Dipartimento di Informatica
Università del Piemonte Orientale "A. Avogadro"
Via Bellini 25/G, 15100 Alessandria
http://www.di.unipmn.it

DBNet, a tool to convert Dynamic Fault Trees into Dynamic Bayesian Networks

Author: Stefania Montani, Luigi Portinale,
Andrea Bobbio, Daniele Codetta-Raiteri
({stefania, portinal, bobbio, raiteri}@unipmn.it),
Marco Varesio
(mvaresio@gmail.com)

TECHNICAL REPORT TR-INF-2005-08-02-UNIPMN
(August 2005)

The University of Piemonte Orientale Department of Computer Science Research Technical Reports are available via WWW at URL
`http://www.di.mfn.unipmn.it/`.
Plain-text abstracts organized by year are available in the directory

Recent Titles from the TR-INF-UNIPMN Technical Report Series

2005-01 *Bayesan Networks in Reliability*, Langseth, H., Portinale, L., April 2005.

2004-08 *Modelling a Secure Agent with Team Automata*, Egidi, L., Petrocchi, M., July 2004.

2004-07 *Making CORBA fault-tolerant*, Codetta Raiteri D., April 2004.

2004-06 *Orthogonal operators for user-defined symbolic periodicities*, Egidi, L., Terenziani, P., April 2004.

2004-05 *RHENE: A Case Retrieval System for Hemodialysis Cases with Dynamically Monitored Parameters*, Montani, S., Portinale, L., Bellazzi, R., Leonardi, G., March 2004.

2004-04 *Dynamic Bayesian Networks for Modeling Advanced Fault Tree Features in Dependability Analysis*, Montani, S., Portinale, L., Bobbio, A., March 2004.

2004-03 *Two space saving tricks for linear time LCP computation*, Manzini, G., February 2004.

2004-01 *Grid Scheduling and Economic Models*, Canonico, M., January 2004.

2003-08 *Multi-modal Diagnosis Combining Case-Based and Model Based Reasoning: a Formal and Experimental Analisys*, Portinale, L., Torasso, P., Magro, D., December 2003.

2003-07 *Fault Tolerance in Grid Environment*, Canonico, M., December 2003.

2003-06 *Development of a Dynamic Fault Tree Solver based on Coloured Petri Nets and graphically interfaced with DrawNET*, Codetta Raiteri, D., October 2003.

2003-05 *Interactive Video Streaming Applications over IP Networks: An Adaptive Approach*, Furini, M., Roccetti, M., July 2003.

2003-04 *Audio-Text Synchronization inside mp3 file: A new approach and its implementation*, Furini, M., Alboresi, L., July 2003.

2003-03 *A simple and fast DNA compressor*, Manzini, G., Rastero, M., April 2003.

2003-02 Engineering a Lightweight Suffix Array Construction Algorithm, Manzini, G., Ferragina, P., February 2003.

2003-01 *Ad Hoc Networks: A Protocol for Supporting QoS Applications*, Donatiello, L., Furini, M., January 2003.

Contents

Abstract

The unreliability evaluation of a system including dependencies involving the state of components or the failure events, can be performed by modelling the system as a Dynamic Fault Tree (DFT). The combinatorial technique used to solve standard Fault Trees is not suitable for the analysis of a DFT. The conversion into a Dynamic Bayesian Network (DBN) is a way to analyze a DFT. This paper presents a software tool allowing the automatic analysis of a DFT exploiting its conversion to a DBN. First, the architecture of the tool is described, together with the rules implemented in the tool, to convert dynamic gates in DBNs. Then, the tool is tested on a case of system: its DFT model and the corresponding DBN are provided and analyzed by means of the tool. The obtained unreliability results are compared with those returned by other tools, in order to verify their correctness. Moreover, the use of DBNs allows to compute further results on the model, such as diagnostic and sensitivity indices.

1 Introduction

The modeling possibilities offered by *Fault Trees* (FT), one of the most popular techniques for dependability analysis of large, safety critical systems, can be extended by relying on Bayesian Networks (BN) [1, 2, 3, 4, 5].

This formalism allows to relax some constraints which are typical of FT, such as the hypothesis that elementary events are always modeled as binary objects (working/failed), are probabilistically independent, and interact just through Boolean AND/OR connections. In [1], it has been shown how FT can be directly mapped into BN, and that the basic inference techniques on the latter may be used to obtain classical parameters computed from the former. In addition, BN allow to represent local dependencies and to perform both predictive and diagnostic reasoning.

In [6], we have shown how BN can provide a unified framework in which also *Dynamic FT* (DFT) [7, 8], a rather recent extensions able to treat complex types of dependencies, can be represented.

In particular, 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) gate. WSP are dynamic gates modeling one or more principal components that can be substituted by one or more backups (spares), with the same functionality (Fig. 1(a)). The WSP gate fails when the number of operational powered spares and/or principal components is less than the minimum required. 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

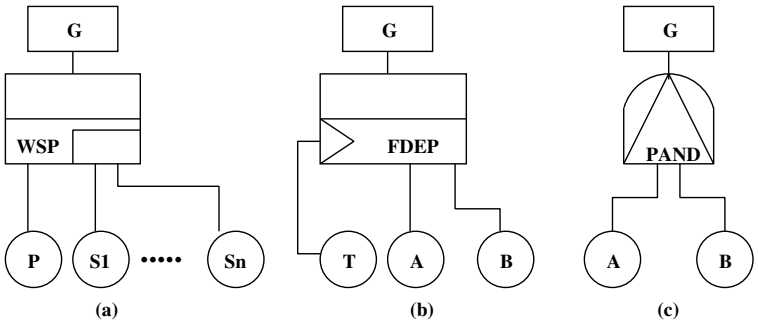

Figure 1: Dynamic gates in a DFT.

