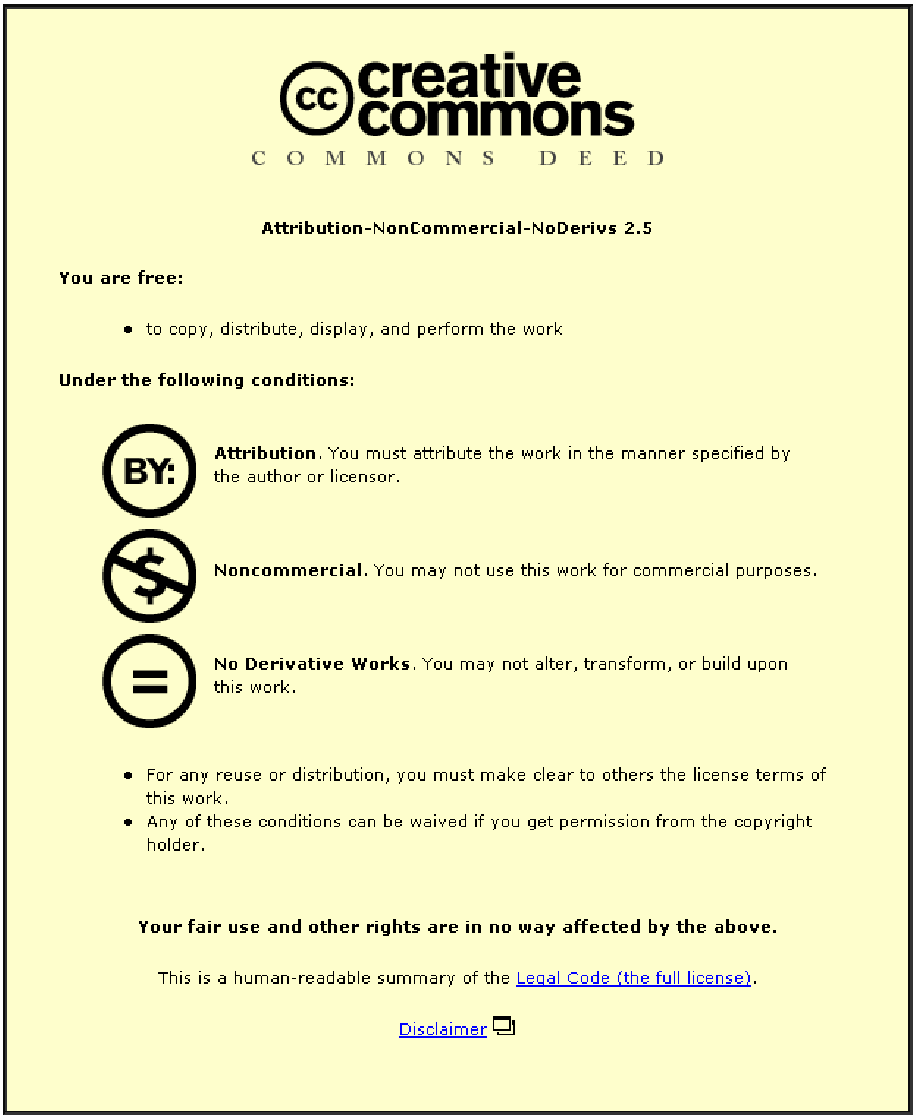


Progression of the Binary Decision Diagram Conversion Methods

Lisa Bartlett
Systems Engineering Department

Loughborough University

Content

- Background to Research.
- Brief introduction to Binary Decision Diagrams.
- Ordering Problems for Conversion.
- Conversion Methods:
  - Heuristic Approaches.
  - Ranking Methods.
  - Selection Procedures.
  - Progressive Ordering.
- Conclusions.

Background to Research

- Common risk assessment tool - Fault Tree Analysis.

- Commercial fault tree packages now readily available, use traditional kinetic tree theory for analysis.

- Latest development for fault tree analysis is the Binary Decision Diagram (BDD) methodology. More accurate, more efficient.

- Problems with conversion from fault tree to this BDD format. Advantages of technique can not be utilised.

Background to Research

- For practitioners to use BDD technique implementation into commercial software requires alleviation of conversion problem.

- Numerous conversion methods available.

- Discussion of four main areas.

- Ongoing research.

BDDs – Reasons for Introduction

- Fault Tree Analysis is a commonly used means of assessing system reliability performance.

- This analysis technique has limitations, especially when dealing with large fault tree structures.

- Most successful recent development is the Binary Decision Diagram (BDD) approach.

- Shown to improve both the efficiency of determining the minimal cut sets of the fault tree and accuracy of the calculation procedure used to determine the top event parameters.

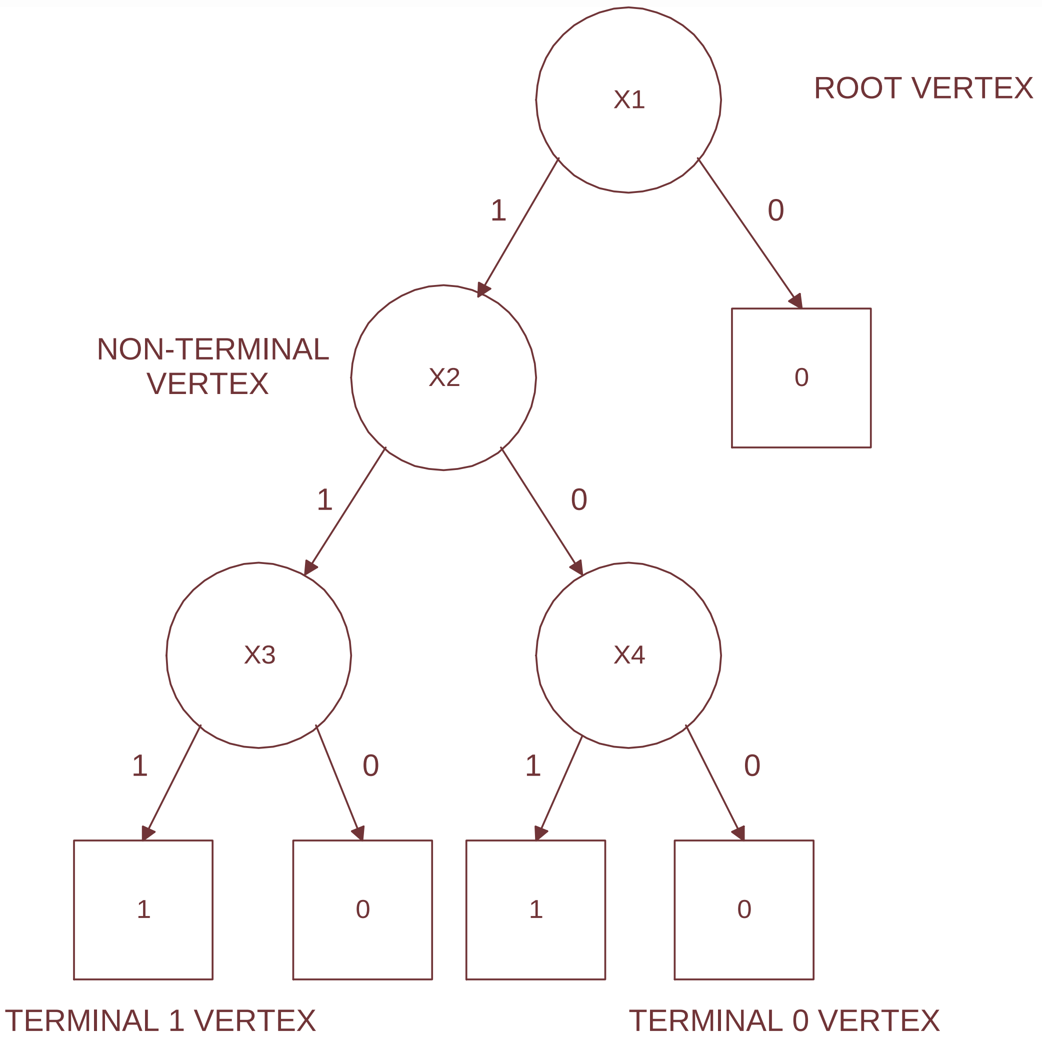
Using the BDD

- To utilise the BDD approach, the fault tree must be converted to BDD format.
- Conversion process relatively easy to implement, requires basic events of fault tree to be placed in an order.
- Ordering scheme chosen is critical to the size of the BDD produced and hence the efficiency of the technique.
- In BDD format, usual qualitative and quantitative analysis can be performed.

