Dipartimento di Informatica
Università del Piemonte Orientale "A. Avogadro"
Viale Teresa Michel 11, 15121 Alessandria
http://www.di.unipmn.it

A GSPN Semantics for Continuous Time Bayesian Networks with Immediate Nodes

D. Codetta Raiteri, L. Portinale (daniele.codetta_raiteri@mfn.unipmn.it, luigi.portinale@mfn.unipmn.it)

TECHNICAL REPORT TR-INF-2009-03-03-UNIPMN
(March 2009)

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

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

2009-02 *The TAAROA Project Specification*, C. Anglano, M. Canonico, M. Guazzone, M. Zola, February 2009.

2009-01 *Knowledge-Free Scheduling Algorithms for Multiple Bag-of-Task Applications on Desktop Grids*, C. Anglano, M. Canonico, February 2009.

2008-09 *Case-based management of exceptions to business processes: an approach exploiting prototypes*, S. Montani, December 2008.

2008-08 *The ShareGrid Portal: an easy way to submit jobs on computational Grids*, C. Anglano, M. Canonico, M. Guazzone, October 2008.

2008-07 *BuzzChecker: Exploiting the Web to Better Understand Society*, M. Furini, S. Montangero, July 2008.

2008-06 *Low-Memory Adaptive Prefix Coding*, T. Gagie, Y. Nekrich, July 2008.

2008-05 *Non deterministic Repairable Fault Trees for computing optimal repair strategy*, M. Beccuti, D. Codetta Raiteri, G. Franceschinis, July 2008.

2008-04 *Reliability and QoS Analysis of the Italian GARR network*, A. Bobbio, R. Terruggia, June 2008.

2008-03 *Mean Field Methods in Performance Analysis*, A. Bobbio, M. Gribaudo, M. Telek, March 2008.

2008-02 *Move-to-Front, Distance Coding, and Inversion Frequencies Revisited*, T. Gagie, G. Manzini, March 2008.

2008-01 *Space-Conscious Data Indexing and Compression in a Streaming Model*, P. Ferragina, T. Gagie, G. Manzini, February 2008.

2007-03 *A fuzzy approach to similarity in Case-Based Reasoning suitable to SQL implementation*, S. Montani, L. Portinale, October 2007.

2007-02 *Space-Conscious Compression*, T. Gagie, G. Manzini, June 2007.

2007-01 *Markov Decision Petri Net and Markov Decision Well-formed Net formalisms* , M. Beccuti, G. Franceschinis, S. Haddad, February 2007.

2006-04 *New Challenges in Network Reliability Analysis*, A. Bobbio, C. Ferraris, R. Terruggia, November 2006.

2006-03 *The Engineering of a Compression Boosting Library: Theory vs Practice in BWT compression*, P. Ferragina, R. Giancarlo, G. Manzini, June 2006.

A *GSPN* Semantics for *CTBN* with Immediate Nodes

Contents

Abstract

In this report we present an extension to Continuous Time Bayesian Networks (*CTBN*) called Generalized Continuous Time Bayesian Networks (*GCTBN*). The formalism allows one to model, in addition to continuous time delayed variables (with exponentially distributed transition rates), also non delayed or "immediate" variables, which act as standard chance nodes in a Bayesian Network. This allows the modeling of processes having both a continuous-time temporal component and an immediate (i.e. non-delayed) component capturing the logical/probabilistic interactions among the model's variables. The usefulness of this kind of model is discussed through an example concerning the reliability of a simple component-based system. A semantic model of *GCTBN*s, based on the formalism of Generalized Stochastic Petri Nets (*GSPN*). is outlined, whose purpose is twofold: to provide a well-defined semantics for *GCTBN*s in terms of the underlying stochastic process, and to provide an actual mean to perform inference (both prediction and smoothing) on *GCTBN*s. The example case study is then used, in order to highlight the exploitation of *GSPN* analysis for posterior probability computation on the *GCTBN* model.

Acronym list:

BN	Bayesian belief Network
CIM	Conditional Intensity Matrix
CPT	Conditional Probability Table
CTBN	Continuous Time Bayesian Network
CTMC	Continuous Time Markov Chain
DBN	Dynamic Bayesian Network
FTA	Fault Tree Analysis
GCTBN	Generalized Continuous Time Bayesian Network
GSPN	Generalized Stochastic Petri Net
MRF	Markov Random Field

1 Introduction

Probabilistic graphical models for reasoning about processes that evolve over time have been deeply investigated; they allow for a factorization of the state space of the process, resulting in better modeling and inference features. Such models are usually based on graph structures, grounded on the theory of Bayesian belief Networks (*BN*) or Markov Random Fields (*MRF*) [1]. When time is taken into account, the main choice concerns whether to consider it as a discrete or a continuous dimension. In the former case, models like Dynamic Bayesian Networks (*DBN*) have become a natural choice [2, 3, 4]; however, there is not always an obvious discrete time unit and, when the process is characterized by several components evolving at different rates, the finer granularity dictates the rules for the discretization [5]. Moreover, if evidence is irregularly spaced in time, all the intervening time slices still have to be dealt with (even if no evidence is available at a given time point). For these reasons, models based on Bayesian Networks, but with a continuous time representation of the temporal evolution have started to be investigated. Continuous Time Bayesian Networks (*CTBN*) have been firstly proposed in [6, 7] and then refined in [8]. Extensions have also been proposed both regarding the use of indirect graph models [9] and the use of Erlang-Coxian distributions on the transition time [10].

The goal of this paper is to propose another kind of extension and, in particular, a generalization of the standard *CTBN* framework, by allowing the presence of nodes which have no explicit temporal evolution; the values of such nodes are, in fact, "immediately" determined, depending on the values of other nodes in the network. This will allow the modeling of processes having both a continuous-time temporal component and a static component capturing the logical/probabilistic aspects determined by specific events occurring in the modeled process.

Actually, such a kind of modeling is possible in the framework of *DBN* where the notion of temporal arc is used to distinguish nodes having a temporal evolution, from nodes just contributing to the "logic" of the model

[5, 11]. However, as we have already noticed, since a *DBN* assumes a discrete dimension of time, this may result problematic for several applications, as well as computationally expensive when absolute time of events are important [10]. Our work is then, at the best of our knowledge, the first attempt trying to mix in the same network, continuous-time delayed nodes with standard chance node.

The possibilities offered by this generalization, 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 specific modules or subsystems, whose behavior has to be modeled for the analysis. For instance, in Fault Tree Analysis (*FTA*), basic events represent the system components with their failure rates, while non-basic events are logical gates identifying modules of the system under examination [12]. In Dynamic Fault Trees [13], logical gates identifying sub-modules, can be combined with dynamic gates, modeling time-dependent dependencies (usually assuming continuous time) among components or sub-modules. Also in this case, it is very important to distinguish, at the modeling level, between delayed and immediate entities (see also [5]). Of course, similar considerations apply in other tasks as well, as in medical diagnosis, financial forecasting, biological process modeling, etc...

The report is organized as follows: in Sec. 2 the *GCTBN* formalism is defined; in Sec. 3 a reliability case study is introduced, together with the corresponding *GCTBN* modeling; in Sec. 4 a semantic model, based on the formalism of Generalized Stochastic Petri Nets (*GSPN*) [16] is defined: first, the rules to translate a *GCTBN* into a *GSPN* are introduced, then, the corresponding model for the case study is discussed; in Sec. 5 we outline how to perform prediction and smoothing inference on a *GCTBN*, by means of transient analysis on the corresponding *GSPN*; the approach is applied on the *GSPN* model of the case study; finally, conclusions and future works are reported in Sec. 7.

2 The generalized *CTBN* model

Following the original paper in [6], a *CTBN* is defined as follows:

Definition 2.1 *Let $X = X_1 \dots X_n$ be a set of discrete variables, a* CTBN *over X consists of two components. The first one is an initial distribution P_X^0 over X (possibly specified as a standard* BN *over X). The second component is a continuous-time transition model specified as (1) a directed graph G whose nodes are $X_1, \dots X_n$ (and with $Pa(X_i)$ denoting the parents of X_i in G); (2) a conditional intensity matrix $Q_{X_i|Pa(X_i)}$ for every $X_i \in X$.*

We can now introduce the notion of a Generalized *CTBN* (*GCTBN*).

Definition 2.2 *Given a set of discrete variables $X = \{X1 \dots X_n\}$ partitioned into the sets D (delayed variables) and I (immediate variables) (i.e. $X = D \cup I$ and $D \cap I = \emptyset$), a Generalized Continuous Time Bayesian Network (*GCTBN*) is a pair $N = \langle P_X^0, G \rangle$ where*

- *P_X^0 is an initial probability distribution over X;*
- *G is a directed graph whose nodes are $X_1 \dots X_n$ (and with $Pa(X_i)$ denoting the parents of X_i in G) such that*
  1. *there is no directed cycle in G composed only by nodes in the set I;*
  2. *for each node $X \in I$ a conditional probability table $P[X|Pa(X)]$ is defined (as in standard* BN*);*
  3. *for each node $Y \in D$ a conditional intensity matrix $Q_{Y|Pa(Y)}$ is defined (as in standard* CTBN*).*

Delayed (or temporal) nodes are, as in case of *CTBN*, nodes representing variables with a continuous time evolution ruled by exponential transition rates, and conditioned by the values of parent variables (that may be either delayed or immediate). Delayed nodes have a Conditional Intensity Matrix (*CIM*) of rates associated with

them. Immediate nodes are introduced in order to capture variables whose evolution is not ruled by transition rates associated with their values, but is conditionally determined, at a given time point, by other variables in the model. Such variables are then treated as usual chance nodes in a *BN* and have a standard Conditional Probability Table (*CPT*) associated with them.

A few words are worth to be spent for the structure of the graph modeling the *GCTBN*. While it is in general possible to have cycles in the graph (as in *CTBN*) due to the temporal nature of some nodes, such cycles cannot be composed only by immediate nodes. Indeed, if this was the case, we would introduce static circular dependencies among model variables.

Finally, it is worth noting that the initial distribution P_X^0 can in general be specified only on a subset of X. In particular, let $R \subset I$ be the set of root nodes (i.e. node with no parent in G) which are immediate, then the initial distribution can be computed as

$$P_X^0 = P_{R \cup D}^0 \prod_{Y_j \in (I-R)} P[Y_j | Pa(Y_j)]$$

In fact, while it is necessary to specify an initial distribution over delayed variables, the distribution on the immediate variables can be determined depending on the values of their parents; of course if an immediate variable is modeled as a root node, a prior probability is needed[1].

