2014 IEEE 26th International Conference on Tools with Artificial Intelligence

On Diagnosis of Violations of Constraints in Petri Net Models of Discrete Event Systems

Behzad Bordbar, Ahmed Al-Ajeli and Mohammed Alodib

***Abstract*— Failure detection in partially observable model-based Discrete Event Systems requires modelling failures as unobservable events within the system. Representing failures as events is not always realistic. For example, some classes of failure are in form of violations of constraints such as Service Level Agreement (SLA) and Quality of Service (QoS). These forms of failures do not represent events by themselves. They have to be modelled as additional events. Modifying the plant model is not always acceptable. Firstly, this may make the models large, causing extra computational complexity. Secondly, adding extra transitions is not always acceptable from engineers' perspective, because these constraints may change over the time leading to alternations of models every time these constraints are changed. To address this issue, this paper presents a new definition of diagnosability which extends the existing definition. In the new definition, a formalism has been introduced which captures failures as logical constraints instead of events. We show that starting from a Petri net, if the failure is expressed in Yen's logic, we can create a new Petri net with additional transitions, including transitions modelling failure, such that detection of violation of the constraint in the first Petri net is converted to diagnosis of failure in the second.**

I. INTRODUCTION

Automata and Petri nets are two common modeling languages used in model-based diagnosis of failure in Discrete Event Systems (DESs) [1]–[5]. A common practice is to represent failures as a part of the plant's model. For example, in Automata and Petri nets models of the plants, we create unobservable transitions for representing failure. However, this style of the modelling of failure is not always realistic. Sometimes failure is created as a result of violation of Service Level Agreement (SLA) or Quality of Service (QoS). For example, consider the so-called Right-First Time (RFT) failure [6] which is of interest to telecommunication services. Right-First Time (RFT) failure occurs when a process fails to complete a task First-Time and it is forced to repeat a part of the task again. This happens when one or more tasks are repeated, indicating incorrect execution of the task in the first place. Such occurrences of failure may result in violations of Service Level Agreement (SLA), causing financial penalties or customer dissatisfaction.

If the failure is expressed as a constraint, there is no event in the system that represents failure. One can argue that if a failure is caused by a violation of a constraint, we can always modify the model of the plant to include extra transitions (or/and states) to model the occurrences of the failure. This would require alterations of the models, which in our experience, is not always acceptable by the engineers. Since the SLA and QoS requirements change over time, if violations of such constraints are modelled by adding transitions, the model of the plant must change whenever such constraints are modified. In addition, in some cases, adding extra events or transitions may result in cumbersome models. To model RFT failure, potentially duplicates of many transitions must be created to mark undesirable repetition of the multiple events. This can result in a serious distortion of an originally elegant design, resulting in a large and complex model.

To address this issue, we shall extend the existing diagnosability theory by presenting a new formalism in which failures are *not* captured as events. Motivated by the use of SLA and QoS, we shall model failure as Yen's logic statement that represents violation of the constraints. Our contributions consist of introducing a new definition of diagnosability in which a failure is modelled as logical constraints. We show that this new definition is an extension of the existing definition [1], [3] of diagnosability. In this work, we prove that the question of diagnosability and designing diagnosers to detect a violation of such constraints in Petri nets can be converted to the same question for the existing definitions. Namely, for a Petri net $\mathcal{N}$ and a constraint c which, if satisfied, a failure f has happened, we can create another Petri net $\mathcal{N}'$ with an extra transition modelling f, such that detecting the failure in this Petri net implies the violation of the constraint has occurred in $\mathcal{N}$. We show that if we can produce a diagnoser for detecting failure f in $\mathcal{N}'$, then that diagnoser can be used to detect a violation of c in $\mathcal{N}$.

This paper is organized as follows. Section II presents a short review of the Petri nets' theory. Diagnosability theory and Yen's logic are discussed in III and IV respectively. In Section V we shall describe our running example. Following that we shall formulate a definition of diagnosability which extends the existing definitions. In Section VI we shall present our method of diagnosis of violation of constraints. An example of applying this method for diagnosis of violation of constraints is presented in section VII. Related work will be the subject of section VIII

II. BACKGROUND ON PETRI NETS

A *Petri net* is a four tuple $\mathcal{N} = (P, T, pre, post)$, P is a finite set of places, T is a finite set of transitions, $pre : P \times T \to \mathbb{N}$ and $post : P \times T \to \mathbb{N}$ [7]. In this paper

This work was supported by School of Computer Science, University of Birmingham, Birmingham B15 2TT, United Kingdom.

Behzad Bordbar and Ahmed Al-Ajeli are with school of Computer Science, University of Birmingham, United Kingdom {B.Bordbar, A.K.O.Al-Ajeli}@cs.bham.ac.uk

Mohammed Alodib is with Qassim University, Buraidah 51411, Qassim, Saudi Arabia alodib@qu.edu.sa

1082-3409/14 $31.00 © 2014 IEEE
DOI 10.1109/ICTAI.2014.106

$\mathbb{N} = \{0,1,2,\dots\}$ is the set of non-negative integers and $\mathbb{Z}$ is the set of all integers. For a given transition t, an *input* (*output*) place of t is a place p such that $pre(p,t)$ ($post(p,t)$) is positive, respectively. We write ${}^\bullet t$ ($t^\bullet$) for the set of all input (output) places of a transition t, respectively.

