Decision Diagrams

Gianpiero Cabodi ***Stefano Quer***

Politecnico di Torino

Torino, Italy

{gianpiero.cabodi,stefano.quer}@polito.it

http://staff.polito.it/{gianpiero.cabodi,stefano.quer}/

Reference

❖ **Papers**

R. E. Bryant

"Binary Decision Diagrams and Beyond: Enabling Technologies for Formal Verification"

IEEE ICCAD 1995

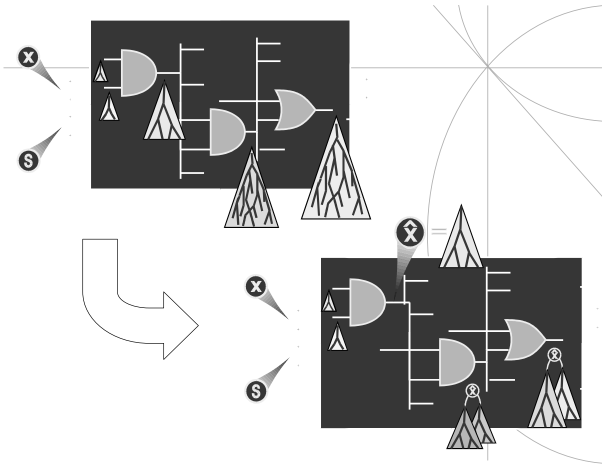
General Ideas for Alternatives DDs

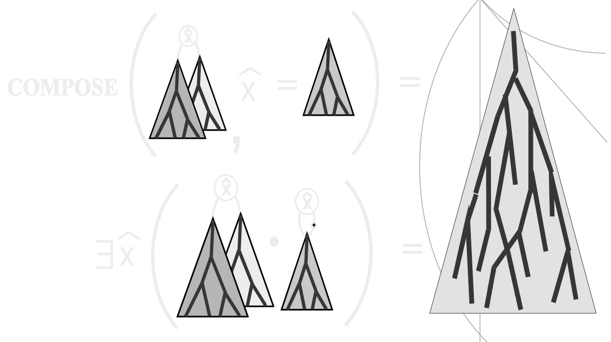
❖ When BDDs are too complex it is possible to
- Decompose BDDs
- Modifying the representation
  - Change the reduction rules
  - Change the function decomposition
  - Relax variable ordering requirements

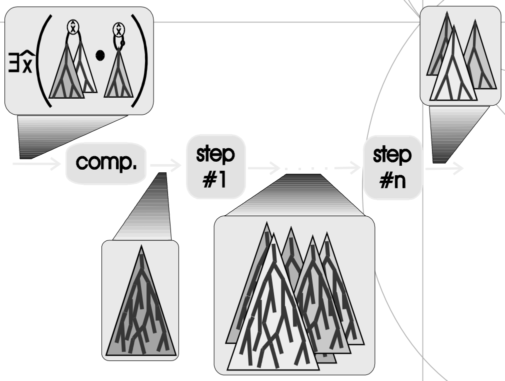
Boolean Function Decomposition

❖ **If the BDDs are too large to be represent**
- **State transition and output functions**
- **State sets**
- **Intermediate computations**

❖ **Then it is possible to *decompose and manipulate Boolean functions in decomposed form***







When/How to insert cut-points?

- ❖ Frequently
  - ◆ Good orderings do not exist or expensive
  - ◆ Sub-blocks require different orderings
- ❖ Auxiliary variables
  - ◆ Improve performances for poor orderings
  - ◆ Overcome conflicting requirements
- ❖ Selection criteria
  - ◆ Automatic structure-based insertion
  - ◆ Manual function based insertion

BDD Derivatives

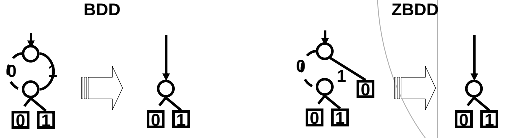
- ❖ MDD: Multi-valued BDDs
  - ◆ natural extension, have more then two branches
  - ◆ can be implemented using a regular BDD package with binary encoding
    - ✧ advantage that binary BDD variables for one MV variable do not have to stay together -> potentially better ordering
- ❖ ADDs: (Analog BDDs) MTBDDs
  - ◆ multi-terminal BDDs
  - ◆ decision tree is binary
  - ◆ multiple leafs, including real numbers, sets or arbitrary objects
  - ◆ efficient for matrix computations and other non-integer applications

- ❖ FDDs: Free BDDs
  - ◆ variable ordering differs
  - ◆ not canonical anymore
- ❖ … And many more …..

