Path-based calculation of MTTFF, MTTFR and asymptotic unavailability with the stochastic process algebra tool CASPA *

Johann Schuster and Markus Siegle

Universität der Bundeswehr München

Abstract

CASPA is a stochastic process algebra tool for performance and dependability modelling, analysis and verification. It is based entirely on the symbolic data structure MTBDD (Multi-Terminal Binary Decision Diagram) which enables the tool to handle models with very large state space. This paper describes an extension of CASPA's solving engine for path-based approximation of MTTFF (Mean Time To First Failure), MTTFR (Mean Time To First Recovery) and asymptotic unavailability by MTBDD algorithms. To keep the paper self-contained, we briefly recall a symbolic variant of the k-shortest-path algorithm of Azevedo et al., which works in conjunction with a symbolic variant of Dijkstra's shortest path algorithm. A non-trivial case study illustrates the use of this kind of path-based analysis and compares the path-based unavailability calculation to the results from standard Markovian analysis.

*This work is based on the paper *Symbolic calculation of k-shortest paths and related measures with the stochastic process algebra tool CASPA* [6]

1 Introduction

In order to better understand the behaviour of fault-tolerant systems, it is often very helpful to obtain information about the possible sequences of events which may lead to system failure. Therefore, in addition to standard steady-state and transient measures, path-based analysis is an important ingredient to model-based dependability analysis (see for example [4]). Knowing the most likely paths from an initial, error-free state to the set of states where the system has failed, and knowing the associated probabilities and average durations of such paths, is very valuable for system designers and may assist them in debugging and improving their fault-tolerant system.

In this paper, we discuss algorithms for the path-based analysis as implemented in the modelling tool CASPA [10, 7, 2], which is a tool for performance and dependability modelling, based on a stochastic process algebra. CASPA is completely based on symbolic, i.e. Multi-Terminal Binary Decision Diagram (MTBDD)-based, techniques that enable the tool to generate and analyse Markov chains with very large state spaces in a highly efficient manner. The tool has shown to be very suitable for dependability evaluation purposes, for example it is also used as a back-end solver for the OpenSESAME modelling tool [9]. Here we describe a new analysis feature of CASPA that uses path-based calculations. In this paper we use the term "k-shortest path" synonymous to "k-most probable path". We briefly recall the symbolic algorithm for the calculation of k-shortest paths presented in [6] and show how it can be used to calculate also MTTFF (Mean Time To First Failure), MTTFR (Mean Time To First Recovery) and asymptotic unavailability. As the terms are not defined uniquely in the literature, we emphasise that we use the term MTTFF (for repairable systems) to denote the average time to the first failure starting from the completely repaired system (which is the initial state for our model). In this paper, MTTFF will be used to approximate the MUT (Mean Up Time). We would like to note that [4] uses the term MTTF to describe the mean time to

first failure. Our calculation of MTTFR, the mean time from the first failure to (first) recovery, will be used to approximate the MDT (Mean Down Time).

The paper is organised as follows: Section 2 recalls the algorithms presented in [6] for calculating the shortest (i.e. most probable) path in a given transition system. In Section 3 we show how to approximate the MTTFF, MTTFR and asymptotic unavailability on the basis of k-shortest-path calculations. Section 4 presents a non-trivial application case study that illustrates the efficiency of the implemented symbolic data structures and algorithms, and Section 5 concludes the paper.

2 Preliminaries

This section briefly summarises the algorithms given in [6] for calculating the k-shortest paths. The basis of the algorithms is a set of labels L, a set of states S and a labelled transition system $Trans$ defined as follows:

$$Trans \subseteq S \times L \times [0,1] \times S,$$

We only allow "parallel" transitions with different labels, i.e. if $(x,a,p,y) \in Trans$ and $(x,a',p',y) \in Trans$, then $a \neq a'$ must hold. The real number in the interval $[0,1]$ is the probability of taking the corresponding transition, so the auxiliary condition

$$\forall x \in S : \sum_{(x,a,p,y) \in Trans} p = 1.$$

must hold. As an abbreviation we will use $x \stackrel{a,p}{\rightarrow} y$ for the tuple (x,a,p,y). Fig. 1(a) shows an example of such a transition system. In addition, we define another transition system $Trans_{max}$ out of $Trans$ and call it the *maximal*