A *state* of a Petri nets, known as a *marking*, is represented as $M : P \to \mathbb{N}$ capturing the number of tokens in each place. We sometimes represent a marking as an $1 \times n$ matrix of non-negative integers, assuming that the set of places are ordered to correspond the coordinates of the matrix. A transition t is *enabled* at a marking M if for each $M \geq pre(.,t)$, where $pre(.,t)$ is an $1 \times n$ matrix with coordinates $pre(p,t)$ for $p \in P$. An enabled transition can *fire* resulting in a new marking M, denoted by $M \xrightarrow{t} M'$, where $M' = M + post(.,t) - pre(.,t)$. A sequence of transitions $\mathbf{s} = t_1 \dots t_k$ of T is called *enabled* at a marking M, if there are marking $M_1, \dots, M_k$ so that $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \cdots \xrightarrow{t_k} M_k$. In this case, we write $M \xrightarrow{\mathbf{s}} M_k$ and refer to M_k as a state *Reachable* from M. We write $R(\mathcal{N},\ M)$ for the set of all reachable states from M. The initial state of the system is represented by an *initial marking* M_0. We will write $(\mathcal{N},\ M_0)$ for a Petri net with its initial marking. Suppose that $\mathcal{N} = (P,T,pre,post)$ and $\mathcal{N}' = (P',T',pre',post')$ are two Petri nets with initial marking M_0 and M_0', respectively. We say $(\mathcal{N}',\ M_0')$ is an *extension* of $(\mathcal{N},\ M_0)$ and write $(\mathcal{N},\ M_0) \preceq (\mathcal{N}',\ M_0')$ iff $P \subset P'$, $T \subset T'$, $pre'\,|_{P\times T} = pre$, $post'\,|_{P\times T} = post$, and $M'_0\,|_P = M_0$. In other words, $(\mathcal{N}',\ M_0')$ extends $(\mathcal{N},\ M_0)$ by (possibly) adding extra places to P, extra transitions to T. In addition, all arcs connecting places and transitions in $\mathcal{N}$ and the marking of the places in M_0 are present in $\mathcal{N}'$.

The set of all finite-length strings of the transitions in T is denoted by T^* and is called the *Kleene-closure* of T. As a result, members of T^* are created from concatenation of finite number of elements of T. In particular, T^* contains the empty string ε, so that $t\varepsilon = \varepsilon t = t$ for all $t \in T$. Every subset of T^* is called a *language on the alphabet* T. Suppose that μ_1 and μ_2 are two mappings from T to $\mathbb{Z}$, we write $\mu_1 \odot \mu_2$ for the inner product of the two mapping, i.e. $\mu_1 \odot \mu_2 = \sum_{t\in T} \mu_1(t)\mu_2(t)$. The Parikh image $\# : T^* \to \mathbb{N}^T$ is a map which assigns to every word $\mathbf{s}$ a map $\#(\mathbf{s})$ that produces the number of the occurrences of each event in $\mathbf{s}$. In other words, for $\#(\mathbf{s}) : T \to \mathbb{N}$, $\#(\mathbf{s})(t)$ is the number of occurrences of $t \in T$ within the word $\mathbf{s}$. Sometimes we write $\#(t,\mathbf{s})$ to represent the number of the occurrences of t in $\mathbf{s}$. Suppose that $\mathbf{z} \in \mathbb{Z}^T$. For each $\mathbf{s} \in T^*$, $\mathbf{z} \odot \#(\mathbf{s}) = \#(\mathbf{s}) \odot \mathbf{z}$ is the same as $\sum_{t\in T} \mathbf{z}(t) \times \#(t,\mathbf{s})$.

The set of sequence of transitions resulting in a reachable marking is called the *Language* of the Petri net and is denoted by $L(\mathcal{N},\ M_0)$ i.e. $L(\mathcal{N},\ M_0) = \{\mathbf{s} \in T^* \mid \exists M\ M_0 \xrightarrow{\mathbf{s}} M\}$. We will use $\mathbf{s}, \mathbf{s}_0, \mathbf{s}_1, \dots$ for representing sequences of transitions. A Petri net $(\mathcal{N},\ M_0)$ is called *bounded* if there is an upper bound on the number of tokens that can arrive in each place, i.e. $\exists k \forall M \in R(\mathcal{N},\ M_0) \forall p \in P,\ M(p) \leq k$.

III. Diagnosability in Petri nets

As we are presenting an extension to the theory of diagnosability, we will be introducing concepts with the same name as the one used in the existing theory of diagnosability in Discrete Event Systems. *When there is a chance of ambiguity we shall use the phrase* ***classic*** *to clarify*.

