Causality and Temporal Dependencies in the Design of Fault Management Systems

Marco Bozzano
Fondazione Bruno Kessler
bozzano@fbk.eu

Reasoning about causes and effects naturally arises in the engineering of safety-critical systems. A classical example is Fault Tree Analysis, a deductive technique used for system safety assessment, whereby an undesired state is reduced to the set of its immediate causes. The design of fault management systems also requires reasoning on causality relationships. In particular, a fail-operational system needs to ensure timely detection and identification of faults, i.e. recognize the occurrence of run-time faults through their observable effects on the system. Even more complex scenarios arise when multiple faults are involved and may interact in subtle ways.

In this work, we propose a formal approach to fault management for complex systems. We first introduce the notions of fault tree and minimal cut sets. We then present a formal framework for the specification and analysis of diagnosability, and for the design of fault detection and identification (FDI) components. Finally, we review recent advances in fault propagation analysis, based on the Timed Failure Propagation Graphs (TFPG) formalism.

1 Introduction

Modern complex engineering systems, such as satellites, airplanes and traffic control systems need to be able to handle faults. Faults may cause failures, i.e. conditions such that particular components or larger parts of a system are no longer able to perform their required function. As a consequence, faults can compromise system safety, creating a risk of damage to the system itself or to the surrounding infrastructure, or even a risk of harm to humans.

For these reasons, complex system implement fault management systems. There are different ways to deal with faults. *Fault avoidance* tries to prevent design faults, through rigorous development methodologies. Not all faults, however, can be prevented, e.g. hardware faults may happen due to wear-out of components. *Fault tolerance*, on the other hand, aims at making a system robust to faults that may occur during system operation, by using, e.g., a redundant architecture, and by replicating critical components. A fault tolerant system often implements some mechanisms to detect, identify and recover from, faults – i.e. an FDIR (Fault Detection, Identification and Recovery) sub-system. In all cases, the design of complex systems requires evaluating and quantifying the likelihood and the consequences of failures. This process is called *safety assessment*. Classical techniques for safety assessment include Fault Tree Analysis (FTA) and Failure Modes and Effects Analysis (FMEA) [42].

In this paper, we introduce and review some recent work on the design of fault management systems. Our work leverages the use of model-based design methodologies and formal verification and validation techniques based on model checking [28, 4]. We begin by introducing Fault Tree Analysis in Sect. 2. We then present a formal framework for the design of fault detection and identification components, in Sect. 3. We discuss techniques to analyze fault propagation, based on the so-called *Timed Failure Propagation Graphs* (TFPGs) formalism, in Sect.4. Finally, we conclude in Sect.5 by outlining some future directions.

A. Groce and S. Leue: 2nd International Workshop on Causal Reasoning
for Embedded and safety-critical Systems Technologies (CREST'17)
EPTCS 259, 2017, pp. 39–46, doi:10.4204/EPTCS.259.4

