Dipartimento di Informatica
Università del Piemonte Orientale "A. Avogadro"
Spalto Marengo 33, 15100 Alessandria
http://www.di.unipmn.it

TECHNICAL REPORT
TR-INF-2004-03-04-UNIPMN
(March 2004)

Dynamic Bayesian Networks for Modeling Advanced Fault Tree Features in Dependability Analysis

Author: S. Montani, L. Portinale, A. Bobbio
({stefania,portinal,bobbio}@unipmn.it)

Dynamic Bayesian Networks for Modeling Advanced Fault Tree Features in Dependability Analysis

Stefania Montani, Luigi Portinale*, Andrea Bobbio
Dipartimento di Informatica
Universitá del Piemonte Orientale "A. Avogadro"
Alessandria (ITALY)
e-mail: {stefania,portinal,bobbio}@unipmn.it

March 22, 2004

Abstract

Fault Trees *(FT)* are one of the most popular techniques for dependability analysis of large, safety critical systems. It has been shown [1] that *FT* can be directly mapped into Bayesian Networks (*BN*) and that the basic inference techniques on the latter may be used to obtain classical parameters computed from the former. In this paper, we show how *BN* can provide a unified framework in which also *Dynamic FT* (*DFT*), a recent extensions able to treat complex types of dependencies, can be represented. In particular, we propose to characterize dynamic gates within the *Dynamic Bayesian Network* framework (*DBN*), by translating all the basic dynamic gates into the corresponding *DBN* model. The approach has been tested on a complex example taken from the literature. Our experimental results testify how *DBN* can be safely resorted to if a quantitative analysis of the system is required. Moreover, they are able to enhance both the modeling and the analysis capabilities of classical *FT* approaches, by representing local dependencies and by performing general inference on the resulting model.

1 Introduction

Fault Trees (*FT*) are one of the most popular techniques for dependability analysis of large, safety critical systems. They allow one to represent the combination of elementary causes that lead to the occurrence of an undesired catastrophic event named the *Top Event* (*TE*). By specifying failure probabilities on the basic components of the modeled system (the elementary causes of the *TE*, also called *basic events*), then the whole system unreliability (probability of the *TE*) at a given mission time can be computed.

* Corresponding author

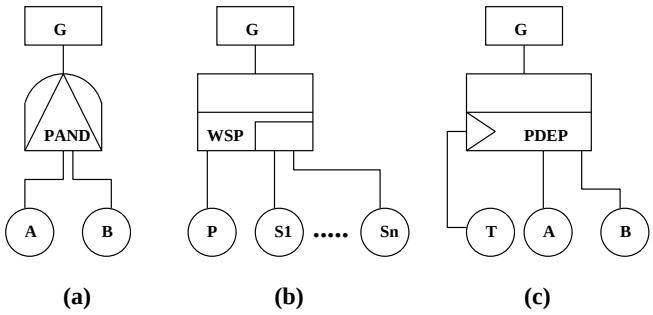


Figure 1: Dynamic Gates of a *DFT*

In recent years, an effort has been documented in the literature, aimed at increasing the modeling power of *FT* by including new primitive gates, able to accommodate complex kinds dependencies. This augmented *FT* language is referred to by the authors as *dynamic FT* [5, 6, 10]. Dynamic Fault Trees (*DFT*) introduce four basic (dynamic) gates: the warm spare (*WSP*), the sequence enforcing (*SEQ*), the probabilistic dependency (*PDEP*) and the priority AND (*PAND*).

WSP are dynamic gates able to model one or more input principal components that can be substituted by one or more backups (spares), which cover the same functionality (figure 1(b)). The *WSP* gate fails when the number of operational powered spares and/or principal components is less than the minimum required. In particular spares can fail even while they are dormant, but the failure rate of an unpowered spare is lower than the failure rate of the corresponding powered one. More precisely, being λ the failure rate of a powered spare, the failure rate of the unpowered spare is $\alpha\lambda$, with $0 \leq alpha \leq 1$ called the *dormancy factor*. Spares are more properly called "hot" if $\alpha = 1$ and "cold" if $\alpha = 0$.

A *SEQ* gate forces its inputs to fail in a particular order: when a *SEQ* is found in a *DFT*, it never happens that the failure sequence takes place in different orders. *SEQ* gates can be modeled as a special case of a cold spare [10], so they will not be considered in the following[1].

In the *PDEP* gate (figure 1(c)), one trigger event T causes other dependent components to become unusable or inaccessible with probability $p_{dep} \leq 1$. In particular, when the trigger event occurs, the dependent components are all *immediately* forced to fail (with the specified probability). The separate failure of a dependent component, on the other hand, has no effect on the trigger event. *PDEP* has also a non-dependent output, that simply reflects the status of the trigger event.

