# Fast Convert OR-Decision Table to Decision Tree

Phaisarn Sutheebanjard
Graduate School of Information Technology
Siam University
Bangkok 10163, Thailand
mr.phaisarn@gmail.com

Wichian Premchaiswadi
Graduate School of Information Technology
Siam University
Bangkok 10163, Thailand
wichian@siam.edu

***Abstract*— Recently a new type of decision table has been proposed, OR-decision table. The idea of the OR-decision table is one in which any of the actions in the set may be performed in order to satisfy the corresponding condition, while the classical decision table requires the execution of all the actions in the set of actions (AND-decision table). Due to the characteristic of the OR-decision table, this paper proposes the fast OR-decision table conversion to a decision tree algorithm.**

***Keywords—decision table, decision table conversion, decision tree, connected component labeling***

## I. INTRODUCTION

A decision table is a tabular form for displaying decision logic. Decision tables, like flowcharts and if-then-else logic, associates conditions with actions to be performed. Decision tables are used in many data processing applications and various actions must be carried out in response to the outcomes of a set of condition tests. Decision tables specify what subset of the actions are appropriate for any combination of test condition outcomes.

A decision table that requires the execution of all the actions in the set of actions to be performs is called an *AND-decision table*. While the new type of decision table is one in which any of the actions in the set of actions may be performed, this type of decision table is called an *OR-decision table*. The OR-decision table was first introduced by Grana *et al* [1]. in 2010. They merge the equivalent actions of an AND-decision table into the OR-decision table to reduce the size of the decision table.

Grana *et al*. proposed two steps to convert an OR-decision table to a decision tree. First use the greedy algorithm to reduce the OR-decision table to single action decision table. Second use the dynamic programming method [2] to convert the single action decision table to a decision tree. The result of the method is an optimum decision tree, but the computation time for the dynamic programming method is exponential time [3] and requires an enormous amount of memory.

This paper proposes a fast convert OR-decision table to a decision tree. The proposed method uses less memory and computation time compared to dynamic programming. The result of the method is a nearly optimum decision tree.

## II. BACKGROUND

### A. Decision table

A decision table is a tabular form that presents a set of conditions and their corresponding actions. Examples of decision tables are provided in Fig. 1 and 2. The conditions are identified by $c_1$- $c_3$ and the actions are identified by $a_1$-$a_4$. Each row in the entry section is called a rule $r_1$-$r_8$, which is a condition and action pair. The conditions may only be true (mask as "1") or false (mask as "0") and the actions may only be true (mask as "1") or false (leave blank). This type of decision table is called a *limited entry decision table* (LEDT) [4].

Typically, a decision table that has only one action for a rule is called a *single action decision table*, as shown in Fig. 1. In a decision table, if the execute operation is applied to all actions marked with 1s in the rule, it is called an *AND-decision table*. Otherwise, if a set of actions in a rule are equivalent and the execute operation is applied to any of the actions, it is called an *OR-decision table* [1], as shown in Fig. 2.

| | | Condition | | | Action | | | |
|---|---|---|---|---|---|---|---|---|
| | | $c_1$ | $c_2$ | $c_3$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
| Rule | $r_1$ | 0 | 0 | 0 | | 1 | | |
| | $r_2$ | 0 | 0 | 1 | 1 | | | |
| | $r_3$ | 0 | 1 | 0 | | | 1 | |
| | $r_4$ | 0 | 1 | 1 | | | | 1 |
| | $r_5$ | 1 | 0 | 0 | | 1 | | |
| | $r_6$ | 1 | 0 | 1 | | | 1 | |
| | $r_7$ | 1 | 1 | 0 | 1 | | | |
| | $r_8$ | 1 | 1 | 1 | | | | 1 |

Figure 1. Single action decision table

| | | Condition | | | Action | | | |
|---|---|---|---|---|---|---|---|---|
| | | $c_1$ | $c_2$ | $c_3$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
| Rule | $r_1$ | 0 | 0 | 0 | | 1 | | |
| | $r_2$ | 0 | 0 | 1 | 1 | | | |
| | $r_3$ | 0 | 1 | 0 | | | 1 | 1 |
| | $r_4$ | 0 | 1 | 1 | | | | 1 |
| | $r_5$ | 1 | 0 | 0 | | 1 | | |
| | $r_6$ | 1 | 0 | 1 | | 1 | 1 | |
| | $r_7$ | 1 | 1 | 0 | 1 | | | |
| | $r_8$ | 1 | 1 | 1 | | | | 1 |

