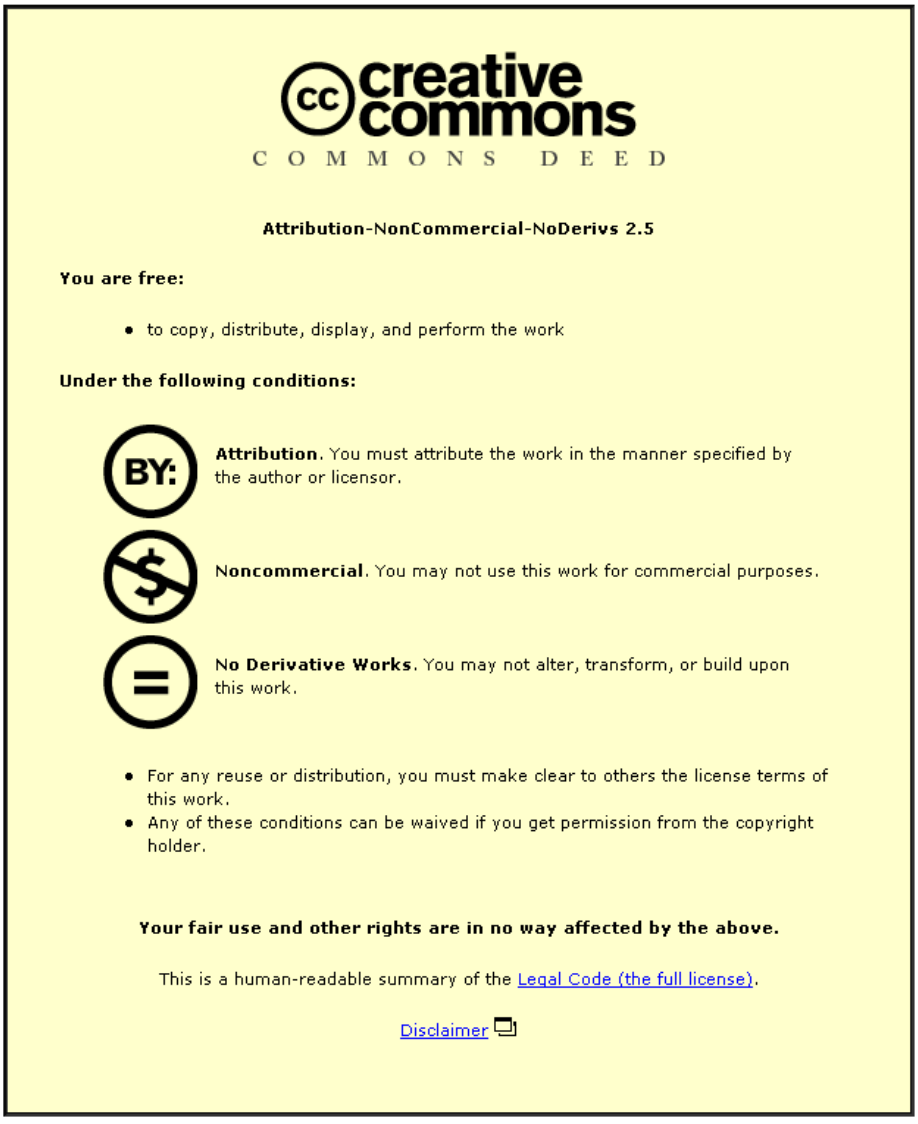

creative commons

COMMONS DEED

Attribution-NonCommercial-NoDerivs 2.5

You are free:

- to copy, distribute, display, and perform the work

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor.

Noncommercial. You may not use this work for commercial purposes.

No Derivative Works. You may not alter, transform, or build upon this work.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the Legal Code (the full license).

Disclaimer

A binary decision diagram method for phased mission analysis of non-repairable systems

S J Dunnett and **J D Andrews***
Department of Aeronautical and Automotive Engineering, Loughborough University, Loughborough, UK

The manuscript was received on 9 March 2006 and was accepted after revision for publication on 19 May 2006.

DOI: 10.1243/1748006XJRR27

Abstract: Phased mission analysis is carried out to predict the reliability of systems which undergo a series of phases, each with differing requirements for success, with the mission objective being achieved only on the successful completion of all phases. Many systems from a range of industries experience such missions. The methods used for phased mission analysis are dependent upon the repairability of the system during the phases. If the system is non-repairable, fault-tree-based methods offer an efficient solution. For repairable systems, Markov approaches can be used.

This paper is concerned with the analysis of non-repairable systems. When the phased mission failure causes are represented using fault trees, it is shown that the binary decision diagram (BDD) method of analysis offers advantages in the solution process. A new way in which BDD models can be efficiently developed for phased mission analysis is proposed. The paper presents a methodology by which the phased mission models can be developed and analysed to produce the phase failure modes and the phase failure likelihoods.

Keywords: phased mission analysis, fault tree analysis, binary decision diagrams, non-repairable systems

1 INTRODUCTION

Many systems undergo phased missions when performing the functions for which they are designed. A phased mission is one where the overall mission can be split into a number of phases, performed in sequence, in which the objectives change from one phase to the next. The mission is successfully completed if all phases are successfully accomplished. For example an aircraft flight could be considered as a mission with phases: taxiing to the runway, take-off, ascend to the correct altitude, cruise, descend, land, and taxi back to the terminal. Each phase uses differing functional elements of the system and so the causes of failure in each phase will be different.

**Corresponding author: Department of Aeronautical and Automotive Engineering, University of Loughborough, Loughborough, Leicestershire LE11 3TU, UK. email: J.D.Andrews@lboro.ac.uk*

In such circumstances, mission reliability cannot be determined by studying the individual phases of the mission in isolation, for two reasons.

1. The phase failure probabilities are not statistically independent. Some components or subsystems influence the success, or failure, of more than one phase.
2. It cannot be assumed that all components are functioning at the start of each phase, even if they can be assumed functional at the start of the mission. In the example of an aircraft flight, if the landing gear fails following take-off during the ascend, cruise, or descend phases, it will remain that way until the landing phase, when the mission will fail. Failures can remain unrevealed in some phases and mission failure will occur not when the failure events happen but on the point of transfer from one phase to another.

The methods for conducting systems reliability assessments for phased missions utilize different techniques depending on whether the system can

be repaired or not. For systems which are non-repairable throughout the duration of the mission, such as an aircraft flight, fault tree methods offer the most efficient solution. Esary and Ziehms [**1**] used fault trees to represent the failure conditions for each mission phase which can be combined and, with the use of a minimal cut set reduction method, evaluated to obtain the mission unreliability. A component's failure during each phase of the mission was considered as a separate event. This significantly increases the size of the problem for analysis. Zang *et al*. [**2**] extended the fault-tree-based method to utilize binary decision diagrams (BDDs) as a more effective representation of the phase failure logic for efficient analysis.

By considering the failure of each phase individually, the problem complexity can be reduced [**3**]. The conditions for a failure in any specified phase occur when all previous phases have completed successfully, and then either the failure conditions occur, causing in-phase failure, or the failure conditions already exist when the phase is entered, causing phase transition failure. This approach again employs BDDs and develops a new event algebra to account for the non-independence of the component failure events from one phase to the next. In addition to the overall mission unreliability, the method also yields the phase unreliabilities, which can be used in risk studies where the consequences of failure differ from phase to phase. (For example, consider the consequences of failure in the taxiing phase or the cruise phase of an aircraft flight.)

