Fault Trees, Decision Trees, And Binary Decision Diagrams: A Systematic Comparison

Lisandro A. Jimenez-Roa

Formal Methods and Tools (FMT), University of Twente, The Netherlands. E-mail: l.jimenezroa@utwente.nl

Tom Heskes

Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands. E-mail: Tom.Heskes@ru.nl

Marielle Stoelinga

Formal Methods and Tools (FMT), University of Twente, The Netherlands. E-mail: m.i.a.stoelinga@utwente.nl

In reliability engineering, we need to understand system dependencies, cause-effect relations, identify critical components, and analyze how they trigger failures. Three prominent graph models commonly used for these purposes are fault trees (FTs), decision trees (DTs), and binary decision diagrams (BDDs). These models are popular because they are easy to interpret, serve as a communication tool between stakeholders of various backgrounds, and support decision-making processes. Moreover, these models help to understand real-world problems by computing reliability metrics, minimum cut sets, logic rules, and displaying dependencies. Nevertheless, it is unclear how these graph models compare. Thus, the goal of this paper is to understand the similarities and differences through a systematic comparison based on their (i) purpose and application, (ii) structural representation, (iii) analysis methods, (iv) construction, and (v) benefits & limitations. Furthermore, we use a running example based on a Container Seal Design to showcase the models in practice. Our results show that, given that FTs, DTs and BDDs have different purposes and application domains, they adopt different structural representations and analysis methodologies that entail a variety of benefits and limitations, the latter can be addressed via conversion methods or extensions. Specific remarks are that BDDs can be considered as a compact representation of binary DTs, since the former allows sub-node sharing, which makes BDDs more efficient at representing logical rules than binary DTs. It is possible to obtain cut sets from BDDs and DTs and construct a FT using the (con/dis)junctive normal form, although this may result in a sub-optimal FT structure.

Keywords: fault tree analysis, decision tree, binary decision diagram, systematic comparison, reliability engineering, decision making, graph models.

1. Introduction

Three commonly used graph models are Fault Trees (FTs), Decision Trees (DTs), and Binary Decision Diagrams (BDDs). These models are popular because they provide a graphic representation of a hierarchical data structure. They are widely used in different domains and applications such as reliability engineering, system analysis, and computer memory optimization (Lee et al., 1985; Rokach and Maimon, 2014; Kubica et al., 2021) (see Table 1).

This paper aims at understanding the explicit similarities and differences between these models. This comparison is important, for example, to facilitate the selection of a model for a given application, or to clarify terminology such as "Tree" or "Decision" for less experienced users. To this end, we focus on comparing these models systematically, as well as discuss conversion methods to transform one model into another.

The structure of this paper is as follows. Section 2 describes our methodology, and Sections 3 to 5 present the results per model: FTs, DTs and BDDs, respectively. In Section 6, we discuss the conversion methods, and finally, Sections 7 and 8 contain the discussion and conclusions.

2. Methodology

Our comparison follows a systematic approach proposed by Smith et al. (2017). Our main steps are (1) to propose five main categories encompassing different aspects of interest, namely (i) purpose & application, (ii) structural representation, (iii) analysis, (iv) construction, and (v) benefits & limitations; (2) to refine each category into aspects; (3) to adapt a running example from

Proceedings of the 31th European Safety and Reliability Conference.
Edited by Bruno Castanier, Marko Cepin, David Bigaud and Christophe Berenguer
 Published by Research Publishing, Singapore
ISBN: 981-973-0000-00-0 :: doi: 10.3850/981-973-0000-00-0_esrel2021-paper

Table 1.: Comparison between FTs, DTs and BDDs.

