IRIS AperTO

UNIVERSITÀ DEGLI STUDI DI TORINO

AperTO - Archivio Istituzionale Open Access dell'Università di Torino

**Computing optimal repair strategies by means of NdRFT modeling and analysis**

**This is the author's manuscript**

*Original Citation:*

*Availability:*

This version is available http://hdl.handle.net/2318/147271 since 2016-06-29T11:10:27Z

*Published version:*

DOI:10.1093/comjnl/bxt134

*Terms of use:*

Open Access

Anyone can freely access the full text of works made available as "Open Access". Works made available under a Creative Commons license can be used according to the terms and conditions of said license. Use of all other works requires consent of the right holder (author or publisher) if not exempted from copyright protection by the applicable law.

(Article begins on next page)

27 April 2024

# Computing Optimal Repair Strategies by means of NdRFT Modelling and Analysis

MARCO BECCUTI[1], GIULIANA FRANCESCHINIS[2], DANIELE CODETTA-RAITERI[2] AND SERGE HADDAD[3]

*1. Dipartimento di Informatica, Università di Torino, Italy.*
*2. Dipartimento di Scienze e Innovazione Tecnologica, Università del Piemonte Orientale, Italy.*
*3. Laboratoire Spécification et Vérification,École Normale Supérieure de Cachan, France*
*Email: beccuti@di.unito.it,{giuliana, raiteri}@mfn.unipmn.it, haddad@lsv.ens-cachan.fr*

**In this paper, the *Non deterministic Repairable Fault Tree* (NdRFT) formalism is proposed: it allows the modeling of failures of complex systems in addition to their repair processes. Its originality with respect to other Fault Tree extensions allows us to address repair strategy optimization problems: in an NdRFT model, the decision as to whether to start or not a given repair action is non deterministic, so that all the possibilities are left open. The formalism is rather powerful allowing the specification of self revealing events, components degradation, whether local repair, global repair and preventive maintenance can be applied, and the resources needed to start a repair action. The optimal repair strategy with respect to some relevant system state function, e.g. system unavailability, can then be computed by solving an optimization problem on a *Markov Decision Process* derived from the NdRFT. Such derivation is obtained by converting the NdRFT model into an intermediate formalism called Markov Decision Petri Net (MDPN). In the paper, the NdRFT syntax and semantics are formally described, together with the conversion rules into MDPN. The application of NdRFT is illustrated through examples.**

*Keywords: Fault Tree; Optimal repair strategy; Markov Decision Process; Markov Decision Petri Net.*

*Received XX XXXX XXXX; revised XX XXXX XXXX*

## 1. INTRODUCTION

Fault Trees (FT) [1] are a well-known formalism for the evaluation of the dependability of complex systems. They provide an intuitive representation of the system in terms of their failures, modeling how the combinations of failure events relative to the components of systems can cause the failures of the sub-systems or of the whole system. A typical measure computable by means of FTs is system reliability as a function of time.

Many extensions of this formalism have been proposed to enhance the features of FT for the design and the assessment of systems (e.g. Dynamic FT [2], Parametric FT [3], etc.). One of these extension, the Repairable FT (RFT) [4], was presented to evaluate the effect of different repair policies on a repairable system.

In [5], we presented a new FT extension, called *Non deterministic Repairable Fault Tree* (NdRFT) which has been designed to define and solve repair strategy optimization problems. In an NdRFT model the possible repair strategies are not predefined; on the contrary, the best strategy minimizing the failure probability of the global system or, more generally, optimizing some function of the system state, is automatically computed. This is done by defining the NdRFT semantics in terms of a *Markov Decision Process* (MDP), a formalism embedding non deterministic and probabilistic behavior [6], and then solving the optimization problem using the methods available for MDPs. The generation of the MDP is achieved by an intermediate translation of the NdRFT model into a *Markov Decision Petri Net* (MDPN) [7]: this allows us to reuse the efficient algorithms devised to derive an MDP from an MDPN. A direct translation from NdRFT to MDP, requires the implementation of a mechanism to combine the failure/repair events of all components into a single complex transition or action: this comes for free if the MDPN analysis algorithms are used. Moreover, the compositional translation technique proposed is modular, and it allows us to perform refinements locally in the translation without the need to adjust the MDP generation algorithm accordingly: this opens the way to several interesting extensions mentioned in the paper.

The NdRFT formalism expresses in an elegant way several possible start repair options which are based on the possibility for events to be self revealing or not, only self revealing events detection can trigger a repair process, and on the notion of local versus global repair, to account for the possibility to repair single components or whole

subsystems. The repair of each component can start as a consequence of the detection of different events according to cost considerations and under the constraint of availability of the required resources.

The NdRFT allows the modeler to express in a familiar language (NdRFT extends FT) which events may cause the system to fail and the repair options in the system; in this way, it avoids dealing with a larger, unstructured and state-level MDP model, which is instead automatically derived from the NdRFT.

This paper completes and extends the results presented in [5] by including a proof of correctness for the proposed approach and by presenting a new feature which allows the user to model degradation and preventive maintenance. Moreover, a discussion on how to improve NdRFT solution efficiency and a new case study are added. It is structured as follows: Sec. 2 presents some related work concerning FT extensions and analysis methods and tools; in Sec. 3 the formal definition of the NdRFT syntax and its MDP semantics is provided; Sec. 4 explains the rules for deriving from a NdRFT model the corresponding MDPN; in the same section some efficiency issues of the proposed solution method are discussed. A new extension allowing the user to model degradation and preventive maintenance of the system components is reported in Sec.5. Two examples of application are presented in Sec. 6. Finally, in Appendix A the correctness of the translation rules is proven.

## 2. RELATED WORK

**Fault Trees.** An FT is a Directed Acyclic Graph (DAG) with two types of nodes: events and gates. An example is shown in Fig. 1 (disregarding node annotations). Events represent the failure of components, subsystems or system. We consider an event as a Boolean variable: it is initially *false* and it becomes *true* when the failure occurs. The events graphically represented as a rectangle with an attached circle are called *Basic Events* (BEs) and model the failure of the components of the system; such events are stochastic, so their occurrence is ruled by some probability distribution. The event nodes depicted as rectangles represent the failure of subsystems; we call them *Internal Events* (IEs). *Gates* are connected by means of arcs to several input (basic or internal) events and to a unique (internal) output event; the effect of a gate is the propagation of the failure to its output event if a particular combination of its input events occurs. In the standard version of the FT formalism, three types of gates are present and correspond to the *AND*, *OR* and *"K out of N"* Boolean functions. Finally, we have a unique event called *Top Event* (*TE*), modeling a failure of the whole system. The FT corresponds to a Boolean formula expressing the *TE* truth value as a function of its variables (BEs).

The analysis of an FT model allows the user to compute several dependability measures such as the system reliability, the system minimal cut-sets, the mean time to failure, the criticality of each component [1]; in particular, the system *reliability* at time $t$ is the probability that the system has been working in the time interval $(0,t)$. The most efficient way to perform the analysis of an FT consists of generating the *Binary Decision Diagram* (BDD) [8] representing the same Boolean formula expressed by the FT and computing on it the above measures.

**Extending FT with repair processes.** In the literature several tools and formalisms have been proposed to extend FT with repair processes. For instance *Stars Studio* and *ASTRA* [9] extend the FT formalism by modeling the repair of single components (BEs). Usually the modeler can associate with a BE, in addition to the failure rate, also the repair time or rate of the component. The behavior of the component is equivalent to a *Continuous Time Markov Chain* (CTMC) [10] with two states: working and failed. The repair process is triggered by the component failure and has effect only on the same component.

The *SHARPE* tool [10] allows hierarchical modeling: the probability to occur of a BE is set equal to some measure computed on another kind of model, for instance a CTMC or a *Generalized Stochastic Petri Net* (GSPN) [11] defined by the user. In this way, the failure and repair mode of a component may be more complex than a simple transition from the working state to the failure state and vice-versa.

The *HIMAP* tool [12] deals with FTs including repairs, according to two approaches [13]: 1) the modeler can design a FT model and the tool automatically converts it into the equivalent CTMC that the modeler can edit in order to represent and analyze the presence of repairs; 2) the modeler can design a FT model where some BEs are declared as repairable. The tool converts the subtrees including the repairs into a CTMC, and analyzes them in this form.

*Dynamic Fault Trees* (DFT) [14] are a particular extension of FT where dependencies between BEs are set by means of *dynamic gates*. The analysis of the model is performed by exploiting *Input-Output Interactive Markov Chains*. In [15], dependencies involving components or subsystems can be expressed by means of specific arcs; the model is evaluated by means of a *Boolean logic driven Markov process* (BDMP). In [16] and [15] the model can include repair, but each repair process involves a single component instead of a subsystem, and the possibility of a limited number of repair facilities is not considered. In [16] the possibility to include a *Repair Station* is mentioned with the goal of representing more complex repair processes. In [15] a component can have an increasing failure rate, and imperfect repair can be modeled.

*Reliability Block Diagrams* (RBD) [10] are a formalism with the same goal of FTs. An RBD represents, in terms of series and parallel constructs, the combinations of components that have to be working in order for the system to be working. As DFTs extend FTs, RBDs have been extended by including the possibility of representing particular forms of dependencies, multi-state components and repairable components. The resulting formalism is called *Dynamic Reliability Block Diagrams* (DRBD) [17]. The solution of a DRBD model is performed by automatic conversion into GSPN, according to pre-defined rules.

All the works described so far consider the single component repairs. The *Repairable Fault Tree* (RFT) formalism [4], instead models the repair of subsystems. This kind of repair is a complex process characterized by several parameters defining a *repair policy*. They are the event triggering the repair process, the mean time to detect the subsystem failure, the set of repairable components, the mean time to repair each component, the number of repair facilities, the components repair order, etc. A repair process is represented by a new node called *Repair Box* (RB) and establishes several dependencies among the events in the RFT: this requires the analysis of the model by generating its state space, but is limited to the modules of the RFT that contain RBs, while the rest of the model is solved resorting to standard FT analysis. In [4], the state space analysis of the modules containing repair has been achieved by conversion into GSPN. Applications of RFT models to real cases are available in [18].

In the RFT formalism, the repair policy (or strategy) is *pre-defined* by the modeler and is associated with the RB primitive; therefore the only way for the modeler to identify the best policy consists of analyzing the system according to several repair policies by constructing several RFT models, and by comparing the system availability values returned by the analysis of RFT models. Therefore the RFT formalism does not automatically determine the best repair policy.

The ability to determine the optimal repair policy given all the repair possibilities is an issue concerning several fields of engineering. So far, this problem has usually been faced in the literature in analytical ways, typically in form of operations research problems [19, 20, 21].

**Dynamic Bayesian Networks.** More recently the analysis of DFT models has been addressed by the use of *Dynamic Bayesian Networks* (DBN) [22]. DBNs extend Bayesian Networks by providing an explicit discrete temporal dimension. The way to convert a DFT into DBN is described in [23]. With respect to CTMC or GSPN, the use of DBN avoids the generation of the whole state space, and takes advantage of the factorization in the temporal probability model. The use of DBN allows the modeler to compute by means of inference procedures, predictive and diagnostic measures conditioned by observations about the state of the system or the state of its components, during the system mission time. The approach based on DBN has been extended to deal also with the repair of components or subsystems, as shown in [24].

DBNs can be extended by including decision nodes and cost nodes. The resulting formalism is called *Dynamic Decision Network* (DDN) [25, 26]. A decision node represents a decision concerning an action to be performed, and influences a subset or all of the random variables. A cost node instead, represents a reward function influenced by a subset or all of the random variables. As in DBNs random variables are replicated in two or more time slices[1], also in DDNs random variables, decision nodes and cost nodes are replicated in two or more time slices. The DDNs are exploited to solve optimization problems that are similar to those objects of this paper: random variables represent the evolution of the state of the system, decision nodes represent the choices, while the cost node represents the function to be optimized. In this sense, a DDN model provides the factorized representation of a discrete time MDP or a discrete time *Partially Observable MDP* (POMDP) [27]. Despite the fact that DDNs represent in a compact form very large complex (PO)MDPs, such a form does not reduce the complexity of the exact solution because it requires the generation and solution of the underlying MDP. Some techniques have been developed to exploit the factorized form of DDNs, in order to obtain the approximate solution of the underlying (PO)MDP [28]. Both DDNs and MDPNs are high-level languages to express an MDP, but the former derives from Bayesian Networks, while the latter derives from Petri Nets.

[1]When the Markovian assumption holds, only two time slices are enough to model the temporal evolution of the system state.

## 3. NON DETERMINISTIC RFT

The NdRFT formalism presented in this paper provides an intuitive notation to express alternative repair strategies and effective methods to compute an optimal strategy w.r.t. a given objective function (minimizing costs, maximizing system availability or a combination of the two) defined in terms of states and events. While in the RFT formalism, the best repair strategy is not the result of the model analysis, in the NdRFT formalism instead, the optimal repair strategy is the result of the model analysis. The NdRFT formalism expresses several possible start repair options based on: 1) the concept of self revealing events, whose occurrence can trigger a repair process; 2) the notion of local versus global repair action, including the possibility that a basic component repair can be triggered by different events; 3) the notion of global repair supervisor component; 4) the notion of resource requirements for each type of repair action. Given this information, the analysis of the NdRFT model can provide an optimal repair strategy. This is done by generating a MDP from the NdRFT, through an intermediate translation into a *Markov Decision Petri Net* (MDPN) [7]: this allows us to reuse the efficient algorithms devised to derive an MDP from an MDPN.

### 3.1. NdRFT syntax

In this section the NdRFT formalism is first presented by means of an example, then it is formally defined.

In the NdRFT example in Fig. 1 (whose meaning will be explained in Sec. 6) the events are depicted in different ways according to their characteristics: self revealing/not self revealing, repairable/not repairable and local/global repair strategy (resp. *alarm*, *rep* and *str* attributes). Down arrows next to BEs, labeled with a number, indicate failure probabilities; up arrows next to repairable BEs, or to internal events with global strategy, labeled with a number, indicate the repair continuation probability. The number of required repair resources is also specified for these events.