Where components can fail and be repaired during the mission, Markov methods are used to calculate the mission failure probability [**4**–**6**].

The research reported in this paper develops an efficient, systematic generation of the mission BDD for phased missions. While using a single model to represent the conditions for mission failure, as with the method of La Band and Andrews [**3**], the phase failure probabilities can also be deduced.

2 BINARY DECISION DIAGRAMS

BDDs offer an alternative approach to fault trees to represent a system's failure logic. Their structure enables the system failure probability to be deduced efficiently and without resort to any approximations. However, they are difficult to construct directly from the system definition and are generally obtained by converting from a fault tree.

A BDD is a directed acyclic graph with all paths starting at the top root vertex and terminating in one of two terminal vertices, 1 or 0. A terminal 1 vertex indicates top event occurrence, and a terminal 0 vertex indicates top event non-occurrence. The diagram consists of terminal and non-terminal vertices connected by branches; the non-terminal vertices correspond to basic events. Each non-terminal vertex has two branches leaving it, a 1 branch and a 0 branch, which indicate occurrence and non-occurrence respectively of the basic event. The construction of the BDD requires the basic events to be ordered. In Fig. 1 an example of a BDD is shown: it represents the logic function $A.B + B.C$ with the ordering $A < B < C$ assumed (note that the symbol $+$ represents logical OR, and the symbol . represents logical AND).

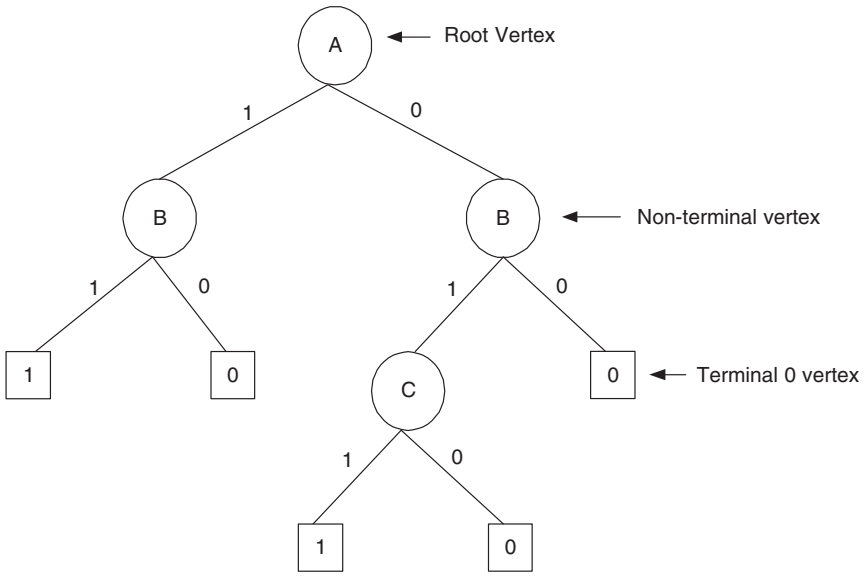

Fig. 1 BDD

Paths through the BDD indicate the component conditions which determine the system state as given by the terminal node. So, in the BDD shown in Fig. 1, if A fails and B fails, a terminal 1 node is reached, which means that, under these conditions, system failure occurs (i.e. this specifies a cut set).

3 BINARY DECISION DIAGRAM CONSTRUCTION

The conventional rules used for constructing BDDs from fault trees have been covered extensively in previous papers [**7**, **8**] and will therefore only be summarized here. From the structure of the BDD shown in Fig. 1 it can be seen that each intermediate node can be represented as an if–then–else (ite) triple of the form

$$\text{ite}(X, f_1, f_2) \tag{1}$$

where X is a Boolean variable, f_1 is a logic function to consider if X is true (on the 1 branch) and f_2 is a logic function to consider if X is false (on the 0 branch). When combining two logic functions expressed in this way, the rules are as follows: if $h = \text{ite}(X, h_1, h_2)$ and $k = \text{ite}(Y, k_1, k_2)$ are two logic functions, then

$$h \oplus k = \begin{cases} \text{ite}(X, h_1 \oplus k_1, h_2 \oplus k_2), & X = Y \\ \text{ite}(X, h_1 \oplus k, h_2 \oplus k), & X < Y \end{cases} \tag{2}$$

where $\oplus$ represents the AND/OR logic operator.

The method used for constructing the BDDs in this paper is one whereby the BDDs for each phase in the mission can be constructed separately and then combined to form the mission BDD. It is accomplished by noting the structures which result when inputs for each gate type are combined. For example, consider the BDDs representing the logic expressions $A + B + C$ and $A.B.C$ shown in Figs 2(a) and (b) respectively, where the ordering $A < B < C$ has been assumed.

For an OR gate the BDD structure forms a chain where the basic events are linked by their 0 branches. The AND gate BDD structure has the basic events linked on their 1 branches. This concept can also be extended to combining logical expressions which are inputs to fault tree gates. For example, if h and k are logic expressions which are inputs to an OR gate, then the BDD representing the combined logical expression can be obtained by replacing the terminal 0 vertices in the BDD representing the first logical expression by the BDD representing the second logical expression. For inputs to an AND gate, each terminal 1 vertex is replaced by the second BDD. Once the combined BDD is formed, prior to simplification, it is possible that the ordering assigned to the basic event variables will be destroyed and some variables may appear more than once on paths from the root vertex to the terminal nodes. Care must then be taken to deal with repeated variables on the same path. If by combining BDDs there is now, in the new BDD, a variable which also appears earlier in the structure, this repeated variable can be removed. The first occurrence of the variable is the one which fixes its state. So, if on some path a variable is first passed through on its 1 branch, this fixes that the component is in its failed state on this path. For the second occurrence, the node is omitted and replaced with a link directly to the node on its 1 branch; everything below its 0 branch is deleted. If the path first passes the basic event on its 0 branch, the component failure has not occurred on this path, the second occurrence of the node is replaced by a