Consider a Petri net $(\mathcal{N},\ M_0)$ with a set of transitions T. Suppose that T is partitioned into two sets: observable transitions T_o and unobservable transitions T_u. We further assume that faults are unobservable transitions, i.e. $T_f \subset T_u$, in which T_f is the set of transitions which are modelling occurrences of failure. In this paper, we extend classic theory of diagnosability to the case where T_f can be empty. Consider the *projection* function $\pi : T \to T_o \cup \{\varepsilon\}$ that maps unobservable transitions to the empty string ε, i.e. $\pi(t) = \varepsilon$ for $t \in T_u$ while, $\pi(t) = t$ for $t \in T_o$. The projection function π can be extended to the Kleene-closure of T by $\pi : T^* \to (T_o \cup \{\varepsilon\})^*$ where for each sequence of transitions $\mathbf{s}$ and each transition t, $\pi(\mathbf{s}t) = \pi(\mathbf{s})\pi(t)$. We assume $\pi(\varepsilon) = \varepsilon$ and that $t\varepsilon = \varepsilon t = \varepsilon$ for each $t \in T_u$.

A system may have more than one type of failure. So T_f is partitioned to $T_f^1, \dots, T_f^r$ representing different types of failure. T_o represents all the events that are observable in the systems, such as events which can be recognised via a sensor. Hence, in every execution of events, a sequence of events from T_o can be observed. A diagnoser (as explained later in this section) uses such information to identify if a fault has happened or may have happened. The following definition from [3] extends the classic definition of diagnosability of [1].

***Definition** 1:* Consider a Petri net $(\mathcal{N},\ M_0)$, which has no deadlock after firing of a transition $t_f \in T_f^i$ where $i = 1, \dots, r$. We say $\mathcal{N}$ is *diagnosable with respect to fault class* T_f^i if there are no two firing sequences $\mathbf{s}_1$ and $\mathbf{s}_2 \in T^*$ satisfying the following conditions:

1) $\pi(\mathbf{s}_1) = \pi(\mathbf{s}_2)$,
2) no failure transition appears in $\mathbf{s}_1$, i.e. $\forall\ t_f \in T_f^i,\ \mathbf{s}_1 \in (T \setminus T_f^i)^*$,
3) there exists at least one failure transition of T_f^i in $\mathbf{s}_2$,
4) it is possible to make $\mathbf{s}_2$ arbitrary long after the occurrence of a fault $t_f \in T_f^i$.

The above definition states that in a diagnosable system it is not possible to come across two execution sequences with the same observable behaviour ($\pi(\mathbf{s}_1) = \pi(\mathbf{s}_2)$), so that only one of them has failure transitions. For further details about the above definition we refer the reader to [3] .

A diagnoser is an automaton built from the model of the system to be diagnosed. This automaton has only observable events of the system and is used to estimate the current states of the system after observing a sequence of events. Diagnosers achieve two major goals : (i) check if the system is diagnosable and (ii) on-line monitoring of the system for the purpose of fault diagnosis. For further details on diagnosis and diagnosability, see [1].

IV. Yen's Logic

Consider a Petri net $(\mathcal{N},\ M_0)$, each formula in the Yen's logic is created from three elements: *Variables*, *Terms* and *Atomic Predicates*. There are two types of variables: *marking variables* $\mu_1, \mu_2, \dots$ ranging over the marking of $\mathcal{N}$. As a

result, marking variables are assigned to Petri net markings, which are denoted by $M, M_0, \ldots$. For example, we write $\mu_i := M_i$ to assign *marking variable* μ_i to the marking M_i. *Transition Sequence Variables* $\sigma_1, \sigma_2, \ldots$ ranging over finite sequence of the transitions. Transition variables are assigned to members of T^*, which are denoted by $\mathbf{s}, \mathbf{s}_0, \mathbf{s}_1, \ldots$.

Terms are defined recursively via the following three rules: 1) each $\mathbf{c} \in \mathbb{N}^P$ is a term, where P is the set of places in $\mathcal{N}$. 2) for each pair of marking variables μ_i, μ_j, where $i < j$, $\mu_j - \mu_i$ is a Term. 3) Finally, for any pair of Terms T_1 and T_2, $T_1 - T_2$ is also a Term. There are two types of *Atomic Predicates*: *Transition Predicates* and *Marking Predicates.*
Transition Predicates: For each $\mathbf{z} \in \mathbb{Z}^T$ and $c \in \mathbb{N}$ both $\mathbf{z} \odot \#(\sigma_i) > c$ and $\mathbf{z} \odot \#(\sigma_i) \geq c$ are Predicates. In particular, we can see that (by using $-\mathbf{z}$) $\mathbf{z} \odot \#(\sigma_i) \sim c$ and for transitions t, $\#(t, \sigma_1) \sim c$, where $\sim \in \{<, \leq, >, \geq\}$ are all Predicates.
Marking Predicates: For each marking variable μ, each place $p \in P$ and each $z \in \mathbb{Z}$, $\mu(p) \geq z$ and $\mu(p) > z$ are Predicates. In addition, for each pair of terms T_1, T_2 and places p_1, p_2 the expressions $T_1(p_1) = T_2(p_2)$, $T_1(p_1) > T_2(p_2)$ and $T_1(p_1) < T_2(p_2)$ are Predicates.