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 [8], so they will not be considered any more in the following[1]. In the PDEP gate (Fig. 1(b)), one trigger event T causes other dependent components to become unusable or inaccessible with probability $pdep \leq 1$. In particular, when the trigger event occurs, the dependent components will 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 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 force such a strong assumption: it simply detects the failure order and fails just in one case (in Fig. 1(c) a failure occurs iff A fails before B, but B may fail before A without producing a failure in G).

The quantitative analysis of DFT typically requires to expand the model in its state space, and to solve the corresponding Continuous Time Markov Chain (CTMC) [7]. Through a process known as modularization [9, 10, 11], it is possible to identify the independent sub-trees with dynamic gates, and to use a different Markov model (much smaller than the model corresponding to the entire FT) for

[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.

each of them. Nevertheless, there still exists the problem of state explosion.

In order to alleviate this limitation, as stated above, we have proposed a translation of the DFT into a *Dynamic Bayesian Network* (DBN). A DBN is a discrete time model where the system is represented at several time slices, and conditional dependencies among variables at different slices are introduced, to capture the temporal evolution [12, 13, 14]. When the Markov assumption holds (and in particular when we are dealing with a first order Markov process) the future slice at time $t + \Delta t$ is conditionally independent of the past ones given the present slice at time t [15]. 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* (CPT);

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 $t + \Delta t$ 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.

With respect to CTMC, the use of a DBN allows one to take advantage of the factorization in the temporal probability model. 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. In this paper, we describe a tool we have implemented to realize an automatic translation of a DFT into the corresponding DBN. The tool embeds an algorithm able to independently convert the different kinds of dynamic gates that may be found in a FT into a DBN, and then to build the overall network by linking the single DBN taking into account the types of connections that can exist among the gates themselves. The tool is provided with a user-friendly graphical interface, through which the user can draw the DFT in input, and then ask for the translation (or draw the DBN directly). Diagnostic and/or predictive computation can then be performed on the DBN, by resorting to standard algorithms (either exact or approximate) available for DBN, exploiting the factorization of the resulting probabilistic model. We also show an application of the tool functionalities to a real world example.

Our experimental results demonstrate how DBN can be safely resorted to if a quantitative analysis of the system is required. Moreover, the tool offers the possibility of drawing the system model relying on the more diffused FT language,

and of analyzing the model by means of the more powerful DBN. Finally, the exploitation of a DBN allows to obtain quantitative analysis results in a reasonable time, shorter with respect to DFT.

2 The *DBNet* tool

2.1 Tool functionalities

The tool we have implemented is called *DBNet* and allows the user to:

1. edit a (dynamic) Bayesian network and draw inferences on it;
2. edit a dynamic fault tree;
3. automatically convert a DFT into the corresponding DBN, on which both predictive and diagnostic inference can then be drawn.

Items (1) and (2) are described below; item (3), which represents the core contribution of this paper, is extensively treated in sections 2.2 and 2.3. The tool architecture is depicted in Fig. 2.

2.1.1 Editing a dynamic Bayesian network

By means of the tool graphical interface, which has been built relying on the *Draw-Net* graphical tool [16, 17], the user is allowed to draw the intra-slice conditional dependency model at time t, to duplicate it at time $t + \Delta t$, and finally to provide the inter-slice conditional dependency model. The term Δ is the discretization step, and can be set by the user trough the graphical interface. CPT can be easily inserted relying on a user friendly functionality, in which every row is automatically completed by calculating the last entry as the difference between 1 and the sum of the other values.

Additionally, it is possible to identify query nodes (not necessarily the *Top Event* (TE) i.e. the global system failure, as usually required in FT analysis), and to provide a stream of observations, each one coupled with its observation time.

When the network has been fully characterized, the user can activate a *monitoring* procedure, i.e. ask for the calculation of the probability of the query node(s) from the initial time t_0 to the mission time t_m, given a certain discretization step, and given the observation stream if available. The calculation of the TE is a special case of monitoring, with an empty stream of observations. In addition to the output of classical FT analysis, on the Bayesian network *smoothing* can be required as well. In this case, the past history of the system is rebuilt, on the basis

