Breaking the Limits of Redundancy Systems Analysis

Clemens Dubslaff, Kai Ding, Andrey Morozov, Christel Baier, and Klaus Janschek

Technische Universität Dresden, Germany.
E-mail: {clemens.dubslaff, kai.ding, andrey.morozov, christel.baier, klaus.janschek}@tu-dresden.de

Redundancy mechanisms such as triple modular redundancy protect safety-critical components by replication and thus improve systems fault tolerance. However, the gained fault tolerance comes along with costs to be invested, e.g., increasing execution time, energy consumption, or packaging size, for which constraints have to be obeyed during system design. This turns the question of finding suitable combinations of components to be protected into a challenging task as the number of possible protection combinations grows exponentially in the number of components. We propose *family-based approaches* to tackle the combinatorial blowup in redundancy systems modeling and analysis phases. Based on systems designed in SIMULINK we show how to obtain models that include all possible protection combinations and present a tool chain that, given a probabilistic error model, generates discrete Markov chain families. Using symbolic techniques that enable concise family representation and analysis, we show how SIMULINK models of realistic size can be protected and analyzed with a single family-based analysis run while a one-by-one analysis of each protection combination would clearly exceed any realistic time constraints.

Keywords: Redundancy, fault tolerance, model-based stochastic analysis, probabilistic model checking, SIMULINK.

1. Introduction

Fault tolerance plays a significant role in the design of safety-critical systems as it enables a system to continue functioning in the presence of faults. The key underlying technique to achieve fault tolerance is provided by *redundancy*, i.e., mechanisms to discover and evaluate faults depending on the behaviors of replicated system components. For instance, components can be protected by *triple modular redundancy (TMR)* where the component is triplicated and their results are processed through a majority voting mechanism into a single output. Though protecting components reduces the overall probability of failure, it increases costs in terms of, e.g., the systems packaging size, energy consumption, production costs, or execution time. These costs hinder to apply the naive approach of protecting all system components to maximize reliability and motivate the task of protecting only some components towards a good tradeoff between reliability and costs. However, the number of possible protection combinations grows exponentially in the number of components, which renders the design of redundancy systems a challenging task: Systems designers might have to model and analyze a huge amount of protection combinations before they find a combination that achieves a satisfactory tradeoff. As even the (tradeoff) analysis of a single protection combination can take a significant amount of time, this iterative development cycle is likely to exceed time constraints. Furthermore, one might be not interested in a protection combination that is only satisfactory but optimal with respect to the tradeoff, rendering an exhaustive analysis of all combinations inevitable.

To tackle the aforementioned challenges when modeling and analyzing redundancy systems, we propose to use *family-based approaches* (see, e.g., [3, 8, 10, 22]) where a single *family model* comprises the behaviors of all protection combinations. First, such approaches avoid modeling each protection combination individually and to use an automated generation of any combination out from the family model. Thus, the design process of redundancy systems turns from alternating modeling and analyzing phases to a single modeling phase followed by an analysis phase those results yield suitable protection combinations. Second, family-based approaches enable an *all-in-one analysis* where the family model is analyzed in a single run instead of analyzing each family member in isolation. This allows for exploiting commonalities between the family members using symbolic representations and analysis operations, e.g., by *binary decision diagrams* (BDDs, cf. [4, 18]). In the feature-oriented systems domain [22], such symbolic techniques have shown drastic speedups for quantitative analyses [10] especially when family members share lots of behaviors. As redundancy mechanisms introduce several identical components in the system, redundancy systems are naturally eligible for such a concise model representation and analyses using symbolic methods. Especially when one is interested in *optimal* protection combinations for which an exhaustive analysis of all family members can hardly be avoided, such an all-in-one

Proceedings of the 29th European Safety and Reliability Conference.
Edited by Michael Beer and Enrico Zio

analysis might mitigate the limits induced by the combinatorial blow-up in the number of system components.

We demonstrate the benefits of family-based approaches for the modeling and analysis of redundancy systems by a tool chain where redundancy mechanisms are introduced in SIMULINK models and a family-based analysis is performed on discrete Markov chain (DTMC) families using the symbolic probabilistic model checker PRISM [15]. Our SIMULINK models with redundancy are obtained by an annotative approach where SIMULINK blocks following the model-based redundancy technique (MORE, cf. [9]). Specifically, we consider SIMULINK redundancy design patterns such as *comparison*, *voting* (i.e., TMR), and *sparing* that replace SIMULINK blocks amendable for protection in combination with a probabilistic error model. For the automated generation of the DTMC family model, we employ SIMPARS and OPENERRORPRO [20].