Finally, the *PAND* gate (figure 1(a)) reaches a failure state iff all of its input components have failed in a preassigned order (from left to right in graphical notation). While the *SEQ* gate allows the events to occur only in a preassigned order and states that a different failure sequence can never take place, the *PAND* does not model such a strong assumption: it simply detects the failure order and fails just in one case (in figure 1(a) a

[1] The conceptual difference between the two kind of gates is that the inputs to a *SEQ* do not need to be a component and its set of spares, but can be components covering any kind of function in the *FT*.

failure occurs iff A fails before B, but B may fail before A without producing a failure in G).

DFT are typically solved by automatic conversion to the equivalent Markov model [5]. Through a process known as modularization [8, 2, 7] it is possible to identify the independent subtrees with dynamic gates and use a different Markov model (much smaller than the model corresponding to the entire *FT*) for each of them. Nevertheless, there still exists the problem of state explosion, since the set of states can be significantly large.

In order to overcome these limitations, we propose to characterize dynamic gates within the Dynamic Bayesian Networks (*DBN*) framework. Static *BN* have been shown suitable to model and analyze systems in place of standard *FT* [15, 1], so we aim at exploiting *BN* features also when dinamic gates behavior has to be modeled. The paper provides a translation of dynamic gates into *DBN* by comparing the approach to standard Markov chain representation of the gates. We also present the reliability analysis of a real-world system (a cardiac assist device presented in [13]); results demonstrate how *DBN* can be safely exploited for quantitative analysis, as well as for enhancing modeling and analysis of the given system.

2 *DFT* gates and *DBN* modeling

The standard *DBN* representation model adopts a discrete time approach, where several time slices are explicited, togheter with information about transitions from a time slice to the next ones. In particular, each time slice contains a set of (time-indexed) random variables, some of which are typically not observable [4, 14, 12].

When the Markov assumption holds (and in particular when we are dealing with a first order Markov process) the future slice at time $t+1$ is conditionally independent of the past ones given the present slice at time t [9]. In this case, it is sufficient to represent two consecutive time slices and the network is fully specified if it is provided with: (1) the prior probabilities for root variables at time $t = 0$; (2) the intra-slice conditional dependency model, together with the corresponding conditional probability tables; (3) the inter-slice conditional dependency model and probability tables (i.e. the transition model), which explicit the temporal probabilistic dependencies between variables. In particular, a variable at time $t+1$ may depend not only on its "historical" copy (i.e. on the same variable at time t), but also on the values of other variables in the previous time slice.

The above model of a *DBN* is usually called 2TBN (two time-slice Temporal Bayesian Network)[12, 3]. In the following, we will consider a particular representation for 2TBNs, called the *canonical form* [3]. Let X be a net variable and X_t its copy at time t: the *canonical set* of net variables is the set
$\{X : X_t \in \cup_k Parents(X^k{}_{t+1})\}$; i.e., a canonical variable is a variable having childrens at the next time slice. A 2TBN is in canonical form if only the canonical variables are represented at time t.

The motivation for adopting a *DBN* instead of the corresponding Markov chain (as

it is done in classical approaches to dependability) is that, by decomposing the state of a complex system in its constituent variables, the network is able to take advantage of sparsness in the temporal probability model [12]. As a matter of fact, the conditional independence assumptions enables a compact representation of the probabilistic model, allowing the system designer or analyst to avoid the complexity of specifying and using a global-state model (like a standard Markov Chain) when the dynamic module of the considered *FT* is significantly large.

However, we must pay attention to the fact that, by modeling a dynamic *FT* module with a *DBN* we are considering the factorized representation of a *Discrete Time Markov Chain* (*DTMC*). This differs from the model adopted in the reliability analysis of a *DFT* which is usually a *Continuous Time Markov Chain* (*CTMC*). The results provided by a *CTMC* are in fact slightly different. As a matter of fact, the two models are not exactly equivalent, since in a *CTMC* transitions occur in a continuous fashion. As a consequence, while in a discrete time model we can consider the event given by two components failing at the same time, the same is no longer true in a continuous time model. This could force us to make some assumptions as in the case of the *PAND* gate [10]; in particular, if we observe the contemporary failure of two inputs, having lost the sequence of failure information, we do not know if the *PAND* is forced to fail or not, and we have to make some hypothesis. Usually a contemporary fault implies a fault in the gate.

In the next subsections, we detail how dynamic gates introduced above can be converted into a *DBN*. Section 3 applies our models to an example, taken from the literature, that shows a *DFT* including different gate types.

2.1 Warm spare gate

In a *DFT*, different configurations of warm spares can be designed. As an example, let us consider a situation where a single component A can be substituted by a spare $S1$. If also $S1$ fails, it can be substituted by a second spare $S2$ (which may be identical to $S1$). If every component (either principal or spare) is failed, the gate produces a fault.