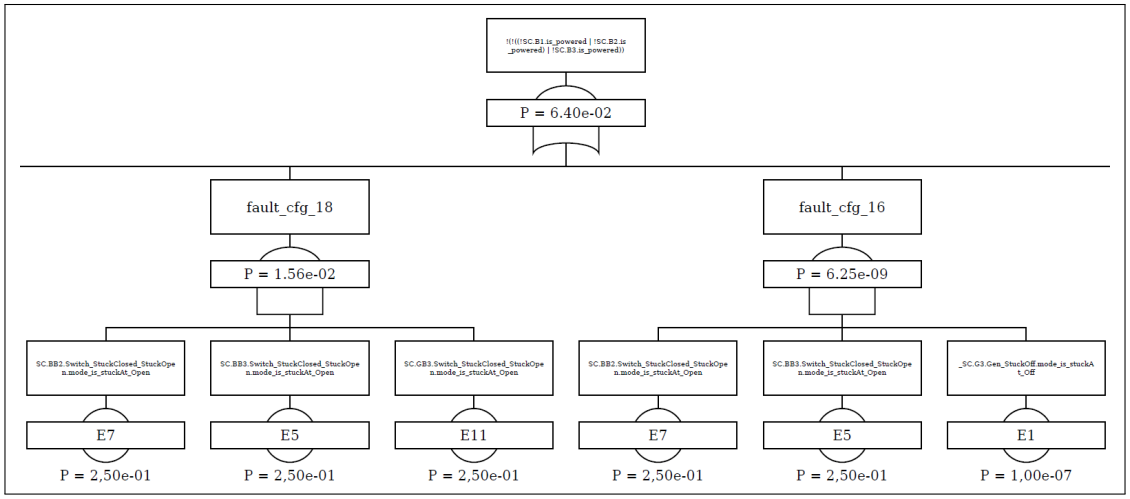

Figure 1: An example fault tree.

2 Fault Tree Analysis

Fault Tree Analysis (FTA) [42] is a classical technique for safety assessment. It is a deductive technique, whereby an undesired state (the so called *top level event* (TLE) or *feared event*) is specified, and the system is analyzed for the possible fault configurations (sets of faults, a.k.a. basic events) that may cause the top event to occur. Fault configurations are arranged in a tree, which makes use of logical gates to depict the logical interrelationships linking such events with the TLE, and which can be evaluated quantitatively, to determine the probability of the TLE. An example fault tree is shown in Figure 1.

Of particular importance in safety analysis is the list of *minimal* fault configurations, i.e. the *Minimal Cut Sets (MCSs)*. More specifically, a cut set is a set of faults that represents a necessary, but not sufficient, condition that may cause a system to reach the top level event. Moreover, minimality implies that every proper sub-set of a MCS is not itself a cut set.

Our approach to FTA is based on formal techniques, and specifically on Model-Based Safety Analysis (MBSA) [21, 5, 12, 37, 18, 20, 16]. MBSA automates complex and error-prone activities such as the generation of MCSs. This is done by looking for minimal fault assignments, in symbolic models where selected variables represent the occurrence of faults [18]. Cut sets are assignments to such variables that lead to the violation of the top level event. Formal verification tools for MBSA include xSAP [7].

Recent work [17] further improves existing algorithms, by providing a fully automated generation of MCSs based on state-of-the-art IC3 techniques [22]. The approach is *anytime*, in that it is able to compute an approximation (lower bound and upper bound) of the set of MCSs, by generating them for increasing cut set cardinality. This approach in inspired by the layering approach of [2], but it improves over it in several respects, such as scalability and convergence. The approach builds upon IC3-based parameter synthesis [25], by providing several enhancements based on the specific features of the problem.

3 A Formal Framework for FDI

Fault Detection and Identification (FDI) is carried out by dedicated modules, called FDI components, running in parallel with the system. The detection task is the problem of understanding whether a component has failed, whereas the identification task aims to understand exactly which fault occurred. In

general, detection and identification may also apply to conditions other than faults.

Typically, faults are not directly observable and their occurrence can only be inferred by observing the effects that they have on the observable parts of the system. An FDI component processes sequences of observations (made available by sensors) and triggers a set of alarms in response to the occurrence of faults.

A formal foundation to support the design of FDI has been described in [14, 15], where a *pattern based* language for the specification of FDI requirements is proposed. [14] focuses on synchronous systems, whereas [15] extends the framework to include the asynchronous composition of the system with the diagnoser.

The first ingredient for specifying an FDI requirement is given by the condition to be monitored, called *diagnosis condition*. The second ingredient is the relation between the diagnosis condition and the raising of an alarm. An *alarm condition* is composed of two parts: the diagnosis condition and the delay. The delay relates the time between the occurrence of the diagnosis condition and the raising of the corresponding alarm. The language supports various forms of delay: exact (EXACTDEL, after exactly n steps), bounded (BOUNDDEL, within n steps) and finite (FINITEDEL, eventually).

