Direct Construction of ROBDDs

- In the past lectures, we learnt:
  - $ITE(f, g, h) = f \cdot g + f' \cdot h$
  - $f \odot g = x'(f_{x'} \odot g_{x'}) + x(f_x \odot g_x)$
  - ITE at top-nodes of $f, g, h \rightarrow$ ITE at their cofactors
  - Operate on graphs of f, g, h and derive the graph for $Z = ite(f, g, h)$
  - Shannon's expansion always w.r.t. top-node of f, g, h
- Use of a symbol table as a unique table
  - Every time you create a new node, Reduce it
  - Then, compute its *Key:* $\{low(v), v, high(v)\}$
  - No duplicate keys to be stored in the hash-table

The ITE Algorithm

$ITE(f, g, h)\{$

if (terminal case)

return trivial result;

else

$x =$ top variable of f, g, h;

$e = ITE(f_{x'}, g_{x'}, h_{x'})$;

$t = ITE(f_x, g_x, h_x)$;

if ($t == e$)/* redundant node */

return t; /* or return e */

/* Look-up the unique table for isomorphic subtrees */

$r = find\_or\_add\_in\_unique\_table(e, x, t)$;

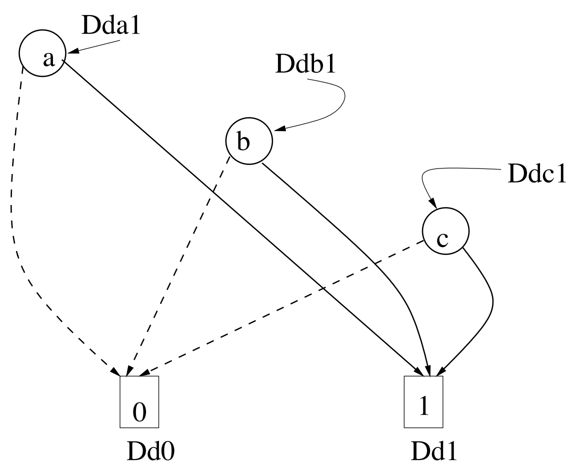
Update unique table, if required;

return r;

end if

ROBDD Construction Example

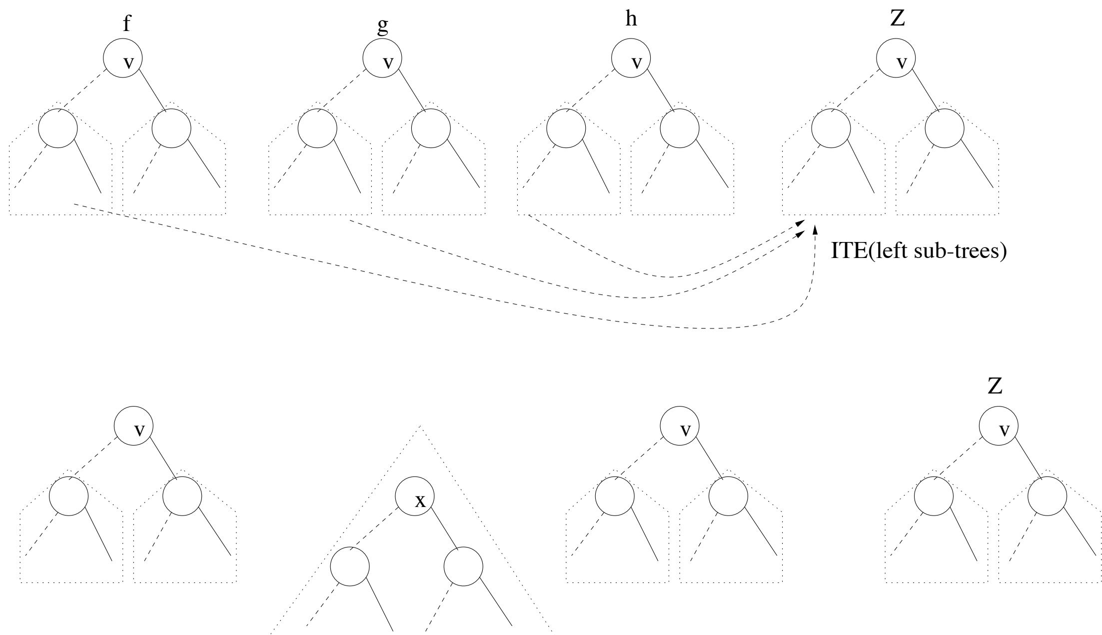
- $f = a + c; g = b + c; f \cdot g = ab + c$
- First construct trivial ROBDDs for a, b, c
- Assume var order a, b, c
- Fill-up the Unique Table (symbol/hash table)