As illustrative case studies we issue two control loops modeled in SIMULINK: a proportional-integral-derivative (PID) controller and a velocity control loop (VCL) of an aircraft velocity model borrowed from the SIMULINK example set [1]. While the PID family comprises 64 members with comparably small model sizes the VCL has 65 536 family members those state space exceeds 10^{11} states, making symbolic methods for their analysis inevitable. First, we synthesize protection combinations in the PID example that are optimal with respect to tradeoffs expressed through *quantiles* [5, 6], i.e., maximizing the number of control loop rounds where the probability of failure is guaranteed to be below a given threshold. This example shows that our approach can be used to investigate properties that can hardly be analyzed using de-facto standard simulative approaches. Second, we determine protection combinations in the VCL model that are Pareto-optimal with respect to the probability of failure within two rounds of the control loop and its execution time, solving the following problems:

(1) minimize the probability of failure while staying within a given execution time, and
(2) minimize execution time while not exceeding a certain probability of failure.

We show that for the VCL model an all-in-one analysis gains a speedup in three orders of magnitude compared to the one-by-one analysis. In particular, the presented all-in-one analysis manages to obtain results in less than 5 hours while a one-by-one analysis would require around 250 days of computing time, clearly exceeding acceptable time constraints in systems design.

Outline. Section 2 discusses related work and techniques used in the paper. The general approach towards SIMULINK family models with redundancy and their translation into DTMC families is described in Section 3. In Section 4 the analysis of PID and VCL families is carried out and optimal protections are synthesized. We close our paper and discuss further work in Section 5.

2. Related Work and Concepts

Probabilistic Model Checking. For the analysis of Markovian stochastic models, probabilistic model checking (PMC, cf. [4]) is an automated technique that has been successfully applied to numerous real-world case studies to analyze systems performance and Quality of Service. We rely on *discrete Markov chains (DTMCs)* as stochastic model, i.e., state-transition graphs where the transitions are purely probabilistic. *Symbolic methods* can compete with the well-known state-space explosion problem by concise model representations, e.g., through *binary decision diagrams* (BDDs, cf. [4, 18]). The prominent PMC tool PRISM [15] uses *multi-terminal BDDs* [2, 11] for a purely symbolic analysis without the use of an enumerative model representation. It is well-known that the size of symbolic representations by BDDs is sensitive to the so-called *variable order*. In [14], variable-reordering techniques towards a compact state-space representation have been introduced for PRISM. This enabled the analysis of large-scale systems and speedup their analysis, e.g., for the all-in-one family-based analysis of feature-oriented systems [8, 10]. Family-based synthesis using symbolic PMC towards optimal system configurations has been detailed in [3].

Reliability Analysis and Variability in SIMULINK. Reliability analysis of SIMULINK using verification techniques has been considered in [13], where a handcrafted tool chain from SIMULINK to the programming language LUSTRE in combination with the SCADE design verifier has been used. The MODIFI approach presented in [21] implements fault-injection capabilities in SIMULINK to evaluate error handling mechanisms. However, their error model is non-probabilistic and their analysis method is based on simulations – an extension of this work towards a stochastic reliability analysis thus is not as easy. In [7] a transformation of SIMULINK models to continuous-time Markov chains is presented, used to perform dependability analysis using the model checker PRISM. Variability modeling in SIMULINK has been considered in [12], using a direct implementation of a delta-oriented approach. As we will show, we do not have to rely on this powerful formalism to model families of redundancy systems. However, their approach could possibly be combined towards a family-based analysis of a great variety of SIMULINK models.

DEPM and OpenErrorPro. The *Dual-graph Error Propagation Model* (DEPM, [19]) comprises a state-based model of the control- and data-flow, both linked to a stochastic error model. In contrast to (static) fault-tree analysis [17], DEPMs can capture recurrent and hierarchical behaviors. OpenErrorPro is a tool that provides the automated generation from DEPMs to its DTMC semantics in terms of Prism code, enabling the analysis of DEPMs for manifold properties by a great variety of tools. Also due to OpenErrorPro's support of many base-line formalisms that can be translated to DEPMs, they are well-suited for the reliability analysis of safety-critical systems. For instance, Simulink models can be translated to DEPMs using SimPars [20].

