n]\ (M_i^0 \cap M_i^1 = \emptyset)$,

2) $\forall i \in [1, n]\ (M_i^0 \cup M_i^1 \neq \emptyset)$,

3) $\bigcup_{i=1}^{n}(M_i^0 \cup M_i^1) = S$.

Definition 2.2. Let the set d_nS be a special decomposition of the set S.

For any Boolean tuple $(\alpha_1, \alpha_2,..., \alpha_n)$ the ordered set

$$c_nS = \{M_1^{\alpha_1}, M_2^{\alpha_2},..., M_n^{\alpha_n}\}$$

will be called a special covering for the set S under the special decomposition d_nS, if

$$\bigcup_{i=1}^{n} M_i^{\alpha_i} = S.$$

Note that for any $i \in [1, n]$ the subsets M_i^0 and M_i^1 cannot simultaneously belong to the special covering, but one of them exactly belongs.

Let for some natural number n,

$$d_nS = \{(M_1^0, M_1^1),..., (M_n^0, M_n^1)\}$$

be an ordered set of arbitrary ordered pairs of subsets of the set S.

The permutation of the components of some ordered pairs of the set d_nS, let them be

$$\{(M_{i_1}^0, M_{i_1}^1),..., (M_{i_к}^0, M_{i_к}^1)\} \subseteq d_nS,$$

if the order of pairs in d_nS does not change, will be called an I-transformation of the set d_nS. The resulting set is denoted as $(i_1,..., i_k)I(d_nS)$ or simply as $I(d_nS)$:

$$I(d_nS) = \{(M_1^{\alpha_1}, M_1^{\bar{\alpha}_1}),..., (M_n^{\alpha_n}, M_n^{\bar{\alpha}_n})\}.$$

The ordered pairs included in this decomposition are defined as follows:

$(M_i^{\alpha_i}, M_i^{\bar{\alpha}_i}) = (M_i^0, M_i^1)$, if the components of the i-th pair are not displaced,

$(M_i^{\alpha_i}, M_i^{\bar{\alpha}_i}) = (M_i^1, M_i^0)$, if the components of the i-th pair are displaced.

Obviously, the transition from the set I(d_nS) to the set d_nS is also an I-transformation.

Recall that, according to Lemma 1.3 in [13], for any special decomposition of the set S, any I-transformation preserve the possibility of being a special decomposition of the set S and having a special covering for S under such a decomposition.

Thus, with the same pairs of subsets and for any Boolean tuple $(\alpha_1, \alpha_2,..., \alpha_n)$ in generally the special decomposition looks like

$$d_n\mathrm{S} = \{(M_1^{\alpha_1}, M_1^{\bar{\alpha}_1}),..., (M_n^{\alpha_n}, M_n^{\bar{\alpha}_n})\}.$$

We will also use some terms introduced in [13] to distinguish the subsets included in ordered pairs of a special decomposition according to the order of their location in these pairs.

The subsets $M_1^{\alpha_1},..., M_n^{\alpha}$ are called the subsets of 0-domain,

The subsets $M_1^{\bar{\alpha}_1},..., M_n^{\bar{\alpha}_n}$ are called the subsets of 1-domain.

If the ordered pair $(M_n^{\alpha_i}, M_n^{\bar{\alpha}_i}) \in d_nS$, then the subset $M_n^{\alpha_i}$ belongs to the 0-domain and the subset $M_n^{\bar{\alpha}_i}$ belongs to the 1-domain. If the components of this ordered pair are permuted, then the subset $M_n^{\bar{\alpha}_i}$ becomes a subset of the 0-domain, and the subset $M_n^{\alpha_i}$ becomes a subset of the 1-domain.

In addition, for any $\alpha \in \{0, 1\}$, we will use the following notations:

$$M^{\alpha} = \bigcup_{i=1}^{n} M_i^{\alpha_i} \text{ and } M^{\bar{\alpha}} = \bigcup_{i=1}^{n} M_i^{\bar{\alpha}_i}.$$

$$sM^{\alpha} = \{M_1^{\alpha_1},..., M_n^{\alpha_n}\} \text{ and } sM^{\bar{\alpha}} = \{M_1^{\bar{\alpha}_1},..., M_n^{\bar{\alpha}_n}\}.$$

The set sM^{α} will be called a set of α-components or α-domain of the decomposition d_nS.

$(i_1,..., i_k)sM^{\alpha}$ is the set obtained by replacing the subsets $M_1^{\alpha_1},..., M_n^{\alpha_n}$ with the subsets $M_1^{\bar{\alpha}_1},..., M_n^{\bar{\alpha}_n}$, respectively, in the set sM^{α}.

$(i_1,..., i_k)sM^{\alpha}$ will be called a set of α-components of the ordered pairs of decomposition $(i_1,..., i_k)I(d_nS)$.

One of the main goals of the concept special decomposition is to search for a special covering for a given set under a given special decomposition.

With the introduction of the concept of a special decomposition of a set, a problem naturally arises. We will call it a problem of existence of a special covering for a set. Let's formulate it.

Given a special decomposition of a set. If there exists a special covering for this set under the given special decomposition?

Next, we will study the relationship between this problem and the Boolean Satisfiability Problem.

3. Special Decompositions and Boolean Functions as Formal Languages

Let $f(x_1,..., x_n)$ be a Boolean function represented in conjunctive normal form (CNF) [5, 19] with m clauses. It is assumed that the clauses of the function are numbered 1,..., m, so we will use the notation c_j for the j-th clause of the function, $j \in [1, m]$. That is,

$$f(x_1,..., x_n) = \bigwedge_{j=1}^{m} c_j.$$

The clause c_j of the literals $x_{i_1}^{\alpha_1},..., x_{i_k}^{\alpha_k}$ is considered with the following conditions:

$c_j = x_{i_1}^{\alpha_1} \vee ... \vee x_{i_k}^{\alpha_k}$, where $i_1,..., i_k$, $k \in [1, n]$, $\alpha_k \in \{0,1\}$.

In addition, for any i $\in [1, n]$, $x_i^0 = \bar{x}_i$ and $x_i^1 = x_i$.

For simplicity, we assume that no variable and its negation are included in the same clause simultaneously. Obviously, it does not limit the set of functions being considered.

The Boolean Satisfiability Problem or satCNF problem is the problem of determining whether a given Boolean function $f(x_1,..., x_n)$ represented in conjunctive normal form is satisfiable, that is, whether there exists a Boolean tuple $(\sigma_1,...,\sigma_n)$ such that $f(\sigma_1,...,\sigma_n) = 1$.

Our goal is to show that the problem of finding a special covering for a set given a special decomposition of this set is an NP-complete problem [11]. To do this, we first form concrete formal languages [2, 11] over some alphabets. In general, we will use the following alphabet:

$$\Sigma = \{0, 1, x^0, x^1, \star, \circ, e, \varepsilon\}.$$

However, in some individual cases, for convenience, we use the alphabets

$$\Sigma_1 = \{0, 1, x^0, x^1, \star\} \subseteq \Sigma \text{ and } \Sigma_2 = \{0, 1, \star, \circ, e, \varepsilon\} \subseteq \Sigma.$$

The expressions will be represented as strings over these alphabets.

The length of a string is considered to *be the number of symbols included in it* [2].

The symbols 0, 1, x^0, x^1 will be used to form different literals. The literals will be used with the following meaning:

$$x_i^0 = x^0 b(i) = \bar{x}_i \text{ and } x_i^1 = x^1 b(i) = x_i.$$

We will use also the notation x_i^{α} meaning that $\alpha \in \{0,1\}$.

For simplicity we assume that $b(i)$ is a binary expression

of the number i. It will be called an index of the literal.

Obviously, the number of variables of the function will also be the number of literals with pairwise different indices included in the clauses of the function.

Similarly, the symbols e, 0, 1 will be used to form expressions such as $e_i = eb(i)$, for designation elements of sets, where $b(i)$ is the binary expression of the number i.

However, when assigning literals and set elements, we will often use subscripts.

The symbols '$\star$' and '$\triangleleft$' are separating symbols. And 'ε' is a special symbol, which will mean the empty string corresponding to the empty set.

Let's for any natural numbers n and m define the languages $CLAUSE(n)$ and $CNF(n, m)$ over the alphabet

$$\Sigma_1 = \{0, 1, x^0, x^1, \star\} \subseteq \Sigma.$$

For any Boolean tuple $(\alpha_1,..., \alpha_k)$:

CLAUSE $(n) = \{\langle x_{i_1}^{\alpha_1} \ldots x_{i_k}^{\alpha_k} \rangle / \{x_{i_1}^{\alpha_1},..., x_{i_k}^{\alpha_k}\}$ is a set of literals such that $(k, i_j \in [1, n]) \& (i_r \neq i_l$, if $r \neq l)\}$.

It is important to note, that the language $CLAUSE(n)$ does not contain an empty string. Also, it does not contain a string that includes any variable and its negation simultaneously.

We will say that the string $\langle x_{i_1}^{\alpha_1} x_{i_2}^{\alpha_2} \ldots x_{i_k}^{\alpha_k} \rangle$ corresponds to the set $\{x_{i_1}^{\alpha_1}, x_{i_2}^{\alpha_2},..., x_{i_k}^{\alpha_k}\}$ as well as we will say that the string is formed by the clause.

Let c_{j_i} for different indices denote different strings included in the language CLAUSE(n). Then, we define the language CNF(n, m) as follows:

CNF$(n, m) = \{\langle c_{j_1} \star c_{j_2} \star ... \star c_{j_m} \rangle / \forall i \in [1, m], c_{j_i} \in$ CLAUSE$(n)\}$.

Let's now consider the alphabet $\Sigma_2 = \{0, 1, \star, \triangleleft, e, \varepsilon\}$. Using the symbols e, 0 and 1, we compose expressions over the Σ_2 to denote elements of an arbitrary set

$$S = \{e_{j_1}, e_{j_2}, ..., e_{j_m}\}.$$

Since the empty subset plays an important role in the special decomposition, we need a symbol for forming the strings corresponding to the pairs of subsets with an empty component. That is, we will use some symbol to form strings corresponding to the ordered pairs of subsets, such as (M, $\emptyset$) or ($\emptyset$, M) for some nonempty subset M. Let $\varepsilon \in \Sigma$ be such a symbol.

Now for a natural number m we define the languages $SAB(m)$, $SAB^{\varepsilon}(m)$ and $PAIR(m)$ over the alphabet Σ_2 as follows:

SUB$(m) = \{\langle e_{j_1} e_{j_2} ... e_{j_k} \rangle / (k \in [1, m]) \& (j_p \neq j_q$, if $p \neq q)\}$.

$SUB^{\varepsilon}(m) = \{\varepsilon\} \cup$ SUB(m).

Note that the language $SAB(m)$ does not contain an empty string.

We say that the string M $= \langle e_{j_1} e_{j_2} ... e_{j_k} \rangle$ corresponds to the set $\{M\} = \{e_{j_1}, e_{j_2},..., e_{j_k}\}$ or the string M is formed by the set $\{M\}$.

For any set N, we denote by $\langle N \rangle$ the string that is formed by the set N. This means that for a set N of at most m elements $\langle N \rangle \in SUB(m)$.

With such designation we define the language PAIR(m).