3 An illustrative example

We now consider a case study which can be easily modeled in form of *GCTBN*. This is a typical case in the field of reliability analysis, and consists of a small system composed by the main component A and its "warm" spare component B. This means that initially both components are working, but A is active while B is dormant; in case of failure of A, B is activated in order to replace A in its function. We assume that the activation of B occurs with probability $p = 0.99$, so in case of failure of A and the contemporary dormant state of B, the component B may keep its state with probability $1 - p = 0.01$.

The expression "warm" referred to the spare component B indicates that the probability of failure of B is not null while B is dormant, and such value is increased while B is active. So, the spare component B may fail as well, and this can happen before or after the failure of A; if B fails before A, then B can not replace A.

The whole system works if the component A is working, or if A is failed and B is active. The system is considered as failed if A is failed and B is dormant or failed. We suppose that only while the system is failed, the components A and B can undergo repair. As soon as the repair of one of the components is completed, the component re-starts in working state: if A is repaired the system becomes operative again and the repair of B is suspended; if instead B is repaired, this may determine one of these two situations: 1) B may become active with probability $p = 0.99$ and consequently the system becomes operative again and the repair of A is suspended. 2) B may become dormant with probability $1 - p$, so the system is still failed and the repair of B goes on[2].

The time to fail of the components is a random variable ruled by the negative exponential distribution whose parameter is called failure rate and is the inverse of the mean time to failure of the component. In the case of the main component A, the failure rate is λ_A =1.0E-06 h^{-1}. The failure rate of B, λ_B, changes according to its current state: if B is active (A is failed), λ_B is also equal to 1.0E-06 h^{-1}. If instead B is dormant (A is not failed) λ_B is equal to 5.0E-07 h^{-1} (i.e. it is discounted by a dormancy factor $\alpha = 0.5$); In other words, the

[1] Actually, since prior probabilities on immediate root nodes are a special case of *CPT*, then we could also simply write $P_X^0 = P_D^0 \prod_{Y_j \in I} P[Y_j | Pa(Y_j)]$, to emphasize the fact that, for the specification of the temporal evolution of the model, the only initial distribution is on delayed nodes (the other parameters are actually a fixed specification on the network).

[2] Notice that, in principle, different repair policies can be adopted [14]; the one proposed in this example is a kind of subsystem local policy, with the repair of the minimal set of necessary working components.

failure rate of B is reduced with respect to the failure of A, while B is dormant; if instead B is active in order to replace A, the failure rate of B is the same as the failure rate of A.

The time to repair of a component is a random variable, still ruled by the negative exponential distribution having as parameter the repair rate equal to the inverse of the mean time to repair the component. A and B have the same repair rate μ_A =μ_B =0.01 h^{-1}.

By considering the above system specifications, the Continuous Time Markov Chain (*CTMC*) [15] modeling the system's evolution over time is provided in Fig. 1. States are represented in the order A, B, SYS (i.e.

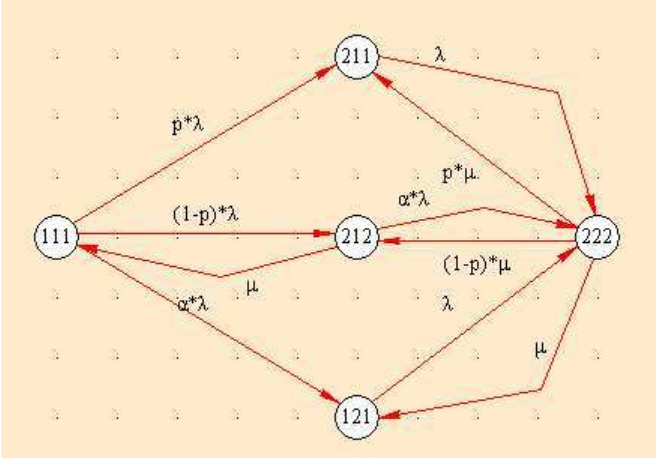

Figure 1: *CTMC* model of the case study.

state 111 means $A = 1, B = 1$ and $SYS = 1$, and so on, while the value 1 stands for the working state and the value 2 stands for the failed state).

3.1 The *GCTBN* model

The case study described above can be represented by the *GCTBN* model in Fig. 2 where the variables A, B, SYS represent the state of the component A, of the component B, and of the whole system respectively. All the variables are binary because each entity can be in the working state or in the failed state (for the component B, the working state comprises both the dormancy and the activation). In particular, we represent the working state with the value 1, and the failed state with the value 2.

The variable A influences the variable B because the failure rate of the component B depends on the state of A, as described above. Both the variables A and B influence the variable SYS because the state of the whole system depends on the state of the components A and B, as described above. The arcs connecting the variable SYS to A and B respectively, concern the repair of the components A and B: only while the system is failed, they can be repaired.

The variables A and B in the *GCTBN* model in Fig. 2 are delayed variables (Sec. 2) and are drawn as double-circled nodes: their value (state of the component) can be equal to 1 or 2, and varies after a random period of time according to the values of the other variables. Both variables implicitly incorporate a *CTMC* composed by two states: 1 (working) and 2 (failed). The initial probability distribution reported in Tab. 1 holds for both variables A and B: the probability to be initially equal to 1 is set to 1, while the probability to be initially equal to 2 is 0. This is due to the assumption that both components are initially supposed to work.

The *CTMC* incorporated by the variables A and B is shown in Fig. 3: the transition from 1 to 2 occurs after a random period of time ruled by the negative exponential distribution according to the failure rate λ. The transition from 2 to 1 is ruled by the same distribution, but according to the repair rate μ.

In the case of A, the current value of the rates λ_A and μ_A depends on the current value of the variable SYS, the only one influencing A. This is shown by the *CIM* reported in Tab. 2 where we can notice that the rate μ_A (the repair rate in the case study) is not null only if the state of SYS is 2. The rate λ_A (the failure rate in the case study) instead, is constant (the state of the system influences only the repair of the component A).

In the case of the variable B, the current value of the rates λ_B and μ_B depends on the current value of the variables A and SYS, as shown by the *CIM* appearing in Tab. 3 where λ_B (the failure rate of the spare component 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 the variable A, the rate μ_B is not null only if the value of SYS is 2. Notice that the combination $A = 1, SYS = 2$ is impossible, so the corresponding entries are not significant.

The variable SYS is immediate and is shown as a circle node in Fig. 2. This means that it changes its value as soon as the variables influencing it, change their value (Sec. 2). The variable SYS depends on both the variables A and B because the state of the system is determined by the state of both the components A and B, as described in the case study. The variable SYS is characterized by the *CPT* appearing in Tab. 4 and expressing the working and failure conditions for the whole system specified in the case study. 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 the spare component B), or the value 2 with probability 0.01 (this implies that B is still dormant).

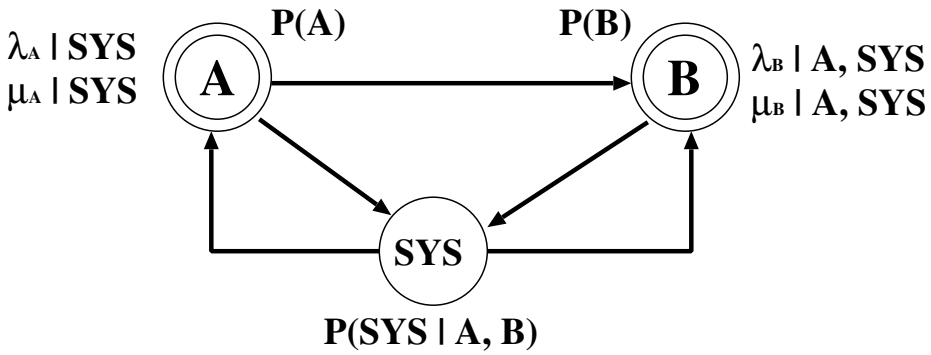

Figure 2: *GCTBN* model of the case study.

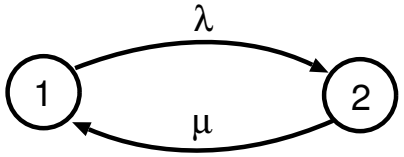

Figure 3: The *CTMC* incorporated in the variables A and B in the *GCTBN* model in Fig. 2.

The introduction of the immediate variable SYS is actually an important modeling facility, since it allows one to directly capture the static (or immediate) interactions between A and B, resulting in a probabilistic choice about the whole system status (i.e. a system working normally in case the replacement of A with B is successful, or a system fault in the opposite case). Without the use of an immediate variable, it is hard to factorize the model using variables A and B. A possibility could be that of modeling 3 values for B, namely $B = 1$ (dormant), $B = 2$ (active), $B = 3$ (failed); this may allow to distinguish a system fault ($A = 2$ and $B = 1$) from a successful switch from A to B ($A = 2$ and $B = 2$), when B is not in a fault state. In such a case a standard *CTBN* could be devised as in figure 4. However, in such a case we cannot model the instantaneous

Value	Prob.
1	1
2	0

Table 1: Initial probability distribution for the values of the variables A and B in the *GCTBN* model in Fig. 2.

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

Table 2: *CIM* for the variable A in the *GCTBN* model in Fig. 2.

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

Table 3: *CIM* for the variable B in the *GCTBN* model in Fig. 2.

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

Table 4: *CPT* for the variable SYS in the *GCTBN* model in Fig. 2.

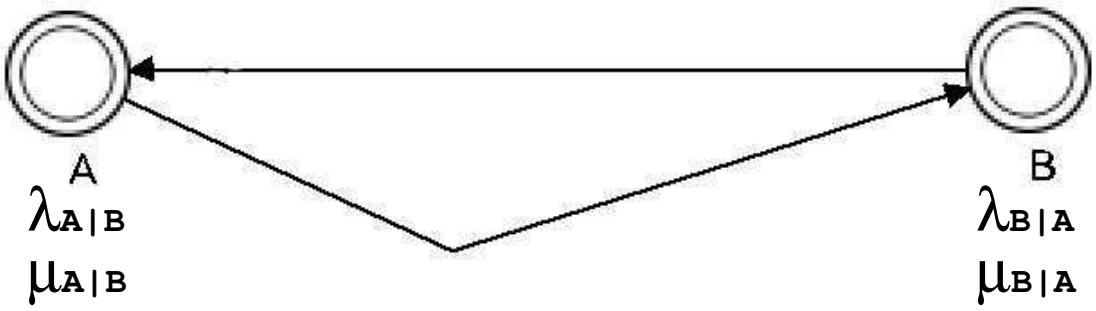

Figure 4: A possible *CTBN* model of the case study.