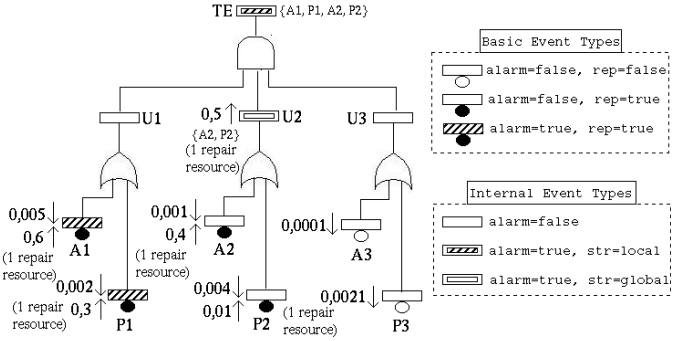


**FIGURE 1.** The NdRFT model of the AHRS.

Examples of not repairable BEs in the example are $A3$ and $P3$: these represent components that cannot be recovered after a failure. An example of self revealing and repairable BE is $A1$: its failure immediately enables a repair action of the component, while the repair of a non self revealing (but repairable) event, like e.g. $A2$, can only be enabled by a self revealing internal event connected to it: for $A2$ it can be $U2$ or $TE$. In the example of Fig. 1 we have only one type of resource and each repair action requires only one resource.

The event attribute *str* defines the granularity of the repair process triggered by the occurrence of a self revealing IE $e$: if the repair strategy is *global* (as for $U2$ in the example), *all* the repairable basic components ($A2$ and $P2$ in the example) involved in the repair process (expressed through attribute *e.torep* and graphically represented by a set of BE identifiers in braces next to the IE) are repaired simultaneously and brought back to the working state when the global repair process terminates. This means that a global repair process is a single action (e.g. representing the substitution of a down server with a new server); while a global repair action is ongoing, the involved basic components (those in *e.torep*) cannot be simultaneously involved in any other repair action (global or local). In the case of *local* repair (as for $TE$ in the example), for each repairable BE component in *e.torep*, it is possible to decide to repair or not such component; moreover the repair of single components may not start simultaneously (e.g. when there are not enough free repair resources). A BE can appear in the *torep* set of several internal events; for example $A2$ and $P2$ are in the *torep* set of both $U2$ and $TE$: when only one of the two BEs is down, the local strategy could be more appropriate, but it can be activated only if $TE$ is down. Otherwise, if both $A2$ and $P2$ are down the global repair of $U2$ may be more convenient. Observe that given the example NdRFT structure, $U2$ can immediately detect when one or both events $A2$ and $P2$ are down, and trigger the substitution of both.

To complete the example let us consider some possible *cost* function, defined in terms of *states* and *events*: let $FailEv$ denote the subset of NdRFT events that have failed in a given state of the system under study, and let $RepairEv$ be the subset of events that are undergoing a repair action (basic repairable events for local repair actions, internal events with a global repair strategy for global repair actions). A first cost function example is:

$$\begin{cases} cost.state(FailEv, RepairEv) = \mathbf{1}_{FailEv}(TE); \\ \forall\, e, cost.event(e) = 0, \end{cases} \tag{1}$$

where $\mathbf{1}_A(x)$ is an indicator function that returns 1 when $x \in A$, 0 otherwise: in this case the goal is to minimize the probability of a global system failure. In this example the cost of repair actions is not taken into account, while the next example instead takes also the repair cost into account:

$$\begin{cases} cost.state(FailEv, RepairEv) = \\ \quad = down_penalty \cdot \mathbf{1}_{FailEv}(TE) + \\ \quad \sum_{e \in RepairEv} rep_cost_per_time_unit(e); \\ \forall\, e, cost.event(e) = start_repair_cost(e); \end{cases} \tag{2}$$

where $down_penalty$ is the cost of having system down per time unit, $rep_cost_per_time_unit(e)$ is the cost per time unit of a repair associated with event $e$, and $start_repair_cost(e)$ is a fixed cost paid every time a repair process associated with event $e$ starts. In the example a repair cost could be defined for the repair actions of events $A1$, $P1$, $A2$, $P2$, and $U2$: the cost can be an instantaneous start repair cost or a cost that accumulates during the whole repair process (or both).

We now present the formal definition of the NdRFT formalism.

DEFINITION 3.1 (Non deterministic Repairable FT). *An NdRFT is a six-tuple:* $\mathscr{S} = \langle \mathscr{E}, \mathscr{G}, \mathscr{A}, \mathscr{R}, res_0, cost \rangle$ *where:*

$\mathscr{E}$ *is the set of events.*

$\mathscr{G}$ *is the set of gates;* $\mathscr{E} \cap \mathscr{G} = \emptyset$. *A gate* $g$ *has a type*[2] *denoted* $g.type \in \{\texttt{and}, \texttt{or}\}$.

$\mathscr{A}$ *is the set of arcs, a subset of* $\mathscr{E} \times \mathscr{G} \cup \mathscr{G} \times \mathscr{E}$. *For* $x$ *belonging to* $\mathscr{E} \cup \mathscr{G}$, *we denote* $x^\bullet \equiv \{y \mid (x,y) \in \mathscr{A}\}$ *and* ${}^\bullet x \equiv \{y \mid (y,x) \in \mathscr{A}\}$. $\mathscr{A}$ *satisfies:*

1. $\forall g \in \mathscr{G}, |g^\bullet| = 1$ *and* $\forall e \in \mathscr{E}, |{}^\bullet e| \leq 1$
2. *There is exactly one event, denoted* $\top$ *and called Top Event, s.t.* $\top^\bullet = \emptyset$*; all other events satisfy* $|e^\bullet| \geq 1$
3. *The set of events can be partitioned into* basic *events* $\underline{\mathscr{E}} \equiv \{e \mid {}^\bullet e = \emptyset\}$ *and* internal *events* $\overline{\mathscr{E}} \equiv \{e \mid {}^\bullet e = 1\}$

$\mathscr{R}$ *is a finite set of repair resource types;* $res_0 \in Bag(\mathscr{R})$ *are the available resources, where* $Bag(\mathscr{R})$ *is a generalization of a set (called multiset) that can contain several occurrences of the same element from set* $\mathscr{R}$.

$\forall e \in \underline{\mathscr{E}}$ *the following attributes are defined:*

1. ***alarm*** $\in \{\text{true}, \text{false}\}$*: it states if* $e$ *is* self revealing*;*

[2]Since the proposed optimization method is based on the state space, other gate types could easily be considered, including dynamic ones: in this paper only and/or gates are considered for the sake of space.

2. ***fprob*** $\in [0,1]$*: it is the failure probability associated with e;*
3. ***rep*** $\in \{\text{true}, \text{false}\}$*: it specifies if e is* repairable*. $\underline{\mathscr{E}}_R$ denotes the set of repairable basic components $\underline{\mathscr{E}}_R = \{e \in \underline{\mathscr{E}} | e.rep = true\}$;*
4. ***rprob*** $\in [0,1]$*: it defines a repair continuation probability (i.e. $1 - e.rprob$ is the probability to complete the repair at each time step). It is defined only if* $e$.rep=true*;*
5. ***res*** $\in Bag(\mathscr{R})$*: it specifies a multiset of resources required by the local repair process of e.*

*$\forall e \in \overline{\mathscr{E}}$ the following attributes are introduced:*

1. ***alarm*** $\in \{\text{true}, \text{false}\}$*: it states if e is* self revealing *and can trigger a repair process;*
2. ***str*** $\in \{\text{local}, \text{global}\}$*: it defines the repair strategy associated with e. It is defined only if* $e$.alarm=true*. $\overline{\mathscr{E}}_{GR}$ denotes the set of internal events with global repair strategy $\overline{\mathscr{E}}_{GR} = \{e \in \overline{\mathscr{E}} | e.alarm = true \wedge e.str = global\}$;*
3. ***rprob*** $\in [0,1]$*: it defines a repair continuation probability and it is defined only if* $e$.str=global*;*
4. ***res*** $\in Bag(\mathscr{R})$*: it specifies a multiset of resources required by the global repair process (defined only if* $e$.str=global*).*
5. ***torep*** $\in 2^{\underline{\mathscr{E}}}$*: it denotes the set of BEs which is repaired by a repair process triggered by e. $\forall e' \in e.torep$ a path (i.e. a set of arcs in $\mathscr{A}$) connecting e to $e'$ must exist and if $e.str = \texttt{local} \rightarrow e'.rep =$* true*;*

*cost: it defines the* ***penalty*** *produced by a failed system or subsystem, and the* ***repair cost****. It has two components:*

1. ***state****: $2^{\mathscr{E}} \times 2^{\mathscr{E}} \rightarrow \mathbb{R}$ it is applied to a subset of down components and/or a subset of components under repair at a given time instant giving a penalty value;*
2. ***event****: $\overline{\mathscr{E}}_{GR} \cup \underline{\mathscr{E}}_R \rightarrow \mathbb{R}$ it indicates for each start repair event the corresponding instantaneous cost.*

*this function is used to set the optimization problem (of course the goal is cost minimization).*

### 3.2. MDP semantics of NdRFT

**MDP definition.** A (discrete time and finite) MDP is a dynamic system where the transitions between states follow a two-step process. First, one non deterministically selects an action inside the subset of enabled actions. Then one samples the new state w.r.t a probability distribution depending on the current state and the selected action. The non deterministic step represents a decision taken by a controller in order to manage the system (e.g. the decision of repairing a subset of failed components).

The probabilistic step takes into account that the effect of an action statistically depends on non modeled (or unknown) parameters.

In order to formally define the objective to optimize, one associates a reward with any state and selected action (the reward can also be interpreted as a cost). The following definition formalizes these concepts.

DEFINITION 3.2 (Markov Decision Process, MDP:). *An MDP $\mathscr{M}$ is a four-tuple $\mathscr{M} = \langle S, A, p, r \rangle$ where:*

*1. S is a finite set of states,*

*2. A is a finite set of actions defined as $\bigcup_{s \in S} A_s$ where $A_s$ is the set of enabled actions in s,*

*3. $\forall s \in S, \forall a \in A_s, p(\cdot|s,a)$ is a (transition) probability distribution over S such that $p(s'|s,a)$ is the probability to reach $s'$ from s by triggering action a,*

*4. $\forall s \in S, \forall a \in A_s, r(s,a) \in \mathbb{R}$ is the reward associated with state s and action a.*

Once an action choice is fixed for each state, the MDP behaves like a Markov chain and different global measures on the random path can be defined as for example the (discounted) sum of rewards or the average of the rewards. The goal of the analysis is computing the optimal value of the measure, and when possible, computing the associated strategy. In finite MDPs, efficient solution techniques have been developed to this purpose [6] and different tools are based on this theory.

**NdRFT semantics.** In this paragraph, we will define precisely the dynamic behavior of an NdRFT, which can be described by an MDP. Let us first define the MDP states:

*$MDP_{NdRFT}$ state.* A state of the MDP corresponding to a given NdRFT is a pair: $\rho = \langle \{st_e\}_{e \in \mathscr{E}}, \{sup_e\}_{e \in \mathscr{E}} \rangle$ with

- $st_e \in \{Up, Down, LocRep, GlobRep_u, GlobRep_d\}$ represents the state of the component/subsystem associated with event $e$.
  If $e \notin \underline{\mathscr{E}}$, $st_e \in \{Up, Down\}$ can be derived from the FT structure and the state of all BEs;
- $sup_e \in \mathscr{E} \cup NULL$ indicates for each BE $e$ under repair, which is the *supervisor* of the repair process:
  if $st_e \in Up, Down \Rightarrow sup_e = NULL$
  if $st_e = LocRep \Rightarrow sup_e = e$
  if $st_e \in GlobRep_u, GlobRep_d \Rightarrow sup_e = e' : e \in e'.torep$

*LocRep* identifies the components involved in a local repair process; instead $GlobRep_u$ and $GlobRep_d$ identify components involved in a global repair, and the subscript distinguishes between components that were *Up* or *Down* when the repair started.

The initial state $\rho_0$ is: $\forall e \in \mathscr{E}, st_e^0 = Up \wedge sup_e^0 = NULL$. We shall denote $sup(\mathscr{E}) = \cup_{e \in \mathscr{E}} sup_e$.

For each state $\rho$, it is possible to define the multiset $res_\rho$ of busy resources as: $res_\rho = \sum_{e \in sup(\mathscr{E})} e.res$. Of course at each time point the following condition must be verified: $res_\rho \subseteq res_0$ defined as $\forall r \in \mathscr{R}, res_\rho(r) \leq res_0(r)$, where $res_i(r)$ denotes the multiplicity of $r$ in $res_i$.

*Actions and transitions.* Let us define the set $A_\rho$ of actions that can be chosen in state $\rho$: each action $a \in A_\rho$ is a mapping $\underline{\mathscr{E}} \rightarrow \{Repair_e, NotRepair\}$ satisfying the following constraints:

**TABLE 1.** Table of the state change due to the choice of action $a$ in the non deterministic step

| condition on $a$ | $st_e$ | $st'_e$ | $sup'_e$ |
|---|---|---|---|
| $a(e) = NoRepair$ | any | $st_e$ | $sup_e$ |
| $a(e) = Repair_{e'}, e' = e \vee (e' \neq e \wedge e'.str = local)$ | $Down$ | $LocRep$ | $e$ |
| $a(e) = Repair_{e'}, e' \neq e \wedge e'.str = global \wedge e \in e'.torep$ | $Up$ | $GlobRep_u$ | $e'$ |
| $a(e) = Repair_{e'}, e' \neq e \wedge e'.str = global \wedge e \in e'.torep$ | $Down$ | $GlobRep_d$ | $e'$ |

**TABLE 2.** Table of the state change due to the probabilistic step

| Event type | $st_e$ | $st'_e$ | $sup'_e$ | $prob_e$ |
|---|---|---|---|---|
| $e \in \underline{\mathscr{E}}$ | $Down$ | $Down$ | $NULL$ | 1 |
| $e \in \underline{\mathscr{E}}$ | $Up$ | $Up$ | $NULL$ | $1 - e.fprob$ |
| $e \in \underline{\mathscr{E}}$ | $Up$ | $Down$ | $NULL$ | $e.fprob$ |
| $e \in \underline{\mathscr{E}}$ | $LocRep$ | $LocRep$ | $sup_e$ | $e.rprob$ |
| $e \in \underline{\mathscr{E}}$ | $LocRep$ | $Up$ | $NULL$ | $1 - e.rprob$ |
| $e' \in \overline{\mathscr{E}}_{GR} \cap sup(\underline{\mathscr{E}})$ | $\forall e \in e'.torep\ GlobRep_u/GlobRep_d$ | $GlobRep_u/GlobRep_d$ | $sup_e$ | $e'.rprob$ |
| $e' \in \overline{\mathscr{E}}_{GR} \cap sup(\underline{\mathscr{E}})$ | $\forall e \in e'.torep\ GlobRep_u/GlobRep_d$ | $Up$ | NULL | $1 - e'.rprob$ |

- $a(e) = Repair_e \Rightarrow st_e = Down \wedge ((e.rep = true \wedge e.alarm = true) \vee (\exists e' : st_{e'} = Down \wedge e'.str = local \wedge e \in e'.torep))$
- $a(e) = Repair_{e'} \Rightarrow st_e \in Up, Down \wedge st_{e'} = Down \wedge e'.alarm = true \wedge e'.str = global$; in this case it must be $\forall e'' \in e'.torep\ \ a(e'') = Repair_{e'}$
- $res_\rho + \sum_{e:\exists e', a(e')=Repair_e} res_e \subseteq res_0$

Once an admissible action $a \in A_\rho$ is chosen, an intermediate state $\langle \rho, a \rangle$ is reached: here a probability distribution allows us to determine the state change; the probability distribution can be derived from the distributions of failure occurrence and repair completion events.