PAIR $(m) = \{\langle M \triangleleft N \rangle / (M \in SUB^{\varepsilon}(m)) \& (N \in SUB^{\varepsilon}(m)) \& (\{M\} \cap \{N\} = \emptyset) \& (M \cup N \neq \emptyset)\}$

Note that for any strings $M \in SUB^{\varepsilon}(m)$ and $N \in SUB^{\varepsilon}(m)$, we consider the string $\langle M \triangleleft N \rangle$ to be an ordered pair of strings M and N. Also, for any natural number m,

$\langle M \triangleleft \varepsilon \rangle \in$ PAIR(m) and $\langle \varepsilon \triangleleft N \rangle \in$ PAIR(m)

only with $\{M\} \neq \emptyset$ and $\{N\} \neq \emptyset$.

For convenience, we will use subscripts and superscripts to distinguish between the notation of the substrings included in $PAIR(m)$. That is, for some natural number i, we will use the notation $\langle M_i^{\alpha} \triangleleft M_i^{\bar{\alpha}} \rangle$ for the string $\langle M \triangleleft N \rangle \in$ PAIR(m). Then, for any natural numbers n and m, we define the language $d(n, m)$ over the alphabet Σ_2:

d$(n, m) = \{\langle M_1^{\alpha} \triangleleft M_1^{\bar{\alpha}} \star ... \star M_n^{\alpha} \triangleleft M_n^{\bar{\alpha}} \rangle / \forall i \in [1, n]$ $\langle M_i^{\alpha} \triangleleft M_i^{\bar{\alpha}} \rangle \in$ PAIR(m), and m is the number of elements with pairwise different indices in all subsets$\}$.

The evidence for the following statements is obvious.

Proposition 3.1. For any natural numbers n and m, holds the following:

1) if $\langle x_{i_1}^{\alpha_1} ... x_{i_k}^{\alpha_k} \rangle \in$ CLAUSE(n), and the set $\{x_{l_1}^{\beta_1},..., x_{l_k}^{\beta_k}\}$ is any permutation of the set of literals $\{x_{i_1}^{\alpha_1},..., x_{i_k}^{\alpha_k}\}$, then also $\langle x_{i_1}^{\beta_1},..., x_{i_k}^{\beta_k} \rangle \in$ CLAUSE(n),
2) if $\langle c_{j_1} \star ... \star c_{j_m} \rangle \in$ CNF(n, m), and the set $\{c_{l_1},..., c_{l_m}\}$ is any permutation of the set $\{c_{j_1},..., c_{j_m}\}$, then $\langle c_{l_1} \star ... \star c_{l_m} \rangle \in$ CNF(n, m).
3) $\langle c_{j_1} \star ... \star c_{j_m} \rangle \in$ CNF(n, m) if and only if the clauses corresponding to the strings designated as c_{j_1}, c_{j_2},..., c_{j_m} form a certain function f, represented in CNF with n variables and with m clauses.
4) if the string $\langle e_{j_1} ... e_{j_k} \rangle \in$ SUB(m) and the set $\{e_{l_1},..., e_{l_k}\}$ is any permutation of the set $\{e_{j_1}, ..., e_{j_k}\}$, then $\langle e_{l_1} ... e_{l_k} \rangle \in$ SUB(m).
5) if $\langle M \triangleleft N \rangle \in$ PAIR(m) then $\langle N \triangleleft M \rangle \in$ PAIR(m),
6) if $\langle M_1^{\alpha} \triangleleft M_1^{\bar{\alpha}} \star ... \star M_n^{\alpha} \triangleleft M_n^{\bar{\alpha}} \rangle \in$ d(n, m) and the set $\{(M_{i_1}^{\alpha}, M_{i_1}^{\bar{\alpha}}),..., (M_{i_n}^{\alpha}, M_{i_n}^{\bar{\alpha}})\}$ is any permutation of the set $\{(M_1^{\alpha}, M_1^{\bar{\alpha}}),..., (M_n^{\alpha}, M_n^{\bar{\alpha}})\}$ then

$\langle M_{i_1}^{\alpha} \triangleleft M_{i_1}^{\bar{\alpha}} \star ... \star M_{i_n}^{\alpha} \triangleleft M_{i_n}^{\bar{\alpha}} \rangle \in$ d(n, m)

7) $\langle M_1^{\alpha} \triangleleft M_1^{\bar{\alpha}} \star ... \star M_n^{\alpha} \triangleleft M_n^{\bar{\alpha}} \rangle \in$ d(n, m) if and only if the set $S = \bigcup_{i=1}^{n}(\{M_i^{\alpha}\} \cup \{M_i^{\bar{\alpha}}\})$ consists of m elements and the ordered set of ordered pairs

$\{(\{M_1^{\alpha}\}, \{M_1^{\bar{\alpha}}\}),..., (\{M_n^{\alpha}\}, \{M_n^{\bar{\alpha}}\})\}$, is a special decomposition of the set S, where $\langle M_i^{\alpha} \rangle$ corresponds to the set $\{M_i^{\alpha}\}$ for $i \in [1, n]$.

Thus, strings of different languages have some basic properties of the corresponding sets.

Sometimes, if it does not lead to an ambiguity, we will use

the notation M instead of {M} for the set corresponding to the string ⟨M ⟩.

To estimate the complexity of recognizing these languages, we turn to Turing machines using definitions in [2, 11, 14]. Note that we need to specify the input data for any language. Actually, we have defined these languages for any natural numbers, and the membership of strings to languages depends on these numbers. So, for any language, we compare the corresponding parameters with these numbers. For any string w we will use $|w|$ to denote the number of symbols included in it.

Proposition 3.2. For any natural number n and m, there exist Turing machines T_C, T_{CNF}, T_{SUB}, T_{PAIR} and T_d that recognize the languages CLAUSE(n), CNF(n, m), SUB(m), PAIR(m), and d(n, m), respectively, in polynomial time.

Proof. (i) Let w be a string over the alphabet Σ. We assume that Σ is also an alphabet for each simulated Turing machine. It is convenient and easy simulate a 4-tape Turing machines with an input tape, two work tapes and an output tape [2].

Let's describe the general workflow of T_C. Receiving the string w, T_C does:

(i.1) checks if w starts with one of the symbols x^0 or x^1,

(i.2) checks whether a binary code follows any occurrence of the symbol x^0 or x^1.

(i.3) checks if one of the characters x^0 or x^1, or the special symbol "end of string", follows the binary code. Meanwhile, if T_C encounters an "end of string" symbol, it stops.

(i.4) checks whether the index of current literal is less than or equal to n.

(i.5) checks, whether the index of the current literal differs from the indices of other literals included in the string.

If any of the points (i.1) - (i.5) has a negative answer, T_C rejects the string, otherwise it continues to work until it has considered all pairs.

It is easy to see that with the described workflow, for any natural number n, T_C accepts only the strings belonging to the language CLAUSE(n) and rejects all other strings.

Obviously, the total number of operations required to perform the points (i.1) - (i.3), does not exceed the number $c \times |w|$ for some constant c. Consider the points (i.4) and (i.5):

(i.4) to compare the literal indices with the number n, T_C records the number n on the first work tape, the index of the current literal on the second work tape, and does the following:

1) if the index length is greater than the length of n's record, then T_C rejects the string w,
2) if the length of index is less than the length of n's record, then T_C records the next literal index on the second work tape and continues working,
3) if the lengths are equal, T_C runs over both records on work tapes and compares corresponding binary digits with the same orders in these records. If, in the same order, 0 occurs on the first tape, and 1 on the second tape, T_C rejects the string w, otherwise continues working.

Thus, T_C records each index on the second tape, compares its length with the n's length and compares its digits with n's digits. So, it needs no more than $c \times |w|$ operations for some constant c.

(i.5) let the condition (i.1) - (i.4) be satisfied. This means that the string w has the form

$$w = \langle x_{i_1}^{\alpha_1} \dots x_{i_k}^{\alpha_k} \rangle$$

and T_C will check if the binary numbers i_1,..., i_k are pairwise distinct. It is easy to notice that

$$|w| = |i_1| + \dots + |i_k| + k.$$

For the point (i.5) T_C considers all pairs of literal indices looking for a pair of literals with the same indices: $i_j = i_l$. So, T_C sequentially records the pairs on work tapes, runs over both records and compares corresponding binary digits in the same orders in these records. If a pair of different digits occurs in the current order or the lengths of records are different, T_C moves on to comparing other pairs of indices, otherwise it rejects the string w.

Obviously, T_C requires no more than c $\times$ min($|i_j|$, $|i_l|$) operations for some constant c, to compare the pair of indices i_j and i_l.

It is also easy to prove that the total number of comparison operations of all pairs of literal indices included in the string w does not exceed the number c $\times$ $(|w|)^2$ for some constant c.

(ii) Let's now estimate the complexity of recognizing the language CNF(n, m) for any natural numbers n and m. We describe the general workflow of Turing machine T_{CNF}, that accepts the string w, if $w \in$ CNF(n, m) and rejects it otherwise. Receiving the string w, T_{CNF} does:

(ii.1) checks if the number of occurrences of the symbol '⋆' is equal to (m – 1),

(ii.2) checks whether any occurrence of the character "⋆" in the string is located between a binary digit and a literal.

(ii.3) checks whether any substring is one of the following positions in the string w:

1) between the start and the symbol "⋆",
2) between two symbols "⋆",
3) between the symbol "⋆" and the symbol 'end of string', and does not contain the symbol "⋆", is belongs to the language CLAUSE(n).

If any of the points (ii.1) - (ii.3) has a negative answer, T_{CNF} rejects the string, otherwise it accepts the string w. It is easy to see that with the described workflow, for any natural number n and m, T_{CNF} accepts only the strings belonging to the language *CNF*(n, m) and rejects all other strings.

Obviously, the total number of operations required to perform the points (ii.1) and (ii.2), does not exceed the number $c \times |w|$ for some constant c.

With regard to point (ii.3), if (ii.1) and (ii.2) are satisfied, then w looks like

$$w = \langle c_{j_1} \star c_{j_2} \star ... \star c_{j_m} \rangle,$$

where c_{j_1}, c_{j_2},..., c_{j_m} are clauses. According to point (i.5), these clauses are recognized by Turing machines in no more than

$$c \times (|c_{j_1}|^2 + |c_{j_2}|^2 + ... + |c_{j_m}|^2)$$

operations for some constant c. On the other hand, it is obvious that

$$(|c_{j_1}|^2 + |c_{j_2}|^2 + ... + |c_{j_m}|^2) \leq (|c_{j_1}| + |c_{j_2}| + ... + |c_{j_m}|)^2.$$

Therefore, it is easy to see, that the total number of operations required to accept or reject the string w, does not exceed the number $c \times (|w|)^2$.

(iii) The general operating principles of the Turing machine T_{SAB} are similar to those of the T_C. To recognize the string of the language $SUB(m)$, T_{SUB} does the following:

(iii.1) checks whether the string w starts with the symbols e,

(iii.2) checks whether a binary code follows any occurrence of the symbol e.

(iii.2) checks whether the current binary code is followed by the symbol e or the special symbol "end of string".

(iii.4) checks whether the number of occurrences of the symbol e is less than or equal to m.

(iii.5) checks, whether the index of the current literal differs from the indices of other literals included in the string.

If any of these points has a negative answer, T_{SUB} rejects the string and accepts it otherwise.

Obviously, to accept or reject the string w, T_{SUB} needs no more than $c \times (|w|)^2$ operations.