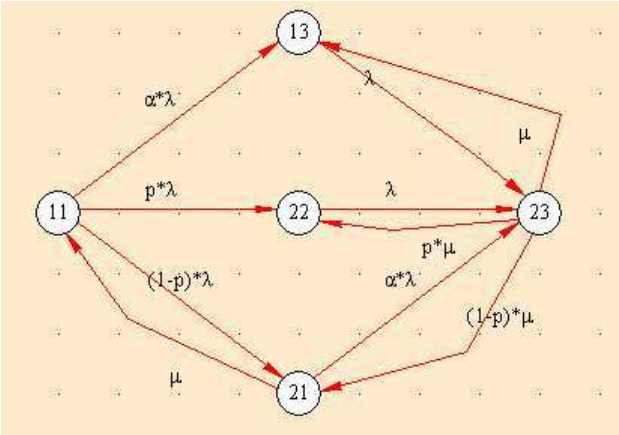

Figure 5: *CTMC* model of the case study with 3 states for B ($dormant, active$ and $faulty$).

state change of both A and B from $A = 1, B = 1$ to $A = 2, B = 2$, since in a continuous time model A and B cannot change their values simultaneously. This is evidenced if we consider the *CTMC* that should result from the modeling of the 3 states of B, that we can see in Fig. 5. Again each state is labeled with the pair of values for A and B respectively; we can notice that, we have to introduce a transition from state 11 (i.e. $A = 1, B = 1$) to state 22 (i.e. $A = 2, B = 2$) with a rate $p\lambda$, to model the possibility of a fault in A associated with a contemporary successful switch on B (on the contrary, with probability $1 - p$, B does not change its state, meaning that the system becomes faulty). Of course, fault states in Fig. 5 are 21 and 23. The *CTMC* of figure 5 is isomorphic to that of figure 1, but it cannot be obtained from the *CTBN* of Fig. 4, because of the above mentioned transition. This suggest, that the use of immediate nodes makes easier to factorize the model, as usually done with *BN*-like formalism.

4 A Petri Net semantics for *GCTBN*

The combination in a single model of entities explicitly evolving over time with entities whose determination is "immediate", is not a novelty; as we have already noticed, in the framework of probabilistic graphical models, *DBN*s provide an example, in case of discrete time. In case of continuous time, a model having such a features can be found in the framework of Petri Nets, namely Generalized Stochastic Petri Nets (*GSPN*) [16]. The formal definition of a *GSPN* can be found in Sec. 4.1. *GSPN*s are stochastic Petri nets, where there are two different sets of transitions, namely *timed* with an exponentially distributed delay, and *immediate* transitions (with no delay), having priority over temporal ones. This means that, in case both an immediate and a temporal transition are enabled, the firing of the former takes precedence over the firing of the latter.

In a *GSPN*, two or more immediate transitions may be enabled at the same time; in this case, *weights* and *priorities* can be used to rule the firing of such transitions. A weight and/or a priority can be assigned to an immediate transition. Using weights, given several immediate transitions enabled to fire, higher is the weight of a transition, higher is its probability to fire. Using priorities, given several immediate transitions enabled to fire, the transition with highest priority fires.

In *GSPN*s we have also two types of arcs: oriented arcs and inhibitor arcs. Oriented arcs are used to connect places to transitions and vice-versa, with the aim of moving tokens when transitions fire. Inhibitor arcs connect a place to a transition with the aim of disabling the transition if the place is not empty. A cardinality (multiplicity) can be associated to an arc; in the case of oriented arcs, the cardinality indicates the number of tokens to be moved on that arc when the transition fires; in the case of inhibitor arcs, the cardinality indicates the number of tokens inside the place, necessary to disable the transition.

The stochastic process associated with a *GSPN* is a homogeneous continuous time semi-Markov process that can be analyzed either by solving the so called *Embedded Markov Chain* or by removing from the set of possible states, the so-called *vanishing states* or *markings* and by analyzing the resulting *CTMC* [16]. Vanishing states are the state (or markings) resulting from the firing of immediate transitions; they can be removed, since the system does not spend time in such states. This removal operation has also the advantage of reducing (often in a significant way) the set of possible states to be analyzed.

The measures returned by the *GSPN* analysis, can be obtained also by means of *GSPN* simulation. The use of simulation instead of analysis, is useful when the number of states (and state transitions) in the *CTMC*, is very high. In this situation, the analysis becomes computationally expensive, or even unfeasible.

The *GreatSPN* tool [23] allows to design, analyze and simulate *GSPN* models. In Sec. 5.3, we resort to such tool in order to compute the intermediate results in the prediction and smoothing inference procedures applied to the *GSPN* derived from the *GCTBN* model of the case study. Further information on the *GSPN* formalism, analysis and simulation, can be found in [16].

A *GCTBN* model can be expressed in terms of a *GSPN*, by means of a set of translation rules (Sec. 4.2). This translation is twofold:

- first, it provides a well-defined semantics for a *GCTBN* model, in terms of the underlying stochastic

process it represents;

- second, it provides an actual mean to perform inference on the *GCTBN* model, by exploiting well-studied analysis techniques for *GSPN*s.

In fact, solution techniques for *GSPN*s have received a lot of attention, especially with respect to the possibility of representing in a compact way the underlying *CTMC* and of solving it efficiently [17, 18, 19]. Once a *GCTBN* has been compiled into a *GSPN*, then such techniques can be employed to compute inference measures on the original *GCTBN* model (see Sec. 5).

There are two main analyses that can be performed with a *GSPN*: *steady state* and *transient analysis*. In the first case, the equilibrium distribution of the states is computed, while in the latter, such a distribution is computed at a given time point. In particular, solving a *GSPN* (for either steady state or transient analysis) can provide the probability distribution of the number of tokens in each place. This information can then be exploited, in order to perform inference on the original *GCTBN* model as it will be shown in Sec. 5.

4.1 *GSPN* formal definition

The *GSPN* formalism is given by the tuple
$GSPN = (P, T, A, \#, r, w, \pi, card)$
where

- P is the set of the places.
- $T = T_i \cup T_t$ is the set of the transitions; it is the union of two sets:
  - T_i is the set of the immediate transitions;
  - T_t is the set of timed transitions.
- A is the set of arcs; it is the union of two sets:
  - $A_d \subseteq (P \times T) \cup (T \times P)$ is the set of the oriented arcs;
  - $A_h \subseteq P \times T$ is the set of the inhibitor arcs.
- $\# : P \to \mathbb{N}$ is the function returning the marking of a place.
- $r : T_t \to \mathbb{R}^+$ is the function returning the firing rate of a timed transition.
- $w : T_i \to \mathbb{R}^+$ is the function returning the weight of an immediate transition.
- $\pi : T_i \to \mathbb{N} - \{0\}$ is the function returning the priority of an immediate transition.
- $card : A \to \mathbb{N} - \{0\}$ is the function returning the cardinality of an arc.
- Given $t \in T$
  - $\bullet_d t = \{(p, t) \in A_d\}$ is the set of the oriented arcs coming from places and directed to t.
  - $t\bullet_d = \{(t, p) \in A_d\}$ is the set of oriented arcs coming from t and directed to places.
  - $\bullet_h t = \{(p, t) \in A_h\}$ is the set of inhibitor arcs coming from places and directed to t.
- Given $t \in T$, t is enabled to fire if all the following conditions hold:
  - $\forall (p, t) \in \bullet_d t, \#(p) \geq card((t, p))$
  - $\forall (p, t) \in \bullet_h t, \#(p) < card((t, p))$

If m' is the marking of $p \in P$ before the firing of t, and m'' is the marking of p after the firing of t, then the effect of the firing of t is the following:

- $\forall (p,t) \in {\bullet_d}t, m" = m' - card((p,t))$
- $\forall (t,p) \in t{\bullet_d}, m" = m' + card((p,t))$

4.2 Translation rules from *GCTBN* to *GSPN*

This section provides some formal rules to derive the *GSPN* equivalent to a *GCTBN* model. First, we introduce first the way to convert the delayed variables, then, the way to convert immediate variables.

4.2.1 Conversion of delayed variables into *GSPN*

A delayed variable X of the *GCTBN* model is converted into several elements of the equivalent *GSPN*:

- the place X' whose marking represents the value of X; we introduce here the function $pl : GCTBN.(I \cup D) \rightarrow GSPN.P$ returning the *GSPN* place corresponding to a variable of the *GCTBN*. In the case of X, $pl(X) = X'$.
- the place X'_init with $\#(X'\_init) = 1$, used to set the initial marking of X';
- the set $T^{init}_{X'}$ of immediate "init" transitions to set the initial marking of X';
- the set $T^{D}_{X'}$ of timed transitions to change the marking of X';
- the set $T^{imm}_{X'}$ of immediate transitions to change the marking of X'.

The set $T^{init}_{X'}$ is created in the following way: given the initial probability distribution P^0_X over X (Sec. 2), for each $x \in X : P^0(x) \neq 0$ the transition $t \in T^{init}_{X'}$ is created such that all the following conditions hold:
$w(t) = P^0(x)$
$\pi(t) = 1$
${\bullet_d}t = \{(X'\_init, t)\}$
$card((X'\_init, t)) = 1$
${\bullet_h}t = \emptyset$
$t{\bullet_d} = \{(t, X')\}$
$card((t, X')) = x$

The set $T^{init}_{X'}$ will contain an immediate transition for each initial value x of X whose probability is not null. Such transitions are all enabled to fire by the presence of one token inside X'_init: the weight of each transition corresponds to the probability of the corresponding initial value x of X. The effect of the firing of a transition t in $T^{init}_{X'}$ consists of removing the token inside X'_init (this disables the firing of the other transitions in $T^{init}_{X'}$) and setting the marking of X' to the initial value x of X corresponding to t. Such setting is done by means of the cardinality of the arc (t, X') equal to x. Once the initial marking of X' is set, the transitions in $T^{init}_{X'}$ can not fire any more because X'_init has become empty.

The set $T^{D}_{X'}$ is created in the following way: for each ordered couple (x_1, x_2) such that $x_1, x_2 \in X$ and $x_1 \neq x_2$, the transition $t \in T^{D}_{X'}$ is created such that all the following conditions hold:
${\bullet_d}t = \{(X', t)\}$
$card((X', t) \in {\bullet_d}t) = x_1$
${\bullet_h}t = \{(X', t)\}$
$card((X', t) \in {\bullet_h}t) = x_1 + 1$
$t{\bullet_d} = \{(t, X')\}$
$card((t, X')) = x_2$

This means that for each possible value transition from x_1 to x_2 by the variable X, a timed transition t is created to model such change of value. Such transition is enabled to fire when the place X' contains exactly x_1 tokens; this is because of the cardinality of the oriented arc $(X', t) \in {}^\bullet_d t$ and of the inhibitor arc $(X', t) \in {}^\bullet_h t$. The effect of the firing of t consists of setting the marking of X' to x_2, by means of the oriented arc (t, X'). The dependency of the transition rates of X on the values of the parent variables, is modeled in the *GSPN* by the marking dependent firing rate of t. This rate changes according to the marking of the places representing the parent variables of X. For each combination of the marking values of such places, the firing rate of t assumes a different value.