Key	Value
{NULL, 0, NULL}	Dd0
{NULL, 1, NULL}	Dd1
{Dd0. a, Dd1}	Dda1
{Dd0, b, Dd1}	Ddb1
{Dd0, c, Dd1}	Ddc1

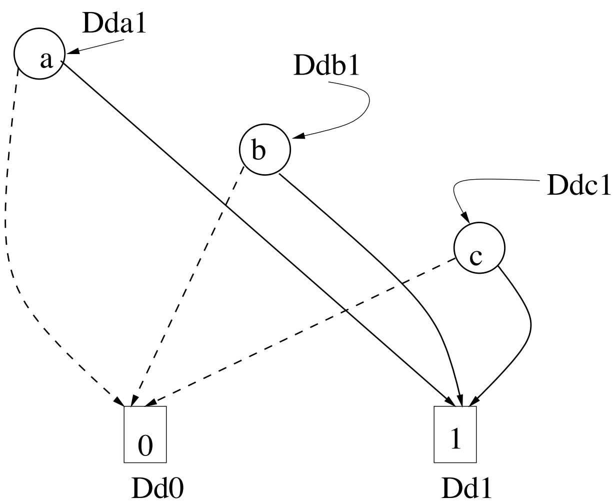
Constructing ROBDDs using ITE

- Few things to keep in mind....
- When f, g, h have same top variable...
- When the f, g, h may not have same top-vars



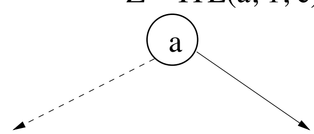
Left sub-tree of Z =
ITE(left subtree of f,
tree rooted at g,
left subtree of h)

Construct $f = a + c$



Key	Value
{NULL, 0, NULL}	Dd0
{NULL, 1, NULL}	Dd1
{Dd0. a, Dd1}	Dda1
{Dd0, b, Dd1}	Ddb1
{Dd0, c, Dd1}	Ddc1

Z = ITE(a, 1, c) = ITE(Dda1, DD1, Ddc1)