Further Methods in Reliability Analysis. Simulation-based approaches are the usual method for reliability analysis. However, as faults are usually *rare events* with a relatively small probability, sufficient confidence in analysis results are difficult to achieve. We are aware of techniques that could circumvent these issues (see, e.g., [16] for an overview) but their implementation would have went far behind the purpose of this paper and such approaches would not allow for a tradeoff analysis in terms of quantiles.

Symbolic techniques for the reliability analysis of safety-critical systems have been first and foremost applied to tackle the state-space explosion system, e.g., in the field of fault-tree analysis (see, e.g., [17]). We are not aware of any application that uses such techniques also to compete with the combinatorial blowup in the number of system configurations such as protection mechanisms we deal with in this paper.

3. From Simulink to DTMC Families

The core of our approach towards a family-based analysis of Simulink models with various protection combinations lies in the generation of Simulink models with redundancy that can be transformed into DTMC families. We illustrate the modeling and the step by step transformation using the Simulink model of a proportional-integral-derivative (PID) controller. A PID is one of the most important and widely used feedback controllers to apply accurate and responsive correction to a control function, essential for many industry areas such as aerospace, process control, manufacturing, and robotics. Figure 1 shows the Simulink PID controller using separate blocks for the P, I, and D terms.

3.1. *Protecting* Simulink *Blocks*

To introduce redundancy mechanisms into Simulink models, we consider syntactic transformation rules that describe how to obtain protected blocks from non-protected ones. Specifically, we consider here the following three redundancy mechanisms:

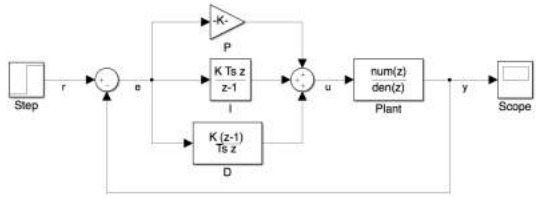


Fig. 1. PID controller is designed Simulink with separate blocks for the P, I, and D terms.

(comparison) The block is duplicated and both outputs are compared. In case their output differs a dedicated failure state is reached. Otherwise, the output is the one of both blocks.

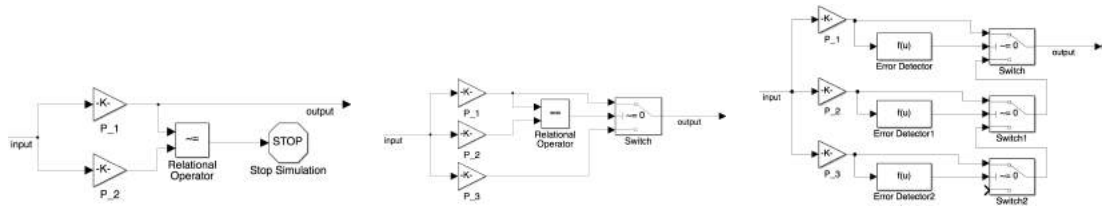
(voting) Following the triple-modular-redundancy principle (TMR), the block is triplicated and the output is based on a majority decision.

(sparing) One block is operational and the remaining two blocks serve as spares. If an error in an active block is detected by a built-in error detection unit, a spare block takes over.

Figure 2 illustrates how to obtain protections from a given block, exemplified by the P term of the PID controller. Note that these patterns are modeled in such a way that that they share most of the behaviors by using multiple times the original block, compare block, and switch block.

3.2. Simulink *Models with Redundancy*

The general workflow of our approach towards a DTMC family and their analysis out from Simulink models is depicted in Figure 3 using the PID example. First, blocks in the base-line model are annotated with the types of protections that should be considered, e.g., comparison, voting, and sparing. In Figure 3, we annotated voting protections for the P and D term in the PID example (indicated by shaded blocks). Then, the annotated base-line model is transformed to a Simulink model that includes redundancies by replacing every block **x** with a switch block **x_s** those purpose is to select the protection mechanism, followed by the redundancy blocks illustrated in the Figure 2 according to the type annotations. The switch depends on a free variable **x_pm** that stands for the kind of protection chosen – the output of the switch connect to the inputs of the selected mechanism **x_pm**. Thus, by setting the variable **x_pm**, e.g., to "no" or "voting", the control flow of the model either follows no protection or the voting protection, respectively. The final Simulink model with redundancy resulting



