A GSPN based tool to inference Generalized Continuous Time Bayesian Networks

Daniele Codetta-Raiteri, Luigi Portinale
DiSIT, Computer Science Institute, University of Piemonte Orientale
Viale Teresa Michel 11, Alessandria, Italy
{dcr, luigi.portinale}@di.unipmn.it

ABSTRACT

We present a software tool for the analysis of Generalized Continuous Time Bayesian Networks (GCTBN) which extend CTBN introducing in addition to continuous time delayed variables, non delayed or "immediate" variables whose evolution is conditionally and immediately determined by the values of other variables in the model. The tool is based on the conversion of a GCTBN model into a Generalized Stochastic Petri Nets (GSPN) which is an actual mean to perform the inference (analysis) of the GCTBN.

1. INTRODUCTION

Bayesian Networks (BN) [3] are a widely used formalism for representing uncertain knowledge in probabilistic systems. BN are defined by a graph where nodes corresponds to discrete random variables having a conditional dependence on the parent nodes. Root nodes are nodes with no parents, and marginal prior probabilities are assigned to them. An added value is that BN allow not only a forward (or predictive) analysis, but also a backward (or diagnostic) analysis, where the posterior probability distribution of any set of variables can be computed, given the observation of the values of other variables (evidence).

BN have been recently investigated as very promising formalisms for reliability analysis [3, 8] because BN can easily model probabilistic dependencies, multi-state components, state dependent failure probability, imperfect coverage of failure and repair, noisy Boolean relations, etc.

In this paper, we present a software tool for the analysis of *Generalized Continuous Time Bayesian Network* (GCTBN) [6] characterized by a temporal dimension (Sec. 2) and the presence of both delayed and immediate variables evolving with a random delay or in an instantaneous way, respectively (Sec. 3). This distinction can be exploited in several applications. For example, in system reliability analysis, it is very practical to distinguish between system components (having a temporal evolution) and subsystems whose state is immediately determined by the state of their components.

VALUETOOLS 2013, December 10-12, Torino, Italy

DOI 10.4108/icst.valuetools.2013.254400

This is shown by the case study presented in Sec. 4. Actually a GCTBN is characterized by the same stochastic process of a *Generalized Stochastic Petri Net* (GSPN) [7], as described in Sec. 5. So, the software tool (Sec. 6) is based on the conversion of a GCTBN into a GSPN, and the GCTBN analysis (or *inference*) is actually performed on the GSPN through a sequence of transient solutions.

2. RELATED WORK

BN models for reasoning about processes that evolve over time have been deeply investigated. When time is taken into account, the main choice concerns whether to consider it as a discrete or a continuous dimension. In the first case, models like *Dynamic Bayesian Networks* (DBN) [8] have become a natural choice. In the second case, *Continuous Time Bayesian Networks* (CTBN) [5] have been proposed and contain variables with a continuous time evolution. In [6] CTBN have been extended to GCTBN. At the moment, the GSPN based approach is the only method developed to solve GCTBN models. This method will be useful in the future to verify inference results if a direct solver will be developed for GCTBN. CTBN solution methods [5] may be extended to deal with GCTBN. However, CTBN solution usually provides approximate results, while the GSPN based approach returns exact results. Moreover, solution techniques for GSPN have recently received a lot of attention, especially with respect to the possibility of solving the underlying state space efficiently, in order to deal with extremely large models [4].

3. GCTBN FORMALISM

D is the set of *delayed* variables having a continuous time evolution (inherited from CTBN). The transition from a value to another one, is ruled by exponential transition rates defined in the *Conditional Intensity Matrix* (CIM) [5] associated with the node. A delayed node X is characterized also by the initial probability distribution P_X^0 of its possible values. So, a delayed node implicitly incorporates a *Continuous Time Markov Chain* (CTMC) [7]. If a delayed node is a root node (has no parent nodes) the transition rates are constant, otherwise the rates are conditioned by the values of the parent nodes.