(iv) The general operating principles of the Turing machine T_{PAIR}.

To recognize the string w of the language $PAIR(m)$, T_{PAIR} does the following:

(iv.1) checks whether w consists of two substrings separated by the symbol '▫'.

(iv.2) checks whether these substrings are included in the language $SUB^{\varepsilon}(m)$.

(iv.3) checks whether the intersection of these substrings is empty.

(iv.4) checks whether one of the substrings differs from ε.

Taking into account the previous reasoning, it can be argued that the total number of operations required to accept or reject a string w does not exceed the number $c \times (|w|)^2$.

(v) Let's now describe the general workflow of Turing machine T_d that recognizes the string of language $d(n, m)$ in a polynomial time.

That is, receiving the string w, the Turing machine T_d runs over it and does:

(v.1) checks if the string w consists of n substrings separated from each other by $(n - 1)$ symbols '⋆'.

(v.2) checks whether each substring is included in $PAIR(m)$.

(v.3) checks whether the number of elements with pairwise different indices in the string w is equal to m.

If any of those conditions is not satisfied, then the T_d rejects the string.

To estimate the number of operations for point (v.1), It is enough to check:

1) whether the number of occurrences of the symbol '⋆' is equal to $(n - 1)$,
2) whether any occurrence of the symbol '⋆' is located between some binary code and the symbol e.

Obviously, the number of required operations does not exceed the number $c \times |w|$.

Point (v.2): let $|w_i|$ be the number of symbols included in the i-th substring. Then, to check if the i-th substring is included in $PAIR(m)$, T_d needs to perform $c \times (|w_i|)^2$ operations. Since $|w_1| + |w_2| + ... + |w_n| \leq |w|$, then to check if all substrings are included in $PAIR(m)$, T_d needs to perform $c \times (|w|)^2$ operations.

Point (v.3): if (v.1) and (v.2) are satisfied, then the string w contains elements with binary codes. Receiving the string w, T_d does the following:

1) records w on the first work tape, and marks an element in it. Let it be the element e_j.
2) records e_j on the second work tape.

T_d runs over the string w and sequentially compares all non-marked elements in it with the element e_j. All elements in that have the same index as e_j are marked.

Running through the entire string w, T_d continues as follows:

1) records e_j on the output tape,
2) looks for the next non-marked element in w and, finding it, repeats the described operations.

If there are no unmarked elements left in w, then obviously the output tape contains all elements with pairwise distinct indices. So, T_d compares their number with m.

Taking into account the arguments in (i.5), it can be argued that the number of needed operations does not exceed the number $c \times |w|^2$.

Thus, strings of any of the described languages are recognized in polynomial time with respect to the length of the input string. ∇

4. Reducibility of Languages

Although there are different definitions of the concept of polynomial reducibility [4, 11, 18], in general they all express a similar relationship between different languages. In this section we will study the reducibility of the Boolean satisfiability problem to the problem of existence of a special covering for a set and vice versa. We will use definitions from [2, 11].

Let we are given a Boolean function $f(x_1,..., x_n)$ represented in conjunctive normal form with m clauses, and let $S = \{e_1, e_2,..., e_m\}$ be any non-empty tuple of m elements. Recall that we use the notation e_j for instead of the string

eb(j) of the alphabet Σ_2.

To each element of the set S, we associate a clause of the function $f(x_1,..., x_n)$ so that the element $e_j = eb(j)$ corresponds to the j-th clause of the function in the sequential enumeration of the clauses. We will call eb(j) the number or the code of the clause, and will identify it with its corresponding clause. So, for brevity, we will denote the set of clauses of the function $f(x_1,..., x_n)$ as $S(f) = \{e_1,..., e_m\}$ and use these elements as clauses. So, the entry $x_i^{\alpha_i} \in e_j$ will mean the inclusion of the literal $x_i^{\alpha_i}$ in the clause corresponding to e_j. Also, the notation $\langle e_j \rangle \in CLAUSE(n)$, will mean that the clause corresponding to e_j belongs to the language $CLAUSE(n)$ and the string $\langle e_j \rangle$ is formed by the clause e_j. So, we designate the following string by f_n:

$$f_n = \langle e_1 \star e_2 \star ... \star e_m \rangle,$$

Obviously, f_n will be the string corresponding to the function $f(x_1,..., x_n)$ in conjunctive normal form, and $f(x_1,..., x_n)$ will be a function corresponding to the string f_n. To denote the subsets of clauses will be used the capital letters corresponding to the function designation:

For some i $\in [1, n]$ we form the following sets:

$F_i^0 = \{e_j / e_j \in S(f)$ and $\bar{x}_i \in e_j\}$ and

$F_i^1 = \{e_j / e_j \in S(f)$ and $x_i \in e_j\}$.

Obviously, $F_i^0 \subseteq S(f)$ and $F_i^1 \subseteq S(f)$.

Proposition 4.1. If for some natural numbers n and m, $f_n \in CNF(n, m)$, then

1) $\forall$ i $\in [1, n] \langle F_i^0 \triangleleft F_i^1 \rangle \in PAIR(m)$,

2) $\bigcup_{i=1}^{n}(F_i^0 \cup F_i^1) = S(f)$.

Proof. To proof of (i) it is enough to prove that for any $i \in [1, n]$,

1) $\langle F_i^0 \rangle \in SUB^{\varepsilon}(m)$ and $\langle F_i^1 \rangle \in SUB^{\varepsilon}(m)$,

2) $F_i^0 \cap F_i^1 = \emptyset$,

3) $F_i^0 \cup F_i^1 \neq \emptyset$,

The proof follows from definitions of languages $SUB^{\varepsilon}(m)$, PAIR(m) and CNF(n, m).

The proof of (ii) follows from the definitions of the sets fM_i^{α} and $fM_i^{\bar{\alpha}}$. ∇

Lemma 4.2. For any natural numbers n and m and for any string $f_n \in CNF(n, m)$, there exists a Turing machine that, receiving the string f_n, outputs the string $\langle F_1^0 \triangleleft F_1^1 \star ... \star F_n^0 \triangleleft F_n^1 \rangle$ in polynomial time with respect to the length of the string f_n.

Proof. Let's describe the general workflow of such a Turing machine, with alphabet Σ.

Similar to Proposition 3.2, we will use a Turing machine denoted by T_1, with input tape, output tape and with two work tapes [2]. The T_1's operation procedure consists of following stages:

Receiving a string of clauses, T_1 records the codes of clauses on the first work tape in the same order in which the clauses are on the input tape.

To form the current strings F_i^0 and F_i^1 for $i \in [1, n]$, T_1 uses the second work tape, successively recording the pairs $(i, 0)$ and $(i, 1)$ on it. We say that T_1 scans a code or a pair of indices, meaning that it scans the symbols that form them.

After recording the numbers of clauses, T_1 starts working, recording (1, 0) on the second work tape and ε on output tape.

Each action of T_1 is determined by configuration of four records on different tapes scanned at current moment. We denote it (H_1, H_2, H_3, H_4) with the following meanings:

H_1 is current scanned record on input tape,

H_2 is current scanned record on first work tape,

H_3 is current scanned record of pair $(i, 0)$ on the second work tape, where $i \in [1, n]$,

H_4 is current scanned record on output tape.

The first configuration is (x_j^{α}, e_1, (1, 0), ε), x_j^{α} is the first symbol of the string f_n, $\alpha \in \{0, 1\}$.

At the beginning of the work, T_1 scans these records on the tapes.

To form F_i^0 and F_i^1, T_1 performs actions depending on configuration:

with configuration ($\bar{x}_i$, e_j, $(i, 0)$, ε),

1) records e_j instead of ε, on the output tape,

2) scans the symbol e_j on the output tape,

3) scans the symbol following of $\bar{x}_i$ on the input tape,

with configuration ($\bar{x}_i$, e_j, $(i, 0)$, e_k),

1) adds e_j to the output tape and scans it,

2) scans the symbol following of $\bar{x}_i$ on the input tape,

with configuration ($\bar{x}_i$, e_j, $(s, 0)$, ε) or ($\bar{x}_i$, e_j, $(s, 0)$, e_k), where $i \neq s$,

scans the symbol following of $\bar{x}_i$ on the input tape,

with configuration ($\star$, e_j, $(i, 0)$, e_k),

1) scans the symbol e_{j+1} on the first work tape

2) scans the symbol following of the symbol '$\star$' on the input tape,

with configuration (end of input string, e_j, $(i, 0)$, e_k),

1) adds the symbol '▫' after e_k on the output tape,

2) records the pair $(i, 1)$ on the next place following the pair $(i, 0)$ and scans it,

3) - records the symbol ε after the symbol '▫' on the output tape and scans it,

4) scans the first symbol on input tape,

with configuration ($\bar{x}_i$, e_j, $(s, 1)$, ε) or ($\bar{x}_i$, e_j, $(s, 1)$, e_k),

scans the symbol following of x_i^{α} on the input tape,

with configuration (end of input data, e_j, $(i, 1)$, e_k) and for $i < n$,

1) adds the symbol '$\star$' on the next place following the e_k on output tape,

2) records the symbol ε on the next place of '$\star$' on the output tape and scans it,

3) records the pair $(i+1, 0)$ on the next place following the pair $(i, 1)$ on the second work tape and scans it,

scans the first symbol on input tape,

with (end of input data, e_j, $(n, 1)$, e_k), the Turing's machine

T_1 stops.

It is easy to see that T_1 outputs the string

$$\langle F_1^0 \triangleleft F_1^1 \star ... \star F_n^0 \triangleleft F_n^1 \rangle.$$

Also, the number of actions performed by T_1 to complete the described procedure is polynomial with respect to the length of input data of the string f_n, denoted by $|f_n|$. More precisely, the number of mentioned operations does not exceed the number $c \times |f_n|^2$ for certain constant c. ∇

Let's denote by $T_1(f_n)$ the string formed as a result of T_1's work on input f_n:

$$T_1(f_n) = \langle F_1^0 \triangleleft F_1^1 \star ... \star F_n^0 \triangleleft F_n^1 \rangle.$$

Lemma 4.3. For any natural numbers n and m,

$f_n \in \text{CNF}(n, m)$ if and only if $T_1(f_n) \in d(n, m)$.

Proof. Obviously, if $f_n \in \text{CNF}(n, m)$ then, according to Proposition 4.1 i),

$$T_1(f_n) = \langle F_1^0 \triangleleft F_1^1 \star ... \star F_n^0 \triangleleft F_n^1 \rangle \in \text{d}(n, m).$$

Now, suppose that $T_1(f_n) \in \text{d}(n, m)$. In this case for any $i \in [1, n]$ the following holds:

1) $F_i^0 \cap F_i^1 = \emptyset$, thus, no clause is included in the subsets F_i^0 and F_i^1 simultaneously.
2) $F_i^0 \cup F_i^1 \neq \emptyset$, that is $F_i^0 \neq \emptyset$ or $F_i^1 \neq \emptyset$.

In fact, for any $i \in [1, n]$, we get the following:

1) there is no clause including the literals $\bar{x}_i$ and x_i simultaneously.
2) the literal $\bar{x}_i$ is included in some clause or the literal x_i is included in some clause.

It means that the number of literals with pairwise different indexes is equal to n.