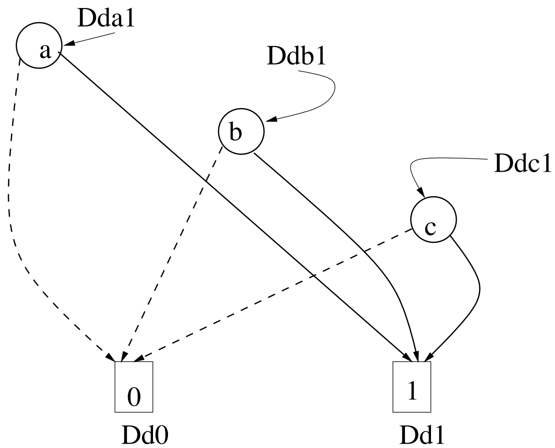


TOP-VAR = a

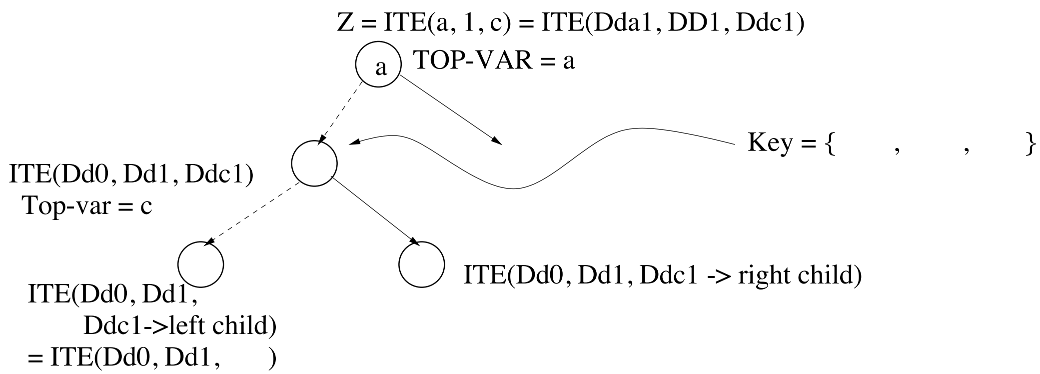
ITE(Dda1->left child, Dd1, Ddc1)

= ITE(Dd0, Dd1, Ddc1)

$f = a + c$ Contd....

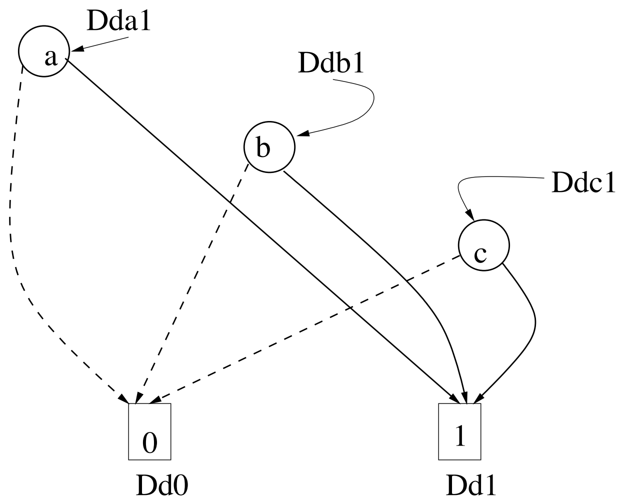


Key	Value
{NULL, 0, NULL}	Dd0
{NULL, 1, NULL}	Dd1
{Dd0. a, Dd1}	Dda1
{Dd0, b, Dd1}	Ddb1
{Dd0, c, Dd1}	Ddc1

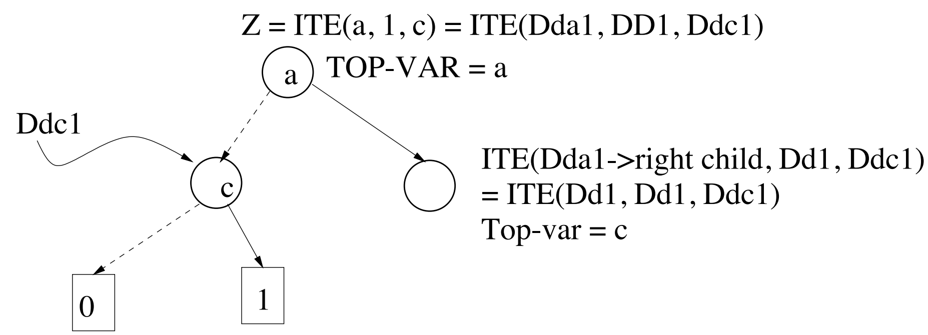


- Go-up a recursion level and compute the right sub-tree

$f = a + c$ Contd. further...