The framework supports further aspects that are important for the specification of FDI requirements. The first one is the *diagnosability* [40], i.e., whether the sensors convey enough information to detect the required conditions. A non-diagnosable system (with respect to a given property) is such that no diagnoser exists, that is able to diagnose the property. The above definition of diagnosability might be stronger than necessary, since diagnosability is defined as a global property of the system. In order to deal with non-diagnosable systems, a more fine-grained, local notion of *trace diagnosability* is introduced, where diagnosability is localized to individual traces. This notion extends the results on diagnosability checking from [26].

The second aspect is the *maximality* of the diagnoser, that is, the ability of the diagnoser to raise an alarm as soon as possible and as long as possible, without violating the correctness condition.

The pattern-based language defined in [14, 15] is based on temporal logic. In particular, the patterns are provided with an underlying formal semantics expressed in epistemic temporal logic [34], where the *knowledge* operator is used to express the certainty of a condition, based on the available observations. The language is called Alarm Specification Language with Epistemic operators (ASL_K). Diagnosis conditions and alarm conditions are formalized using LTL with past operators, whereas the definitions of trace diagnosability and maximality require epistemic logic. The full specification, covering the concepts of (system and trace) diagnosability and maximality, is shown in Figure 2.

The formalization encodes properties such as *alarm correctness* (whenever an alarm is raised by the FDI component, then the associated condition did occur), and *alarm completeness* (if an alarm is not raised, then either the associated condition did not occur, or it would have been impossible to detect it, given the available observations). Alternative approaches that define diagnosability as epistemic properties include [29] and [35], where the latter extends the definition of diagnosability to a probabilistic setting. However, these works focus on finite-delay diagnosability only, and do not consider the notion of trace diagnosability.

The framework described in [14, 15] covers several verification and validation problems. The validation problem aims to check whether the requirements capture the desired behaviors and exclude unwanted ones. Known techniques for requirements validation [27] include checking their consistency, their compatibility with some possible scenarios, whether they entail some expected properties and whether they are realizable. The verification problem, on the other hand, checks whether a candidate diagnoser fulfills a given set of requirements. These checks can be done using a model checker for temporal epistemic logic such as MCK [30] or, if the specification falls in the pure LTL fragment, using a model checker such