The *DBN* corresponding to this gate is shown in figure 2, together with the corresponding Markov chain. Component failure rates are shown as λ-values, while state names are triples $X_1X_2X_3$ corresponding to $A, S1, S2$ respectively and with $X_i = 0$ if the component is working, $X_i = 1$ otherwise; e.g., 001 means A and $S1$ working, $S2$ failed. As for every *DBN* we will consider in the following, the net of figure 2 is a 2TBN in canonical form. It can be observed that each component node at time $t+1$ depends on its copy at time t (we consider persistence of faults). Moreover, $S1$ depends on A because, if A was working at time t, then $S1$'s failure rate at time $t+1$ is still $\alpha\lambda$; on the other hand, if A has failed, $S1$'s failure rate becomes λ. The situation of $S2$ is analogous, but it depends on both the principal component and on the first spare (since it will be powered only when both nodes have failed). The *WSP* gate is modeled as a deterministic AND node among its three inputs A, $S1$ and $S2$.

We consider a constant failure rate for components and an exponentially distributed failure time; the probability that component C (with failure rate λ_C) fails in Δt time

instances is then $1 - e^{-\lambda_C \Delta t}$. The probabilities of failure in the network are assigned as below[2], where we consider a discretization step of $\Delta t = 1$ (in all not reported cases, the probability of failure is 0);

$$Pr\{A(t+1) = 1 | A(t) = 1\} = 1$$

$$Pr\{A(t+1) = 1 | A(t) = 0\} = 1 - e^{-\lambda_A}$$

$$Pr\{S1(t+1) = 1 | A(t) = 0, S1(t) = 0\} = 1 - e^{-\alpha\lambda_{S1}}$$

$$Pr\{S1(t+1) = 1 | A(t) = 1, S1(t) = 0\} = 1 - e^{-\lambda_{S1}}$$

$$Pr\{S1(t+1) = 1 | S1(t) = 1\} = 1$$

$$Pr\{S2(t+1) = 1 | S1(t) = 0, S2(t) = 0, A(t) = 0\} = 1 - e^{-\alpha\lambda_{S2}}$$

$$Pr\{S2(t+1) = 1 | S1(t) = 1, S2(t) = 0, A(t) = 0\} = 1 - e^{-\alpha\lambda_{S2}}$$

$$Pr\{S2(t+1) = 1 | S1(t) = 0, S2(t) = 0, A(t) = 1\} = 1 - e^{-\alpha\lambda_{S2}}$$

$$Pr\{S2(t+1) = 1 | S1(t) = 1, S2(t) = 0, A(t) = 1\} = 1 - e^{-\lambda_{S2}}$$

$$Pr\{S2(t+1) = 1 | S2(t) = 1\} = 1$$

Hot and cold spares can be modeled analogously, by setting α to 1 or to 0 respectively.

As an extension to classical *DFT* approaches, resorting to *DBN* we can also model the presence of repair mechanisms in the system. In particular, if a component has failed at time t, if repair is allowed it will recover from failure with probability $r \leq 1$.

More complex configurations can be taken into account as well. For example, suppose repair is required only after the overall *WSP* gate failure. In this case, we can model the *WSP* as in figure 3. Note that A (as well as $S1$ ans $S2$) at time $t+1$ depends on *WSP* at time t. Actually, if *WSP* (and therefore also all the input components) have failed at time t, the repair of A at time $t+1$ will be tented with success probability r_A; i.e., $Pr\{A(t+1) = 1 | A(t) = 1\} = 1 - r_A$. Similar considerations hold for $S1$ and $S2$.

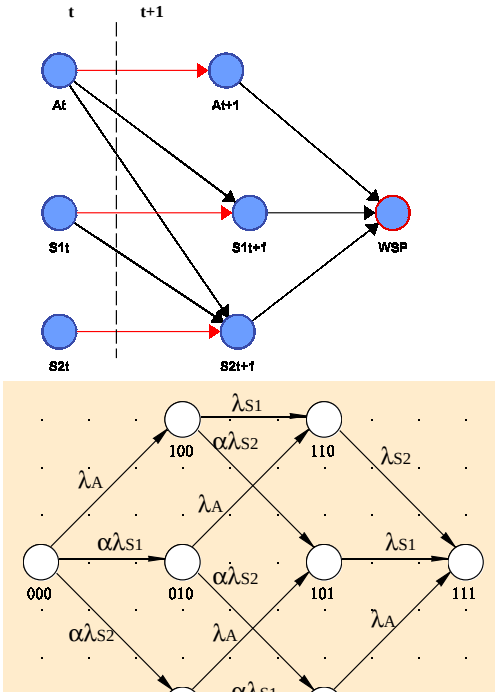
2.2 Probabilistic dependency gate

Since the trigger event of a *PDEP* gate determines an *immediate* failure (with probability p_{dep}) of its dependent components, a subsystem including a *PDEP* can be completely characterized resorting to intra-slice (i.e. static) conditional dependencies. Nevertheless, exploting a dynamic network allows us to resort to a common framework for dynamic gates representation.