Figure 2. (AND/OR) decision table

### B. OR-decision table

OR-decision tables include the possibility to represent more than one (equivalent) action for each set of conditions. The OR-decision table was proposed by Grana *et al*. They defined an OR-decision table in which any of the action in the set may be performed in order to satisfy the corresponding condition. Fig. 3 shows the example of OR-decision table that proposed by [1].

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | x=p | x=q | x=r | x=s | x=p+r | x=r+s |
| 0 | - | - | - | - | 1 | | | | | | | |
| 1 | 0 | 0 | 0 | 0 | | 1 | | | | | | |
| 1 | 1 | 0 | 0 | 0 | | | 1 | | | | | |
| 1 | 0 | 1 | 0 | 0 | | | | 1 | | | | |
| 1 | 0 | 0 | 1 | 0 | | | | | 1 | | | |
| 1 | 0 | 0 | 0 | 1 | | | | | | 1 | | |
| 1 | 1 | 1 | 0 | 0 | | | 1 | 1 | | | | |
| 1 | 1 | 0 | 1 | 0 | | | | | | | 1 | |
| 1 | 1 | 0 | 0 | 1 | | | 1 | | | 1 | | |
| 1 | 0 | 1 | 1 | 0 | | | | 1 | 1 | | | |
| 1 | 0 | 1 | 0 | 1 | | | | 1 | | 1 | | |
| 1 | 0 | 0 | 1 | 1 | | | | | | | | 1 |
| 1 | 1 | 1 | 1 | 0 | | | 1 | 1 | 1 | | | |
| 1 | 1 | 1 | 0 | 1 | | | 1 | 1 | | 1 | | |
| 1 | 1 | 0 | 1 | 1 | | | | | | | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | | | | 1 | 1 | 1 | | |
| 1 | 1 | 1 | 1 | 1 | | | 1 | 1 | 1 | 1 | | |

Figure 3. OR-decision table

To produce an optimal decision tree from OR-decision tables, Grana *et al*. converted the OR-decision tables into a single action decision table by using a heuristic greedy procedure. Upon the transformation of the single action decision table into an optimal decision tree, they used the dynamic programming technique [2].

### *C. Dynamic Programming*

The concept of dynamic programming is the optimal solution that can be built from optimal sub-solutions. This is the case because the building of a decision sub-tree for each restriction is a separate problem that can be optimally solved independently of the others but sub-diagrams often overlap the resulting interaction, which destroys the independence of the sub-problems [3] as shown in figure 4.

## III. OR-DECISION TABLE CONVERSION ALGORITHM

This section describes the OR-decision table conversion algorithm to convert an OR-decision table into a decision tree with fast computation time. The conditions in the decision table can be either a complete decision table (every combination of outcomes appears as an elementary rule) or a compressed decision table. The proposed method is straightforward and requires less computation time.

There are four steps in the proposed algorithm. First, check the OR-decision table to see if it can be a leaf node or a condition node by counting the number of actions in each column. If the total number of action occurrences (‘1’) equals to the number of rows in the OR-decision table, then assign this action to be leaf node, otherwise this OR-decision table will be a condition node (non-leaf node).

Second, find the action that has the maximum number of occurrences in the OR-decision table. For each row in the same column, if the action equals ‘1’, then check the condition columns (*x*, *p*, *q*, *r*, *s*) to count the number of ‘-’ (n) and then add $2^n$ to the total number of occurrences. After completing this check on every (action) column in the table, compare the total number of occurrence in every (action) column. The action that has the maximum number of occurrence is chosen to perform the next step.