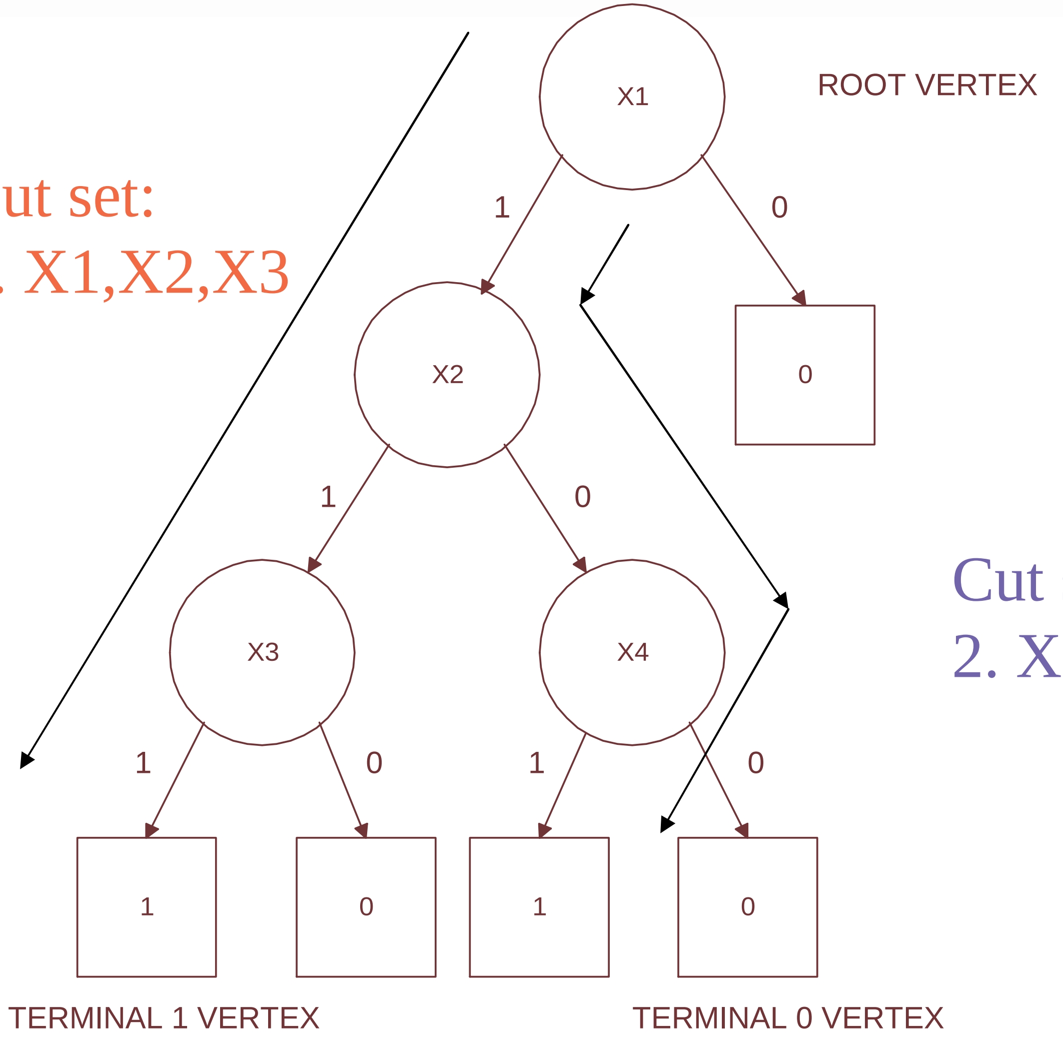
BDD Architecture

- Directed acyclic graph.
- All paths through the BDD start at the root vertex.
- Paths terminate in one of two states
  - 1 state		system failure
  - 0 state		system success
- BDD composed of terminal and non-terminal vertices, which are connected by branches.
- Non-terminal vertices correspond to the basic events of the fault tree. Terminal vertices represent system state.

BDD Architecture continued



BDD Architecture - Cut Sets



Cut set:
1. X1,X2,X3

Cut set:
2. X1,X4

BDD Construction

- Fault tree must be converted to the appropriate diagram.

- Two methods used:

  1) Using Top Event Logic Function

  2) Using If-Then-Else Method

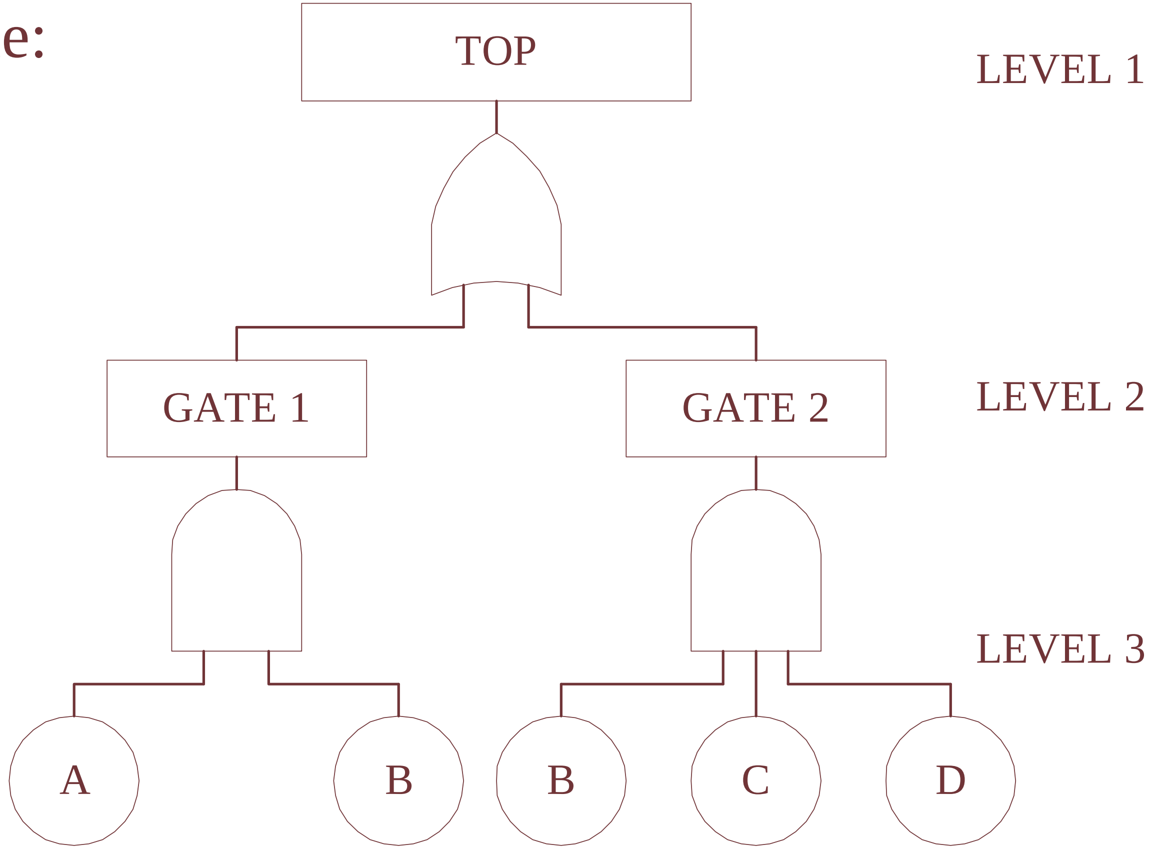
BDD Construction - Method 1

- Logic function representing the top event of the fault tree can be used to generate the BDD.