We introduce now the function $Pa : GCTBN.(I \cup D) \rightarrow GCTBN.(I \cup D)$ which returns the parent variables of a particular variable of the *GCTBN*. So, $Pa(X)$ returns the variables influencing X; in particular, assuming that such variables are ordered, $Pa_i(X)$ returns the i-th parent variable of X.

The firing rate of t has to be marking dependent:

$$\forall c \in \times_i (Pa_i(X)), r(t)|cond = \begin{cases} q_{x_1,x_2}|c \in Q_{X|Pa(X)} & \text{if } q_{x_1,x_2}|c \neq \infty \\ 0 & \text{if } q_{x_1,x_2}|c = \infty \end{cases}$$

where $cond = \bigwedge_j (\#(pl(Pa_j(X))) = c_j)$
In other words, if c is a particular combination of the values of the parent variables of X, the firing rate of t, $r(t)|cond$, is set to $q_{x_1,x_2}|c$ if such value is finite. $q_{x_1,x_2}|c$ is the entry in the conditional intensity matrix of X at the row x_1 and the column x_2, for the combination c of the values of the parent variables of X. The condition $cond$ for the firing rate $r(t)|cond$ is the translation of the combination c in terms of markings of the places corresponding to the parent variables of X: $cond$ is the conjunction of several sub-conditions, each requiring that the marking of the place corresponding to the i-th parent of X has to be equal to c_i, where c_i is the value of the i-th parent of X in c.

If instead $q_{x_1,x_2}|c$ has an infinite value, this means that the variable X changes its value from x_1 to x_2 instantaneously. In this particular case, a timed transition is not suitable to represent the instantaneous change of value; for this reason, $r(t)|cond$, the firing rate of the timed transition t in this case, is set to 0. In other words, t can not fire if $cond$ holds. Since the change of value from x_1 to x_2 is instantaneous, this has to be modeled by an immediate transition belonging to the set $T^{imm}_{X'}$: each time $q_{x_1,x_2}|c = \infty$, we create the immediate transition $t' \in T^{imm}_{X'}$ such that all the following conditions hold:
${}^\bullet_d t' = \{(X', t'), \bigcup_k (Pa_k(X'), t')\}$
$card((X', t') \in {}^\bullet_d t) = x_1$
$card((Pa_k(X'), t') \in {}^\bullet_d t') = c_k$
${}^\bullet_h t' = \{(X', t')\}$
$card((X', t') \in {}^\bullet_h t) = x_1 + 1$
$t'^\bullet_d = \{(t', X'), \bigcup_k (t', Pa_k(X'))\}$
$card((t', X')) = x_2$
$card((t', Pa_k(X')) \in t'^\bullet_d) = c_k$
$\pi(t') = 1$
$w(t') = 1$

Let us consider the *GCTBN* model in Fig. 2 described in Sec. 3.1. Fig. 6 shows the *GSPN* expressing the delayed variable A in the *GCTBN*. In this *GSPN* the marking of the place A' represents the value of A, and the place A'_init is used to set the initial marking of A', together with the set of immediate transitions $T^{init}_{A'} = \{A\_init\_1, A\_init\_2\}$, with $w(A\_init\_1) = 1$ and $w(A\_init\_2) = 0$, according to the initial probability distribution of A (Tab. 1). The set of timed transitions modifying the marking of A' is $T^D_{A'} = \{A\_1\_2, A\_2\_1\}$. The firing rates of these transitions are marking dependent: the conditions and the values of the rates are reported in Tab. 5.

Fig. 7 shows the *GSPN* expressing the delayed variable B in the *GCTBN* in Fig. 2. Such *GSPN* has the same graph structure of the *GSPN* in Fig. 6 because both A and B can be equal to 1 or 2. In Fig. 7, the marking of the place B' represents the value of B, and the place B'_init is used to set the initial marking of B', together with

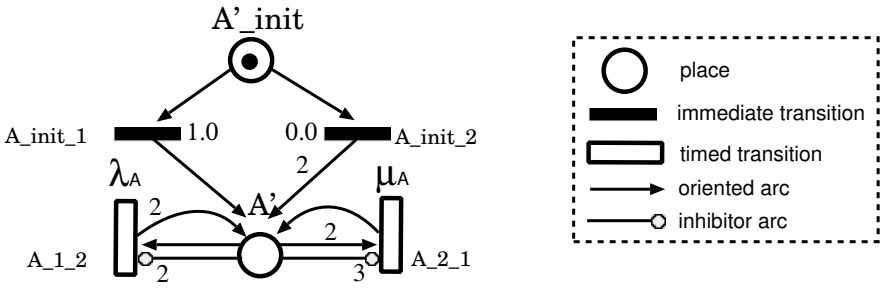

Figure 6: The *GSPN* corresponding to the variable A in the *CTBN* in Fig. 2.

Transition	condition ($cond$)	firing rate (r)
A_1_2	$\#(SYS') = 1$	1.0E-06 h^{-1}
A_1_2	$\#(SYS') = 2$	1.0E-06 h^{-1}
A_2_1	$\#(SYS') = 1$	0 h^{-1}
A_2_1	$\#(SYS') = 2$	0.01 h^{-1}

Table 5: Firing rates of the timed transitions in Fig. 6.

$T^{init}_{B'} = \{B\_init\_1, B\_init\_2\}$, with $w(B\_init\_1) = 1$ and $w(B\_init\_2) = 0$. The set of timed transitions modifying the marking of B' is $T^{D}_{B'} = \{B\_1\_2, B\_2\_1\}$. The firing rates of these transitions are marking dependent (Tab. 6).

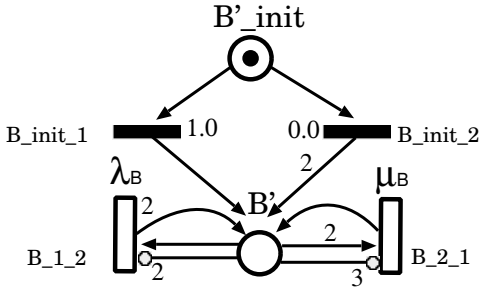

Figure 7: The *GSPN* corresponding to the variable B in the *CTBN* in Fig. 2.

4.2.2 Conversion of immediate variables into *GSPN*

An immediate variable Y of the *GCTBN* model is converted into several elements of the equivalent *GSPN*:

- the place Y' whose marking corresponds to the values of Y ($pl(Y) = Y'$);
- the set $T^{I}_{Y'}$ of the immediate transitions used to initialize the value of the marking of Y' and to re-set it whenever the marking of one of the places corresponding to the parent variables of Y' changes;
- the place $empty\_Y'$ used when the marking of the place Y' has to be re-set;
- the set $T^{reset}_{Y'}$ of immediate transitions used to remove the tokens inside Y' when the marking of this has to be re-set.

Transition	condition ($cond$)	firing rate (r)
B_1_2	$\#(A') = 1 \wedge \#(SYS') = 1$	5.0E-07 h^{-1}
B_1_2	$\#(A') = 1 \wedge \#(SYS') = 2$	5.0E-07 h^{-1}
B_1_2	$\#(A') = 2 \wedge \#(SYS') = 1$	1.0E-06 h^{-1}
B_1_2	$\#(A') = 2 \wedge \#(SYS') = 2$	5.0E-07 h^{-1}
B_2_1	$\#(A') = 1 \wedge \#(SYS') = 1$	0 h^{-1}
B_2_1	$\#(A') = 1 \wedge \#(SYS') = 2$	0 h^{-1}
B_2_1	$\#(A') = 2 \wedge \#(SYS') = 1$	0 h^{-1}
B_2_1	$\#(A') = 2 \wedge \#(SYS') = 2$	0.01 h^{-1}

Table 6: Firing rates of the timed transitions in Fig. 7.

The set $T^I_{Y'}$ is created in the following way: each $e \in \times_i Pa_i(Y) \times Y$ corresponds to an entry of the *CPT* of Y, while $P(e)$ is the probability of such entry. Each e is a combination of $n = \sum_i |Pa_i(Y)| + |Y|$ values, where the first $n-1$ values are referred to the parent variables of Y, while the n -th one is referred to Y. For each e such that $P(e) \neq 0$, an immediate transition $t \in T^I_{Y'}$ is created such that the following conditions hold:
$w(t) = P(e)$
$\pi(t) = 1$
$\bullet_d t = \bigcup_i (pl(Pa_i(Y)), t)$
$\forall (pl(Pa_i(Y)), t) \in \bullet_d t, card(pl(Pa_i(Y), t)) = e_i$
$t\bullet = \bigcup_i (t, pl(Pa_i(Y))) \cup (t, Y')$
$\forall (t, pl(Pa_i(Y))) \in t\bullet_d, card(t, pl(Pa_i(Y))) = e_i$
$card(t, Y') = e_n$
$\bullet_h t = \bigcup_i (pl(Pa_i(Y)), t) \cup (Y', t)$
$\forall (pl(Pa_i(Y), t) \in \bullet_h t, card(pl(Pa_i(Y), t)) = e_i + 1$
$card(Y', t) = 1$
In other words, for each entry e of the *CPT* of Y, the immediate transition t is created. Its effect is is setting the value of the marking of Y' to the to value of Y in e. The transition t is enabled to fire when Y' is empty and the markings of the places corresponding to the parent variables of Y assume exactly the values of such variables in e. The weight of t is set to the probability of e in the *CPT*. In this way, we model in the *GSPN* the role of each entry e in the *CPT* of Y.

In the *GCTBN*, Y directly depends on $Pa(Y)$. However, if the set $Pa(Y)$ contains immediate variables, then the change of such variables is ruled by the change of their parents, eventually being delayed variables. So, what really determines a change in Y is the set of the "Closest" Delayed Ancestors (CDA) of Y. Let $CDA(Y) = \{X : X \in D \wedge \exists$ a path from X to Y with no intermediate immediate node$\}$.

Therefore, each time that the marking of a place in $pl(CDA(Y))$ changes, first, the place Y' has to become empty, then its marking has to be set according to the current marking of the places in $pl(Pa(Y))$. To set place Y' empty, we introduce the place Y'_empty and the transition set $T^{reset}_{Y'}$; the second step is done by the transition set $T^I_{Y'}$ as explained above.