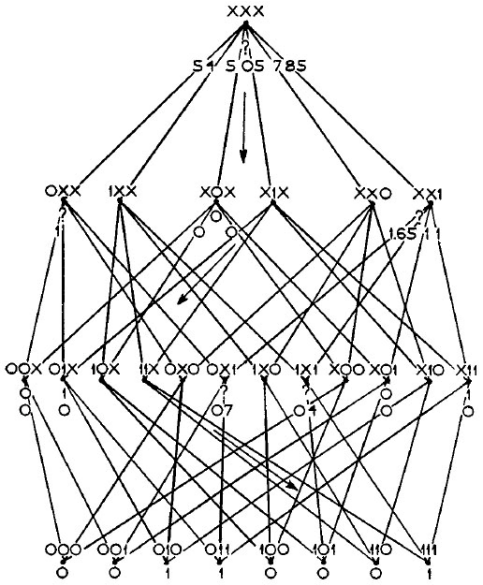

Figure 4. The dynamic programming lattice of 3 variables.

Third, find the maximum number of occurrence of the unique value of each condition. From the action that was selected in the previous step; consider only rows where the action equals ‘1’, count the number of ‘0’ and ‘1’ occurrences separately in each condition columns (*x*, *p*, *q*, *r*, *s*). Upon counting, if ‘-’ was found in the same row, count the number of ‘-’ (n) in that row. If the condition is ‘0’, then add $2^n$ to the total number of ‘0’ occurrences, otherwise if the condition is ‘1’, add $2^n$ to the total number of ‘1’ occurrences. After finishing this process, multiply each number of occurrences by its weight, the condition that has maximum value is selected to be the condition node.

Finally, separate the OR-decision table from the previous step into two sub-tables by grouping the rows that contain ‘0’ into one table and rows that contain ‘1’ into another table. Continue the process by using the two sub-tables, repeat the first step until every action is a leaf node. The proposed algorithm is shown in figure 5.

```
1.  Check the OR-decision table to see if it can be leaf node or
    condition node
  For col in length(Actions)
    FlagLeafNode = True
    For row in length(Table)
        If Actions [col]!='1' then
            FlagLeafNode = False
    If FlagLeafNode==True then
        Assign_Leaf_Node = Actions[col]
            Return Assign_Leaf_Node # leaf node,
  Goto step 2  # condition node

2.  Find the action that has maximum number of occurrence in
    OR-decision table
  For col in length(Actions)
    Count_actions[col] = 0
    For row in length(Table)
        If Table[row][col]=='1' then
            No_of_dash = count dash in Condition[row]
            Count_actions[col] += 2^no_of_dash
  SelectAction = Max(Count_actions)
```

3. Find the maximum number of occurrence of the unique value of each condition.

```
For col in length(Condition)
    Count_condition0[col] = 0
    Count_condition1[col] = 0
    For row in length(Table)
        If Table[row][SelectAction]==‘1’ then
            No_of_dash = count dash in Condition[row]
            If Table[row][col]==‘0’ then
                Count_condition0[col] += 2^no_of_dash
            Else If Table[row][col]==‘1’ then
                Count_condition1[col] += 2^no_of_dash
    Count_condition0[col] *= weight[col]
    Count_condition1[col] *= weight[col]
Selected_condition= Max(Count_condition0, Count_condition1)
Condition_node = Selected_condition
```

4. Separate OR-decision table into two sub-tables with selected condition, and repeat step one for each sub-table.

Figure 5. Proposes OR-decision table conversion algorithm

In order to describe the proposed algorithm and make it more easily understood, in the following example, we created a decision tree from the OR-decision table in Fig. 3 by setting the condition weight to 1.0. The resulting decision tree is shown in Fig. 6 (a) and the hierarchy tree of the OR-decision table is shown in Fig. 6 (b) and the content of each OR-decision table is shown in Fig. 7-21.