The above are Atomic Predicates which are used to produce Predicates. A Predicate is either a Marking Predicate, or Transition Predicates or disjunctive normal forms i.e. $\bigvee_{1 \leq i \leq r} \bigwedge_{1 \leq j \leq m_i} \phi_i^j$, where each ϕ_i^j is either a Marking Predicate or a Transition Predicate. Following [8] and [9], we are interested in predicated formulas of the form $\phi(\mu_1, \ldots, \mu_n, \sigma_1, \ldots, \sigma_n)$ i.e. formulas which are compliant with the above description and have equal number of marking and transition variables. For such predicate formulas we say an execution sequence $s := M_0 \xrightarrow{\mathbf{s}_1} M_1 \xrightarrow{\mathbf{s}_2} \ldots M_{n-1} \xrightarrow{\mathbf{s}_n} M_n$ if $\phi(M_1, \ldots, M_n, \mathbf{s}_1, \ldots, \mathbf{s}_n)$ is true. In other words, assignments $\mu_i := M_i$ and $\sigma_i := \mathbf{s}_i$ where $1 \leq i \leq n$ make ϕ true. In this case we write $\mathbf{s} \vDash \phi$, where ϕ is an abbreviation of $\phi(\mu_1, \ldots, \mu_n, \sigma_1, \ldots, \sigma_n)$. We also say a Petri net $\mathcal{N}$ satisfies ϕ, denoted by $\mathcal{N} \vDash \phi$, if ϕ satisfies at least one of its execution sequences of $\mathcal{N}$. In other words, $\mathcal{N} \vDash \phi$ iff $\exists \mathbf{s} \in L(\mathcal{N}, M_0)$ such that $\mathbf{s} \vDash \phi$.

V. Description of the Problem

To describe the problem that has motivated this paper, we shall make use of a simplified business process used within a typical telecommunication company. Suppose the scenario that a domestic customer telephones to report a malfunction such as the broadband connection being slow. We refer to such problems and malfunctions as "tasks" or "jobs". The following example describes a simplified business process from the arrival of the job to its completion.

Example 1: In Petri net of Fig. 1, when the tasks arrive (firing of s), depending on the nature of the problem which is reported, every task is allocated to one of the three Departments. Within each Department, there are a few large and complex workflows which we have (seriously) simplified to two cases. Either the problem is resolved or the engineers discover that the allocated job can NOT be resolved within their Department. This would be a case of wrong allocation of jobs and can arise from a multitude of reasons, among

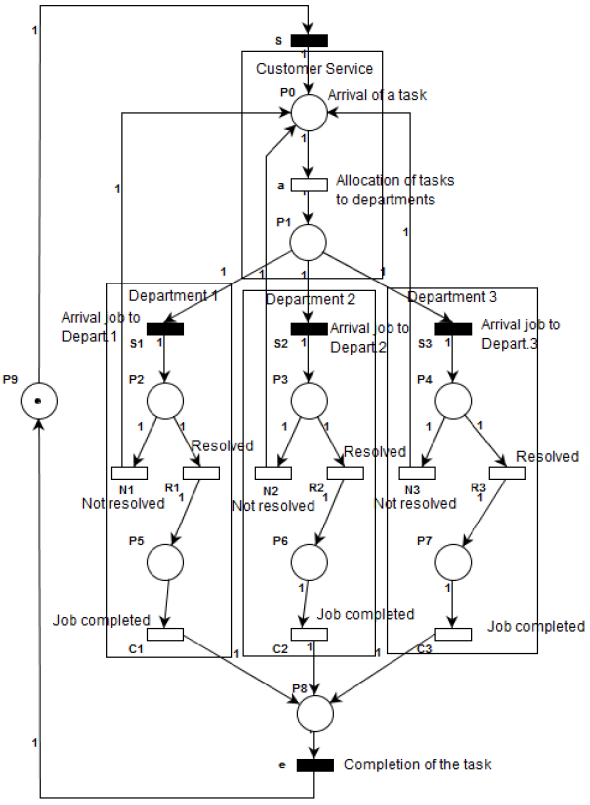

Fig. 1. A Problem to Resolve System

them wrong information from the customers or wrong assignment of jobs or the case that one fault triggers another. In the case that the job is resolved, the Department declares that "Job Completed" by firing of $C1$, $C2$ or $C3$, which ultimately results in the firing of e marking the "Completion of the (overall) task." In case that a Department is not able to complete the job (firing of $N1$, $N2$ or $N3$), further investigation is required. As a result, a token is placed in p_0 so that the job is re-allocated by the Customer Service department. We assume that transitions s and e, which mark arrival and completion of jobs are observable. In addition, transitions that mark arrival of the jobs in each Department ($S1$, $S2$ and $S3$) are also observable, as they are used by the Department to inform the Customer of the progress of the job. For example if the customer is accessing through a browser to make an online report, he is informed that the relevant department will deal with the problem. Observable transitions in Fig. 1 are depicted by solid rectangles, while empty rectangles represent unobservable transitions.