In addition, each clause e_j included in some set F_i^α cannot be empty, since if $e_j \in F_i^\alpha$, then $x_i^\alpha \in e_j$. Therefore $f_n \in \text{CNF}(n, m)$. ∇

We denote by $dS(f)$ the special decomposition of the set of clauses of the function f:

$$\text{dS}(f) = \{(F_1^0, F_1^1), ..., (F_n^0, F_n^1)\}.$$

If there is a special covering for $S(f)$ in the decomposition $dS(f)$, we denote it as $c_n S(f)$ [13]:

$$c_n S(f) = \{F_1^{\alpha_1}, F_2^{\alpha_2}, ..., F_n^{\alpha_n}\}.$$

For natural numbers n and m, we define the languages sat$CNF(n, m)$ and sC(n, m): satCNF(n, m) = $\{\langle f_n \rangle$ / f is in CNF with m clauses and f is a satisfiable function$\}$, sC$(n, m) = \{\langle M_1^0 \triangleleft M_1^1 \star ... \star M_n^0 \triangleleft M_n^1 \rangle \in \text{d}(n, m)$/there is a tuple $(\alpha_1, ..., \alpha_n)$ suchthat

$$\cup_{i=1}^n \{M_i^{\alpha_i}\} = \cup_{i=1}^n (\{M_i^0\} \cup \{M_i^1\})\}.$$

Theorem 4.4. For any natural numbers n and m,

$f_n \in \text{satCNF}(n, m)$ if and only if $T_1(f_n) \in \text{sC}(n, m)$.

Proof. Suppose that for some natural n and m, $f_n \in \text{sat}CNF(n, m)$ and $f(x_1, ..., x_n)$ is the function corresponding to the string f_n. According to the definition, $f(x_1, ..., x_n)$ is represented in conjunctive normal form with m clauses and is a satisfiable function. That is, there is a Boolean tuple $(\sigma_1, ..., \sigma_n)$ such, that $f(\sigma_1, ..., \sigma_n) = 1$.

We show, that the set $\{F_1^{\sigma_1}, F_2^{\sigma_2}, ..., F_n^{\sigma_n}\}$ is a special covering for the set $S(f)$.

By Proposition 4.1 and Lemma 4.3, it suffices to prove that $\cup_{i=1}^n F_i^{\sigma_i} = S(f)$. Let's show that any clause of the set $S(f)$ is included in some subset

$$F_i^{\sigma_i} \subseteq \{F_1^{\sigma_1}, F_2^{\sigma_2}, ..., F_n^{\sigma_n}\}$$

Let there be a clause e_j that is not included in any of these subsets. This means that none of the literals $x_1^{\sigma_1}, ..., x_n^{\sigma_n}$ is included in the clause e_j, and thus e_j is a disjunction of some literals of the form $x_i^{1-\sigma_i}$. Since $\sigma_i^{1-\sigma_i} = 0$ for any i $\in [1, n]$, then for given assignment of variables $e_j = 0$, which contradicts the assumption that $f(\sigma_1, ..., \sigma_n) = 1$. Therefore, any clause is included in some subset $F_i^{\sigma_i}$ that is included in the set $\{F_1^{\sigma_1}, F_2^{\sigma_2}, ..., F_n^{\sigma_n}\}$, So this set is a special covering for S(f),

$$\cup_{i=1}^n F_i^{\sigma_i} = \cup_{i=1}^n (\{F_i^0\} \cup \{F_i^1\}).$$

Therefore,

$$T_1(f_n) = \langle F_1^\alpha \triangleleft F_1^{\bar{\alpha}} \star ... \star F_n^\alpha \triangleleft F_n^{\bar{\alpha}} \rangle \in sC(n, m).$$

Now assume that for some natural numbers n and m, $T_1(f_n) \in sC(n, m)$. This means that $T_1(f_n) \in d(n, m)$ and there are $(\sigma_1, ..., \sigma_n)$ such that the set $\{F_1^{\sigma_1}, ..., F_n^{\sigma_n}\}$ is a special covering for the set $S(f)$.

Recall that the literal $x_i^{\alpha_i}$ is included in all clauses of the subset $F_i^{\alpha_i}$. Therefore, if $x_i^{\alpha_i} = 1$, then the value of all clauses in $F_i^{\alpha_i}$, is equal to 1, that is:

for any $i \in [1, n]$ and $j \in [1, m]$, if $x_i^{\alpha_i} = 1$ and $e_j \in F_i^{\alpha_i}$, then $e_j = 1$.

Therefore, if $\sigma_1 = \alpha_1, ..., \sigma_n = \alpha_n$, then $f(\sigma_1, ..., \sigma_n) = 1$.

Comparing this with the results of Lemma 4.3, that is:

$T_1(f_n) \in d(n, m)$ if and only if $f_n \in CNF(n, m)$,

we obtain that $f_n \in satCNF(n, m)$. ∇

Remark 4.4.1. According Proposition 3.2, the Turing machine T_{CNF} recognizes the strings of language $CNF(n, m)$ in a polynomial time with respect to the length of string.

Let's for any natural numbers n and m, define the function

J_1: $\Sigma^* \to \Sigma^*$ as follows:

$J_1(w) = T_{CNF}(w)$, if $w \in CNF(n, m)$ and $J_1(w) = \langle \varepsilon \triangleleft \varepsilon \rangle$, if $w \notin CNF(n, m)$.

It is evident, that J_1 is a polynomial time computable function [2] with computable time not exceeded the number $c \times |w|^2$ for certain constant c.

Also, it is evident, that for any string $w \in \Sigma^*$ and for natural numbers n and m,

$w \in CNF(n, m)$ if and only if $J_1(w) \in d(n, m)$.

Recall, that $\langle \varepsilon \triangleleft \varepsilon \rangle \in \Sigma^*$ and $\langle \varepsilon \triangleleft \varepsilon \rangle \notin d(n, m)$.

Theorem 4.5. For any natural numbers n and m,

$$\text{satCNF}(n, m) \leq_p \text{sC}(n, m).$$

Proof. It is obvious that $satCNF(n, m) \subseteq CNF(n, m)$ and $sC(n, m) \subseteq d(n, m)$.

By Lemma 4.2, the Turing machine T_1 generates the string $T_1(f_n)$ in polynomial time for any string $f_n \in CNF(n, m)$. According to Theorem 4.4 and Remark 4.4.1, for any string w, and for any natural numbers n and m,

$f_n \in satCNF(n, m)$ if and only if $J_1(f_n) \in sC(n, m)$.

Since J_1 is a polynomial time computable function, then the language $satCNF(n, m)$ is reduced polynomially to the language $sC(n, m)$. ∇

Now let's show that the problem of the existence of a special covering is polynomially reducible to the satCNF problem. Let $\{e_1,..., e_m\}$ be a nonempty set of m elements composed of the alphabet Σ_2 as above. Let also $\langle dS \rangle$ be a string of the language $d(n, m)$:

$$\langle \text{dS} \rangle = \langle M_1^0 \triangleleft M_1^1 \star ... \star M_n^0 \triangleleft M_n^1 \rangle \in \text{d}(n, m).$$

We form the string belonging to the language $CNF(n, m)$, based on the string $\langle dS \rangle$.

Let for any element $e_j \in S$, $L(e_j)$ be the set of literals composed as follows:

$$L(e_j) = \{ x_i^{\alpha_i} / (i \in [1, n]) \& (\alpha_i \in \{0,1\}) \& (e_j \in \{M_i^{\alpha_i}\})\}.$$

In addition, let $l(e_i)$ be the clause formed by literals of the set $L(e_i)$ and let $c(e_i)$ be the string over the alphabet Σ_1, which forms by the set $L(e_i)$:

$c(e_j) = \langle x_{i_1}^{\alpha_1} x_{i_2}^{\alpha_2} ... x_{i_k}^{\alpha_k} \rangle$, where any literal belongs to the set $L(e_j)$.

We compose the string $\langle c(e_1) \star c(e_2) \star ... \star c(e_m) \rangle$ over the alphabet Σ_1, which corresponds to the Boolean function in conjunctive normal form with m clauses, denoted by $g(dS) = \wedge_{i=1}^{m} l(e_i)$.

It is easy to see, that as a result of formation of all clauses $c(e_i)$, the number of literals
in clauses with pairwise different indexes will be equal to n. Therefore,

$$\langle c(e_1) \star c(e_2) \star ... \star c(e_m) \rangle \in \text{CNF}(n, m).$$

Lemma 4.6. If $\langle dS \rangle = \langle M_1^0 \triangleleft M_1^1 \star ... \star M_n^0 \triangleleft M_n^1 \rangle \in d(n, m)$, then the string

$$\langle c(e_1) \star ... \star c(e_m) \rangle$$

is formed in polynomial time with respect to the length of the string $\langle dS \rangle$.

Proof. Similar to the proof of Lemma 4.2, we describe the general workflow of a particular Turing machine, denoted by T_2, over the alphabet Σ, which time outputs the string

$$\langle c(e_1) \star c(e_2) \star ... \star c(e_m) \rangle,$$

when input the string

$$\langle M_1^0 \triangleleft M_1^1 \star ... \star M_n^0 \triangleleft M_n^1 \rangle.$$

Let's consider a Turing machine T_2, with input tape, output tape and two work tapes.

The general principles of T_2's operation are as follows:

To form the clause $c(e_i)$, it runs over the symbols of input data and with finding the element e_i included in $M_j^{\alpha_j}$, adds the literal $x_j^{\alpha_j}$ to the literals forming the clause $c(e_i)$ on the output tape.

At the first stage of work, T_2 records pairs of indexes (j, α_j) of all subsets $M_j^{\alpha_j}$ on the first work tape in the same order in which the subsets are on the input tape.

The second stage of work consists of forming the clauses $c(e_i)$ for $i \in [1, m]$.

In order to generate a clause corresponding to the element e_i, T_2 uses the second work tape, adding e_i on it and scanning e_i during the formation procedure $c(e_i)$.

After recording the pairs (j, α_j), T_2 starts work by recording e_1 on the second work tape.

To determine each step of T_2 it is enough to consider the configuration of three current scanned records on different tapes, which we denote by $(H_1, H_2, H_3,)$:

H_1 is the scanned symbol on input tape,

H_2 is the scanned pair of indexes on the first work tape of considering subset,

H_3 is the scanned element on the second work tape, that is, the element e_i if the clause $c(e_i)$ is forming.

The first configuration that will be considered to start working over the input data, is

$$(e_k, (1, 0), e_1),$$

where e_k is the first symbol of input data.

To form $c(e_i)$, the Turing's machine T_2 performs actions depending on configuration:

with configurations $(\varepsilon, (j, \alpha_j), e_i)$ or $(e_k, (j, \alpha_j), e_i)$, where $k \neq i$,

scans the next symbol on input tape.

with configuration (e_i, (j, α_j), e_i),

1) adds the literal $x_j^{\alpha_j}$ on output tape,

2) scans the next symbol on input tape.

with configuration (▫, (j, 0), e_i),

1) scans the next symbol on input tape,

2) scans the pair (j, 1) on the first work tape.

with configuration ($\star$, (j, 1), e_i)

1) scans the next symbol on input tape,

2) scans the pair (j+1, 0) on the first work tape.

with configuration (*end of input string*, (j, 1), e_i) and $i < m$,

1) adds $\star$ to the output tape,

2) adds the symbol e_{i+1} to the second work tape and scans it,

3) scans the first symbol of input tape,

4) scans the pair (1, 0) on the first tape,

5) starts to run over the input data again, to detect the element e_{i+1}.

with configuration (*end of input string*, (n, 1), e_m), T_2 stops.