Table 1 shows the state change corresponding to the non deterministic step due to action $a$ in state $\rho$, where $st'_e$ and $sup'_e$ indicate the state of $e$ in the intermediate state $\langle \rho, a \rangle$ (after taking action $a$). Observe that according to the definition of admissible actions, any global repair must start simultaneously for all BEs included in the $torep$ attribute of the IE $e'$ which triggers the repair; $e'$ is thus set as supervisor for all BEs in $e'.torep$.

Table 2 shows the state change corresponding to the probabilistic step from the intermediate state $\langle \rho, a \rangle$ to the new state $\rho'$; in this table $st'_e$ and $sup'_e$ indicate the state of $e$ in $\rho'$ while $st_e$ and $sup_e$ refer to $\langle \rho, a \rangle$. The probabilistic step probability is defined as $prob = \prod_e prob_e$ where $prob_e$ is the probability indicated in the last column of the table, and the product is indexed on the following set of events $\{e \in \underline{\mathscr{E}} \wedge st_e \in \{Up, Down, LocRep\}\} \cup \{e \in \overline{\mathscr{E}}_{GR} \cap sup(\underline{\mathscr{E}})\}$, i.e. all basic events not involved in any global repair, and all IE supervising an ongoing global repair.

The last two lines indicate the fact that in case of global repair triggered by IE $e'$ all the basic components in $e'.torep$ change state simultaneously at the end of the global repair.

Summarizing, the dynamics of the MDP corresponding to an NdRFT is defined in terms of two steps: a non deterministic one selecting the subset of repair actions that should start and a probabilistic one probabilistically choosing the newly occurred failures and which ongoing repair activities have to be completed.

The MDP definition includes also a reward function: it is derived from the NdRFT cost function as follows. Let $\rho$ be a MDP state and let $a$ be an action corresponding to a non deterministic step that may occur in $\rho$. Then the reward function $r(\rho, a)$ is defined as follows:

$$cost.state(FailEv(\rho), RepairEv(\rho)) + \sum_{e \in ev_rep(a)} cost.event(e)$$

where $FailEv(\rho)$ is the set of events that are down (including those under repair) in state $\rho$, $RepairEv(\rho) = \rho.sup(\mathscr{E})$ is the set of supervisors of ongoing repairs, and $ev_rep(a)$ is the set of events for which a start repair action exists in $a$. Obviously, more complex reward functions could be defined by updating consistently $r(\rho, a)$, as discussed later.

This completes the definition of the MDP underlying a given NdRFT.

The computation of the optimal strategy requires three steps: (1) generation of the MDP from the NdRFT, (2) analysis of the MDP, (3) presentation of the results in a form that is understandable for the designer. These steps can be automatized. The first step can be implemented in two ways: defining an algorithm that generates the set of reachable states, the corresponding non deterministic actions and consequent probabilistic state change, or translating the NdRFT in an intermediate model for which the above tasks have already been defined and implemented. In this paper we propose to use the second approach and provide an algorithm for translating an NdRFT into a *Markov Decision Petri Net* (MDPN) [7]. From the MDPN model an MDP can be automatically derived.

### 3.3. Discussion

The NdRFT model is a discrete time one. This may seem in contrast with the fact that failure models often refer to continuous time distributions (exponential, Weibull, ...). However, as thoroughly discussed in [29], it is possible to capture in a quite precise way failure processes with discrete time distributions. This can be obtained as an approximation of a continuous distribution (through discretization) but it

may also arise naturally when the probability of a failure occurrence is related to the occurrence of a given number of events (e.g. when equipment operates in cycles or on the basis of service requests, the failure probability may depend on the number of cycles performed or requests served rather than on the actual time elapsed). An application of the discretization approach is presented in [23, 24] where DFT analysis is based on DBN, as explained in Sec. 2.

Due to the discrete time assumption, the specification of the failure occurrence and repair process of each (basic) *repairable* component $x$ is given by probability $P_{Fail}(x)$ and $P_{Repair}(x)$. $P_{Fail}(x)$ (resp. $P_{Repair}(x)$) represents the probability that a failure occurs (resp. to stay under repair) at any (discrete) time step provided the corresponding component is up (resp. is down and under repair). As a consequence, the time to failure of a component ($TtF_e$), and its repair time ($repTime_e$) have a geometric distribution:
$P(TtF_e = k) = (1 - P_{Fail}(e))^{k-1} P_{Fail}(e)$
$P(repTime_e = k) = P_{Repair}(e)^{k-1}(1 - P_{Repair}(e))$

Even if in the current work we only model the geometric law, the formalism could be extended to model different (not memoryless) probability distributions. This kind of extension would allow to represents component aging as well as other interesting phenomena like, e.g., the accumulation of corrupted data, and could be coupled with the possibility to apply preventive maintenance rather than failure-triggered repair only. The modular translation technique presented in next section makes the addition of the mentioned extensions easier; obviously these extensions cause an increase in the computational cost whose impact should be carefully evaluated.

In NdRFT the repair policy is not fixed a priori (as in RFT): the choice to repair or not a failed repairable component is non deterministic. This leads to an MDP semantics: as a consequence, we can compute the optimal repair strategy minimizing some cost function. For instance, one possible goal could be to minimize the unavailability of the global system (this can be expressed by assigning a non null penalty to the states where the *TE* is true). Observe that even for this simple objective function, finding the optimal strategy is not trivial, since NdRFT accounts for limited repair resources (each repair action can be associated with a multiset of required resources to complete it), a constraint that is usually disregarded in other extended FT formalisms. Introducing also penalties for performance degradation and repair action cost makes it important to have an automatic procedure to find the optimal strategy.

In NdRFT we can model processes where the components or the subsystems under repair become again available as soon as possible (maybe in a degraded state) without waiting for the repair of all its down BE components. Moreover, the notion of self revealing components allows us to specify which events are a sort of alarm, so that the corresponding repair activity can start (this generalizes the notion of *trigger event* introduced in RFT). This feature allows us to model situations where some faults do not generate symptoms until they are combined with other fault events which make them emerge later on.

Finally different repair actions may share components (although a given component can be subject to only one repair action at a time): this increases the flexibility in the choice among the possible repair strategies that may be pursued, while still allowing a simple and clean semantics based on the notions of self revealing components and of global vs. local repair strategy.

## 4. TRANSLATION FROM NDRFT TO MDPN

In this section we describe how to obtain the corresponding MDPN model from an NdRFT model. An informal introduction to the MDPN formalism is provided first, then the pattern-based translation algorithm is presented. The generation of the MDP from the MDPN model is performed as described in [7].

**A brief introduction to MDPNs.** MDPNs were first introduced in [7] as high level models to specify the behavior of an MDP. The main features of the high level formalism are the possibility to specify the general behavior as a composition of the behavior of several components (those that are subject to local non deterministic choice are called controllable, otherwise they are called non controllable); moreover any non deterministic or probabilistic transition of an MDP can be composed by a set of non deterministic or probabilistic steps, each one involving a subset of components.

An MDPN model is composed of two parts, both specified using the PN formalism with priorities associated with transitions: the $PN^{nd}$ subnet and the $PN^{pr}$ subnet (describing the non deterministic (ND) and probabilistic (PR) behavior respectively). The two subnets share the set of places, while having disjoint transition sets. In both subnets the transitions are partitioned into "run" and "stop" subsets, and each transition has an associated set of components involved in its firing (in the $PN^{nd}$ only controllable components can be involved). Transitions in $PN^{pr}$ have a "weight" attribute, used to compute the probability of each firing sequence. Run transition firings represent intermediate steps in a ND/PR transition at the MDP level, while stop transitions represent the final step in a ND/PR transition, for all components involved in it. An MDPN model behavior alternates between ND transition sequences and PR transition sequences, initially starting from a ND state. The PR sequences are determined according to the $PN^{pr}$ structure, start with a PR state reached by an ND state, and include exactly one stop transition for each component; the ND sequences are determined by the $PN^{nd}$ structure, start from a ND state reached by a PR state, and include exactly one stop transition for each controllable component plus possibly a stop "global" transition. Moreover, in the MDPN formalism we can specify a reward/cost function, called $rs()$, associated with every system state and one, called $rt()$, associated with every non deterministic transition.

The generation of the MDP corresponding to a given MDPN has been described in [7]: it consists of (1) a composition step, merging the two subnets in a single net, (2) the generation of the Reachability Graph (RG) of the

**TABLE 3.** Correspondence between MDP states and MDPN markings.

| **Basic Events** | |
|---|---|
| Event status | Marking |
| *Up* | $m(Up_e) = 1$ |
| *Down* | $m(Down_e) = 1 \wedge m(NotInv_e) = 1$ |
| *LocRep* | $m(UnderRepair_e) = 1 \wedge m(NotInv_e) = 0$ |
| $GlobRep_{u/d}$ | $m(Up/Down_e) = 1 \wedge m(NotInv_e) = 0$ $\wedge \exists! e' s.t. e \in torep(e') \wedge UnderRepair_{e'} = 1$ |
| **Internal Events** | |
| Event status | Marking |
| *Up* | $m(OutComp_e) = 0$ |
| *Down* | $m(OutComp_e) = 1$ |
| *Supervisor* of *global repair* ($sup(e') = e$) | $m(UnderRepair_e) = 1$ $\forall e' \in torep(e)$, $m(NotInv_{e'}) = 0 \wedge m(UnderRepair_{e'}) = 0$ |

composed net using a conventional interleaving semantics, then any path in the RG can be partitioned into sub-paths alternating PR and ND sequences: time advances at each start of an ND sequence, (3) two reduction steps transforming each PR and ND sequence in the RG into a single MDP transition. Observe that during this step, all the PR/ND sequences that represent any interleaving corresponding to the same deterministic or probabilistic firing sequence, are merged into a unique representative.

In the next subsections a pattern based approach to generate a MDPN mimicking the dynamic behavior of an NdRFT is presented. We introduce the set $Comp^{pr} = \mathscr{E}$ of components of the MDPN and the subset $Comp^{nd} = \underline{\mathscr{E}}_R \cup \overline{\mathscr{E}}_{GR}$ of controllable components.

The $PN^{pr}$ and $PN^{nd}$ are obtained from the NdRFT model using a pattern-based approach. We illustrate the method describing the basic patterns, and how to instantiate and compose them.

### 4.1. Generating the PR subnet

Fig. 2 shows how each BE can be translated in a $PN^{pr}$ submodel according to their *alarm* and *rep* attributes: each non repairable event is translated into subnet **A**, while each repairable event is translated into subnet **B**. The names assigned to the places have been chosen with the aim of making clear the association between their marking and the event state names that have been used to specify the MDP semantics of NdRFT models (e.g. the marking of places $Up_e$, $Down_e$,$UnderRepair_e$ represent the *Up*, *Down*, and *LocRep* event states mentioned in the MDP semantics section). Table 3 provides a summary of such an association referring to non deterministic markings, reached at the end of a *maximal probabilistic firing sequence* (defined later). A complete discussion on such a correspondence can be found in the Appendix.

Run and stop transitions have different icons, so that they can be easily distinguished. Moreover, each transition has a priority (label $prioR_i$, $prioS_i$, $prioSI_i$ and $prioRI_i$ indicated next to each transition) and a weight, which is renormalized w.r.t. the set of enabled transitions to obtain a firing probability. At each probabilistic step an *Up* component can either remain *Up* (sequence $WorkR_e$, $WorkS_e$) or go *Down* (sequence $FailR_e$, $FailS_e$); each transition participating in this first step involves only one component, namely $e$. The chosen priority reproduces the propagation of basic event states to obtain intermediate event states, so that it introduces a partial order on the NdRFT events according to $prioSI1_i < prioSI2_i < prioRI_i < prioS1_i < prioS2_i < prioR_i$.

A *Down* component can either remain *Down* (stop transition $FailS_e$) or start its repair (run transition *Repair*, either followed by the sequence $ContRepR_e$ and $ContRepS_e$, meaning that the repair has not completed in the current time unit, or by the sequence $EndRepR_e$, $EndRepS_e$ if the repair completes). Place $Assigned_e$ is set by the $PN^{nd}$ when a decision to repair $e$ is taken. The marking of places $AvRes_i$ represents the current available resources, and the multiplicity of their input arcs the resources released when the repair ends. A token in place $NotInv_e$ means that the component corresponding to the BE is currently not involved in any repair action. The *fprob* and *rprob* attributes associated with the events are used to properly weight the transitions representing a failure occurrence and end/continuation of repair actions: *fprob* is associated with transition $FailR_e$, $1 - fprob$ is associated with transition $WorkR_e$, *rprob* (representing the probability of continuing to repair) is associated with transition $ContRepR_e$, finally $1 - rprob$ is associated with transition $EndRepR_e$.

Observe that the only effective conflicts to be resolved on the $PN^{pr}$ model are the following free choice conflicts: $WorkR_e$ vs. $FailR_e$ (for each basic event), $ContRepR_e$ vs. $EndRepR_e$ (for each locally repairable basic event), plus the free choice conflict $ContRepGR_e$ vs. $EndRepGR_e$ for each global repair action (whose translation pattern is commented hereafter). Hence the weights assigned to all other transitions are irrelevant, since they will eventually fire once enabled (i.e. their firing never resolves a conflict).

Let us now discuss the translation pattern ensuring the propagation of the state from basic to internal events, and of the global repair actions, associated with some internal event. The conversion rule for an AND/OR gate corresponding to a given internal event $e$ is shown in Figs. 3 and 4. Subnets **C** and **E** simply model the propagation of the failures from the input events of the gate to its output event. These patterns are actually "templates" that must be instantiated according to the set of inputs of the AND/OR gate, which in general includes a subset of internal events, and a subset of basic events. The "run" transition $And_e$ can fire iff all the corresponding events $e_i$ are down; while each "run" transition $OrR_i$ can fire iff the corresponding event $e_i$ is down. Observe that $e_i$ can be a basic event or an internal one.

