6) Pseudo code: next page.

Apply on example: \( f = ab + ac + bc, \ a > b > c \)

Original ROBDD for \( f \)

Traverse and visit \( V_{b1}, V_{b2} \)

\( f_b = a + c \)

\( \downarrow \)

Visit \( \text{high}(V_{b1}), \text{high}(V_{b2}) \)

Delete edges connecting them \( w/ \ V_{b1} \)

Set \( \text{high}(Va) \) to \( \text{high}(V_{b1}) \)

Connect \( \text{high}(V_{b1}) \) to \( Va \)

Set \( \text{low}(Va) \) to \( \text{high}(V_{b2}) \)

Connect \( \text{high}(V_{b2}) \) to \( Va \)

Delete \( V_{b1}, V_{b2} \) and all edges to them

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

\( \leftarrow \)
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

1: function ROBDDVarElim($f, i$)  
2: if $v$ = top($f$) then \(\triangleright v\) is the variable of top node
3: \hspace{1cm} return $f_v$ or $f_{v'}$ \(\triangleright\) Directly return $f_v$ or $f_{v'}$ when requiring negative cofactor
4: else
5: \hspace{1cm} while BFS_Traverse($f$) do
6: \hspace{2cm} if idx($v$) = $i$ then \(\triangleright\) Reach nodes of variable $x_i$
7: \hspace{3cm} Edge>Delete($v$, low($v$))
8: \hspace{3cm} Edge>Delete($v$, high($v$)) \(\triangleright\) Delete its edges to children
9: \hspace{3cm} for all parent($v$) do \(\triangleright\) For all of its parent nodes
10: \hspace{4cm} if $x_i$ is positive cofactor then
11: \hspace{5cm} Redirect edge (parent($v$), $v$) to (parent($v$), high($v$))
12: \hspace{5cm} else
13: \hspace{6cm} Redirect edge (parent($v$), $v$) to (parent($v$), low($v$))
14: end if
15: \hspace{2cm} Clean-up if a node has no reference  \(\triangleright\) Please refer to Example 1
16: \hspace{1cm} end for
17: \hspace{1cm} Node>Delete($v$)
18: end if
19: end while
20: $f_{x_i}$ $\leftarrow$ Reduce(top($f$)) \(\triangleright\) Please refer to Example 2
21: return $f_{x_i}$
22: end if
23: end function

Note: low($v$), high($v$), idx($v$) 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.
Example 1 (Clean-up)

\[ f = ab + ac + bc, \quad a > b > c \]

calculating \( f_a \):  

After deleting edge \( \langle V_a, V_{b_2} \rangle \)

\( V_{b_2} \) is node with no reference

So delete \( V_{b_2} \) and any edges connecting to it.

Example 2

\[ f = ab + ac + bc, \quad a > b > c \]

calculating \( f_c \):

\[ \langle V_{b_1}, 0 \rangle \]

redirect \( \langle V_{b_1}, V_c \rangle \rightarrow \langle V_{b_1}, 1 \rangle \)
\[ \langle V_{b_2}, V_c \rangle \rightarrow \langle V_{b_2}, 1 \rangle \]

\( V_{b_1} \) is redundant!

\( V_{b_1} \) is redundant!

reduce, delete \( V_{b_1} \)