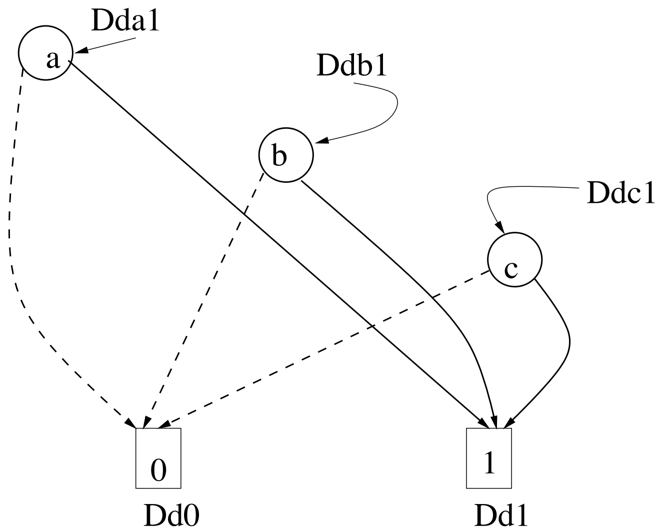


Key	Value
{NULL, 0, NULL}	Dd0
{NULL, 1, NULL}	Dd1
{Dd0. a, Dd1}	Dda1
{Dd0, b, Dd1}	Ddb1
{Dd0, c, Dd1}	Ddc1

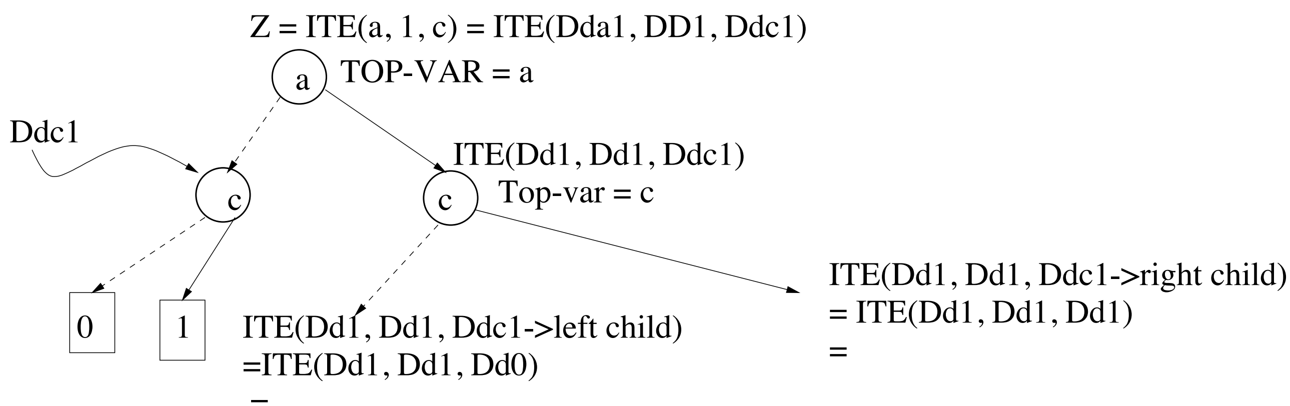


•

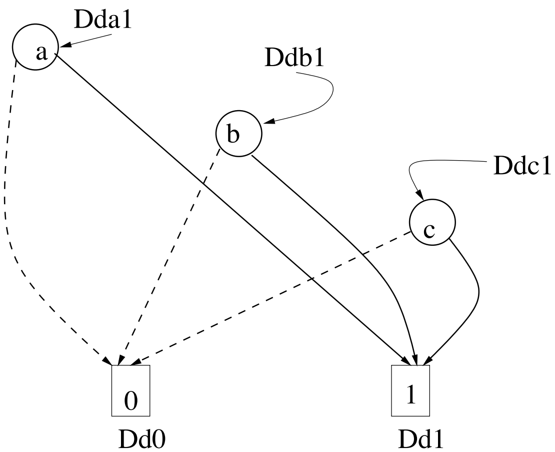
$f = a + c$ Contd. even further...