Every time that the marking of a place in $pl(CDA(Y))$ varies, one token has to appear in the place Y'_empty in order to indicate that the place Y' must become empty. Again, let the function $CDA_i(Y)$ returns the i-th closest delayed ancestor of Y; for each $CDA_i(Y)$ and for each $t \in T^D_{pl(CDA_i(Y))}$, an oriented arc (t, Y'_empty) is added to $t\bullet$. In other words, for each transition modifying the marking of a place corresponding to a CDA variable of Y, we create an arc from such transition to the place Y'_empty. Therefore each time that such transition fires, one token appears in Y'_empty enabling the firing of one of the immediate transitions in the set $T^{reset}_{Y'}$. $T^{reset}_{Y'}$ is created in the following way: for each $y \in Y$, the immediate transition $t \in T^{reset}_{Y'}$ is created such that all the following conditions hold:

$w(t) = 1$
$\pi(t) = 2$
$\bullet_d t = \{(empty\_Y', t), (Y', t)\}$
$card(empty\_Y', t) = 1$
$card(Y', t) \in \bullet_d t = x$
$\bullet_h t = (Y', t)$
$card(Y', t) \in \bullet_h t = x + 1$

For each possible marking of Y', one transition in the set $T^{reset}_{Y'}$ is enabled to fire with the effect of removing any token inside Y' and removing the only token inside Y'_empty. When this occurs, some of the transitions in the set $T^{I}_{Y'}$ are enabled to fire in order to set the new marking of Y' according to the current value of place set $pl(Pa(Y))$. The higher priority (π) of the transitions in $T^{reset}_{Y'}$ determines the correct order of firing of all immediate transitions.

Let us consider the *GCTBN* model in Fig. 2 described in Sec. 3.1. Fig. 8 shows the equivalent *GSPN* containing the subnets representing the variables A and B (described in Sec. 4.2.1) and the subnet representing the variable SYS. In particular, in the third subnet, the marking of the place SYS' represents the value of the variable SYS whose set of CDA variables is $CDA(SYS) = \{A, B\}$. In this example, $CDA(SYS) = Pa(SYS)$. The set of places corresponding to $CDA(SYS)$ is $pl(CDA(SYS)) = \{A', B'\}$. The set of immediate transitions setting the marking of the place SYS' according to the marking of the places in $pl(Pa(SYS))$ is $T^{I}_{SYS'} = \{set\_SYS\_1, set\_SYS\_2, set\_SYS\_3, set\_SYS\_4, set\_SYS\_5\}$. The weight of each of these transitions is reported in Fig. 2, close to the transition. The place $empty\_SYS'$ is used to remove any token inside SYS', together with the immediate transitions in the set $T^{reset}_{SYS'} = \{reset\_SYS\_1, reset\_SYS\_2\}$, with $\pi(reset\_SYS\_1) = \pi(reset\_SYS\_2) = 2$. Since $pl(CDA(SYS)) = \{A', B'\}$, the place $empty\_SYS'$ is reached by an oriented arc coming from each of the transitions in the set $T^{D}_{A'} \cup T^{D}_{B'}$= $\{A\_1\_2, A\_2\_1\} \cup \{B\_1\_2, B\_2\_1\}$= $\{A\_1\_2, A\_2\_1, B\_1\_2, B\_2\_1\}$.

The description of the *GSPN* in Fig. 8 is completed in Sec. 4.3.

4.3 The *GSPN* model for the case study

According to the conversion rules described in Sec. 4.2, the *GCTBN* of the case study in Fig. 2 can be converted into the *GSPN* model shown in Fig. 8 where the places A', B' and SYS' correspond to the variables in the *GCTBN* model. The value of a *GCTBN* variable is mapped into the marking (number of tokens) of the corresponding place in the *GSPN*. Let us consider the place B' in the *GSPN*: the marking of the place B' can be equal to 1 or 2, the same values that the variable B in the *GCTBN* can assume. B is a delayed variable and its initialization is modeled in the *GSPN* by the immediate transitions B_init_1 and B_init_2 called *"init" transitions*. Such transitions are both initially enabled to fire with the effect of setting the initial marking of the place B' to 1 or 2 respectively. The probability of these transitions to fire corresponds to the initial probability distribution of the variable B (Tab. 1)[3].

The variation of the marking of the place B' is determined by the timed transitions B_1_2 and B_2_1. The transition B_1_2 is enabled to fire when the place B' contains one token; the effect of its firing is setting the marking of B' to 2. The transition B_2_1 instead, can fire when the marking of the place B' is equal to 2, and turns it to 1. The dependency of the transition rate of a variable on the values of the other variables in the *GCTBN* model, becomes in the *GSPN* model, the dependency of the firing rate of a timed transition on the markings of the other places. For instance, in the *GCTBN* model the variable B depends on A and SYS; let us consider λ_B which is the transition rate of B from 1 to 2 depending on the values of the variables A and SYS (Tab. 3). In the *GSPN* model, λ_B becomes the firing rate of the timed transition B_1_2 whose value depends on the marking of the places A' and SYS', and assumes the same values reported in Tab. 3. The firing rate of

[3]In case of transitions having weights (or rates) equal to 0, they can be omitted from the model; in Fig. 8 they are shown, for the sake of generality.

the timed transition B_2_1 instead, will correspond to the rate μ_B reported in Tab. 3, still depending on the marking of the places A' and SYS'.

The initialization of the marking of the place A' is modeled by the immediate init transitions B_init_1 and B_init_2, while the variation of its marking is modeled in a similar way by the timed transitions A_1_2 and A_2_1, but in this case their firing rate will depend only on the marking of the place SYS', because in the *GCTBN* model the variable A depends only on the variable SYS. Such variable is immediate in the *GCTBN* and depends on A and B. Therefore in the *GSPN*, each time the marking of the place A' or of the place B' is modified, the marking of SYS' has to be immediately updated: each time the transition A_1_2, A_2_1, B_1_2 or B_2_1 fires, one token appears in the place $empty\_SYS'$; this determines the firing of the immediate transition $reset\_SYS\_1$ or $reset\_SYS\_2$, with the effect of removing any token inside the place SYS'. At this point, the marking of such place has to be set according to the current marking of the places A' and B'. This is done by one of the immediate transitions set_SYS_1, set_SYS_2, set_SYS_3, set_SYS_4, set_SYS_5. Each of them corresponds to one entry having not null probability in the *CPT* of the variable SYS in the *GCTBN* model (Tab. 4). Each of such immediate transitions has the same probability (weight) and the same effect on the marking of the place SYS', of the corresponding entry in the *CPT*.

5 Inference

Standard inference tasks in a temporal probabilistic model are *prediction* and *smoothing* [4]. Let X_t be a set of variables at time t and $y_{a:b}$ any stream of observations from time point a to time point b (i.e. a set of instantiated variables). *Prediction* is the task of computing $P(X_{t+h}|y_{1:t})$ for some horizon $h > 0$, i.e. predicting a future state taking into consideration the observation up to now (a special case occurs when $h = 0$ and is called *Filtering* or *Monitoring*). *Smoothing* is the task of computing $P(X_{t-l}|y_{1:t})$ for some $l < t$, i.e. estimating what happened l time points in the past, given all the evidence (observations) up to now.

Such tasks can be accomplished, depending on the model adopted, by inference procedures usually based on specific adaptation of standard algorithms for *BN*s. For instance, in *DBN* models, both exact algorithms based on junction tree [4] as well as approximate algorithms exploiting the net structure [20] or based on stochastic simulation can be employed. In case of *CTBN* , exact inference may often be impractical, so approximations through message-passing algorithms on cluster graphs [7, 8], or through sampling [21, 9] have been proposed.

In the present work, we take advantage of the correspondence between *GCTBN* and *GSPN*, in order to propose inference algorithms (both for prediction and smoothing), based on *GSPN* solution algorithms.

Computing the probability of a given variable assignment $X = x_i$ at time t, will correspond to compute the probability of having i tokens in the place modeling X at time t.

5.1 Prediction Inference

The task of prediction consists in computing 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 evidence e_{t_j} consists of a (possibly different) set of instantiated variables.

Prediction can then be implemented by repeatedly solving for transient the corresponding *GSPN* at the different times corresponding to the observations and the query. Of course, any observation will condition the evolution of the model, so the suitable conditioning operations must be performed before a new *GSPN* resolution.

Let $Pr\{E\}$ be the probability of the event E, computed from the resulting distribution of a *GSPN* transient (for instance given a variable X, $P(X = x) = Pr\{\#X' = x\}$ where $\#X'$ is the number of tokens in place X' corresponding to X). The pseudo-code for the prediction procedure is shown in Fig. 9.

Notice that, in the special case of *filtering*, the last evidence would be available at the query time (i.e. $t = t_k$ in Fig. 9); in such a case, the update of the transition weights (last statement in the `for` cycle) is not necessary,

