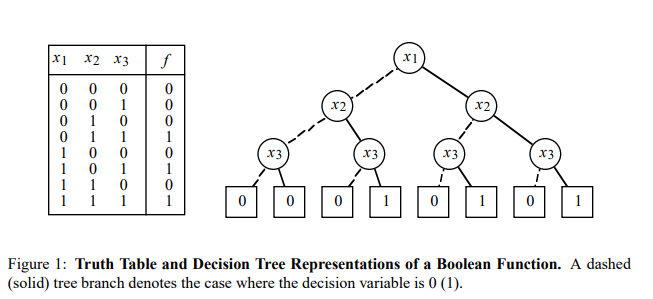
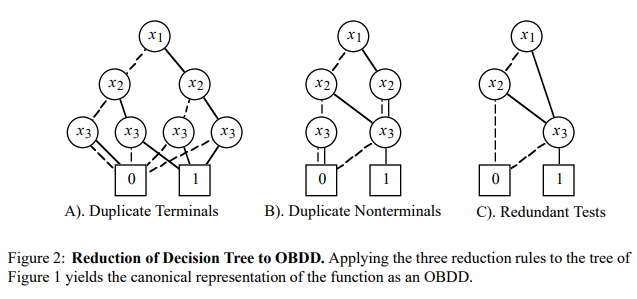
# P1 BDD



* All the nodes in different bdds has the same order

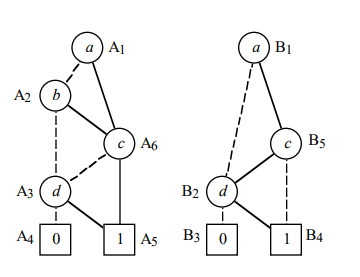
# P2 the algorithm

## Reduction

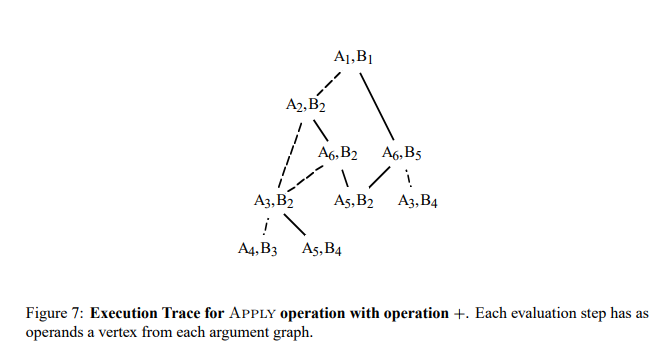


* Collapse the bottom
* Combine the same node from bottom to top
* Kill the node only point to one node

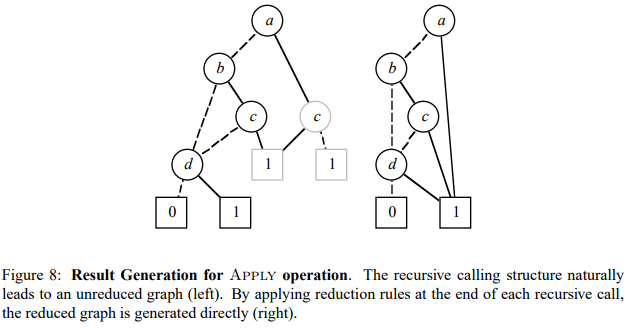
## Apply



* 2 bdds with different numbers of node



* The order of execution



* Left one is unreduced one, then reduce to the right one.

# P3 how to parallelize

## reduction

reach the bottom then distribute the task by level.

Also can do ASAP so every time nodes at lower level is done, the corresponding upper one is ready to go.

## Apply

Distribute by level from top to bottom.

2 kinds of operations:

Combine the same node(c,d): test the node if it’s totally same, 1 then add it to new tree, 0 then keep both node and wait for reduction.

Add a new node(b): insert all the node at the lower level including itself to the tree.

## P4 method we choose

MPI:

Strong details control. (can send part of the structure and can spread the tasks manually)

Hard to use when condition not quite certain. (random bdd trees might need different kind of strategy)

Pthread:

create threads when needed, no manual communication command lines. (EZ to use)

no need for parser for the node structure. (EZ to use)

auto distribution might cause some trouble

CUDA:

GPU is not really good at these complex operations with no huge simple tasks.

VXM(Vector one):

Don’t really need vectorized computation ability.