In the above example firing of $N1$, $N2$ or $N3$ result in a repetition of a chain of activities that indicates wrong allocation of jobs to the departments. Since the activities are repeated, the job is not completed Right First Time (RFT). In this case, we say RFT failure has happened. Right First Time failures are becoming increasingly important in Telecom industry [6]. Occurrence of a RFT failure may result in

unhappy customers, increases cost of resolving the problems and may entail financial penalties. As a result development of methods of discovery of RFT failures so that remedial actions can be adopted is essential. In addition, in large organisations such methods must be automated to allow dealing with large systems.

In the above example, the transition a marks allocation of jobs and the transition e marks the completion of a job. Ideally, to ensure no RFT, we wish that every allocated job is completed. In other words, for each execution sequence $\mathbf{s}$ of the Petri net.

$$\#(a,\mathbf{s}) = \#(e,\mathbf{s}) \tag{1}$$

If equation (1) happens, we have no RFT failure. However, it is not possible to completely eliminate the RFT failure in a large telecommunication company. As a result, the management sets Service Level Agreement (SLA) such as the number of failures should be below a value $\delta \geq 0$ to specify acceptable levels of failure. SLA is satisfied iff for each execution sequences, equation (2) is true.

$$\#(a,\mathbf{s}) - \#(e,\mathbf{s}) \leq \delta \tag{2}$$

a) Description of the problem :: Petri net of Fig. 1 represents a model of a plant and equation (2) represents a constraint (a SLA) which if violated, a failure has happened. *Petri net of Fig. 1 has no failure transitions*. As a result, existing fault diagnosis techniques can not be directly applied. One can argue that, one must model failure by modifying the Petri net of Fig. 1. This would mean adding extra transitions and places to *simulate* violation of equation (2). In our experience, this is not an easy task. In addition, modifying Fig. 1 may result in cumbersome and large Petri nets which will be hard to understand. Thirdly, advocates of modelling failure must modify his design as soon as the SLA changes. As a result, there is a clear scope for extending existing fault diagnosis techniques in Petri nets for the case that the fault is associated to a violation of constraints such as SLA.

b) Violation of equation (2) can be represented as Yen's logic formulas :: Each sequence $\mathbf{s}$ that violates equation (2) has the value $\Delta(\mathbf{s}) := \#(a,\mathbf{s}) - \#(e,\mathbf{s}) > \delta$. Such sequence $\mathbf{s}$ can be broken into three consecutive sequences $\mathbf{s}_1$, $\mathbf{s}_2$ and $\mathbf{s}_3$ such that $\Delta(\mathbf{s}_1) = \delta$, $\Delta(\mathbf{s}_2) = 1$ and (possibly) $\Delta(\mathbf{s}_3) \geq 0$. Hence if equation (2) is violated, we conclude

$$\exists M_1,M_2 \exists \mathbf{s}_1,\mathbf{s}_2 \; ((M_0 \xrightarrow{\mathbf{s}_1} M_1 \xrightarrow{\mathbf{s}_2} M_2) \wedge ((\#(a,\mathbf{s}_1) = 1) \wedge (\#(a,\mathbf{s}_2) - \#(e,\mathbf{s}_2) = \delta))) \tag{3}$$

Conversely, if (3) is valid, then $\mathbf{s} = \mathbf{s}_1\mathbf{s}_2$ is an execution sequence that violates equation (2). Since equation (3) is a Yen's logic formula, see section IV, violation of equation (2) can be expressed in Yen's logic.

VI. Diagnosis of violations of constraints

In the previous section, we made a case for extending the theory of diagnosability to the cases that failure is not modelled as the events within the plant. In this section we shall present an extension of the classical notion of diagnosability, see section III, to deal with the failure caused by violations of the constraints. Suppose $\mathcal{N} = (P,T,pre,post)$ is a Petri net with an initial marking M_0. To simplify, **we assume that** $(\mathcal{N}, M_0)$ **is deadlock free**. In the Petri net of Fig. 1 a token in place p_9 at initial marking ensures that the Petri net is live and safe. Suppose the set of transitions $T = T_o \cup T_u$ is divided into observable T_o and unobservable T_u, $T_o \cap T_u = \emptyset$. Notice there is no notion of failure transition. However, we assume that there is a Yen's logic predicate ϕ representing a a violation of constraint which means a failure has happened. So, a sequence of execution $\mathbf{s}$, for which $\mathbf{s} \vDash \phi$ contains a failure.

