An Idea For Solving 3SAT

Delavar Qasemi

eng.delavarqasemi@gmail.com

Abstract : In this paper, we present an algorithm that can be used to convert φ into a solvable form. This is done using binary combinations and the relations between them. Binary combinations are all 2-Clauses that can be obtained from n literal of φ and with their help we convert φ into a readable form (φ_2). φ_1 and φ_2 are equivalent (φ_1 is the same as φ_2), so we can solve φ_2 instead of solving φ_1. The algorithm performs the transform operation in 4 steps , and in each step it converts φ_i to φ_{i+1}. φ_i s and φ are equivalent. It can be proven that in the worst case φ_5 is solvable. The time and space order of this algorithm is $O(n^{96})$.

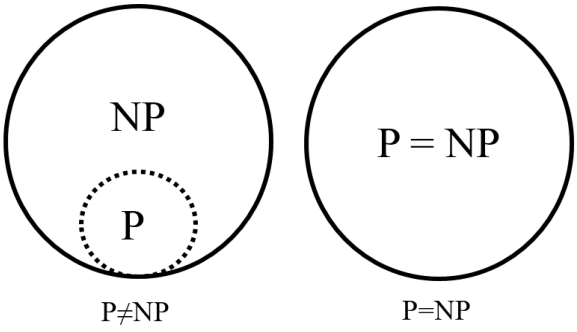
Introduction and Definitions

- **Literal :** Literals are variables and inputs of φ. Each literal is expressed x or $\bar{x}$. Literals take value 1 or 0 .
- **Clause :** Each clause is a compound of several literals or their opposites, for example $(x+p+\bar{q}+r)$ is a 4-Clause . If $x=0, p=0, q=1, r=0$, the $(x+p+\bar{q}+r)$ will be 0.
- **3CNF :** A Boolean expression is made up of And multiple 3-Clauses. The φ is a 3CNF made up of n literals and m clauses.
- **Binary combination :** Each binary combination is a 2-Clause labeled with a new variable. For example, $A_1=(x+\bar{z})$ is a binary combination.
- φ_i : This algorithm transform φ into a new 3CNF such as φ_i , $(i=1,2,3,4,5)$ at 4 steps. φ_1 is actually the φ as the starting point of the algorithm. In *i=1,* φ_1 transformed into φ_2 , in *i=2* φ_2 transformed into φ_3 , in *i=3* φ_3 transformed into φ_4 and finally in *i=4* φ_4 transformed into φ_5 .
- **Merge :** By merging, we can discover new clauses from the clauses of φ . For example, by merging $(x+y+p)$ and $(\mathrm{m}+\mathrm{n}+\bar{p})$, we can obtain $(\mathrm{m}+\mathrm{n}+x+y)$. Note that by discovering $(\mathrm{m}+\mathrm{n}+x+y)$, $(x+y+p)$ and $(\mathrm{m}+\mathrm{n}+\bar{p})$ cannot be eliminated, but $(\mathrm{m}+\mathrm{n}+x+y)$ is added to φ like the other clauses.
- **Expansion :** Any clause can be expanded into higher clauses, for example, any 3-Clause can be expanded into 6-Clause. Each 3-Clause is equivalent to $(n-3)(n-4)\in O(\mathrm{n}^2)$ 6-Clause. To convert a 3-Clause into 6-Clause, we simply add 3 literals to it in sequence.

For example, the 3-Clause $(a+b+c)$ can be expanded into 8 6-Clauses.

$$(a+b+c+p+q+r)(a+b+c+\bar{p}+q+r)$$
$$(a+b+c+p+q+\bar{r})(a+b+c+\bar{p}+q+\bar{r})$$
$$(a+b+c+p+\bar{q}+r)(a+b+c+\bar{p}+\bar{q}+r)$$
$$(a+b+c+p+\bar{q}+\bar{r})(a+b+c+\bar{p}+\bar{q}+\bar{r})$$

The 3SAT problem is one of the unsolved problems and no solution has been found yet. Scientists in this field describe these problems as PvsNP. If an algorithm can be found that can solve 3SAT in polynomial time, then P=NP, and if it can be proven that no such algorithm exists, then P!=NP.



PvsNP

Of course, algorithms have been designed that can work in some situations, but these algorithms are random and approximate and cannot provide a definitive answer.

