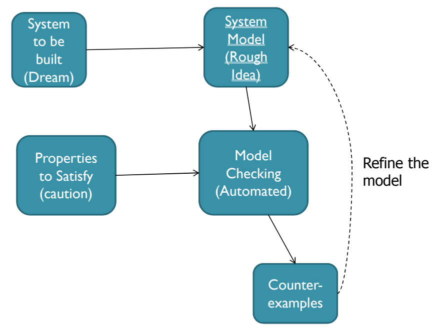
Binary Decision Diagrams - I

CS 4271
Abhik Roychoudhury
National University of Singapore

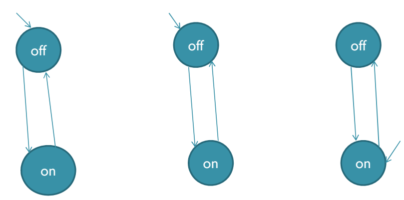
Overall picture



Complexity of Model Checking

- In terms of
  - $|\varphi|$ size of formula
  - $|S|$ number of states in M
  - $|R|$ number of transitions
- At each level of nesting of φ
  - Employ the EG, EU, EX algorithms
  - Efficient EG algorithm is $O(|S| + |R|)$
  - Similarly for EU, EX algorithms
- Complexity is $O(|\varphi| * (|S| + |R|))$

State Explosion



Global State transition system is a composition of the local state machines.
3 components, 2 states each == up to $2^3 = 8$ states
N components, m states each == up to m^N states

State Explosion Problem

- Our CTL model checking algorithm is linear in size of state space and formula
  - Suffers from State Space Explosion problem since the state space is exponential in the number of system components.
- We need more space efficient representation of sets of states and transition relation
  - Reduced Ordered Binary Decision Diagrams (ROBDD) is a data structure to achieve this.

ROBDD – the context

- To discuss ROBDD, let us first discuss BDD.
- BDD is a data structure for compactly representing boolean functions.
  - Boolean functions have a fundamental role in computing
  - Suitable for directly modeling combinational circuits
  - Can capture the set of states and transition relation of finite state machines corresponding to sequential circuits.
    - **Leads to more space efficient model checking**

BDD – historical note

- Succinct representation of boolean functions
  - More succinct than truth tables
- Can represent combinational circuits originally used in circuit equivalence checking
  - For two combinational circuits C1 and C2 to check that they implement the same boolean function
- Application to model checking in the early '90s
  - Represent the sets of states and transitions of a Kripke Structure in a compact fashion.

Boolean function

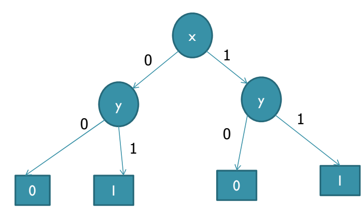
- A mapping
  - $\{F,T\}^n \rightarrow \{F,T\}$
- Might be expressed in various forms
  - Tabular notation
    - Truth Table
  - Closed form representation
    - Propositional logic formula e.g. $F(x,y) = x \wedge y$

Truth Table

x	y	f(x, y)
0	0	0
0	1	0
1	0	0
1	1	1

Enumerates the truth assignments which make the function true

Boolean functions and Binary Decision Trees



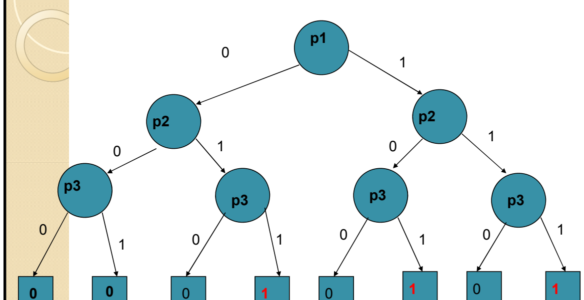
No more space efficient than truth tables.