	Template	$Maximality = False$	$Maximality = True$
$Diag = System$	EXACTDEL	$G(\llcorner A \lrcorner \rightarrow Y^n \beta) \wedge G(\beta \rightarrow X^n \llcorner A \lrcorner)$	$G(\llcorner A \lrcorner \rightarrow Y^n \beta) \wedge G(\beta \rightarrow X^n \llcorner A \lrcorner) \wedge$ $G(\llcorner KY^n \beta \lrcorner \rightarrow \llcorner A \lrcorner)$
	BOUNDDEL	$G(\llcorner A \lrcorner \rightarrow O^{\leq n} \beta) \wedge G(\beta \rightarrow F^{\leq n} \llcorner A \lrcorner)$	$G(\llcorner A \lrcorner \rightarrow O^{\leq n} \beta) \wedge G(\beta \rightarrow F^{\leq n} \llcorner A \lrcorner) \wedge$ $G(\llcorner KO^{\leq n} \beta \lrcorner \rightarrow \llcorner A \lrcorner)$
	FINITEDEL	$G(\llcorner A \lrcorner \rightarrow O\beta) \wedge G(\beta \rightarrow F \llcorner A \lrcorner)$	$G(\llcorner A \lrcorner \rightarrow O\beta) \wedge G(\beta \rightarrow F \llcorner A \lrcorner) \wedge$ $G(\llcorner KO\beta \lrcorner \rightarrow \llcorner A \lrcorner)$
$Diag = Trace$	EXACTDEL	$G(\llcorner A \lrcorner \rightarrow Y^n \beta) \wedge$ $G((\beta \rightarrow X^n \llcorner KY^n \beta \lrcorner) \rightarrow (\beta \rightarrow X^n \llcorner A \lrcorner))$	$G(\llcorner A \lrcorner \rightarrow Y^n \beta) \wedge$ $G((\beta \rightarrow X^n \llcorner KY^n \beta \lrcorner) \rightarrow (\beta \rightarrow X^n \llcorner A \lrcorner)) \wedge$ $G(\llcorner KY^n \beta \lrcorner \rightarrow \llcorner A \lrcorner)$
	BOUNDDEL	$G(\llcorner A \lrcorner \rightarrow O^{\leq n} \beta) \wedge$ $G((\beta \rightarrow F^{\leq n} \llcorner KO^{\leq n} \beta \lrcorner) \rightarrow (\beta \rightarrow F^{\leq n} \llcorner A \lrcorner))$	$G(\llcorner A \lrcorner \rightarrow O^{\leq n} \beta) \wedge$ $G((\beta \rightarrow F^{\leq n} \llcorner KO^{\leq n} \beta \lrcorner) \rightarrow (\beta \rightarrow F^{\leq n} \llcorner A \lrcorner)) \wedge$ $G(\llcorner KO^{\leq n} \beta \lrcorner \rightarrow \llcorner A \lrcorner)$
	FINITEDEL	$G(\llcorner A \lrcorner \rightarrow O\beta) \wedge$ $G((\beta \rightarrow F \llcorner KO\beta \lrcorner) \rightarrow (\beta \rightarrow F \llcorner A \lrcorner))$	$G(\llcorner A \lrcorner \rightarrow O\beta) \wedge$ $G((\beta \rightarrow F \llcorner KO\beta \lrcorner) \rightarrow (\beta \rightarrow F \llcorner A \lrcorner)) \wedge$ $G(\llcorner KO\beta \lrcorner \rightarrow \llcorner A \lrcorner)$

Figure 2: ASL_K specification patterns. Color key: cyan for diagnosability, red for maximality, orange for correctness, yellow for completeness.

as NuSMV [24]. The framework, finally, addresses the problem of automated synthesis of a diagnoser from a given specification. The idea is to generate an automaton that encodes the set of possible states (called *belief states*) that represent the estimation of the state of the system after each observation. Each belief state of the automaton is annotated with the alarms that are satisfied in all the states of the belief state. The algorithm resembles the construction by Sampath [40] and Schumann [41]. It also extends the results of [36], which did not consider maximality and trace diagnosability. Finally, we mention the problem of synthesizing observability requirements, i.e. automatically discovering a set of observations that is sufficient to guarantee diagnosability. This problem is investigated in [10], which also addresses the issue of synthesizing cost-optimal sets of observations.

The framework has been evaluated in the AUTOGEF [3] and FAME [31, 9] projects, funded by the European Space Agency, on a case study based on the EXOMARS Trace Gas Orbiter.

4 Timed Failure Propagation Graphs

Classical safety assessment techniques such as FTA and FMEA do not have a comprehensive support for analyzing the timing of failure propagations, and make it difficult to obtain a global integrated picture of the overall failure behavior of a system. This in turn makes it difficult to check whether a given FDIR architecture is able to handle all possible faults and their propagation effects. To address these issues, *Timed Failure Propagation Graphs* (TFPGs) [39, 1] have been recently investigated as an alternative framework for failure analysis.

TFPGs are labeled directed graphs that represent the propagation of failures in a system, including information on timing delays and mode constraints on propagation links. TFPGs can be seen as an abstract

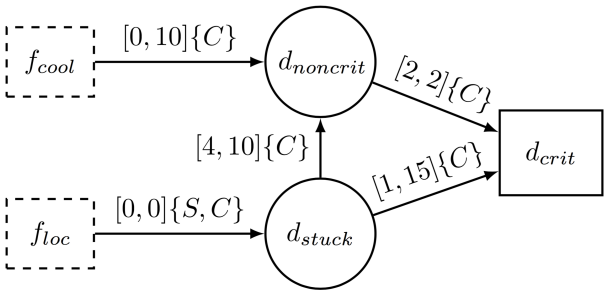

Figure 3: An example TFPG. Dotted boxes are failure mode nodes, solid boxes discrepancy `AND` nodes, and circles discrepancy `OR` nodes.

