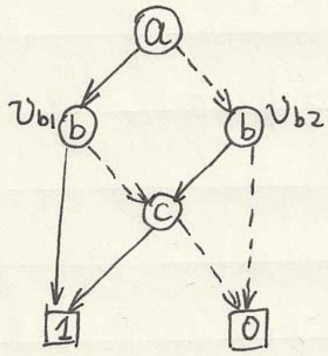
6) Pseudo code : next page.

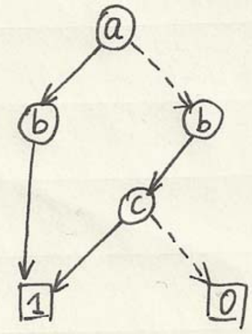
Apply on example : $f = ab + ac + bc$, $a > b > c$



Original ROBDD for f

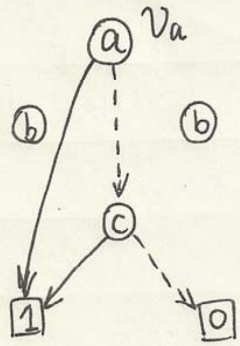
Traverse and visit v_{b1}, v_{b2}

⇒



Delete edges to $low(v_{b1})$, $low(v_{b2})$

⇓



Visit $high(v_{b1})$, $high(v_{b2})$

delete edges connecting them w/ v_{b1},v_{b2}

set $high(v_a)$ to $high(v_{b1})$

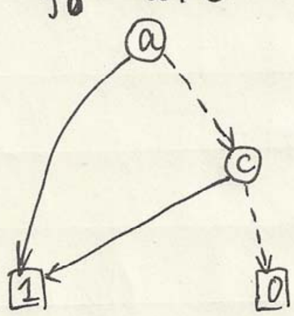
connect $high(v_{b1})$ to v_a

set $low(v_a)$ to $high(v_{b2})$

connect $high(v_{b2})$ to v_a

⇐

$f_b = a + c$



delete v_{b1}, v_{b2} and all edges to them

reduce ROBDD, delete disconnected subtrees, merge repeated nodes and isomorphic subtrees

Problem 6

Problem description: Given a ROBDD f with variables $(x_1, x_2, \ldots, x_n)$ which is ordered by $x_1 > x_2 > \cdots > x_n$. Our objective is to transform f to ROBDD f_{x_i} eliminating variable x_i (or $f_{x_i'}$ when x_i' is negative cofactor)

Algorithm 1 Arbitrary variable elimination algorithm on ROBDD

```
function ROBDDVarElim(f, i)
    if v = top(f) then                                   ▷ v is the variable of top node
        return f_v or f_v' ▷ Directly return f_v or f_v' when requiring negative cofactor
    else
        while BFS_Traverse(f) do
            if idx(v) = i then                           ▷ Reach nodes of variable x_i
                Edge.Delete(v, low(v))
                Edge.Delete(v, high(v))                  ▷ Delete its edges to children
                for all parent(v) do                     ▷ For all of its parent nodes
                    if x_i is positive cofactor then
                        Redirect edge ⟨parent(v), v⟩ to ⟨parent(v), high(v)⟩
                    else
                        Redirect edge ⟨parent(v), v⟩ to ⟨parent(v), low(v)⟩
                    end if
                    Clean-up if a node has no reference  ▷ Please refer to Example 1
                end for
                Node.Delete(v)
            end if
        end while
        f_xi ←Reduce(top(f))                             ▷ Please refer to Example 2
        return f_xi
    end if
end function
```

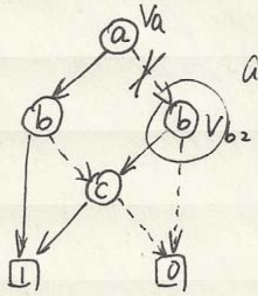
Note: $low(), high(), idx()$ means the child on FALSE edge, child on TRUE edge, and index of variables of this node. Their definitions and function *Reduce*() can be found in *Graph-Based Algorithms for Boolean Function Manipulation* by R. E. Bryant, which is linked on class webpage.

科目 ECE/CS 5745/6745 班级 姓名 学号 Fall 2014 月 日

Example 1 (Clean-up)

$f = ab + ac + bc$, $a > b > c$,

calculating f_a:



after deleting edge $\langle V_a, V_{b2} \rangle$

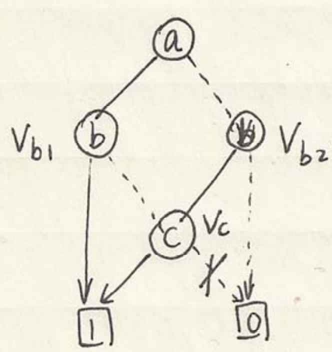
V_{b2} is node with no reference

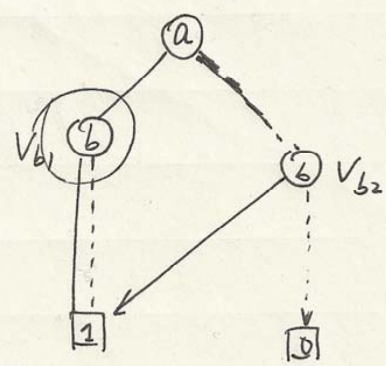
so delete V_{b2} and any edges connecting to it.

Example 2

$f = ab + ac + bc$, $a > b > c$

calculating f_c:





delete $\langle V_c, 0 \rangle$

redirect $\langle V_{b1}, V_c \rangle \rightarrow \langle V_{b1}, 1 \rangle$

$\langle V_{b2}, V_c \rangle \rightarrow \langle V_{b2}, 1 \rangle$

V_{b1} is redundant!

reduce, delete V_{b1}