***Definition** 2:* Suppose $\mathbf{c}$ represents a constraint which its violation is seen as a failure of type T_f^i. Also, assume that the violation of $\mathbf{c}$ is expressed as Yen's logic formula ϕ on a Petri net $(\mathcal{N}, M_0)$ which is deadlock free. We say a failure of type T_f^i is diagnosable if there are no execution sequences $\mathbf{s}_1$ and $\mathbf{s}_2$ in $L(\mathcal{N}, M_0)$ satisfying the following:

1) $\pi(\mathbf{s}_1) = \pi(\mathbf{s}_2)$,
2) $\mathbf{s}_1 \vDash \phi$ while $\mathbf{s}_2 \nvDash \phi$,
3) $\mathbf{s}_2$ is of arbitrary length after satisfying ϕ.

The above definition is an extension of the definition 1 by Cabasino et al [3] which itself extends the definition of diagnosability of [1].

***Lemma** 1:* The definition 1 is a special case of the definition 2.

Proof: Assume that in a given Petri net some of the events are modelled as failure transitions, i.e. there are classes of failure $T_f^1,\ldots,T_f^n \subset T_u$ so that firing of any transition $t \in T_f^i$ means that a failure of class T_f^i has occurred. Consider a constraint of the form $\sum_{t\in T_f^i} \#(t,\mathbf{s}) < 1$, where $\mathbf{s}$ is a sequence of the execution of the Petri net. If this constraint is evaluated as true, then none of the failures in T_f^i will appear in $\mathbf{s}$. The constraint

$$\bigwedge_{i=1}^{n} \sum_{t\in T_f^i} \#(t,\mathbf{s}) < 1 \tag{4}$$

means that no failure transition of $T_f^1 \cup \cdots \cup T_f^n$ will appear in $\mathbf{s}$. According to both Remark 8 and the proof of Theorem 9 in [9], equation (4) can be written in a Yen's logic format as

$$\exists M_1 \exists \mathbf{s} \; ((M_0 \xrightarrow{\mathbf{s}} M_1) \wedge (\bigvee_{i=1}^{n} (\sum_{t\in T_f^i} \#(t,\mathbf{s}) \geq 1))) \tag{5}$$

in which any appearance of a failure transition of any class T_f^i will make the value of the sum greater than or equal to one. As a result, the Yen's logic formula above will be satisfied i.e. a failure has happened. ■

From Lemma 1, we can conclude that the problem of diagnosis of a failure in DESs modelled using Petri net can be reduced to the satisfiability problem for Yen's logic formula.

VII. A DES APPROACH TO DIAGNOSE VIOLATIONS OF CONSTRAINTS

Suppose that $\mathcal{N} = (P, T, pre, post)$ is a Petri net with initial marking M_0. Let us assume that $\mathbf{c}$ is a constraint which its violation represents a failure. Also, assume that ϕ is a Yen's logic formula that represents the failure, i.e. violation of $\mathbf{c}$. In section VI, we presented a definition of diagnosability that extends the classic notion of diagnosability. In the new definition, we do not have any concept of failure transitions. The systems under study in this paper are partially observable. As a result, the set of transitions T is partitioned to observable T_o and unobservable T_u transitions. In this section we show that the extended definition of diagnosability and creation of diagnosers can be reduced to similar problems in classic diagnosability theory of Discrete Event Systems. The outline of the solution, which is depicted in Fig. 2, involves creation of a new Petri net $\mathcal{N}'$. This new Petri net has failure transitions T'_f so that

1) A violation of constraint $\mathbf{c}$ can be diagnosed in $\mathcal{N}$ according to definition 2 iff $\mathcal{N}'$ is diagnosable with respect to the fault classes in T'_f, definition 1.
2) Any diagnoser that diagnoses occurrences of the failure events of T'_f in $\mathcal{N}'$ can diagnose a violation of constraint $\mathbf{c}$ in $\mathcal{N}$.

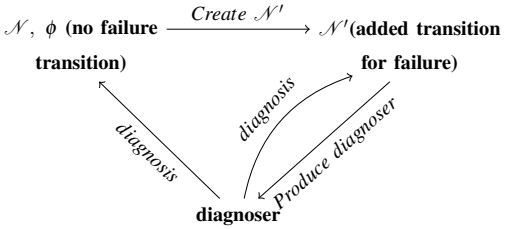

Fig. 2. Producing diagnosers to identify violation of constraint ϕ

In what follow, we suppose π, π' denote the projection maps in $\mathcal{N}$ and $\mathcal{N}'$, respectively.

***Lemma** 2:* Suppose that $\mathcal{N} = (P, T, pre, post)$ is a deadlock free Petri net with an initial marking M_0. Assume that $\mathbf{c}$ is a constraint which its violation is a Yen's logic formula of the form $\psi := \phi_1 \wedge \phi_2 \cdots \wedge \phi_n$, where each ϕ_i is a marking predicate. Then, we can create, in polynomial time, a new Petri net $\mathcal{N}' = (P', T', pre', post')$ with initial marking M'_0 such that