I is the set of *immediate* variables whose value immediately changes according to the values of the parent variables. Immediate nodes are introduced in order to capture variables whose evolution is not ruled by transition rates associated with their values, but is conditionally and immediately determined, at a given time point, by the values of other

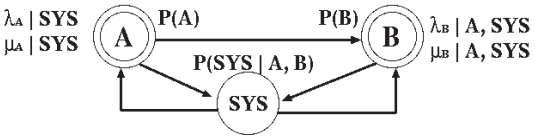

Figure 1: GCTBN model of the case study.

a)

$1 \to 2$		$2 \to 1$	
SYS	λ_A	SYS	μ_A
1	1.0E-06 h^{-1}	1	0 h^{-1}
2	1.0E-06 h^{-1}	2	0.01 h^{-1}

b)

$1 \to 2$			$2 \to 1$		
A	SYS	λ_B	A	SYS	μ_B
1	1	5.0E-07 h^{-1}	1	1	0 h^{-1}
1	2	–	1	2	–
2	1	1.0E-06 h^{-1}	2	1	0 h^{-1}
2	2	5.0E-07 h^{-1}	2	2	0.01 h^{-1}

c)

A	B	SYS	Prob.	A	B	SYS	Prob.
1	1	1	1	2	1	1	0.99
1	1	2	0	2	1	2	0.01
1	2	1	1	2	2	1	0
1	2	2	0	2	2	2	1

Table 1: a) CIM of A. b) CIM of B. c) CPT of SYS.

variables in the model. Immediate nodes are then treated as usual chance nodes in a BN and have a standard *Conditional Probability Table* (CPT) [3] associated with them. In case of an immediate root node, its CPT actually specifies a prior probability distribution.

3.1 Stochastic process

An immediate node I_j directly depends on its parent nodes ($Pa(I_j)$). However, if the set $Pa(I_j)$ contains immediate nodes, then the change of such nodes is ruled in turns by the change of their parents, possibly being delayed variables. So, what really determines a change in I_j is not $Pa(I_j)$, but instead the set of the *"Closest" Delayed Ancestors* of I_j ($CDA(I_j)$) [6]. Such set contains any delayed variable D_k such that a path from D_k to I_j exists and contains no intermediate delayed nodes.

The evolution of a system modeled through a GCTBN occurs as follows: the initial state is given by the assignment of the initial values of the variables, according to P_X^0 (immediate root nodes, if any, keep their initial value during the model evolution). Given the current system state (represented by the joint assignment of the model variables, both delayed and immediate), a value transition of a delayed variable D_k will occur, after an exponentially distributed delay, by producing a new state called a *"vanishing state"*; given the new vanishing state, a new assignment is determined to any immediate variable I_j such that D_k belongs to $CDA(I_j)$. The assignment to I_j is consistent with the CPT of I_j. The resulting state, called a *"tangible state"*, is the new actual state of the system, from which the evolution can proceed with a new transition of value by a delayed variable. The same state classification can be recognized in GSPN.

4. CASE STUDY

We now consider a simple case study in the field of reliability analysis [6]. It is composed by the main component A and its spare component B. Initially A is active while B is dormant; in case of failure of A, B is activated in order to replace A. However, the activation of B may fail with probability 0.01. If B fails before A, B can not replace A. The system is considered as failed if A is failed and B is dormant or failed. Only while the system is failed, A and B undergo repair. As soon as the repair of one of them is completed, the component re-starts in working state and consequently the system becomes operative again; this implies that the repair of the other component is suspended.

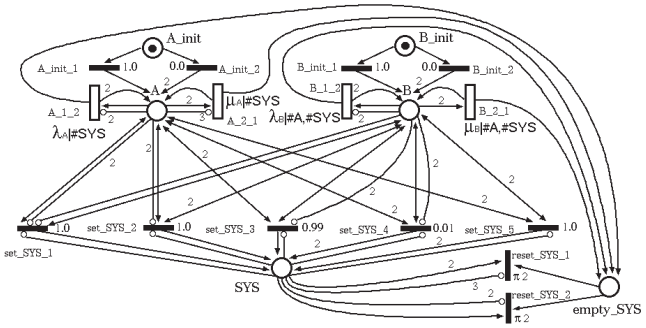

Figure 2: GSPN obtained from the GCTBN.

In principle, different repair policies can be adopted and modelled; the one proposed in this example recovers the minimal set of necessary working components. This is just to illustrate a particular dependency of A and B on the state of the system. The aim of the case study is to illustrate the approach which can be applied to more complex systems.

4.1 GCTBN model

The case study is represented by the GCTBN in Fig. 1 where the variables A, B, SYS represent the state of the components and of the whole system, respectively. All the variables are binary because each entity can be in the working (value 1) or in the failed state (value 2); for B, the working state comprises both the dormancy and the activation.

A influences B because the failure rate of B depends on the state of A. Both A and B influence SYS because the state of the whole system depends on the state of A and B. The arcs connecting SYS to A and B respectively, concern the repair of A and B only while the system is failed.