(a) comparison (b) voting (c) sparing

Fig. 2. Redundancy mechanisms for SIMULINK blocks, illustrated for a P element

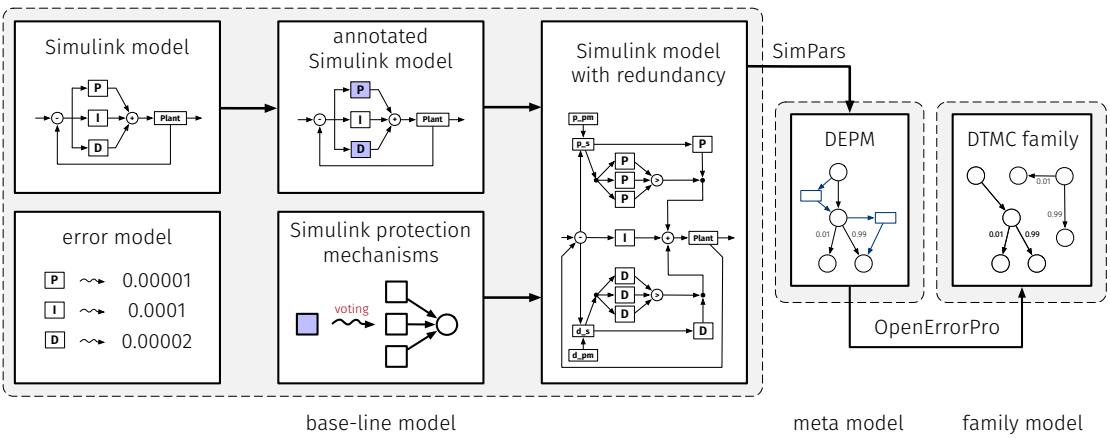


Fig. 3. Schema of the approach, obtaining DTMC families from annotated SIMULINK models

from the PID controller possibly protecting the P and D term with the voting pattern is depicted in the center of Figure 3. Note that this model stands for a family of controllers comprising four family members that can be selected by setting variables **p_pm** and **d_pm** to either "no" or "voting". Hence, a redundancy systems designer does not have to model each protection combination in isolation but only has to specify the syntactic protection rules and annotate blocks for protection – the resulting SIMULINK model with the desired protections can easily be selected through choosing the switch variables.

3.3. *DTMC Families*

Having obtained the SIMULINK model with redundancy that stands for a family of models with different protection combinations, we use SIMPARS to generate a Dual-graph Error Propagation Model (DEPM) [20] preserving the switch variables **x_pm** as data elements. For this, we employ an error model that assigns to each SIMULINK block the probability for some fault occurring in this block. This simple error model could also imagined to involve further SIMULINK blocks or statistical data about faults. In the last step towards DTMC families, PRISM code representing the family of DTMCs that models the control flow and fault propagation is automatically generated using the tool OPENERRORPRO [19]. Also in this step, switch variables are maintained such that also single family members can be extracted from the DTMC family member by choosing protection mechanisms in the switch variables of the PRISM model.

4. Family-based Analysis of Redundancy Systems

The DTMC families generated using the approach we sketched in the last section enable a family-based reliability analysis. We illustrate the benefits of such analyses for the PID example and a large-scale family of a protected aircraft velocity control loop (VCL). Both model a control loop that operates in *rounds*, i.e., starting with initial values of data at the beginning, each round is considered to start when the values of data modified in the last execution of the control loop is fed again as input. For an analysis using the symbolic probabilistic model checker PRISM [15] we then considered the following reliability properties:

(pfail) What is the failure probability in n rounds?
(qround) What is the maximal number of rounds

Table 1. PID analysis results for **(pfail)** with $n = 10$ and **(qround)** with $\theta = 3{\cdot}10^{-4}$

