ISSN: 2710-0588 DOI: 10.29556/FSE.202107_1(4).0003

Three Propositions of the Milner-Welsh Theorem

Hongmei Pei[1,*], Meili Zhang[1]

[1]Department of Basic, Dalian Naval Academy, Dalian, China.

*Corresponding Author

Abstract

Combinatorial method is a very effective method to study Karp conjecture. The combinatorial method was first used to prove that many graph properties are elusive. An important combinatorial technique developed during this period was summarized by Milner and Welsh as the Milner-Welsh Theorem, which has an important application in determining the complexity of decision trees of graphic properties. In this paper, we give two important propositions about the Milner-Welsh theorem, and use these propositions to judge the elusive of the connectivity of graphs.

Keywords

Complex; False-assignments; Boolean Function; Elusive; Decision Tree Complexity.

1. Introduction

It is well known that graph theory has extensive applications in many areas. An important thing concerning to application is to treat graph with computer. Usually, a graph can be installed into a computer by encoding the entries of the triangular part of its adjacency matrix. One problem arising naturally is: Can we detect whether a graph G has some specific property without decoding all the entries of the upper triangular part of its adjacency matrix? That is, given a graph property P on n vertices, must it take $n(n-1)/2$ queries in the worst case to determine whether a graph G is in P? People guess the answer is yes when P is nontrivial and monotone. This is the well-known Karp conjecture which is still open and becomes a well-known difficult problem in computational complexity theory.

2. Preliminary

Definition 2.1 [1] A Boolean function is a function whose variable values and function value all are in {0,1}.

There are three main operations in Boolean function, which are $x \vee y$, $x \wedge y$ and $\bar{x}$. We call $\vee$as disjunction, $\wedge$as conjunction, and $\bar{x}$ as negation operation, please refer Table 1 for their definitions. Anyone of Boolean function can be expressed by three kinds of operations, which are disjunction, conjunction and negation operation. For example, Boolean function $f(x,y)$ is: $f(x,y) = (\bar{x} \wedge y) \vee (x \wedge \bar{y})$. For simplicity, we mark $x \vee y$ and $x \wedge y$ as $x + y$ and xy, thus the Boolean function that listed above can be expressed as $f(x,y) = \bar{x}y + x\bar{y}$.

Table 1. Main Operation of Boolean Function

x	y	$x+y$	xy	$\bar{x}$
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0

ISSN: 2710-0588 DOI: 10.29556/FSE.202107_1(4).0003

An assignment for a Boolean function is a mapping from its variables to {0,1}, each variable gets one value from the assignment. For n variables function $f(x_1, x_2, \cdots, x_n)$, the number of its assignment is 2^n.

Definition 2.2 [1] An assignments that makes $f(x_1, x_2, \cdots, x_n) = 1$ we call it a true-assignment, and an assignment that makes $f(x_1, x_2, \cdots, x_n) = 0$ we call it a false-assignment.

Besides the function expression, Boolean function can also be expressed by using binary tree which have special labels. A decision tree of a Boolean function, is a rooted binary tree, whose non-leaf vertices are labeled by its variables, and leaves are labeled by 0 and 1. On every path that from root to leaf (called root leaf path for short), a variable appears no more than once, and, the two edges that from every inner vertex to its children also be labeled by 0 and 1 respectively. Give a group of variable assignments, and the corresponding function value can be calculated as below rules: start from root, and investigate the vertex. If its label is variable , while $x_i = 0$, then go down along the side labeled 0; While $x_i = 1$, then go down along the side labeled 1 until to a leaf, the label of the leaf is function value. As show in Figure 1, One of the decision trees of Boolean function $f(x_{12}, x_{13}, x_{23}) = x_{12}x_{13} + x_{12}x_{23} + x_{13}x_{23}$.

Definition 2.3 [1] A decision tree of a Boolean function, is a rooted binary tree, whose non-leaf vertices are labeled by its variables, and leaves are labeled by 0 and 1.

Definition 2.4 [1] The depth of a decision tree is the maximum length of all paths from root to leaves.

For Boolean function $f(x_1, x_2, \cdots, x_n)$ whose variable number is n, it has limited numbers of decision trees, the number of them is $C_n^1 (C_{n-1}^1)^2 (C_{n-2}^1)^4 \cdots (C_2^1)^{2^{n-2}}$ all together.