All these state propagation transitions must fire before the firing of "stop" transitions $AndS_e$ and $OrS_e$, so that their priority $prioSI1_e$ is the lowest one.

Subnets **D** and **F**, shown on the right of Figs. 3 and 4, model the propagation of the failures from the input events

of the gate to its output event and the associated global repair process. In particular, place $IdleSupervisor_e$ is marked when no global repair process involving the supervisor internal event $e$ has started yet. The start of a global repair process, represented by the firing of the "run" transition $RepairG_e$, involving component $e$, is enabled when the $Assigned_e$ place is marked (indicating that the $PN^{nd}$ subnet, in the previous ND step, has decided to assign the required resources for such a global repair process to supervisor $e$). The firing of "stop" transition $NoGS_e$, instead, means that no global repair supervised by $e$ will start in the current time step. Observe that this transition has lower priority than $RepairGR_e$, hence the repair process starts as soon as the required resources have been assigned to supervisor $e$. Place $UnderRepair_e$ represents the fact that the global repair process supervised by $e$ is ongoing: if the "run" transition $ContRepGR_e$ fires, followed by the "stop" transition $ContRepGS_e$, the repair process will not end in the current time step, while if the "run" transition $EndRepGR_e$ fires (setting the resources free), followed by the "stop" transition $EndRepGS_e$, the global repair process will end in the current time step (all the above mentioned transitions involve component $e$): this triggers the firing of the transitions $ResetR_1$ and $ResetR_n$ (all involving basic component $e_i$ in the set of basic events supervised by $e$ in the global repair process), ensuring that all basic events involved in the repair process are reset to the $Up$ state (place $Up_i$ marked) and the corresponding $NotInv_e$ place is marked again.

The translation algorithm visits all the events in the NdRFT and generates an appropriate PN submodel for each of them (the selection of the appropriate PN submodel follows the indications depicted in the template figures). Finally all submodels are composed by merging the places with identical label, leading to the entire probabilistic subnet of the MDPN. The complete specification of the translation algorithm from NdRFT to MDPN can be found in [30].

### 4.2. Generation of the ND subnet

The corresponding $PN^{nd}$ subnet is built from the template subnets depicted in Figs. 5 and 6. The basic idea is that the $PN^{nd}$ submodel must decide whether a repair action must be started for each down BE and for each self revealing internal event which may trigger a global repair process. For any repairable BE $e$ (corresponding to a controllable component in the MDPN), firing of stop transition $NotRepair_e$, involving only component $e$, means that a non repair decision has been taken for event $e$, while firing of stop transition $Repair_e$, also involving only component $e$, corresponds to the opposite decision: observe that the second decision can be taken only if $e$ is self revealing and, in state $Down$, the needed resources are available (input places $AvRes_i$ contain enough tokens) and the event is not involved in any global repair process (input place $NotInv_e$ marked). In detail the subnet **H** models the non deterministic behavior of a not self revealing repairable BEs, while the subnet **G** models the non deterministic behavior of a self revealing repairable BEs.

The start of local repair actions triggered by self revealing internal events is modeled by subnet **L**, where it is possible to observe the repetition of subnet **G** for as many times as the number of local repairs potentially triggered by the internal event $e$ (the test arcs from place $OutComp_e$ to the $Repair_i$ transitions model the fact that the repair can start only if the internal event $e$ is $Down$). Finally the start of a global repair action triggered by an internal event $e$ is modeled by subnet **M**: observe that a global repair process requires a single set of resources, starts for the set of supervised BEs as a whole, and requires that none of the supervised BEs are involved in any other repair process; on the other hand a local repair action triggered by an internal event $e$ may start in different time steps for each basic event supervised by $e$ (as long as $e$ is still down and the conditions to start the local repair are satisfied). The two stop transitions $Repair_e$ and $NotRepair_e$ represent the two possible choices: each of them involves only component $e$.

Subnet **I** (as well as the $RUNGL_e$ transition in subnet **L**) is needed for technical reasons: it is used to "clear" the state of the internal events which must be recomputed at the end of each probabilistic step (after all fail/repair choices have been taken for all BEs and the continue/end of repair choices have been taken for all ongoing global repairs).

Again the final $PN^{nd}$ submodel is obtained by properly composing the subnets generated for each event in the NdRFT and the special transition $StopGL$ (subnet **M**) used to end the non deterministic phase of the global system. During the composition phase the places and the transitions with identical label are merged.

Finally the MDPN reward functions $rs()$ and $rt()$ have to be defined. They can be derived from the NdRFT $cost$ as follows: let $m_\rho$ denote a marking of the MDPN corresponding to state $\rho$ in the MDP defining the NdRFT semantics (it is shown in Appendix that there is a one-to-one correspondence between the states of the MDP defining the semantics of a given NdRFT and the non deterministic markings reached by a probabilistic step in the MDPN obtained by translation): $rs(m_\rho) = cost.state(FailEv(\rho), RepairEv(\rho))$ and $rt(Repair_e) = cost.event(e), rt(t) = 0, \forall t \in T^{nd} \setminus \{Repair_e : e \in \overline{\mathscr{E}}_{GR} \cup \underline{\mathscr{E}}_R\}$

In summary, the cost definition of a NdRFT can be automatically translated into the MDPN reward functions, and the optimization problem expressed on the NdRFT can be transposed to an equivalent optimization problem in the MDPN.

The correctness of this translation is proven in Appendix A.

### 4.3. Improving NdRFT efficiency

In this subsection we introduce two methods to improve the NdRFT efficiency in terms of reduction of the RG size.

**A transition priority assignment.** This method of associating priorities with the MDPN transitions reduces the possible interleavings corresponding to equivalent non deterministic or probabilistic firing sequences, so that it

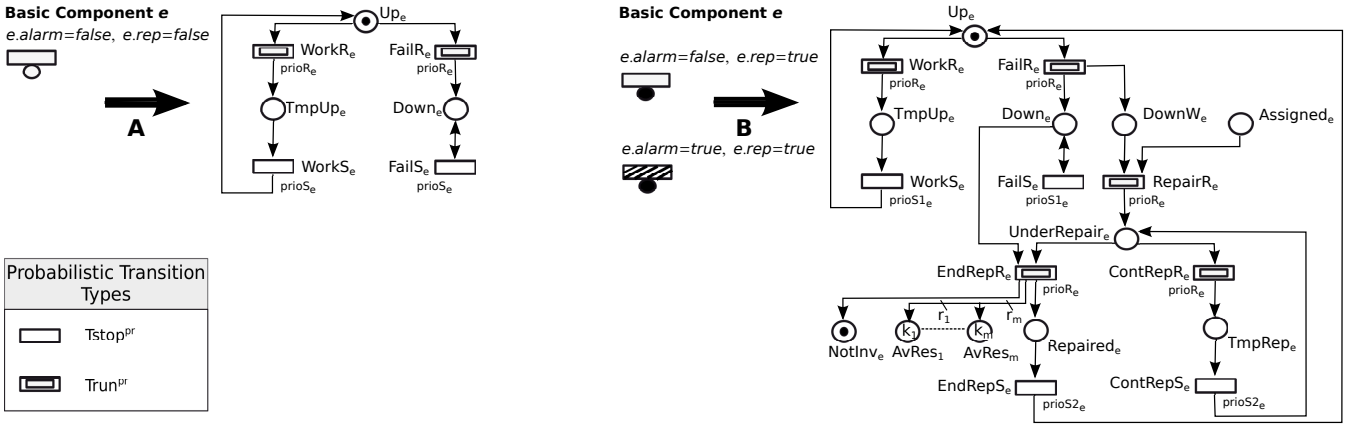


**FIGURE 2.** Translation of the NdRFT BEs into submodels of $PN^{pr}$ of an MDPN.

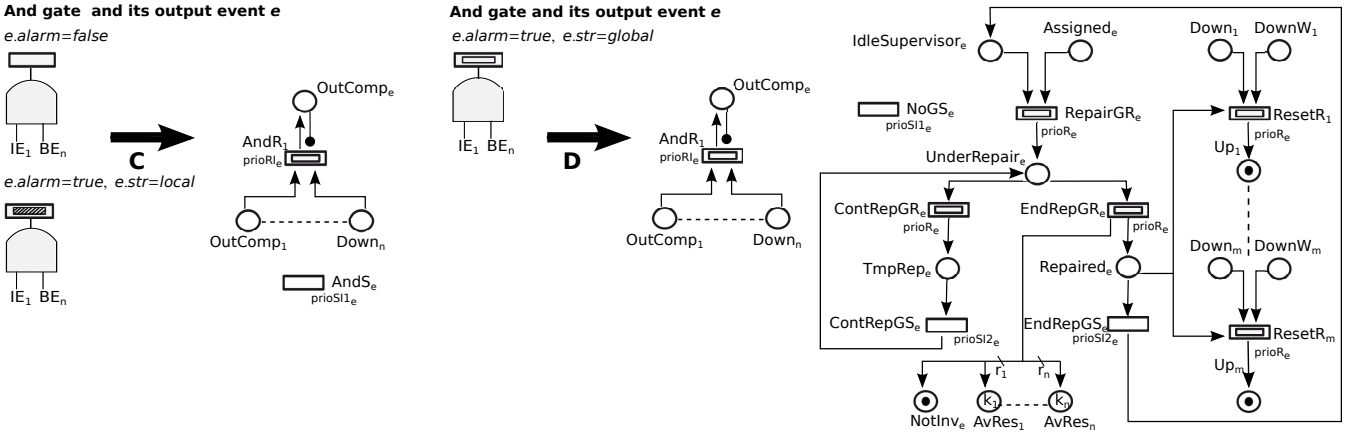


**FIGURE 3.** Translation of the NdRFT And gate plus its output event into submodels of $PN^{pr}$ of an MDPN.

allows a reduction of the number of states of RG[3]. The method requires fixing a total order on the NdRFT events that must be compatible with the partial order induced by the NdRFT structure that is based on the dependencies: the *TE* is the lowest among all the events and two events are in relation $e < e'$ if $e'$ is in the subtree of $e$. The total order can be specified through an injective function $(ord : \mathscr{E} \rightarrow \mathbb{N})$ so that $\forall e, e' e < e' \Rightarrow ord(e) < ord(e')$. Hence the priority assignment for the *PR* subnet is defined as follows:

$$\forall t_e \in T^{pr} \Rightarrow prio_{t_e} = PrioTemplate(t_e) + ord(e) * C_{pr}$$

where $PrioTemplate : T \rightarrow \{prioS1, prioS2, prioR, prioSI1, prioSI2, prioRI\}$ is a function which returns the template priority associated with a transition and $C_{pr}$ is a constant equal to $|\{prioS1, prioS2, prioR\}| = |\{prioSI1, prioSI2, prioRI\}| = 3$.

A similar priority assignment method is adopted for the *ND* subnet.

- $\forall t_e \in Tstop^{nd} \Rightarrow prio_{t_e} = ord(e)$, where $e$ is an event in $Comp^{nd}$;
- $STOPGL \Rightarrow prio_{STOPGL} = Max_{e \in \mathscr{E}}(ord(e)) + 1$ ;
- $\forall RunGL_e, RunGL_{e'} \in Trun^{nd} \Rightarrow prio_{RunGL_e}, prio_{RunGL_{e'}} > Max_{e \in \mathscr{E}}(ord(e)) + 1 \wedge prio_{RunGL_e} \neq prio_{RunGL_{e'}}$.

[3]Observe that the assignment of a different priority level to a pair of transitions that cannot be in conflict is irrelevant; on the other hand if the transitions are potentially in conflict then their priority assignment can constrain the set of possible strategies that will be considered at the MDP level and may exclude the optimal one.

The experiments presented in Sec. 6 have been performed by applying the priority assignment method described above.

**Replacing independent subtrees with BEs.** Another method to reduce the number of states of RG is based on replacing the NdRFT independent subtrees (modules) with BEs. In fact, such subtrees can be solved in isolation with the proper technique: combinatorial or state space analysis respectively. Then, they can be replaced by a BE with a properly computed failure and repair probability: the MDP can then be generated from this simplified NdRFT model. Unfortunately, the possibility to specify several repair options and to share repair resources among different repair actions, introduces strong dependencies among the events that cause state changes in the model, so that it is not so frequent that independent subtrees are present in the model. Anyway a subtree sharing no events with other subtrees can be a module in a NdRFT model, in (at least) the following particular situation: the subtree contains no repairable components and only OR gates.