A and B are delayed variables (Sec. 3) and implicitly incorporate a CTMC composed by two states: 1 and 2. Since both components are initially supposed to work, the initial probability distribution is set equal to 1 for states $A = 1$ and $B = 1$. In the CIM of A (Tab. 1.a), we can notice that the rate μ_A is not null only if the value of SYS is 2. The rate λ_A instead, is constant. In the CIM of B (Tab. 1.b), λ_B is increased only when A is equal to 2 and SYS is equal to 1 (this implies that B is active). As in the case of A, the rate μ_B is not null only if the value of SYS is 2. The combination $A = 1, SYS = 2$ is impossible, so the corresponding entries are not significant.

SYS is immediate (Sec. 3) and is characterized by the CPT appearing in Tab. 1.c. In particular, SYS is surely equal to 1 if A is equal to 1, and surely equal to 2 if both A and B are equal to 2. In the case of A equal to 2 and B equal to 1, SYS assumes the value 1 with probability 0.99 (this implies the activation of B), or the value 2 with probability 0.01 (this implies that B fails to activate).

Procedure PREDICTION
INPUT: a set of queried variables Q, a query time t, a set of temporally labeled evidences $e_{t_1}, \ldots e_{t_k}$ with $t_1 < \ldots t_k < t$
OUTPUT: $P(Q_t|e_{t_1}, \ldots e_{t_k})$

- let $t_0 = 0$;
for $i = 1$ **to** k **{**
- solve the GPSN transient at time $(t_i - t_{i-1})$;
- compute from transient, $p_i(j) = Pr\{X_j|e_{t_i}\}$ for $X_j \in D \cup R$;
- update the weights of the immediate init transitions of X_j according to $p_i(j)$; **}**
- solve the GPSN transient at time $(t - t_k)$;
- compute from transient, $r = Pr\{Q\}$;
- output r;

Figure 3: The prediction inference procedure.

Procedure SMOOTHING
INPUT: a set of queried variables Q, a query time t, a set of temporally labeled evidences $e_{t_1}, \ldots e_{t_k}$ with $t < t_1 < \ldots t_k$
OUTPUT: $P(Q_t|e_{t_1}, \ldots e_{t_k})$;

{ - Let N be the cardinality of possible assignments q_i $(1 \leq i \leq N)$ of Q;
- A: **array**[N];
for $i = 1$ **to** N A[i]=SMOOTH(q_i); //possibly in parallel
- output **normalize**(A); **}**

Procedure SMOOTH**(q) {**
- $t_0 = t$;
- solve the GPSN transient at time t;
- compute from transient, $r = Pr\{Q = q\}$;
- $ev = q$;
for $i = 1$ **to** k **{**
- compute from transient, $p_{i-1}(j) = Pr\{X_j|ev\}$ for $X_j \in D \cup R$;
- update the weights of the immediate init transitions of X_j according to $p_{i-1}(j)$;
- solve the GPSN transient at time $(t_i - t_{i-1})$;
- compute from transient, $p_i(e) = Pr\{e_{t_i}\}$
- $r = r \cdot p_i(e)$;
- $ev = e_{t_i}$; **}**
- output r; **}**

Figure 4: The smoothing inference procedure.

5. INFERENCE TASKS

Prediction consists in computing $P(Q_t|e_{t_1}, \ldots, e_{t_k})$ which is the posterior probability at time t of a set of queried variables $Q \subseteq (D \cup I)$, given a stream of observations (evidence) $e_{t_1}, \ldots, e_{t_k}$ from time t_1 to time t_k with $t_1 < \ldots t_k < t$. Every observation e_{t_j} consists of a (possibly different) set of instantiated variables.

Smoothing consists in computing $P(Q_t|e_{t_1}, \ldots, e_{t_k})$ which is the probability at time t of a set of queried variables $Q \subseteq (D \cup I)$, given a stream of observations (evidence) $e_{t_1}, \ldots, e_{t_k}$ from time t_1 to time t_k with $t < t_1 < \ldots t_k$.

The difference between prediction and smoothing relies on the fact that in the former case, while computing the probability at time $t + \Delta$ with $\Delta \geq 0$, only the evidence (observations) gathered up to time t is considered. On the contrary, in the case of smoothing the whole evidence stream is always considered in the posterior probability computation.

For diagnostic purposes, prediction can be exploited for the assessment of the current or future system state while monitoring of the system behaviour. Smoothing instead, may be exploited in order to reconstruct the history of the system components for a kind of temporal diagnosis.