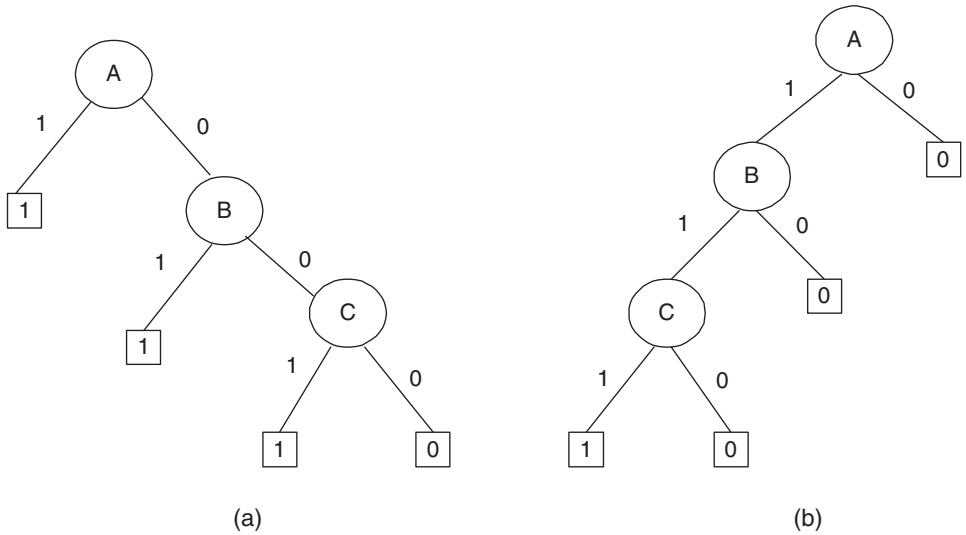

Fig. 2 BDDs representing (a) $A + B + C$ and (b) $A.B.C$

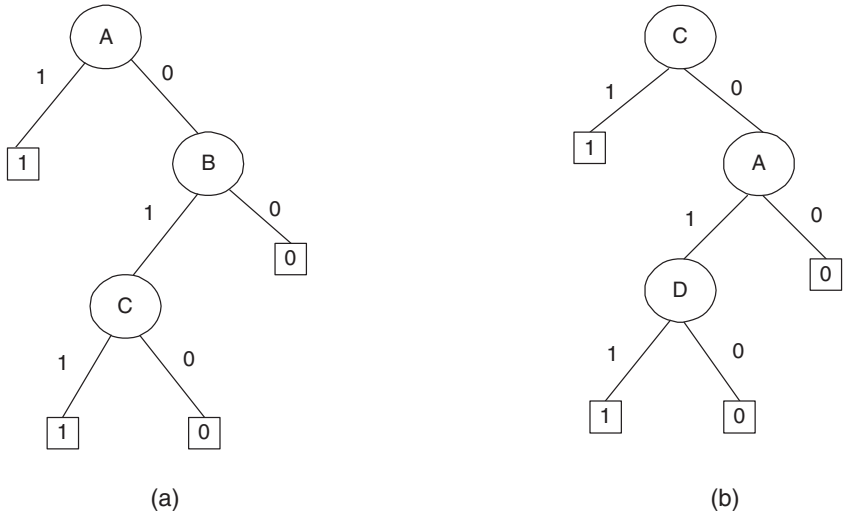

Fig. 3 BDDs representing (a) h and (b) k

direct link to the node on its 0 branch, and the nodes connected to the 1 branch are deleted. The following standard collapsing operations can also be applied to obtain the most efficient BDD.

1. If the two sons of a node X are equivalent, then delete node X and direct all its incoming edges to its left son.
2. If nodes A and B are equivalent, then delete node B and direct all its incoming edges to A.

A 'son' of a node is simply the node directly attached to either its 1 or 0 branch. For example, consider $h = A + B.C$ and $k = C + A.D$; the BDDs representing these functions are shown in Figs 3(a) and (b), with the ordering $A < B < C$ assumed in the first BDD and $C < A < D$ in the second.

To obtain the BDD representing $h + k$, the BDD representing k replaces the two terminal 0 vertices in the BDD representing h. The resulting BDD is shown in Fig. 4.

As can be seen, variables are repeated on paths, e.g. both A and C are repeated on the same path. Considering the path 0 branch of A, 1 branch of B, and 0 branch of C takes us to the repeated variable C. As this lies on the 0 branch of C, the repeated variable can be removed and all that lies on its 0 branch kept. This then leads to the repeated variable A; as this lies on the 0 branch of the first occurrence of A, the repeated variable can be removed and all that lies on its 0 branch kept. This is just a terminal 0 vertex. The other repetition of A can be dealt with in the same manner. The resulting BDD is shown in Fig. 5(a). By identifying that the variable B is irrelevant, the final, reduced BDD is shown in Fig. 5(b).

This method of combining BDDs can be adopted to model phased mission problems efficiently. The methodology to do this will be developed and demonstrated by considering a simple example.

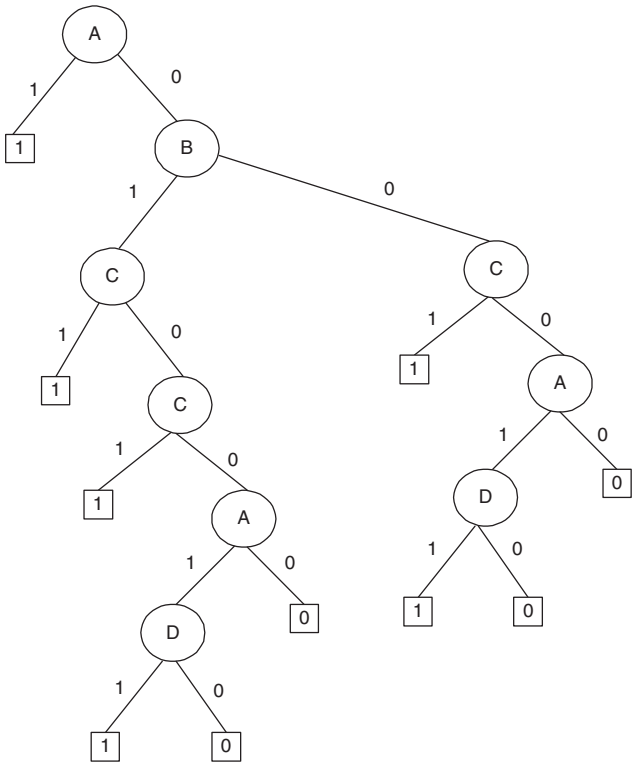

Fig. 4 BDD representing $h + k$

4 PHASED MISSION EXAMPLE

A simple phased mission problem consisting of non-repairable components A, B, C, and D (Fig. 6(a)), will be considered here in order to demonstrate an efficient method of BDD construction for phased mission analysis.

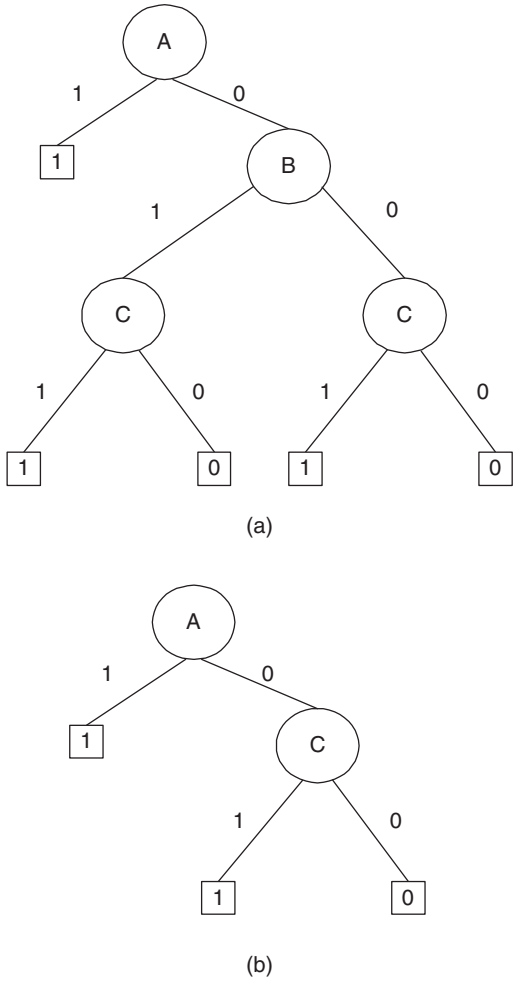

Fig. 5 Simplified BDD structure

During phase 1, which lasts until time t_1, the success of the mission is dependent upon the fact that component A and at least one of components B and D functions. Successful completion of phase 1 means that the mission then enters phase 2, which requires component B and at least one of components A and C to function between times t_1 and t_2. Successful completion of that phase then means that the system enters phase 3, and so on. The mission is successfully accomplished if all phases are completed successfully.