Figure 4 shows the *DBN* and the corresponding Markov chain for a *PDEP* gate in a configuration in which the trigger event T (with failure rate λ_T) has two dependent components A and B (with failure rate λ_A and λ_B respectively). States of the Markov chain follow the notational convention illustrated for figure 2 for the triple T, A, B.

As usual, each component at time $t+1$ depends on the component itself at time t. Moreover, the dependent components will fail with probability p_{dep} if the trigger has failed

[2]For the sake of brevity, we just report basic probabilistic information on which to build actual CPT for the net.



Fig

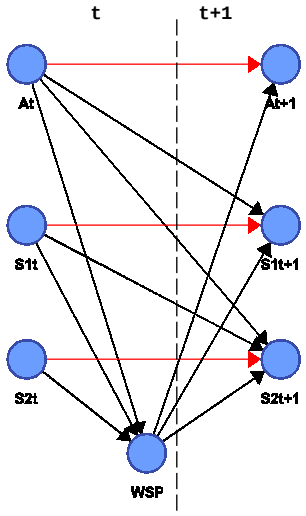


Figure 3: *DBN* for the *WSP* gate in the example when a repair mechanism is allowed

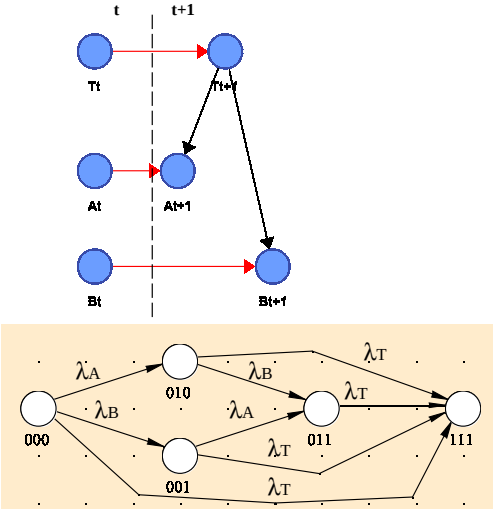


Figure 4: *DBN* and *CTMC* for the *PDEP* gate in the example

in the same time slice. The probabilities of failure of A can be summarized as follows (again in all not reported cases the probability of failure is 0 and a discretization step $\Delta t = 1$ is assumed):

$$Pr\{T(t+1) = 1 | T(t) = 1\} = 1$$

$$Pr\{T(t+1) = 1 | T(t) = 0\} = 1 - e^{-\lambda_T}$$

$$Pr\{A(t+1) = 1 | A(t) = 1\} = 1$$

$$Pr\{A(t+1) = 1 | A(t) = 0, T(t+1) = 0\} = 1 - e^{-\lambda_A}$$

$$Pr\{A(t+1) = 1 | A(t) = 0, T(t+1) = 1\} = p_{dep}$$

Similarly to A we can set failure probabilities for the dependent component B (using λ_B). The *PDEP* gate simply mirrors the trigger status and is not reported in the network.

A largely exploited special case of *PDEP* is the functional dependency gate (*FDEP*); in the latter the trigger event leads to a definite failure of dependent components. This can obviously be modeled as discussed in this section, by just setting $p_{dep} = 1$.

2.3 Priority AND gate

PAND gates model situations where a control component may prevent the system to crash (with ruinous consequences) because of the failure of a standard component. In such cases, a failure of the control component before the failure of the standard one prevents the recovery action of the control component, leading to a (sub)-system failure. Consider the gate of figure 1(a): we can model the failure sequence by considering two different failure modes for component B, mode `ff` (failed by first) and mode `fs` (failed by second). We can then decide if B fails with mode `ff` or with mode `fs` by considering transition of component A from time t to $t+1$. Figure 5 shows the resulting *DBN* and

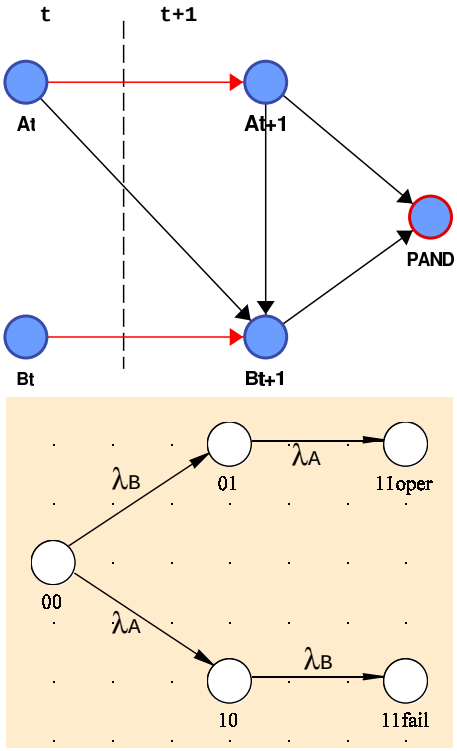


Figure 5: *DBN* and *CTMC* for *PAND* gate

the corresponding *CTMC*; Nodes B_t and B_{t+1} (modeling component B) have 3 different values namely 0=`working`, 1=`ff`, 2=`fs`. Node PAND is a deterministic node with the following functional rules