		D:–				D:c				D:v				D:s			
	I	P:–	P:c	P:v	P:s	P:–	P:c	P:v	P:s	P:–	P:c	P:v	P:s	P:–	P:c	P:v	P:s
(pfail) in 10^{-5}	–	6.1	5.2	5.2	6.1	3.5	2.6	2.6	3.5	3.5	2.6	2.6	3.5	6.1	5.2	5.2	6.1
	c	4.4	3.5	3.5	4.4	1.8	0.9	0.9	1.8	1.8	0.9	0.9	1.8	4.4	3.5	3.5	4.4
	v	4.4	3.5	3.5	4.4	1.8	0.9	0.9	1.8	1.8	0.9	0.9	1.8	4.4	3.5	3.5	4.4
	s	6.1	5.2	5.2	6.1	3.5	2.6	2.6	3.5	3.5	2.6	2.6	3.5	6.1	5.2	5.2	6.1
(qround)	–	43	50	50	43	75	100	100	73	75	100	100	75	43	50	50	43
	c	60	75	75	60	150	300	300	151	150	300	300	151	60	75	75	60
	v	60	75	75	60	149	299	299	151	149	299	299	151	60	75	75	60
	s	43	50	50	43	75	100	100	73	75	100	100	75	43	50	50	43

Table 2. Statistics to the analysis experiments

case study	property	parameter	states	all-in-one analysis		one-by-one analysis	
				nodes	time [s]	Σ nodes	time [s]
PID	**(pfail)**	$n = 10$	6 781 782	16 935	12.9	240 946	37.8
	(qround)	$\theta = 0.0003$	”	”	6 693.1	”	2 531.4
VCL	**(pfail)**	$n = 2$	$4.7{\cdot}10^{13}$	1 949 466	17 344.1	$\approx 1.2{\cdot}10^{12}$	$\approx 2.2{\cdot}10^{7}$

in which the system can guarantee a failure probability below some threshold θ?

The first property is a standard reliability property while the second is a *quantile* [5, 6]. We rely on an error model that assigns a fault probability of 10^{-5} to each SIMULINK block. Throughout presenting the results we abbreviate the redundancy mechanisms for SIMULINK blocks as follows: “–” stands for no protection, “c” for comparison, “v” for voting, and “s” for sparing. All experiments were carried out[a] using PRISM supporting variable reordering techniques [14].

4.1. *Analysis of the PID Controller*

For the analysis of the PID controller, we annotate the PID SIMULINK model of Figure 1 with all the protection mechanisms explained in Section 3.1. Applying the three redundancy mechanisms thus yields a SIMULINK model with redundancy that depends on the choice of three switch variables selecting the protection combinations. Hence, after the automated translation of the model using SIMPARS and OPENERRORPRO, we obtain a DTMC family comprising $4^3 = 64$ protection combinations. We then analyzed the **(pfail)** property with a parameter of $n = 10$. The results provided in Table 1 show that the comparison and voting patterns have higher impacts to protect blocks with a slight advantage for voting. This is even more apparent when considering the **(qround)** property, also depicted in Table 1. Here, for guaranteeing a failure probability below $\theta = 3{\cdot}10^{-4}$, a PID that is fully protected by comparison or voting “survives” seven times longer than an unprotected or with sparing protected PID. For obtaining these results, we performed both, a naive one-by-one analysis where every member of the family is analyzed in isolation and an all-in-one analysis. However, due to the small number of family members, the all-in-one analysis has only slight advantages to the one-by-one analysis.

4.2. *The Velocity Control Loop Model*

To illustrate our approach on a large-scale family of SIMULINK protections, we issue a velocity control loop (VCL) of an aircraft model. Our model is a simplified version of the aircraft model borrowed from the SIMULINK example set [1] that itself is based on a long-haul passenger aircraft flying at cruising altitude and speed, adjusting the

[a] Hardware setup: Intel Xeon E5-2680@2.70GHz, 128 GB RAM; Turbo Boost and HT disabled; Debian GNU/Linux 9.1