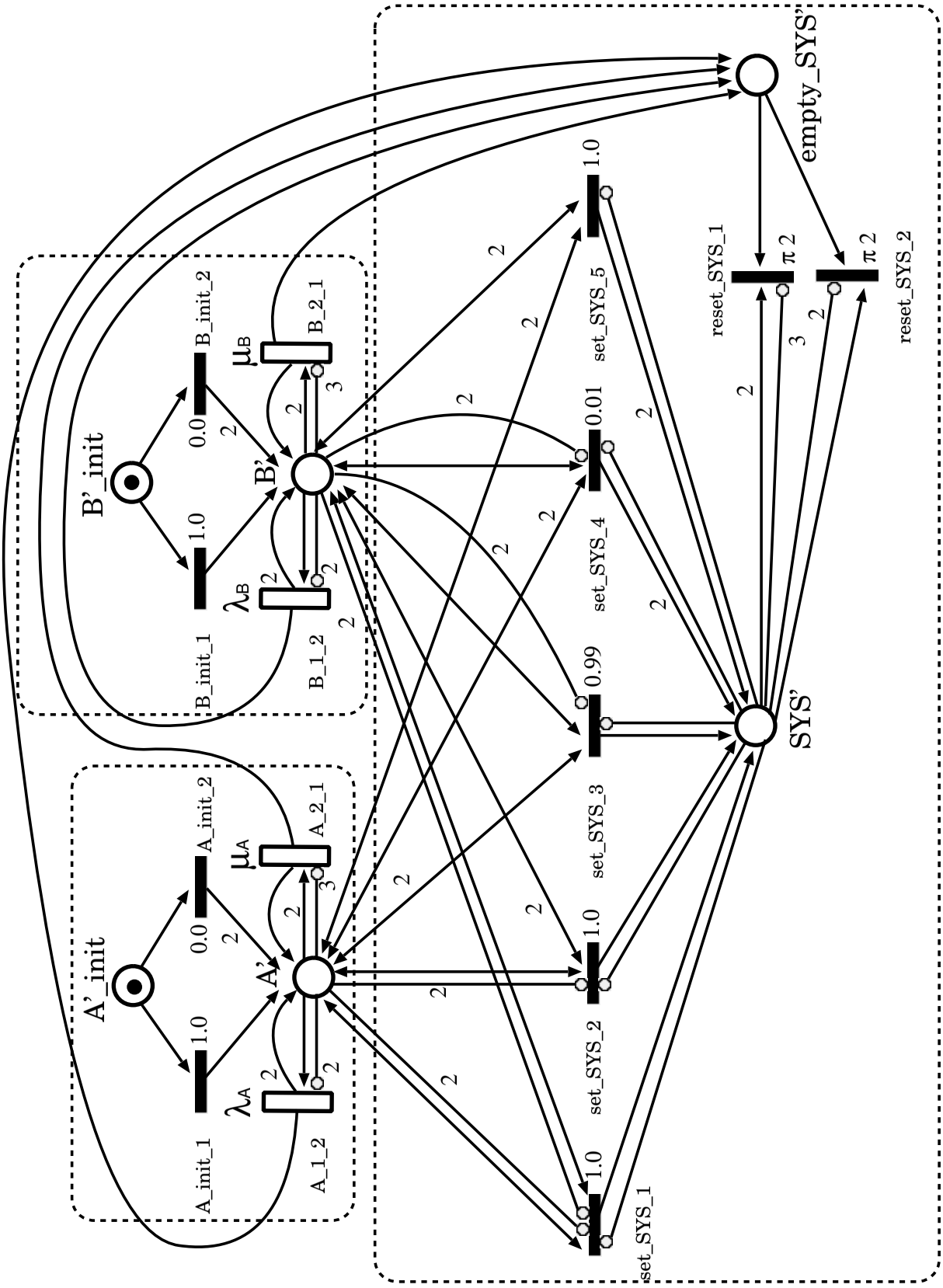

Figure 8: *GSPN* model obtained from the *GCTBN* in Fig. 2.

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 *GSPN* 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 *GSPN* transient at time $(t - t_k)$;
- compute from transient, $r = Pr\{Q\}$;
- output r;

Figure 9: The prediction inference procedure.

as well as the final transient solution. The procedure would then simply output $Pr\{Q|e_t\}$ computed from the last transient analysis.

In case there is evidence available at time $t_0 = 0$, then if the evidence is on variables $X \in D \cup R$, then it is incorporated into their "init" distribution; if the evidence is on variables $X \in I - R$, then the "init" of the other variables are updated by solving the transient at time $t_0 = 0$.

5.2 Smoothing Inference

The smoothing task needs a little more attention than prediction; it consists in computing 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 $t < t_1 < \ldots t_k$. As for prediction, the idea is to perform smoothing by repeatedly solving the corresponding *GSPN* at the different times corresponding to the observations and the query; however, the problem is how to condition on variables observed at a time instant that follows the current one. The idea is then to try to reformulate the problem in such a way that it can be reduced to a prediction-like task. The approach is then based on the application of Bayes rule as follows:

$P(Q_t | e_{t_1}, \ldots e_{t_k}) = \alpha P(Q_t) P(e_{t_1}, \ldots e_{t_k} | Q_t)$
$= \alpha P(Q_t) P(e_{t_1} | Q_t) \ldots P(e_{t_k} | e_{t_1}, \ldots e_{t_{k-1}}, Q_t)$

In this way, every factor in the above formula is conditioned on the past and can be implemented as in prediction. However, the computation of the normalization factor α, requires that a separate computation must be performed for every possible assignment of the query Q. The interesting point is that such computations are independent, so they can be possibly performed in parallel[4]. Once the computation has been performed for every query assignment, than results can be normalized to get the actual required probability values.

Again, let $Pr\{E\}$ be the probability of event E resulting from the current transient analysis; the pseudo-code for the smoothing procedure is shown in Fig. 10. The `normalize` operator just divides any entry of the vector A by the sum of all the entries, in order to provide the final probability vector of the query.

[4] An alternative can be to directly compute the denominator of the Bayes formula (i.e. the probability of the evidence stream); however, this requires a larger number of transient solutions if the length of observation stream is greater than the number N of assignments of Q (i.e.if $k > N$), as is usually the case.

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 ≤ i ≤ N) of Q;
- A: array[N];
for i = 1 to N {
     //possibly in parallel
     A[i]=SMOOTH(q_i); }
- output normalize(A);
}
```

Procedure SMOOTH**(q)**