Category	Aspect	FT	DT	BDD
(i) Purpose and application	-	Reliability Engineering; Root Cause Analysis; Decision-Making	Classification; Regression; Decision-making	Model checking; Protocol validation; Verification
(ii) Structural representation	(ii.a) Structure	Top-down directed acyclic graph	Hierarchical top-down graph	Directed acyclic graph
	(ii.b) Main symbols	Top/intermediate/basic events, logic gates	Root / decision / leaf nodes	Root / transitional nodes
	(ii.c) Leaves information	Failure frequency / probability density funct.	Numeric & Nominal data	Boolean data {0, 1}
	(ii.d) No. of children	Multiple	2*	2
	(ii.e) Sub-node sharing?	Yes	No	Yes
	(ii.f) Gates/Nodes information	Logic rules (*AND*, *OR*, *VOT*)	Attribute splitting criteria	Binary splitting criteria
(iii) Analysis	(iii.a) Analysis direction	Bottom-up	Top-down	Top-down
	(iii.b) Type of analysis	Qualitative & Quantitative	Quantitative	Qualitative & Quantitative
	(iii.c) Allows cut sets computation	Yes	Yes**	Yes
	(iii.d) Extensions	Dynamic FT, Repairable FT, Attack FT, State/Event FT	Causal DT, Clustering DT, Binary DT, Survival DT, Oblivious DT	Ordered BDD, Reduced OBDD
(iv) Construction	(iv.a) Common way of construction	Manually built based on expert judgement	Data-driven: i.e., (un)supervised learning	Through Boolean functions
	(iv.b) Induction algorithms	IFT, LIFT, DDFT, ILTA	ID3, C4.5, CART	Shannon expansion formula, *IFT* method
(v) Benefits and limitations	(v.a) Main benefits	Allows dependability analysis, and computation of reliability metrics	Intuitive and easy to follow. Easily derived from data	Since based on Boolean algebra, it allows faster computations
	(v.b) Main limitations	Subjective, and based on assumptions of statistical independency	Changes in the root node can generate a whole new structure	Constrained to Boolean data. Changes in the root node can generate a new BDD

* Except when using multivariate splitting criteria, then it is > 2.

** Only when the DT maps to classes *no failure* and *failure*.

Stamatelatos et al. (2002) consisting of a FT for a *Container Seal Design*, and model it by DTs and BDDs; (4) to discuss conversion methods to transform one model into another (here it is important to clarify that minimum compatibility requirements must be met between models, e.g., DTs must model diagnostic decision rules and binary variables); and (5) summarize the different aspects per model.

3. Fault Trees

3.1. *Purpose and application*

Fault Tree Analysis (FTA) is a key method in reliability engineering and root cause analysis, to support decisions in system design and maintenance. FTA is ISO standardized (IEC, 2006) and has been used in a wide range of domains including automotive, aerospace, and nuclear industries (Kabir, 2017).

Technically, a Fault Tree (FT) is a directed acyclic graph that models why a system fails, by identifying how low-level failures propagate through the system and lead to the system-level failure.

3.2. *Structural representation*

FTs are composed of different symbols (Fig. 1(a-b)), whose objective is to model the logic relations between the events (basic and intermediate, Fig. 1.(a)) and the top event. Logical transitions between events must meet the conditions set by the *gate symbols* (Fig. 1.(b)).

FTs distinguish several event types (Fig. 1.(a)):

(i) *Basic events*, symbolized by a circle, corresponds to an event that initiates the failure of the system in question. The basic events

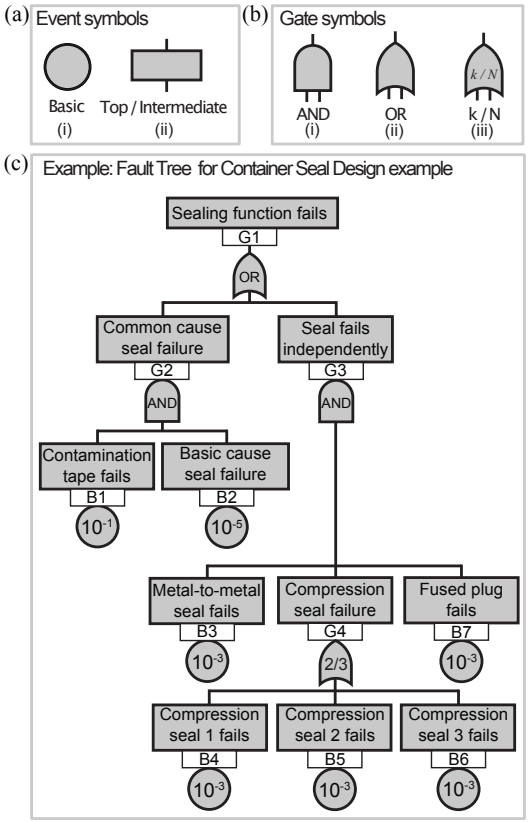

Fig. 1.: Elements in an FT: (a) event symbols, (b) gate symbols, and (c) example of an FT, adapted from: Stamatelatos et al. (2002).

do not require further refinement; in other words, the resolution achieved is adequate.

(ii) *The top event*, symbolized by a rectangle, is localized at the top of the FT. It models the failure of the complete system under consideration. *Intermediate events* are associated with (sub)system/component failures, and are equipped with a logical gate.

Three commonly used gate types in FTs are (Fig. 1.(b)):

(i) *AND gates* indicate that the associated event will only occur if *all* the associated (basic or intermediate) events occur.