$PAND = 1$ `if A(t+1)=1 and B(t+1)=2`

$PAND = 0$ `otherwise`

Conditional probabilities for the faulty values of the component are set as below (as usual not reported probabilities are assumed to be 0):

$$Pr\{A(t+1) = 1 | A(t) = 1\} = 1$$

$$Pr\{A(t+1) = 1 | A(t) = 0\} = 1 - e^{-\lambda_A}$$

$$Pr\{B(t+1) = 1 | A(t) = 0, A(t+1) = 0, B(t) = 0\} = 1 - e^{-\lambda_B}$$

$$Pr\{B(t+1) = 2 | A(t) = 0, A(t+1) = 1, B(t) = 0\} = 1 - e^{-\lambda_B}$$

$$Pr\{B(t+1) = 2 | A(t) = 1, A(t+1) = 1, B(t) = 0\} = 1 - e^{-\lambda_B}$$

$$Pr\{B(t+1) = 1 | B(t) = 1\} = 1$$

$$Pr\{B(t+1) = 2 | B(t) = 2\} = 1$$

We can notice that, given that we assume persistence of faults, the parent configuration $\{A(t) = 1, A(t+1) = 0, B(t) = 0\}$ is not possible; this means that we can put any value in the CPT of $B(t+1)$ in correspondence to the above configuration. It is also worth noting

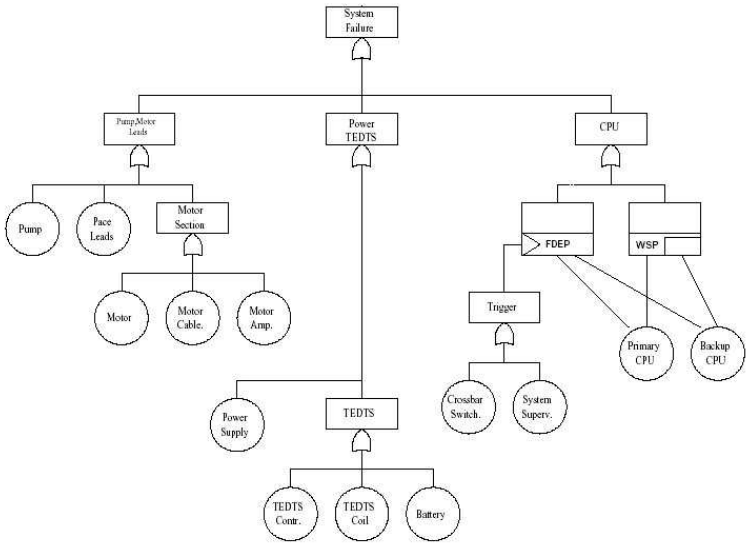


Figure 6: The Fault Tree for the cardiac assist device [13]

the choice made for parent configuration $\{A(t) = 0, A(t+1) = 1, B(t) = 0\}$ corresponds to assume that a contemporary fault of both A and B will result in a fault of the whole gate. If the opposite would have been modeled, then the given row had to be substituted with the following one

$$Pr\{B(t+1) = 1 | A(t) = 0, A(t+1) = 1, B(t) = 0\} = 1 - e^{-\lambda_B}$$

modeling the fact that if both A and B fail at the next time instant, then a failure mode not leading to a global fault (e.g. `ff`) will be set for B.

3 Dependability analysis of a cardiac assist device

In [13], a real-world example of the use of *DFT* is presented, by considering the reliability modeling and analysis of a cardiac assist device. The system consists of an external part and of a patient-implanted one. The external part includes a TEDTS (Transcutaneous Energy and Data Transmission Systems), able to transmit power and information to the implanted portion. The internal device is governed by a primary CPU, provided with a (warm) spare. It also includes an electronic supervisor, a crossbar switch, a mechanical blood pump and a motor. Some pace leads are attached to the heart and connected to the electronic supervisor for monitoring. The failure of either the crossbar switch or of the system supervisor fails both the primary and backup CPU.

Figure 6 (taken from [13]) shows the corresponding *FT*. At the highest level, the system can be divided into three subtrees, whose root nodes will be referred to as Pump-Motor-Leads, Power-TEDTS and CPU.