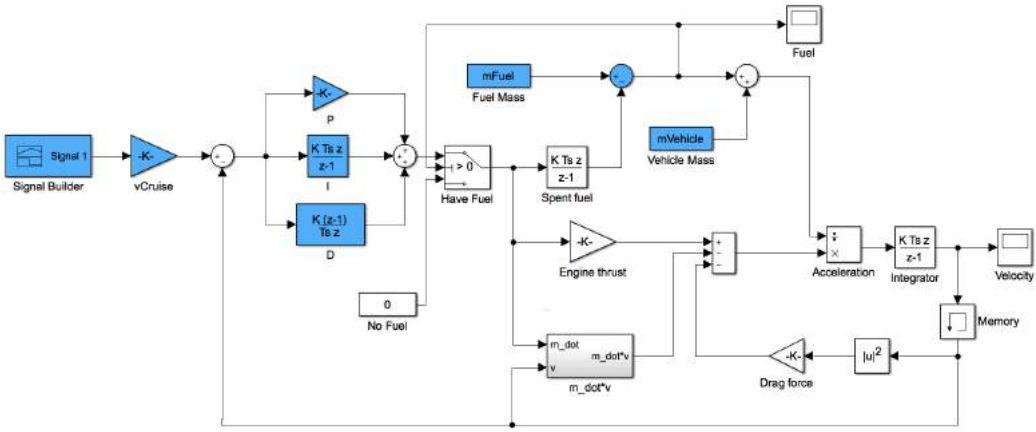


Fig. 4. The SIMULINK aircraft velocity control loop (VCL) model, blocks to be protected highlighted

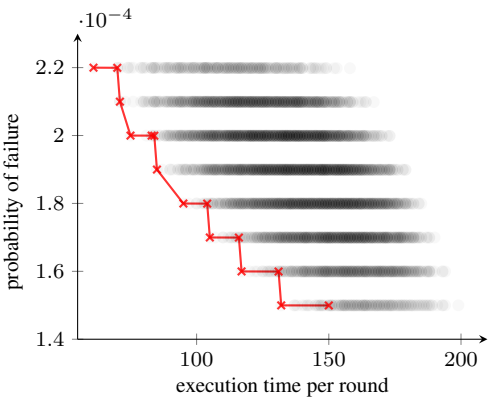


Fig. 5. Configurations Pareto optimal with respect to execution time and probability of failure within two rounds

fuel flow rate to control the aircraft velocity. Figure 4 shows the SIMULINK model where the eight blocks amendable for protection mechanisms are shaded. Note that the VCL also includes the PID controller from the previous example. On each of these blocks we applied the redundancy mechanisms comparison, voting, and sparing as described in Section 3.2, resulting in $4^8 = 65\,536$ combinations of protections. After applying the transformations of Section 3, we first analyzed the generated DTMC family against the **(pfail)** property. Here, we used both, an all-in-one approach performing the analysis on a single family model, and (partially) a one-by-one approach, checking each combination of protections separately. We can already observe from the statistics in Table 2 that the generated DTMC family model requires symbolic techniques to be analyzed by an all-in-one analysis due to the massive size of the state space. To estimate the sizes of models and analysis times for a one-by-one, we considered 655 randomly generated instances of the family (i.e., around 1% of the family members) as analyzing all family members in isolation clearly would have exceeded time constraints. This fact is already underpinned by the size and and analysis times for the randomly generated instances, ranging up to $1.6 \cdot 10^{11}$ states requiring 5 093.922 seconds of analysis time. As even single instances have this magnitude of model sizes and symbolic representations, one can already estimate that our approach exploiting redundancy through symbolic representations is viable. The one-by-one analysis of all 655 randomly generated family members took around 60.4 hours such that we estimate the whole analysis for all 65 536 protection combinations would take more than 250 days of computation time. This demonstrates a speedup of three orders of magnitude an all-in-one analysis yields compared to an exhaustive one-by-one analysis.

4.2.1. *Synthesis of Optimal Tradeoff Protections*

To investigate the tradeoff between execution time and reliability, we measured the impact of protection mechanisms on the execution time of one round [9]. Without any protection, each round required 61 time units to be executed, increased by timings for each protection shown in Table 3. As the all-in-one analysis provided the results for each protection combination in the family model, we hence can easily compute the execution time per round for each combination and relate them to their reliability properties. Figure 5 depicts for each family member the probability of failure within two rounds (property **(pfail)**) and its costs in terms of execution time. The line at the left indicates the Pareto front, which directly yields the optimal protection combinations when either fixing constraints on the probability of failure or execution time. Pareto-optimal configurations are shown in Table 4, where the protection combina-

Table 3. Impact of protections on one-round execution time

block	comparison	voting	sparing
P term	10	15	9
I term	15	23	14
D term	14	22	13
Signal Builder	10	15	9
Fuel Mass	10	15	9
Subtract	12	18	13
Vehicle Mass	10	15	9
vCruise	10	15	9

tions correspond to the chosen protections for the blocks in the order of Table 3, i.e., the combination "- - c - s - - -" stands for protecting the D term block with comparison and the Fuel Mass block with sparing. Similar to the plain PID example, we observe that protection with comparison has great impact on the probability of failure, appearing in most of the Pareto-optimal combinations. Voting does not show up in optimal combinations (although possibly as good as comparison by means of gained fault tolerance) due to its comparably high execution time (see Table 3). Sparing does not reduce the probability of failure as much as comparison and voting, but has good timing characteristics such that it appears at some occasions in the Pareto-optimal combinations.