We derive a more compact data structure to improve space usage

Boolean function representation

p1	p2	p3	f(p1, p2,p3)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

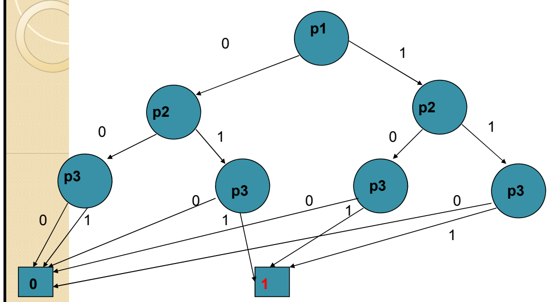
Decision trees



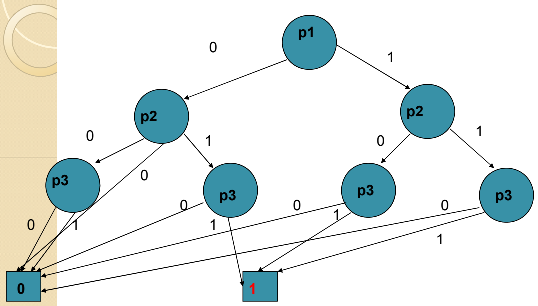
Avoiding blow-up

- No blow-up avoided in the truth table and decision tree representations.
  - Still explicitly enumerate the results of all the 2^k possible valuations.
  - Need a more optimized representation of boolean functions.
  - Transform the decision tree to a DAG to remove isomorphic sub-trees etc. etc. etc.

Txfm1: Collapse leaves

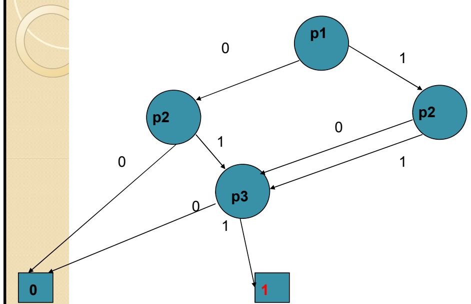


Txfm2: Redundant tests

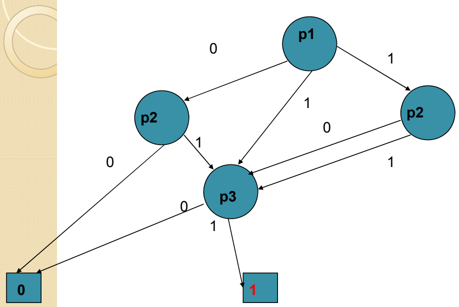


WARNING: Slide has several animations, printout may be confusing. Please view it in screen

Txfm3: Collapse internal nodes

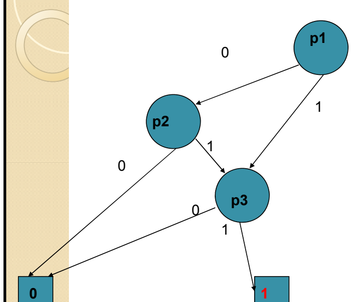


More Txfm may be applicable



WARNING: Slide has several animations, printout may be confusing. Please view it in screen

Final representation



Represents our function.

"Binary decision diagram" (BDD)

0-1 annotations can be dropped

Left edge is 0-successor

Right edge is 1-successor

BDD

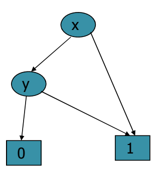
- A BDD is a finite directed acyclic graph s.t.
  - It has a unique initial node
  - All leaves are labeled 0 or 1
  - All internal nodes are labeled with a boolean variable.
  - Each internal node has exactly two children (the outgoing edges are labeled 0 and 1)
- A representation for boolean functions.