- Process involves for each node/vertex:
  - Substituting in the value of 1 (component fails)
  - Substituting in the value of 0 (component functioning)

BDD Construction - Method 1

- Example:



- Top event logic function: T = A.B + B.C.D

BDD Construction - Method 1

- First ordering decided upon: $A < B < C < D$

- Select each variable in turn.

- Assign the value of 1 to the event, to consider the effect on the top event when the component has failed. Repeat considering component functioning.

- Process repeated with next variable on resultant logic functions.

- Process continued until terminal vertices reached.

BDD Construction - Method 1

- Consider first variable: A

- Consider failure of A:

  Logic expression is: (1).B + B.C.D = B + B.C.D

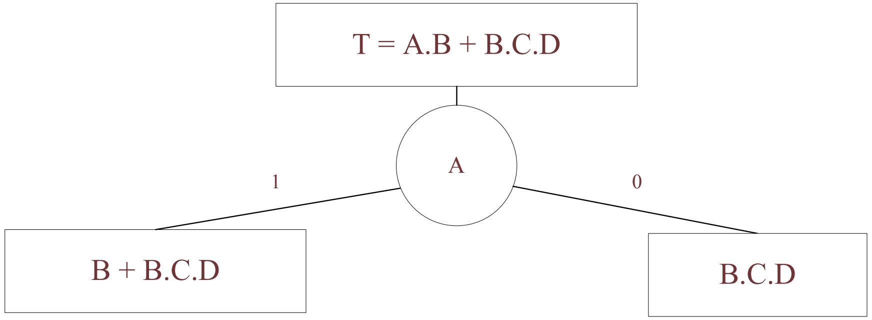
- Consider functioning of A:

  Logic expression is: (0).B + B.C.D = B.C.D

BDD Construction - Method 1

- Root vertex of BDD is: A

- BDD so far looks like:



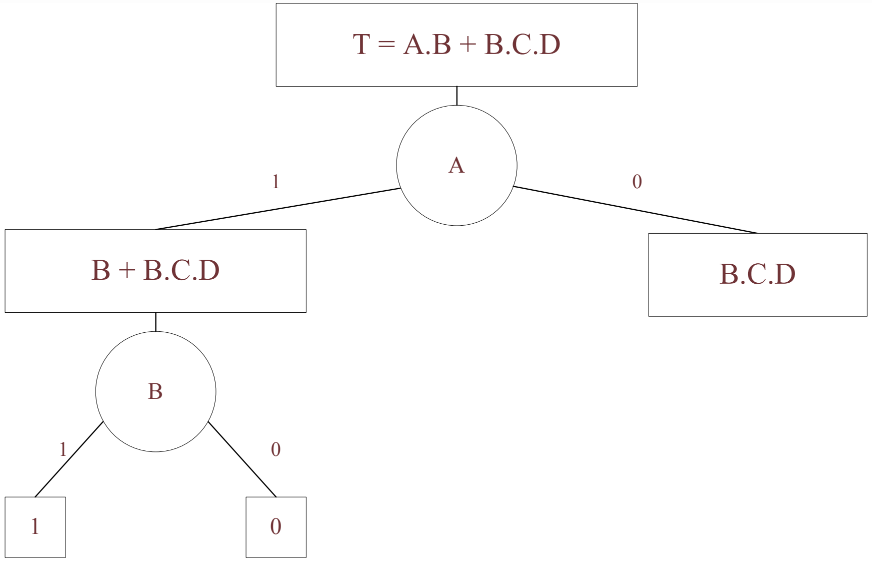
- Outcome on 1 branch of A: B + B.C.D

  Consider next variable: B

- Failure of B: 1 + (1).C.D = 1
- Functioning of B: 0 + (0).C.D = 0

BDD Construction - Method 1

- BDD is now:



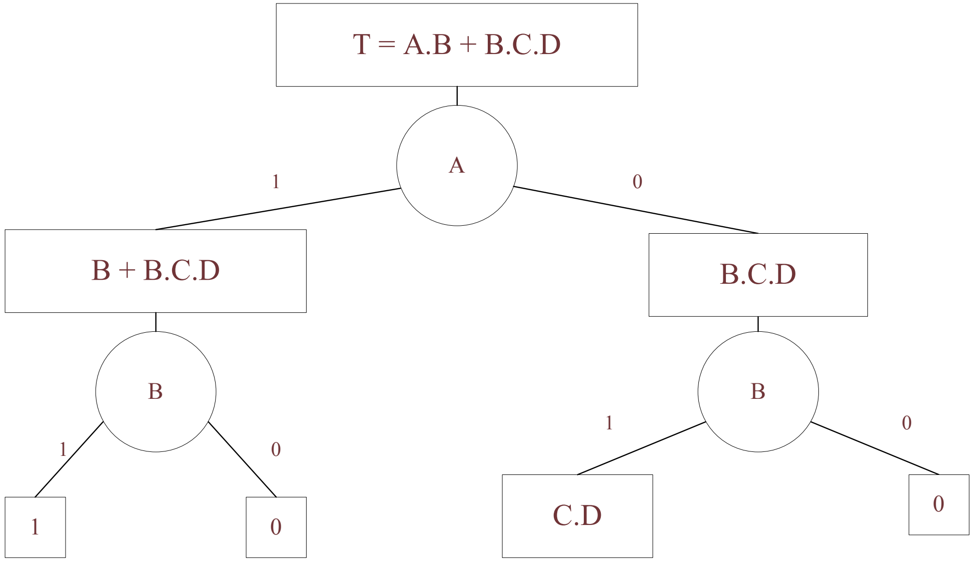
- Outcome on 0 branch of A: B.C.D

  Consider next variable: B

- Failure of B: (1).C.D = C.D
- Functioning of B: (0).C.D = 0

BDD Construction - Method 1

- BDD is now:



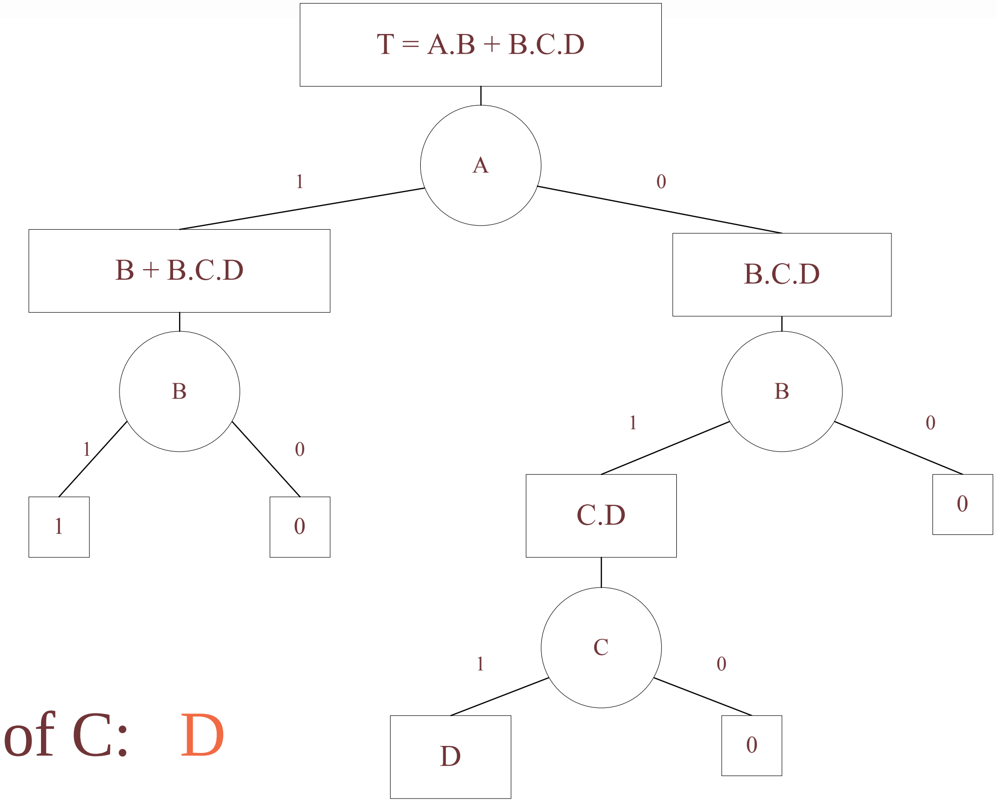
- Outcome on 1 branch of B: C.D

  Consider next variable: C

- Failure of C: (1).D = D
- Functioning of C: (0).D = 0

BDD Construction - Method 1

- BDD is now:

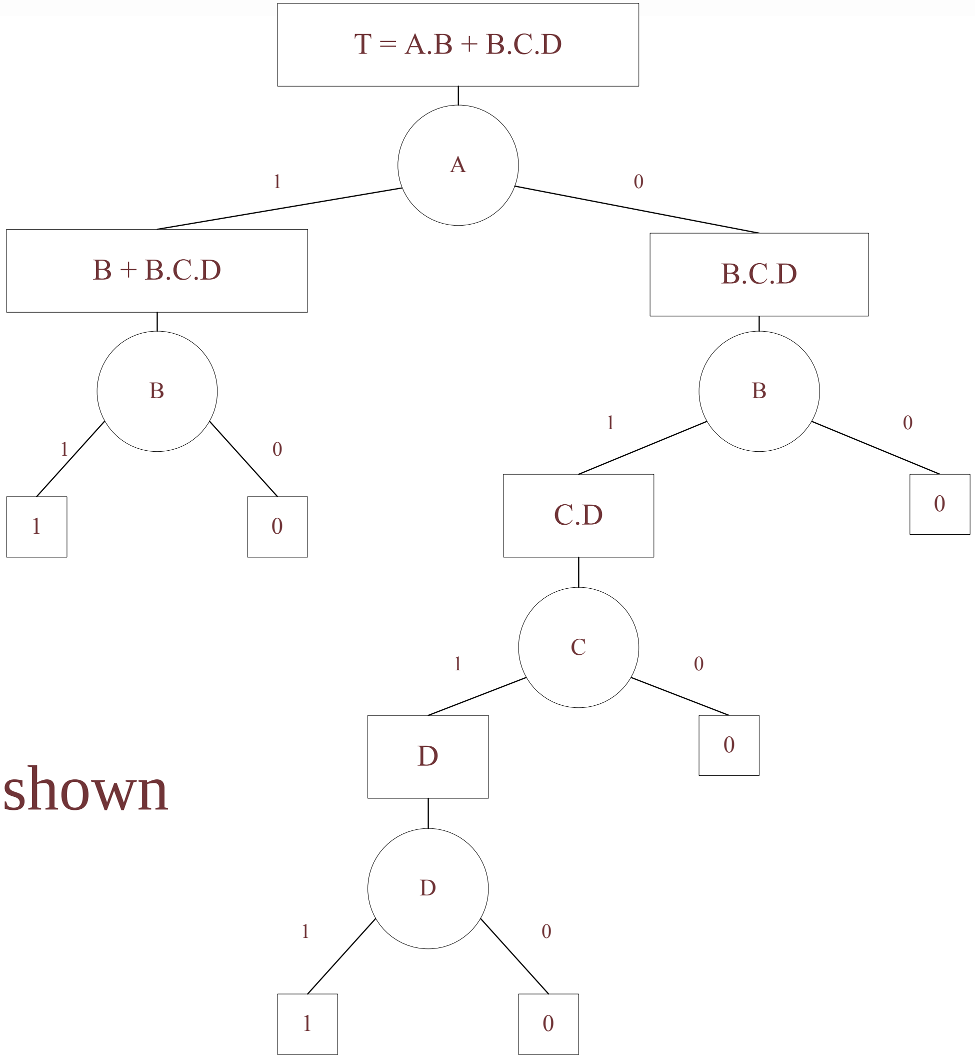


- Outcome on 1 branch of C: D

- BDD for the final logic expression will lead directly to a 1 and 0 terminal vertices.

BDD Construction - Method 1

- Resulting BDD:



Boolean equations shown on diagram

Ordering Problems

- In constructing the BDD the ordering of the basic events is crucial to the size of the resulting diagram.

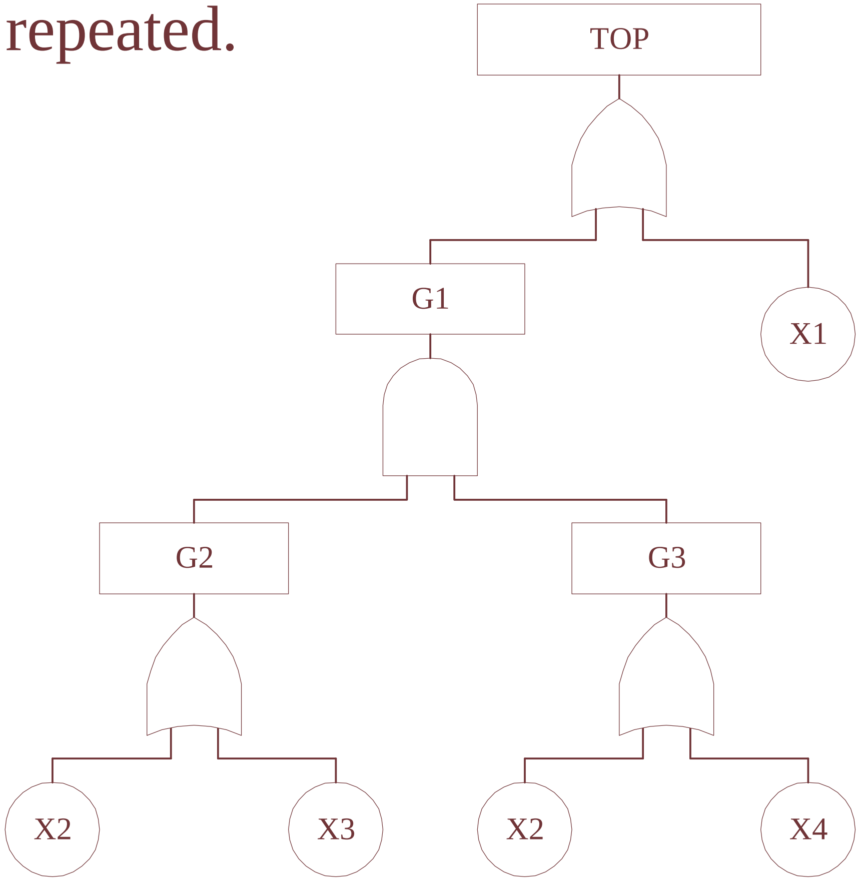
- Using an inefficient ordering scheme will produce a non-minimal BDD structure.

- Different ordering schemes will produce BDDs of different sizes.

- The smaller the BDD the more optimal the diagram.