5.1 Inference on GSPN

GCTBN and GSPN share the same stochastic process composed by *tangible* states characterized by an actual duration, and *vanishing* states where the system spend no time (Sec. 3.1). According to the rules described in [6], a GCTBN can be converted into the equivalent GSPN. In Fig. 2 the places A, B and SYS represent the variables in Fig. 1; their evolution is due to the timed or immediate transitions [6].

A single transient analysis of the GSPN at the query time is not enough to perform the GCTBN inference which must take into account the set of observations (evidence) and their times of occurrence. So, the GSPN model has to be modified and analyzed at each time of interest. The results obtained at each time, must be properly combined in order to return the effective probability distribution of the queried variables. This process is described in the GSPN based algorithms for prediction and smoothing tasks reported in Fig. 3 and 4, respectively; more details are available in [6].

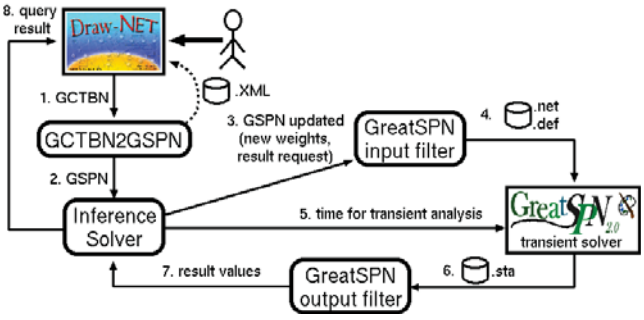

Figure 5: Scheme of the software tool.

6. SOFTWARE TOOL

By means of our tool, the GCTBN model is built and evaluated through the following steps, as depicted in Fig. 5.
1. The user designs the GCTBN including the observations and the query to be evaluated. Then, the user invokes the analysis of the GCTBN. This is done by means of the graphical tool *Draw-Net* [2] (Fig. 6) which can be configured for any kind of formalism. The GCTBN is passed to the translator *GCTBN2GSPN*.
2. The GCTBN is converted into the equivalent GSPN by means of *GCTBN2GSPN* following and implementing the specific translation rules [6]. The resulting GSPN is passed to the *Inference Solver*.
3. According to the inference procedure (filtering or smoothing) required by the user, the *Inference Solver* updates the GSPN with the next measures to be computed and the new transition weights (Sec. 5.1). The updated GSPN is passed to the *GreatSPN input filter*.
4. The *GreatSPN input filter* stores the GSPN in a couple of files (.net, .def) conforming to the *GreatSPN* [1] format. In particular, the .net file contains the specification of places and transitions in the GSPN, while the .def file contains the specification of the measures to be computed.
5. The .net and .def files together with the time of analysis, are passed to the *GreatSPN transient solver*.
6. The *GreatSPN transient solver* performs the analysis of the GSPN at the time specified by the *Inference Solver*. The values of the results are stored in the .sta file.
7. The *GreatSPN output filter* loads the results from the .sta file and provides them to the *Inference Solver*. The steps from 3 to 7 are repeated according to the inference

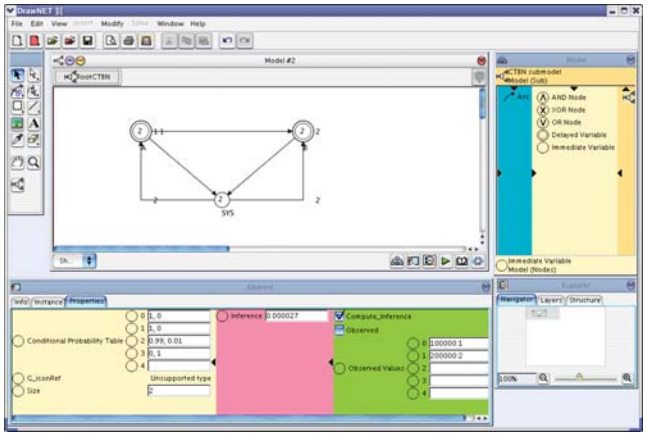

Figure 6: The GCTBN visualized by *Draw-Net*.

