The Dynamic Fault Tree Rare Event Simulator

Carlos E. Budde[1(✉)], Enno Ruijters[2], and Mariëlle Stoelinga[1,3]

[1] Formal Methods and Tools, University of Twente, Enschede, The Netherlands
{c.e.budde,m.i.a.stoelinga}@utwente.nl
[2] BetterBe, Enschede, The Netherlands
mail@ennoruijters.nl
[3] Department of Software Science, Radboud University, Nijmegen, The Netherlands

Abstract. The dynamic-fault-tree rare event simulator, DFTRES, is a statistical model checker for dynamic fault trees (DFTs), supporting the analysis of highly dependable systems, e.g. with unavailability or unreliability under 10^{-30}. To efficiently estimate such low probabilities, we apply the Path-ZVA algorithm to implement Importance Sampling with minimal user input. Calculation speed is further improved by selective automata composition and bisimulation reduction. DFTRES reads DFTs in the Galileo or JANI textual formats. The tool is written in Java 11 with multi-platform support, and it is released under the GPLv3. In this paper we describe the architecture, setup, and input language of DFTRES, and showcase its accurate estimation of dependability metrics of (resilient) repairable DFTs from the FFORT benchmark suite.

1 Introduction

Our modern societies depend heavily on complex electro-mechanical systems, making it essential to ensure that such systems are reliable. An industry-standard technique to assess reliability is fault tree analysis. However, an unavoidable bottleneck of this technique is that exact analysis becomes too memory-intensive for complex *dynamic fault trees* (DFTs [6]). Alternatively, Monte Carlo simulation can be used to statistically estimate the likelihood of undesired events such as system failure. Although constant in memory usage, this approach takes unacceptably long times to converge when a system failure is rare, i.e. highly unlikely. An effective solution then is to use *rare event simulation* (RES [15]).

This paper presents DFTRES[1]: a statistical analysis tool for DFTs that applies *Importance Sampling* (IS [10]). IS is one of the most efficient approaches to perform RES analyses, and allows DFTRES to drastically speed up accurate estimations of rare failures in repairable DFTs. Whereas most RES techniques rely on expert input, DFTRES allows a fully automatic application of IS [18].

[1] Available at https://github.com/utwente-fmt/DFTRES.

This work was partially funded by NWO project 15474 (*SEQUOIA*).

The original version of this chapter was revised: Reference 5 has been corrected. The correction to this chapter is available at https://doi.org/10.1007/978-3-030-59854-9_21

M. Gribaudo et al. (Eds.): QEST 2020, LNCS 12289, pp. 233–238, 2020.
https://doi.org/10.1007/978-3-030-59854-9_17

Related Work. Various tools exist to analyse DFTs, see [19]. The model checker Storm [11] offers a DFT front-end. Storm produces exact results through model checking, requiring the full state-space, and does not support repairs. Other tools for rare event simulation of automata include Plasma Lab [12], where the user must manually parameterise the model, and FIG [2] and modes [3], which implement a RES method other than IS, less suited to analyse DFTs.

Previous versions of DFTRES were experimentally evaluated in [18] and [9], where it was called "FTRES." In Sect. 3 we mention new features that have been implemented ever since, most prominently weak-bisimulation reduction during initial automata composition, and so-called forcing for time-bounded properties.

Organization of the Paper. After some background in Sect. 2, we explain the operation and structure of DFTRES in Sect. 3, and show its performance in Sect. 4.

2 Rare Event Simulation for Fault Trees

Fault trees are an industry-standard graphical formalism for reliability analysis [19]. A (dynamic) fault tree models possible failures of a system by decomposing it into *basic events*, denoted by circles and representing elemental failure causes of components, and *gates*, denoted by various symbols and representing how failures interact and which combinations of smaller failures lead to system failure. Figure 1 shows an example: the top *AND*-gate (G1) means that both G2 and A must fail for the system to fail. G2 is a *SPARE*-gate, meaning that B and its spares S1 and S2 must fail; but the spares cannot not fail before they are used. Insp denotes a periodic simultaneous inspection and repair of all basic events.