Table 4. Pareto-optimal protection combinations

combination	exec. time	prob. of failure
c c c s c c c s	150	$1.4995{\cdot}10^{-4}$
c c c - c c c -	132	$1.4997{\cdot}10^{-4}$
c c c - s c c -	131	$1.5994{\cdot}10^{-4}$
c - c - c c c -	117	$1.5997{\cdot}10^{-4}$
c - c - s c c -	116	$1.6994{\cdot}10^{-4}$
c - c - c - c -	105	$1.6997{\cdot}10^{-4}$
c - c - s - c -	104	$1.7994{\cdot}10^{-4}$
- - c - c - c -	95	$1.7997{\cdot}10^{-4}$
- - c - c - - -	85	$1.8997{\cdot}10^{-4}$
- - c - s - - -	84	$1.9994{\cdot}10^{-4}$
- - - - c c - -	83	$1.9997{\cdot}10^{-4}$
- - c - - - - -	75	$1.9998{\cdot}10^{-4}$
- - - - c - - -	71	$2.0997{\cdot}10^{-4}$
- - - - s - - -	70	$2.1994{\cdot}10^{-4}$
- - - - - - - -	61	$2.1998{\cdot}10^{-4}$

Remarks on Limitations. Whereas impossible for the quantile property, the standard reliability property asking for the probability of failure within a fixed number of rounds could also be evaluated using simulation-based approaches. However, due to the relatively small probability of faults in each P, I, or D term, we were not able to perform statistical model checking with sufficient confidence in either the PID and VCL model. We also performed analyses for the quantile property **(qround)**, but due to the size of the models already single instances of the DTMC family could not be analyzed within a week of computation such that we dropped an exhaustive study of this property for the VCL model.

4.2.2. *Further Techniques Applied*

As usual, automatically generated models for PRISM do not admit a good variable ordering for their concise symbolic representation via MTBDDs and thus require post-processing steps to be amendable for a formal analysis (see, e.g., [8, 14]). Within our tool chain, things were actually worse as even single instances of the OPENERRORPRO generated DTMCs could not be built without either running out of memory or taking several days before interrupting the building process. To enable an analysis of our models, we had to apply the following post-processing steps on the generated VCL model.

Reset Value Optimization. Thanks to the control- and data-flow models in the DEPM meta model, we used standard graph algorithms applied for each data element to determine those control-flow locations where the data is never read before it is written again. For the minimal control-flow locations (with respect to the control-flow order) we reinitialized the value of the data storage to the value it has at the initial control-flow location. The ratio behind this optimization is that the state-space is reduced by joining naively bisimilar states where data values do not have impact on the future behaviors.

Iterative Variable Reordering. We exploited further the sensitivity of MTBDD represented models to its variable ordering. For this, we successively built subfamily models, iteratively adding protection mechanisms to each block and performing variable reordering [14] to determine a suitable variable order.

5. Discussion and Further Work

We proposed to use family-based approaches for the modeling and analysis of redundancy models to overcome limitations imposed by the combinatorial blowup that arises when protecting system components. To illustrate the approach, we pre-

sented a tool chain that enables reliability analysis of SIMULINK models with redundancy. In future, we aim at further automating the handcrafted steps in this tool chain, e.g., incorporating the optimizations done in Section 4.2.2 into OPENERRORPRO.

Note that although presented in the specific setting for SIMULINK designs, our approach is applicable to many redundancy system models. In particular, the tool chain we present enables to use many base-line models supported by the tool used for the automated translation towards DTMC families, e.g., by OPENERRORPRO [19], and the full analysis power of properties that can be checked for DTMCs using, e.g., by symbolic probabilistic model checkers such as PRISM [15].