(ii) *OR gates* indicate that the associated event will only occur if *at least* one of the associated (basic or intermediate) events occur.

(iii) *k/N gates* (or *VOT gates*) indicate that the associated event will occur minimally k of the N the events associated to the gate will occur.

Various FT extensions exist to model more complex dependability patterns, see Vesely et al. (1981); Dugan et al. (1992).

3.3. *Fault Tree Analysis*

FTA enables two types of analyses. *Qualitative* analysis is based on the FT structure and aims at finding the critical system components. *Minimal cut sets* are minimal combinations of component failures that lead to a system failure. Small cut sets point to system vulnerabilities. For example, a minimal cut set in Fig. 1.(c) is given by *Contamination tape fails* (B1), and *Basic cause seal failure* (B2).

Quantitative analyses aim at computing various dependability metrics, such as system *reliability* (i.e., probability that the system fails in a period of time); the *availability* (i.e., the percentage of time that the system remains operational); *mean time to failure* (i.e., average time before the first failure). These metrics require the leaves of the FT to be equipped with failure probabilities, either as probability density functions or constant probabilities. An extensive list of algorithms for analysis of FTs is provided in Ruijters and Stoelinga (2015).

3.4. *Construction of FTs*

Traditionally, FTs are handcrafted by experts on a system of interest. Automatic algorithms for induction of FTs are IFT (Madden and Nolan, 1970), LIFT (Nauta et al., 2018), based on Evolutionary Algorithms (Linard et al., 2019), Bayesian Networks (Linard et al., 2019), ILTA (Waghen and Ouali, 2019), and DDFT (Lazarova-Molnar et al., 2020). Two main associated challenges are (i) discovering the FT structure that efficiently and completely represents the system failure mechanisms for a given top event, and (ii) finding patterns in a dataset containing information about the system failure mechanisms.

3.5. *FT example*

Fig. 1.(c) shows a fault tree modeling the failure of a sealing mechanism for a container, adapted from Stamatelatos et al. (2002). The sealing function fails either due to a common cause of seal failure occurs or if the seals fail independently. For the former, it is necessary that the contamination tape fails and a basic seal failure occurs. For the latter, it is necessary for the metal-to-metal seal, the fused plug, and at least two of the three compression seals to fail.

As shown in Fig. 1.(c), the top event is refined in three intermediate events and independent basic events. This example, for simplicity, only considers *AND*, *OR* and *VOT* gates. Since there is a quantification of the failure probability of each basic event, a *qualitative* analysis can be carried out. Given the failure probabilities of the basic events in Fig. 1.(c) and assuming independence between all basic events, the failure probability $P(G_1)$ of the top event *Sealing function fails* reads

$$P(G_1) = P(G_2) + P(G_3) - P(G_2)P(G_3)\,,$$

where we have, with $p = 10^{-3}$ the failure probability of the events B_4 through B_6,

$$P(G_4) = 3p^2(1-p) + p^3 \approx 3 \times 10^{-6}$$
$$P(G_3) = P(B_3)P(G_4)P(B_7) \approx 3 \times 10^{-12}$$
$$P(G_2) = P(B_1)P(B_2) = 1 \times 10^{-6}\,,$$

and hence $P(G_1) \approx 1 \times 10^{-6}$.

3.6. *Benefits and limitations*

Some important advantages of FTs are that they (i) are based on probability theory, (ii) enable computation of different metrics (e.g., reliability) to aid decision making, (iii) due to its interpretability, they serve as a communication tool across multiple disciplines, helping to align stakeholders, (iv) facilitate the interpretation of different failure mechanisms, helping to identify critical components. Limitations of FTs include (i) the hand-made way in which they are traditionally constructed makes them costly and time consuming, (ii) the assumption that basic events are independent is not always fulfilled, (iii) validation in FTs is carried out based on expert judgment, which makes it subjective and prone to human error, (iv) difficult to collect appropriate data, (v) AND/OR gates are not always expressive enough. Some of the above aspects were also pointed out by Sarbayev et al. (2019).

4. Decision Trees

4.1. *Purpose and application*

Decision Trees (DT) are flowchart-like structures that model decisions and their possible consequences. DTs are used by decision makers, mainly due to its intuitive interpretation. Technically, DTs serve as a classifier presented as a hierarchical top-down graph that generates a set of decision rules (Thomas et al., 2020). DTs can cope with numerical and nominal attributes, and are often used in classification and regression problems. DTs are applied in text classification, diagnosis of diseases, fraud detection, speech recognition, video analysis, among others (Rokach and Maimon, 2014).