Key	Value
{NULL, 0, NULL}	Dd0
{NULL, 1, NULL}	Dd1
{Dd0. a, Dd1}	Dda1
{Dd0, b, Dd1}	Ddb1
{Dd0, c, Dd1}	Ddc1

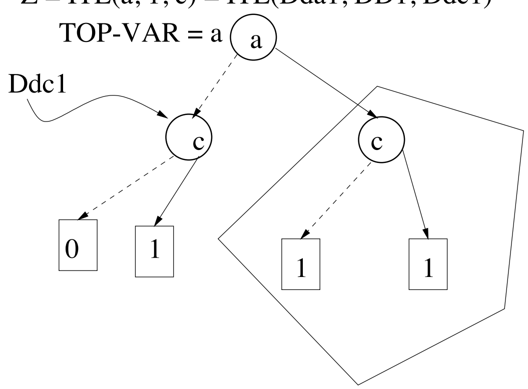


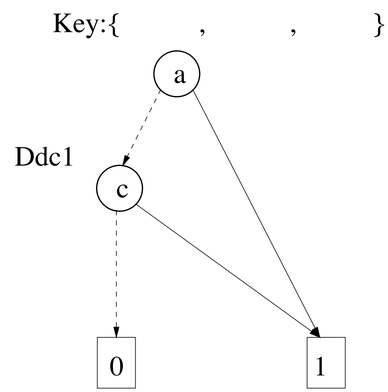
Final ROBDD for $f = a + c$



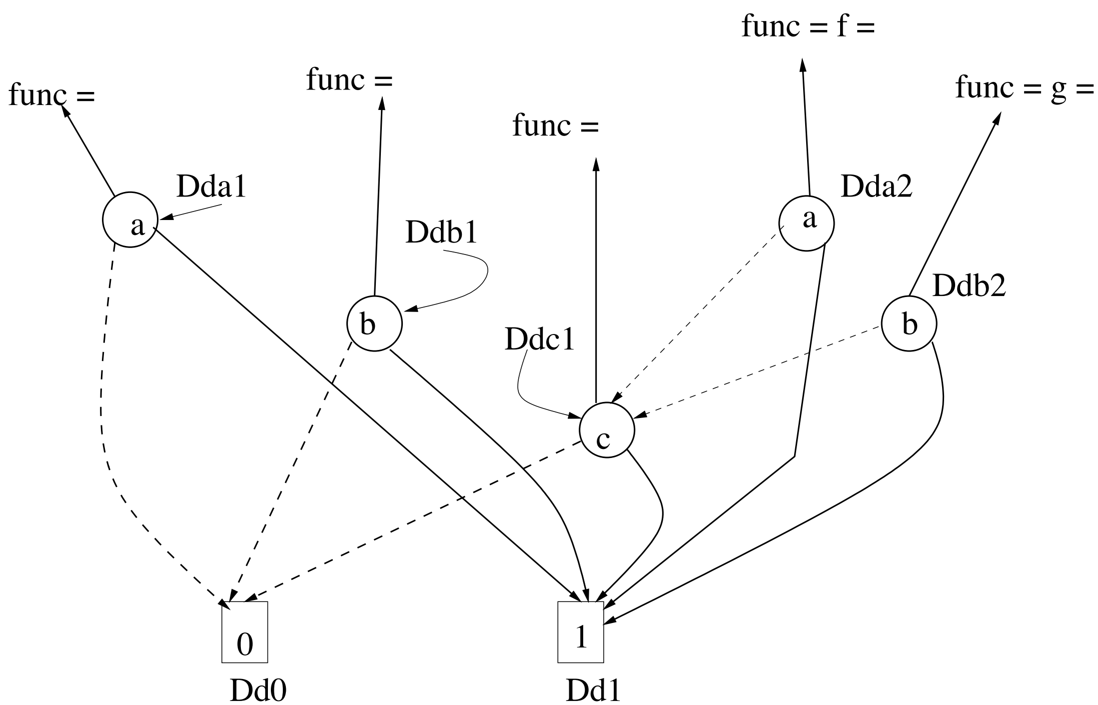
Key	Value
{NULL, 0, NULL}	Dd0
{NULL, 1, NULL}	Dd1
{Dd0. a, Dd1}	Dda1
{Dd0, b, Dd1}	Ddb1
{Dd0, c, Dd1}	Ddc1