The phase failure fault trees are given for this mission in Fig. 6(b).

5 CONSTRUCTION OF PHASE BDDs

The first stage in the methodology is to convert all the phase fault trees to their BDD form. Considering phase 1, the top event, 'failure in phase 1' is given by the logical expression $A_1 + B_1.D_1$, where the notation A_i represents the failure of component A in phase i. Assuming the ordering $A_1 < B_1 < D_1$, and using the rules that for the OR gate the events are linked by their 0 branches and for an AND gate by their 1 branches, the BDD representing this expression is shown in Fig. 7(a). In phased mission problems the state of components which are used in subsequent phases are important, as they have a bearing on the future performance of the system. Hence, in the BDDs representing each phase, paths ending in a terminal 0 vertex, representing system functioning through that phase, must consider all components used in that and subsequent phases. Therefore the BDD representing failure in phase 1, shown in Fig. 7(a), must be extended so that all component states are defined on the paths ending in a terminal 0 vertex where the mission will pass to the next phase. On the terminal 0 connected to node B_1, the conditions of A_1 and B_1 are specified (both work). This needs to be expanded to include all states that components C_1 and D_1 can reside in. For the second terminal 0 (the 0 branch of the D_1 node) the status of A_1, B_1, and D_1 is fixed. It only remains to consider all states for C_1. The extended BDD is shown in Fig. 7(b).

The BDDs developed for failure of the other three phases are illustrated in Fig. 8.

The ordering of basic events can be different for each phase (as it is in this example) to obtain the most efficient representation of the phase failure logic.

6 CONSTRUCTION OF THE MISSION BDD

These BDDs representing failure on the separate phases will now be used to construct a mission BDD. The overall structure of the mission BDD is shown in Fig. 9. There are now more than two system outcomes identified, which means that this new failure logic structure departs subtly from the BDD. The end states represent either a failure in a phase, labelled f_1, f_2, f_3, or f_4 for failure in the phase indicated, or a mission success, labelled 0.

Once constructed, the BDD can then be subjected to a reduction process to ensure that the component state logic is consistent on any branch.

The resulting BDD can be reduced by use of the following rules.

Rule 1. $A_i.\ A_j = 0$, as a non-repairable component A cannot fail in phase i and phase j.
Rule 2. $A_i.\ \overline{A_j} = 0$ for $i < j$, as a component cannot be repaired once failed.
Rule 3. If K_1 is a minimal cut set for phase j, and the components which make up this cut set fail in phases prior to j, then upon entering phase j the system will fail.

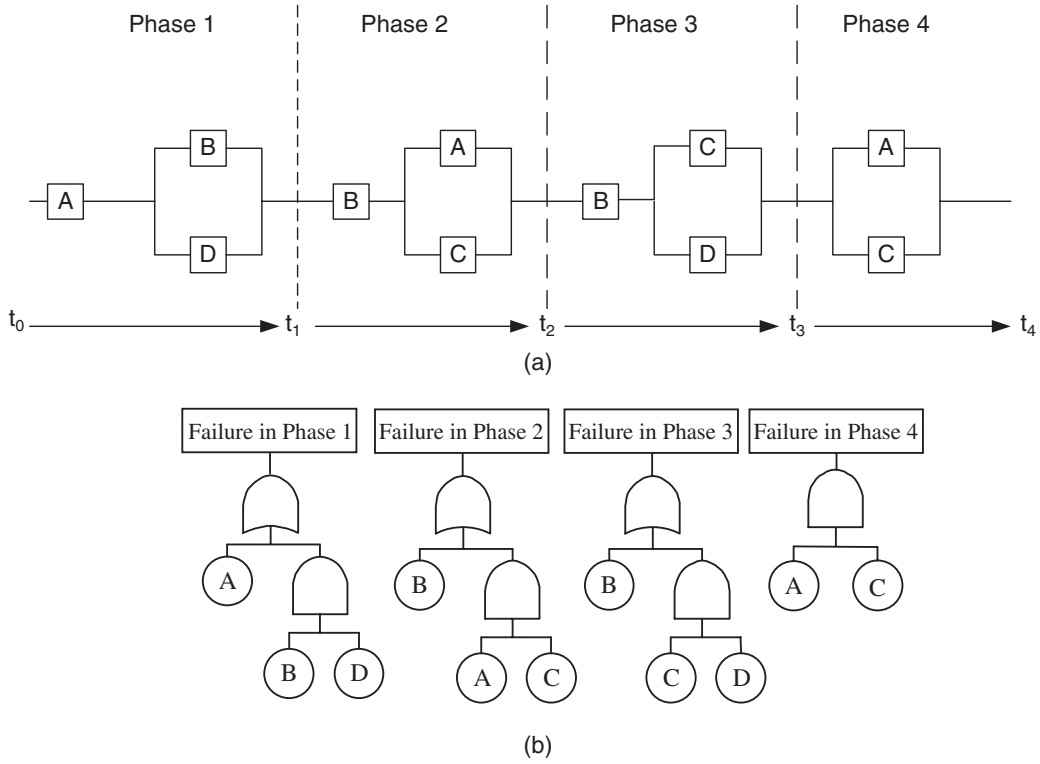

Fig. 6 (a) Reliability network of a simple phased mission system; (b) fault tree representation of individual phase failures

$\overline{A_i}$ represents the fact that component A functions throughout phase i. The system example illustrated in Fig. 6 is considered again to demonstrate the construction process. The BDDs shown in Figs 7(b) and 8(a), representing failure in phases 1 and 2 can be combined into a single BDD representing failure in phases 1 or 2 by replacing the terminal 0 vertices in the BDD in Fig. 7(b) with the BDD in Fig. 8(a). The resulting BDD is shown in Fig. 10(a). This has been reduced using the rules 1 to 3 above.

Apply rules 1 and 2 along the following paths:

1. along the 0 branch of A_1, 0 branch of B_1, 1 branch of D_1, 1 branch of C_1, 0 branch of B_2, and 1 branch of A_2;
2. along the 0 branch of A_1, 0 branch of B_1, 0 branch of D_1, 1 branch of C_1, 0 branch of B_2, and 1 branch of A_2.

In both cases, remove node C_2 and connect directly to f_2, as component C has already failed in phase 1.