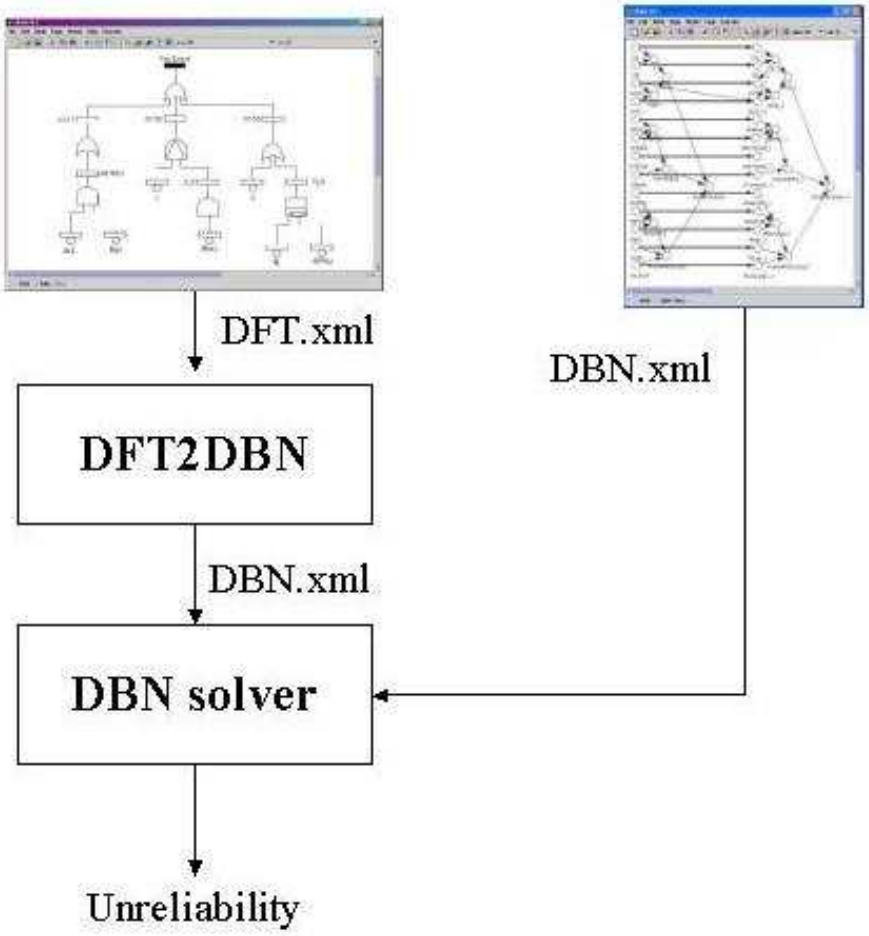

Figure 2: The *DBNet* tool architecture.

of the stream of observations. This procedure allows to obtain diagnostic knowledge (e.g. given that the TE has been observed, the probability of failure of basic components is queried).

Algorithms for monitoring and smoothing on DBN [14] have been implemented by resorting to Intel PNL (*Probabilistic Networks Library*), a set of open-source C++ libraries (see http://www.intel.com/research/mrl/pnl), to which we have provided some minor adjustments. PNL includes two classes of algorithms: the first class is composed by exact algorithms, based on the calculation of the junction tree; the second class includes parameterized algorithms, which require to cluster the overall network in a set of subprocesses.

Subprocesses are then solved independently, thus speeding up the computation time. The algorithms provide the exact result if the clusters have been correctly identified, i.e. if every subprocess evolves independently of the others; an algorithm to automatically find the correct clusters could be implemented as well.

2.1.2 Editing a dynamic Fault Tree

Modelling the failure mode of a system as a DBN might be complicated for the user, while drawing the DFT model and generating automatically the corresponding DBN, is more practical. In this way, the DFT becomes an high level formalism allowing the user to express in a straightforward way the relations between the components in the failure mode of the system, whose modelling in terms of DBN primitives would be less comfortable.

Draw-Net already included a DFT editor. We have ameliorated it by providing the possibility of drawing dynamic gates, and of indicating query variables and observed variables. On the DFT, the user can also specify the discretization step, as well as the mission time, and the inference algorithm to be adopted on the corresponding DBN.

This information is directly inherited by the corresponding DBN when translation is required. In this way, the DFT is exploited as an easy and well known formalism, to which the user is typically already familiar, through which all the needed data for DBN inference can be given in input. In the future, we plan to extend these capabilities, by adding ad hoc structures to the DFT, which can then be naturally characterized in the corresponding DBN: for example, we will allow the insertion of multi-valued nodes, and the specification of conditional dependencies among basic events. Moreover, we plan to automatically calculate the clusters to be exploited by the approximated inference algorithms, by a modularization procedure on the DFT [11].

The DBN structure corresponding to the DFT, as well as all the CPT are then automatically generated by the tool, and (forward or backward) inference can be finally executed. In the next section we describe the details of the translation

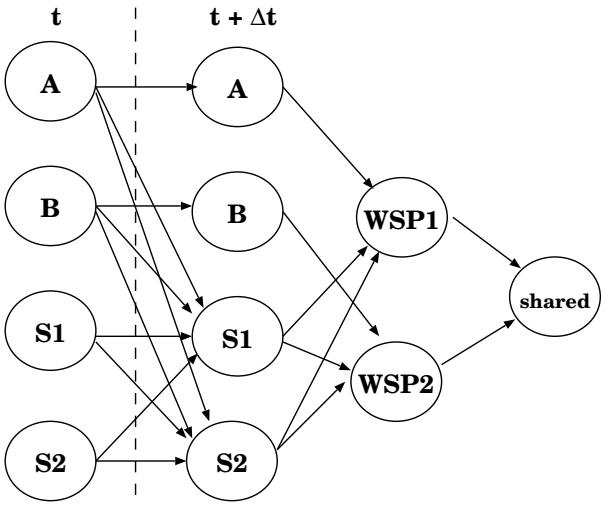

Figure 3: DBN for the WSP gate.

procedure.

2.2 Translating dynamic gates

In this section, we present the conversion of each dynamic gate (WSP, PDEP, PAND) in the corresponding DBN. The rules to convert Boolean gates (AND, OR, K out of N) can be found in [1].