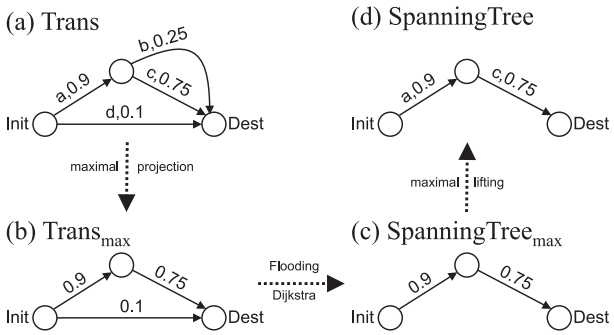


Figure 1: maximal projection, Flooding Dijkstra and maximal lifting

projection.

$$\begin{aligned} Trans_{max} \quad := \quad & \{(x,p,y) \in S \times [0,1] \times S| \\ & (\exists a \in L : (x,a,p,y) \in Trans) \wedge \\ & (\forall a \in L, p' \in [0,1] : ((x,a,p',y) \in Trans) \Rightarrow (p' \leq p))\}. \end{aligned}$$

The existence condition ensures that a lifting to $Trans$ exists and the second condition ensures that p is maximal. So these are the maximum transition probabilities one can get by choosing an arbitrary action from a source to a target state. Again, we will use $x \xrightarrow{p} y$ as a synonym for the tuple (x,p,y), cf. Fig. 1(b) as the maximal projection of the example.

Another definition is needed to read the action labels from the path: The maximal lifting of a subset $T \subseteq Trans_{max}$ to $Trans$:

$$T^* \quad := \quad \{(x,a,p,y) \in Trans | \exists (x,p,y) \in T\}.$$

In general, this lifting is not unique: There can be more than one action fulfilling the maximality condition while p as the maximum is unique for a certain pair $(x,y) \in S^2$. An example for the maximal lifting can be seen in the arrow from Fig. 1(c) to (d).

All algorithms in [6] are given by set-theoretic operations in order to allow for efficient MTBDD-implementations (which are given in the appendix of [6]).

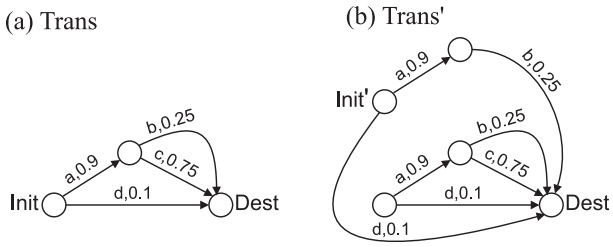


Figure 2: Trans and Trans'

2.1 Spanning tree algorithm

Using the maximal projection of $Trans$, the action labels are abstracted away and one can proceed with a variant of Dijkstra's algorithm to obtain a spanning tree. In our case we are not interested in the shortest path but in the maximum probability, which is an equivalent problem formulation. Our Dijkstra-Variant is a multi-update approach that allows for an implementation by MTBDD operations. We call it Flooding-Dijkstra (cf. Sec. 2 in [6]). The Flooding Dijkstra algorithm is sketched in by the arrow from Fig. 1(b) to (c).

2.2 Reading the action labels

Once the spanning tree is known, it has to be lifted to a subset of $Trans$ and a certain path (there might be more than one with the maximum probability) has to be chosen (cf. Sec. 3.1 in [6]). In the example, the shortest path could be read from Fig. 1(d).

2.3 Second shortest path

Starting with $Trans$ and the shortest path a new transition system $Trans'$ can be constructed with the ideas of [1, 8]. The desired property of $Trans'$ is that a shortest path in $Trans'$ (from Init' to Dest) is a second-shortest path in $Trans$ (cf. Sec. 3.2 in [6]). An example is shown in Fig. 2. With this algorithms at hand, the k-shortest path can be calculated iteratively for an arbitrary but fixed k.