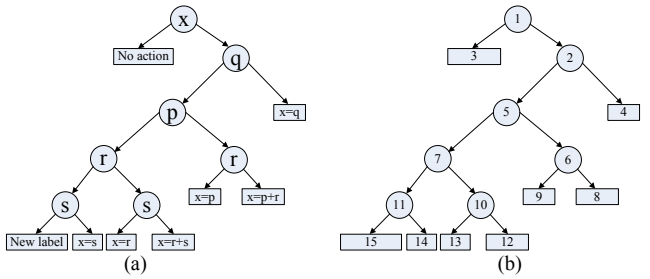


Figure 6. Decision tree convert from OR-decision table in Fig. 3. (a) notation of each node, (b) table number

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | d=x | b=x | r=r | s=x | x=p+r | x=r+s |
| 0 | - | - | - | - | 1 | | | | | | | |
| 1 | 0 | 0 | 0 | 0 | | 1 | | | | | | |
| 1 | 1 | 0 | 0 | 0 | | | 1 | | | | | |
| 1 | 0 | 1 | 0 | 0 | | | | 1 | | | | |
| 1 | 0 | 0 | 1 | 0 | | | | | 1 | | | |
| 1 | 0 | 0 | 0 | 1 | | | | | | 1 | | |
| 1 | 1 | 1 | 0 | 0 | | | 1 | 1 | | | | |
| 1 | 1 | 0 | 1 | 0 | | | | | | | 1 | |
| 1 | 1 | 0 | 0 | 1 | | | 1 | | | 1 | | |
| 1 | 0 | 1 | 1 | 0 | | | | 1 | 1 | | | |
| 1 | 0 | 1 | 0 | 1 | | | | 1 | | 1 | | |
| 1 | 0 | 0 | 1 | 1 | | | | | | | | 1 |
| 1 | 1 | 1 | 1 | 0 | | | 1 | 1 | 1 | | | |
| 1 | 1 | 1 | 0 | 1 | | | 1 | 1 | | 1 | | |
| 1 | 1 | 0 | 1 | 1 | | | | 1 | 1 | | | |
| 1 | 0 | 1 | 1 | 1 | | | | 1 | 1 | 1 | | |
| 1 | 1 | 1 | 1 | 1 | | | 1 | 1 | 1 | 1 | | |
| Total | | | | | **16** | 1 | 6 | 8 | 5 | 6 | 2 | 2 |

1. This is condition node
2. Select action: ‘*no action*’
3. Count input=’0’: [16, 0, 0, 0, 0]
   Count input=’1’: [0, 0, 0, 0, 0]
   Select input: ‘*x*’
4. Split table by condition ‘*x*’

Figure 7. OR-decision table number 1

| Condition | | | | | Action | | | |
|---|---|---|---|---|---|---|---|---|
| x | p | q | r | s | | | assign | merge |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| x | 0 | 0 | 0 | 0 | | 1 | | | | | | |
| X | 1 | 0 | 0 | 0 | | | 1 | | | | | |
| X | 0 | 1 | 0 | 0 | | | | 1 | | | | |
| X | 0 | 0 | 1 | 0 | | | | | 1 | | | |
| X | 0 | 0 | 0 | 1 | | | | | | 1 | | |
| x | 1 | 1 | 0 | 0 | | | 1 | 1 | | | | |
| x | 1 | 0 | 1 | 0 | | | | | | | 1 | |
| x | 1 | 0 | 0 | 1 | | | 1 | | | 1 | | |
| x | 0 | 1 | 1 | 0 | | | | 1 | 1 | | | |
| x | 0 | 1 | 0 | 1 | | | | 1 | | 1 | | |
| x | 0 | 0 | 1 | 1 | | | | | | | | 1 |
| x | 1 | 1 | 1 | 0 | | | 1 | 1 | 1 | | | |
| x | 1 | 1 | 0 | 1 | | | 1 | 1 | | 1 | | |
| x | 1 | 0 | 1 | 1 | | | | | | | 1 | 1 |
| x | 0 | 1 | 1 | 1 | | | | 1 | 1 | 1 | | |
| x | 1 | 1 | 1 | 1 | | | 1 | 1 | 1 | 1 | | |
| Total | | | | | 0 | 1 | 6 | **8** | 5 | 6 | 2 | 2 |

1. This is condition node
2. Select action: ‘*x=q*’
3. Count input=’0’: [x, 4, 0, 4, 4]
   Count input=’1’: [x, 4, 8, 4, 4]
   Select input: ‘q’
4. Split table by condition ‘*q*’