Obviously, on the output tape we will obtain the string

$$\langle c(e_1) \star c(e_2) \star ... \star c(e_m) \rangle.$$

It is easy to see that the number of actions performed by T_2 to complete the described procedure is polynomial with respect to the length $|dS|$ of the string $\langle dS \rangle$.

The number of mentioned actions does not exceed the number $c \times |dS|^2$. ∇

Let's denote by $T_2(dS)$ the string formed as a result of T_2's work on input $\langle dS \rangle$:

$$T_2(\mathrm{dS}) = \langle c(e_1) \star c(e_2) \star ... \star c(e_m) \rangle.$$

Obviously, T_2 forms the string $T_2(dS)$ based on the string $\langle dS \rangle$ for any special decomposition dS of the set $\{e_1,..., e_m\}$.

Theorem 4.7. For any natural numbers n and m,

$\langle dS \rangle \in sC(n, m)$ if and only if $T_2(dS) \in satCNF(n, m)$.

Proof. Let $\langle dS \rangle \in sC(n, m)$. This means that the total number of elements with pairwise different indexes in all subsets is equal to m. So, each of the elements $e_1,..., e_m$ is included in some of subsets forming the string $\langle dS \rangle$, and there is a Boolean tuple (α_1, α_2,..., α_n) such that

$$\bigcup_{i=1}^{n} M_i^{\alpha_i} = \bigcup_{i=1}^{n} (M_i^0 \cup M_i^1).$$

This means that for any element $e_i \in \bigcup_{i=1}^{n} (M_i^0 \cup M_i^1)$, there exists a subset

$M_j^{\alpha_j} \in \{M_1^{\alpha_1}, M_2^{\alpha_1},..., M_n^{\alpha_n}\}$ such that $e_i \in M_j^{\alpha_j}$

On the other hand, if $e_i \in M_j^{\alpha_j}$ then $x_j^{\alpha_j} \in l(e_i)$. Thus, the number of forming clauses will be m, which means that the number of strings corresponding to clauses also will be m.

Let's for simplicity denote by $g_n = T_2(dS)$, that is $g_n = \langle c(e_1) \star c(e_2) \star ... \star c(e_m) \rangle$.

Also, we denote by g the function corresponding to the string g_n.

Since $x_j = \alpha_j$ implies $x_j^{\alpha_j} = 1$, then ($\sigma_1 = \alpha_1$) &... & ($\sigma_n = \alpha_n$) implies

$$g(\sigma_1,..., \sigma_n) = g(dS)(\sigma_1,..., \sigma_n) = 1.$$

Therefore, $T_2(dS) = \langle c(e_1) \star c(e_2) \star ... \star c(e_m) \rangle \in satCNF(n, m)$.

Now suppose, that $T_2(dS) \in sat\ CNF(n, m)$. That is, g is a satisfiable function, which means that for some Boolean tuple ($\sigma_1,..., \sigma_n$), $g(\sigma_1, \sigma_2,..., \sigma_n) = 1$.

According to Theorem 4.4,

$g_n \in$ sat $CNF(n, m)$ if and only $T_1(g_n) \in sC(n, m)$.

Since $T_1(g_n) = \langle G_1^0 \text{ ▫ } G_1^1 \star ... \star G_n^0 \text{ ▫ } G_n^1 \rangle$, then for some Boolean tuple $\sigma_1,..., \sigma_n$, we obtain

$$\bigcup_{i=1}^{n} G_i^{\sigma_i} = \bigcup_{i=1}^{n} (G_i^0 \cup G_i^1).$$

This means that for any clause $l(e_i)$, there exists a subset $G_j^{\sigma_j}$ such that

$$G_j^{\sigma_j} \in \{G_1^{\sigma_1}, G_2^{\sigma_2},..., G_n^{\sigma_n}\} \text{ and } l(e_i) \in G_j^{\sigma_j}.$$

But then $e_i \in M_j^{\sigma_j}$, because of definition of $G_j^{\sigma_j}$:

$G_j^{\sigma_j} = \{l(e_k) / l(e_k) \in S(g)$ and $l(e_k)$ contains $x_j^{\sigma_j}$, ($k \in [1, m]$)$\}$.

At the same time the clause $l(e_i)$ contains the literal $x_j^{\sigma_j}$ only if $e_i \in M_j^{\sigma_j}$.

Since any element e_i determines the composition of one clause, and any clause is defined by one element, then it is easy to prove that for any element e_i there exists a subset $M_j^{\sigma_j} \in \{M_1^{\sigma_1}, M_2^{\sigma_2},..., M_n^{\sigma_n}\}$ such, that $e_i \in M_j^{\sigma_j}$.

So, $\bigcup_{i=1}^{n} M_i^{\sigma_i} = \bigcup_{i=1}^{n} (M_i^0 \cup M_i^1)$.

Therefore, $\langle M_1^0 \text{ ▫ } M_1^1 \star ... \star M_n^0 \text{ ▫ } M_n^1 \rangle \in sC(n, m)$. ∇

Remark 4.7.1. According Proposition 3.2, the Turing machine T_d recognizes the strings of language $d(n, m)$ in a polynomial time with respect to the length of string. More precisely, this time is estimated as $c \times |w|^2$, where w is an input string.

Let's for natural numbers n and m, define the function J_2: $\Sigma^* \to \Sigma^*$ as follows:

$J_2(w) = T_2(w)$, if $w \in d(n, m)$ and $J_2(w) = \langle \varepsilon \text{ ▫ } \varepsilon \rangle$ if $w \notin d(n, m)$.

It is evident, that J_2 is a polynomial time computable function with computable time not exceeded the number $c \times |w|^2$ for certain constant c.

It is also obvious that for any natural numbers n and m, and for any string $w \in \Sigma^*$,

$w \in d(n,m)$ if and only if $J_2(w) \in CNF(n,m)$.

Recall, that $\langle \varepsilon \circ \varepsilon \rangle \in \Sigma^*$ and $\langle \varepsilon \circ \varepsilon \rangle \notin d(n, m)$.

Theorem 4.8. For any natural numbers n and m, $sC(n, m) \leq_p satCNF(n, m)$.

Proof. It is obvious that $sC(n, m) \subseteq d(n, m)$ and $satCNF(n, m) \subseteq CNF(n, m)$.

According to Lemma 4.6, for any natural numbers n and m, there is a Turing's machine T_2, which for any string

$$\langle dS \rangle = \langle M_1^0 \circ M_1^1 \star ... \star M_n^0 \circ M_n^1 \rangle \in d(n, m)$$

forms the string

$$T_2(dS) = \langle c(e_1) \star c(e_2) \star ... \star c(e_m) \rangle$$

in polynomial time with respect to the length of string $\langle dS \rangle$. According to the Theorem 4.7 and Remark 4.7.1, for any string w, and for any natural numbers n and m,

$\langle dS \rangle \in sC(n, m)$ if and only if $J_2(dS) \in satCNF(n, m)$.

Therefore, $sC(n, m) \leq_p satCNF(n, m)$.

Since J_2 is a polynomial time computable function, then the language $sC(n, m)$ is polynomially reduced to the language $satCNF\ (n, m)$. ∇

Theorem 4.9. For any natural numbers n and m, the language $satCNF\ (n, m)$ and the language $sC(n, m)$ are polynomially equivalent.

Proof. Follows of Theorems 4.5 and 4.8.

Corollary 4.9.1 The problem of existence of a special covering is an NP-complete problem.

Proof. Follows from the Theorem 4.9. ∇

The theorem 4.9 gives us possibility to use the concept of special decomposition when discuss the questions concerning to satisfiability of a function. That is, instead of investigating questions related to the satisfiability a given function in conjunctive normal form, we can investigate corresponding questions related to the existent of a special covering for the set of clauses under a special decomposition of this set.

5. Replaceability of Subsets

In this section, to study important properties of special decompositions, we introduce a new concept called replaceable subsets.

Let $S = \{e_1, ..., e_m\}$ be a nonempty set, and let for some Boolean tuple $(\alpha_1, ..., \alpha_n)$,

$M_1^{\alpha_1}, M_1^{\bar{\alpha}_1}, ..., M_n^{\alpha_n}, M_n^{\bar{\alpha}_n}$ be arbitrary subsets of the set S such that the set

$$d_nS = \{(M_1^{\alpha_1}, M_1^{\bar{\alpha}_1}), ..., (M_n^{\alpha_n}, M_n^{\bar{\alpha}_n})\}$$

is a special decomposition of the set S.

Recall that in this case

$sM^0 = \{M_1^{\alpha_1}, ..., M_n^{\alpha_n}\}$ and $sM^1 = \{M_1^{\bar{\alpha}_1}, ..., M_n^{\bar{\alpha}_n}\}$.

Definition 5.1. If for some $\alpha \in \{0,1\}$, the set of α-components of the special decomposition d_nS covers the set S, then it will be called a special sM^α-covering for the set S.

According to Lemm 1.5 in [13], there exists a special covering for the set S under the decomposition d_nS if and only if for some $\alpha \in \{0,1\}$ there exists an sM^α-covering for the set S under some special decomposition $I(d_nS)$. The search for an sM^0-covering in a decomposition d_nS consists of finding ordered pairs such that, by permuting their components, the missing elements of the 0-domain are moved to the 0-domain.

Obviously, if the 0-domain of some special decomposition covers the set S, then permutating the components of all ordered pairs of this decomposition, we obtain sM^1-covering under another special decomposition and vice versa. So, the notation sM^α-covering, will mean that $\alpha \in \{0, 1\}$.

Definition 5.2. Let d_nS be a special decomposition of a set S such that $M_i^{\alpha_i} \in sM^0$ and the 0-domain of the decomposition does not cover the set S.

1) We say that the subset $M_i^{\alpha_i}$ is an immediate sM^0-replaceable subset, if $M^0 \subseteq (i)M^0$.
2) We say that $M_i^{\alpha_i}$ is an sM^0-replaceable subset if it is an immediate sM^0-replaceable or, if d_nS contains ordered pairs (let $i_1, ..., i_k$ be their indices) such that $M^0 \subseteq (i, i_1, ..., i_k)\ M^0$.
3) We say that $e \in S$ is an sM^0-reachable element if there is an ordered pair $(M_i^{\alpha_i}, M_i^{\bar{\alpha}_i})$ such that

$(M_i^{\alpha_i} \in sM^0)$ & $(e \notin M^0)$ & $(e \in M_i^{\bar{\alpha}_i})$, and M_i^α is an sM^0-replaceable subset. We will also say that the element e is associated with sM^α-replaceability of the subset $M_i^{\alpha_i}$.

The essence of the concept of sM^α-replaceability is to perform permutations of the components of some ordered pairs, as a result of which the α-domain receives a new element without losing its own elements. It is easy to see the M^α-replacement of a subset is an I-transformation of the given decomposition. Therefore, according to the Lemma 1.5 in [13], for any special decomposition, as a result of any replacement, we obtain a new special decomposition.