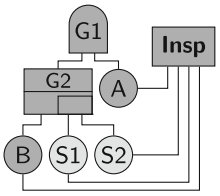


Fig. 1. A repairable DFT

When basic events are decorated with failure probabilities or rates, it is possible to compute numerical resilience metrics of the system. These include *reliability*, the probability that the system remains functional until some given

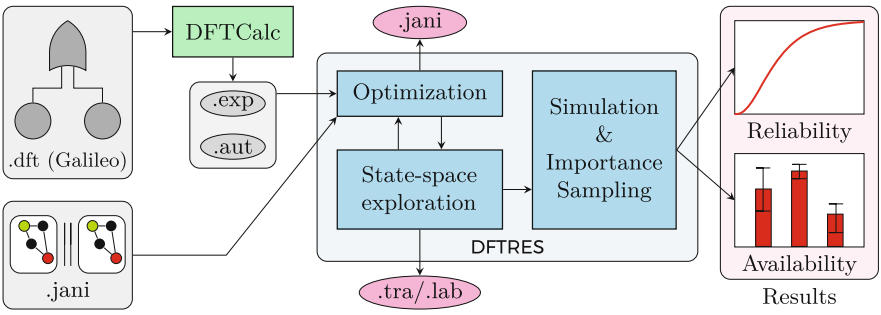


Fig. 2. The overall structure of DFTRES

"mission time," and also (for systems with repairable components) *availability*, the average fraction of time that the system is functional.

For large fault trees, particularly with complex dynamic gates describing time-dependent failure effects or with complex repair policies, exact numerical analysis becomes infeasible due to time and memory exhaustion. Such systems may still be analyzed using Monte Carlo simulation, at the expense of requiring many simulation runs for high accuracy, particularly when the event of interest (system failure) is highly unlikely.

DFTRES addresses this problem using Importance Sampling with the Path-ZVA algorithm [14]. This IS scheme effectively adjusts the failure rates to make system failures more likely, performs simulation runs, then corrects for the adjusted failure rates to estimate the original failure probability. This allows for high-accuracy estimations in relatively few simulation samples [18].

3 DFTRES

The architecture of DFTRES is depicted in Fig. 2: a fault tree in the widely-used Galileo format [9,11,17,20] (e.g. Fig. 3) is translated into a network of automata by DFTCalc [1], and input into DFTRES. Alternatively, a network of automata in JANI format [4] can be input directly. DFTRES then performs several optimizations to reduce the state-space, generates (a part of) the composed state-space, and performs (IS) simulations to estimate numeric metrics such as system reliability. The automata and composed system can also be output for analysis by other tools.

```
1 toplevel "G1";
2 "G1" and "G2" "A";
3 "G2" wsp "B" "S1" "S2";
4 "A"  lambda=1.7e-5 dorm=1 phases=2 interval=1;
5 "B"  lambda=1.1e-3 dorm=1 phases=3 interval=2;
6 "S1" lambda=0.0021 dorm=0 phases=1 interval=1;
7 "S2" lambda=0.0021 dorm=0 phases=1 interval=1;
8 "Insp" 2insp4 "A" "B" "S1" "S2";
```

Fig. 3. (Extended) Galileo for Fig. 1

DFTRES begins its analysis with an optimization stage (new since [9]): transitions of the automata that cannot synchronize are removed and all automata are reduced modulo weak bisimulation. Further, so-called *don't care* optimization is performed by collapsing and discarding groups of states without observable behavior. Pairs of automata with a small composed state-space (by default at most 256 states) are composed and reduced again, and this process is repeated until no more compositions can be made.