Z = ITE(a, 1, c) = ITE(Dda1, DD1, Ddc1)





$$f \cdot g = (a + c) \cdot (b + c)$$

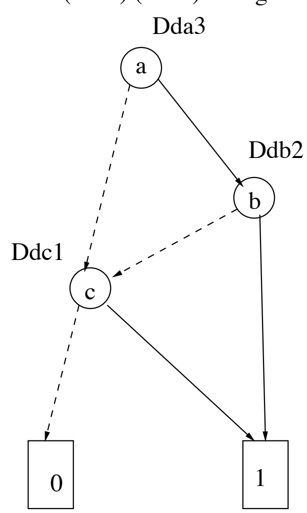


- Homework: Compute $f \cdot g$ yourself!
- Homework: Compute $(f = ab) \cdot (g = ab')$ yourself and notice how the graph reduces to $Dd0$.

Equivalence Verification

- $f = (a + c)(b + c)$; $g = ab + c$
- Compute f, compute keys of every node, update symbol table
- Do the same with g and prove to yourself that $f = g$

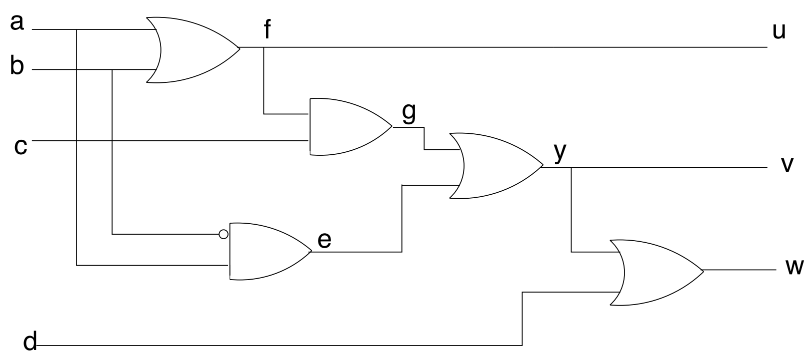
f = (a + c) (b + c) <=> g = ab + c

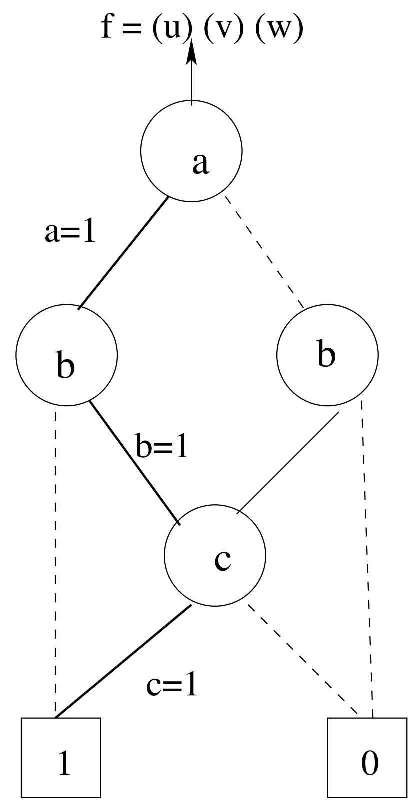


Key(f) = {Ddc1, a, Ddb2}
Key(g) = {Ddc1, a, Ddb2}
Key(f) = Key(g) <=> f = g!

Application to SAT

- For the circuit shown, SAT(u=v=w=1) = ac + bc + ab'
- BDD $(u \cdot v \cdot w)$ shown. How to pick a solution?





BDD Size Problems and Variable Order

- Change Var order $\rightarrow$ ROBDD structure and size changes!
- Worst-case ROBDD size $\rightarrow$ no reduction $\rightarrow$ full-blown tree.
- Given a Boolean function, how do we forecast a good var order?
- Intractable Problem! Though some heuristics exist....
- Var $x \in$ many cubes $\rightarrow$ keep it up in the tree
- **Variable Ordering**: Interchange order of variables and see the change in size of ROBDD
- **Dynamic Var Ordering**: Do this while constructing OBDDs
- Careful: Var ordering of ALL the BDDs in the manager has to go through re-ordering.
- Multipliers: There exists NO good ordering. Take any order, at least one output would have worst-case scenario!