Theorem 5.3. Let for some Boolean tuples $(\tau_1, ..., \tau_n)$ and $(\alpha_1, ..., \alpha_n)$, the set

$$c_nS = \{M_1^{\tau_1}, ..., M_i^{\tau_i}, ..., M_n^{\tau_n}\}$$

be a special covering for the set S under the special decomposition d_nS,

$$d_nS = \{(M_1^{\alpha_1}, M_1^{\bar{\alpha}_1}), ..., (M_n^{\alpha_n}, M_n^{\bar{\alpha}_n})\}.$$

Then, for any $i \in [1, n]$, the subset $M_i^{\bar{\tau}_i}$ is an $sM^{\bar{\tau}_i}$-replaceable subset.

Proof. For any $i \in [1, n]$, if $M^{\bar{\tau}_i} \subseteq (i)\ M^{\bar{\tau}_i}$, then $M^{\bar{\tau}_i}$ is an immediate $sM^{\bar{\tau}_i}$-replaceable subset. Suppose that for some $i \in [1, n]$, the subset $M_i^{\bar{\tau}_i}$ contains elements that are

not included in other subsets of the $\bar{\tau}_i$-domain. Since c_nS is a special covering for the set S, then obviously, for any element $e \in M_i^{\bar{\tau}_i}$ there is a subset $M_j^{\tau_j}$ such that $(M_j^{\tau_j} \in c_nS)$ & $(e \in M_j^{\gamma_j})$.

Suppose that for some $\gamma_1,..., \gamma_k \in \{0, 1\}$, $M_{i_1}^{\gamma_1},..., M_{i_k}^{\gamma_k}$ are the subsets such that

$$(\{M_{i_1}^{\gamma_1},..., M_{i_k}^{\gamma_k}\} \subseteq c_nS) \& (M_i^{\bar{\tau}_i} \subseteq \bigcup_{j=1}^{k} M_{i_j}^{\tau_j}).$$

It is obvious that some of these subsets are included in the τ_i-domain, since by assumption, $M_i^{\bar{\tau}_i}$ is not immediate replaceable subset. Let them be the subsets $M_{j_1}^{\beta_1},..., M_{j_l}^{\beta_l}$. That is,

$$(\{M_{j_1}^{\beta_1},..., M_{j_l}^{\beta_l}\} \subseteq c_nS) \& (\{M_{j_1}^{\beta_1},..., M_{j_l}^{\beta_l}\} \subseteq sM_i^{\tau_i}).$$

Let's consider the case, when $\tau_i = \alpha_i$. For $\tau_i = \bar{\alpha}_i$, will be a similar proof. In this case, the ordered pair $(M_i^{\tau_i}, M_i^{\bar{\tau}_i}) \in d_nS$ is the same as $(M_i^{\alpha_i}, M_i^{\bar{\alpha}_i})$. But then $\{M_{j_1}^{\beta_1},..., M_{j_l}^{\beta_l}\} \subseteq sM_i^{\alpha_i}$.

Therefore, the subsets $M_{j_1}^{\beta_1},..., M_{j_l}^{\beta_l}$ are moved to the $\bar{\tau}_i$-domain by permutations of the components of the ordered pairs

$$(M_i^{\alpha_i}, M_i^{\bar{\alpha}_i}), (M_{j_1}^{\beta_1}, M_{j_1}^{\bar{\beta}_1}),..., (M_{j_l}^{\beta_l}, M_{j_l}^{\bar{\beta}_l}).$$

If some of the subsets $M_{j_1}^{\bar{\beta}_1},..., M_{j_l}^{\bar{\beta}_l}$ contain elements that are not included in other subsets in the $\bar{\tau}_i$-domain, then by the same reasoning as for the subset $M_i^{\bar{\tau}_i}$, we find the corresponding ordered pairs and perform permutations of their components.

Since as a result of all permutations new subsets included in c_nS move into the $\bar{\tau}_i$-domain, this domain will not lose elements. Therefore, $M_i^{\bar{\tau}_i}$ is an $sM^{\bar{\tau}_i}$-replaceable subset. ∇

Corollary 5.3.1. Let a special decomposition d_nS of the set S be given such that some element $e \in S$ is not included in the α-domain of this decomposition, $e \notin M^{\alpha}$.

If e is not an sM^{α}-reachable element, then there is no special covering of the set S under the decomposition of d_nS.

Proof. Follows directly from Theorem 5.3.∇

Corollary 5.3.2. If there exists a special covering for the set S under the decomposition d_nS, then for any $i \in [1, n]$ the following is true:

the subset M_i^0 is sM^0-replaceable or the subset M_i^1 is sM^1-replaceable.

Proof. Let for some Boolean tuple $(\alpha_1, \alpha_2,..., \alpha_n)$, the set $c_n\mathrm{S} = \{M_1^{\alpha_1},..., M_n^{\alpha_n}\}$ be a special covering for the set S. Obviously, the conditions of Theorem 4.2 are satisfied. So,

for $\alpha_i = 0$, then the subset M_i^1 is an sM^1-replaceable subset,

for $\alpha_i = 1$, then the subset M_i^0 is an sM^0-replaceable subset. ∇

In search of sM^0-reachability of an element $e \in \mathrm{S}$ such that $(e \notin M^0)$ & $(e \in M_i^{\bar{\alpha}_i})$, we look for ordered pairs that ensure this reachability. At the same time, an important question arises when studying the existence of a special covering for a set.

Let $\{e_{q_1},..., e_{q_l}\} = S \setminus M^0$. The elements $e_{q_1},..., e_{q_l}$ can occur in different subsets in the 1-domain of the decomposition $d_n\mathrm{S}$. So, to find a special covering for the set S, we can perform a sequential search for sM^0-reachability of each element of the set $\{e_{q_1},..., e_{q_l}\}$:

1) search for ordered pairs that will ensure the sM^0-reachability of an element included in the set $\{e_{q_1},..., e_{q_l}\}$ under the decomposition $d_n\mathrm{S}$, and, having found them, permute their components. The result will be a new special decomposition.
2) we perform the same procedure for some other element of the set $\{e_{q_1},..., e_{q_l}\}$ under the new decomposition and continue the similar procedure for all elements.

Let's find out whether finding a special covering depends on the order in which the missing elements are considered.

Definition 5.4. Let we are given a special decomposition $d_n\mathrm{S}$ of the set S. For an ordered set $\{e_{i_1},..., e_{i_l}\} \subseteq \mathrm{S}$, we define a decomposition denoted by $d_nS^0(e_{i_1},..., e_{i_l})$:

(i) if $e_{i_1} \in M^0$, then $d_nS^0(e_{i_1})$ coincides with the decomposition $d_n\mathrm{S}$.

(ii) if $e_{i_1} \notin M^0$, then we consider the following two cases:

(ii.1) e_{i_1} is an sM^0-reachable element under the decomposition $d_n\mathrm{S}$.

In this case, $d_nS^0(e_{i_1})$ is defined as a decomposition obtained by permuting the components of the ordered pairs that ensure this reachability.

(ii.2) e_{i_1} is not an sM^0-reachable element under the decomposition $d_n\mathrm{S}$.

In this case, we consider the resulting decomposition as an empty set, $d_nS^0(e_{i_1}) = \emptyset$.

(iii) let for some $1 \leq k < l$, $d_nS^0(e_{i_1},..., e_{i_k}) \neq \emptyset$.

If the element $e_{i_{k+1}}$ is included in the 0-domain of this decomposition, then the decomposition $d_nS^0(e_{i_1},..., e_{i_k}, e_{i_{k+1}})$ coincides with the decomposition $d_nS^{\alpha}(e_{i_1},..., e_{i_k})$.

If $e_{i_{k+1}}$ is not included in the α-domain of the decomposition $d_nS^0(e_{i_1},..., e_{i_k})$, then we consider the following two cases:

(iii.1) $e_{i_{k+1}}$ is an sM^0-reachable element under the decomposition $d_nS^0(e_{i_1},..., e_{i_k})$.

In this case, $d_nS^0(e_{i_1},..., e_{i_k}, e_{i_{k+1}})$ is defined as a decomposition that is obtained by permuting the components of ordered pairs that ensure this reachability.

(iii.2) $e_{i_{k+1}}$ is not an sM^0-reachable element under the decomposition $d_nS^0(e_{i_1},..., e_{i_k})$. In this case, we will consider

the decomposition $d_nS^0(e_{i_1},..., e_{i_k}, e_{i_{k+1}})$ as an empty set.

(iv) if for some $1 \leq k < l$, $d_nS^0(e_{i_1},..., e_{i_k}) = \emptyset$, then also, $d_nS^0(e_{i_1},..., e_{i_k}, e_{i_{k+1}}) = \emptyset$.

Theorem 5.5. Let the special decomposition d_nS of the set S be given such that

$$\{e_{q_1},..., e_{q_l}\} = S \setminus M^0.$$

(i) if there exists a special covering for the set S under the special decomposition d_nS, and $\{e_{j_1},..., e_{j_l}\}$ is an arbitrary permutation of the elements of the set $\{e_{q_1},..., e_{q_l}\}$, then:

(a) $d_nS^0(e_{j_1},..., e_{j_l}) \neq \emptyset$,

(b) the 0-domain of the decomposition $d_nS^0(e_{j_1},..., e_{j_l})$ is an sM^0-covering for S.

(ii) if $d_nS^0(e_{j_1},..., e_{j_k}) = \emptyset$ for an arbitrary ordered set $\{e_{j_1},..., e_{j_k}\}$ consisting of some elements of the set $\{e_{q_1},..., e_{q_l}\}$, then there does not exist a special covering for the set S under the decomposition d_nS.

Proof. (i) Let for some Boolean tuple $(\alpha_1,..., \alpha_n)$, the set $c_nS = \{M_1^{\alpha_1},..., M_n^{\alpha_n}\}$ be a special covering for the set S under the decomposition d_nS. The condition $S \setminus M^0 = \{e_{q_1},..., e_{q_l}\}$ implies that $c_nS \neq sM^0$ and $\{e_{q_1},..., e_{q_l}\} \subseteq M^1$. Therefore, for any $e_{q_j} \in \{e_{q_1},..., e_{q_l}\}$, there is a subset included in c_nS, which contains the element e_{q_j}.

Let $M_{i_1}^{\bar{\tau}_1},..., M_{i_k}^{\bar{\tau}_k}$, be subsets such that the elements of the set $\{e_{q_1},..., e_{q_l}\}$ are located these subsets, where $\{\tau_1,..., \tau_k\} \in \{\alpha_1,..., \alpha_n\}$. In addition,

$$(\{M_{i_1}^{\bar{\tau}_1},..., M_{i_k}^{\bar{\tau}_k}\} \subseteq M^1) \& (M_{i_1}^{\bar{\tau}_1},..., M_{i_k}^{\bar{\tau}_k} \in c_nS).$$

Let for some $\{\tau_1,..., \tau_k\} \in \{\alpha_1,..., \alpha_n\}$, all elements of the set $\{e_{q_1},..., e_{q_l}\}$ be located in the subsets $M_{i_1}^{\bar{\tau}_1},..., M_{i_k}^{\bar{\tau}_k}$, in addition,

$$(\{M_{i_1}^{\bar{\tau}_1},..., M_{i_k}^{\bar{\tau}_k}\} \subseteq M^1) \& (M_{i_1}^{\bar{\tau}_1},..., M_{i_k}^{\bar{\tau}_k} \in c_nS).$$

It is easy to see that for any subset $M_{i_j}^{\bar{\tau}_j} \in \{M_{i_1}^{\bar{\tau}_1},..., M_{i_k}^{\bar{\tau}_k}\}$ the conditions of Theorem 5.3 are satisfied, which means that the subset $M_{i_j}^{\tau_j}$ is sM^0-replaceable. Without limiting the generality of the reasoning, we can assume that $e_{j_1} \in M_{i_1}^{\bar{\tau}_1}$. So, as a result of sM^0-replacement of the subset $M_{i_j}^{\tau_j}$, we obtain the special decomposition $d_nS^0(e_{j_1}) \neq \emptyset$.