3 MTTFF and MTTFR

In this section we need to consider sojourn times of states, therefore the purely probabilistic setup of the previous sections does not suffice. We thus define two sets of transitions, which are directly generated by CASPA from the input language. Let $Trans_M$ be the subset of Markovian transitions

$$Trans_M \subseteq S \times L \times \mathbb{R}_{>0} \times S,$$

where similarly to Sec. 2 S is a set of states and L a set of labels. The real number in the tuple describes the Markovian transition by means of the parameter λ (transition rate) of the corresponding exponentially distributed random variable. Analogously, we define the set of immediate transitions by $Trans_I$ as follows:

$$Trans_I \subseteq S \times L \times \mathbb{R}_{>0} \times S.$$

The real number in this case is interpreted as a weight which is normalised to a probability at the end of the model generation process [2]. By the maximum progress assumption, Markovian transitions that compete with immediate transitions are removed from $Trans_M$. Using the transition probabilities of the immediate transitions and calculating the transition probabilities of the Markovian transitions (from the specified transition rates), a transition system $Trans$ like the one defined in Sec. 2 can be obtained easily.

For the following subsections we also need the set of failure states of the system, which will be denoted by $Failure$. For the calculation of the time corresponding to a path, the notation of the cumulative outgoing rate of a state $x \in S$ is important:

$$\lambda(x) := \sum_{x \xrightarrow{a,\lambda} y \in Trans_M} \lambda$$

Using the cumulative outgoing rate, an associated average time can be calculated for every path in the Transition system $Trans_M \cup Trans_I$ starting at a certain

state x_0. We define:

$$getTime(x_0, path) := \sum_{x \in S_M(path)} \frac{1}{\lambda(x)}$$

Here $S_M(path)$ ranges over all source states in $Trans_M \cap path$ and x_0 is the first state of $path$. Note that x_0 is redundant as path contains all the information we need, but it makes calculations in Sec. 3.2 easier to read.

3.1 Calculation of MTTFF

An approximation of the MTTFF (cf. definition in Sec. 1) is calculated as follows [4]:

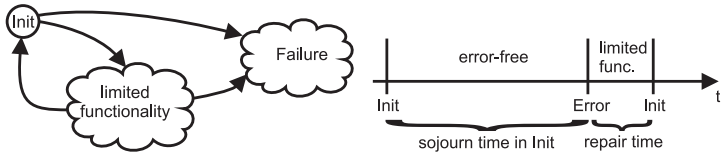
$$MTTFF \approx \frac{1}{\alpha \cdot \lambda(Init)} \tag{1}$$

Being in the state $Init$, α is the probability of visiting at least one state of the set $Failure$ before returning to Init. Clearly, α can be approximated by adding up the probabilities of the k-shortest paths from $Init$ to the set $Failure$.

This approximation in Eq. 1 can be motivated as seen in Fig. 3. Starting from the state $Init$, we are interested in the mean time necessary to reach an Error before returning to the initial state as seen in Fig. 3(a). An exemplary scenario is shown in Fig. 3(b), where a non-critical error occurs and the system can recover to the error-free state Init. As in this approximation we assume fast repairs, the repair time is negligible in contrast to the sojourn time $T := \frac{1}{\lambda(Init)}$ in state Init, so the time for a loop from Init to Init is taken to be T. Therefore one can approximate

$$MTTFF \approx T \cdot \alpha + 2 \cdot T \cdot (1-\alpha) \cdot \alpha + \ldots = \alpha \cdot T \cdot \sum_{i=0}^{\infty} (i+1) \cdot (1-\alpha)^i$$

and from this one can deduce Eq. (1) by a geometric series argument. Note that if the system does not have fast repairs, an alternative approach similar to the calculation of MTTFR in Sec. 3.2 can be taken for the calculation of MTTFF.