Figure 8. OR-decision table number 2

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | d=x | b=x | r=r | s=x | x=p+r | x=r+s |
| x | - | - | - | - | 1 | | | | | | | |
| Total | | | | | **16** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. This is leaf node
   Assign leaf node: ‘*no action*’

Figure 9. OR-decision table number 3

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | d=x | b=x | r=r | s=x | x=p+r | x=r+s |
| x | 0 | x | 0 | 0 | | | | 1 | | | | |
| x | 1 | x | 0 | 0 | | | 1 | 1 | | | | |
| x | 0 | x | 1 | 0 | | | | 1 | 1 | | | |
| x | 0 | x | 0 | 1 | | | | 1 | | 1 | | |
| x | 1 | x | 1 | 0 | | | 1 | 1 | 1 | | | |
| x | 1 | x | 0 | 1 | | | 1 | 1 | | 1 | | |
| x | 0 | x | 1 | 1 | | | | 1 | 1 | 1 | | |
| x | 1 | x | 1 | 1 | | | 1 | 1 | 1 | 1 | | |
| Total | | | | | 0 | 1 | 6 | **8** | 5 | 6 | 2 | 2 |

1. This is leaf node
   Assign leaf node: ‘*x=q*’

Figure 10. OR-decision table number 4

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | d=x | b=x | r=r | s=x | x=p+r | x=r+s |
| x | 0 | x | 0 | 0 | | 1 | | | | | | |
| x | 1 | x | 0 | 0 | | | 1 | | | | | |
| x | 0 | x | 1 | 0 | | | | | 1 | | | |
| x | 0 | x | 0 | 1 | | | | | | 1 | | |
| x | 1 | x | 1 | 0 | | | | | | | 1 | |
| x | 1 | x | 0 | 1 | | | 1 | | | 1 | | |
| x | 0 | x | 1 | 1 | | | | | | | | 1 |
| x | 1 | x | 1 | 1 | | | | | | | 1 | 1 |
| Total | | | | | 0 | 1 | **2** | 0 | 1 | 2 | 2 | 2 |

1. This is condition node
2. Select action: ‘*x=p*’
3. Count input=’0’: [x, 0, x, 2, 1]
   Count input=’1’: [x, 2, x, 0, 1]
   Select input: ‘*p*’
4. Split table by condition ‘*p*’

Figure 11. OR-decision table number 5

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | d=x | b=x | r=r | s=x | x=p+r | x=r+s |
| x | x | x | 0 | 0 | | | 1 | | | | | |
| x | x | x | 1 | 0 | | | | | | | 1 | |
| x | x | x | 0 | 1 | | | 1 | | | 1 | | |
| x | x | x | 1 | 1 | | | | | | | 1 | 1 |
| Total | | | | | 0 | 0 | **2** | 0 | 0 | 1 | 2 | 1 |

1. This is condition node
2. Select action: ‘*x=p*’

3. Count input='0': [x, x, x, 2, 1]
   Count input='1': [x, x, x, 0, 1]
   Select input: '*r*'
4. Split table by condition '*r*'

Figure 12. OR-decision table number 6

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | x=p | x=q | x=r | x=s | x=p+r | x=r+s |
| x | x | x | 0 | 0 | | 1 | | | | | | |
| x | x | x | 1 | 0 | | | | | 1 | | | |
| x | x | x | 0 | 1 | | | | | | 1 | | |
| x | x | x | 1 | 1 | | | | | | | | 1 |
| Total | | | | | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |

1. This is condition node
2. Select action: '*new label*'
3. Count input='0': [x, x, x, 1, 1]
   Count input='1': [x, x, x, 0, 0]
   Select input: '*r*'
4. Split table by condition '*r*'

Figure 13. OR-decision table number 7

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | x=p | x=q | x=r | x=s | x=p+r | x=r+s |
| x | x | x | x | 0 | | | | | | | 1 | |
| x | x | x | x | 1 | | | | | | | 1 | 1 |
| Total | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 |