Since the transition from the decomposition d_nS to the decomposition $d_nS^0(e_{j_1})$ is an I-transformation, then according to Lemma 1.5 in [13], there exists a special covering for S under the decomposition $d_nS^0(e_{j_1})$. If the element e_{j_2} has moved to the 0-domain as a result of an sM^0-replacement associated with the element e_{j_1}, then according to Definition 5.4,

$d_nS^0(e_{j_1}) = d_nS^0(e_{j_1,}e_{j_2})$ and $d_nS^0(e_{j_1}, e_{j_2}) \neq \emptyset$.

If e_{j_2} has not moved to the 0-domain, then by the same reasoning as in the case of the element e_{j_1}, there exists a subset $M_{i_s}^{\bar{\tau}_s}$ such, that $(M_{i_s}^{\bar{\tau}_s} \in c_nS) \& (e_{j_2} \in M_{i_s}^{\bar{\tau}_s})$.

By Theorem 5.3, the subset $M_{i_s}^{\tau_s}$ is sM^0-replaceable. Since the replacement is associated with the element e_{j_2}, then performing the M^α-replacement of the subset $M_{i_s}^{\tau_s}$, we obtain:

1) $d_nS^0(e_{j_1}, e_{j_2}) \neq \emptyset$,
2) $d_nS^0(e_{j_1}, e_{j_2})$ is a special decomposition,
3) there is a special covering for the set S under the decomposition $d_nS^0(e_{j_1}, e_{j_2})$.

Repeating the similar procedure, no more than l times, we obtain:

1) $d_nS^0(e_{j_1},..., e_{j_l}) \neq \emptyset$,
2) $d_nS^0(e_{j_1},..., e_{j_l})$ is a special decomposition,
3) there is a special covering for the set S under the decomposition $d_nS^0(e_{j_1},..., e_{j_l})$.

It is obvious that when forming the decomposition $d_nS^0(e_{j_1},..., e_{j_l})$ all elements of the set $\{e_{j_1},..., e_{j_l}\}$ move into its 0-domain. At the same time, the 0-domain of the corresponding decomposition does not lose elements at any stage. Therefore, the 0-domain of the decomposition $d_nS^0(e_{j_1},..., e_{j_l})$ is an sM^0-covering for the set S.

(ii) Now let $\{e_{j_1},..., e_{j_k}\}$, where $k \leq l$, be an ordered set obtained by an arbitrary permutation of arbitrary elements of the set $\{e_{q_1},..., e_{q_l}\}$, such that that $d_nS^0(e_{j_1},..., e_{j_k}) = \emptyset$.

Recall, that $d_nS^0(e_{j_1},..., e_{j_k}) = \emptyset$ in the following cases:

(ii.1) $d_nS^0(e_{j_1}) = \emptyset$. This means, that under the decomposition d_nS there does not exist sM^0-replaceable subset, associated with e_{j_1} so, there does not exist a special covering for S.

(ii.2) for some element e_{j_r} $(1 \leq r < k)$,

$d_nS^0(e_{j_1},..., e_{j_r}) \neq \emptyset$ & $d_nS^0(e_{j_1},..., e_{j_r}, e_{j_{r+1}}) = \emptyset$.

This means that under the decomposition $d_nS^0(e_{j_1},..., e_{j_r})$, there does not exist an sM^0-replaceable subset associated with the element $e_{j_{r+1}}$. But then, by Corollary 4.2.1, there cannot exist a special covering for the set S under the decomposition $d_nS^0(e_{j_1},..., e_{j_r})$. Since the decomposition $d_nS^0(e_{j_1},..., e_{j_r})$ is obtained as a result of an I-transformation of the decomposition d_nS, then by Lemma 1.5 in [13], the decomposition d_nS does not allow a special covering for the set S. ∇

6. Applications

Let $S(f) = \{c_1,..., c_m\}$ be the set of clauses the Boolean function $f(x_1,..., x_n)$, and let

$$d_n\mathrm{S}(f)) = \{(\mathrm{F}_1^0,\ \mathrm{F}_1^1),...,\ (\mathrm{F}_n^0,\ \mathrm{F}_n^1)\}$$

be special decomposition of the set $\mathrm{S}(f)$.

For a subset $S_1(f) \subseteq \mathrm{S}(f)$ and $S_1(f) \neq \emptyset$, we say that $S_1(f)$ is a satisfiable subset if there exists a Boolean assignment tuple $(\alpha_1,...,\ \alpha_n)$, such that every clause included in $S_1(f)$ evaluates to 1 when assigning values $\alpha_1,...,\ \alpha_n$ to variables $x_1,...,\ x_n$.

Theorem 6.1. Let $(x_1,...,\ x_n)$ be a function represented in conjunctive normal form with m clauses, and let $d_nS(f)$ be a special decomposition of the set $\mathrm{S}(f)$.

Then, as a result of any I-transformation of the decomposition $d_n\mathrm{S}(f)$, the set of clauses $\mathrm{S}(f)$ is divided into two satisfiable subsets.

Proof. Let for some Boolean tuple $(\sigma_1,...,\ \sigma_n)$, the set

$$(r_1,...,\ r_p)\mathrm{I}\ (d_n\mathrm{S}(f)) = \{(\mathrm{F}_1^{\sigma_1},\ \mathrm{F}_1^{\bar{\sigma}_1}),...,\ (\mathrm{F}_n^{\sigma_n},\ \mathrm{F}_n^{\bar{\sigma}_n})\}$$

be an I-transformation of the decomposition $d_n\mathrm{S}(f)$. Then, $\sigma_1,...,\ \sigma_n$ will be as follows:

$\sigma_i = 0$, if $i \notin \{r_1,...,\ r_p\}$ and $\sigma_i = 1$, if $i \in \{r_1,...,\ r_p\}$.

Consider an arbitrary subset in the 0-domain of this decomposition $\mathrm{F}_k^{\sigma_k} \in \{\mathrm{F}_1^{\sigma_1},...,\ \mathrm{F}_n^{\sigma_n}\}$
and the clauses included in it. By definition

$\mathrm{F}_k^{\sigma_k} = \{c_j\ /\ c_j \in \mathrm{S}(f)$ and c_j contains the literal $x_k^{\sigma_k},\ j \in [1,\ m]\}$.

Obviously, if $x_k = \sigma_k$ then all clauses in the subset $\mathrm{F}_k^{\sigma_k}$ will take the value 1. So, if $x_1 = \sigma_1,...,\ x_n = \sigma_n$, then the set of clauses included in the 0-domain of this decomposition will be satisfiable.

It is easy to see that by the same reasoning as for the 0-domain, $(\bar{\sigma}_1,...,\ \bar{\sigma}_n)$ will be the satisfying tuple for the 1-domain. Obviously, if $S_0(f)$ and $S_1(f)$ denote the sets of clauses included in the 0-domain and 1-domain, respectively, then $\mathrm{S}(f) = S_0(f) \cup S_1(f)$. ∇

Remark 6.2. Recall that any Boolean function in conjunctive normal form generates a special decomposition of the set of its clauses, and any special decomposition of a set generates a Boolean function represented in conjunctive normal form. In sec. 6 in [13] is shown that by permuting the components of an ordered pair, new function is generated.

Let $d_n\mathrm{S}(f)$ be a special decomposition generated by the function $f(x_1,...,\ x_n)$. Consider an ordered pair $(\mathrm{F}_i^0,\ \mathrm{F}_i^1) \in d_n\mathrm{S}(f)$, assuming that

$\mathrm{F}_i^0 = \{c_{l_1},...,\ c_{l_p}\}$ and $\mathrm{F}_i^1 = \{c_{j_1},...,\ c_{j_q}\}$.

By the definition of these subsets,

1) the literal $\bar{\mathrm{x}}_i$ is included in all clauses of the set $\{c_{l_1},...,\ c_{l_p}\}$,
2) the literal x_i is included in all clauses of the set $\{c_{j_1},...,\ c_{j_q}\}$.
3) the literals $\bar{\mathrm{x}}_i$ and x_i are not included in any other clauses.

When permuting the components of the ordered pair $(\mathrm{F}_i^0,\ \mathrm{F}_i^1)$, we obtain the special decomposition $(i)I(d_nS(f))$, in which the i-th ordered pair has the form $(\mathrm{F}_i^1,\ \mathrm{F}_i^0)$. The remaining ordered pairs coincide with the corresponding ordered pairs of $d_nS(f)$.

$$(i)I(d_n\mathrm{S}(f)) = \{(\mathrm{F}_1^0,\ \mathrm{F}_1^1),...,\ (\mathrm{F}_i^1,\ \mathrm{F}_i^0),...,\ (\mathrm{F}_n^0,\ \mathrm{F}_n^1)\}.$$

We form a function denoted as $(i)h^I(x_1,...,\ x_n)$ in accordance with Section 3 based on the special decomposition $(i)I(d_n\mathrm{S}(f))$. Recall that if

$$d_n\mathrm{S} = \{(M_1^0,\ M_1^1),...,\ (M_n^0,\ M_n^1)\}$$

is a special decomposition of a set $\mathrm{S} = \{e_1,...,\ e_m\}$, then for any element $e_j \in \{e_1,...,\ e_m\}$, the clause $c(e_j)$ will formed by the following set of literals, $\{x_i^{\alpha_i}\ /\ e_j \in M_i^{\alpha_i}\}$. Respectively,

$$(i)h^I(x_1,...,\ x_n) = \wedge_{j=1}^m c(e_j).$$

The ordered sets $d_nS(f))$ and $(i)I(d_n\mathrm{S}(f))$ differ only in the i-th ordered pair. That is,

$\mathrm{F}_i^0 = \{c_{l_1},...,\ c_{l_p}\}$ is located in the 1-domain,

$\mathrm{F}_i^1 = \{c_{j_1},...,\ c_{j_q}\}$ is located in the 0-domain, of the resulting decomposition.

Let's denote by $c_1',...,\ c_m'$ the clauses of the function $h^I(x_1,...,\ x_n)$. That is,

$$S(h^I) = \{c_1',...,\ c_m'\}.$$

Since $h^I(x_1,...,\ x_n)$ is generated by the decomposition $(i)I(d_nS(f))$, then:

1) for any $c_{j_r} \in \{c_{j_1},...,\ c_{j_q}\}$, the corresponding new clause c_{j_r}' will contain the literal $\bar{\mathrm{x}}_i$
2) for any $c_{l_r} \in \{c_{l_1},...,\ c_{l_p}\}$, the corresponding new clause c_{l_r}' will contain the literal x_i.

Let $d_n\mathrm{S}(h^I) = \{(\mathrm{H}_1^0,\ \mathrm{H}_1^1),...,\ (\mathrm{H}_n^0,\ \mathrm{H}_n^1)\}$ denotes the special decomposition $(i)I(d_n\mathrm{S}(f))$. According to the formation procedure of the function $h^I(x_1,...,\ x_n)$, we obtain:

1) the literal $\bar{\mathrm{x}}_i$ is included in any of the clauses included in H_i^0,
2) the literal x_i is included in any of the clauses included in H_i^1.