1) $(\mathcal{N},\ M_0) \preceq (\mathcal{N}',\ M'_0)$, i.e. $P \subset P'$, $T \subset T'$ and $M'_0\,|_P = M_0$,
2) $(\mathcal{N}',\ M'_0)$ is deadlock free,
3) $\mathcal{N}'$ has new transitions, among them $t_{end} \in T' \setminus T$, so that if t_{end} is fired the marking of $\mathcal{N}'$ changes to M'_0. In particular, for each place p, where $M'_0(p) > 0$, $post'(t_{end}, p) = M'_0(p)$,
4) for every execution sequence $\mathbf{s}$ of $(\mathcal{N},\ M_0)$, if $\mathbf{s} \vDash \psi$, then there is an execution sequence $\mathbf{s}'$ of $(\mathcal{N}',\ M'_0)$ such that $\mathbf{s}'$ contains t_{end} and $\pi(\mathbf{s}) = \pi'(\mathbf{s}')$. Conversely, for every execution sequence $\mathbf{s}'$ of $(\mathcal{N}',\ M'_0)$ which contains t_{end}, there exist an execution sequence $\mathbf{s}$ in $(\mathcal{N},\ M_0)$ such that $\mathbf{s} \vDash \psi$ and $\pi'(\mathbf{s}') = \pi(\mathbf{s})$.

Proof: The proof is a direct consequence of Theorem 9 of [9], which states that there is a transition t_{end} so that $\mathcal{N} \vDash \psi$ iff firing of t_{end} results in marking $\vec{\mathbf{0}}$. This means (4) and (1) are satisfied. In effect, firing of t_{end}, which has no output places, empties the Petri net of all tokens resulting in a deadlock by going to state $\vec{\mathbf{0}}$. To make the Petri net deadlock free, we follow the method suggested in [10] to connect t_{end} to all places p with $M'_0(p) > 0$ and give the arc weight of $M'(p)$. As a result firing of t_{end} will produce the initial marking. Hence (2) and (3) are satisfied. ■

***Lemma** 3:* Suppose that $\mathcal{N} = (P, T, pre, post)$ is a deadlock free Petri net with an initial marking M_0. Suppose that ϕ is a Yen's logic formula with predicates of the form $\phi := \psi_1 \vee \psi_2 \cdots \vee \psi_r$, where each $\psi_i := \phi_i^1 \wedge \phi_i^2 \cdots \wedge \phi_i^n$ in which ϕ_i^j is a marking predicate. Then, we can create, in polynomial time, a new Petri net $\mathcal{N}' = (P', T', pre', post')$ with initial marking M'_0 such that

1) $(\mathcal{N},\ M_0) \preceq (\mathcal{N}',\ M'_0)$ i.e. $P \subset P'$, $T \subset T'$ and $M'_0\,|_P = M_0$,
2) $(\mathcal{N}',\ M'_0)$ is deadlock free,
3) T' has a set of extra (unobservable) failure transitions $T'_f = \{t^1_{end},\ t^2_{end}, \ldots,\ t^r_{end}\}$ (i.e. $T'_f \subseteq T' \setminus T$) such that firing of each t^i_{end} result in the initial marking M'_0,
4) for every execution sequence $\mathbf{s}$ of $(\mathcal{N},\ M_0)$, if $\mathbf{s} \vDash \phi$, then there is an execution sequence $\mathbf{s}'$ of $(\mathcal{N}',\ M'_0)$ such that $\mathbf{s}'$ contains at least one of the failure transitions of T'_f and $\pi(\mathbf{s}) = \pi'(\mathbf{s}')$. Conversely, for every execution $\mathbf{s}'$ of $(\mathcal{N}',\ M'_0)$ which contains at least one of the failure transitions of T'_f, there exist an execution sequence $\mathbf{s}$ in $(\mathcal{N},\ M_0)$ such that $\mathbf{s} \vDash \phi$ and $\pi'(\mathbf{s}') = \pi(\mathbf{s})$.

Proof: The proof is a direct consequence of Lemma 2. We need to repeat the theorem r times. ■

***Theorem** 1:* For any given Yen's logic formula ϕ on a deadlock free Petri net $(\mathcal{N},\ M_0)$, we can produce a new deadlock free Petri net $(\mathcal{N}',\ M'_0)$ so that

1) $(\mathcal{N},\ M_0) \preceq (\mathcal{N}',\ M'_0)$,
2) $\mathcal{N}$ and $\mathcal{N}'$ have the same set of observable transitions,
3) for every execution sequence $\mathbf{s}$ of $(\mathcal{N},\ M_0)$, if $\mathbf{s} \vDash \phi$, then there is an execution sequence $\mathbf{s}'$ of $(\mathcal{N}',\ M'_0)$ such that $\mathbf{s}'$ contains a transition from T'_f and $\pi(\mathbf{s}) = \pi'(\mathbf{s}')$. Conversely, for every execution $\mathbf{s}'$ of $(\mathcal{N}',\ M'_0)$ which contains a transition from T'_f, there exist an execution sequence $\mathbf{s}$ in $(\mathcal{N},\ M_0)$ such that $\mathbf{s} \vDash \phi$ and $\pi'(\mathbf{s}') = \pi(\mathbf{s})$.