```
{
- t_0 = t;
- solve the GSPN 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 ∈ D ∪ R;
     - update the weights of the immediate init
        transitions of X_j according to p_{i-1}(j);
     - solve the GSPN transient at time (t_i - t_{i-1});
     - compute from transient, p_i(e) = Pr{e_{t_i}}
     - r = r p_i(e);
     - ev = e_{t_i}; }
- output r;
}
```

Figure 10: The smoothing inference procedure.

5.3 Reliability Inference in the case study

Consider again the case study of Fig. 2. We can ask for different reliability measures, by exploiting the conversion of the *GCTBN* into a *GSPN*; for example, we can easily compute the reliability of the whole system, by asking for the probability $P(SYS = 2)$ over time. This reduces to compute the probability of having 2 tokens into place SYS', on the corresponding *GSPN* (i.e. $Pr\{\#SYS' = 2\}$). This is done, by solving the transient at the required time instants. Results for our example are reported in Tab. 7 (column Unreliability). Since the modeled system is repairable, it makes sense to ask for the steady state distribution, in order

Time (h)	Unreliability
100000	$8.0E-06$
200000	$1.4E-05$
300000	$1.9E-05$
400000	$2.3E-05$
500000	$2.7E-05$

Table 7: Unreliability results from the case study.

to understand whether the system is reliable in the long run. By solving the *GSPN* for steady state, we can indeed compute that the probabilities of components A and B being faulty in the long run are about 50% (in particular, 0.496681 and 0.500026 respectively), while the probability of the whole system being faulty (i.e. $Pr\{\#SYS' = 2\} = P(SYS = 2)$) is 0.000051, meaning the a good reliability is assured by the use of the spare and by the repair strategy.

Concerning prediction, let us consider to observe the system working at time $t_1 = 100000\ h$ ($SYS = 1$ at $t_1 = 10^5\ h$) and the system failed at time $t_2 = 200000\ h$ (i.e. $SYS = 2$ at $t = 2 \cdot 10^5\ h$). By considering the procedure outlined in Fig. 9 we can compute the probability of component A being working at time $t = 5 \cdot 10^5\ h$ (conditioned by the observation stream) as follows: (1) we solve the transient at $t = 10^5\ h$ and we compute the probabilities of A and B, conditioned by the observation $SYS = 1$; (2) we use the above computed probabilities as the new init probabilities for the places A' and B' of the *GSPN*; (3) we solve the transient for another time interval $t = 10^5\ h$ and we compute the probabilities of A and B, conditioned by the observation $SYS = 2$; (4) we use the above computed probabilities as the new init probabilities for the places A' and B' of the *GSPN*; (5) we solve the transient for a time interval $t = 3 \cdot 10^5\ h$ and we finally compute the probability of the query of A.

Tab. 8 shows the computed value during the above process. The last row shows the required results.

Time (h)	$P(A = 1\|ev)$	$P(A = 2\|ev)$	$P(B = 1\|ev)$	$P(B = 2\|ev)$
100000	0.909228	0.090772	0.952445	0.047555
200000	0	1	0.071429	0.928571
500000	**0.521855**	**0.478145**	-	-

Table 8: Probabilities for prediction inference in the case study (ev is the current accumulated evidence).

Concerning smoothing inference, let us suppose to have observed the system working at time $t_1 = 300000\ h$ and failed at time $t_2 = 500000\ h$. We ask for the probability of component A at time $t = 200000\ h$, conditioned by the above evidence. By considering the procedure outlined in Fig. 10 we can compute the required probabilities as follows: (1) we first consider the case $A = 1$; (2) we solve the transient at $t = 2 \cdot 10^5\ h$ and we compute $r1 = P(A = 1)$; (3) we condition A and B on $A = 1$ and we determine the new init probabilities for A and B; (4) we solve the transient for $t = 10^5\ h$ (to reach time $3 \cdot 10^5\ h$) and we compute $r2 = P(SYS = 1)$; we also condition A and B on $SYS = 1$ and we use such values as new init probabilities for places A' and B';

(5) we solve the transient for $t = 2 \cdot 10^5$ h (to reach time $5 \cdot 10^5$ h) and we compute $r3 = P(SYS = 2)$; (6) we compute the un-normalized probability of $A = 1$ as $p1 = r1 * r2 * r3$; By performing the above steps also for the case $A = 2$ we can similarly compute the un-normalized probability of $A = 2$, namely $p2$. A simple normalization over $p1$ and $p2$ will then produce the required results. Tab. 9 shows the computed value during the above process (partial results $r1, r2, r3$ are shown in bold).

$P(A = 1)$ at $t = 2 \cdot 10^5$ = **0.833086**

Time (h)	$P(A = 1\|ev)$	$P(B = 1\|ev)$	$P(SYS = 1\|ev)$	$P(SYS = 2\|ev)$
200000	1	0.891238	-	-
300000	0.913981	0.854028	**0.999988**	-
500000	-	-	-	**0.000022**

p1=0.0000183277

$P(A = 2)$ at $t = 2 \cdot 10^5$ = **0.166914**

Time (h)	$P(A = 1\|ev)$	$P(B = 1\|ev)$	$P(SYS = 1\|ev)$	$P(SYS = 2\|ev)$
200000	0	0.999922	-	-
300000	0.056648	0.952429	**0.999950**	-
500000	-	-	-	**0.000049**

p1=0.0000081784

$P(A = 1\|ev)$	$\frac{p1}{p1+p2}$=**0.691452**
$P(A = 2\|ev)$	$\frac{p2}{p1+p2}$=**0.308548**

Table 9: Probabilities for smoothing inference in the case study (ev is the current accumulated evidence).

5.4 Verifying inference results on the *DBN* model

In Sec. 5, we proposed a couple of procedures to compute prediction and smoothing measures respectively, on the *GSPN* corresponding to the *GCTBN* model of the system. In Sec. 5.3, such algorithms were applied on the *GSPN* model in Fig. 8 derived from the *GCTBN* in Fig. 2, with the aim of computing the following measures:

- Prediction: the state probability of the component A at time 500000 h, given the observation that the system is working at time t_1=100000 h and is failed at time t_2=200000 h.
- Smoothing: the state probability of the component A at time 200000 h given the observation that the system is working at time t_1=300000 h and is failed at time t_2=500000 h.

The results are reported in Tab. 8 and in Tab. 9 respectively. In order to verify their correctness, in this section, we compare them with the results obtained on the Dynamic Bayesian Network (*DBN*) [2, 3, 4] in Fig. 11 modeling the same system described in Sec. 3.

Given a set of time-dependent state variables $X_1 \dots X_n$ and given a *BN* N defined on such variables, a *DBN* is essentially a replication of N over two time slices $t - \Delta$ and t (being Δ the so called discretization step), with the addition of a set of arcs representing the transition model. Arcs can connect nodes at different slices or nodes at the same slice. If $X_i^{t-\Delta}$ is the copy of variable X_i at time slice $t - \Delta$, the *CPT* of $X_i^{t-\Delta}$ specifies the distribution $P[X_i^{t-\Delta}|Y^{t-\Delta}]$ where $Y^{t-\Delta}$ is any set of variables at slice $t - \Delta$ different than X_i (possibly the empty set). If X_i^t is the copy of variable X_i at time slice t, the *CPT* of X_i^t specifies the distribution $P[X_i^t|X_i^{t-\Delta}, Y^{t-\Delta}, Y^t]$ where $Y^{t-\Delta}$ is any set of variables at slice $t - \Delta$ different than $X_i^{t-\Delta}$ (possibly the empty set), and Y^t is any set of variables at slice t different than X_i^t (possibly the empty set).

Value	Prob.
0	1
1	0

Table 10: Initial probability distribution for the values of the variables A and B in the *DBN* model in Fig. 11.

A	B	S	Prob.
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0.99
1	0	1	0.01
1	1	0	0
1	1	1	1

Table 11: *CPT* for the variable S in the *DBN* model in Fig. 11.

In the *DBN* depicted in Fig. 11, the variables A, B, S represent the state of the component A, the component B and the whole system respectively, at the time slice $t - \Delta$. The variables $A\#$, $B\#$, $S\#$ have the same role, but at the time slice t. In the *DBN* of the case study, all the variables are binary: the value 0 is used to model the working state, while the value 1 is used to model the failed state[5].

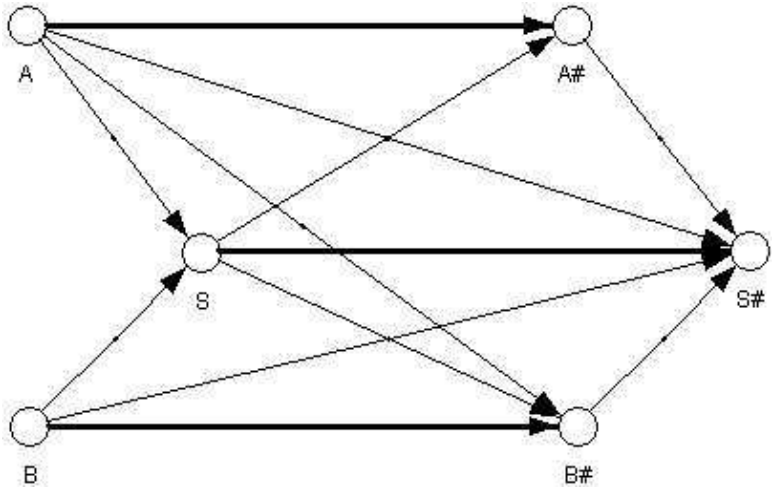

Figure 11: The *DBN* model of the case study.

At the time slice $t - \Delta$, the variables A and B have no parent variables, so their *CPT* contains the initial probability distribution of their values: since all the components are assumed to be working at the initial time, for both A and B the initial probability to be equal to 0 is null, and the initial probability to be equal to 1 is 1 (Tab. 10). The variable S at the time slice $t - \Delta$ depends on A and B: the *CPT* of S expresses the working and

[5]This differs from the *GCTBN* model (Sec 3.1) where the value 1 stands for the working state, while the value 2 stands for the failed state.

failure conditions about the system, and is reported in Tab. 11[6].

At the time slice t, the variable $A\#$ represents the temporal evolution of the component A due to its failure or repair: the state transition from working to failed (failure) is modeled by the dependency of $A\#$ on its copy A at the time slice $t - \Delta$; the state transition from failed to working (repair) is represented by the dependency of $A\#$ on both A and S because the repair of the component A is allowed only while the system (S) is failed. Since the time to failure or repair is random and is ruled by the negative exponential distribution, in the *CPT* of the variable $A\#$ reported in Tab. 12, several entry probabilities correspond to the failure or repair probability at time Δ according to such distribution and the failure or repair rate (λ_A or μ_A).

The variable $B\#$ at the time slice t represents the temporal evolution of the component B. The failure rate of such component changes according to the state of the component A and the eventual activation of B, while the repair of B is allowed only while the system is failed. Therefore $B\#$ depends on A, B and S at the time slice $t - \Delta$. The *CPT* of $B\#$ is reported in Tab. 13 where the value of several entry probabilities are equal to the failure and repair probability at time Δ according to λ_B and μ_B respectively.

Finally, the variable $S\#$ at the time slice t represents the evolution of the whole system according to the evolution of the components A and B. So, $S\#$ depends on S, A, B, $A\#$ and $B\#$. The *CPT* of $S\#$ appears in Tab. 14 and represents the working and failure conditions of the system according to the state of A and B, including the possibility that the component B is not activated in case of failure of A. This *CPT* is composed by 48 entries, but we avoid to report those entries expressing impossible combinations of the values of S, A, B, $A\#$ and $B\#$.

The inference of the *DBN* model of the case study has been performed by means of the RADYBAN tool [5]. The system unreliability has been computed by performing the prediction task assuming no observations and querying the variable $S\#$ at several mission times; the results are reported in Tab. 15 where the probability that $S\#$ is equal to 1 provides the system unreliability. These values are very similar to the unreliability values computed on the *GSPN* model in Fig. 8 and reported in Tab. 7[7]. This validates the correctness of the conversion of the *GCTBN* model of the system into *GSPN*.