4.2. *Structural representation*

A DT is composed of nodes containing control statements based on attributes (or features). Among the common elements in a DT (Fig. 2.(a-b)) are the (i) *decision nodes* where the first on top is known as the *root node*, (iii) incoming and outgoing *edges*, and (iv) the *leaf nodes* at the bottom of the structure representing the end point of a decision path. Each decision node is associated to an attribute, and the branches to a discrete category or to a range of values.

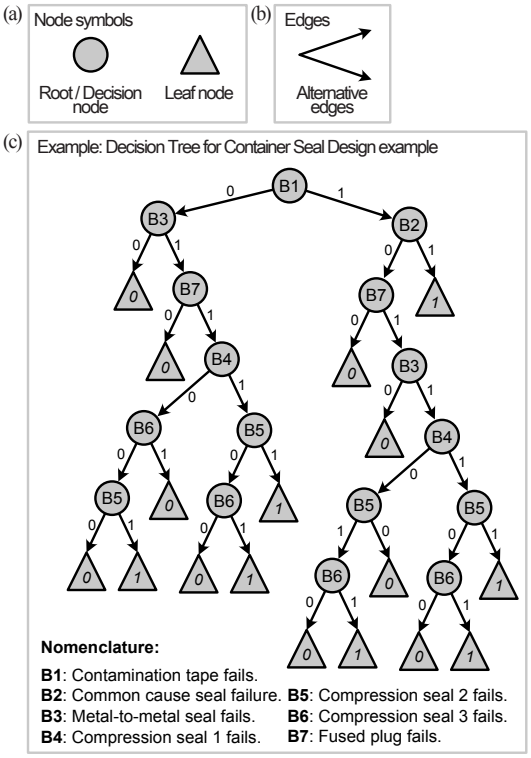

Fig. 2.: (a) node symbols, (b) edges, (c) DT model corresponding to the FT from Fig. 1.(c).

4.3. *Evaluation of DTs*

There are two main types of DTs: *classification trees* and *regression trees*. The first represents a function that maps all available samples (i.e., input data) into a predefined set of discrete categories, also known as *labels*. The second attempts to predict a continuously-valued target attribute based on a set of input attributes. An example of how to evaluate the decision rules in a DT is presented in Section 4.5.

4.4. *Construction of DTs*

Induction algorithms, also called *inducers*, build a model that generalizes the relationship between input attributes and target attributes based on a given training set of examples (Rokach and Maimon, 2014). DT inducers look for the best feature upon which to perform split by means of *splitting criteria*. Among the most commonly used are the *Gini index*, and *gain ratio*. *C4.5* (Quinlan, 1993) is a well-known DT induction algorithm. Rokach and Maimon (2014) provides a detailed list of other DTs inducers.

4.5. *DT example*

To learn a DT that represents the same qualitative information as the FT in Fig 1.(c), we proceeded

as follows. First, we randomly generated 1000 data points, by drawing the basic events independently from a binomial distribution with probability of success equal to 0.5 and calculating the corresponding top event (0 for no failure and 1 for failure) by following the logical rules of the FT. The randomly drawn values for the basic events represent the input x and the value of the top event the output y.

Then, a *Binary Decision Tree* for classification was induced based on the above described dataset in a supervised fashion (i.e., mapping x into y), using the function *fitctree* in Matlab which uses the CART algorithm (Breiman et al., 1984). The results are presented in Fig. 2.(c). The basic events in the FT (Fig. 1.(c)) become decision nodes in the DT. The leaf nodes in the DT are Boolean, representing the classes no failure and failure.

By interpreting the decision rules of the DT one could say that if the *contamination tape* does not fail ($B_1 = 0$) and the *metal-to-meal seal* does not fail ($B_3 = 0$) the system will not fail. Or if the *contamination tape* fails ($B_1 = 1$) and the *basic cause seal failure* occurs ($B_2 = 1$), the system fails. This DT encodes all cut sets of the FT, which can be obtained by concatenating all decision nodes with value 1 that lie on a path to a leaf node with value 1, yielding for example: $\{B_3, B_7, B_4, B_5\}$, $\{B_3, B_7, B_4, B_6\}$, etc.

4.6. *Benefits and limitations of DT*

DTs have the following benefits (i) they are intuitive and easy to follow by technical and non-expert users, (ii) navigating through the branches associated to fault states enables the identification of useful logical rules (similar to the concept of cut sets in FTs), (iii) there are many algorithms for learning DTs from data that scale favorably with the number of data points. DTs have the following limitations (i) it is possible to have several identical sub-trees, which affects efficiency and computational performance. (ii) small variations in a splitting node located near the root of the tree can result in a completely new structure.