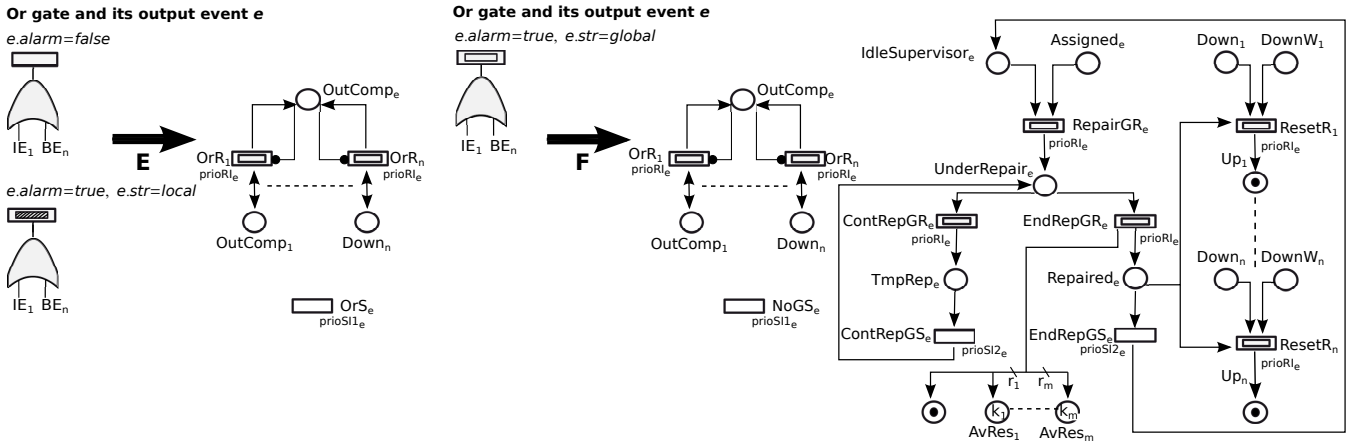


**FIGURE 4.** Translation of the NdRFT OR gate plus its output event into submodels of $PN^{pr}$ of an MDPN.

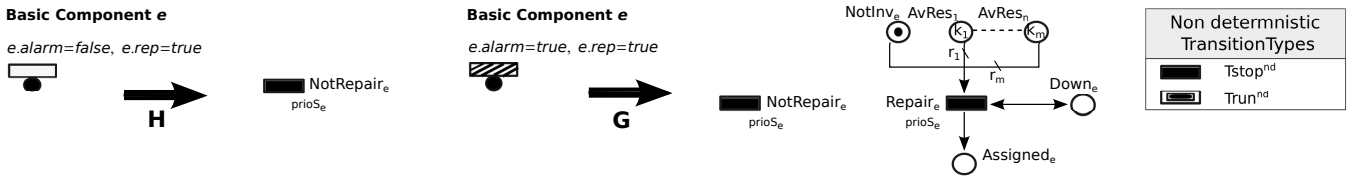


**FIGURE 5.** Translation of the NdRFT BEs into submodels of $PN^{nd}$ of an MDPN.

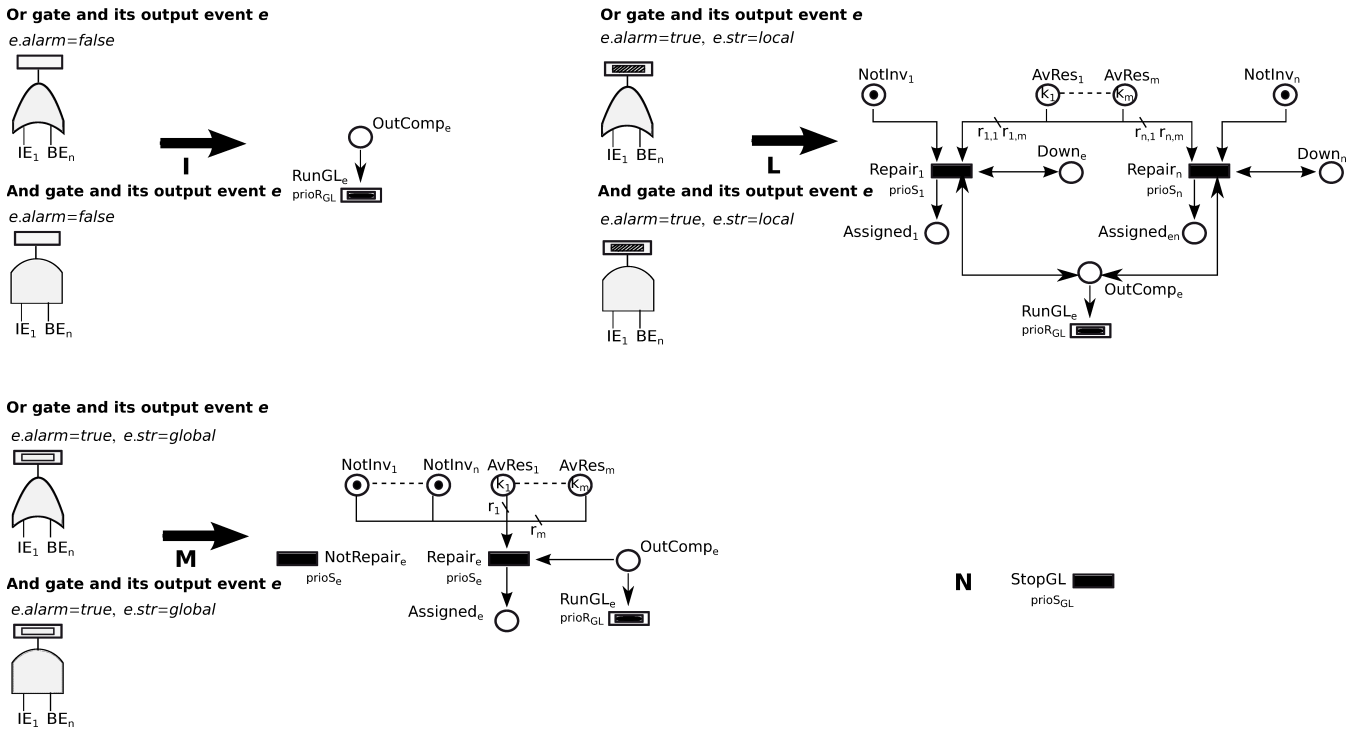


**FIGURE 6.** Translation of the NdRFT gate into submodels of $PN^{nd}$ of an MDPN.

## 5. AN NDRFT EXTENSION TO PROVIDE DEGRADATION AND PREVENTIVE MAINTENANCE

In this section we introduce a possible extension to the formalism which allows one to account for the components aging or the accumulation of corrupted data increasing the failure probability. This, combined with the possibility to add preventive maintenance, extends the possible strategies which can be studied with our proposed approach and

relaxes the constraint of using only geometric distribution to model the BE failure probability.

Such extension, thanks to the proposed modular translation technique can be smoothly introduced requiring only to locally update the BE definition and its (probabilistic and non deterministic) MDPN templates.

### 5.1. New definition for BE

To take into account the components degradation and aging, the following BE attributes must be introduced or updated:

- $D \in \mathbb{N}$: it is a new attribute defining the number of degradation/aging stages;
- $sprob : [0,\dots,D-1] \to [0,1]$: it is a new attribute associating each degradation/aging stage with the corresponding probability to move into the next stage remaining *Up*. Observe that $sprob(0)$ is the probability to move into the first degradation/aging stage.
- $fprob : [0,\dots,D] \to [0,1]$: this attribute is redefined so that it associates with each degradation/aging stage the corresponding failure probability. Observe that $fprob(0)$ is the failure probability when the BE is not degraded and $1 - e.fprob(i) - e.sprob(i)$ is the probability for a BE $e$ to remain *Up* in the stage $i$.

To account for preventive maintenance of a repairable BE $e$ (i.e. *e.rep=true*) it is necessary to introduce the following new attribute:

- $prev : [1,\dots,D] \to \{true, false\}$: it defines the degradation/aging stages in which a preventive maintenance may be executed.

In Table 4 all the attributes associated with BEs are summarized.

Before presenting the new BE translation, it is important to highlight that in this extension we do not consider long jumps between stages (i.e. stage $i+j$ with $j > 1$ is not directly reachable from stage i), however this constraint could be easily removed increasing the complexity of the BE definition and translation. The number of stages ($D$) has a considerable impact on the cost (memory and execution time) of the model solution since the RG of the composed model and the MDP size grows exponentially with respect to it. Instead, the number of stages enabling preventive maintenance can substantially impact the MDP solution cost since a larger number of possible repair strategies must be investigated.

### 5.2. New translation modules for BEs

Fig. 7 shows how each BE $e$ is translated in a $PN^{pr}$ submodel according to its *alarm* and *rep* attributes: each non repairable event is translated into subnet **A**, while each repairable event is translated into subnet **B**. In particular the submodels portion highlighted by a dashed box corresponds to a single degradation/aging stage, and it is repeated $e.D$ times. Hence, token in place $Up\text{-}i_e$ means that the BE $e$ is the $i^{th}$ degradation/aging stage.

The $fprob(i)$ and $sprob(i)$ with $i > 0$ are used to properly weight the transitions $FailR\text{-}i_e$ and $WorkMovS\text{-}i_e$ respectively; while the weight of transition $WorkR\text{-}i_e$ is $1 - e.fprob(i) - e.sprob(i)$.

Moreover for each degradation/aging stage in which a preventive maintenance may be executed we insert a transition $RepairMR\text{-}i_e$ connected in input to places $Up\text{-}i_e$ and $Assigned_e$ and in output to places $Down_e$ and $UnderRapair_e$.

Fig. 8 shows how each BE $e$ with preventive maintenance is translated in a $PN^{nd}$ submodel according to its attribute *alarm*: each repairable not self-revealing component is translated into subnet **C**, while each repairable self-revealing component is translated into subnet **D**. Practically with respect to the previous translation in Fig 5 a stop transition $RepairM\text{-}i_e$ is introduced for each stage in which is possible to activate a preventive maintenance. Observe that place $Up\text{-}i_e$ is connected with a test arc with $RepairM\text{-}i_e$ to assure that the preventive maintenance can be chosen only when the component $e$ is in the stage $i$.

## 6. APPLICATION EXAMPLES

In this section we describe two application examples. The first example is used to show how NdRFT allows us to efficiently investigate different optimal repair strategies by changing the associated cost function and how the model complexity depends on the repair options represented in the model. The second one instead, has been developed to present in details an example of optimal repair policy and to discuss a possible extension in the direction of preventive maintenance.

All the obtained results have been performed thanks to a framework for the design and the solution of NdRFT models that we have developed extending the one presented in [31].

**First application example.** The example we report is inspired to the *Active Heat Rejection System* (AHRS) presented in [32]. The block scheme of our version of the AHRS's architecture is composed by three redundant thermal rejection units $U1$, $U2$ and $U3$. Each $Ui$ is composed by the heat source $Ai$ and the power source $Pi$. The $Ui$ unit fails if its heat source $Ai$ has failed or if its power source $Pi$ has failed. The failure of the whole system ($TE$) occurs if all the thermal rejection units have failed.

The NdRFT model in Fig. 1 shows that in our version of the AHRS, several components are repairable ($A1$, $P1$, $A2$, $P2$), where their failures can be self-revealing or not. Two repair processes can be activated: 1) a global repair process in case of the failure of $U2$ and involving the components $A2$ and $P2$; 2) a local repair process in case of the system failure ($TE$) and involving the components $A1$, $P1$, $A2$ and $P2$ or in case of failure of the $A1$ (resp. $P1$) and involving the component $A1$ (resp. $P1$). In case of global repair, one repair resource is used to repair the subsystem; instead in case of local repair, one resource has to be dedicated to the repair of each component of the system. We suppose that in

**TABLE 4.** Attributes associated with a BE

| Name | Values | Description |
|---|---|---|
| D | $\mathbb{N}$ | number of degradation/aging stages. |
| *fprob(i)*, $i \in [0,\ldots,D]$ | $\mathbb{R}$ | failure probability associated with the stage *i*. |
| *sprob(i)*, $i \in [0,\ldots,D-1]$ | $\mathbb{R}$ | probability to move into the stage $i+1$ from stage *i* remaining *Up*. |
| *rep* | {*true*,*false*} | it indicates whether the event is locally repairable or not. |
| **If *rep*= *true* then the following attributes are defined** | | |
| *rprob* | $\mathbb{R}$ | probability to continue the current repair in the next time step. |
| *res(j)* | $\mathbb{N}$ | number of required resources j needed to trigger a repair process. |
| *prev(k)*, $k \in [1,\ldots,D]$ | {*true*,*false*} | it indicates if the preventive maintenance can be initiated in stage *k*. |

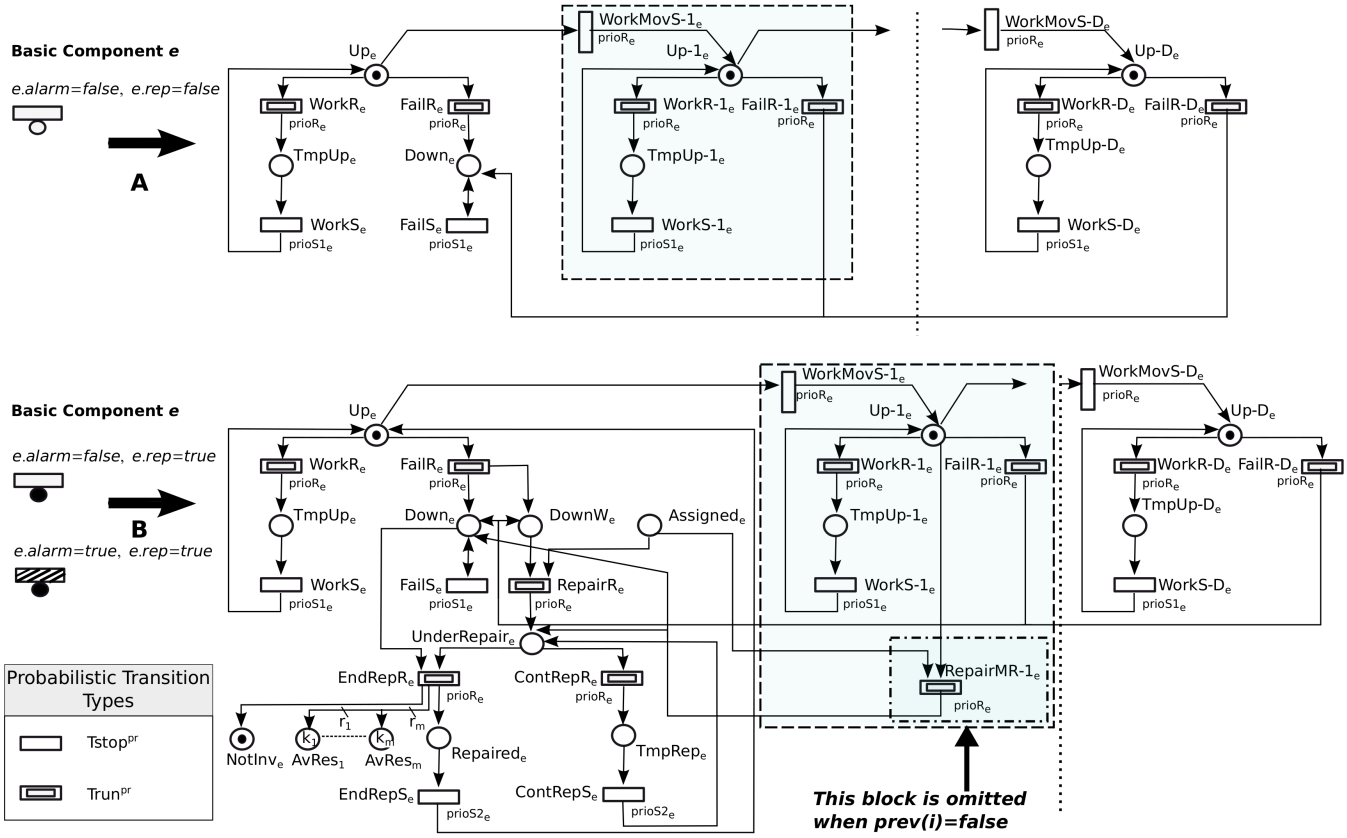


**FIGURE 7.** Translation of NdRFT BEs with degradation and preventive maintenance into submodels of $PN^{pr}$.

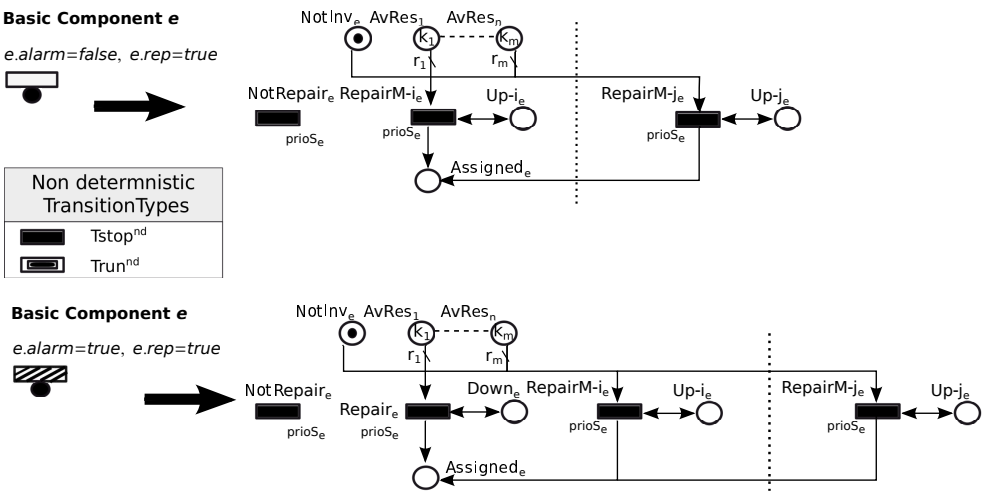


**FIGURE 8.** Translation of NdRFT BEs with degradation and preventive maintenance into submodels of $PN^{nd}$.

our case study, two repair resources are available (Fig. 1).