Similarly, apply rules 1 and 2 to the following paths:

1. along the 0 branch of A_1, 0 branch of B_1, 1 branch of D_1, 1 branch of C_1, 0 branch of B_2, and 0 branch of A_2;
2. along the 0 branch of A_1, 0 branch of B_1, 1 branch of D_1, 0 branch of C_1, 0 branch of B_2, 1 branch of A_2, and 0 branch of C_2;
3. along the 0 branch of A_1, 0 branch of B_1, 1 branch of D_1, 0 branch of C_1, 0 branch of B_2, 0 branch of A_2, and 1 branch of C_2;
4. along the 0 branch of A_1, 0 branch of B_1, 1 branch of D_1, 0 branch of C_1, 0 branch of B_2, 0 branch of A_2, and 0 branch of C_2.

This results in a terminal 0 vertex.

Applying rules 1 and 2 to the path, 0 branch of A_1, 0 branch of B_1, 0 branch of D_1, 1 branch of C_1, 0 branch of B_2, and 0 branch of A_2 removes C_2. Applying rule 3 to the path, 0 branch of A_1, 1 branch of B_1, and 0 branch of D_1 results in the terminal node f_2, as B is a single order minimal cut set for phase 2. The resulting reduced BDD is shown in Fig. 10(b), where the 1 and 0 labels on the branches leaving each intermediate node have been omitted for clarity. This BDD was then combined with the BDD representing failure in phase 3 (the phase 3 BDD is attached to the 0 branches of the combined phase 1 and 2 BDD) and reduced, using rules 1 to 3, to obtain a single BDD representing failure in phase 1, 2, or 3. The resulting BDD has 47 outcomes of which 31 are failures, two are failure in phase 1, nine are failure in phase 2, and 20 are failure in phase 3. Combining this BDD with the BDD representing failure in phase 4, (Fig. 8(c)), the BDD representing mission failure is obtained. This is shown in Fig. 11, where the 1 and 0 labels on the branches leaving each intermediate node have again been

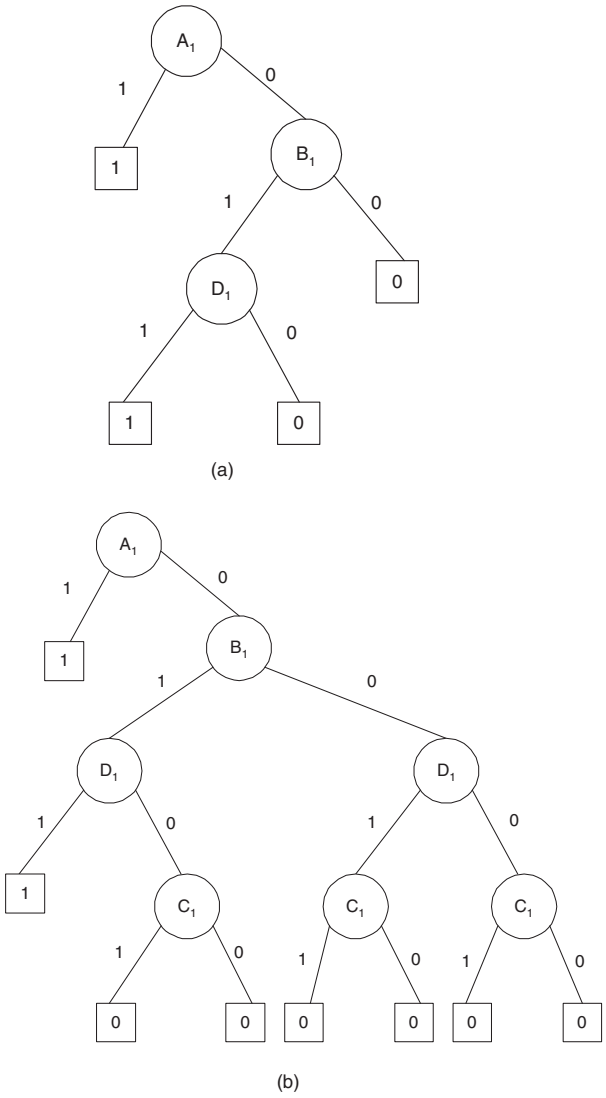

Fig. 7 BDDs representing failure in phase 1

omitted for clarity. Each left edge leaving a variable node is the '1 branch' and each right edge the '0 branch'. In this case there are 47 failure outcomes; two in phase 1, nine in phase 2, 20 in phase 3, and 16 in phase 4. This is equivalent to the results obtained using the cause consequence diagram method for this problem [**9**]. This method of combining the BDDs for the different phases can obviously be applied to any phase mission problem for a non-repairable system, but in order to demonstrate how the method is applied has been shown here for the particular example taken.

7 QUANTIFICATION

7.1 Phase failure modes

Each path through the mission BDD will terminate in either a phase failure or the mission success. The component status conditions encountered

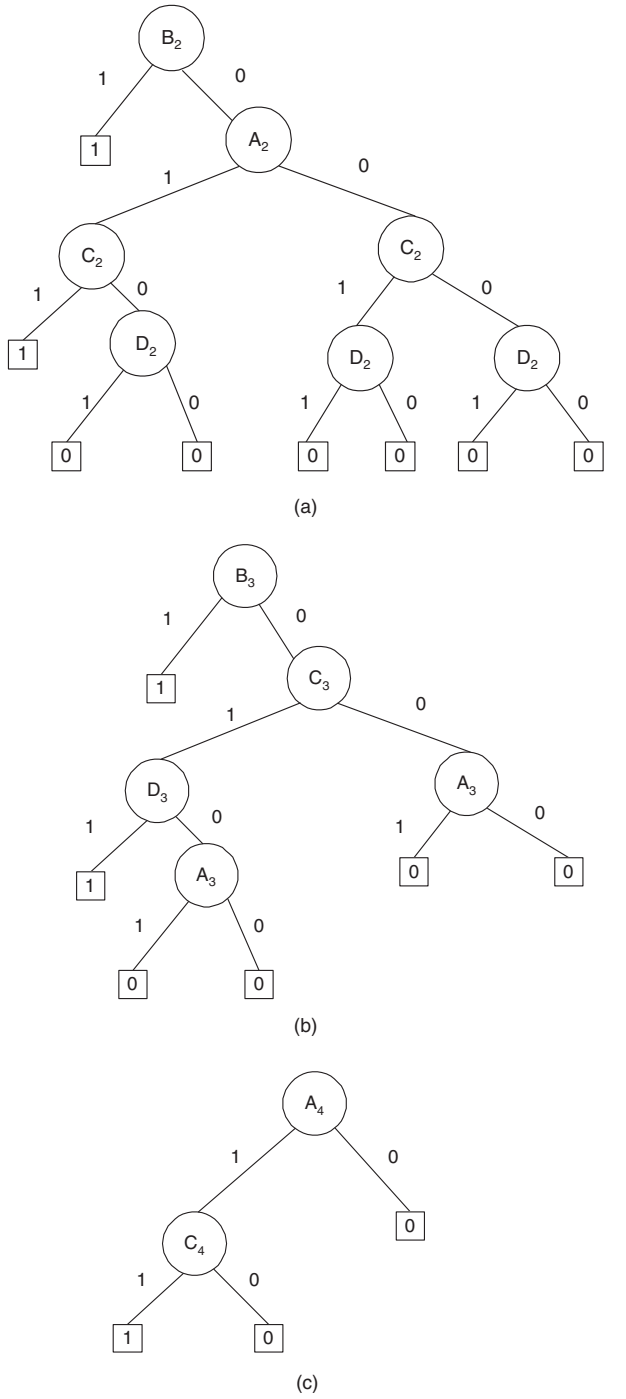

Fig. 8 BDDs representing failure in (a) phase 2, (b) phase 3, and (c) phase 4

along each path will specify over which phases the components must work and which phase it is possible for the failure to occur in order to cause the specified system event. From the path conditions which result in system failure in any phase the phase failure modes can be determined. A phase failure mode is a list of the necessary and sufficient conditions experienced at component-level events (phases working and phases where failure can occur), which lead to the specified phase failure.