5. Binary Decision Diagrams

5.1. *Purpose and application*

Binary Decision Diagrams (BDDs) were originally introduced by Lee (1959) and are heavily used in model checking (Bryant, 2018), hardware design & verification, protocol validation, and automated deduction (Rauzy and Dutuit, 1997). BDDs also provide efficient algorithms to calculate the failure probabilities and minimal cut sets in a fault tree (Bryant, 1992; Reay, 2002).

Syntactically, a BDD is a (connected and single-rooted) directed acyclic graph (see Fig. 3). It provides a very succinct representation and manipulations of Boolean expressions Bryant (1992).

5.2. *Structural representation*

BDDs consist of *Transitional nodes*, or *non-terminal vertices*. These are depicted by circles and contain binary control statements or *function variables*. Terminal nodes, also called *leaf nodes* or *terminal vertices*, are represented by squares, and are labeled with either 1: *True* or, 0: *False*. Conventionally, the vertices (Fig. 3.(b)) are depicted as a solid line if the output of the transitional node is 1, and dashed lines if the output is 0.

5.3. *Evaluation of BDDs*

A reason for the popularity of BDDs is their efficient handling of many operations on Boolean functions, including negation, conjunction and disjunction: Given BDD representations of F and G over the same variable ordering, efficient algorithms exist to compute a compact BDD for $\neg F$, $F \wedge G$ and $F \vee G$ (Bryant, 1986).

Given a BDD over a set of variables V, one easily evaluates this BDD over an assignment of V: One starts from the root, and for each variable $v \in V$, one takes the left branch / solid line if $v = 1$, and right branch / dashed line if $v = 0$. Eventually, one reaches a leaf yielding the output for this assignment.

5.4. *Construction of BDDs*

Given a Boolean function F, the BDDs representation for F can be constructed recursively applying the *Shannon expansion formula* (Akers, 1978)

$$F(x_1, x_2, ...) = x_1 F(1, x_2, ...) \vee (1 - x_1) F(0, x_2, ...)$$

Here F is a Boolean function; and x_i are binary values. In addition, compact representations of BDDs (known as reduced BDDs) for a given order of variables can be obtained using *collapsing operations* by eliminating/merging nodes and sub-diagrams (Friedman and Supowit, 1987).

5.5. *BDD example*

By applying the *Shannon expansion formula* (Section 5.4), we convert the FT in Fig. 1.(c) into a BDD (Fig. 3.(c)), with the associated Boolean function:

$$\begin{aligned} F_1(B_1, B_2, ..., B_7) &= F_2(B_1, B_2) + F_3(B_3, ..., B_7) \\ F_2(B_1, B_2) &= B_1 \cdot B_2 \\ F_3(B_3, ..., B_7) &= B_3 \cdot B_7 \cdot (B_4 \cdot B_5 + \\ &\quad B_4 \cdot B_6 + B_5 \cdot B_6) \end{aligned}$$

Here F_2 and F_3 are Boolean functions that model respectively the intermediate events in Fig. 1.(c). One can see great similarity between this BDD and the DT in Fig. 2.(c), except that a BDD allows shared nodes and the DT cannot.

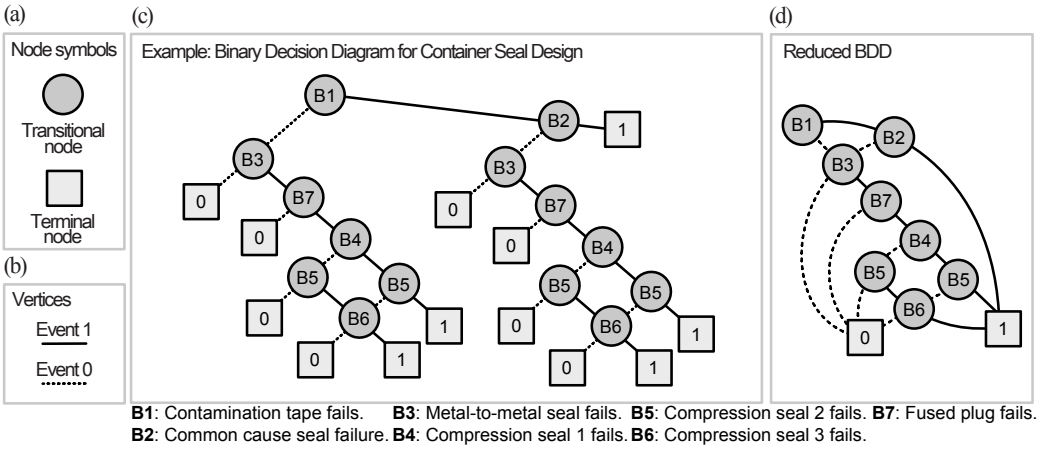

Fig. 3.: (a) BDD node symbols, (b) BDD vertices, (c) BDD for FT from Fig. 1.(c), (d) Reduced BDD.