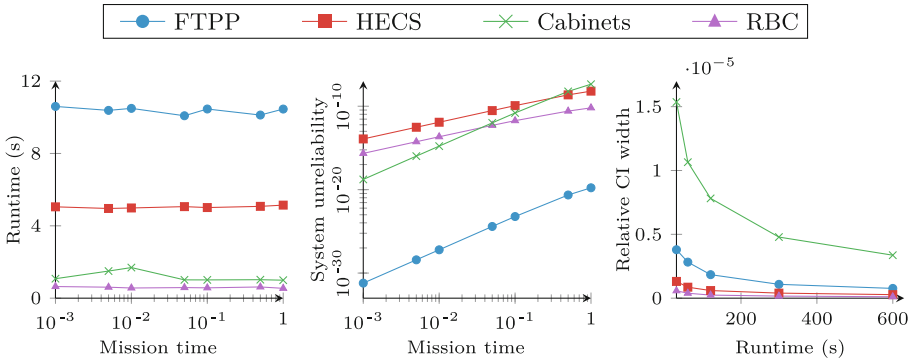
Finally, to compute relevant metrics, simulation is performed using IS, namely the Path-ZVA algorithm [14] and, (new since [9]) for time-bounded properties, forcing [13]. Supported metrics are reliability (time-bounded or -unbounded reachability) and availability (steady-state probability). Mean time to failure (expected reward) can also be estimated, but not using IS. Simulation runs are sampled, in parallel on multi-core systems, until a specified time bound or simulation number is reached, or a desired relative or absolute estimated error is reached. Results are presented as (by default) 95% *confidence intervals* (CIs)[2].

[2] While every effort is made to provide accurate confidence intervals, their coverage can fall considerably below 95% due to the extreme probability distributions involved [8].

DFTRES is released under the GPLv3, and is cross-platform due to its implementation in Java, without run-time dependencies. It requires only a Java compiler and Make [7] to build. Galileo input is provided by DFTCalc, which is supported on Linux and Mac. DFTRES is designed to be easily extensible to additional input formats and IS schemes. DFTRES's command-line interface provides many options, but typically requires only the model file, property, and desired accuracy. For instance, "`java -jar DFTRES.jar -a --relErr 0.05 model.dft`" estimates availability (`-a`) to a relative error of 0.05. More examples can be found in an artifact prepared for experimental reproduction [5].

4 Experimental Evaluation

We estimated the (un)reliability and (un)availability of four repairable DFTs from the FFORT benchmark [17]: Cabinets-2-2, FTPP-2-2-repair, HECS-2-2-repair, and RBC. All experiments ran in an 8-core Intel® i7-6700 with 24 GB RAM. The results are shown in Fig. 4.



(a) Unreliability: runtime (b) Unreliability: estimate (c) Unavailability: rel. error

Fig. 4. Experimental results

We estimated the system unreliability (i.e. the probability that the system fails before) mission times 1.0, 0.5, 0.1, 0.05, 0.01, 0.005, and 0.001. We built 95% CIs for 5% relative error: Fig. 4b shows how the unreliability decreases exponentially—from right to left—as a function of the mission time. Figure 4a plots the runtime needed for CI with 5% accuracy. Unlike traditional simulation, runtime is almost independent of the value being estimated. Instead, the model structure and complexity is the primary factor affecting analysis time, mainly governed by the length of the shortest path(s) to a rare event.

Figure 4c shows unavailability analyses. We let estimations run for 0.5, 1, 2, 5, and 10 min, and measured the relative width of the resulting CI. With longer runtime DFTRES builds more accurate, narrower intervals: the precision improves

approximately as the square root of time, which can be explained by observing that the standard error of the mean decreases as the square root of the number of samples.

In [5] we provide an artifact to easily reproduce our experiments. It runs in Debian-based Linux distributions, such as the virtual machine available at https://figshare.com/articles/tacas20ae_ova/9699839.