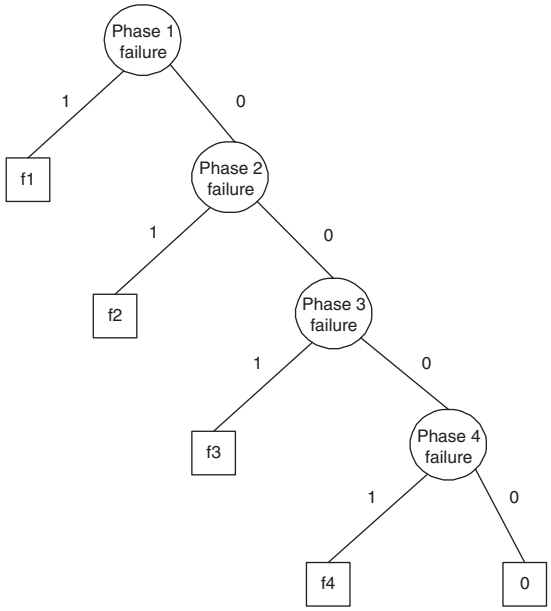

Fig. 9 BDD representing the mission

For systems with non-repairable components it is possible to introduce a phase algebra to determine the phase failure modes. This algebra was established in reference [**3**]; it will account for the non-independence of component conditions from one phase to the next, and enable the minimal conditions required for phase failure to be established, which will be used in the phase failure probability calculations. It is only of interest in which phases the component failure could occur to contribute to the phase failure. When considering the disjoint paths for phased mission systems with non-repairable components, account must be taken of the fact that, if a component fails in phase i, then it must have worked through phases $1, \ldots, i-1$. This is expressed algebraically as

$$\overline{A_{1,i-1}}.A_i = A_i \tag{3}$$

where $\overline{A_{j,k}}$ represents the functioning of component A from the start of phase j to the end of phase k.

Using rules 1 and 2 from the section on construction of the mission BDD with the condition

$$A_i.A_i = A_i \tag{4}$$

the phase failure modes can be evaluated. This is illustrated for the first two phases of the mission BDD shown in Fig. 10(b). The disjoint paths leading to failure in phase 1 are

$$\begin{array}{l} A_1 \\ \overline{A_1}.B_1.D_1 \end{array} \tag{5}$$

The initial disjoint paths leading to failure in phase 2 and their subsequent reduction to the phase failure modes are

$$\begin{array}{l}
\overline{A_1}.B_1.\overline{D_1} \\
\overline{A_1}.\overline{B_1}.D_1.C_1.B_2 \rightarrow \overline{A_1}.D_1.C_1.B_2 \\
\overline{A_1}.\overline{B_1}.D_1.C_1.\overline{B_2}.A_2 \rightarrow \overline{B_{12}}.D_1.C_1.A_2 \\
\overline{A_1}.\overline{B_1}.D_1.\overline{C_1}.B_2 \rightarrow \overline{A_1}.D_1.\overline{C_1}.B_2 \\
\overline{A_1}.\overline{B_1}.D_1.\overline{C_1}.\overline{B_2}.A_2.C_2 \rightarrow \overline{B_{12}}.D_1.A_2.C_2 \\
\overline{A_1}.\overline{B_1}.\overline{D_1}.C_1.B_2 \rightarrow \overline{A_1}.\overline{D_1}.C_1.B_2 \\
\overline{A_1}.\overline{B_1}.\overline{D_1}.C_1.\overline{B_2}.A_2 \rightarrow \overline{B_{12}}.\overline{D_1}.C_1.A_2 \\
\overline{A_1}.\overline{B_1}.\overline{D_1}.\overline{C_1}.B_2 \rightarrow \overline{A_1}.\overline{D_1}.\overline{C_1}.B_2 \\
\overline{A_1}.\overline{B_1}.\overline{D_1}.\overline{C_1}.\overline{B_2}.A_2.C_2 \rightarrow \overline{B_{12}}.\overline{D_1}.A_2.C_2
\end{array} \tag{6}$$

Phase failure modes for the other phases are evaluated in the same way; this produces 20 phase 3 failure modes and 16 phase 4 failure modes, which are not listed here.

7.2 Phase and mission failure probability

Each path on a BDD through to a terminal f_i vertex is disjoint, and therefore the phase failure probability can be obtained by summing the probabilities of the disjoint paths. In order to find the mission failure probability it is necessary to sum the probabilities of all the disjoint paths ending in any terminal f_i.

Using the method described in this paper, more information can be gained than just mission failure probability, as the BDD also gives the probability of failure in each phase. This has an advantage over other methods [**1**], which produce only the mission failure probability.

The example phased mission used to illustrate the method is considered again; q_{A_i} is used to represent the failure probability of component A in phase i, and Q_i is taken to be the failure probability in phase i. Then

$$q_{A_i} = \int_{t_{i-1}}^{t_i} f_A(t)\mathrm{d}t \tag{7}$$

where $f_A(t)$ is the failure density function for component A in phase i, and phase i runs from t_{i-1} to t_i. For phase 1

$$Q_1 = q_{A_1} + (1 - q_{A_1})q_{B_1}q_{D_1} \tag{8}$$

For phase 2 then

$$\begin{aligned}
Q_2 = {} & (1-q_{A_1})(1-q_{D_1})q_{B_1} + (1-q_{A_1})q_{D_1}q_{C_1}q_{B_2} \\
& + (1-q_{B_1}-q_{B_2})q_{D_1}q_{C_1}q_{A_2} + (1-q_{A_1})(1-q_{C_1})q_{D_1}q_{B_2} \\
& + (1-q_{B_1}-q_{B_2})q_{D_1}q_{A_2}q_{C_2} + (1-q_{A_1})(1-q_{D_1})q_{C_1}q_{B_2} \\
& + (1-q_{B_1}-q_{B_2})(1-q_{D_1})q_{C_1}q_{A_2} \\
& + (1-q_{A_1})(1-q_{D_1})(1-q_{C_1})q_{B_2} \\
& + (1-q_{B_1}-q_{B_2})(1-q_{D_1})q_{A_2}q_{C_2}
\end{aligned}$$