Reduced BDD

- A BDD is reduced if none of the following optimizations can be applied to it for size reduction
  - Share terminal nodes
  - Remove redundant tests
  - Share non-terminal nodes
    - Identical sub-graphs are represented once

Ordered BDD

- A BDD is ordered if the boolean variables appearing in the BDD appear in the same order along each root-to-leaf path.
  - All the variables may not appear in all the paths



ROBDD

- Reduced and ordered BDD
  - w.r.t. a specific variable order
- The variable order ensures a normal form
  - Given
    - a boolean function f(x, y, z) and
    - a total order among x, y, z (e.g. z > x > y)
  - The ROBDD representation of f(x, y, z) is unique.

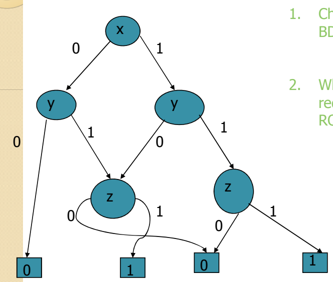
Reduction Algorithm

- We assume the following annotations for the BDD
  - Value of terminal node v, denoted by val(v)
  - Variable at a non-terminal node denoted by var(v)
  - Left and Right children of a node v denoted by low(v) and high(v)
    - The 0 and 1 successors.
  - Each node v has an index(v) capturing the level # of v i.e.
    - Index(v) < Index(low(v)) and Index(v) < Index(high(v))

Reduction Algorithm

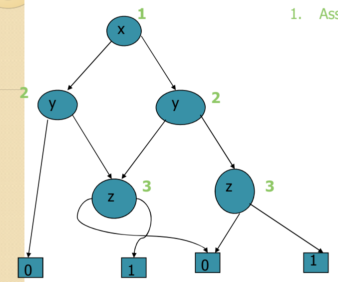
- Merge leaves with same value;
- For id := n downto 1 do
  - for all nodes v with index(v) = id do
    - if low(v) = high(v) then replace v by low(v);
    - if $\exists$ v' (index(v')=id $\wedge$ low(v')=low(v) $\wedge$ high(v')=high(v))
    - then substitute v by v'
  - endfor
- endfor

Exercise



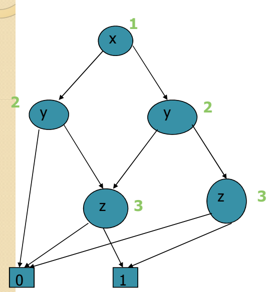
1. Check that the following BDD is ordered.
2. What would be the reduction steps to get a ROBDD ?

Exercise



1. Assign indices for levels

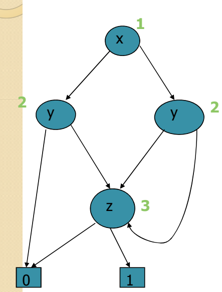
Exercise



2. Reduction algorithm

Combine leaf nodes

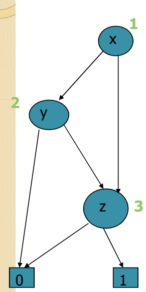
Exercise



2. Reduction algorithm

-- optimizations for id= 3

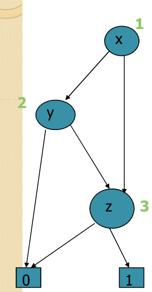
Exercise



2. Reduction algorithm

-- optimizations for id= 2

Exercise



2. Reduction algorithm

-- optimizations for id = 1

None, final answer

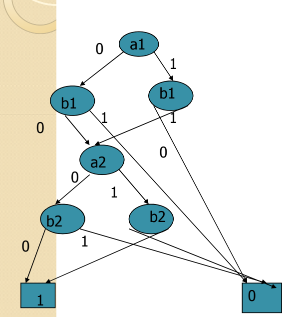
Variable Ordering Problem

- Given a boolean function f and a fixed variable order, there cannot be two ROBDDs.
- Given a boolean function f, there can be several ROBDDs implementing the function for different variable orders.
- The choice of variable order can make a dramatic difference in BDD size
  - From polynomial to exponential in terms of the number of boolean variables.