2.2.1 Warm spare gate

In a DFT, different configurations of warm spares can be designed. Of particular interest are those in which the same pool of spares is shared across a set of WSP. In this case, each principal component is allowed to request the items in the pool in a precise order - if more than one is still dormant. As an example, let us consider a situation where two components A and B can be substituted by two spares S1 and S2. In particular, S1 is B's spare, and will substitute A only if: (i) B is working, and (ii) S2 is failed. If B fails, it will request the activation of S1, and only if it is unavailable it will activate S2. S2 is A's spare: analogous considerations hold. Every gate fails iff its principal component and all the available (i.e. working and dormant) spares in the pool fail. The DBN corresponding to this situation is shown in Fig. 3.

It can be observed that each component node at time $t + \Delta t$ depends on its copy at time t (we consider persistence of faults). Moreover, each spare depends

on the two principal components, and on the other spare. Each spare is modeled as a stochastic variable assuming four values, namely: dormant, operative on A, operative on B and failed. If both principal components are working, each spare maintains a failure rate equal to $\alpha\lambda$ at time $t+\Delta t$. On the other hand, if B is down, S1 switches to a failure rate equal to λ (since the spare is now in the active mode); the same happens if B is working, but A and S2 are both failed. S2 works dually on its principal component A. Each WSP gate (in the example, the one having A as its principal component, and the one having B as its principal component) is modeled as a deterministic AND node among its three inputs: the principal components and the two spares in the pool. The overall set of WSP sharing the pool can be modeled as a $k : n$ gate (2:4 in the example), where n is the number of principal components and of available spares in the pool, and k is equal to the number of WSP gates sharing the spares.

2.2.2 Probabilistic dependency gate

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

Fig. 4 shows the DBN for a PDEP gate in a configuration in which the trigger event T has two dependent components A and B. As usual, each component at time $t + \Delta t$ depends on the component itself at time t. Moreover, the dependent components will fail (with probability $pdep$) if the trigger has failed in the same time slice.

2.2.3 Priority AND

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 Fig. 1(c): we can model the failure sequence by introducing a new stochastic variable PF, that explicitly keeps track of the order in which A and B fail. PF at time $t + \Delta t$ depends on all the variables at time t. In particular, if it was already failed at time t, it will remain failed at time $t+\Delta t$. It will also fail iff A(t)=1 and B(t)=0, i.e. A fails before B. The PAND gate is modeled as a logic AND among its three inputs A, B and PF at time $t+\Delta t$. Fig. 5 shows the resulting DBN. An hypothesis on how to deal with contemporary faults of A and B has to be made; for example, we have made the choice that a

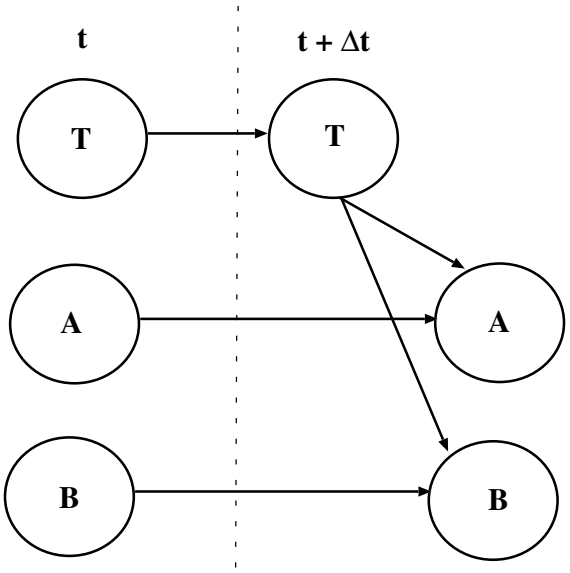

Figure 4: DBN for the PDEP gate.

contemporary fault sets PF($t + \Delta t$) to 1, and therefore leads to a fault of the whole gate.

2.3 Combining the modules into a single DBN

The three gates we have examined in the previous section can be connected, in order to build a complex DFT. To understand what combinations can be modeled, and how we can provide an automatic translation of the DFT into the corresponding DBN, we have to recall how the gates themselves are meant to be applied.

In particular, according to [8]:

1. the dependent events of a PDEP can only be basic events, which could be the input of another dynamic gate;

2. WSP can have only basic events as an input, and two or more spare gates can share some spare components; a set of WSP sharing a pool of spares are treated as a single module for translation (see section 2.2);

3. PAND can have basic events or spare gates as an input; two or more PAND can also be combined in a cascade manner.

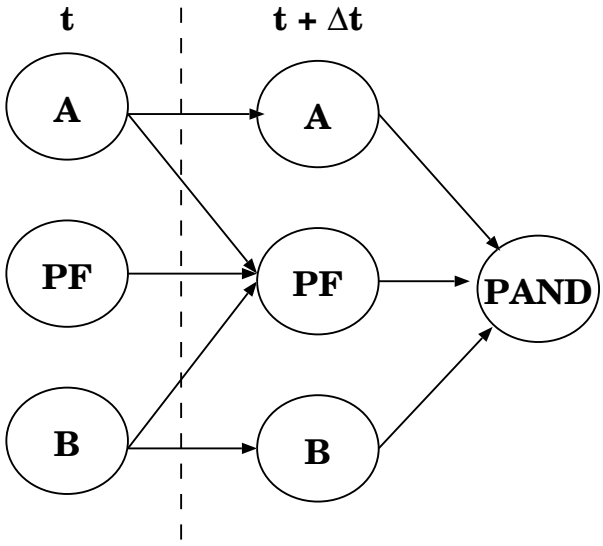

Figure 5: DBN for the PAND gate.

These simplifications let us derive a general algorithm for building the DBN corresponding to the overall DFT. The procedure follows a modular approach, in the sense that it builds the output DBN by combining the various DBN corresponding to the different gates.