φ has n literals. The literals are the inputs to the problem. Each literal can take the value 0 or 1. In general, literals can have 2^n different values. 2^n can sometimes be a very large number and its calculation is impossible. For example, for a small n like $n=64$, 2^{64} cannot be calculated and to find it the computer must be turned on for centuries. Computing 2^{64} is not possible even with the fastest systems and will not be possible in the future. This is why we cannot determine whether a problem is *SAT* or *UNSAT* by examining all the inputs.

Clauses express constraints and dos and don'ts on literals. For example, $(x+y+z)$ requires literals x, y, z not to be 0 at the same time. The more clauses in a problem, the more information can be obtained about the paths and relationships between literals. . New clauses can be discovered by methods such as merging, but the question is, what is the minimum number of clauses and of what length do we need to solve the problem? There is no answer for the value of k, and in the worst case, k is considered equal to n.

2-Clauses and Binary Combinations

2-Clauses are the conjunction of two literals or their opposites. For example, $(a+b)$ is a 2-Clause that states that a and b must not be 0 at the same time.2-clauses have an important property that other k-clauses do not have. It can be proven with certainty that merging two 2-Clauses always produces a 2-Clause. For example, merging $(a+b)$ and $(\bar{b}+c)$ yields $(a+c)$. This property only holds true for 2-Clauses. Merging 3-Clauses may result in a 2-, 3-, or 4-Clause.

We use this property of 2-Clauses. To begin with, it is enough to know that any 3-Clause like $(a+b+c)$ is the product of + two 2-Clauses like $(a+b)$ and $(b+c)$. Or a 4-Clause like $(a+b+c+d)$ is the product of + two 2-Clauses like $(a+b)$ and $(c+d)$.

Binary combinations are the basis of our algorithm. Binary combinations are the set of all 2-Clauses that can be obtained from n literals of φ. The number of binary combinations is $4n^2$. We labele each binary combination with a new variable like $(1 \le i \le 4n^2), A_i$. For example, A_1 is a binary combination $A_1 = \left(x+\bar{z}\right)$.

Rewriting clauses with new variables

We can rewrite the clauses of φ with new variables and arrive at equivalent clauses.

Lemma 1 :

Assuming the following variables

$A_1 = (x+y), A_2 = (x+z), A_3 = (\mathrm{y}+z), A_4 = (x+x), A_5 = (\mathrm{y}+y), A_6 = (\mathrm{z}+z)$

,we can express $(x+y+z)$ as follows.

$(A_1+A_2)(A_1+A_3)(A_2+A_3)(A_1+A_6)(A_2+A_5)(A_3+A_4)$

Proof :

According to truth table 1, $(x+y+z)$ and $(A_1+A_2)(A_1+A_3)(A_2+A_3)(A_1+A_6)(A_2+A_5)(A_3+A_4)$ are equivalent to each other.

x	y	z	$(x+y+z)$	$(A_1+A_2)(A_1+A_3)(A_2+A_3)(A_1+A_6)(A_2+A_5)(A_3+A_4)$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

truth table 1

Relationships between Binary Combination

We have already mentioned that a new 2-Clause can be derived from merge of other 2-Clauses. To show these relationships, we use the change of variable and new clauses.

Lemma 2 :

If the binary combinations $A_1 = (x+z), A_2 = (y+\bar{z})$ are , then the relations $(A_1 + A_2), (\overline{A_1} + \overline{A_2} + A_3), A_3 = (x+y)$ also hold.

Proof : Truth table 2 shows the truth of the relations.

x	y	z	A_1	A_2	$A_3 = (x+y)$	$(\overline{A_1} + \overline{A_2} + A_3)$	$(A_1 + A_2)$
0	*0*	*0*	*0*	*1*	*0*	*1*	*1*
0	*0*	*1*	*1*	*0*	*0*	*1*	*1*
0	*1*	*0*	*0*	*1*	*1*	*1*	*1*
0	*1*	*1*	*1*	*1*	*1*	*1*	*1*
1	*0*	*0*	*1*	*1*	*1*	*1*	*1*
1	*0*	*1*	*1*	*0*	*1*	*1*	*1*
1	*1*	*0*	*1*	*1*	*1*	*1*	*1*
1	*1*	*1*	*1*	*1*	*1*	*1*	*1*

truth table 2

With the use of the clauses $(A_1 + A_2), (\overline{A_1} + \overline{A_2} + A_3)$, we can show the relations between the binary combinations A_1, A_2, A_3, since A_1 contains z and A_2 contains $\bar{z}$, then $(A_1 + A_2)$ always holds. $(\overline{A_1} + \overline{A_2} + A_3)$ guarantees that if A_1, A_2 holds, then A_3 will also hold. $(\text{if } A_1 = 1 \text{and} A_2 = 1 \Rightarrow A_3 = 1)$