Ordering Problems - Example

- Consider this simple fault tree. The tree has four basic events where X2 is repeated.

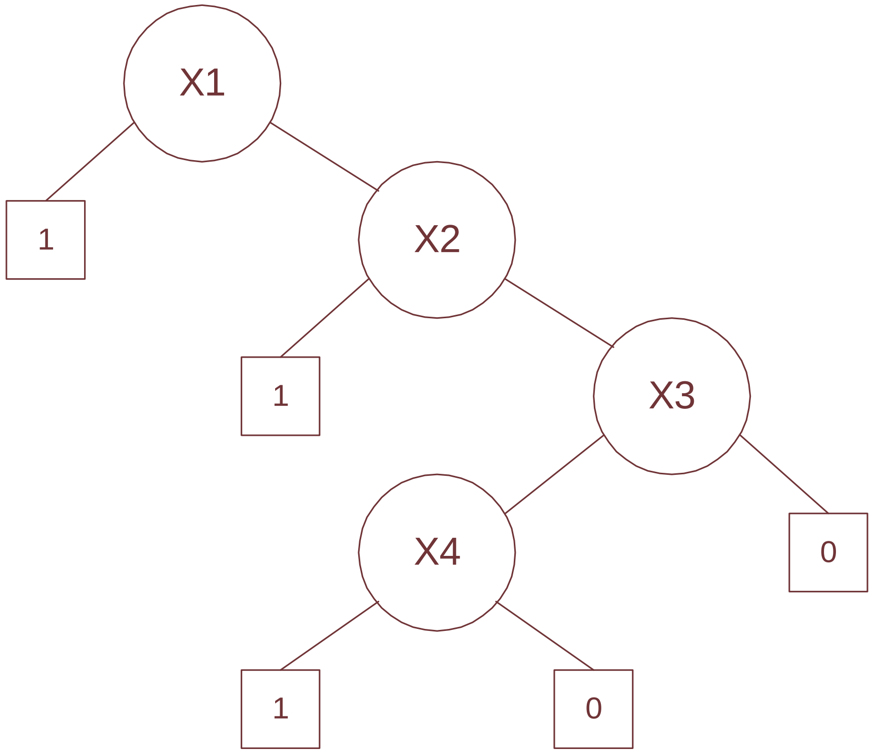


Min Cut Sets:
X1
X2
X3.X4

Ordering Problems - Example

- Using the basic event ordering permutation:

  X1 < X2 < X3 < X4

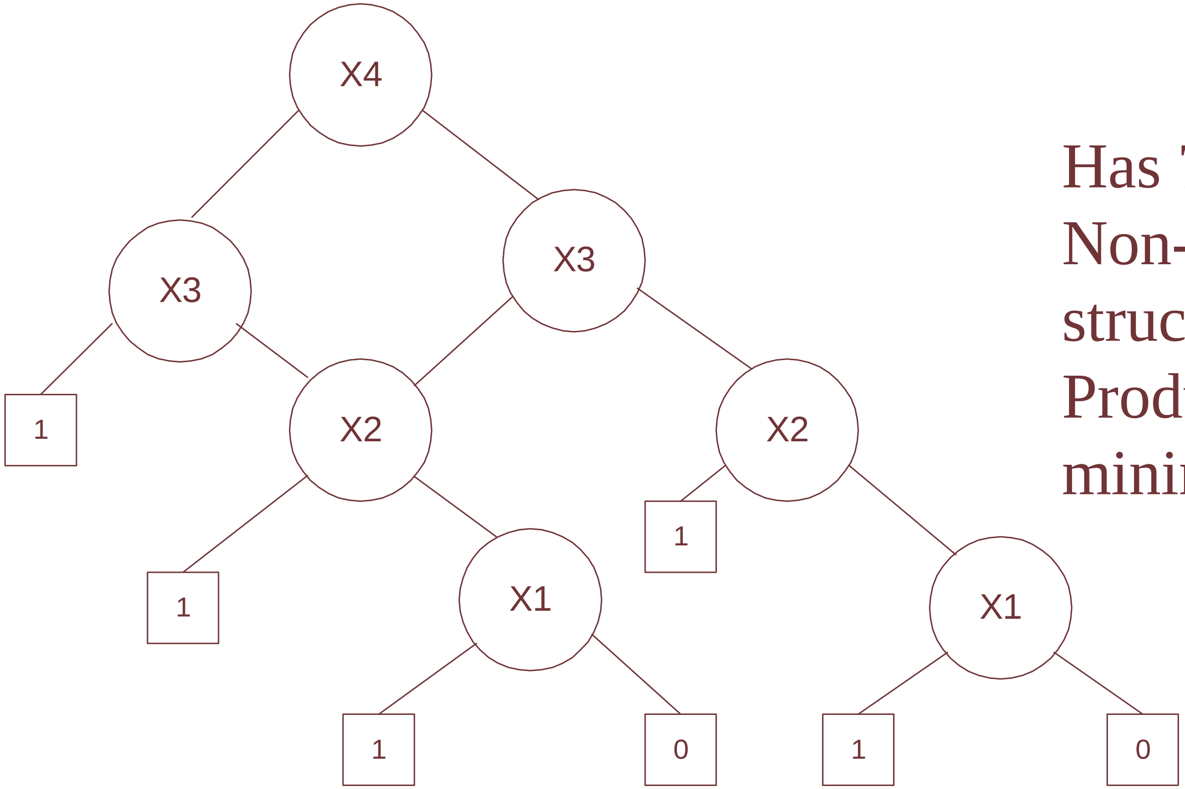


Only has 4 nodes.
Minimal structure.
Produces only
minimal cut sets.

Ordering Problems - Example

- Using the basic event ordering permutation:

X4 < X3 < X2 < X1



Has 7 nodes.
Non-Minimal structure.
Produces non-minimal cut sets.

Ordering Problems - Example

- For larger fault trees the resultant BDD would be much larger.

- Non-mimimal BDDs need to undergo minimisation procedure to obtain minimal cut sets which can cause an undesirable increase in computer time.

- In the worst case of using a poor ordering permutation, the diagram may be unsolvable.

Ordering Problems - Example

- Objective to produce ordering scheme that achieves the ‘best’ BDD for any fault tree, to utilise advantages of BDD technique, allow use in commercial package.

- To date, no universal scheme which will guarantee ‘best’ BDD.

- Major developments in this research area are the focus of this presentation.

Ordering Methodologies

- Application in two domains – circuit analysis and fault tree analysis.
- Due to differences in logic function for circuits ordering methods for this domain are not applicable for FTA.
- Two types of approach:
  - Static – generate an ordering before conversion.
  - Dynamic – generate BDD using static method, then exchange variables to make smaller.

Ordering Methodologies

- Dynamic methods:
  - Focus mainly on circuit diagrams although have been applied to fault trees.
  - Generate a BDD using a static method and then involve swapping the variables to make the diagram smaller.
  - Can reduce the size of the BDD.
  - Limited use in reliability analysis due to the time taken for implementation.

Ordering Methodologies

- Static methods:

  - Static ordering methods produce a variable ordering prior to the construction of the BDD.
  - Two categories: structural and weighted.

  - **Structural schemes** perform an organised traversal of the tree, ordering variables as they are encountered.
  - Preserve the neighbourhood properties of the tree.

  - **Weighted methods** allocate weights to the variables.
  - Do not necessarily preserve the neighbourhood.

Ordering Methodologies

- Common approach – static heuristic methods.
- Latter advances in three main directions:
  - Ranking methods.
  - Selection mechanisms.
  - Progressive ordering.