Consider a pair of dynamic gates G1 and G2. Let B be the output of G1 (or an event triggered by G1 if G1 is a PDEP), and an input to G2 (B is therefore the element connecting the two gates). The algorithm works as follows:

1. generate independently DBN1 and DBN2 for G1 and G2 respectively, along the lines explained in section 2.2;

2. connect the two networks in correspondence of the nodes they share; when new arcs enter a node, an adjustment to its CPT is required, as follows:

   a. add all the parents derived from DBN1 and DBN2 as columns in the new CPT;

   b. in every entry of the table, set the probability of failure of the node to the maximum between the corresponding entries of the CPT in DBN1 and in DBN2. In this way the strongest effect on the probability of failure of the shared component is always considered.

3. repeat the above step for every shared node between the two networks (more nodes could be shared e.g. when G1 is a PDEP, and it triggers more than one input to G2).

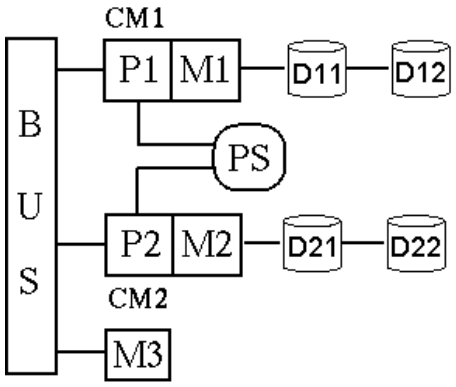

Figure 6: The scheme of the multiprocessor computing system.

As a final remark, recall that in section 1, we have cleared that SEQ can be treated as cold spares. In [8], on the other hand, it is explained that SEQ can receive also WSP and PAND in input, determining a difference in the gates behavior.

Anyway, note that our algorithm would not change if we removed the hypothesis that cold spares can have only basic events as an input: therefore we do not need to explicitly reintroduce SEQ gates in our framework, and our algorithm appears to be general enough to cover all the modeling possibilities.

3 An example

In this section, an example is reported in order to show how it is possible to convert a DFT model in a DBN and to perform the unreliability evaluation on it.

3.1 Description of the example

The example is inspired from [18] and consists of a multiprocessor computing system; it is composed by two computing module: CM1 and CM2. Each of them contains one processor, one memory and two hard disks: a primary and a backup disk. In the case of CM1, they are respectively indicated as P1, M1, D11, D12. In the case of CM2, they are: P2, M2, D21, D22 (see Fig. 6).

Initially, the primary disk is accessed by the computing modules to store and retrieve programs and data, while the backup disk contains the copy of the information inside the primary disk, and is accessed only periodically for update operations. For this reason, in this situation, the failure rate of the backup disk is lower than the failure rate of the primary one. If the primary disk fails, it is

replaced in its function by the backup disk; from this moment, the failure rate of the backup disk is equal to the failure rate of the primary one. In other words, the backup disk of a computing module, is the spare of the primary disk.

A spare memory (M3) is present in the system; its aim is replacing M1 or M2 in the case of failure. Moreover, the computing modules are connected by means of the bus N. Finally, both P1 and P2 are functionally dependent on the power supply PS; this means that the failure of PS forces the failure of P1 and P2.

3.2 The failure mode of the system and its DFT model

The whole system fails if the bus N fails preventing the connection between the computing modules, or if both of them are failed. This is represented in the DFT model of the system (Fig. 7), by the *Top Event* (TE) which is the output of an OR gate having these input events: the basic event N, relative to the bus, and the event S indicating the failure of both computing modules. S is the output of an AND gate whose input events are CM1 and CM2.

A computing module fails if both disks are failed, or if the processor is failed, or if the internal memory is failed and it can not be replaced by M3 because M3 is failed too, or it is already replacing the internal memory of the other computing module. So, in the DFT model, CM1 is the output of an OR gate having the events Disk1, P1 and Mem1 as input events; such events correspond respectively to the conditions determining the failure of a computing module. The event Disk1 is the output of a WSP gate having the basic event D11 as main component and the basic event D12 as spare component; the event Mem1 is the output of a WSP gate whose input events are M1 (main component) and M3 (spare component).

The failure of CM2 is modelled in the DFT model in a similar way, by the subtree rooted in the event CM2 which is symmetric to the subtree whose root is CM1. The basic event M3 is connected to two WSP gates since it can replace either M1 or M2.

The functional dependency of P1 and P2 on PS, is modelled by a PDEP gate whose probability is equal to 1, and having PS as trigger event, and the basic events P1 and P2 as dependent events.

The time to fail of any component in the system is a random variable ruled by the negative exponential distribution; Tab. 1 shows the failure rate of every component, together with the dormancy factors of the spare components. Fig. 7 has been drawn by means of the graphical interface of our tool, described in section 3.5.