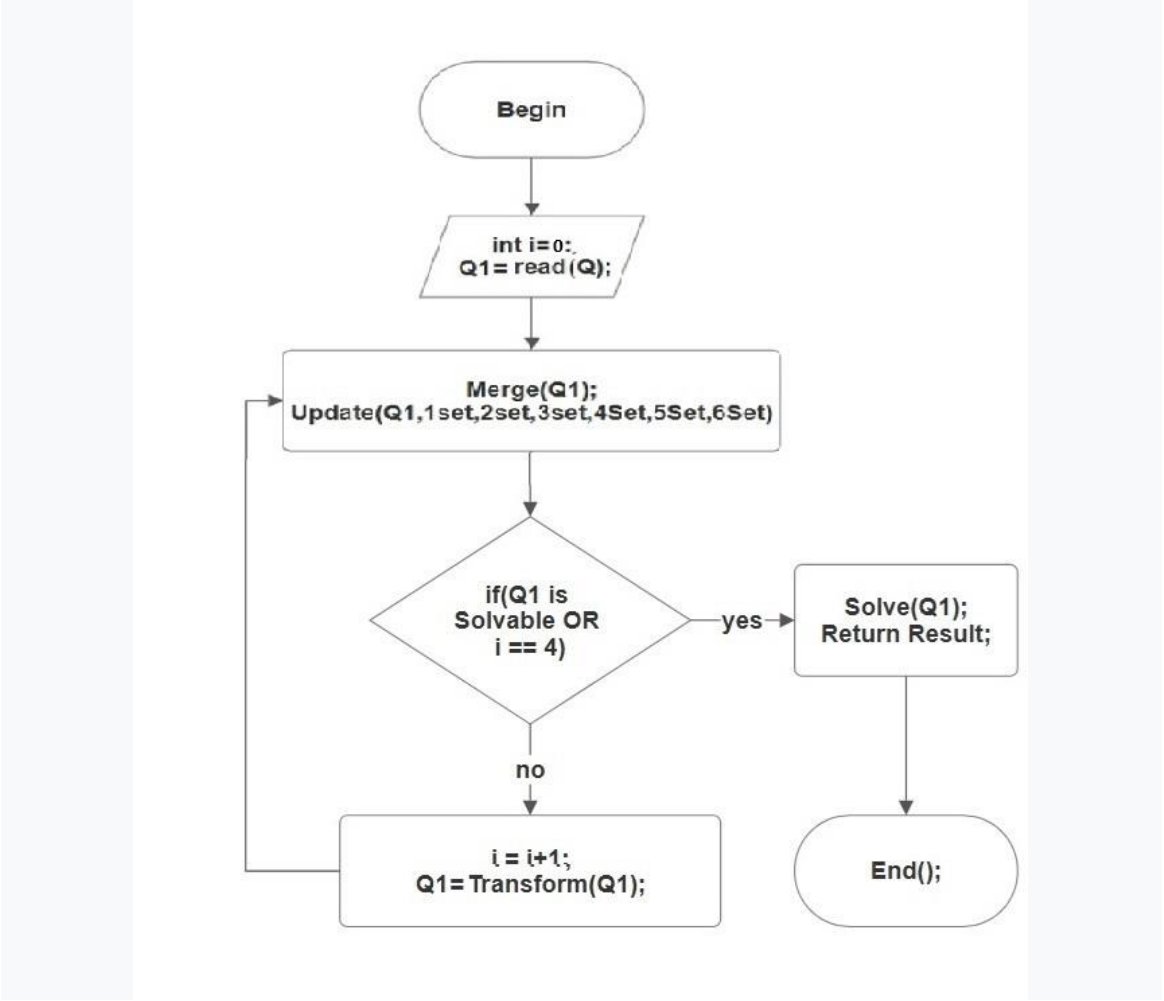
Transformation φ

First, we create a copy of φ named φ_1, then we obtain by merging 1-, 2-, 3-, 4-, 5-, and 6-Clauses of φ_1. If φ_1 is solvable, we solve it. Otherwise, we obtain all possible 2-Clauses from the literals of φ_1 and labele each one with a new variable like A_i. Then we rewrite all the 1-, 2-, 3-, 4-, 5-, and 6-Clauses of φ_1 with new variables as in Lemma 1. Also, as in Lemma 2, we express the relations between binary combinations as 2- and 3-Clauses. The question is, what help do these tasks and data give us?