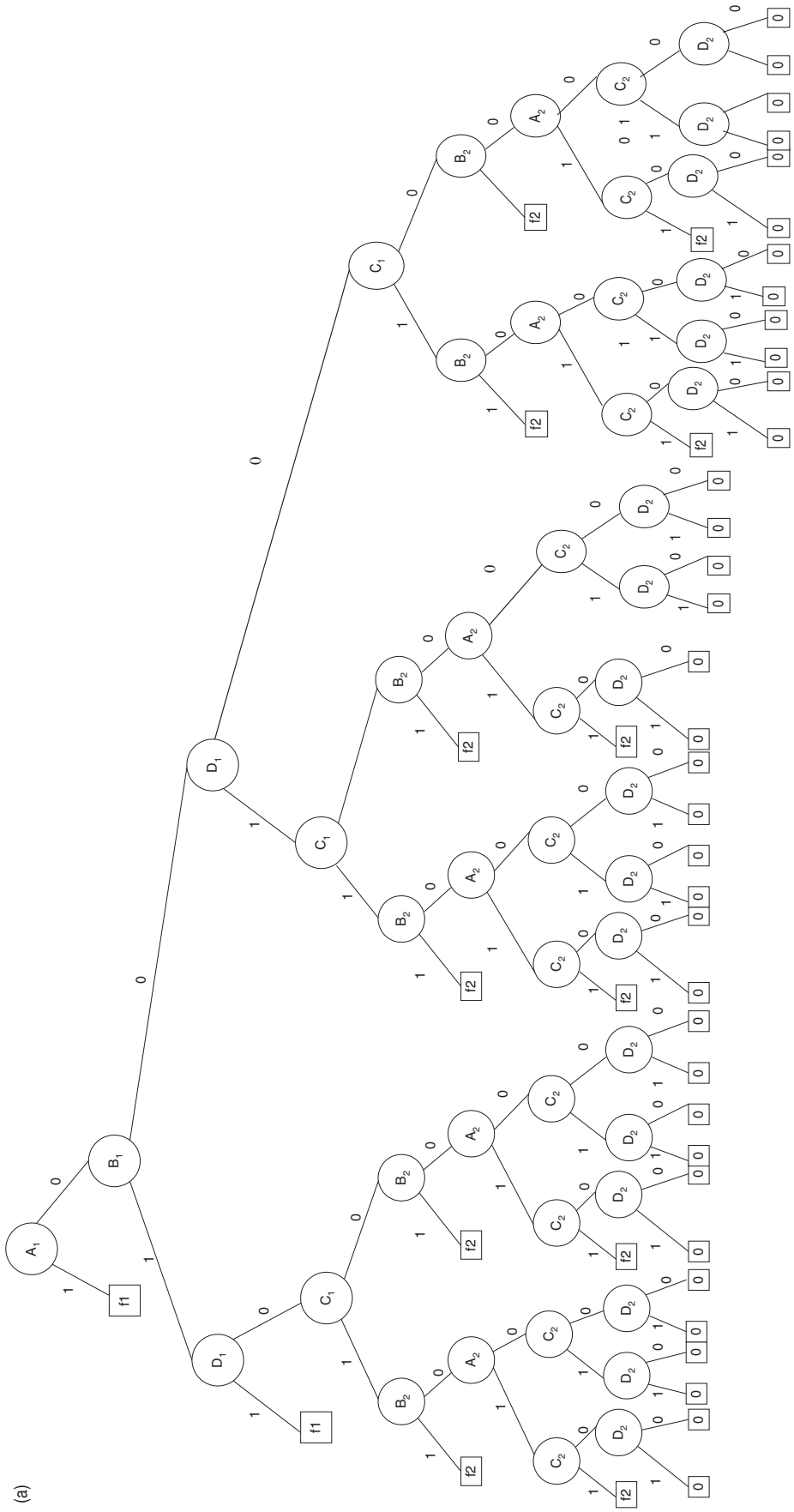

Fig. 10 (a) BDD representing failure in phases 1 or phase 2

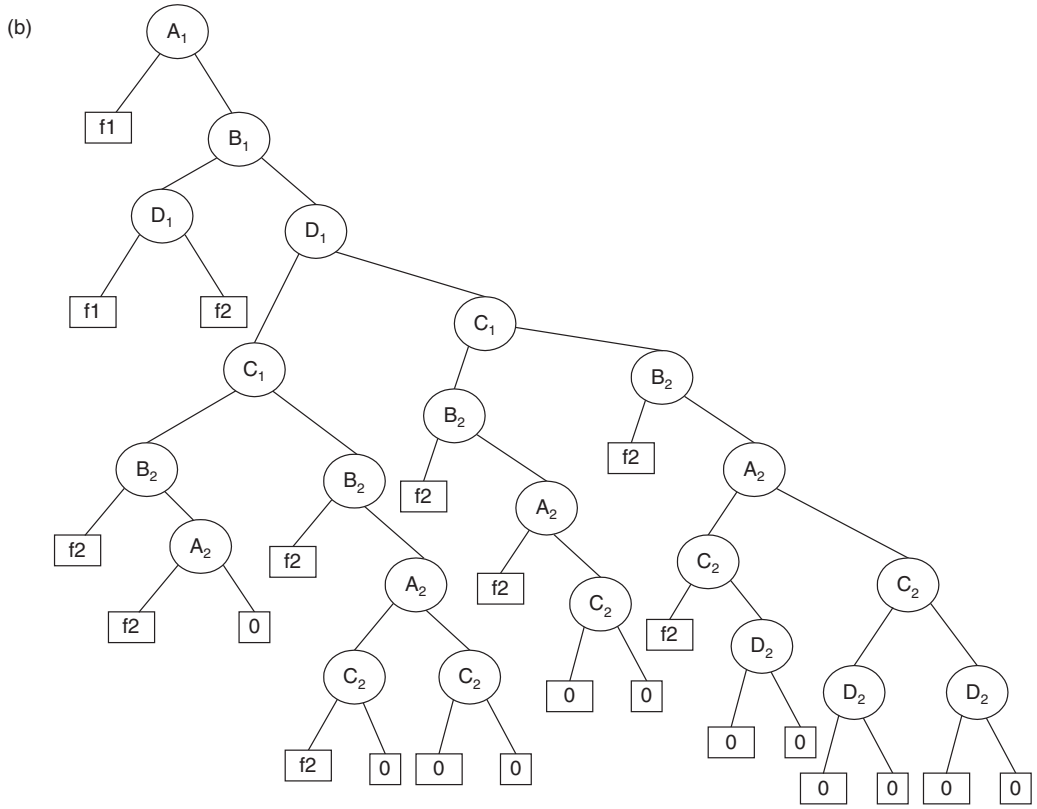

Fig. 10 (b) Reduced BDD representing failure in phase 1 or phase 2

Expressions can also be determined for Q_3 and Q_4 using the same method. The probability Q_{misson} of the mission failure can then be determined using

$$Q_{\text{mission}} = \sum_i Q_i \tag{9}$$

8 GENERAL ALGORITHM

8.1 Mission BDD construction

1. Utilizing constructs for AND and OR gates produces the BDD for each phase from the phase fault trees. These are expanded in comparison with usual BDDs, as all possible states of components required in future stages must be explicitly considered on the paths leading to phase success.
2. By considering the mission failure as an OR combination of the phase failures, a mission BDD can be constructed. Terminal mission BDD nodes denote the phase in which the mission fails or that mission success has been achieved. Since it has more than two possible outcomes, this mission BDD is subtly different from a conventional BDD.
3. Phase failure BDDs are incorporated, one by one, into the mission BDD to yield a structure where mission states are determined by the performance of the components in each phase.
4. The mission BDD is reduced to its simplest form by removing paths which represent impossible component conditions such as failing more than once or working in phases which follow a failure.