Component	Subtree	Failure Rate
Motor	Pump-Motor-Leads	0.00002
Motor Cable	Pump-Motor-Leads	0.000014
Motor Amplifier	Pump-Motor-Leads	0.000028
Pump	Pump-Motor-Leads	0.000009
Pace Leads	Pump-Motor-Leads	0.00001
TEDTS Controller	Power-TEDTS	0.00004
TEDTS Coil	Power-TEDTS	0.000018
Battery	Power-TEDTS	0.000008
Power Supply	Power-TEDTS	0.00009
Crossbar Switch	CPU	0.00002
System Supervisor	CPU	0.00009
Primary CPU	CPU	0.00006
Backup CPU	CPU	0.00006

Table 1: Failure Rates for the Cardiac Assist Device

In particular, the CPU subtree includes two dynamic gates; i.e. a *FDEP* gate with two dependent components, that act as the principal and the spare components of a *WSP* gate. The trigger event of the *FDEP* is the logical *OR* of the failure of the crossbar switch and of the system supervisor; the dependent components are the two CPUs. We have set the dormancy factor for the backup CPU to $\alpha = 0.5$.

Table 1 lists the values of the failure rates we have applied in the example. To test the correcteness of our modeling proposal, we have first solved the system relying to classical methodologies; in particular, a Markov chain was used to model the dynamic subtree CPU as shown in figure 7. Figure 7 also shows the corresponding *DBN*. Notice that the *DBN* is obtained by composing the subnets corresponding to the dynamic gates of the subtree (as well as the subnet for the logical *OR* of the root of the CPU subtree). CPTs of nodes belonging to different subtrees are obtained by exploiting the semantics of each gate; for example the presence of the trigger will determine a failure on both CPUs, independently from other events. This differs from the way how the *CTMC* is generated, since in that case a simulation of the possible state transitions, starting from the initial state where all the components of the subtree are working, is needed [10].

We have performed standard reliability analysis on the whole *DFT* using the SHARPE tool[3] [16]. The probability of the *TE* from 100 to 1000 hours has been computed and it is reported in Table 2, column 2.

We have then converted the *DFT* into the corresponding *DBN* along the lines of previous section for the dynamic part and as illustrated in some previous works for the static part (see [15, 1]). The whole network is shown in figure 8.

[3]Actually, SHARPE does not allow the direct analysis of a *DFT*, so we have appended the results computed by the tool for the *CTMC* of fig. 7, to the standard (static) *FT* resulting from the original one, by removing the dynamic module CPU.

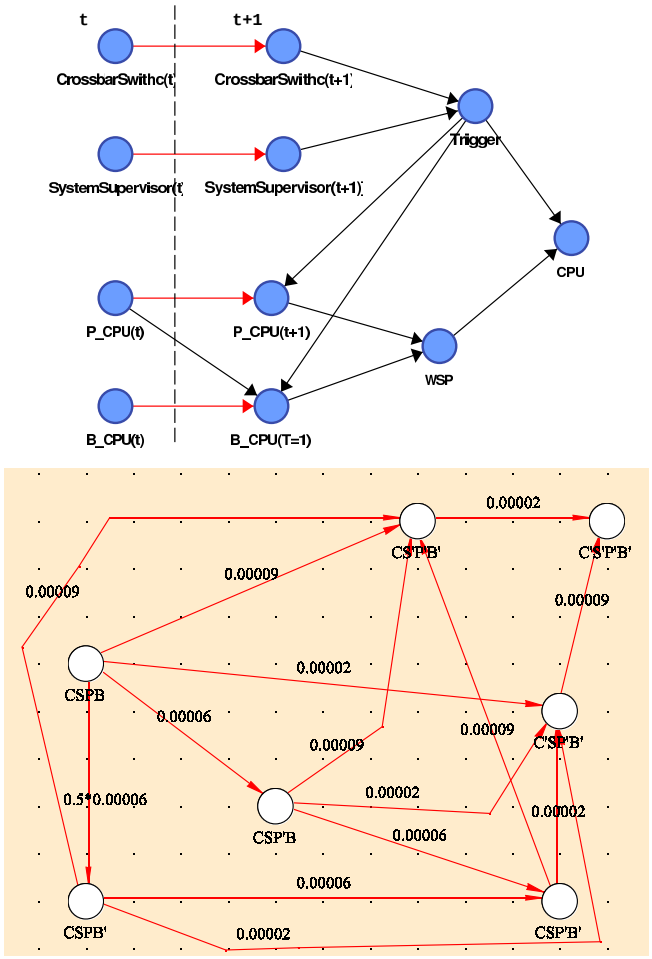


Figure 7: *DBN* and *CTMC* for subtree CPU. In the *CTMC* C=CrossbarSwitch, S=Supervisor, P=Primary CPU, B=Backup CPU; X means that componente X is working, while X' means that component X is failed.

Time (hours)	FTA	Dynamic *BN*
100	0.03413101	0.03442142
200	0.06714543	0.06770761
300	0.09907735	0.09989352
400	0.12995997	0.13101314
500	0.15982557	0.16109957
600	0.18870556	0.19018496
700	0.21663050	0.21830059
800	0.24363008	0.24547687
900	0.26973319	0.27174334
1000	0.29496789	0.29712873

Table 2: *TE* computation using standard *DFT* analysis (SHARPE) and *DBN* inference (BK algorithm).