Two cost functions defined according to Eq. (2) in Sec. 3 have been studied: the former allows minimization of the *TE* probability:

$$\begin{aligned} & down_penalty = -1000 \\ & \forall e \in RepairEv, \\ & \quad rep_cost_per_time_unit(e) = start_repair_cost(e) = 0 \end{aligned}$$

while the latter allows the minimization of the average repair cost (including a penalty for *TE* being down), given the local and global start repair costs:

$$\begin{aligned} & down_penalty = -1000 \\ & \forall e \in RepairEv, \\ & \quad rep_cost_per_time_unit(e) = 0 \\ & \quad start_repair_cost(e) = \begin{cases} -1 & e.str = global \\ -100 & otherwise \end{cases} \end{aligned}$$

The RG of the MDPN model obtained by the NdRFT in Fig. 1 has 3,189 states, while the underlying MDP has 389 states. This difference in the number of states between the RG of the MDPN and the MDP is due to the fact that the MDPN formalism gives a component-based view of probabilistic and non deterministic behaviors of the system. At the MDPN level, complex non deterministic and probabilistic behaviors are expressed as a composition of simpler non deterministic or probabilistic steps, which are reduced to a single step in the final MDP.

Since the failure of the two non repairable components ($A3$ and $P3$) alone is not sufficient to induce the failure of the TE, the system can always come back to the up state (possibly in degraded mode); hence it is interesting to compute the average reward and the optimal strategy of the underlying MDP at infinite horizon. For the two considered cost functions, and the corresponding derived optimal strategy, we have computed the *TE* probability by solving the DTMC obtained from the underlying MDP assuming the optimal strategy is always applied. In particular we have obtained that the *TE* probability in steady state for the first case is 0.0151943, while in the second case the *TE* probability increases to 0.0161111 due to the fact that $P2$ and $A2$ will always be repaired by the global repair process triggered by $U2$, since it has a lower repair cost, but on the other hand $U2$ takes longer to complete the repair because it has a *rprob* higher than those of the $P2$ and $A2$ local repair processes.

For a further illustration of the potential of NdRFT we have performed two additional experiments: by manually modifying the MDPN model obtained from the NdRFT we have imposed the immediate repair of a component upon a failure occurrence (this corresponds to assigning a higher priority to transitions *Repair* than to transitions *NotRepair* in the non deterministic subnet, so that if there are available resources a repair cannot be postponed). In details, in the first experiment when more than one component is down the repair ordering is not fixed a priori, but it is dynamically computed when solving the MDP to minimize the TE probability; while in the second experiment the repair order is fixed a priori (i.e. $A1$, $A2-P2$, $P1$): this is implemented by imposing the same ordering to the priorities associated with the *Repair* transitions. As expected, the TE probability derived by solving the DTMCs implementing the computed repair strategies for these two experiments are greater (i.e. prob(TE) is 0.0161134 and 0.0920372 respectively) than the one obtained in the previous experiment in which we let the MDP compute the optimal repair strategy w.r.t. the TE probability minimization.

Tab. 5 shows some experiments in which the dimension of our example is increased. We have replicated the subtrees of the NdRFT model in Fig. 1.b. For instance, in Tab. 5, "$2,2,2$" means that we have duplicated the subtrees rooted in $U1$, $U2$, $U3$ respectively, while $1,1,2$ means that we have duplicated only the subtree of $U3$. In particular the first column shows the model complexity, the second and the third the RG size (number of states) and its computation time, the next two columns the MDP size (number of states) and its generation and solution time. The computation has been performed with an INTEL Centrino DUO 2.4 with 2Gb memory capacity. These results show that state space grows very fast when redundancy is increased, so that the model becomes quickly intractable even using the priority based optimization presented in Section 4. Moreover, we have to highlight that the state space growth depends also on the repair options (actions) applied in the model (Tab. 5). For case 2,2,2, if we remove the repair process triggered by the *TE* then the state space size decreases by factor $\sim 2$, while if we remove the global repair process triggered by $U2$ then it decreases by factor $\sim 4$.

**Second application example.** This second example is inspired by the *Multiprocessor system* presented in [3]. The system is composed by two units: the disk access (*DA*) and the computation unit (*CM*). The *DA* unit is composed by two disks ($D1$ and $D2$) in mirroring (RAID-1) connected through a bus (*DBUS*); while the *CM* unit is composed by two processing units: $PU1$ and $PU2$. Each processing unit includes a processor *Pi* and two banks of local memory *MEMi* composed by a memory (*RMi*) and its bus (*BMi*). Moreover, the two processing units share a global memory *SM* composed by two redundant memory banks *BRi*, each including a memory (*Ri*) and its bus (*Bi*)

Fig. 9A shows the NdRFT model for this system. The failure of the whole system (*TE*) occurs if the *DA* unit or the *CM* unit has failed. The *DA* unit fails if the two disks have failed or if its BUS has failed; while the *CM* fails if $PU1$ and $PU2$ have failed. Then, the failure of each $PUi$ occurs if its processor has failed or if its local memory banks and the global memory have failed. Finally *SM* fails if its two memory banks have failed or if its BUS has failed. The NdRFT model in Fig. 9 shows that in our version of multiprocessor system, five components are repairable ($R1$, $B1$, $D1$, $D2$, *DBUS*), so that their local repair processes can be activated in case of failure of *SM*, involving $R1$ and $B1$, in case of failure of *DA*, involving $D1$, $D2$ and *DBUS*, and in case of failure of $D1$ (resp. $D2$), involving $D1$ (resp. $D2$). In our case study we assume that only one repair resource is available and each repair process requires one resource to

**TABLE 5.** Experimental results.

| | Same repair policy of Fig. 1 | | | Without the repair processes triggered by *TE* | | | Without global repair processes triggered by $U2$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | RG | $MDP_{RG}$ | Time | RG | $MDP_{RG}$ | Time | RG | $MDP_{RG}$ | Time |
| Occurrences U1,U2,U3 | States | States | RG+MDP | States | States | RG+MDP | States | States | RG+MDP |
| 2,1,1 | 3.5E+4 | 937 | 12s | 2.7E+4 | 93 | 0s | 1.7E+4 | 122 | 0s |
| 2,2,1 | 4.5E+5 | 7,754 | 32m | 3.8E+5 | 483 | 30m | 2.5E+5 | 633 | 16m |
| 2,2,2 | 2.9E+6 | 32,558 | 236m | 1.7E+6 | 2,567 | 120m | 7.5E+5 | 3,005 | 70m |

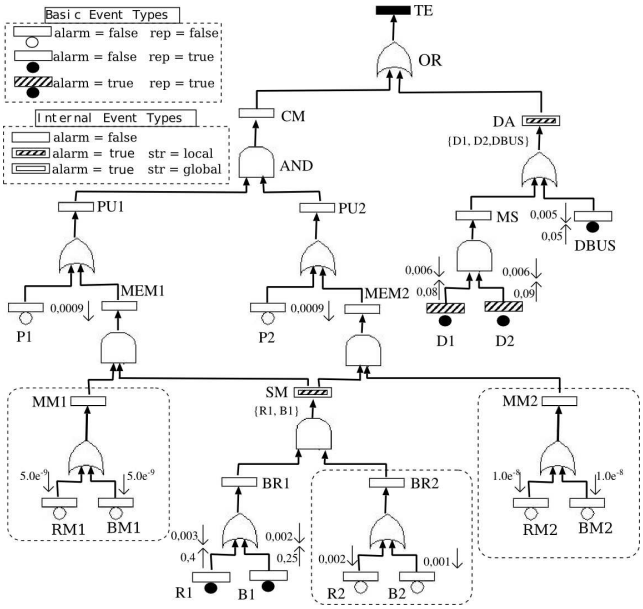


**FIGURE 9.** The NdRFT model of the multiprocessor system.

complete.

For this model we have computed the optimal repair policy that minimizes the *TE* probability at time $t$. The computation of the optimal repair policy has been performed in two steps: first the independent and not repairable subtrees $MM1, MM2$ and $BR2$ have been replaced with BEs as described in Section 4.3, then the simplified NdRFT has been solved. The RG of the MDPN model obtained by the simplified NdRFT has 586,826 states and it has been generated in 88 seconds on a Intel Centrino Duo 2.4Ghz; the underlying MDP has 8,875 states and it has been generated and solved in 697 seconds. The computed optimal repair policy is not trivial even if the system has only five repairable components, since when more repairable components have failed their repair order must be dynamically chosen according to the whole system state. The optimal repair policy can be synthesized as shown in Tab. 6, where the first three columns show the state of subsystems $CM$,$DA$,$SM$, and the last column shows the corresponding optimal repair order. For instance if all subsystems have failed then the optimal repair order is $B1, R1, DBUS, D1, D2$, otherwise if only $CM$ is working then the optimal repair order is $DBUS, D1, B1, R1, D2$.

In order to study this optimal repair strategy we have computed the corresponding *TE* probability at time $t$ by solving the DTMC obtained from the underlying MDP, fixing the action to take in every state according to the computed optimal strategy. We have compared this

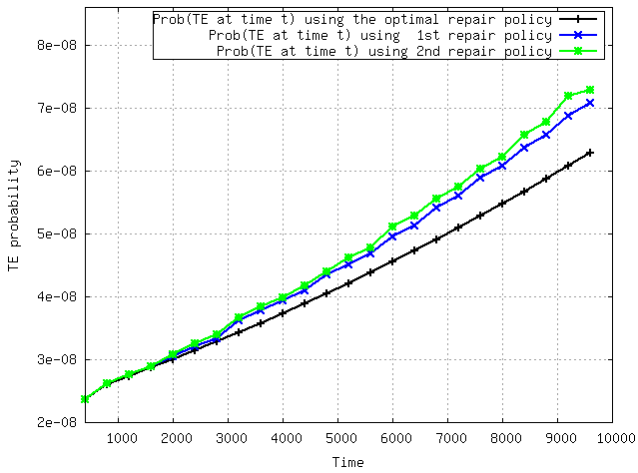


**FIGURE 10.** *TE* probability at time t ($400 \leq t \leq 9600$, step 400)

**TABLE 6.** The repair order suggested by the optimal repair policy

| CM | DA | SM | Repair order |
|---|---|---|---|
| Working | Failed | Failed/Working | DBUS,D1,D2,B1,R1 |
| Failed/Working | Working | Failed | B1,R1,D1,D2 |
| Working | Failed | Failed | DBUS,D1,B1,R1,D2 |
| Failed | Failed | Failed | B1,R1,DBUS,D1,D2 |

probability with those computed using the two following repair orders:1) always repair first all the failed components in subsystem $CM$; 2) always repair first all the failed components in subsystem $DA$. Graph 10 plots the obtained *TE* probabilities at time $t$ with $400 \leq t \leq 9600$ and highlights that the *TE* probability associated with the optimal strategy is lower than those computed according to the other repair policies (e.g when $t = 9600$ it is reduced by factor $\sim 1.20$ w.r.t the others).

Moreover, if we want to study the multiprocessor system taking into account the aging or the accumulation of corrupted data for $D1$ ad $D2$ components, then we need to consider the NdRFT extension introduced in Section 5 since the standard NdRFT does not allow us to model these aspects.

According to the new extension we consider for these two components two stages ($D1.D = D2.D = 2$) with the following associated *fprob*, *sprob* and *prev*:
$Di.fprob(0) = 0.006$ $Di.sprob(0) = 0.04$
$Di.fprob(1) = 0.012$ $Di.sprob(1) = 0.04$ $Di.prev(1) = true$
$Di.fprob(2) = 0.15$ $Di.prev(2) = true$
where $i \in \{1,2\}$.