Then, we have computed the inference measures mentioned at the begin of this section. First, the prediction task assuming to observe $S = 0$ at time t_1=100000 h and $S = 1$ at time t_2=200000 h, has been performed on the *DBN* model in Fig. 11, querying the node $A\#$. The results appear in Tab. 16: we can observe that the probabilities computed at time 500000 h are very close to the values obtained for the same measure on the *GSPN* model in Fig. 8, reported in Tab. 8 and computed according to the prediction inference procedure specified in Fig. 9. In this way, we verify the correctness of such procedure.

Finally, we have performed the smoothing task on the *DBN* model assuming the observation that $S = 0$ at time t_1=300000 h and $S = 1$ at time t_2=500000 h. We have queried again the node $A\#$ at several mission times obtaining the results appearing in Tab.17. We can notice that the probabilities at time 200000 h are very similar to those obtained on the *GSPN* model by means of the smoothing inference procedure specified in Fig. 10. This confirms the correctness of such procedure.

6 Software tool architecture

We plan to develop a software tool for the design and the analysis of *GCTBN* models, based on the approach described in this paper. The architecture of the tool is depicted in Fig. 12 where the model is built and evaluated following these steps:

1. the user designs the the *GCTBN* model and specifies a query to be evaluated on the model. This is done by means of the graphical tool *Draw-Net* [22] (Fig. 13). The *GCTBN* model is passed to the translator from

[6]This *CPT* corresponds to the *CPT* reported in Tab. 4 and concerning the variable $SY\,S$ in the *GCTBN* model in Fig. 2, but with the difference that the working state is represented as 1, and the failed state as 2.

[7]Because of the time discretization in *DBN* models, the results obtained on such models suffer from some approximation. The approximation degree is proportional to the time step Δ used to discretize the mission time

A	S	$A\#$	Prob.
0	0	0	0.999999
0	0	1	0.000001
0	1	0	-
0	1	1	-
1	0	0	0
1	0	1	1
1	1	0	0.01
1	1	1	0.99

Table 12: *CPT* for the variable $A\#$ in the *DBN* model in Fig. 11 (the probabilities of the impossible entries are omitted because they are not significant).

A	B	S	$B\#$	Prob.
0	0	0	0	0.9999995
0	0	0	1	0.0000005
0	0	1	0	-
0	0	1	1	-
0	1	0	0	0
0	1	0	1	1
0	1	1	0	-
0	1	1	1	-
1	0	0	0	0.999999
1	0	0	1	0.000001
1	0	1	0	0.9999995
1	0	1	0	0.0000005
1	1	0	0	-
1	1	0	0	-
1	1	1	0	0.01
1	1	1	0	0.99

Table 13: *CPT* for the variable $B\#$ in the *DBN* model in Fig. 11 (the probabilities of the impossible entries are omitted because they are not significant).

A	B	S	$A\#$	$B\#$	$S\#$	Prob.
0	0	0	0	0	0	1
0	0	0	0	0	1	0
0	0	0	0	1	0	1
0	0	0	0	1	1	0
0	0	0	1	0	0	0.99
0	0	0	1	0	1	0.01
0	0	0	1	1	0	0
0	0	0	1	1	1	1
0	1	0	0	1	0	1
0	1	0	0	1	1	0
0	1	0	1	1	0	0
0	1	0	1	1	1	1
1	0	0	1	0	0	1
1	0	0	1	0	1	0
1	0	0	1	1	0	0
1	0	0	1	1	1	1
1	0	1	0	0	0	1
1	0	1	0	0	1	0
1	0	1	0	1	0	1
1	0	1	0	1	1	0
1	0	1	1	0	0	0
1	0	1	1	0	1	1
1	0	1	1	1	0	0
1	0	1	1	1	1	1
1	1	1	0	0	0	1
1	1	1	0	0	1	0
1	1	1	0	1	0	1
1	1	1	0	1	1	0
1	1	1	1	0	0	0.99
1	1	1	1	0	1	0.01
1	1	1	1	1	0	0
1	1	1	1	1	1	1

Table 14: *CPT* for the variable $S\#$ in the *DBN* model in Fig. 11 (the impossible entries are omitted).

Time (h)	$P(A=0)$	$P(A=1)$
100000	0.999988	**0.000008**
200000	0.999983	**0.000014**
300000	0.999974	**0.000019**
400000	0.999967	**0.000023**
500000	0.999967	**0.000027**

Table 15: Prediction inference results for the system unreliability computed on the *DBN* model in Fig. 11.

Time (h)	$P(A=0\|ev)$	$P(A=1\|ev)$
100000	0.909251	0.090749
200000	0.000000	1.000000
300000	0.532266	0.467726
400000	0.527495	0.472497
500000	**0.524189**	**0.475810**

Table 16: Prediction inference results for the variable $A\#$ in the *DBN* model in Fig. 11 assuming that $ev = 100000 : 0, 200000 : 2$.

Time (h)	$P(A=0\|ev)$	$P(A=1\|ev)$
100000	0.831101	0.168900
200000	**0.689048**	**0.310946**
300000	0.569523	0.430477
400000	0.468789	0.531207
500000	0.000000	1.000000

Table 17: Smoothing inference results for the variable $A\#$ in the *DBN* model in Fig. 11 assuming that $ev = 300000 : 0, 500000 : 2$.

GCTBN to *GSPN* (*GCTBN2GSPN*), while the query is passed to the *Inference Solver* module managing the inference process.

2. The *GCTBN* model is converted into the equivalent *GSPN* model by means of the ad-hoc translator called *GCTBN2GSPN*. The translation is performed according to the translation rules described in Sec. 4.2. The resulting *GSPN* is passed to the *GreatSPN filter*.

3. The *GreatSPN filter* stores the *GSPN* model in a couple of files (.net, .def) conforming to the *GreatSPN* [23] format. This tool allows the analysis of *GSPN* models.

4. According to the sort of query specified by the user (filtering or smoothing) and according to the inference procedure described in Sec. 5, the *Inference Solver* generates a sequence of measures to be computed on the *GSPN* model. Each of these measures is passed to the *GreatSPN* solver in order to be computed ("result request" in Fig. 12).

5. The *GreatSPN* solver performs the analysis of the *GSPN* model computing the measure requested by the *Inference Solver*. The value of the result is returned to the *Inference Solver* ("partial results" in Fig. 12). If other measures have to be computed on the *GSPN* model in order to complete the inference process, then the steps 4 and 5 are repeated. According to the results returned by *GreatSPN*, the *Inference Solver* generates the result of the query requested by the user. Such result is passed to *Draw-Net* for the visualization to the user.

7 Conclusions and Future Works

In this report we have presented a generalized *CTBN* formalism, allowing one to mix in the same model continuous time delayed variables with standard "immediate" chance variables. The usefulness of this kind of model has been discussed through an example concerning the reliability of a simple component-based system. The semantics of the proposed *GCTBN* formalism has been provided in terms of Generalized Stochastic Petri Nets (*GSPN*), a well-known formalism isomorph to semi-Markov processes, through which it is also possible to exploit well established analysis techniques, in order to perform standard prediction or smoothing inference. In particular, adopting *GSPN* solution algorithms as the basis for *GCTBN* inference, allows one to take advantage of specialized methodologies for solving the underlying stochastic process, that are currently able to deal with extremely large models; in particular, such techniques (based on data structures like matrices or decision diagrams) allows for one order of magnitude of increase in the size of the models, to be solved exactly, with respect to standard methods, meaning that models with an order of 10^{10} tangible states can actually be solved [19, 24].

However analyzing a *GCTBN* by means of the underlying *GSPN* is only one possibility that does not take explicit advantage of the structure of the graph as in *CTBN* algorithms [7, 8]. Our future works will try to investigate the possibility of adopting cluster-based or stochastic simulation approximations, even on *GCTBN* models, and in comparing their performance and quality with respect to *GSPN*-based solution techniques. In particular, since Petri Nets are a natural framework for event-based simulation, it would be interesting to investigate how simulation-based approximations can be actually guided by the underlying *GSPN* model. Finally, since symbolic representations (based on matrices or decision diagrams) have been proved very useful for the analysis of *GSPN* models, it would also be of significant interest to study the relationships between such representations and the inference procedures on probabilistic graphical models in general, since this could in principle open the possibility of new classes of algorithms for *BN*-based formalisms.

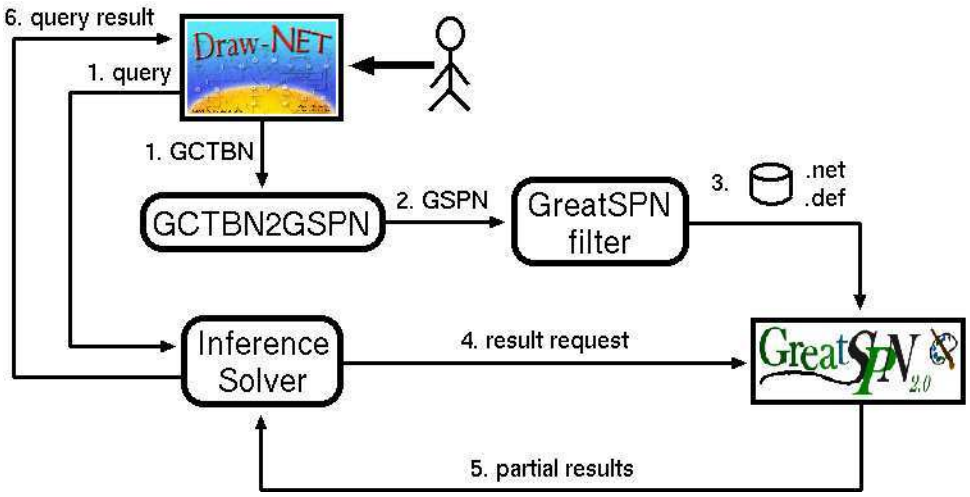

Figure 12: Scheme of the software tool architecture.

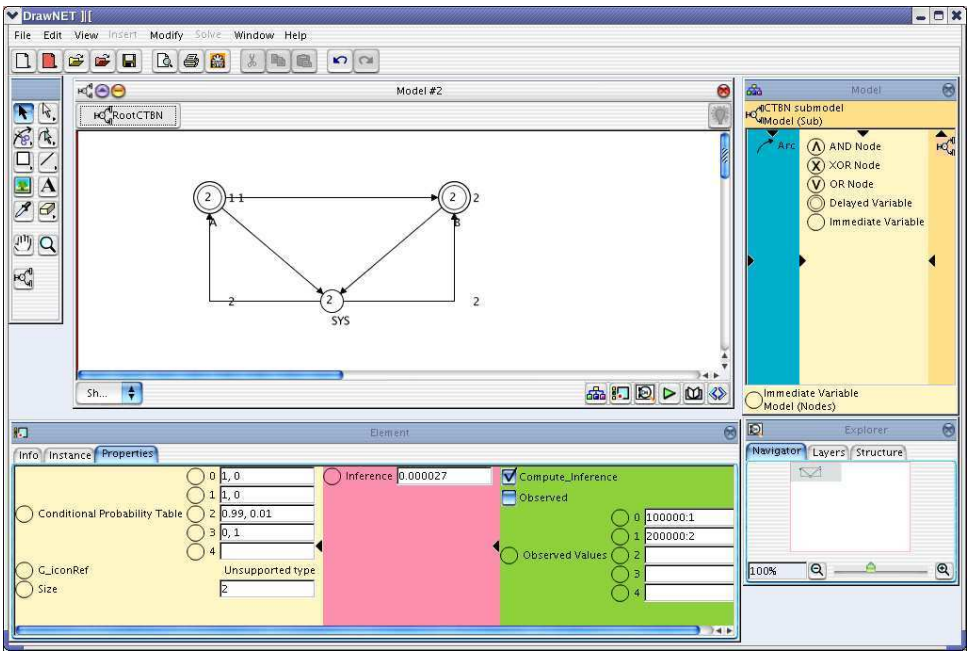

Figure 13: Screenshot of *Draw-Net*.

References

[1] F.V. Jensen and T.D. Nielsen. *Bayesian Networks and Decision Graphs (2nd ed.)*. Springer, 2007.

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

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

[4] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD Thesis, UC Berkley, 2002. http://www.cs.ubc.ca/ murphyk/Thesis/thesis.html.

[5] S. Montani, L. Portinale, A. Bobbio, and D. Codetta-Raiteri. RADYBAN: a tool for reliability analysis of dynamic fault trees through conversion into dynamic bayesian networks. *Reliability Engineering and System Safety*, 93:922–932, 2008.

[6] U. Nodelman, C.R. Shelton, and D. Koller. Continuous Time Bayesian Networks. In *Proc. 18th UAI'02*, pages 378–387, 2002.

[7] U. Nodelman, C.R. Shelton, and D. Koller. Expectation propagation for continuous time Bayesian networks. In *Proc. 21st UAI'05*, pages 431–440, 2005.

[8] S. Saria, U. Nodelman, and D. Koller. Reasoning at the right time granularity. In *Proc. 23rd UAI'07*, pages 421–430, 2007.

[9] T. El-Hay, N. Friedman, and R. Kupferman. Gibbs sampling in factorized continuous time Markov processes. In *Proc. 24rd UAI'08*, 2008.

[10] K. Gopalratnam, H. Kautz, and D.S. Weld. Extending continuous time bayesian networks. In *Proc. AAAI'05*, pages 981–986, Pittsburgh, PA, 2005.

[11] http://www.bayesia.com. BayesiaLab web page.

[12] W. G. Schneeweiss. *The Fault Tree Method*. LiLoLe Verlag, 1999.

[13] 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.

[14] D. Codetta-Raiteri, G. Franceschinis, M. Iacono, and V. Vittorini. Repairable fault tree for the automatic evaluation of repair policies. In *Proc. Intern. Conference on Dependable Systems and Networks (DSN'04)*, pages 157–159, Firenze, Italy, 2004.

[15] K.S. Trivedi. *Probability and Statistics with Reliability, Queuing and Computer Science Applications (2nd Ed.)*. John Wiley & Sons, 2001.

[16] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. J. Wiley, 1995.

[17] G. Balbo, G. Chiola, G. Franceschinis, and G. Molinari Roet. On the efficient construction of tangible reachability graph og GSPN. In *Proc. 2nd Int. Workshop on Petri Nets and Performance Evaluation*, pages 136–145, Madison, WI, 1987.

[18] A.S. Miner. Implicit GSPN reachability set generation using decision diagrams. *Performance Evaluation*, 56(1-4):145–165, 2004.

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

[20] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proceedings UAI 1988*, pages 33–42, 1998.

[21] Y. Fan and C. Shelton. Sampling for approximate inference in continuous time Bayesian networks. In *Proc. 10th Int. Symp. on AI and Mathematics*, 2008.

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

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

[24] A.S. Miner and D. Parker. Symbolic representation and analysis of large probabilistic systems. In *Validation of Stochastic Systems, LNCS 2925*, pages 296–338. Springer, 2004.