Heuristic Approaches

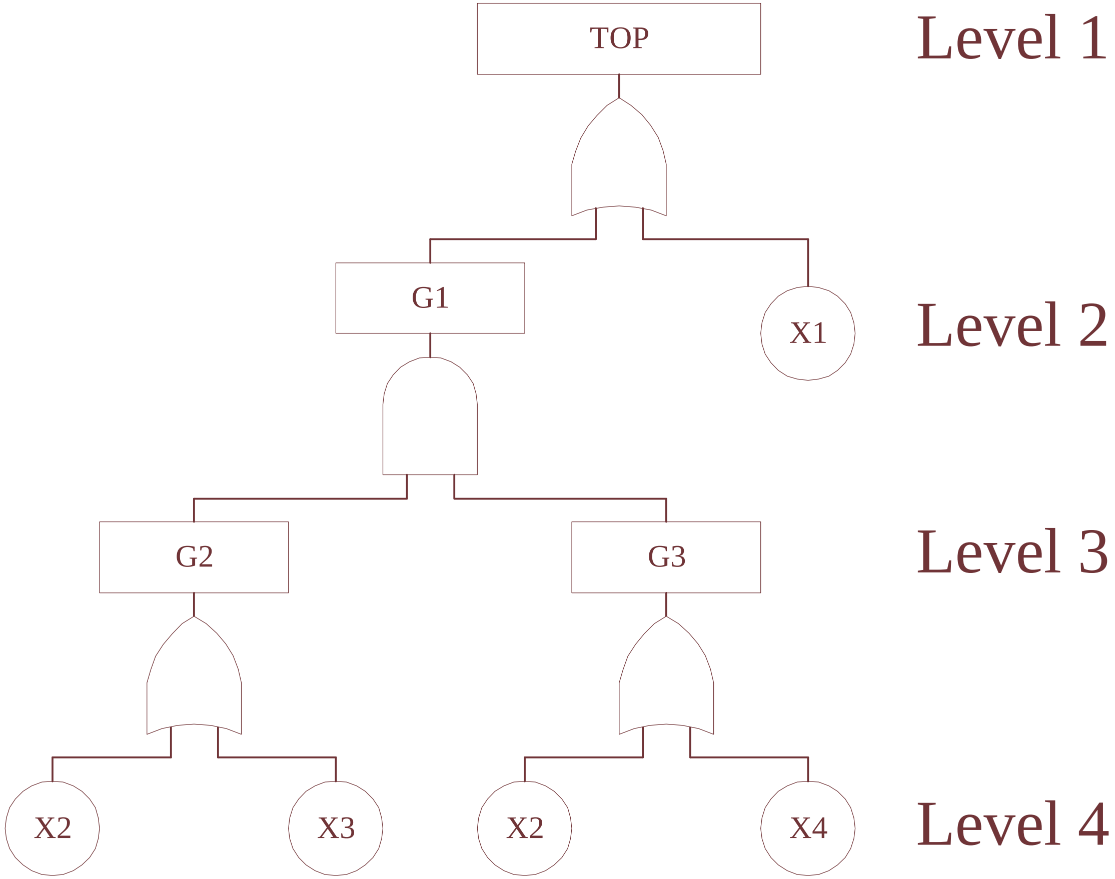
- Structured methodology but based on no mathematical foundations.
- Most common technique obtained by listing the basic events in a top-down, left-right manner.
- BDDs produced using method often are non-minimal.
- Non-minimal BDD must undergo minimisation procedure to obtain minimal cut sets – undesirable increase in computer time.

Heuristics – Top-down, left-right

Ordering:

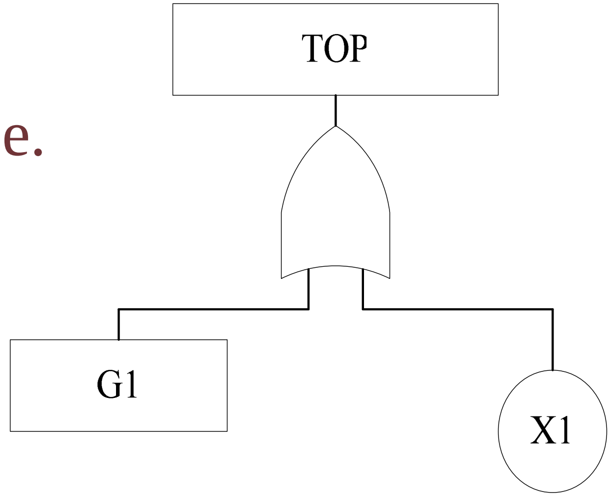
X1 < X2 < X3 < X4

Does produce minimal BDD in this case



Heuristic Approaches – Depth-first

- Alternative approach – depth-first.
- Top-down, left-right principle applied to subtrees of the whole tree separately.
- Each top event input forms a subtree.
- For previous tree – {Gate 1}
{X1}
- Resultant ordering: X2 < X3 < X4 < X1
- BDD non-minimal, 5 nodes and 4 cut sets.



Heuristic Approaches - Comparison

- Research by Sinnamon and Andrews compared six heuristic approaches.

- Vast differences were found in number of computations needed to construct BDD.

- Highlighted that each tree has individual variable ordering that will optimise its size.

- Also general approach that will be ‘best’ for all trees does not seem apparent.

- Supported by Bouissou.

Heuristic Approaches – Pros and Cons

- **Pros:**
  - Simplistic.
  - Easy to implement.
- **Cons:**
  - Affected by how fault tree is drawn.
  - Structured pattern does not allow for different branched events to be positioned next to each other in list.

Ranking Methods

- Considering inadequacies of heuristic approaches, properties required in good ordering method seem to be:
  - Contribution event makes to occurrence of top event should be reflected.
  - Ordering must be robust.
  - Uniquely map the fault tree onto a single event ordering.
- Ranking method to be discussed uses importance measures (conforms to the first two points).

Ranking Methods – Structural Importance

- Importance Measure:

  Signifies the role that a component or minimal cut set plays in either causing or contributing to the occurrence of the top event.

- The role is given a rank in terms of a numerical value.

- Deterministic measure used – structural importance (SMI).

$$SMI_i = \frac{number\ \ of\ \ critical\ \ system\ \ states\ \ for\ \ component\ \ i}{total\ \ number\ \ of\ \ states\ \ for\ \ the\ (n-1)\ \ remaining\ \ components}$$

Application of Ranking Method

- To calculate number of critical states computationally intensive.

- Alternative is to use Lamberts probabilistic importance measure (B_i).

$$B_i = \{ Q(1_i, \underline{1/2}) - Q(0_i, \underline{1/2}) \}$$

- Technique tested on 225 fault trees.

- 77 % of trees yielded BDD of equal or smaller size than best of six alternative heuristic methods.

Pros and Cons of Ranking Method

- **Pros:**
  - Indicates small BDD possible for large number of trees.
- **Cons:**
  - Requires calculation of probability expression for failure mode.
- To warrant use in commercial package, performance needs improvement and derivation also.

Selection Procedures

- Capability of producing a minimal BDD for a specific fault tree is present with current heuristics / approaches.

- Problem is finding heuristic / approach with this capability.

- Mechanism to achieve this is a selection procedure.

- Basis is a rule based pattern recognition approach.