Moreover, for each of these components we model the possibility of a preventive maintenance when they

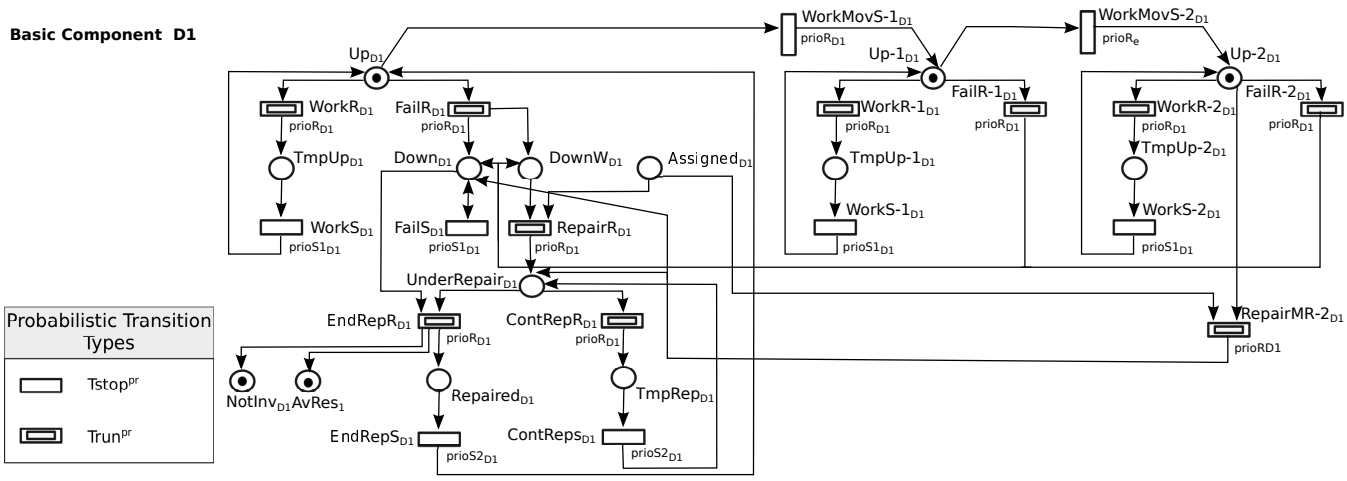


**FIGURE 11.** Translation of D1 with degradation and preventive maintenance into submodels of $PN^{pr}$ of an MDPN.

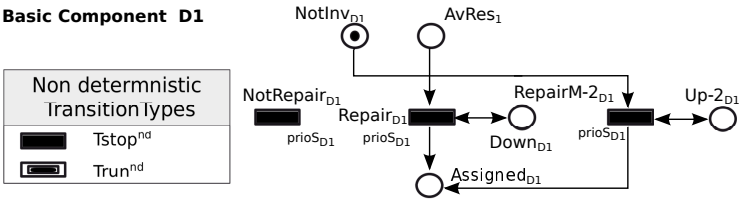


**FIGURE 12.** Translation of D1 with degradation and preventive maintenance into submodels of $PN^{nd}$ of an MDPN.

reach the last stage (i.e. $Up\text{-}2_{D1}$ and $Up\text{-}2_{D2}$ marked). Figures 11 and 12 show the probabilistic and non-deterministic translation for the D1 component. The translation for $D2$ can be directly derived by the one of D1 replacing in the transition, place and priority names $D1$ with $D2$.

The optimal strategy (minimizing the TE probability) without and with preventive maintenance are studied, and the results in Figure 13 shows, as aspected, that the TE probability obtained by applying the optimal strategy with preventive maintenance performs better than that obtained by only providing repair upon failure.

The size of MDPN model without preventive maintenance is increased by a factor $\sim 7.24$ (i.e. 4,250,598 states) with respect to the original one where the aging of component D1 and D2 is not modeled, and it has been generated in 398s. In the same way, its underlying MDP is greater than the original one by a factor $\sim 2.74$ (i.e. 24,353 states) and it has been generated and solved in 1,089s.

Instead, the stored space size of MDPN model with preventive maintenance is increased by a factor $\sim 7.29$ (i.e. 4,280,094 states) with respect to the original one, and it has been generated in 452s. Even if the underlying MDP has the same size of the one derived considering only the aging of component D1 and D2, its solution requires more time (1,389s.) since a greater number of possible repair strategies must be evaluated.

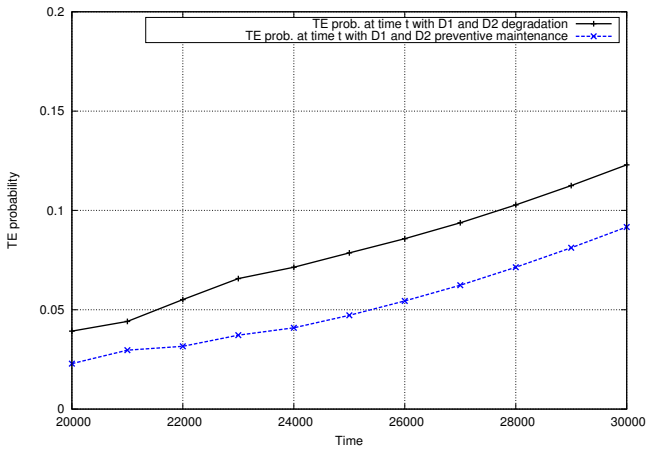


**FIGURE 13.** *TE* probability at time t ($20000 \leq t \leq 30000$, step 1000) assuming D1 and D2 degradation with/without preventive maintenance.

## 7. CONCLUSION AND FUTURE WORK

We have defined a new FT extension called NdRFT that allows us to model failures of complex systems as well as their repair processes. The originality of this formalism w.r.t. other proposals is that it allows us to manage repair strategy optimization problems. Therefore, NdRFT provides an optimal repair strategy that minimizes a given cost function defined by the modeler (e.g. minimize the *TE* probability or minimize the system repair cost).

This is done by defining the NdRFT semantics in terms of an MDP and then solving the optimization problem using the techniques available for MDPs. The reason for keeping

an FT-like formalism to specify the system structure is the fact that this is well established and (more) familiar to designers than MDP. The generation of the MDP is achieved by an intermediate translation of the NdRFT model into an MDPN, so that we can reuse the efficient algorithms devised to derive an MDP from an MDPN (and even more efficient ones when translating NdRFT with redundancy into Markov Decision Well-Formed Net MDWN [7]). The proposed translation is modular, which makes it easier to implement new features added in the formalism. Throughout the paper we have explained how NdRFT allows us to express in an elegant way several possible repair options based on the following concepts: self revealing events, the notion of local versus global repair action, the notion of repair supervisor component in the case of global repair. Components degradation as well as preventive maintenance of system components have also been considered, and a novel extension in this direction has been developed in details in Sec.5, and illustrated through a new example in Sec.6. This extension also shows the effectiveness of the proposed modular translation approach.

NdRFT allows us to represent systems with shared repair resources (possibly of different types) with a limited number of repair resources of each type and concurrent repairs: currently this is the only kind of external constraint included in the model; to the best of our knowledge the main FT extensions appearing in the literature do not take this constraint into account. In its current definition NdRFT looks for an optimal repair strategy assuming that the only constraint in deciding the start of a repair action is the availability of resources. The formalism could be rather easily extended to add other types of constraints like e.g. fixing some priorities among conflicting components when resources are assigned (this would actually reduce the space of possible policies to consider).

Some variations to the repair modes presented in this paper (local or global) will be considered in the future. In particular, in the current definition of the global repair of a subsystem, its components may be Up or Down during the repair. Instead, we may assume that all the subsystem components are considered as Down during the global repair (the global repair may be intended as the replacement of the subsystem). Such an assumption may influence the status (up or down) of some internal event of the NdRFT model, and consequently the possible triggering of other repair actions concerning components (basic events) not involved in the global repair.

Recently, we have proposed the Parametric NdRFT (ParNdRFT) [33], an extension of the NdRFT that automatically exploits the presence of redundancy in the system to reduce the complexity of the model and of its analysis. It is based on the translation of the ParNdRFT into a MDWN, i.e. a model specified by means of a High-Level Petri Net formalism. MDWN allows us to mitigate the state space explosion problem thanks to an existing algorithm that generates a reduced state space called Symbolic Reachability Graph (SRG) [34]. From the SRG it is possible to derive an MDP of reduced size w.r.t that obtained from the RG, and on which the same results can be computed more efficiently. This allows a computational cost reduction, without losing precision: in fact the optimal strategy computed on the reduced MDP is equivalent to the one computed on the ordinary MDP. This possibility is useful when the system is characterized by symmetries and redundancies in its structure.

The NdRFT formalism could be also extended by considering *dynamic gates* [2], which allow us to express functional and temporal dependencies among component failures, as well as preemption of repair resources. A particular dynamic gate allows us to model the presence of spare components able to replace the main ones if failed. The presence of spare components may be integrated in the repair mode: for instance, while a main component is under repair, it may be replaced in its function by a spare.

Finally we are evaluating the possibility of introducing in the NdRFT formalism the concept of *partially observable* state, which implies that the repair action choice is based on a partial knowledge of the system state: this requires to redefine the NdRFT semantics in terms of a Partially Observable MDP. This new feature would allow one to take decisions on the most promising repair actions even in case of incomplete failure identification (a situation that may arise when faults do not generate symptoms or when different failures generate the same symptoms).

## APPENDIX A. TRANSLATION CORRECTNESS

Let us prove that the MDPN obtained by applying the translation procedure described in Sec. 4 produces a Reachability Graph (RG) from which one can derive the MDP corresponding to the NdRFT semantics, defined in Section 3.2.

For this purpose we must define *maximal* non deterministic or probabilistic firing sequences of a MDPN.

DEFINITION A.1. *A maximal non deterministic firing sequence (MNDFS) is characterized by the following properties: (1) it starts either in the initial state or in a state reached by a maximal probabilistic firing sequence, (2) it contains exactly one stop transition for each controllable component, and one "global" stop transition.*

DEFINITION A.2. *A maximal probabilistic firing (MPRFS) sequence is characterized by the following properties: (1) it starts in a state reached by a maximal non deterministic firing sequence, (2) it contains exactly one stop transition for each component.*

In the sequel the correspondence between MDPN and MDP states, as well as the correspondence between MNDFS and MPRFS in the MDPN and MDP actions and probabilistic transitions, are stated and proven.

**MDPN states vs. MDP states.** First of all we need to define the correspondence between a subset of states appearing in the RG of the MDPN (both those reached immediately after the firing of an MPRFS and those reached immediately after an MNDFS, i.e. an action) and the MDP states (see Tab.3).

The state of each BE $e$ ($Up$, $Down$, $LocRep$, $GlobRep_*$) is

represented by the following places:
1. $st_e = Up$ if place $UP_e$ is marked;
2. $st_e = Down$ if place $DOWN_e$ is marked and place $NotInv_e$ is marked;
3. $st_e = LocRep$ if place $UnderRepair_e$ is marked;
4. $st_e = GlobRep_d$ ($st_e = GlobRep_u$) when place $DOWN_e$ ($UP_e$) is marked, places $UnderRepair_e$ and $NotInvolved_e$ are not marked; in this case there must exist exactly one internal event $e'$ s.t. $e'.obs = true$ and $e'.str = global$ and $e \in e'.torep$ and place $UnderRepair_{e'}$ is marked, so that $sup_e = e'$.
The $Up/Down$ state of internal events are derived according to the FT structure (represented by subnets C and E in Figs.3 and 4): an IE $e$ is Down if place $OUTCOMP_e$ is marked at the end of an MPRFS.

It is thus possible to establish a correspondence between each non deterministic marking $m$ (reached immediately after the firing of an MPRFS) of the MDPN and a state $\rho$ of the MDP: we use the notation $m_\rho$ to indicate a non deterministic marking of the MDPN corresponding to state $\rho$ of the MDP. Similarly it is possible to establish a correspondence between each intermediate state $\langle \rho, a \rangle$ of the MDP and a marking $m'$ reached immediately after the firing of an MNDFS of the MDPN. The set of resources in use, expressed by $res_\rho$ in the MDP, is represented in the MDPN by resource-indexed places $AV_RES_r$: the initial marking of $AV_RES_r$ corresponds to the multiplicity of resource $r$ in $res_0$, while the set $res_\rho$ of resources in use in state $\rho$ corresponds to: $res_\rho(r) = res_0(r) - m_\rho(AV_RES_r)$.

The initial marking, corresponding to the initial MDP state, has one token in each place $UP_e$ and as many tokens as the number of available resources of type $r$ in places $AV_RES_r$.

In order to prove that the translation is correct, we have to show that there is a one to one correspondence[4] between the actions $a \in A_\rho$ and the MNDFS $\sigma_a$ enabled in $m_\rho$, that the intermediate state $\langle \rho, a \rangle$ corresponds to the marking $m_{\langle \rho, a \rangle}$ reached by firing $\sigma_a$ in $m_\rho$. Moreover there is a correspondence between the states reachable from the intermediate state $\langle \rho, a \rangle$ and those reachable from $m_{\langle \rho, a \rangle}$ through some MPRFS. Finally the probability of transition $\langle \rho, a \rangle \to \rho'$ is equal to the sum of probabilities associated with the set of MPRFS leading from $m_{\langle \rho, a \rangle}$ to $m_{\rho'}$.

**MDPN non deterministic sequences vs. MDP actions.** From a non deterministic state of the MDPN, one or more alternative MNDFS may fire, each comprising exactly one stop transition for each controllable component (BE repairable event or IE event with a global repair strategy) plus one global stop transition: the combination of all stop transitions in each MNDFS defines a possible action at the MDP level, corresponding to the set of decisions - start repair of component $e$ ("stop" transition $Repair_e$) or do not start repair of component $e$ ("stop" transition $NotRepair_e$) - for each controllable component.

The set of decisions characterizing a specific action $a$ causes a state change (corresponding to the Non Deterministic step described in Section 3.2) witnessed by the marking of the $Assigned_e$ places in the MDPN at the end of the corresponding MNDFS $\sigma_a$. Observe that the conditions expressed in Section 3.2 for moving a BE $e$ from the $Down$ state to the $LocRep$ state or for moving a set of BE to their current state to the $GlobRep_*$ state (provided they are in the $torep$ set of a $Down$ IE $e'$) correspond to the conditions for firing transitions $Repair_e$ or $Repair_{e'}$ in the $PN^{nd}$ subnet.

Indeed, if we consider subnet G in Figure 5, corresponding to a self revealing and repairable BE $e$, a decision to start (local) repair may be taken if (1) $e$ is in state $Down$, (2) the required resources are available, and (3) the component is not yet involved in any other repair action. As an alternative, if the BE $e$ is repairable but not self revealing, and it is in the $torep$ set of some internal event $e'$ with a local repair strategy, then the local repair can start if (1) both the BE $e$ and the internal event $e'$ are $Down$, (2) the required resources are available, and (3) $e$ is not yet involved in any other repair action: this is modeled by subnet L in Figure 6.

The state change from state $Down$ to state $GlobRep_*$ instead is modeled by subnet M in Figure 6, and corresponds to the firing of the stop transition $Repair_e$ where $e$ is a self revealing internal event with associated global repair strategy: this transition may occur only when IE $e$ is in state $Down$, the resources needed for the global repair are all available, and none of the BE in $e.torep$ are involved in any other repair process (places $NotInvolved_{ei}$ marked). Observe that the start of global repair for internal event $e$ actually causes all the BEs in $e.torep$ to switch to the $GlobRep_*$ state simultaneously.

