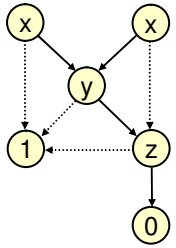
## Efficient Implementation of Prime Tree Operations

[R. Bryant 1986]

- direct implementation of "and $t_1$ $t_2$" is exponential (exponential number of recursive calls)
- **Minimal graph representation** of prime trees yields constant equality test
- **Memoing*** of triples "and $t_1$ $t_2$ = $t_3$" yields $O(n^2)$ algorithm where $n$ is the number of nodes reachable from $t_1$, $t_2$

* Dynamic Programming

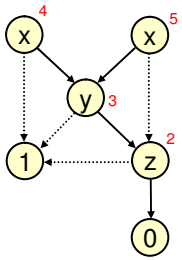
## Minimal Graph Representation




- Every node describes a prime tree
- Graph describes a subtree-closed set of prime trees
- Graph minimal iff different nodes describe different trees

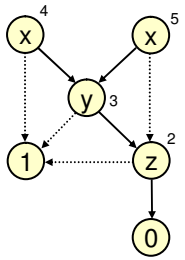
Binary decision diagrams (BDDs)

## Graph → Table



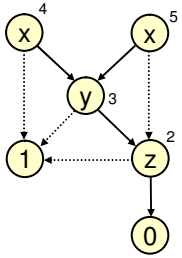

Number nodes of graph

## Graph → Table




| | |
|---|---|
| 2 | (z,1,0) |
| 3 | (y,1,2) |
| 4 | (x,1,3) |
| 5 | (x,2,3) |

# Graph → Table → Function




| i | tab(i) |
|---|---|
| 2 | (z,1,0) |
| 3 | (y,1,2) |
| 4 | (x,1,3) |
| 5 | (x,2,3) |

Graph minimal iff tab injective

# Constant Time Realization of cond

cond(x,n,n') =
  if n=n' then n
  else if (x,n,n') ∈ Dom($tab^{-1}$)
    then $tab^{-1}$ (x,n,n')
    else let n'' = least number not in Dom tab
      in tab := tab[n'':=(x,n,n')] ;
        n''

Implement $tab^{-1}$ with hashing