At the same time, the clauses of the function $(i)h^I(x_1,...,\ x_n)$, not included in the subsets H_i^0 or H_i^1, coincide with the corresponding clauses of the function $f(x_1,...,\ x_n)$. Thus, the function $h^I(x_1,...,\ x_n)$ is obtained by replacing the literal $\bar{\mathrm{x}}_i$ with the literal x_i and the literal x_i with the literal $\bar{\mathrm{x}}_i$ in all clauses of the function $f(x_1,...,\ x_n)$ containing these literals.

We will say that the function $(i)h^I(x_1,...,\ x_n)$ is obtained by inverting the literals of the variable x_i in the function $f(x_1,...,\ x_n)$.

Similarly, the expression $(i_1,..., i_k)h^I(x_1,..., x_n)$ will mean the function generated by the decomposition $(i_1,..., i_k)I(d_n\mathrm{S}(f))$, for any $i_1,..., i_k \in [1, n]$.

Theorem 6.3. A function $f(x_1,..., x_n)$ represented in conjunctive normal form is satisfiable if and only if the function $(i_1,..., i_k)h^I(x_1,..., x_n)$ is satisfiable.

Proof. Let $d_nS(f)) = \{(\mathrm{F}_1^0, \mathrm{F}_1^1),..., (\mathrm{F}_n^0, \mathrm{F}_n^1)\}$ be a special decomposition of the set $S(f)$. Consider the special decomposition $(i_1,..., i_k)I(d_nS(f))$. By definition,

$(i_1,..., i_k)I(d_n\mathrm{S}(f))) = \{(\mathrm{F}_1^{\sigma_1}, \mathrm{F}_1^{\bar{\sigma}_1}),..., (\mathrm{F}_n^{\sigma_n}, \mathrm{F}_n^{\bar{\sigma}_n})\}$,

$(\sigma_i = 0, \text{if } i \notin \{i_1,..., i_k\} \text{ and } \sigma_i = 1, \text{if } i \in \{i_1,..., i_k\})$.

Suppose that the $f(x_1,..., x_n)$ is a satisfiable function, that is, there exists a Boolean tuple $(\alpha_1,..., \alpha_n)$ such that $f(\alpha_1,..., \alpha_n) = 1$. Using the arguments of Remark 6.2 regarding the inversion of literals, it is easy to see that the function $\{i_1,..., i_k\}h^I(x_1,..., x_n)$ will be satisfiable by the assignment tuple $(\beta_1,...,\beta_n)$ if we define this tuple as follows:

$\beta_i = \alpha_i$, if $i \notin \{i_1,..., i_k\}$ and $\beta_i = \bar{\alpha}_1$, if $i \in \{i_1,..., i_k\}$.

That is, $\{i_1,..., i_k\}h^I(\beta_1,...,\beta_n) = 1$.

Obviously, the opposite is also true.

If $\{i_1,..., i_k\}h^I(\beta_1,...,\beta_n) = 1$ for some Boolean assignment tuple $(\beta_1,...,\beta_n)$ then there is a tuple $(\alpha_1,..., \alpha_n)$ such that $f(\alpha_1,..., \alpha_n) = 1$. ∇

Definition 6.4. A conjunctive normal form of a Boolean function will be called a Proportional Conjunctive Normal Form, if each clause contains at least one negative literal or if each clause contains at least one positive literal.

Theorem 6.5. A Boolean function $f(x_1,..., x_n)$ represented in conjunctive normal form is satisfiable if and only if it can be transformed into a function in Proportional conjunctive normal form using literal inversions.

Proof. Let the ordered set $d_n\mathrm{S}(f)) = \{(\mathrm{F}_1^0, \mathrm{F}_1^1),..., (\mathrm{F}_n^0, \mathrm{F}_n^1)\}$ be a special decomposition of the set of clauses $\mathrm{S}(f)$ of the function $f(x_1,..., x_n)$.

One part of the theorem is obvious. Suppose that the function is transformed into a function represented in Proportional conjunctive normal form using literal inversions. Let it be the function $h^I(x_1,..., x_n)$. It is easy to notice, that a function in proportional conjunctive normal form is satisfiable, so by the Theorem 6.3, the function $f(x_1,..., x_n)$ is satisfiable.

The opposite: Let $f(x_1,..., x_n)$ be a satisfiable function. Then, there is a Boolean tuple $(\sigma_1,..., \sigma_n)$ such that $f(\sigma_1,..., \sigma_n) = 1$. Hence, according to Theorem 4.9, there is a special covering for the set $\mathrm{S}(f)$ under the decomposition $d_n\mathrm{S}(f)$. Let it be the set $c_n\mathrm{S} = \{\mathrm{F}_1^{\sigma_1},..., \mathrm{F}_n^{\sigma_n}\}$.

1) If the set c_nS is an sM^α-covering for the set $S(f)$, then all clauses are found in the α-domain of the corresponding decomposition, so by the arguments of Remark 6.2, all they contain a negative literal, if α=0, or all they contain a positive literal, if α=1.
2) Assume that $c_{q_1},..., c_{q_l}$ are clauses of the function f such that for some $\alpha \in \{0,1\}$,

$$\{c_{q_1},..., c_{q_l}\} = \mathrm{S}(f) \setminus M^\alpha.$$

Since there is a special covering for the set $\mathrm{S}(f)$ then by Theorem 4.4, $\{c_{q_1},..., c_{q_l}\}$ is a reachable set. Suppose that $\{(\mathrm{F}_{i_1}^0, \mathrm{F}_{i_1}^1),..., (\mathrm{F}_{i_k}^0, \mathrm{F}_{i_k}^1)\}$ is the set of replacement steps of the decomposition $d_n\mathrm{S}(f)$ that ensure the sM^α-reachability of all elements $\{c_{q_1},..., c_{q_l}\}$. After the permutation the components of all these ordered pairs we obtain the special decomposition $(i_1,..., i_k)I(d_n\mathrm{S}(f)))$. Obviously, as a result, all clauses of the set $\mathrm{S}(f)$ will be in the α-domain of the resulting decomposition.

Consider the function $(i_1,..., i_k)h^I(x_1,..., x_n)$ generated by the special decomposition.

$$(i_1,..., i_k)I(d_n\mathrm{S}(f))).$$

Since all clauses of this function are included in the α-domain of the decomposition.

$$(i_1,..., i_k)I(d_n\mathrm{S}(f))),$$

then all they contain a negative literal, if $\alpha = 0$, or all they contain a positive literal, if $\alpha = 1$.

This means that, the function $(i_1,..., i_k)h^I(x_1,..., x_n)$ is represented in proportional conjunctive normal form. At the other hand, the function $(i_1,..., i_k)h^I(x_1,..., x_n)$ is obtained by inverting the literals of the variable $x_{i_1},..., x_{i_k}$ in the function $f(x_1,..., x_n)$.

Thus, if $f(x_1,..., x_n)$ is a satisfiable function, then it can be transformed into a function represented in proportional conjunctive normal form using literal inversions. ∇

7. Instead of an Afterword

The importance of the result of Theorem 5.5 is that for any $\alpha \in \{0, 1\}$, having a set of elements $\{e_{q_1},..., e_{q_l}\}$ not included in the α-domain of the given decomposition, we can search for the sM^α-reachability of each of these elements, considering them in any order.

The importance of the result of Theorem 6.5 is that the theorem describes one of the necessary natures of the class of satisfiable Boolean functions. That is:

The function is included in this set if and only if it can be transformed into a function in Proportional conjunctive normal form using literal inversions.

At the same time, some important questions associated with these results remain open: How to find ordered pairs that ensure the reachability of a given element, or how conclude that such ordered pairs do not exist. Also, how find the required literals for further inversion.

These questions will certainly determine the direction of research in the next article.

Abbreviations

CNF	Conjunctive Normal Form
satCNF	Satisfiability of Conjunctive Normal Form

Author Contributions

Stepan Margaryan is the sole author. The author read and approved the final manuscript.

Conflicts of Interest

The author declares no conflicts of interest.

References

[1] Allender, E., Bauland, M., Immerman, N., Schnoor, H., Vollmer, H. The complexity of satisfiability problems: Refining Schaefer's theorem. J. Comput. Syst. Sci. 75(4), 245–254 (2009).

[2] Sanjeev Arora, Boaz Barak, Computational Complexity: A Modern Approach 2007.

[3] Armin Biere, Marijn Heule, Hans van Maaren and Toby Walsh, *Handbook of Satisfiability*, IOS Press, 2008.

[4] Cook, S. A., The complexity of theorem - proving procedures. In Conference Record *of Third Annual ACM Symposium on Theory of Computing (*1971), 151–158.

[5] Yves Crama, Peter L. Hammer, Boolean Functions. Theory, Algorithms, and Applications, 2011.

[6] Peter G ́acs, Boston University and L ́aszl ́o Lov ́asz, Yale University, *Complexity of Algorithms Lecture Notes*, Spring 1999.

[7] Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. Theoretical Computer Science 1(3): 237–267, 1976. https://doi.org/10.1016/0304-3975(76)90059-1

[8] S. A. Gizunov and V. A. Nosov, Recognition complexity for Sheffer classes, Mosc. Univ. Math. Bull. 51, no. 4, pp. 86–88, 1996. UDC 519.716.

[9] A. Horn, On sentences which are true of direct unions of algebras. Journal of Symbolic Logic. 16(1): 14–21, 1951. https://doi.org/10.2307/2268661 JSTOR 2268661. S2CID 42534337.

[10] Thomas Jech, *Set Theory*, The Third Millennium Edition, 2003.

[11] Karp, R. M., Reducibility among combinatorial problems. In *Complexity of Computer Computations* (1972), R. E. Miller and J. W. Thatcher, Eds., Plenum Press, 85–103.

[12] (L. A. Levin, "Universal problems of full search", Probl. Inf. Transm., 9: 3 (1973), 265–266).

[13] Stepan Margaryan, Special Coverings of Sets and Boolean Functions, American Journal of Mathematical and Computer Modeling (Volume 10, Issue 3), 84-97. https://doi.org/10.11648/j.ajmcm.20251003.11

[14] J. Donald Monk, Mathematical Logic. Springer-Verlag, 1976.

[15] Daniel Pierre Bovet and Pierluigi Crescenzi. Introduction to the Theory of Complexity. Prentice-Hall, New York, 1993.

[16] Charles C. Pinter, *A Book of Set Theory*, 2014.

[17] E. A. Potseluevskaya, *Approximation of Boolean functions to Schaefer's classes*. Journal of Mathematical Sciences, June 2012, vol 183.

[18] Rogers, H., Jr.: Theory of Recursive Functions and Effective Computability. MIT Press, Cambridge (1987).

[19] Kenneth H. Rosen, *Discrete Mathematics and its Applications*, seventh edition, 2012.

[20] Thomas J. Schaefer, *The complexity of Satisfiability Problems*, 1978. Department of Mathematics University of California, Berkeley, California 94720.

[21] Edward R. Scheinerman, Mathematics: A Discrete Introduction, Third Edition 2013.

[22] Sipser Michael, Introduction to the Theory of Computation, 2012, Cengage Learning.