Since a decision is necessarily taken for any controllable component (exactly one stop transition must fire for each controllable component in any MNDFS), and since for each controllable event it is always possible to take a $NotRepair$ decision, and if the state allows so it is also possible to take the alternative $Repair$ decision, then all possible combination of start/do not start repair decisions corresponding to the allowed actions in the MDP can be obtained, and due to the conditions on the $Repair$ transitions no combination of decisions corresponding to an impossible action can be fired in the MDPN.

**MDPN probabilistic sequences vs. MDP probabilistic state change following an action.** After each MDP action a probabilistic state change occurs: in the MDPN this corresponds to the MPRFS that may follow an MNDFS. The probability of each path is obtained as the product of the probability associated with each transition in the path. Observe that a probabilistic path can be described as the interleaving of several subpaths, one for each component $e$ represented in the MDPN, and ending with a stop transition involving $e$. The transitions firing in each subpath depend on the initial status of the component:
1) if $e$ is $Up$ (place $UP_e$ marked) and not involved in any global repair action (place $NotInvolved_e$ marked) then either the subpath contains the sequence $WorkR_e$, $WorkS_e$, or it contains the sequence $FailR_e$, $FailS_e$: the former does not cause a state change for $e$, while the second corresponds

[4]The correspondence is one to one assuming that the transition priorities are set as explained in Sec.4.3.

to a change from *Up* to *Down*. Observe that if $e$ is *Up* but it is involved in a global repair process, then the subpath for $e$ depends on the probabilistic evolution of its supervisor, hence this case will be discussed together with such evolution;
2) if $e$ is *Down* (place $DOWN_e$ marked) and not involved in any repair action (place $NotInvolved_e$ marked), the subpath shall include only the stop transition $FailS_e$, which does not cause any state change ($e$ remains *Down*);
3) if $e$ is in *LocRep*, which includes also the case of place $Assigned_e$ just marked by the last non deterministic sequence (hence allowing run transition $Repair_e$ to fire, thus marking the $UnderRepair_e$ place) then either the repair process continues (sequence $ContRepR_e$,$ContRepS_e$), thus leaving the component in *LocRep*, or the repair process ends (sequence $EndRepR_e$,$EndRepS_e$), which causes $e$ to come back to *Up*;
4) global repair processes influence the state of the supervised basic events; if a given internal event $e'$ with global repair strategy is *Down*, and the basic events in $e'.torep$ are not involved in any repair process, it can be assigned the resources for the corresponding global repair to start (place $Assigned_{e'}$ marked at the end of an MNDFS): as a consequence the global repair process can start (transition $RepairG_{e'}$, marking the place $UnderRepair_{e'}$), causing a state change of the BEs in $e'.torep$ to $GlobRep_*$ (where $*$ stands for $u$ or $d$ depending on the previous BE status). As in the case of local repair, the *GlobRep* state is an intermediate one, which can become stable if the repair does not end in the current time step (transitions $ContRepGR_{e'}$, $ContRepGS_{e'}$), while in the case in which the repair process ends (transitions $EndRepGR_{e'}$, $EndRepGS_{e'}$) then all the BEs supervised by $e'$ are reset to the *Up* state: this is achieved by firing the transitions $ResetR_{ei}$, $ResetS_{ei}$, or transitions $FreeR_{ei}$, $FreeS_{ei}$ (the last two transitions fire in the case in which $UP_{e_i}$ is already marked, to reset the marking of $NotInvolved_{ei}$). In all cases all BEs supervised by $e'$ switch from *Up*/*Down* to $GlobRep_*$ simultaneously, and from $GlobRep_*$ to *Up* (or from *Down* to *Up* if the repair process lasts only one time step) simultaneously.

It may be the case that while a global repair process is ongoing, some of the BEs in $e'.torep$ that have not yet failed, fail in the current time step (transitions $FailR_{e_i}$, $FailS_{e_i}$ may fire if the choice of continuing the repair supervised by $e'$ has already been taken, i.e. if transitions $ContRepGR_{e'}$, which has priority $prio_5$ has already fired). In this case their state changes from *Up* to $GlobRep_d$ (with a not observable passage through the *Down* state), and will be reset to the *Up* state as soon as the global repair process ends (which might happen in the same time unit). The possible state changes described above for each component, are exactly the same as illustrated in the probabilistic step of the MDP semantics of the NdRFT (see Section 3.2).

The probability of each possible MPRFS is obtained as a product of the normalized weight of the enabled transitions. Transitions corresponding to different components are never in conflict (the failure or end-repair choice of one component cannot influence the choice of any other component, by construction): this means that independently of the chosen interleaving order of the components, the overall probability of moving from a given marking $m$ to a new marking $m'$ only depends on the failure probability of the *Up* components, and on the end-repair probability of the components under repair. If the priorities are set so that a specific order is forced in the MDPN there will be a single MPRFS leading from a given state $m$ (corresponding to a MDP intermediate state $\langle \rho, a \rangle$) to a new marking $m'$ (corresponding to a MDP state $\rho'$), and its probability will be exactly the same as that of the probabilistic step from $\langle \rho, a \rangle$ to $\rho'$. If instead priorities are set so that alternative interleavings may be chosen, leading from $m$ to $m'$, the final result will not change: indeed the sum of the probabilities of the set of MPRFS leading from $m$ to $m'$ can be expressed as the product of the probabilities of the choices taken in each component (which are necessarily the same since the initial and final markings are the same) multiplied by a summation of probabilities that sum up to one (these are the relative weights of the possible interleaving, which eventually converge to the same final state).

Finally observe that each MPRFS comprises a subsequence of $AND_e$ and $OR_e$ transitions, which are needed to propagate the correct *Up* or *Down* state (place $OUTCOMP_e$ unmarked or marked respectively) of all IEs. This subsequence is deterministic (although different interleavings could be possible depending on the priority assignment) since it depends only on the state of the BEs, and hence it contributes as a factor 1 to the probability product. The priorities of these transitions are set so that they are fired after all probabilistic choices have been made, but before the firing of the stop transitions of all components.

This completes the proof. In fact, from the initial marking the set of possible actions in the MDP are in one to one correspondence with the MNDFS of the MDPN, the reached intermediate states are in one to one correspondence, and from these the same probabilistic state changes may occur, leading to corresponding new states. This also indicates how the MDP can be derived from the MDPN RG, only the markings from which maximal firing sequences are originated are kept, and the maximal firing sequences are substituted with the corresponding transitions in the MDP. This translation is performed in linear time w.r.t. the number of NdRFT nodes.

## ACKNOWLEDGMENTS

The work of Marco Beccuti has been supported in part by project ”AMALFI - Advanced Methodologies for the Analysis and management of Future Internet” sponsored by Universitá di Torino and Compagnia di San Paolo, and by project grant Nr. 10-15-1432/HICI from the King Abdulaziz University of Saudi Arabia.

## REFERENCES

[1] Schneeweiss, W. (1999) *The Fault Tree Method*. LiLoLe Verlag, Hagen, Germany.

[2] Manian, R., Coppit, D., Sullivan, K., and Dugan, J. (1999) Bridging the Gap Between Systems and Dynamic Fault Tree Models. *Annual Reliability and Maintainability Symposium*, Washington DC, USA, Jan, pp. 105–111. IEEE.

[3] Bobbio, A., Franceschinis, G., Gaeta, R., and Portinale, G. (2003) Parametric fault tree for the dependability analysis of redundant systems and its high-level Petri net semantics. *IEEE Transactions on Software Engineering*, **29(3)**, 270–287.

[4] Codetta-Raiteri, D., Franceschinis, G., Iacono, M., and Vittorini, V. (2004) Repairable Fault Tree for the automatic evaluation of repair policies. *Int. Conf. on Dependable Systems and Networks*, Florence, Italy, June, pp. 659–668. .

[5] Beccuti, M., Codetta-Raiteri, D., Franceschinis, G., and Haddad, S. (2008) Non deterministic Repairable Fault Trees for computing optimal repair strategy. *Proc. of the 3rd Int. Conf. on Performance Evaluation, Methodologies and Tools*, Athens, Greece, October, pp. 43–53. ICST.

[6] Puterman, M. (1994) *Markov decision processes : Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York, NY, USA.

[7] Beccuti, M., Franceschinis, G., and Haddad, S. (2007) Markov Decision Petri Net and Markov Decision Well-Formed Net Formalisms. *28th Int. Conf. of Applications and Theory of Petri Nets, on Lecture Notes in Computer Science*, **4546**, 43–62.

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

[9] Contini, S. (1999) *ASTRA - Advanced Software Tool for Reliability Analysis, Theoretical Manual*. European Commission Joint Research Centre (JRC). Ispra, Italy, http://www.jrc.ec.europa.eu.

[10] Sahner, R. A., Trivedi, K. S., and Puliafito, A. (1996) *Performance and Reliability Analysis of Computer Systems; An Example-based Approach Using the SHARPE Software Package*. Kluwer Academic, Boston, USA.

[11] Ajmone-Marsan, M., Balbo, G., Conte, G., Donatelli, S., and Franceschinis, G. (1995) *Modelling with Generalized Stochastic Petri Nets*. Wiley Series in Parallel Computing, John Wiley and Sons, New York, NY, USA.

[12] Krishnamurthi, G., Gupta, A., and Somani, A. K. (1998) HIMAP: Architecture, Features, and Hierarchical Model Specification Techniques. *Proc. of the 10th Int. Conf. on Computer Performance Evaluation: Modelling Techniques and Tools on LN in Computer Science*, Heidelberg, Germany., June, pp. 348–351. Springer Berlin.

[13] A. Anand and A. K. Somani (1998) Hierarchical Analysis of Fault Trees with Dependencies, Using Decomposition. *Annual Reliability and Maintainability Symposium*, Anaheim, CA, USA, Jan, pp. 69–75. IEEE.

[14] Dugan, J., Bavuso, S., and Boyd, M. (1992) Dynamic Fault-Tree Models for Fault-Tolerant Computer System. *IEEE Transactions on Reliability*, **41**, 363–377.

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

[16] Boudali, H., Crouzen, P., and Stoelinga, M. (2007) Dynamic Fault Tree Analysis Using Input/Output Interactive Markov Chains. *Int. Conf. on Dependable Systems and Networks*, Edinburgh, UK, June, pp. 708–717. IEEE Computer Society.

[17] Distefano, S. and Puliafito, A. (2009) Dependability evaluation with dynamic reliability block diagrams and dynamic fault trees. *IEEE Transactions on Dependable and Secure Computing*, **6**, 4–17.

[18] Flammini, F., Mazzocca, N., Iacono, M., and Marrone, S. (2005) Using repairable fault trees for the evaluation of design choices for critical repairable systems. *Proc. of the 9th Int. Symposium on High-Assurance Systems Engineering*, Heidelberg, Germany, October, pp. 163–172. IEEE Computer Society.

[19] Castroa, I. and Sanjuan, E. (2008) An optimal repair policy for systems with a limited number of repairs. *European Journal of Operational Research*, **187**, 84–97.

[20] Righter, R. (2002) Optimal maintenance and operation of a system with backup components. *Probability in the Engineering and Informational Sciences*, **16**, 339–349.

[21] Wang, G. J. and Zhang, Y. L. (2006) Optimal periodic preventive repair and replacement policy assuming geometric process repair. *IEEE Transactions on Reliability*, **55**, 118–122.

[22] P. Weber and L. Jouffe (2003) Reliability modelling with dynamic Bayesian networks. *SafeProcess 2003, 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Washington DC, June, pp. 1–8. .

[23] Montani, S., Portinale, L., Bobbio, A., Varesio, M., and Codetta-Raiteri, D. (2006) A tool for automatically translating Dynamic Fault Trees into Dynamic Bayesian Networks. *Proc. of the Annual Reliability and Maintainability Symposium*, Newport Beach, CA USA, January, pp. 434–441. IEEE.

[24] L. Portinale and D. Codetta-Raiteri and S. Montani (2010) Supporting Reliability Engineers in Exploiting the Power of Dynamic Bayesian Networks. *International Journal of Approximate Reasoning*, **51**, 179–195.

[25] Dean, T. and Kanazawa, K. (1989) A model for projection and action. *Int. Conf. on Artificial Intelligence*, Detroit, USA, August, pp. 985–990. Morgan Kaufmann.

[26] Dean, T. and Wellman, M. P. (1991) *Planning and Control*. Morgan Kaufmann Pub., San Francisco, California, USA.

[27] Lovejoy, W. S. (1991) A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, **28**, 47–66.

[28] Koller, D. and Parr, R. (2000) Policy Iteration for Factored MDP. *Proc. of the Annual Conf. on Uncertainty in Artificial Intelligence*, Stanford, CA USA, June, pp. 326–334. Morgan Kaufmann.

[29] Bracquemond, C. and Gaudoin, O. (2004) A survey on discrete lifetime distributions. *International Journal on Reliability, Quality and Safety Engineering*, **10(1)**, 69–98.

[30] M. Beccuti and G. Franceschinis and D. Codetta-Raiteri and S. Haddad (2008) Non deterministic Repairable Fault Trees for computing optimal repair strategy. Technical Report TR-INF-2008-07-05. Dip. di Informatica, Univ. del Piemonte Orientale, Alessandria, Italy.

[31] Beccuti, M., Codetta-Raiteri, D., Franceschinis, G., and Haddad, S. (2007) A framework to design and solve Markov Decision Well-formed Net models. *Proc. of the 4th IEEE Int. Conf. on Quantitative Evaluation of Systems (QEST'07)*, Edinburgh, Scotland, UK, September, pp. 165–166. IEEE Computer Society.

[32] Assaf, T. and Dugan, J. B. (January 2004) Diagnostic Expert Systems from Dynamic Fault Trees. *Annual Reliability and Maintainability Symposium 2004 Proceedings*, Los Angeles, CA USA, Jun, pp. 444–450. IEEE Computer Society.

[33] Beccuti, M., Franceschinis, G., Codetta-Raiteri, D., and Haddad, S. (2009) Parametric NdRFT for the derivation of optimal repair strategies. *Proc. of the 39th Int. Conf. on Dependable Systems and Networks (DSN-2009)*, Estoril, Lisbon, Portugal, June, pp. 399–408. IEEE Computer Society.

[34] Chiola, G., Dutheillet, C., Franceschinis, G., and Haddad, S. (1993) Stochastic well-formed coloured nets for symmetric modelling applications. *IEEE Transactions on Computers*, **42**, 1343–1360.