(a) Basic setup (b) Timeline

Figure 3: Approximation of MTTFF

3.2 Calculation of MTTFR

As the calculation of MTTFF exploits the fact that the system has fast repairs, a similar approximation cannot be applied for the calculation of the MTTFR. Therefore, we apply the following method, as originally proposed in [4]: For a state $x \in S$ and a subset of target states $T \subseteq S$ we use $PATH(x, T)$ to denote minimal paths from x to T. By minimal we mean in this case that only the last state of the path is in the set T. The probability of a certain path starting from state x is denoted by $P(x, path)$ (this is a by-product of the Flooding Dijkstra algorithm, cf. Sec. 2). Let the function $Target$ return the final state of a path. With this notation we can define the MTTFR for paths starting in state x as

$$MTTFR(x) := \sum_{path \in PATH(x, S \setminus Failure)} P(x, path) \cdot getTime(x, path).$$

We also need the cumulated probability of the explored paths

$$P_{expl} := \sum_{path \in PATH(Init, Failure)} P(Init, path).$$

The MTTFR is then approximated as the weighted sum

$$MTTFR := \frac{1}{P_{expl}} \cdot \sum_{path \in PATH(Init, Failure)} P(Init, path) \cdot MTTFR(Target(path)).$$

From this, the asymptotic unavailability can be calculated as follows:

$$\bar{A}(\infty) = \frac{MDT}{MUT + MDT} \approx \frac{MTTFR}{MTTFF + MTTFR}$$

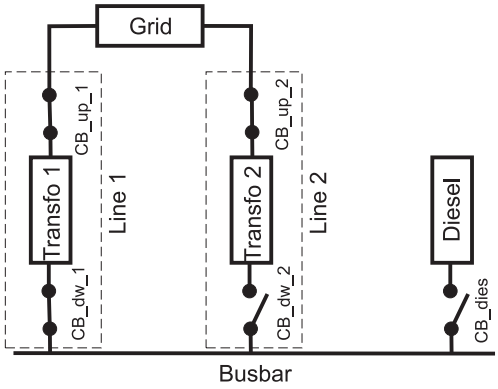


Figure 4: Sketch of the electrical system

Note that in this approximation we also assume that the system has fast repairs (then it holds that $MUT \approx MTTFF$, $MDT \approx MTTFR$). In systems with fast repair the fully repaired system has by far the highest probability (e.g. 0.996 in the example in Sec. 4).

3.3 Approximations in CASPA

In CASPA only trivial truncation criteria are implemented so far: Firstly, we can specify how many paths should be calculated (and therefore be used in the calculations of Sec. 3.1 and 3.2) and secondly, we cancel a path if its probability is below a certain threshold. As stated in [4] we want to stress that we only can expect good approximation results if we are given a model with fast repair, i.e. the repair rates are orders of magnitude bigger than the failure rates.

4 Case study

To show the applicability of our algorithm we modelled the electrical system given in [3]. However, we noticed that the most probable paths published there did not fit the textual description (e.g. an initial failure of transfo 2 would not be possible if it was a cold spare). This is why we now give an alternative description of the model that to our best knowledge produces the paths given in [3].

4.1 Description of the model

The model is shown in Fig. 4. The aim of the system is to provide the busbar with electrical energy. Each of the two main lines consists of upper and lower circuit breakers (CB) and a transformer (transfo). They route electrical energy from the grid to the busbar. If the lines fail or the grid does, the diesel generator has to be used. The initial configuration is as seen in Fig. 4 where only CB_dw_2 and CB_dies are in the open position, the other switches are closed. The following constraints for the operation and dynamic behaviour are given:

- States of the components can be WORKING, STANDBY or FAILED.
- Either line 1, line 2 or the diesel engine is used. As long as the grid works, mode switches can only be line 1 $\leftrightarrow$ line 2 $\leftrightarrow$ diesel, no direct switches from line 1 to diesel and vice versa are allowed.
- The transformers and the grid are hot spares and always fail with the same rate, no matter if they are active (i.e. in the WORKING state) or not.
- The circuit breakers CB_up and CB_dw are cold spares (i.e. they do not fail as long as no current runs over them) and they can produce on-demand-failures (i.e. errors that occur when trying to switch a circuit breaker). The switching may succeed or fail according to a on-demand-failure probability.
- CB_dies may only fail on-demand, it does not fail internally.
- When a transformer fails, its upper circuit breaker has to be opened, otherwise a short-circuit will make the grid unavailable for the other transfo.
- Switching from transfo 1 to transfo 2 means trying to open CB_up_1 and to close CB_dw_2. Note that when transfo 1 fails and CB_up_1 fails to open, there is always an attempt to close CB_dw_2 even if it's clear that in this situation the diesel engine has to be used anyway, as the grid is short-circuited by line 1.
- Switching on the diesel engine means closing CB_dies and trying to start the engine (RS: request to start). Both operations may have on-demand failures.
- Switching back after a repair always works without on-demand failures.
- Whenever an error (either Markovian or on-demand) occurs, a repairman