Through collapsing operations, like merging the nodes B3 and all the terminal nodes, the *Reduced BDD* in Fig. 3.(d) is obtained. The latter, easily identify the minimum cut sets, i.e., all 1-edge vertices that connect to the terminal node of value 1: $\{B_1, B_2\}$, $\{B_3, B_4, B_5, B_7\}$, $\{B_3, B_5, B_6, B_7\}$, and $\{B_3, B_4, B_6, B_7\}$.

5.6. *Benefits and limitations*

Among the advantages offered by BDDs are that (i) they enable a representation of any Boolean function; (ii) there is a wide variety of algorithms that allow performing operations of Boolean functions. One limitation is that finding the order of variables that minimizes the resulting BDD is an NP-hard problem (Bollig and Wegener, 1996).

6. Conversion methods

Conversion methods are mathematical transformations that convert one formalism into another, while preserving relevant properties. So we look for those that allow transitions between FTs, DTs and BDDs (see Fig. 4).

In the transformation **FT** $\rightarrow$ **BDD**, the basic events in the FT become transitional nodes in the BDD, and the top event in the FT is represented in the terminal nodes of the BDD. We identified a few methods where the main idea is to recursively apply the **(i)** *Shannon expansion* until all basic events are converted into BDD nodes (Akers, 1978). Since the variable ordering has a crucial effect on the size of a BDD, a number of heuristic approaches have been developed to find a good order. Examples include **(ii)** *structural importance* of each basic event (Bartlett and Andrews, 2001), **(iii)** a *neural network* approach to choose the best heuristic from a set of alternatives (Bartlett and Andrews, 2002), **(iv)** the *if-then-else* method that focuses on the gates in the fault tree (Remenyte-Prescott and Andrews, 2008), and **(v)** the *component connection method*, where gates with only basic events as inputs are considered (Remenyte-Prescott and Andrews, 2008).

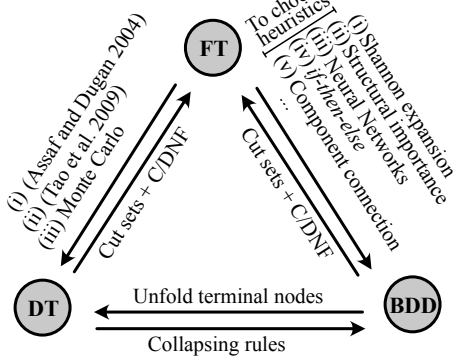

Fig. 4.: Conversion methods among FT, DT, and BDD.

In the transformation **BDD** $\rightarrow$ **FT**, the terminal nodes of the BDD become the top event of the FT, and the transitional nodes of the BDD the basic events of the FT. We did not find specific techniques in the literature that accomplish this transition. Not every BDD can be transformed into a FT, since FTs do not support a NOT function. One approach could be to obtain the cut sets from the BDD and then reconstruct the associated FT e.g., by means of the (con/dis)junctive normal form (C/DNF). Since the resulting FT may not be optimal (e.g., redundant elements), minimization rules should be applied on the FT (Junges et al., 2017).

In the transformation **FT** $\rightarrow$ **DT** the basic events in the FT become decision nodes in the DT, and the top event of the FT the leaf nodes in the DT. A few methods have been developed to accomplish this task. **(i)** Assaf and Dugan (2004)

translate a dynamic fault tree (DFT) to a Markov chain model, and then to a Diagnostic Decision Tree (DDT). **(ii)** Tao et al. (2009) compute from the fault tree the minimal cut sets, top event probabilities and the *Vesely-Fusell* measure, and with this information build the DDT. One may also apply **(iii)** the method that we applied in Section 4.5 to the running example: randomly generate a binary data set using the FT (i.e., Monte Carlo method), and then learn the corresponding DT.

In the transformation **DT** $\rightarrow$ **FT** the decision nodes in the DT become basic events in the FT, and the leaf nodes in the DT the top events in the FT. We did not find any publication to accomplish this transition. Nevertheless, by using all decision rules that connect to a leaf node of value 1, one can get cut sets and build the FT as suggested in the transition BDD $\rightarrow$ FT.