representation of a corresponding dynamic system of greater complexity, describing the occurrence of failures, their local effects, and the corresponding consequences over time on other parts of the system. TFPGs are a very rich formalism: they allow to model Boolean combinations of basic faults, intermediate events, and transitions across them, possibly dependent on system operational modes, and to express constraints over timing delays. In a nutshell, TFPGs integrate in a single artifact several features that are specific to either FMEA or FTA, enhanced with timing information.

An example TFPG is shown in Figure 3. Nodes represent failure modes and discrepancies (i.e., failure effects). Edges model the temporal dependency between the nodes; they are labeled with lower and upper bounds on the propagation delay, and with labels indicating the system modes where the corresponding propagations are possible. The semantics of TFPGs [11, 8] is such that a node is activated when a failure propagation has reached it. An edge is active whenever the source node is active and the system mode enables the propagation. Discrepancies include both `OR` and `AND` nodes. In the former case, any of the incoming edges may trigger the propagation, whereas in the latter case all of them must be active. If an edge is deactivated during the propagation, due to mode switching, the propagation stops.

TFPGs have been investigated in the frame of the FAME project [31, 9], funded by the European Space Agency (ESA). Here, a novel, model-based, integrated process for FDIR design was proposed, which aims at enabling a consistent and timely FDIR conception, development, verification and validation. More recently, [13, 11, 8] have investigated TFPG-based validation and formal analyses. In particular, [13] focuses on the validation of TFPGs, seen as stand-alone models, using Satisfiability Modulo Theories (SMT) techniques. Validation includes several criteria, such as possibility, necessity and consistency checks, and TFPG refinement. [11] addresses TFPG validation (called *behavioral validation*), and tightening of TFPG delay bounds, with respect to a system model of reference. In this context, the TFPG is seen as an abstract version of the system model, and it is possible to check whether the TFPG is complete, i.e. it represents all behaviors that are possible in the system. Behavioral validation is performed by discharging a set of proof obligations, using either a model checker for metric temporal logic, or by reduction to LTL model checking. Finally, [8] develops algorithms for the automatic synthesis of a TFPG from a reference system model. The generated TFPG is guaranteed to be complete with respect to the system model. Graph synthesis is carried out by using model checking routines to compute sets of MCS and by simplifying the resulting graph by means of a set of static rules. Parameter synthesis techniques are used for edge tightening.

TFPGs have also been applied to several case studies in the context of an ESA deep-space probe design [6], whose mission is characterized by high requirements on on-board autonomy and FDIR. In particular, TFPGs have been applied to the failure analysis of the “Solar Orbiter” (SOLO) satellite.

5 Conclusions

In this paper, we have reviewed some recent work on the design of fault management systems. The concepts of causality and temporal dependencies that arise in this setting have similarities with classical theories of causality, such as counterfactual causality [33, 32]. Such theories are defined using structural equations, but can be readily re-formulated for transition systems [38, 23]. A thorough investigation of the implications of causality theories in the context of fault management systems is part of our future work. We outline here some related work and possible directions for future investigation.

The notion of causality in FTA closely resembles the idea of identifying minimal sets of (necessary and sufficient) causes as in classical causality theories. However, given an effect (TLE), FTA is interested in such sets of causes (i.e., faults – identified beforehand) in all possible scenarios, whereas classical theories focus on identifying the causes in a given scenario of interest. Moreover, in FTA a cause (i.e., MCS) need not be a sufficient condition – sometimes an additional condition on the environment might be needed. Such condition resembles the notion on *contingency* in causality theories, and could be represented using FTA gates (e.g., a pair inhibit gate/conditioning event [42]). In [38, 23], causality is extended to encompass the notions of ordering and non-occurrence of events. This approach extends the ordering analysis proposed in [19, 16].