8.2 Mission BDD quantification

1. Failure modes for each phase of the mission are obtained by considering the component conditions represented along each of the BDD paths leading to the specified phase failure.
2. The failure modes are simplified using the phase algebra.
3. Component phase failure probabilities are evaluated.
4. Using the disjoint phase failure modes and the component failure probabilities the phase failure likelihoods are quantified.
5. The phase failure probabilities can be combined with the consequences of phase failure in order to perform a mission risk analysis, or summed to yield the mission failure probability.

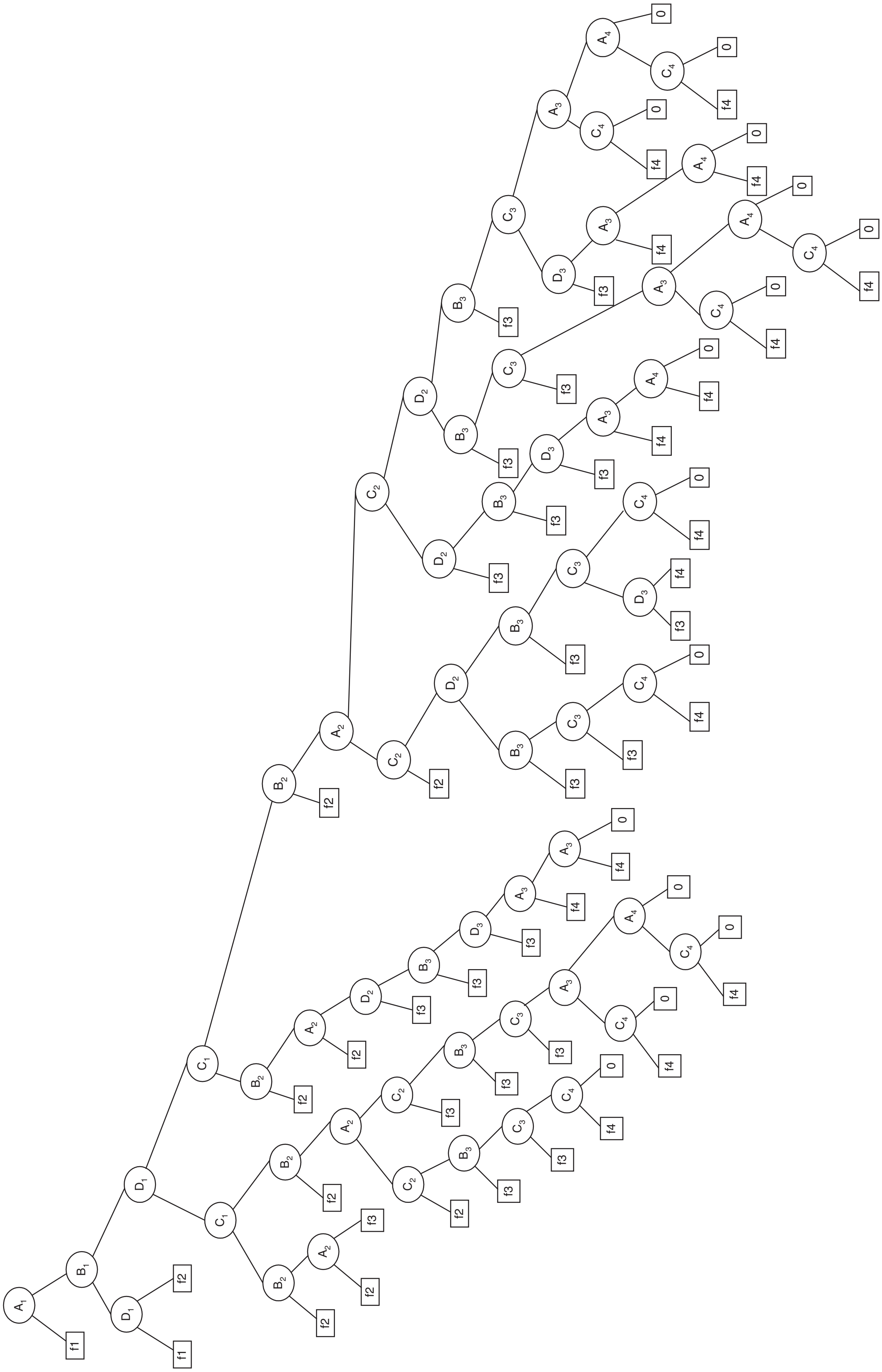

Fig. 11 BDD representing mission failure

9 CONCLUSIONS

1. The method given in this paper provides an effective approach for developing the BDD for a system which undergoes a phased mission. This BDD will identify the causes of each phase failure.
2. The phase failure modes can be identified from the resulting mission BDD.
3. Using the disjoint nature of the phase failure modes and the probability of component failures in specified phases, the phase failure probability for each phase can be obtained. This can be combined with differing consequences of failure in any phase to perform a mission risk analysis.
4. Mission failure probability is obtained by summing the phase failure likelihoods.

REFERENCES

1 Esary, J. D. and **Ziehms, H.** Reliability analysis of phased missions. In *Reliability and fault tree analysis*, 1975, pp. 213–236. (Society for Industrial Applied Mathematics, Philadelphia, Pennsylvania).

2 Zang, X., Sun, H., and **Trivedi, K. S.** A BDD-based algorithm for reliability analysis of phased mission systems. *IEEE Trans. Reliability*, 1999, **48**(1), 50–60.

3 La Band, R. and **Andrews, J. D.** Phased mission modelling using fault tree analysis. *Proc. Instn Mech. Engrs, Part E: J. Process Mechanical Engineering*, 2004, **218**, 83–91.

4 Clarotti, C. A., Contini, S., and **Somma, R.** Repairable multi-phase systems – Markov approaches for reliability evaluation. In *Synthesis and analysis methods for safety and reliability studies* (Eds G. Apostolakis, S. Garribba, and G. Volta), 1980, pp. 45–58 (Plenum, New York).

5 Smotherman, M. and **Zemoudeh, K.** A non-homogeneous Markov model for phase mission reliability analysis. *IEEE Trans. Reliability*, 1989, **38**, 585–590.

6 Smotherman, M. and **Geist, R. M.** Phased effectiveness using a non-homogeneous Markov reward model. *Reliability Engng System Safety*, 1990, **27**, 241–255.

7 Rauzy, A. New algorithms for fault tree analysis. *Reliability Engng System Safety*, 1993, **40**, 203–211.

8 Sinnamon, R. M. and **Andrews, J. D.** Improved efficiency in qualitative fault tree analysis. *Qual. Reliability Engng Int.*, 1997, **13**, 293–298.

9 Vyzaite, G., Dunnett, S., and **Andrews, J. D.** Cause consequence analysis of non-repairable phased missions. *Reliability Engng System Safety*, 2006, **91**, 398–406.