Coral: a tool for compositional reliability and availability analysis[⋆]

Hichem Boudali[1], Pepijn Crouzen[2,⋆⋆], and Mariëlle Stoelinga[1]

[1] Department of Computer Science, University of Twente,
P.O.Box 217, 7500AE Enschede, The Netherlands.
[2] Saarland University, Department of Computer Science,
D-66123 Saarbrücken, Germany.
hboudali@cs.utwente.nl,crouzen@alan.cs.uni-sb.de,marielle@cs.utwente.nl

Dynamic Fault Trees in Reliability Engineering. Reliability and availability measures, such as system failure probability during a given mission time and system mean-time-between-failures, are often important measures to assess in embedded systems design. There exist several techniques and formalisms for reliability/availability assessment. One such formalism is dynamic fault trees (DFT) [6]. DFTs are a graphical, high-level and versatile formalism to analyze the reliability of computer-based systems, describing the failure of a system in terms of the failure of its components. A DFT is comprised of basic events (modeling the failure of physical components) and gates (modeling how component failures induce system failures). DFTs extend standard (or static) fault trees by allowing the modeling of complex system components' behaviors and interactions. Typically, a DFT is analyzed by first converting it into a continuous-time Markov chain (CTMC) and by then computing the reliability measures from this CTMC. For over a decade now, DFTs have been experiencing a growing success among reliability engineers.

Unfortunately, a number of issues remain when using DFTs, most notably: (1) the DFT semantics is rather imprecise and the lack of formality has, in some cases, led to undefined behavior and misinterpretation of the DFT model. (2) DFTs lack modular analysis. That is, even if independent sub-modules exist in a DFT module, these sub-modules can not always be solved separately. Consequently, DFTs become vulnerable to the well-known state-space explosion problem; that is the size of the underlying CTMC grows exponentially with the number of basic events in the DFT. (3) DFTs also lack modular model-building, i.e. there are some rather severe restrictions on the type of allowed inputs to certain gates which greatly diminish the modeling flexibility and power of DFTs.

Our solution: Compositional modeling and analysis of DFTs. We have provided a formal semantics to DFTs by translating DFTs to input/output interactive Markov chains (I/O-IMCs); I/O-IMCs extend CTMCs with discrete input, output and internal actions [8, 4].

By being formal, our semantics provides a rigorous basis for the interpretation and analysis of DFTs, thus solving issue (1) mentioned above. Our semantics is also fully compositional. That is, the semantics of a DFT is expressed in terms of the semantics of its elements (i.e. basic events and gates). More precisely, we provide an I/O-IMC for each DFT element and obtain the semantics of the whole DFT by composing the semantics of all elements, using the parallel composition operator available for I/O-IMCs. This method enables an efficient analysis of DFTs through compositional aggregation, which helps to alleviate the state-space explosion problem by incrementally building the DFT state space. Our technique is completely modular, which allows us to overcome issue (2).

[⋆] This research has been partially funded by the Netherlands Organisation for Scientific Research (NWO) under FOCUS/BRICKS grant number 642.000.505 (MOQS); the EU under grant number IST-004527 (ARTIST2); and by the DFG/NWO bilateral cooperation programme under project number DN 62-600 (VOSS2).

[⋆⋆] The majority of this work was done while the author was at the University of Twente.

We have also tackled issue (3) and lifted some previously enforced restrictions on DFTs, thus extending the applicability of the DFT model.

We have implemented our methodology by developing a prototype tool, called Coral (COmpositional Reliability and Availability anaLysis). This tool uses the CADP tool set [7]. We have compared our approach to the analysis tool Galileo [2], one of the standard analysis tools for DFTs. In many cases, our approach compares favorably to Galileo and demonstrates its effectiveness by seriously reducing the state space to be analyzed [4]. Our prototype tool takes as input a DFT in Galileo's textual format and a composition script describing, in a simple textual format, the order in which the I/O-IMC models must be composed. The tool proceeds in three steps:

1. **Translation:** The DFT is translated into a group of I/O-IMC models. In particular, each DFT element is translated into a corresponding elementary (with few states and few transitions) I/O-IMC model.
2. **Compositional aggregation:** Using the composition script the I/O-IMC models are iteratively composed, abstracted and aggregated until one I/O-IMC model remains.
3. **Analysis:** In most cases, the resulting I/O-IMC model can be easily transformed into a CTMC. Standard analysis techniques for CTMCs, such as transient analysis, implemented in the CADP tool set can then be applied to compute for instance the system unreliability. In some cases the structure of the DFT causes the resulting I/O-IMC to contain non-determinism. The I/O-IMC can then be transformed into a continuous-time Markov decision process for further analysis [3].

The compositional semantics also allows the DFT formalism to be easily extended or modified. In [5] we show how several of these extensions (for instance, repairable components) could be realized in our framework. Such extensions only impact the translation to the corresponding I/O-IMC models of the modified or added DFT elements. Thus only the translation step (i.e. step 1) of the tool is affected.

Future work. The focus of the future work will be to further improve and automate the tool. In particular, we plan to derive the composition script from the DFT model automatically. Since the order in which the I/O-IMCs are composed impacts the size of the generated state space, we are looking for heuristics leading to a minimal state space. Other topics for future research include the use of interface constraints [9] within our methodology. Finally, we are also currently looking at other reliability/availability formalisms and architectural design languages (such as the architecture analysis and design language (AADL) standard and its error model annex [1]) and trying to map their constructs into I/O-IMC models.

References

1. Architecture Analysis and Design Language (AADL). SAE standards AS5506, Nov 2004.
2. Galileo DFT analysis tool. http://www.cs.virginia.edu/~ftree.
3. C. Baier, H. Hermanns, J. P. Katoen, and B. R. Haverkort. Efficient computation of time-bounded reachability probabilities in uniform continuous-time markov decision processes. *Theor. Comput. Sci.*, 345(1):2–26, 2005.
4. H. Boudali, P. Crouzen, and M.I.A. Stoelinga. A compositional semantics for Dynamic Fault Trees in terms of Interactive Markov Chains. *Submitted to ATVA 2007 conference.*
5. H. Boudali, P. Crouzen, and M.I.A. Stoelinga. Dynamic fault tree analysis using input/output interactive markov chains. *Accepted to DSN 2007 conference.*
6. J. B. Dugan, S. J. Bavuso, and M. A. Boyd. Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41(3):363–377, September 1992.
7. H. Garavel, R. Mateescu, F. Lang, and W. Serwe. Cadp 2006: A toolbox for the construction and analysis of distributed processes. *Accepted to CAV 2007 conference.*
8. H. Hermanns. *Interactive Markov Chains*, volume 2428/2002 of *LNCS*.
9. F. Lang. Refined interfaces for compositional verification. In *International Conference on Formal Techniques for Networked and Distributed Systems (FORTE)*, 2006.