Acknowledgments. This work is supported by the DFG through the Collaborative Research Centers CRC 912 (HAEC) and TRR 248 (see https://perspicuous-computing.science, project ID 389792660), the Cluster of Excellence EXC 2050/1 (CeTI, project ID 390696704, as part of Germany's Excellence Strategy), the Research Training Groups QuantLA (GRK 1763) and RoSI (GRK 1907), projects JA-1559/5-1, BA-1679/11-1, BA-1679/12-1, and the 5G Lab Germany.

References

1. Verify model using simulink control design and simulink verification blocks (accessed 19/02/2019). https://mathworks.com/help/slcontrol/ug/model-verification-using-simulink-control-design-and-simulink-verification-blocks-.html.
2. R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic decision diagrams and their applications. *FMSD*, 10(2/3):171–206, 1997.
3. C. Baier and C. Dubslaff. From verification to synthesis under cost-utility constraints. *ACM SIGLOG News*, 5(4):26–46, 2018.
4. C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
5. C. Baier, C. Dubslaff, S. Klüppelholz, M. Daum, J. Klein, S. Märcker, and S. Wunderlich. Probabilistic model checking and non-standard multi-objective reasoning. In *Proc. of FASE'14*, volume 8411 of *LNCS*, pages 1–16. Springer, 2014.
6. C. Baier, C. Dubslaff, S. Klüppelholz, and L. Leuschner. Energy-utility analysis for resilient systems using probabilistic model checking. In *Proc. of PN'14*, volume 8489 of *LNCS*, pages 20–39, 2014.
7. A. Beer, T. Georgiev, F. Leitner-Fischer, and S. Leue. Model-based quantitative safety analysis of matlab simulink / stateflow models. In *MBEES*, 2013.
8. P. Chrszon, C. Dubslaff, S. Klüppelholz, and C. Baier. Family-based modeling and analysis for probabilistic systems - featuring ProFeat. In *Proc. of FASE'16*, volume 9633 of *LNCS*, pages 287–304. Springer, 2016.
9. K. Ding, A. Morozov, and K. Janschek. MORE: MOdel-based REdundancy for Simulink. In *Proc. of SAFECOMP'18*. Springer, 2018.
10. C. Dubslaff, C. Baier, and S. Klüppelholz. Probabilistic model checking for feature-oriented systems. *TAOSD*, 12:180–220, 2015.
11. M. Fujita, P. C. McGeer, and J. C. Yang. Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. *FMSD*, 10(2/3):149–169, 1997.
12. A. Haber, C. Kolassa, P. Manhart, P. M. S. Nazari, B. Rumpe, and I. Schaefer. First-class variability modeling in matlab/simulink. In *Proc. of VaMoS'13*, New York, NY, USA, 2013. ACM.
13. A. Joshi and M. P. E. Heimdahl. Model-based safety analysis of simulink models using scade design verifier. In *Proc. of SAFECOMP'05*, LNCS, pages 122–135. Springer, 2005.
14. J. Klein, C. Baier, P. Chrszon, M. Daum, C. Dubslaff, S. Klüppelholz, S. Märcker, and D. Müller. Advances in symbolic probabilistic model checking with PRISM. In *Proc. of TACAS'16*, volume 9636 of *LNCS*, pages 349–366. Springer, 2016.
15. M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. of CAV'11*, volume 6806 of *LNCS*, pages 585–591, 2011.
16. A. Legay, S. Sedwards, and L. Traonouez. Rare events for statistical model checking an overview. In *Proc. of RP'16*, pages 23–35, 2016.
17. C. Mattarei. *Scalable Safety and Reliability Analysis via Symbolic Model Checking: Theory and Applications*. PhD thesis, ICT International Doctoral School, Trento, 2016.
18. K. L. McMillan. *Symbolic Model Checking*. Kluwer, 1993.
19. A. Morozov, R. Tuk, and K. Janschek. ErrorPro: Software tool for stochastic error propagation analysis. In *Proc. of REES'15*, pages 59–60, 2015.
20. A. Morozov, K. Janschek, T. Krüger, and A. Schiele. Stochastic error propagation analysis of model-driven space robotic software implemented in simulink. In *Proc. of MORSE'16*, pages 24–31. ACM, 2016.
21. R. Svenningsson, J. Vinter, H. Eriksson, and M. Törngren. Modifi: a model-implemented fault injection tool. In *Proc. of SAFECOMP'10*, pages 210–222. Springer, 2010.
22. T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake. A classification and survey of analysis strategies for software product lines. *ACM Comput. Surv.*, 47:1–45, 2014.