The answer is that we can create a new 3CNF expression with the new data. To do this, we first create an empty 3CNF expression called φ_2 , treat the variables as literals of φ_2 . We add all the new clauses to φ_2 . After this, we have two separate 3CNF expressions, one φ_1 and one φ_2 . φ_1 and φ_2 are equivalent (Theorem 1). So we can solve φ_2 instead of φ_1 .

In the next step, we treat φ_2 like φ_1. As in the previous step, we obtain by merging the 1-, 2-, 3-, 4-, 5-, and 6-Clauses of φ_2. If φ_2 is solvable, we solve it, otherwise, like converting φ_1 to φ_2, we transform φ_2 to φ_3.

In the same way, we continue the transformation until we reach a solvable 3CNF expression. In Theorem 5, we will prove that in the worst case, every 3CNF expression will be solvable in the fourth transformation (φ_5).



Flow Chart

Theorems

Theorem 1 : φ and φ_i s are equivalent to each other.

Proof: To prove equivalence, it is enough to substitute the values of the variables in φ_i s to obtain φ again.

Theorem 2 : The number of 3-Clauses φ_2 is at least $\Omega(\mathrm{N}_2{}^{2.5})$.

Proof : Binary combinations like $(\alpha+\beta)$ can be related to combinations like $(\overline{\alpha}+...)$ and $(...+\overline{\beta})$. Instead of three dots, we can put $2(n-1)$ other literals, so each binary combination can create $O(n)$ new 2- and 3-Clause. . In total, we have $4n^2$ binary combinations from which $4n^2 * O(n\ ) = O(n^3)$ new 2- and 3-Clauses can be created. Also, the 3- and 4-Clauses of φ_1 in φ_2 are converted into 2-Clauses, so we can say that φ_2 has at least $\Omega(n^3)$ 2-Clauses. On the other hand, every 2-Clause is equivalent to $O(n^2)$ 3-Clause because any 2-Clause like $(A+B)$

can be expanded into $(A+B+\lambda),(A+B+\overline{\lambda})$. λ can be one of binary combinations whose number is equal to $4n^2$. So the minimum number of 3-Clauses is equal to.

$$\Omega(n^3)*4n^2=\Omega(n^5)=\Omega((n^2)^{5/2})\xrightarrow{N_2=n^2}\Omega(N_2{}^{2.5})$$

Theorem 3 : The number of 3-Clauses φ_3 is at least $\Omega(\mathrm{N}_3{}^{2.75})$.

Proof: : According to Theorem 2, φ_2 has at least $\Omega(\mathrm{N}_2{}^{2.5})$ 3-Clauses, so φ_2 has at least $\Omega(\mathrm{N}_2{}^{5.5})$ 6-Clauses (3-Clauses are expanded to 6-Clauses). The 6-Clauses of φ_2 are transformed into 3-Clauses in φ_3. Therefore, φ_3 has at least $\Omega(\mathrm{N}_2{}^{5.5})$ 3-Clauses. In other words,

$$\Omega(\mathrm{N}_2{}^{5.5})=\Omega((\mathrm{N}_2{}^2)^{5.5/2})\xrightarrow{N_3=N_2^2}\Omega(\mathrm{N}_3{}^{2.75})$$

Theorem 4 : The number of 4-Clauses φ_3 is at least $\Omega(\mathrm{N}_3{}^4)$.

Proof: : In this algorithm, the clauses of step $i+1$ can be obtained in various ways. For example, to find the 4-Clauses of φ_3, we can multiply the 3-Clauses of φ_2 two by two and convert the result into a 4-Clause.

$$(A+B+C),(K+P+S),\{\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{K},\mathrm{P},\mathrm{S}\}\in\varphi_2\_literal$$

$$|\varphi_2\_\mathrm{literals}|=\mathrm{N}_2,|\varphi_2\_clauses|=M_2$$

$$\{H_1,H_2,H_3,H_4\}\in\varphi_3\_literals$$

$$(A+B+C)(K+P+S)=(AK+AP+AS+BK+BP+BS+CK+CP+CS)$$

$$=(A(K+P+S)+B(K+P+S)+\mathrm{CK}+\mathrm{CP}+\mathrm{CS})=((\mathrm{A}+\mathrm{B})(\mathrm{K}+\mathrm{P}+\mathrm{S})+\mathrm{CK}+\mathrm{CP}+\mathrm{CS})$$

$$=(A+B+\overset{(1)}{CK}+CP+CS)(K+P+S+\overset{(2)}{CK}+CP+CS)$$

Part (2) is the same as $(K+P+S)$ and there is no need to rewrite it and it can be deleted. Therefore

$$(A+B+C)(K+P+S)=(A+B+CK+CP+CS)$$

The above result can be converted to a φ_3 4-Clause. The binary combinations of φ_2 are literals of φ_3. So φ_3 has literals like the following.