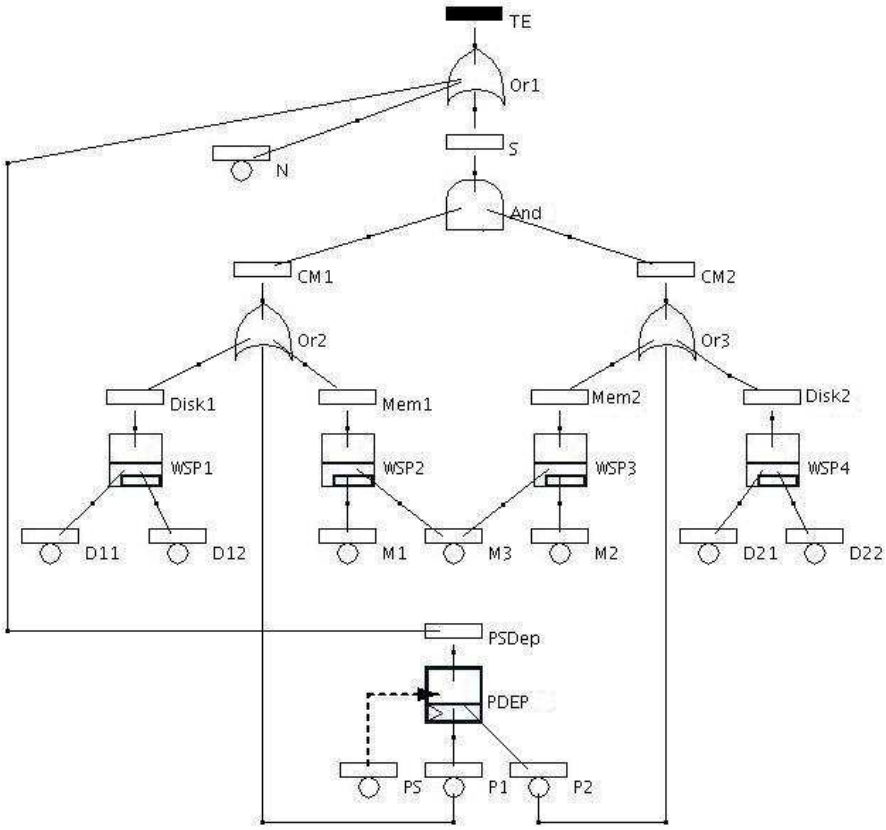

Figure 7: The DFT model of the multiprocessor computing system.

Component	Failure rate λ (h^{-1})	Dormancy factor α
N	2.0E-9	
P1, P2	5.0E-7	
PS	6.0E-6	
D11, D21	8.0E-5	
D12, D22	8.0E-5	0.5
M1, M2	3.0E-8	
M3	3.0E-8	0.5

Table 1: The failure rates and the dormancy factors.

3.3 Conversion of the DFT model to a DBN

In order to perform the unreliability evaluation of the system through its DFT model, in our approach we convert it to a DBN. The DBN corresponding to the DFT in Fig. 7, is shown in Fig. 8 and can be automatically generated given the DFT model, by using our tool.

The nodes inside the DBN derive from the translation of the basic events and of the gates (both static and dynamic) inside the DFT; the translation of the internal events and of the top event, is not necessary [1]. The DBN nodes are duplicated for each time slice (t, $t+\Delta t$) according to the 2TBN representation [14]. Temporal arcs (drawn as thicker lines) connect the nodes representing the temporal copies of the same basic event; in this way, we model the failure persistence allowing the possibility of representing also repair processes.

For instance, the WSP gate in Fig. 7 having D21, D22 as input events, and Disk2 as output event, corresponds in the DBN in Fig. 8, to the subnet composed by the nodes D21, D21_1, D22, D22_1, Disk2_1, together with the arcs connecting such nodes.

3.4 Unreliability results

After the conversion of the DFT in a DBN, we can perform the analysis of the latter by means of our tool. Tab. 2 shows the unreliability of the system versus the mission time varying between 1000 and 5000 hours ($\Delta t = 1h$), obtained by monitoring the node TE_1 without the observation stream for each node; this method is equivalent to the prediction. The obtained results have been successfully verified by comparison with the results returned by other tools run on the same DFT model. Such tools are *DRPFTproc* [19, 20] (based on modularization [11] and conversion to *Stochastic Petri Nets* of dynamic gates) and *Galileo* [21, 22] (based on modularization, *Binary Decision Diagrams* (BDD) [23, 24] and CTMCs). Tab.