Two bit comparator

- Check for equality of two 2-bit numbers
  - Whether (a2,a1) = (b2,b1)
- Implemented by the boolean function
  - F(a1,a2,b1,b2) = (a1 ⇔ b1) ∧ (a2 ⇔ b2)
  - Draw the ROBDD of F(a1,a2,b1,b2) for
    - The variable order a1 < b1 < a2 < b2
    - The variable order a1 < a2 < b1 < b2

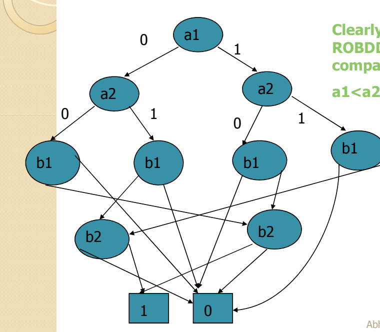
a1 < b1 < a2 < b2



3n+2 nodes for the ROBDD representation of a n-bit comparator with the order

a1<b1<a2<b2<…<an<bn

a1<a2<b1<b2



Clearly $O(2^n)$ nodes in the ROBDD representation of n-bit comparator for the order

a1<a2<…< an<b1<b2<…<bn

Bad news

- Given a boolean function f, different variable orders lead to ROBDD of different size
  - How to find the optimal order ?
- Given function f and a variable order on the input vars. of f, even checking for the optimality of the given variable order is NP-complete.

More Bad news

- For certain boolean functions, the ROBDD is exponential in the # of boolean vars for any variable ordering
  - The n*n bit integer multiplier
  - $(c_1,c_2,\ldots,c_{2n}) = (a_1,a_2,\ldots,a_n) * (b_1,b_2,\ldots,b_n)$
  - Each c_i can be defined by a boolean function
    - **F_i $(a_1,a_2,\ldots,a_n,b_1,b_2,\ldots,b_n)$**
  - **For any variable order on the 2n input variables, at least one of the F_i's ROBDD is ofexponential size (in terms of n)**
    - **Ref: Bryant's 1986 paper (see Lesson Plan in IVLE)**

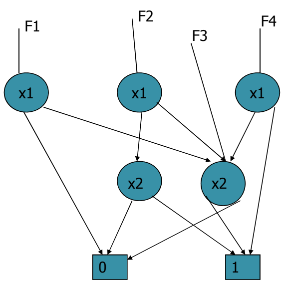
So Far

- BDD
  - As a representation for boolean functions
  - Reduced BDD for compact representation
  - ROBDD for compactly representing normal form
  - No theoretical guarantees about compactness
    - Compactness sensitive to variable order.
    - Cannot compute the "best" variable order.
    - All variable orders may produce an exponential sized BDD for some examples.
- Now, wrapping up …

Shared BDDs

- For a circuit representation, often there are several boolean functions to capture
  - e.g. , a combinational circuit with many outputs
  - These boolean functions may have common sub-functions
  - Shared ROBDD structure to represent all the boolean functions
  - Promote sharing of subgraphs across the representation of different boolean functions !

Shared BDDs



$F1 = x1 \wedge (\neg x2)$

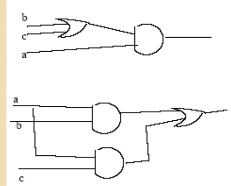
$F2 = x1 \oplus x2$

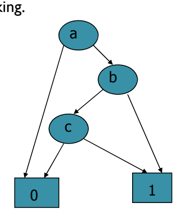
$F3 = \neg x2$

$F4 = x1 \vee (\neg x2)$

Circuit Equivalence Checking

- Normal form of the ROBDD representation can be directly exploited for circuit equivalence checking.





Same ROBDD for the order a < b < c

Summary

- In this class:
  - BDD represent intermediate sets of states during model checking in a space efficient fashion.
  - Canonical form to detect fixed points.
- Next class:
  - Representing states and transitions via BDD
  - Set operations performed during model checking achieved through boolean operations on BDD.
- Next-to-next class:
  - Symbolic model checking algorithm which proceeds by manipulating BDDs.