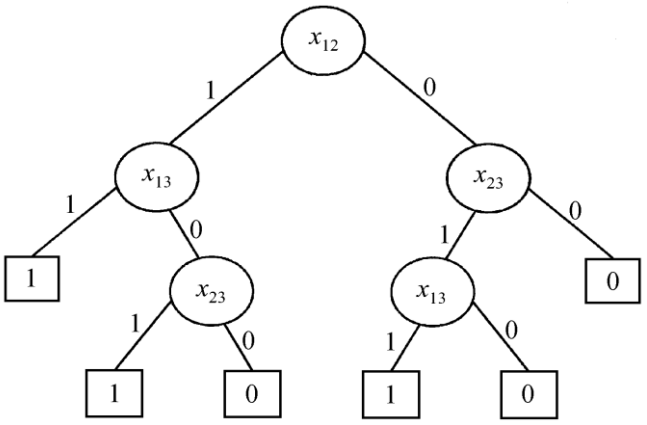

Figure 1. One of the decision trees of Boolean function $f(x_{12}, x_{13}, x_{23}) = x_{12}x_{13} + x_{12}x_{23} + x_{13}x_{23}$

Definition 2.4 [1] The decision tree complexity of f is the minimum depth of all decision trees for computing f, denoting by $D(f)$.

Definition 2.5 [1] A Boolean function f is said to be elusive if $D(f) = n$.

Definition 2.6 [1] A graph property is monotone decreasing if and only if it satisfies the following conditions: if a graph has the property, then all its subgraphs have the property; Conversely, a graph property is monotone increasing if and only if it satisfies the following condition: a graph has this property if its subgraphs have this property.

Definition 2.7 [1] Simple Strategy means: if the mark on the edge of the decision tree is always 0 on the left and 1 on the right, then the simple strategy can be vividly described as: keep going left from the tree root down, unless going left meets leaves marked 0, then it will go right once.

From the definition of the Simple Strategy, we can know that the leaf markers of root and leaf paths found by the Simple Strategy method are all labeled by 1.

ISSN: 2710-0588 DOI: 10.29556/FSE.202107_1(4).0003

Milner-Welsh Theorem [1] A graph property of nonempty monotone decreasing is elusive if it satisfies the following conditions: for any graph G with the property and any edge e of the graph G, $\boldsymbol{e}'$can always be found, such that $(G-e)\cup e'$ still retains the property.

The Milner-Welsh theorem is given on the graph property of monotone decreasing, and it is a special case of Theorem 3.1 below.

3. Main Results

First let's introduce two notations. Let y be a set of assignments to the variable x of some Boolean function. $Z(y)$ is the set of variables assigned to 0, and B is the set of variables assigned to 1.

Theorem 3.1 A sufficient condition that the decision tree of a Boolean function can always find root and leaf paths containing all variables by the Simple Strategy is that: for any set of true assignment y and variable $x_i \in Z(y)$, there exists another set of true assignment z and variable $x_j \in U(y)$, such that $\big(U(y)-\{x_j\}\big)\cup\{x_i\}\supset U(z)$. This condition is also necessary if the Boolean function is monotonically increasing.

Proof: If a root-leaf path is found using a simple strategy, its leaf marker must be 1. Let y be the true assignment consistent with the root-leaf path and make the assignment of variables that do not appear in the root-leaf path 0. That is to say, for any variable x_i that does not appear on the root leaf path, there is $x_i \in Z(y)$. By the condition, there is a variable $x_j \in U(y)$ and another set of true assignment z, such that $\big(U(y)-\{x_j\}\big)\cup\{x_i\}\supset U(z)$. Further, let's say that x_j is the first variable in $U(y)\backslash U(z)$ that appears on the above root leaf path. This means that before x_j, the assignment of the variable on the root leaf is the same as z. Notice that z is true assignment. In this way, when x_j is assigned with the value 1, even if x_j is assigned with the value 0, the function value will not be 0. This contradicts the definition of a simple strategy. So that proves the sufficiency of the condition.