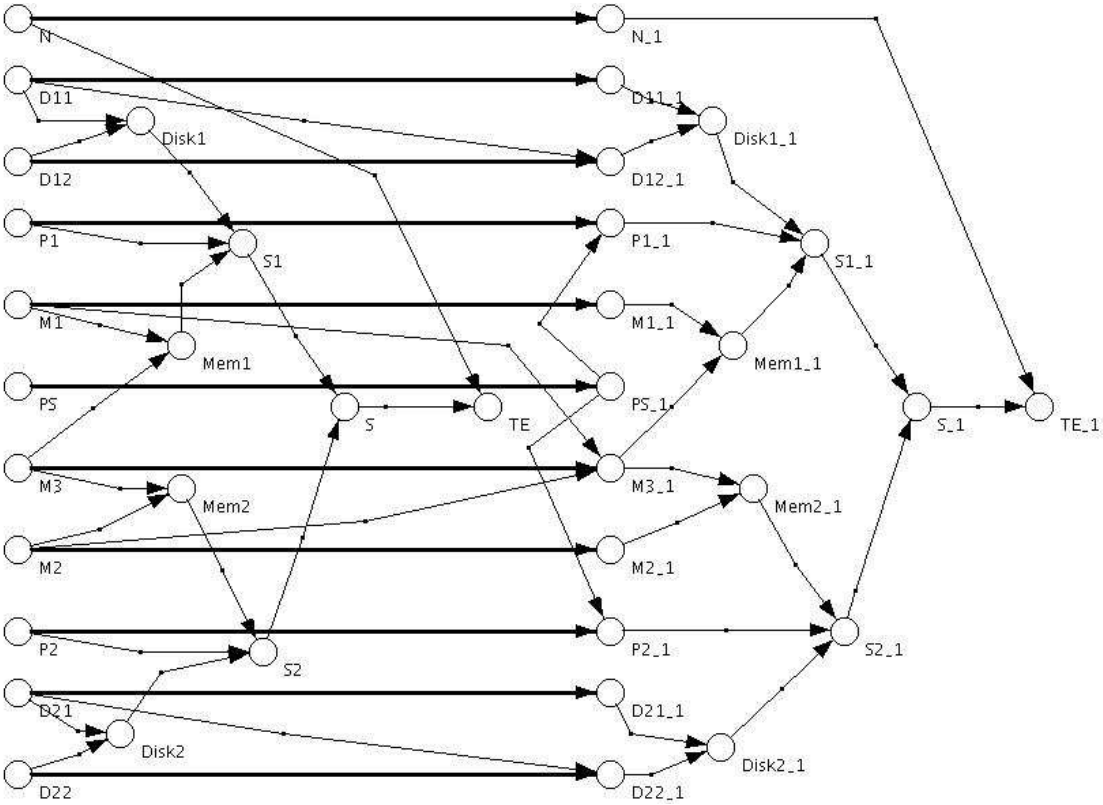

Figure 8: The DBN corresponding to the DFT in Fig. 7.

2 reports also the results obtained using such tools.

The use of DBNs allows to compute further results based on the conditional probability, such as diagnostic and sensitivity indices [25].

3.5 Graphical interface description

Fig. 9 shows a screenshot of the graphical interface (*Draw-Net*) of our tool. It is mainly composed by three windows; the main window allows the user to draw the DFT model, while in the window named *Property Page*, it is possible to set the attributes of the node or of the arc currently selected in the main window. From the *Execute* menu of the main window, the user can run the conversion and the analysis of the DFT model. At the end of such process, the obtained results are displayed in the window called *Solver Execution*.

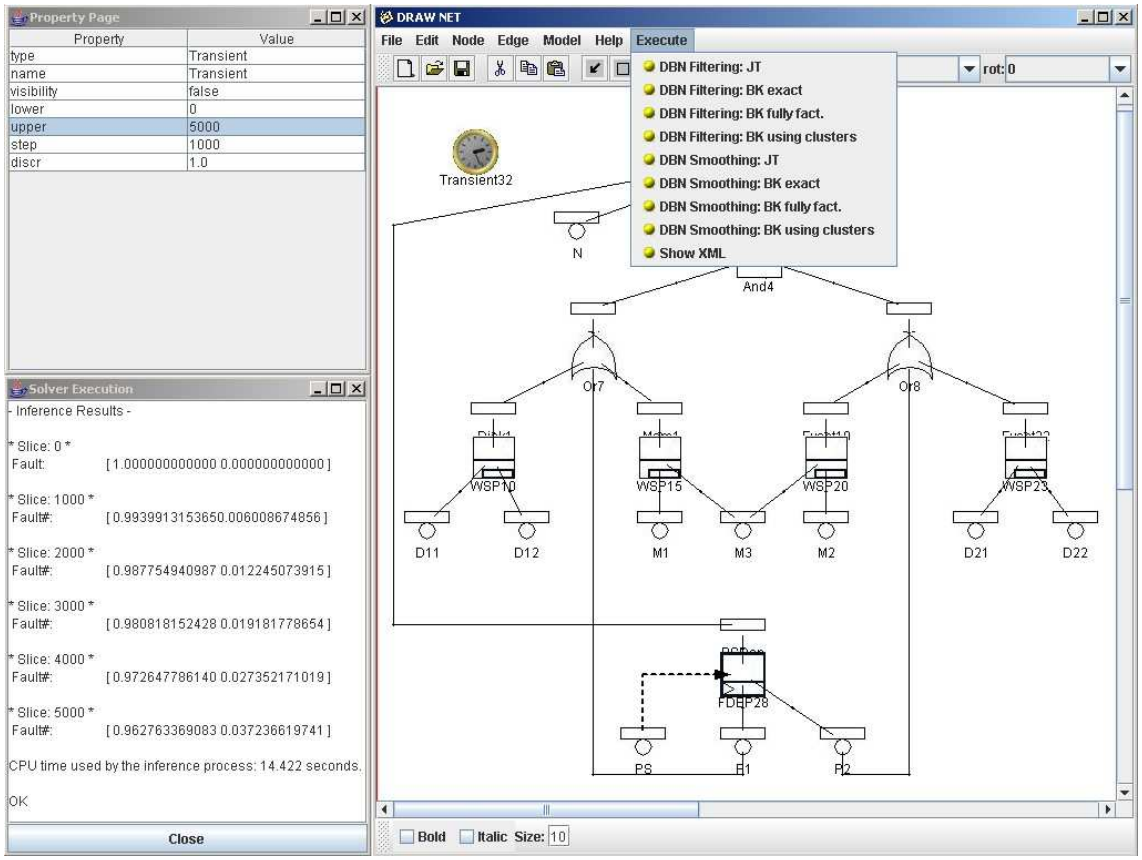

Figure 9: A screenshot of the *DBNet* tool.

time (h)	*DBNet*	*DRPFTproc*	*Galileo*
1000	0.0060086	0.0060088	0.0060088
2000	0.0122452	0.0122455	0.0122455
3000	0.0191820	0.0191832	0.0191832
4000	0.0273523	0.0273548	0.0273548
5000	0.0372379	0.0372413	0.0372413

Table 2: The unreliability results.