The transformation **DT** $\leftrightarrow$ **BDD** is trivial. For **DT** $\rightarrow$ **BDD** *collapsing operations* are needed, which aim at eliminating redundant nodes, redirecting the edges, and applying isomorphism rules to eliminate transitional nodes with two terminal nodes of same value (Zheng, 2018). Vice versa, **BDD** $\rightarrow$ **DT** can be achieved by *unfolding* the terminal nodes in the BDD until each node has a single parent node (except for the root node).

7. Discussion

Section 6 provides different algorithms that enable the transition FT $\rightarrow$ BDD. By representing FTs as BDDs, standard techniques can be applied for qualitative and quantitative analysis. The transition FT $\rightarrow$ DT has been proposed to obtain Diagnostic Decision Trees (DDTs), which are arguably easier to interpret by non-expert users. We did not find any official publication that addresses the transition BDD $\rightarrow$ FT or DT $\rightarrow$ FT, we suspect this is because it is not clear what the added value of the transition BDD $\rightarrow$ FT is, and that making the transition DT $\rightarrow$ FT needs further exploration. However, a way to carry out such transformation is by first obtaining the cut sets from either the BDD or the DT, and then build the FT in its (con/dis)junctive normal form. Since, the resulting FT may be sub-obtimal, it is necessary to apply minimization rules. The transition in both sides of binary DT $\leftrightarrow$ BDD is straightforward since a binary DT is a special case of a BDD, where the former is less efficient than a reduced ordered BDD.

In Table 1 we summarized the aspects to compare between models. Although all three models were born in graph theory, they serve very different purposes and application domains. Consequently, they adopt different structural requirements and analysis methods. Regarding the construction of these models, DTs stands out by the availability of efficient induction algorithms, FTs on the other hand are mainly "handmade". An ordered BDD and a binary DT have the same structure, but since BDDs allow node-sharing, they can be more compact than binary DTs. All three models carry all sorts of benefits and limitations. The limitations can be addressed via extensions or conversion methods.

8. Conclusions

We compared three well-known graph models, fault trees (FT), decision trees (DT) and binary decision diagrams (BDD). We used a running example to exemplify the properties offered by each model, and discuss conversion methods to transition from one model to the formulation of another. We observed that the transformation FT $\rightarrow$ BDD is well investigated, the transformations DT $\rightarrow$ FT or BDD $\rightarrow$ FT are not investigated and we briefly discussed how to carry them out, and that the transformation DT $\leftrightarrow$ BDD is trivial. We conclude that, given their different purposes and application domains, these models adopt different structural representations and analysis methodologies that entail a variety of benefits and limitations. These limitations may be addressed via conversion methods or extensions. Specific remarks are that BDDs can be considered as a compact representation of binary DTs, since the former allow sub-node sharing, which makes BDDs more efficient at representing logical rules than binary DTs. It is possible to obtain cut sets from BDDs and DTs and construct a FT using the (con/dis)junctive normal form, although this may result in a sub-optimal FT structure.

Acknowledgement. This research has been partially funded by NWO under the grant PrimaVera (https://primavera-project.com) number NWA.1160.18.238.

References

Akers, S. B. (1978). Binary decision diagrams. *IEEE Transactions on Computers* (6), 509–516.

Assaf, T. and J. B. Dugan (2004). Diagnostic expert systems from dynamic fault trees. In *Annual Symposium Reliability and Maintainability, 2004-RAMS*, pp. 444–450. IEEE.

Bartlett, L. M. and J. D. Andrews (2001). An ordering heuristic to develop the binary decision diagram based on structural importance. *Reliability Engineering & System Safety 72*(1), 31–38.

Bartlett, L. M. and J. D. Andrews (2002). Choosing a heuristic for the" fault tree to binary decision diagram" conversion, using neural networks. *IEEE Transactions on Reliability 51*(3), 344–349.

Bollig, B. and I. Wegener (1996). Improving the variable ordering of obdds is np-complete. *IEEE Transactions on Computers 45*(9), 993–1002.

Breiman, L., J. Friedman, C. J. Stone, and R. A. Olshen (1984). *Classification and regression trees*. CRC press.

Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers 100*(8), 677–691.

Bryant, R. E. (1992). Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys (CSUR) 24*(3), 293–318.

Bryant, R. E. (2018). Binary decision diagrams. In *Handbook of Model Checking*, pp. 191–217. Springer.

Dugan, J. B., S. J. Bavuso, and M. A. Boyd (1992). Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability 41*(3), 363–377.