Computation of the *TE* probability is a simple matter of standard inference on the *DBN*; in particular it corresponds to a *prediction* inference task over a changing horizon (the time points at which system reliability is required)[12]; this is in turn a special case of *filtering* (or *monitoring*) given that we have empty evidence for every analysis point. We have performed the analysis by resorting to the *BK algorithm* for approximate monitoring [3]; the approach is based on a factored decomposition into independent subprocesses (identified with "clusters" of nodes) and it performs exact inference if clusters are suitably arranged. In our case the only non-trivial cluster is the one comprising nodes for components $CrossbarSwitch$, $Supervisor$, $Primary$ and $BackupCPU$; every other cluster is just a singleton containing only one component node. Computation has been performed by relying on the implementation of Kevin Murphy's BN ToolBox for MATLAB [11]; the probabilities of the *TE* from 100 to 1000 hours are shown in Table 2, column 3. We have also performed some experiments using the BAYESIALAB software package (`www.bayesia.com`), by exploiting stochastic simulations with results comparable to those shown in table 2[4].

As it can be observed the results are basically identical to the ones provided by classical solution methodologies. Differences can be interpreted as an effect of discretization (see section 1); experiments we have performed on several other examples showed that results obtained by *DBN* inference are equivalent to those obtained by solving the corresponding *DTMC*. This allows us to conclude that *DBN* can be safely resorted to, if a quantitative analysis of the system is required. Moreover, they have the advantage of providing a unified framework, in which both static and dynamic components can be analysed.

[4]Notice that the nets reported in the figures of this paper are screenshots taken from BAYESIALAB.