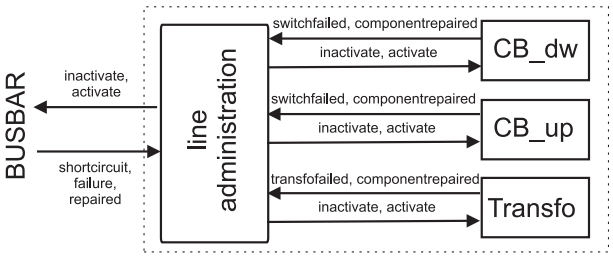


Figure 5: Sketch of the model of a transformer line

(with exponentially distributed repair time) starts the repair.

- For the repairs we use infinite server semantics (i.e. there are infinitely many stochastically independent repairmen).

We have introduced a total ordering of all switching actions as follows: (CB_up_1 > CB_dw_1 > CB_up_2 > CB_dw_2 > CB_dies > RS_dies). Whenever there are multiple switching operations to be done in our model, switching operations will be performed in the order defined by this relation. Following the philosophy of shortest paths to failure states, we stress that whenever a greater switching action fails, we do not explore the path any further (except for the short circuit situation given above). All intended switchings succeed with probability $\frac{999}{1000}$, all failure rates are equal to 10^{-4} per hour and all repair rates equal to 10^{-1} per hour.

4.2 CASPA implementation of the model

The model has been built in CASPA using a compositional modelling approach with synchronisations. Each line is the parallel composition of two switches, a transformer and a line administration. Its synchronising actions are given in Fig. 5. The line administration process has in addition to its internal state a counter variable that keeps track of how many components of the line are currently in the FAILED state. From this it can be determined when the entire line has been repaired. A top-level process (not shown in Fig. 5) synchronises with two line subprocesses and takes care of the grid and the entire diesel line. As the immediate actions used only for synchronising are of no interest for the

resulting paths, they are eliminated, once the overall model has been generated. For the path-based analysis, this partial elimination is done by five symbolic elimination rounds [2] and the resulting model has 3604 reachable states (none of them absorbing). For the steady-state analysis all immediate transitions can be eliminated in 14 symbolic elimination rounds and the resulting ergodic Markov model has 1136 reachable states. Note that the model in [6] used only one single failure state as we did not consider repair events, so there were only 774 states for the path-based analysis.

4.3 Experimental results

All experiments have been carried out on an Intel Xeon 3.06 GHz machine with 2 GB of main memory running SUSE Linux version 9.1.

Nr.	Path	numerical result	theoretical result	mean time (h)
1	Failure_OF_GRID Occurrence_OF_RC_CB_dies	$2.0e-04$	$\frac{1}{5}\frac{1}{1000}$	2.000e+03
2	Failure_OF_GRID OK_OF_RC_CB_dies Occurrence_OF_RS_dies	1.998e-04	$\frac{1}{5}\frac{999}{1000}\frac{1}{1000}$	2.000e+03
3	Failure_OF_GRID OK_OF_RC_CB_dies OK_OF_RS_dies Failure_OF_dies_generator	1.990e-04	$\frac{1}{5}\left(\frac{999}{1000}\right)^2\frac{10^{-4}}{10^{-1}+3\cdot10^{-4}}$	2.010e+03
4	Failure_OF_Transfo2 Occurrence_OF_RO_CB_up_2 Occurrence_OF_RC_CB_dies	2.000e-07	$\frac{1}{5}\left(\frac{1}{1000}\right)^2$	2.000e+03
5	Failure_OF_CB_dw_1 Occurrence_OF_RC_CB_dw_2 Occurrence_OF_RC_CB_dies			
6	Failure_OF_CB_up_1 Occurrence_OF_RC_CB_dw_2 Occurrence_OF_RC_CB_dies			