FDI logic links effects with causes, similar to classical causality theories, but using *observables* only. An alarm, in this context, corresponds to an effect or, more precisely, to a signal which is triggered by the detection/identification of a given effect. Given a fault F and an alarm A, FDI correctness implies that F is (part of) a cause of A, whereas FDI completeness does not necessarily imply that F is a cause of A, since false alarms are possible. However, correctness and completeness together imply that F is the (unique) cause of A. Finally, diagnosability is related to the realizability of FDI logic, and trace diagnosability corresponds to diagnosability in a specific scenario.

Finally, TFPG analyses have similaries with FTA – in fact, TFPG synthesis is built on top of MCS computation. However, TFPGs are more expressive multi-node networks, enriched with time bounds and modes, and nodes may include dependent effects (dicrepancies). A propagation in a TFPG is necessary, in the sense that a discrepancy activation implies the propagation through at least one input propagation link, whereas a propagation is inevitable in the sense that a propagation implies the activation of the correspondent discrepancy. Inevitability may be enforced using time bounds and/or modes.

As part of our future work, we want to analyze more closely the difference between causality and temporal dependencies/temporal correlation. In some scenarios of interest, motivated by practical case studies, it appears that temporal correlation between causes and different effects of the same cause, may lead to identifying a temporal-correlated effect as part of the causes of the effect of interest. Distinguishing causality from temporal correlation would require going beyond the trace-based semantics. We are currently looking for meaningful and sound definitions that can encompass such cases.

Acknowledgments The results presented in this paper are a joint work with several people, including Benjamin Bittner, Alessandro Cimatti, Marco Gario and Stefano Tonetta.

References

[1] S. Abdelwahed, G. Karsai, N. Mahadevan & S.C. Ofsthun (2009): *Practical implementation of diagnosis systems using timed failure propagation graph models*. Instrumentation and Measurement, IEEE Transactions on 58(2), pp. 240–247, doi:10.1109/TIM.2008.2005958.

[2] P.A. Abdulla, J. Deneux, G. Stålmarck, H. Ågren & O. Åkerlund (2004): *Designing Safe, Reliable Systems Using Scade*. In: *Proc. ISoLA 2004*, pp. 115–129, doi:10.1007/11925040_8.

[3] E. Alaña, H. Naranjo, Y. Yushtein, M. Bozzano, A. Cimatti, M. Gario, R. de Ferluc & G. Garcia (2012): *Automated generation of FDIR for the compass integrated toolset (AUTOGEF)*. In: *Proc. DAta Systems In Aerospace, DASIA 2012*, ESA SP 701.

[4] C. Baier & J.-P. Katoen (2008): *Principles of Model Checking*. MIT Press.

[5] P. Bieber, C. Bougnol, C. Castel, J.-P. C. Kehren, S. Metge & C. Seguin (2004): *Safety Assessment with AltaRica*. In: *Building the Information Society, IFIP International Federation for Information Processing* 156, Springer, pp. 505–510, doi:10.1007/978-1-4020-8157-6_45.

[6] B. Bittner (2016): *Formal Failure Analyses for Effective Fault Management: An Aerospace Perspective*. Ph.D. thesis, University of Trento.

[7] B. Bittner, M. Bozzano, R. Cavada, A. Cimatti, M. Gario, A. Griggio, C. Mattarei, A. Micheli & G. Zampedri (2016): *The xSAP Safety Analysis Platform*. In: *Proc TACAS*, Springer, pp. 533–539, doi:10.1007/978-3-662-49674-9_31.

[8] B. Bittner, M. Bozzano & A. Cimatti (2016): *Automated Synthesis of Timed Failure Propagation Graphs*. In: *Proc. IJCAI*, pp. 972–978.

[9] B. Bittner, M. Bozzano, A. Cimatti, R. de Ferluc, M. Gario, A. Guiotto & Y. Yushtein (2014): *An Integrated Process for FDIR Design in Aerospace*. In: *Proc. IMBSA 2014*, *LNCS* 8822, pp. 82–95, doi:10.1007/978-3-319-12214-4_7.