- Approaches researched:

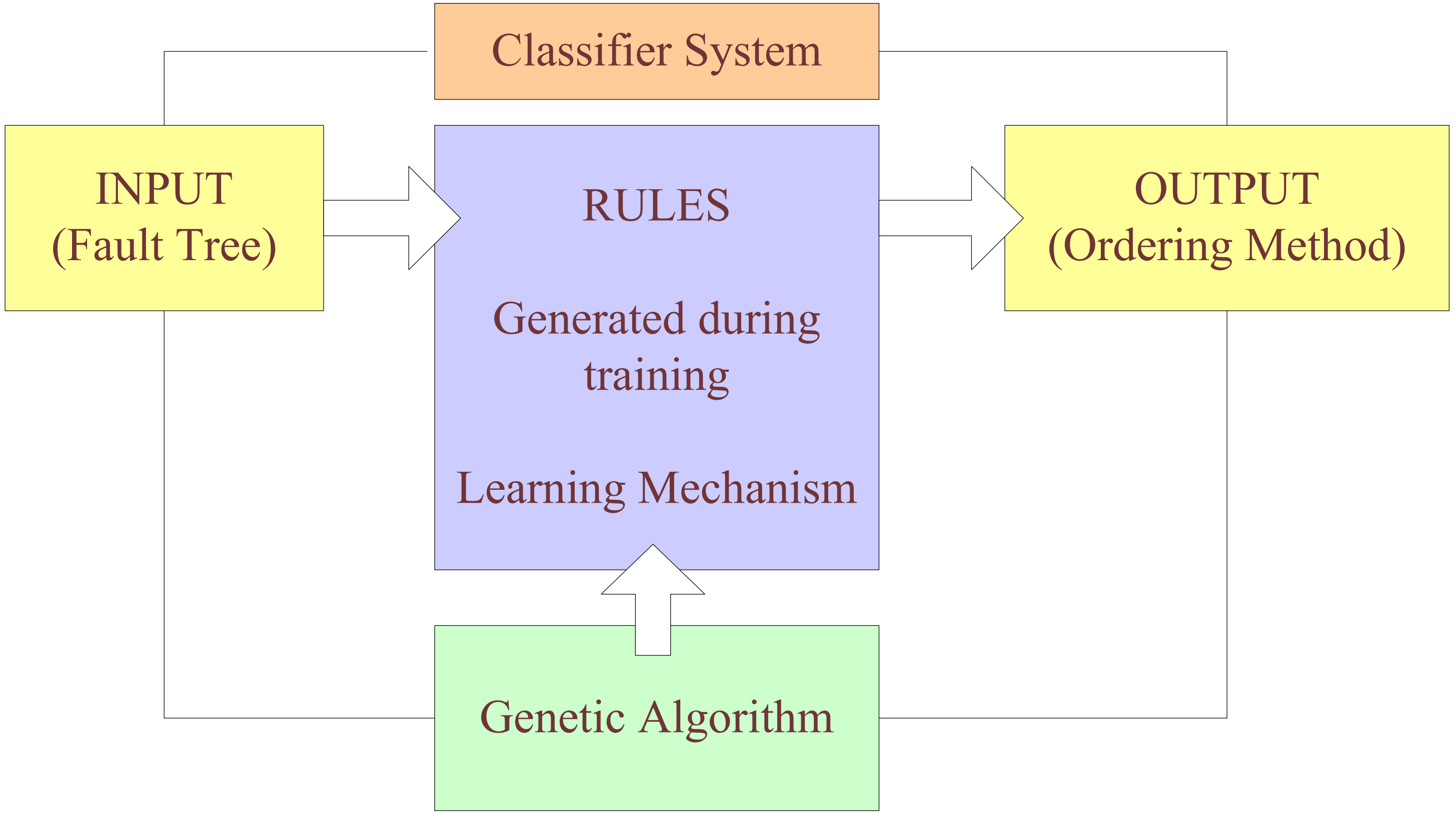
  - Classifier systems
  - Neural networks

Selection Procedures - Fundamentals

- A set of possible ordering heuristics are used.
- A specific heuristic can be selected depending on fault tree characteristics.
- Classifier systems and neural networks are procedures to determine links between heuristic and fault tree characteristic.
- End result a predictive mechanism for selecting appropriate heuristic for minimal BDD generation.

Selection Procedures

- **Classifier system:**



Classifier System

INPUT (Fault Tree)

RULES

Generated during training

Learning Mechanism

OUTPUT (Ordering Method)

Genetic Algorithm

Selection Procedures

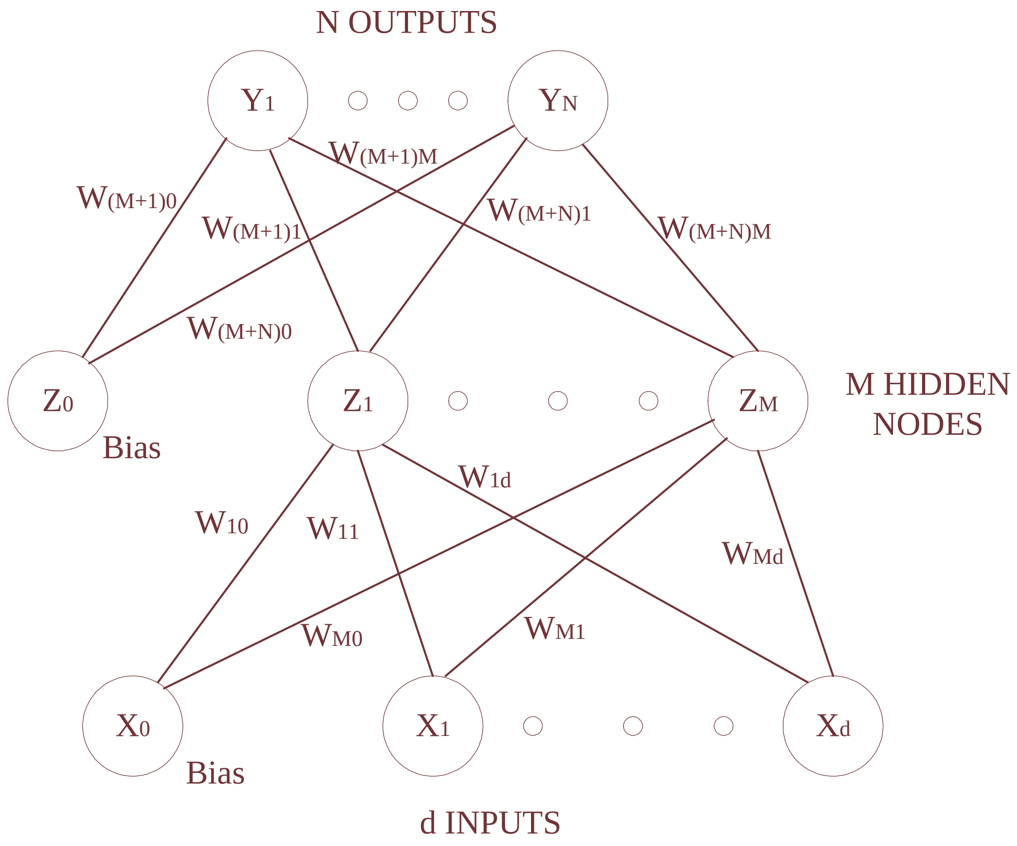
- **Classifier system:**
  - Produce a set of rules through training.
  - Shown potential.
  - Requires more fault tree structures to finalise rules.
  - Rule generation method needs enhancement to meet complexity of problem.

Selection Procedures

- **Neural Networks:**
  - Multi-layer networks.
  - Radial basis functions.
- **Architecture:**
  - Input layer (representing fault tree)
  - Output layer (scheme choices)
  - Middle layers (pattern recognition potential)
  - Link between these in the form of weighted connections.

Selection Procedures

- **Neural Networks (**Multi-layer networks)



Results:
70% success rate on test set of 20 trees.

30% of incorrect predictions second best BDD.