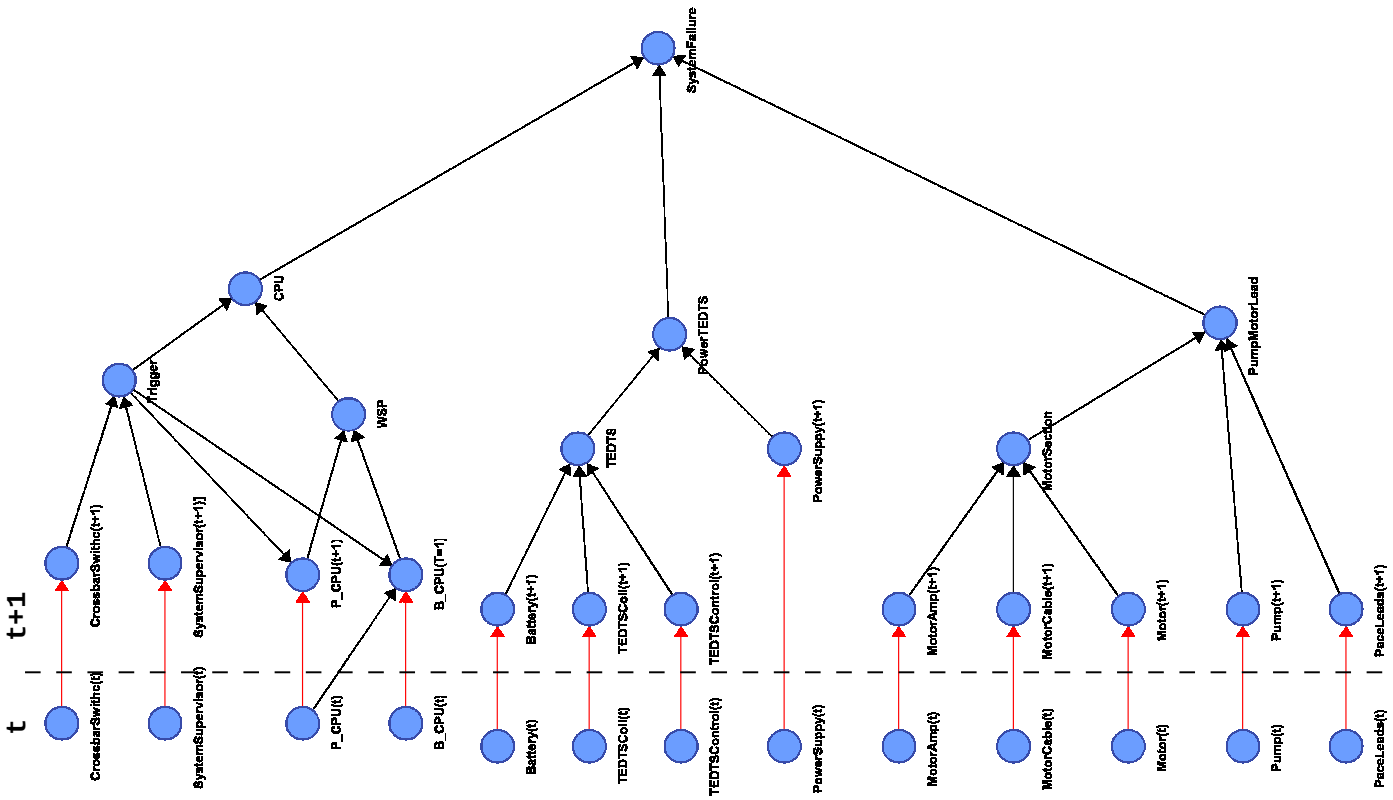


Figure 8: The dynamic Bayesian Network for the Fault Tree in the example

4 Discussion and conclusion

Bayesian Networks provide a robust probabilistic method for reasoning under uncertainty and are becoming widely used in several real world applications. In previous works, we have analyzed the possibility of translating *FT*, a very popular technique for hardware dependability analysis, into the framework of *BN*.

In this paper, we have examined how to translate also special purpose *FT* gates, called dynamic gates, introduced to represent complex dependencies in the *FT*.

FT with dynamic gates are typically solved by conversion to the equivalent Markov model. Through modularization it is possible to identify the independent sub-trees with dynamic gates and to use a different Markov model for each of them. Nevertheless, there still exists the problem of state explosion.

This difficulty can be overcome by modeling the *DFT* as a *DBN*, thus taking advantage of the sparsness in the temporal probability model. The modeling methodology we propose has been quantitatively tested on some examples, one of which has been presented in this paper: the results obtained using the *DBN* were basically identical to the ones obtained using other analysis techniques described in the literature; the only source of approximation was due to the need for discretization.

We can also notice that, resorting to a *BN* formalism (both static or dynamic) has the

well-known advanteges of exploiting all the modeling capabilities of graphical probabilistic models: multi-valued variables (instead of binary events as in *FT*), local dependencies among components (instead of classical s-independence assumption in *FT*), noisy interaction among component behavior (instead of deterministic interaction as in *FT*). In addition, general inference mechanism (combining prediction as well as diagnosis) can be naturally performed on a *BN*, while they are not easily implemented in standard *FT* analysis, especially if evidence is gathered during analysis. These aspects have already been noticed in our previous works concentrating on static *BN* [15, 1], but naturally apply to the *DBN* framework discussed here as well.

In conclusion, we believe that *DBN* represent a quite general framework through which *DFT* can be properly characterized and that could be relied upon in dependability applications.

References

[1] A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla. Improving the analysis of dependable systems by mapping fault trees into bayesian networks. *Reliability Engineering and System Safety*, 71:249–260, 2001.

[2] A. Bobbio and D. Codetta Raiteri. Parametric fault-trees with dynamic gates and repair boxes. In *Proceedings Reliability and Maintainability Symposium RAMS2004*, 2004.

[3] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proceedings UAI'88*, pages 33–42, 1998.

[4] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[5] J. Bechta Dugan, S.J. Bavuso, and M.A. Boyd. Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41:363–377, 1992.

[6] J. Bechta Dugan, S.J. Bavuso, and M.A. Boyd. Fault-trees and Markov models for reliability analysis of fault-tolerant digital systems. *Reliability Engineering and System Safety*, 39:291–307, 1993.

[7] J. Bechta Dugan, K.J. Sullivan, and D. Coppit. Developing a low-cost high-quality software tool for dynamic fault-tree analysis. *IEEE Transactions on Reliability*, 49(1):49–59, 2000.

[8] Y. Dutuit and A. Rauzy. A linear-time algorithm to find modules of fault tree. *IEEE Transactions on Reliability*, 45:422–425, 1996.

[9] U. Kjaerulff. dhugin: a computational system for dynamic time-sliced bayesian networks. *International Journal of Forecasting*, 11:89–101, 1995.

[10] R. Manian, D.W. Coppit, K.J. Sullivan, and J.B. Dugan. Bridging the gap between systems and dynamic fault tree models. In *Proceedings IEEE Annual Reliability and Maintainability Symposium*, pages 105–111, 1999.

[11] K. Murphy. The bayes net toolbox for matlab. *Computing Science and Statistics*, 33, 2001.

[12] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD Thesis, UC Berkley, 2002.

[13] Y. Ou and J. B. Dugan. Sensitivity analysis of modular dynamic fault trees. In *Proceedings International Computer Performance and Dependability Symposium*, pages 35–45, Chicago, 2000. IEEE Computer Society Press.

[14] P.Dagum, A. Galper, and E. Horwitz. Dynamic network models for forecasting. In *Proc. UAI'92*, pages 41–48, 1992.

[15] L. Portinale and A. Bobbio. Bayesian networks for dependability analysis: an application to digital control reliability. In *15-th Conference Uncertainty in Artificial Intelligence, UAI-99*, pages 551–558, 1999.

[16] R. Sahner and K.S. Trivedi. Reliability modeling using SHARPE. *IEEE Transactions on Reliability*, R-36:186–193, 1987.