[10] B. Bittner, M. Bozzano, A. Cimatti & X. Olive (2012): *Symbolic Synthesis of Observability Requirements for Diagnosability*. In: *AAAI Conference on Artificial Intelligence*.

[11] B. Bittner, M. Bozzano, A. Cimatti & G. Zampedri (2016): *Automated Verification and Tightening of Failure Propagation Models*. In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016)*, pp. 907–913.

[12] M. Bozzano, A. Cavallo, M. Cifaldi, L. Valacca & A. Villafiorita (2003): *Improving Safety Assessment of Complex Systems: An Industrial Case Study*. In: *Proc. FME*, *LNCS* 2805, pp. 208–222, doi:10.1007/978-3-540-45236-2_13.

[13] M. Bozzano, A. Cimatti, M. Gario & A. Micheli (2015): *SMT-based Validation of Timed Failure Propagation Graphs*. In: *Twenty-ninth AAAI Conference on Artificial Intelligence*, pp. 3724–3730.

[14] M. Bozzano, A. Cimatti, M. Gario & S. Tonetta (2014): *Formal Design of Fault Detection and Identification Components Using Temporal Epistemic Logic*. In: *Proc. TACAS*, pp. 326–340, doi:10.1007/978-3-642-54862-8_22.

[15] M. Bozzano, A. Cimatti, M. Gario & S. Tonetta (2015): *Formal Design of Asynchronous FDI Components using Temporal Epistemic Logic*. *Logical Methods in Computer Science* 11, doi:10.2168/LMCS-11(4:4)2015.

[16] M. Bozzano, A. Cimatti, J.-P. Katoen, V.Y. Nguyen, T. Noll & M. Roveri (2011): *Safety, Dependability and Performance Analysis of Extended AADL Models*. *Computer Journal* 54(5), pp. 754–775, doi:10.1093/comjnl/bxq024.

[17] M. Bozzano, A. Cimatti, C. Mattarei & A. Griggio (2015): *Efficient Anytime Techniques for Model-Based Safety Analysis*. In: *CAV*, pp. 603–621, doi:10.1007/978-3-319-21690-4_41.

[18] M. Bozzano, A. Cimatti & F. Tapparo (2007): *Symbolic Fault Tree Analysis for Reactive Systems*. In: *Proc. ATVA*, *LNCS* 4762, Springer, pp. 162–176, doi:10.1007/978-3-540-75596-8_13.

[19] M. Bozzano & A. Villafiorita (2003): *Integrating Fault Tree Analysis with Event Ordering Information*. *Proc. ESREL 2003*, pp. 247–254.

[20] M. Bozzano & A. Villafiorita (2010): *Design and Safety Assessment of Critical Systems*. CRC Press (Taylor and Francis), an Auerbach Book, doi:10.1201/b10094.

[21] M. Bozzano, A. Villafiorita et al. (2003): *ESACS: An Integrated Methodology for Design and Safety Analysis of Complex Systems*. *Proc. ESREL 2003*, pp. 237–245.

[22] A.R. Bradley (2011): *SAT-Based Model Checking without Unrolling*. In: *VMCAI*, pp. 70–87, doi:10.1007/978-3-642-18275-4_7.

[23] G. Caltais, S. Leue & M. Reza Mousavi (2016): *(De-)Composing Causality in Labeled Transition Systems*. In: *Proc. CREST: Workshop on Causal Reasoning for Embedded and safety-critical Systems Technologies*, doi:10.4204/EPTCS.224.3.

[24] A. Cimatti, E.M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani & A. Tacchella (2002): *NuSMV 2: An OpenSource Tool for Symbolic Model Checking*. In: *CAV*, pp. 359–364, doi:10.1007/3-540-45657-0_29.