4 Conclusions

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 into the framework of BN. We have also shown how FT with dynamic gates, a recent extension to traditional FT that allows to model complex dependency types, can be characterized in the framework of BN as well. In particular, we have shown how to translate a dynamic FT into a dynamic BN.

Resorting to the DBN formalism has the well-known advantages of exploiting all the modeling capabilities of graphical probabilistic models: multi-valued variables (instead of binary events), local dependencies among components (instead of classical s-independence assumption), noisy interaction among component behavior (instead of deterministic interaction).

In addition, general inference mechanism (combining prediction as well as diagnosis) can be naturally performed on a DBN, while they are not easily implemented in standard FT analysis, especially if evidence is gathered during analysis itself.

Finally, with respect to the use of CTMC, which are traditionally resorted to in order to solve dynamic FT, DBN allow to take advantage of the sparseness in the temporal probability model, and to rely on approximate inference algorithms, thus reducing the computation time.

In this paper, we have described a tool that allows the user to draw a DBN and to ask for diagnostic or predictive inference on it, as well as to draw a DFT, obtain an automatic conversion into the corresponding DBN, and ask for inference calculation.

To test both the proposed conversion methodology and the tool performances, we have run some examples, one of which has been presented in this paper: the results obtained using the DBN are basically identical to the ones obtained using other analysis techniques described in the literature. Our experimental results therefore demonstrate how DBN can be safely resorted to if a quantitative analysis

of the system is required.

In the future, we plan to extend the tool capabilities, by adding ad hoc structures to the DFT, which can then be naturally characterized in the corresponding DBN.

In this way, the tool will offer the possibility of drawing the system model relying on the well known and more diffused FT language, and of analyzing the model itself by means of the more powerful DBN.

ACKNOWLEDGEMENTS

This work was supported by the Italian Ministry of Education, University and Research (MIUR) in the framework of the FIRB-Perf project.

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] P. Weber and L. Jouffe. Reliability modelling with dynamic bayesian networks. In *Proceedings of SAFEPROCESS 2003*, Washington D.C., 2003.

[3] H. Boudali and J. B. Dugan. A new Bayesian network approach to solve dynamic fault trees. In *Proceedings of the Annual Reliability and Maintainability Symposium*, Alexandria, USA, 2005.

[4] H. Langseth. *Bayesian networks with application to reliability analysis*. PhD thesis, Dept. of Mathematical Sciences, Norvegian University of Science and Technology, 2002. (http://www.math.ntnu.no/ helgel/thesis).

[5] J. G. Torres-Toledano and L. E. Sucar. Bayesian networks for reliability analysis of complex systems. *Springer Verlag LNCS*, 1484:195–206, 1998.

[6] S. Montani, L. Portinale, and A. Bobbio. Dynamic bayesian networks for modeling advanced fault tree features in dependability analysis. In *Proceedings of the European Conference on Safety and Reliability*, Tri City, Poland, June 2005.

[7] J. B. 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.

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

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

[10] Anand and A. K. Somani. Hierarchical analysis of fault trees with dependencies, using decomposition. In *Proc Annual Reliability and Maintainability Symposium*, pages 69–75, 1998.

[11] R. Gulati and J. B. Dugan. A Modular Approach for Analyzing Static and Dynamic Fault Trees. In *Proceedings of the Annual Reliability and Maintainability Symposium*, pages 1–7, 1997.

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

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

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

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

[16] G. Franceschinis, M. Gribaudo, M. Iacono, N. Mazzocca, and V. Vittorini. DrawNet++: Model Objects to Support Performance Analysis and Simulation of Complex Systems. *Springer Verlag LNCS*, 2324:233–238, 2002.

[17] G. Franceschinis, M. Gribaudo, M. Iacono, V. Vittorini, and C. Bertoncello. Drawnet++: a flexible framework for building dependability models. In *In Proc. of the Int. Conf. on Dependable Systems and Networks*, Washington DC, USA, June 2002.

[18] M. Malhotra and K. Trivedi. Dependability modeling using Petri Nets. *IEEE Transactions on Reliability*, R-44:428–440, 1995.

[19] A. Bobbio and D. Codetta-Raiteri. Parametric Fault Trees with Dynamic Gates and Repair Boxes. In *Proceedings of the Annual Reliability and Maintainability Symposium*, pages 459–465, Los Angeles, January 2004.

[20] D. Codetta-Raiteri. Development of a dynamic fault tree solver based on colored petri nets and graphically interfaced with drawnet. Technical Report TR-INF-2003-10-06-UNIPMN, Dipartimento di Informatica, Universitá del Piemonte Orientale, October 2003.

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

[22] K. J. Sullivan, J. B. Dugan, and D. Coppit. The Galileo Fault Tree Analysis Tool. In *Proceedings of IEEE International Symposium on Fault Tolerant Computing*, 1999.

[23] A. Rauzy. New Algorithms for Fault Trees Analysis. *Reliability Engineering & System Safety*, 05(59):203–211, 1993.

[24] S. A. Doyle and J. B. Dugan. Dependability Assessment Using Binary Decision Diagrams (BDDs). In *FTCS '95: Proceedings of the Twenty-Fifth International Symposium on Fault-Tolerant Computing*, page 249, Washington, DC, USA, 1995. IEEE Computer Society.

[25] Y. Dutuit, O. Lemaire, and A. Rauzy. New Insight on Measures of Importance of Components and Systems in Fault Tree Analysis. In *Proceedings of the International Conference on Probabilistic Safety Assessment and Management*, pages 729–734, 2000.