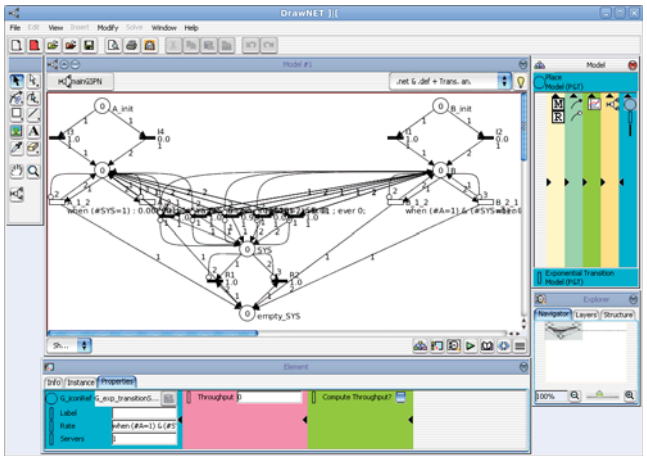

Figure 7: The GSPN visualized by *Draw-Net*.

procedure requested (Sec. 5.1).
8. Given all the intermediate results returned by the *GreatSPN transient solver*, the *Inference Solver* generates the result of the query requested by the user. Such result is passed as standard output to *Draw-Net* for the visualization.

The user does not have to manipulate the GSPN during the process. However the GSPN can be visualized in two ways: **1)** *GCTBN2GSPN* saves the GSPN in a XML file (dashed arc in Fig. 5) which has to be opened by means of *Draw-Net*, as in Fig. 7. **2)** The .net and .def files can be opened by means of the graphical interface of *GreatSPN*.

All the components of the tool, with the exception of the *GreatSPN transient solver*, have been developed exploiting the *DNLib* Java library [2]; its methods can create, modify and retrieve the elements of the models, such as nodes, arcs, measures and their respective attributes. The current version of the tool is a prototype, so it is not yet available for download. It is not included in *GreatSPN*, but we plan to include it in the solver set of *Draw-Net*.

6.1 Inference results

Prediction: let us consider to observe the system working ($SYS = 1$) at time $t_1 = 10^5 h$ and the system failed ($SYS = 2$) at time $t_2 = 2 \cdot 10^5 h$. We compute the probability of component A being working at time $t = 5 \cdot 10^5 h$, conditioned by the observation stream. The result is 0.521855.

Smoothing: we suppose to have observed the system working at $t_1 = 3 \cdot 10^5 h$ and failed at $t_2 = 5 \cdot 10^5 h$. We ask for the probability of component A being failed at $t = 2 \cdot 10^2 h$, conditioned by the above evidence. We obtain 0.308548.

7. CONCLUSIONS

We have presented a tool for the analysis of GCTBN which mix in the same model continuous time delayed variables with standard "immediate" chance variables. The analysis of GCTBN is based on their conversion into GSPN and exploits well established GSPN analysis techniques, in order to perform prediction or smoothing inference. In this way, we take advantage of specialized methodologies for solving the underlying stochastic process (Sec. 2). Exact inference of CTBN model may be impractical if the model size grows, so approximate algorithms are applied. The GSPN based analysis of GCTBN models provide exact results and can be applied to CTBN as well.

8. ACKNOWLEDGMENTS

The authors thank the students Valentina Zogno and Simonetta Busatta for implementing the *GCTBN2GSPN* converter and the *Inference Solver*, respectively.

9. REFERENCES

[1] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN 1.7: Graphical Editor and Analyzer for Timed and Stochastic Petri Nets. *Performance Evaluation*, 24(1&2):47–68, 1995.

[2] D. Codetta-Raiteri, G. Franceschinis, and M. Gribaudo. Defining formalisms and models in the Draw-Net Modeling System. In *International Workshop on Modelling of Objects, Components and Agents*, pages 123–144, Turku, Finland, 2006.

[3] H. Langseth and L. Portinale. Bayesian networks in reliability. *Reliability Engineering and System Safety*, 92(1):92–108, 2007.

[4] A. Miner. Decision diagrams for the exact solution of Markov models. *Proceedings in Applied Mathematics and Mechanics*, 7(1), 2007.

[5] U. Nodelman, C. R. Shelton, and D. Koller. Expectation propagation for continuous time Bayesian networks. In *Conference on Uncertainty in Artificial Intelligence*, pages 431–440, 2005.

[6] L. Portinale and D. Codetta-Raiteri. A GSPN semantics for continuous time Bayesian networks with immediate nodes. Technical Report TR-INF-2009-03-03-UNIPMN, Dipartimento di Informatica, Università del Piemonte Orientale, 2009. http://www.di.unipmn.it.

[7] R. Sahner, K. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems: An Example-based Approach Using the SHARPE Software Package*. Kluwer Academic Publisher, 1996.

[8] P. Weber and L. Jouffe. Reliability modelling with dynamic Bayesian networks. In *Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Washington DC, USA, 2003.