[25] A. Cimatti, A. Griggio, S. Mover & S. Tonetta (2013): *Parameter synthesis with IC3*. In: *Proceedings of FMCAD*, IEEE, pp. 165–168, doi:10.1109/FMCAD.2013.6679406.

[26] A. Cimatti, C. Pecheur & R. Cavada (2003): *Formal Verification of Diagnosability via Symbolic Model Checking*. In: *IJCAI*, pp. 363–369.

[27] A. Cimatti, M. Roveri, A. Susi & S. Tonetta (2012): *Validation of requirements for hybrid systems: A formal approach*. *ACM Transactions on Software Engineering and Methodology* 21(4), p. 22, doi:10.1145/2377656.2377659.

[28] E.M. Clarke, O. Grumberg & D.A. Peled (2000): *Model Checking*. MIT Press.

[29] J. Ezekiel, A. Lomuscio, L. Molnar & S.M. Veres (2011): *Verifying Fault Tolerance and Self-Diagnosability of an Autonomous Underwater Vehicle*. In: *IJCAI*, pp. 1659–1664.

[30] P. Gammie & R. Van Der Meyden (2004): *MCK: Model checking the logic of knowledge*. In: *CAV*, Springer, pp. 256–259, doi:10.1007/978-3-540-27813-9_41.

[31] A. Guiotto, R. De Ferluc, M. Bozzano, A. Cimatti, M. Gario & Y.Yushtein (2014): *Fame process: A dedicated development and V&V process for FDIR*. In: *Proc. DASIA*, *European Space Agency, (Special Publication)* ESA SP 725.

[32] J. Halpern (2015): *A modification of the Halpern-Pearl definition of causality*. In: *Proc. IJCAI 2015*, pp. 3022–3033.

[33] J.Y Halpern & J. Pearl (2005): *Causes and explanations: A structural-model approach. Part I: Causes*. *The British journal for the philosophy of science* 56(4), pp. 843–887, doi:10.1093/bjps/axi147.

[34] J.Y. Halpern & M.Y Vardi (1989): *The complexity of Reasoning About Knowledge and Time. Lower Bounds*. *Journal of Computer and System Sciences* 38(1), pp. 195–237, doi:10.1016/0022-0000(89)90039-1.

[35] X. Huang (2013): *Diagnosability in Concurrent Probabilistic Systems*. In: *AAMAS*, pp. 853–860.

[36] S. Jiang & R. Kumar (2001): *Failure Diagnosis of Discrete Event Systems with Linear-time Temporal Logic Fault Specifications*. In: *IEEE Transactions on Automatic Control*, pp. 128–133, doi:10.1109/ACC.2002.1024792.

[37] A. Joshi, S.P. Miller, M. Whalen & M.P.E. Heimdahl (2005): *A Proposal for Model-Based Safety Analysis*. In: *Proc. DASC*, IEEE Computer Society, doi:10.1109/DASC.2005.1563469.

[38] F. Leitner-Fischer & S. Leue (2013): *Probabilistic Fault Tree Synthesis using Causality Computation*. *International Journal of Critical Computer-Based Systems* 4(2), pp. 119–143, doi:10.1504/IJCCBS.2013.056492.

[39] A. Misra, J. Sztipanovits, A. Underbrink, R. Carnes & B. Purves (1992): *Diagnosability of Dynamical Systems*. In: *Third International Workshop on Principles of Diagnosis.*

[40] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen & D. C. Teneketzis (1996): *Failure diagnosis using discrete-event models*. *IEEE Transactions on Control Systems Technology* 4(2), pp. 105–124, doi:10.1109/87.486338.

[41] A. Schumann (2004): *Diagnosis of discrete-event systems using binary decision diagrams*. *Workshop on Principles of Diagnosis (DX'04)*, pp. 197–202.

[42] W.E. Vesely, M. Stamatelatos, J. Dugan, J. Fragola, J. Minarick III & J. Railsback (2002): *Fault Tree Handbook with Aerospace Applications*. Technical Report, NASA.