References

1. Arnold, F., Belinfante, A., Van der Berg, F., Guck, D., Stoelinga, M.: DFTCalc: a tool for efficient fault tree analysis. In: Bitsch, F., Guiochet, J., Kaâniche, M. (eds.) SAFECOMP 2013. LNCS, vol. 8153, pp. 293–301. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40793-2_27
2. Budde, C.E.: FIG: the finite improbability generator. TACAS 2020. LNCS, vol. 12078, pp. 483–491. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45190-5_27
3. Budde, C.E., D'Argenio, P.R., Hartmanns, A., Sedwards, S.: An efficient statistical model checker for nondeterminism and rare events. Int. J. Softw. Tools Technol. Transf. 1–22 (2020). https://doi.org/10.1007/s10009-020-00563-2
4. Budde, C.E., Dehnert, C., Hahn, E.M., Hartmanns, A., Junges, S., Turrini, A.: JANI: quantitative model and tool interaction. In: Legay, A., Margaria, T. (eds.) TACAS 2017. LNCS, vol. 10206, pp. 151–168. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54580-5_9
5. Budde, C.E., Ruijters, E., Stoelinga, M.: The dynamic fault tree rare event simulator: experimental replication package (2020). https://figshare.com/articles/software/The_Dynamic_Fault_Tree_Rare_Event_Simulator/12235889, https://doi.org/10.6084/m9.figshare.12235889.v2
6. Dugan, J., Boyd, S.B.M.: Fault trees and sequence dependencies. In: Annual Proceedings on Reliability and Maintainability Symposium, pp. 286–293 (1990). https://doi.org/10.1109/ARMS.1990.67971
7. Feldman, S.I.: Make - a program for maintaining computer programs. Softw. Pract. Exp. **9**(4), 255–265 (1979). https://doi.org/10.1002/spe.4380090402
8. Glynn, P.W., Rubino, G., Tuffin, B.: Robustness properties and confidence interval reliability issues. In: Rubino and Tuffin [16], pp. 63–84. https://doi.org/10.1002/9780470745403.ch4
9. Hartmanns, A., et al.: The 2019 comparison of tools for the analysis of quantitative formal models. In: TACAS. LNCS, vol. 11429, pp. 69-92. Springer (2019). https://doi.org/10.1007/978-3-030-17502-3_5
10. Heidelberger, P.: Fast simulation of rare events in queueing and reliability models. ACM Trans. Model. Comput. Simul. **5**(1), 43–85 (1995). https://doi.org/10.1145/203091.203094
11. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.: The probabilistic model checker storm. arXiv e-prints arXiv:2002.07080 (2020). https://arxiv.org/abs/2002.07080
12. Jégourel, C., Legay, A., Sedwards, S.: Command-based importance sampling for statistical model checking. Theor. Comput. Sci. **649**, 1–24 (2016). https://doi.org/10.1016/j.tcs.2016.08.009

13. Nicola, V.F., Shahabuddin, P., Nakayama, M.: Techniques for fast simulation of models of highly dependable systems. IEEE Trans. Reliab. **50**(3), 246–264 (2001). https://doi.org/10.1109/24.974122
14. Reijsbergen, D., de Boer, P.T., Scheinhardt, W., Juneja, S.: Path-ZVA: general, efficient and automated importance sampling for highly reliable Markovian systems. ACM TOMACS **28**(3), 22:1–22:25 (2018). https://doi.org/10.1145/3161569
15. Rubino, G., Tuffin, B.: Introduction to rare event simulation. In: Rubino and Tuffin [16], pp. 1–13. https://doi.org/10.1002/9780470745403.ch1
16. Rubino, G., Tuffin, B. (eds.): Rare Event Simulation Using Monte Carlo Methods. Wiley, Hoboken (2009). https://doi.org/10.1002/9780470745403
17. Ruijters, E., et al.: FFORT: a benchmark suite for fault tree analysis. In: ESREL, pp. 878–885 (2019). https://doi.org/10.3850/978-981-11-2724-3_0641-cd
18. Ruijters, E., Reijsbergen, D., de Boer, P.T., Stoelinga, M.: Rare event simulation for dynamic fault trees. Reliab. Eng. Syst. Safety **186**, 220–231 (2019). https://doi.org/10.1016/j.ress.2019.02.004
19. Ruijters, E., Stoelinga, M.: Fault tree analysis: a survey of the state-of-the-art in modeling, analysis and tools. Comput. Sci. Rev. **15–16**, 29–62 (2015). https://doi.org/10.1016/j.cosrev.2015.03.001
20. Sullivan, K.J., Dugan, J.B.: Galileo user's manual & design overview, v2.1-alpha (1998). www.cse.msu.edu/~cse870/Materials/FaultTolerant/manual-galileo.htm