Ads / Disads of Neural Networks

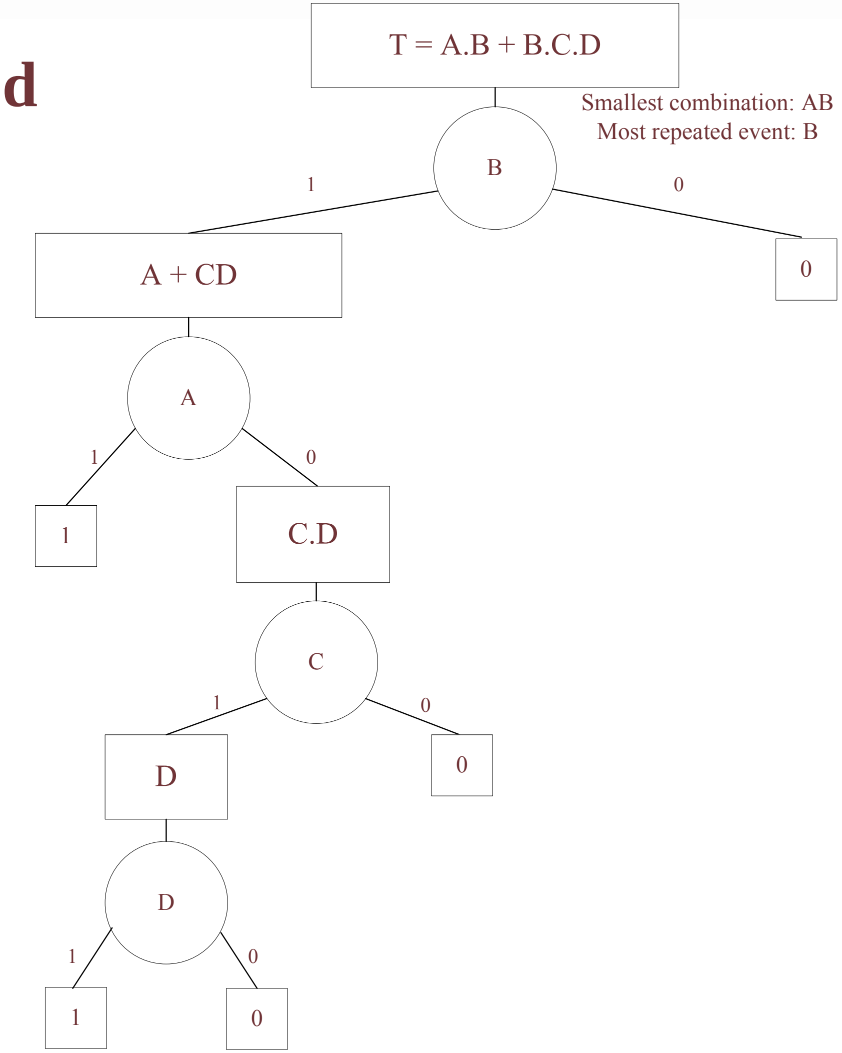
- **Advantages:**
  - Range of ordering methods can be used.
- **Disadvantages:**
  - Doesn’t currently have predictive potential for use in commercial packages.
- Methods to improve predictive capability are under investigation.

Progressive Ordering

- Latest development involves generating an ordering at each phase of conversion, i.e. after each node in the BDD is formed (Rauzy et al.).
- Starts with the logic function of the failure mode.
- Chooses basic events which appear in the smallest combination first.
- Repeated basic events given priority, next smallest combinations used in event of ties.
- Process repeated with resulting logic function at each node.

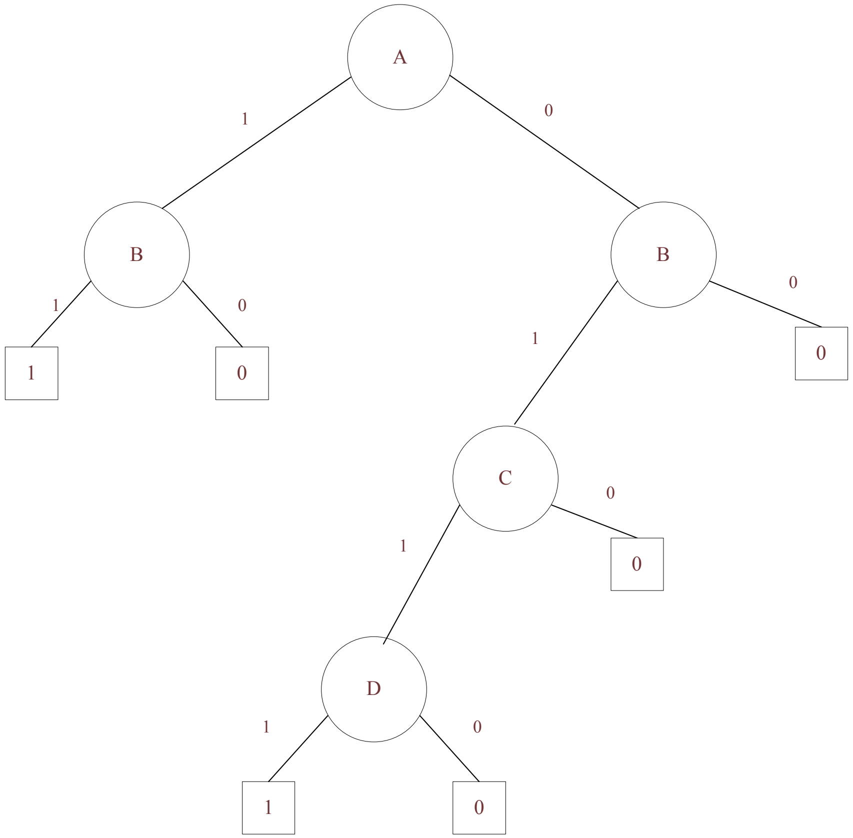
Progressive Ordering – Pros and Cons

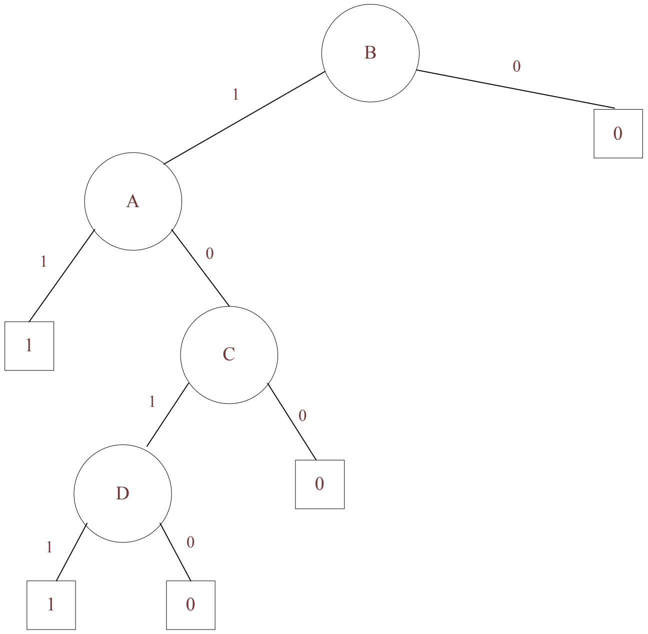
- **Progressive Method**



Progressive Ordering – Comparison

- **Static Method** **Progressive Method**





Progressive Ordering – Pros and Cons

- **Pros:**
  - Method easy to apply.
  - Results good.

- **Cons:**
  - Starting point is logic function which incorporates minimal cut sets.

- Method shows potential yet application needs improvement.

- Fundamentals of approach combined with acceptable means of application area of ongoing research.

Conclusions

- Progression in three main areas:
  - Individual heuristic methods.
  - Selection mechanisms.
  - Progressive ordering.

- Large number of heuristic approaches, simple to use, yet minimal BDD not guaranteed (approx 30% chance).

- Structural importance method has greater potential for producing desired end product, yet derivation method needs work.

Conclusions

- Avenues of most potential for use of the BDD technique in commercial packages are through selection mechanisms and progressive ordering techniques.

- Developments are evolving in a direction that will allow this technique to be integrated into software in an efficient manner.