Table 1: Start of the list of most probable paths

Table 1 shows the 6 most probable paths calculated by our algorithm (more paths can be found in [6]). The first column is the ordering of the paths given by CASPA, the second one the corresponding sequence of actions. The third column shows the numerical result provided by our algorithm, the fourth one shows the exact result calculated by hand. The last column shows the mean time the corresponding path takes. In column 2, the prefix `Failure_` always

means a failure caused by an exponential distribution. The prefix `OK_` means that an on-demand failure did not occur, while prefix `Occurrence_` means that an on-demand failure did occur. The following abbreviations are used: RO (Request to Open), RC (Request to Close) and RS (Request to Start). For the by-hand calculation one only has to take care which failure event(s) and repair event(s) can occur for a certain state. For example, the most probable path is calculated as follows: In the initial configuration, no component has to be repaired and Transfo1, CB_up_1, CB_dw_1, Transfo2 or Grid can fail. Therefore $P\left(\begin{array}{l}\texttt{Failure\_OF\_GRID} \rightarrow \\ \texttt{Occurrence\_OF\_RC\_CB\_dies}\end{array}\right) = \frac{10^{-4}}{5 \cdot 10^{-4}} \cdot \frac{1}{1000} = \frac{1}{5} \cdot \frac{1}{1000}$.
Due to the fact that CASPA uses the CUDD library [5] with a default accuracy Cudd_Epsilon of $1.0 \cdot 10^{-12}$, this is the maximum accuracy one can expect from the results. For example the tool calculates

$$P\left(\begin{array}{l}\texttt{Failure\_OF\_Transfo2} \rightarrow \\ \texttt{OK\_OF\_RO\_CB\_up\_2} \rightarrow \\ \texttt{Failure\_OF\_GRID} \rightarrow \\ \texttt{Occurrence\_OF\_RC\_CB\_dies}\end{array}\right) = P\left(\begin{array}{l}\texttt{Failure\_OF\_Transfo2} \rightarrow \\ \texttt{Occurrence\_OF\_RO\_CB\_up\_2} \rightarrow \\ \texttt{OK\_OF\_RC\_CB\_dies} \rightarrow \\ \texttt{OK\_OF\_RS\_dies} \rightarrow \\ \texttt{Failure\_OF\_dies\_generator}\end{array}\right),$$

but the exact values differ:

$$\frac{1}{5} \cdot \frac{999}{1000} \cdot \frac{10^{-4}}{10^{-1}+4 \cdot 10^{-4}} \cdot \frac{1}{1000} \neq \frac{1}{5} \cdot \frac{1}{1000} \left(\frac{999}{1000}\right)^2 \cdot \frac{10^{-4}}{10^{-1}+3 \cdot 10^{-4}}$$

As the difference is below $1.0 \cdot 10^{-12}$, the values are taken as equal. Reducing Cudd_Epsilon would improve accuracy but also slow down the calculations. We have calculated the asymptotic unavailability using the following characterisation of a non-critical state:

$$\big(\big((\texttt{line 1 and } \overline{\texttt{sc line 2}}) \texttt{ or } (\texttt{line 2 and } \overline{\texttt{sc line 1}})\big) \texttt{ and grid }\big) \texttt{ or diesel}.$$

The first part of the term is the condition that line 1 is working and no short-circuit at line 2 is present (and so on). For the measure of a failure state, the term was negated and converted into disjunctive normal form (DNF) to fit the CASPA syntax of a measure.

4.3.1 Path-based measures and discussion

Table 2 shows a comparison of the results. Column one shows the results of the algorithms described in this paper (using the two most probable paths into every

	MTBDD-path-based	results in [3]	MTBDD-steady-state
MTTFF (h)	$3.2870 \cdot 10^6$	$3.29 \cdot 10^6$	-
MTTFR (h)	4.9792	4.95	-
Unavailability	$1.5148 \cdot 10^{-6}$	$1.51 \cdot 10^{-6}$	$1.5153 \cdot 10^{-6}$
Solution time (s)	92.22	n.a.	3.5

Table 2: Experimental results

failure state and for every failure state the two most probable paths out of it), column two contains the results given in [3] and the last column shows the result by steady-state analysis. The results of [3] and our results are quite similar.