Zero Suppressed BDD's - ZBDD's

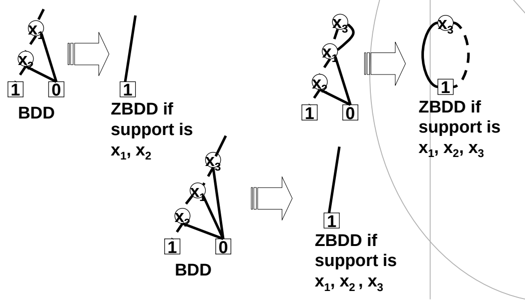
- ❖ ZBDD's were invented by Minato to efficiently represent sparse sets. They have turned out to be useful in implicit methods for representing primes (which usually are a sparse subset of all cubes)
- ❖ To sum up:
  - ◆ Minato, DAC'93
  - ◆ Change in the reduction rules
  - ◆ To represent Sparse sets
  - ◆ At most linear reduction to respect to BDD
  - ◆ Sufficient reduction in practice
  - ◆ Efficient representation for two and multi level logic minimization

- ❖ Different reduction rules
  - ◆ BDD
    - ✧ Eliminate all nodes where then edge and else edge point to the same node
  - ◆ ZBDD
    - ✧ Eliminate all nodes where the then node points to 0. Connect incoming edges to else node
  - ◆ For both
    - ✧ Share equivalent nodes



Canonicity

- ❖ Theorem (Minato)
  - ◆ ZBDD's are canonical given a variable ordering and the support set

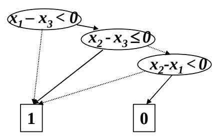


BDD
ZBDD if support is x_1, x_2
ZBDD if support is x_1, x_2, x_3
BDD
ZBDD if support is x_1, x_2 , x_3

Difference Decision Diagrams

- ❖ Møller, Lichtenberg, Andersen, Hulgaard, 1999
- ❖ DDD
  - ◆ Similar to BDDs, but the nodes are separation predicates
  - ◆ Ordering on variables determines order on predicates
  - ◆ Semi-canonical (i.e canonical when φ is a tautology or a contradiction)

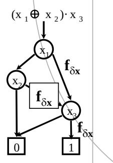
φ : $!(x_1 - x_3 < 0) \vee x_2 - x_3 \leq 0 \vee !(x_2 - x_1 < 0)$



- ◆ Each path leading to '1' is checked for consistency with 'Bellman-Ford'
- ◆ Worst case – an exponential no. of such paths

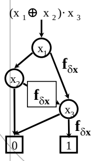
Functional DDs - OFDDs

- ❖ Kebschull & co., EDAC'92
- ❖ Reed-Muller Expansion (Negative and Posite Davio)
  - ◆ f = fØx Å x · (fx Å f Øx) = fx Å Øx · (fx Å fØx)
  - ◆ fdx =fx Å f Øx boolean difference
- ❖ For some functions are exponentially smaller than BDDs
- ❖ The reverse it is also true



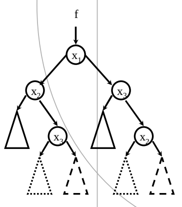
OKFDDs - Ordered Kronecker FDDs

- ❖ Drechsler & co., DAC'94
- ❖ Boole + Reed-Muller Decompositions
- ❖ Exponentially more compact than ROBDDs and OKFDDs
- ❖ Practice: Modest improvements on Average



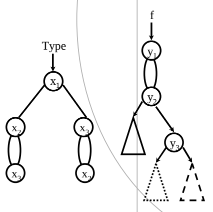
FBDDs - Free BDDs

- ❖ Meinel & co., T-Computer'94, Sieling & co., Theorical Computer Science'95
- ❖ Variables in any order - at most once in any path
- ❖ Non-canonical
- ❖ NP-hard checking equivalence



TFBDDs - Typed FBDDs

- ❖ Meinel & co., T-Computer'94, DAC'95
- ❖ FBDD + Type
- ❖ Canonical
- ❖ Same operational approach as BDDs
- ❖ How to find the Type?

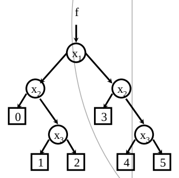


IBDDs - Indexed BDDs

- ❖ **Jain & co. EDAC’92**
- ❖ **Multiple occurrences of input variables**
- ❖ **multiplier Q(n3), hidden weighted bit Q(n2)**

MTBDDs - Multi Terminal BDDs or ADD - Arithmetic DDs

- ❖ **Clarke & co., DAC’93, Somenzi & co., ICCAD’93**
- ❖ **To represent Numeric-Valued Function**
- ❖ **Inefficient for large range**



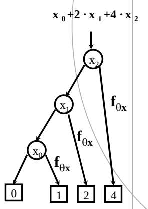
EVBDDs - Edge-Valued BDDs

- ❖ **Lai & co., DAC’92**

BMDs - Binary Moment Diagram

- ❖ **Bryant & co., DAC’95**
- ❖ **f = (1 - x) · f Øx + x · fx = f Øx + x ·(fx - f Øx)**
- ❖ **fqx = fx - f Øx linear moment**

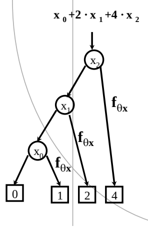
BMD & *BMD



*BMDs - Multiplicative BMDs

- ❖ **Bryant & co., DAC’95**
- ❖ **To reduce the number of nodes modify BMDs adding a weight to an edge**
- ❖ **Linear size in the number of inputs to represent addition, multiplication, exponentiation**

BMD & *BMD



$x_0 + 2 \cdot x_1 + 4 \cdot x_2$
x_2
x_1
x_0
$f_{\theta x}$
$f_{\theta x}$
$f_{\theta x}$
0
1
2
4