To prove the necessity of the condition, let a set of true assignment y and a variable $x_i \in Z(y)$ exist for some monotone increasing Boolean function, such that no true assignment z satisfies $\big(U(y)-\{x_j\}\big)\cup\{x_i\}\supset U(z)$ for any variable $x_j \in U(y)$. This means that if variables other than $\big(U(y)-\{x_j\}\big)\cup\{x_i\}\supset U(z)$ are 0, the value of the function will be 0. Now, we construct a decision tree whose first layers are marked by variables in $Z(y)-\{x_i\}$, and each subsequent layer is marked by variables in $U(y)$. When using a simple strategy to find the root path, since y is the true assignment, the variable in $Z(y)-\{x_i\}$ must be assigned to 0. After that, every variable in $U(y)$ is encountered, and since assigning 0 will result in the function being 0, it must assign 1. When all the variables in $U(y)$ are assigned 1, the function is monotonically increasing, so the value of the function must be 1, that is, the road ends. However, the variable x_i has not yet appeared on the road, which proves the necessity of the condition.

Here, we conduct in-depth and detailed research on Milner-Welsh Theorem and Theorem 3.1. Based on the proving process of the two theorems, this paper can conclude the following two propositions:

Propositions 3.1 The variable x_j must come after the variable x_i in the root path of all variables found by the Simple Strategy (here the variables x_i and x_j have the same referent meaning as $x_i$and x_j in Theorem 2.1). Before and after means that x_i is closer to the root than x_j).

Proof: Converse proof: if not, if x_j appears before x_i, then assigning x_j to 0 will not result in the function value being 0, which is inconsistent with the definition of Simple Strategy, and thus the assumption is not valid. The variable x_j must come after the variable x_i.

Propositions 3.2 For true assignment z, there must be $x_i \in U(z)$.

ISSN: 2710-0588 DOI: 10.29556/FSE.202107_1(4).0003

Proof: Proof by contradiction. Assume $x_i \notin U(z)$, then $\left(U(y) - \{x_j\}\right) \supset U(z)$. Thus, in the true assignment Y, even if the variable x_j is assigned to 0, the assignment obtained is still the true assignment. So on the root path found with the simple strategy, even if you assign x_j to zero, it will not make the function zero, which contradicts the definition of the simple strategy. So the hypothesis doesn't work. So the $x_i \in U(z)$.

From Propositions 3.1, we know that after arbitrary true assignment y and variable x_i are set, select x_j is to select the variable after x_i on the root-leaf path to verify whether the conditions required by the Simple Strategy are met. From Propositions 3.2, we can know that when another true assignment z is selected, the true assignment z containing x_i in $U(z)$ should be selected to verify whether the conditions required by the simple strategy are met. Knowing these two points, when we are verifying whether the Boolean function given satisfies the conditions required by the simple strategy, we can select the variable x_j and the true assignment Z purposefully for verification, so as to make the verification process more purposeful and then simplify the verification process. It can be seen that the two inferences proposed in this paper have strong practical value when using Theorem 2.1 to prove the elusive function.

4. Application

In this paper, we apply the Milner-Welsh Theorem to prove that the connectivity of graphs is elusive.

Proof: Let any connected graph G and any edge e_{ij} outside the graph, by the connectivity of graph G, there exists a path from vertex i to vertex j. In this path, take one edge e_{kh}, remove the edge e_{kh}, and connect the vertices i, j. Since there is a path between vertex k and vertex i, and a path between vertex h and vertex j, there is still a path from vertex h to vertex k in graph $(G - e_{kh}) \cup e_{ij}$. So the graph $(G - e_{kh}) \cup e_{ij}$ is connected. The Milner-Welsh Theorem shows that the connectivity of graphs is a elusive graph property.

5. Conclusion

This paper propose two important propositions about the Milner-Welsh theorem and use them to judge the paradoxical nature of the connectivity of graphs.

Acknowledgments

This research was financially supported by the Dalian Naval Academy for Basic Research.

References

[1] Du Ding-zhu, Decision Tree Theory [M]. Hunan Education Press 1998.

[2] Xie Zheng-long, Xu Qing-lin, Shi Fang-xia, A refinement algorithm of decision tree generation. Journal of Xianyang Normal University, 2006, 21(6): 31-33.

[3] Wang Pei-gen, Boole function and Boole polynomial, Journal of Capital Normal University, 2006, 27(5): 15-18, 21.