$$(A+B)=H_1,$$

$$CK=\overline{(\overline{C}+\overline{K})}=\overline{H_2},$$

$$CP=\overline{(\overline{C}+\overline{P})}=\overline{H_3},$$

$$CS=\overline{(\overline{C}+\overline{S})}=\overline{H_4}$$

$$\{H_1,H_2,H_3,H_4\}\in\varphi_3\_literals$$

Therefore

$$(A+B+C)(K+P+S)=(H_1+\overline{H}_2+\overline{H}_3+\overline{H}_4)$$

Note that φ_2, has M_2 3-Clause. According to the counting principle, we can obtain $M_2{}^2$ 4-Clauses of φ_3 by multiplying the φ_2 clauses.

In φ_3, a binary combination like $H_2 = (\overline{C} + \overline{K})$ makes a 2-Clause like $(H_2 + H_i)$ with all binary combinations containing *C* or *K*. The number of such 2-Clauses is N_2 because H_i can be any literals of φ_2. Clause $(H_2 + H_i)$ can be merged with clause $(H_1 + \overline{H}_2 + \overline{H}_3 + \overline{H}_4)$ to create N_2 clause like $(H_1 + H_i + \overline{H}_3 + \overline{H}_4)$. This result also holds for H_3 and H_4, so in general, by multiplying and merging, we can obtain $M_2{}^2 N_2{}^3$ 4-Clauses of φ_3.

$M_2{}^2 * N_2 * N_2 * N_2 = M_2{}^2 N_2{}^3$

According to Theorem 2, the number of 2-Clauses of φ_2 is at least $\Omega(\mathrm{N}_2{}^{2.5})$, so we have

$M_2{}^2 N_2{}^3 = (\Omega(\mathrm{N}_2{}^{2.5})^2) N_2{}^3 = \Omega(\mathrm{N}_2{}^8) \xrightarrow{N_3 = N_2{}^2} \Omega(\mathrm{N}_3{}^4)$

So in general, by multiplying and merging, we can obtain $\Omega(\mathrm{N}_3{}^4)$ φ_3 4-Clauses .

Theorem 5 : To solve φ , we simply transform φ step by step according to the algorithm. In the worst case, in step 4 (φ_5), we can definitely determine whether φ is *SAT* or *UNSAT*.

Proof: According to Theorem 4, φ_3 has $\Omega(\mathrm{N}_3{}^4)$ 4-Clauses. The 4-Clauses of φ_3 are converted into 2-Clauses in step φ_4, and the 2-Clauses of φ_4 are converted into 1-Clauses in step φ_5. In fact, the 4-Clauses of φ_3 are the literals of φ_5.

So we can say that the number of 2-Clauses of φ_4 is at least $\Omega(\mathrm{N}_4{}^2)$.

$\Omega(\mathrm{N}_3{}^4) = \Omega((\mathrm{N}_3{}^2)^{4/2}) = \Omega((\mathrm{N}_3{}^2)^2) \xrightarrow{N_4 = N_3{}^2} \Omega(\mathrm{N}_4{}^2)$

So the number of 1-Clauses φ_5 is equal to $\Omega(\mathrm{N}_5)$.

$\Omega(\mathrm{N}_4{}^2) \xrightarrow{N_5 = N_4{}^2} \Omega(\mathrm{N}_5)$

According to the above result, all literals of φ_5 are discovered, so in φ_5 all literals of the answer path are obtained as 1-Clauses. So if φ is *SAT*, its answer path will definitely be obtained in φ_5, but if φ is *UNSAT*, it will definitely be null until φ_5.

Conclusion

We conclude that the idea of changing the variable is a new approach to solving the 3SAT problem. Using binary combinations as new variables can be useful because we can express the relationships between them and use their relationships to solve φ . As we have shown, in step 4, it is determined whether φ_5 is *SAT* or *UNSAT*. The largest space required is 6-Clauses of φ_5. Therefore, the 3SAT problem can be solved in space and time complexity $O(\mathrm{N}_5^6) = O((\mathrm{n}^{16})^6) = O(\mathrm{n}^{96})$.