Friedman, S. J. and K. J. Supowit (1987). Finding the optimal variable ordering for binary decision diagrams. In *24th ACM/IEEE Design Automation Conference*, pp. 348–356. IEEE.

IEC (2006). Iec 61025: Fault tree analysis (fta). *International Electrotechnical Commission (IEC), Standards Online*.

Junges, S., D. Guck, J.-P. Katoen, A. Rensink, and M. Stoelinga (2017). Fault trees on a diet: automated reduction by graph rewriting. *Formal Aspects of Computing 29*(4), 651–703.

Kabir, S. (2017). An overview of fault tree analysis and its application in model based dependability analysis. *Expert Systems with Applications 77*, 114–135.

Kubica, M., A. Opara, and D. Kania (2021). Binary decision diagrams. In *Technology Mapping for LUT-Based FPGA*, pp. 25–37. Springer.

Lazarova-Molnar, S., P. Niloofar, and G. K. Barta (2020). Data-driven fault tree modeling for reliability assessment of cyber-physical systems. In *Proceedings of the 2020 Winter Simulation Conference*.

Lee, C.-Y. (1959). Representation of switching circuits by binary-decision programs. *The Bell System Technical Journal 38*(4), 985–999.

Lee, W.-S., D. L. Grosh, F. A. Tillman, and C. H. Lie (1985). Fault tree analysis, methods, and applications - a review. *IEEE Transactions on Reliability 34*(3), 194–203.

Linard, A., D. Bucur, and M. Stoelinga (2019). Fault trees from data: Efficient learning with an evolutionary algorithm. In *Intern'l Symposium on Dependable Software Engineering: Theories, Tools, and Applications*, pp. 19–37. Springer.

Linard, A., M. L. Bueno, D. Bucur, and M. I. A. Stoelinga (2019). Induction of fault trees through bayesian networks.

Madden, M. G. and P. J. Nolan (1970). Generation of fault trees from simulated incipient fault case data. *WIT Transactions on Information and Communication Technologies 6*.

Nauta, M., D. Bucur, and M. Stoelinga (2018). Lift: Learning fault trees from observational data. In *International Conference on Quantitative Evaluation of Systems*, pp. 306–322. Springer.

Quinlan, J. R. (1993). C4.5: Programming for machine learning. *Morgan Kauffmann 38*, 48.

Rauzy, A. and Y. Dutuit (1997). Exact and truncated computations of prime implicants of coherent and non-coherent fault trees within aralia. *Reliability Engineering & System Safety 58*(2), 127–144.

Reay, K. A. (2002). *Efficient fault tree analysis using binary decision diagrams*. Ph. D. thesis, Loughborough University.

Remenyte-Prescott, R. and J. D. Andrews (2008). An enhanced component connection method for conversion of fault trees to binary decision diagrams. *Reliability Engineering & System Safety 93*(10), 1543–1550.

Rokach, L. and O. Maimon (2014). *Data Mining with Decision Trees: Theory and Applications*, Volume 81. World scientific.

Ruijters, E. and M. Stoelinga (2015). Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review 15*, 29–62.

Sarbayev, M., M. Yang, and H. Wang (2019). Risk assessment of process systems by mapping fault tree into artificial neural network. *Journal of Loss Prevention in the Process Industries 60*, 203–212.

Smith, D., B. Veitch, F. Khan, and R. Taylor (2017). Understanding industrial safety: Comparing fault tree, bayesian network, and fram approaches. *Journal of Loss Prevention in the Process Industries 45*, 88–101.

Stamatelatos, M., W. Vesely, J. Dugan, J. Fragola, J. Minarick, and J. Railsback (2002). Fault tree handbook with aerospace applications.

Tao, Y., D. Dong, and P. Ren (2009). Notice of violation of ieee publication principles: Decision trees generation based on fault trees analysis. In *2009 International Forum on Information Technology and Applications*, Volume 2, pp. 178–180. IEEE.

Thomas, T., A. P. Vijayaraghavan, and S. Emmanuel (2020). *Machine Learning Approaches in Cyber Security Analytics*. Springer.

Vesely, W. E., F. F. Goldberg, N. H. Roberts, and D. F. Haasl (1981). Fault tree handbook. Technical report, Nuclear Regulatory Commission Washington DC.

Waghen, K. and M.-S. Ouali (2019). Interpretable logic tree analysis: A data-driven fault tree methodology for causality analysis. *Expert Systems with Applications 136*, 376–391.

Zheng, H. (2018, 10). Binary decision diagrams. `https://www.cse.usf.edu/~haozheng/teach/cda5416/slides/bdd.pdf`. (Accessed on 03/23/2021).