1. This is leaf node
   Assign leaf node: '*x=p+r*'

Figure 14. OR-decision table number 8

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | x=p | x=q | x=r | x=s | x=p+r | x=r+s |
| x | x | x | x | 0 | | | 1 | | | | | |
| x | x | x | x | 1 | | | | | | 1 | | |
| Total | | | | | 0 | 0 | 2 | 0 | 0 | 1 | 0 | 0 |

1. This is leaf node
   Assign leaf node: '*x=p*'

Figure 15. OR-decision table number 9

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | x=p | x=q | x=r | x=s | x=p+r | x=r+s |
| x | x | x | x | 0 | | | | | 1 | | | |
| x | x | x | x | 1 | | | | | | | | 1 |
| Total | | | | | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

1. This is condition node
2. Select action: '*x=r*'
3. Count input='0': [x, x, x, x, 1]
   Count input='1': [x, x, x, x, 0]
   Select input: '*s*'
4. Split table by condition '*s*'

Figure 16. OR-decision table number 10

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | x=p | x=q | x=r | x=s | x=p+r | x=r+s |
| x | x | x | x | 0 | | 1 | | | | | | |
| x | x | x | x | 1 | | | | | | 1 | | |
| Total | | | | | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

1. This is condition node
2. Select action: '*new action*'
3. Count input='0': [x, x, x, x, 1]
   Count input='1': [x, x, x, x, 0]
   Select input: '*s*'
4. Split table by condition '*s*'

Figure 17. OR-decision table number 11

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | x=p | x=q | x=r | x=s | x=p+r | x=r+s |
| x | x | x | x | x | | | | | | | | 1 |
| Total | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

1. This is leaf node
   Assign leaf node: '*x=r+s*'

Figure 18. OR-decision table number 12

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | x=p | x=q | x=r | x=s | x=p+r | x=r+s |
| x | x | x | x | x | | | | | 1 | | | |
| Total | | | | | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

1. This is leaf node
   Assign leaf node: '*x=r*'

Figure 19. OR-decision table number 13

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | x=p | x=q | x=r | x=s | x=p+r | x=r+s |
| x | x | x | x | x | | | | | | 1 | | |
| Total | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

1. This is leaf node
   Assign leaf node: '*x=s*'

Figure 20. OR-decision table number 14

| Condition | | | | | Action | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | assign | | | | merge | |
| x | p | q | r | s | no action | new label | x=p | x=q | x=r | x=s | x=p+r | x=r+s |
| x | x | x | x | x | | 1 | | | | | | |
| Total | | | | | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

1. This is leaf node
   Assign leaf node: '*new label*'

Figure 21. OR-decision table number 15

From the *Block Based Decision Table* (BBDT) proposed by [1], it can produce the 65,536 rules *Pixel Based Decision Table* (PBDT), which contains more than one (equivalent) actions for each set of conditions in the OR-decision table. The method of [1] produces the decision tree that contains 211 leaf nodes. The proposed algorithm was applied to the same OR-decision table, producing a decision tree containing 225 leaf nodes, sparse with less computation time.

## IV. CONCLUSION

In this paper, an OR-decision table conversion algorithm is proposed. The proposed algorithm converts an OR-decision table into a decision tree without converting the OR-decision table into a single action decision table. The result of using the proposed algorithm is that a nearly optimum decision tree is produced that contains 225 leaf nodes sparse over 14 levels. Despite the fact that the proposed algorithm creates a few more leaf nodes when compared to the original work, it consumes less computation time with smaller memory requirments and is easier to implement.

## REFERENCES


[1] C. Grana, D. Borghesani and R. Cucchiara, "Optimized block-based connected components labeling with decision trees," IEEE Transactions on Image Processing, vol 19, issue 6, pp. 1596-1609, June 2010.

[2] H. Schumacher and K. C. Sevcik, "The synthetic approach to decision table conversion," Commun. ACM, vol. 19, no. 6, pp. 343–351, 1976.

[3] B. M. E. Moret, "Decision Trees and Diagrams," ACM Computing Surveys (CSUR), vol.14 no. 4, pp. 593-623, Dec. 1982.

[4] L. T. Reinwald and R. M. Soland, "Conversion of limited-entry decision tables to optimal computer programs i: Minimum average processing time," J. ACM, vol. 13, no. 3, pp. 339–358, 1966.