To verify our path-based algorithms we compared the result for the asymptotic unavailability to the result of steady-state analysis, which fits very well. With the Gauss Seidel algorithm the result was calculated within only 19 iterations in 0.04 seconds. The rest of the 3.5 seconds given in Tab. 2 was spent for the model generation, reachability analysis and so on. This showed that being only interested in the asymptotic unavailability, it is much faster using standard numerical methods and calculating the state-measure.

5 Conclusion

In this paper, we first recalled k-most probable path calculations using variants of Dijkstra's and Azevedo et al. algorithm (c.f. [6]). From this starting point we introduced approximations for MTTFF and MTTFR and asymptotic unavailability that are suitable for symbolic implementations. A non-trivial case study showed the applicability of the approach, but it also revealed that if only the asymptotic unavailability is to be calculated, standard numerical methods are faster than the path-based ones (which require expensive MTBDD operations). The precision of the algorithms has been shown to be dependent on the accuracy of the calculations in the underlying MTBDD package CUDD. With this path-based analysis, the CASPA tool provides an alternative analysis method to the standard numerical analysis. The path-based analysis can further nicely be used for debugging models: If a certain behaviour of a system is known from the specification, the paths in the model can be checked against it.

Acknowledgements: Special thanks to Max Walter for his fruitful comments on the MTTFF/MTTFR calculations. Also we would like to thank the reviewers for their helpful questions and suggestions. Further we would like to thank Deutsche Forschungsgemeinschaft (DFG) who supported this work under grant SI 710/7-1 and for partial support by DFG/NWO Bilateral Research Programme ROCKS.

References

[1] J. Azevedo, J. Madeira, E. Martins, and F. Pires. A Shortest Paths Ranking Algorithm. In *Proc. of the Annual Conference AIRO'90 Operational Research Society of Italy*, pages 1001–1011, IEEE, 1990.

[2] J. Bachmann, M. Riedl, J. Schuster, and M. Siegle. An Efficient Symbolic Elimination Algorithm for the Stochastic Process Algebra tool CASPA. In *SOFSEM 2009: Theory and Practice of Computer Science*, pages 485–496, Špindlerův Mlýn, Czech Republic, 2009. Springer, LNCS 5404.

[3] M. Bouissou and J.-L. Bon. A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes. *Reliability Engineering and System Safety*, 82:149–163, 2003.

[4] M. Bouissou and Y. Lefebvre. A Path-Based Algorithm to Evaluate Asymptotic Unavailability for Large Markov Models. *Proc. of the Annual Reliability and Maintainability Symposium*, pages 32–39, 2002.

[5] CUDD website. `http://vlsi.colorado.edu/~fabio/CUDD/`, (last checked August 2010).

[6] M. Günther, J. Schuster, and M. Siegle. Symbolic calculation of k-shortest paths and related measures with the stochastic process algebra tool CASPA. In *Proc. of the First Workshop on DYnamic Aspects in DEpendability Models for Fault-Tolerant Systems*, pages 13–18, 2010.

[7] M. Kuntz, M. Siegle, and E. Werner. Symbolic Performance and Dependability Evaluation with the Tool CASPA. In *Europ. Perf. Engineering Workshop*, pages 293–307. LNCS 3236, 2004.

[8] W. Schmid. *Berechnung kürzester Wege in Straßennetzen mit Wegeverboten*. PhD thesis, Universität Stuttgart, Fakultät für Bauingenieur- und Vermessungswesen, 2000.

[9] M. Walter, M. Siegle, and A. Bode. OpenSESAME: The Simple but Extensive, Structured Availability Modeling Environment. *Reliability Engineering and System Safety*, 93(6):857–873, 2007.

[10] E. Werner. Leistungsbewertung mit Multi-terminalen Binären Entscheidungsdiagrammen. Master's thesis, Univ. Erlangen, Computer Science 7 (in German), 2003.