Proof: Starting from an arbitrary Yen's logic predicate, we shall apply Lemma 3.2 of [8] and replace any Yen's logic transition predicates with equivalent marking predicates. Then the theorem is a direct result of Lemma 3. ■

As stated in Lemma 3 when a Yen's logic predicate is in conjunctive normal form $\bigvee_{1\le i\le r}\bigwedge_{1\le j\le m_i}\phi_i^j$, each $\bigwedge_{1\le j\le m_i}\phi_i^j$ represents a class of failure, which means that we have a failure if all conditions captured in ϕ_i^j are satisfied on a sequence. Following the notation of Lemma 3, we shall write

$T_f'^i = \{t^i_{end}\}$ to represent the class of failure associated to t^i_{end}, where $i = 1,2,\ldots,r$. Any diagnoser that can identify occurrence of a failure in $T_f' = T_f'^1 \cup \cdots \cup T_f'^r$ in $\mathcal{N}'$ can be used to detect violations of constraint in $\mathcal{N}$. This is because $\mathcal{N}$ and $\mathcal{N}'$ have the same set of observable transitions.

***Theorem** 2:* Suppose that $\mathcal{N}$, $\mathcal{N}'$ and ϕ satisfy conditions of Theorem 1. $\mathcal{N}$ is diagnosable with respect to ϕ iff $\mathcal{N}'$ is diagnosable with respect to T_f'.

Proof: The proof has two parts, *if* and *only if*. To prove *if* part, we suppose $\mathcal{N}$ is diagnosable with respect to ϕ, but $\mathcal{N}'$ is not diagnosable with respect to T_f'. As a result, by definition 1 there are two execution sequences $\mathbf{s}_1', \mathbf{s}_2' \in (T')^*$ such that $\pi'(\mathbf{s}_1') = \pi'(\mathbf{s}_2')$, no failure transition of T_f' appears in $\mathbf{s}_1'$ but $\mathbf{s}_2'$ has at least one failure transition from T_f'. Since $\mathcal{N}$ satisfies Theorem 1, then by condition (3) of which, we infer that there exist two sequences $\mathbf{s}_1$ and $\mathbf{s}_2$ in $(\mathcal{N}, M_0)$ such that $\mathbf{s}_1 \vDash \phi$, $\mathbf{s}_2 \nvDash \phi$ and $\pi(\mathbf{s}_1) = \pi'(\mathbf{s}_1')$, $\pi(\mathbf{s}_2) = \pi'(\mathbf{s}_2')$. Because of $\pi'(\mathbf{s}_1') = \pi'(\mathbf{s}_2')$, then $\pi(\mathbf{s}_1) = \pi(\mathbf{s}_2)$. By definition 2, this contrasts the assumption. Hence, if $\mathcal{N}$ is diagnosable with respect to ϕ, it is necessarily that $\mathcal{N}'$ is diagnosable with respect to T_f'.

We prove now the *only if* part. Assume that $\mathcal{N}'$ is diagnosable with respect to T_f', but $\mathcal{N}$ is not diagnosable with respect to ϕ. Consequently, by definition 2 there are two sequences $\mathbf{s}_1$, $\mathbf{s}_2 \in T^*$ such that $\pi(\mathbf{s}_1) = \pi(\mathbf{s}_2)$, $\mathbf{s}_1 \vDash \phi$, $\mathbf{s}_2 \nvDash \phi$. Again, since $\mathcal{N}'$ satisfies Theorem 1, then by condition (3) of which, we conclude that there are two sequences $\mathbf{s}_1'$, $\mathbf{s}_2'$ in $(\mathcal{N}', M_0')$ such that $\mathbf{s}_1'$ contains at least one failure transition from T_f', but $\mathbf{s}_2'$ does not and $\pi'(\mathbf{s}_1') = \pi(\mathbf{s}_1)$, $\pi'(\mathbf{s}_2') = \pi(\mathbf{s}_2)$. Because of $\pi(\mathbf{s}_1) = \pi(\mathbf{s}_2)$, then $\pi'(\mathbf{s}_1') = \pi'(\mathbf{s}_2')$. This results in contradiction to the assumption. Hence, being $\mathcal{N}'$ diagnosable with respect to T_f' is sufficient condition for $\mathcal{N}$ to be diagnosable with respect to ϕ. ■

***Theorem** 3:* Suppose that $\mathcal{N}$, $\mathcal{N}'$, ϕ and T_f' satisfy conditions of Theorem 1 and $\mathcal{N}$, $\mathcal{N}'$ are diagnosable with respect to ϕ and T_f', respectively. Any diagnoser Δ that can identify occurrence of failure in T_f' can identify if ϕ holds and vice versa.

Proof: The proof is a direct consequence of Theorem 2. ■

Remark: The shape of each individual formula in a Yen's logic predicate is important. For example, a less interesting special case is when in the Yen's logic formula all the non-zero coefficients are for observable transitions. For example, when in $\sum_{t\in T} \mathbf{z}(t) \times \#(t,\mathbf{s})$ we have $\mathbf{z}(t) = 0$ if t unobservable. In such a case, the sum can be calculated from the observable events. This is similar to the case that in classic failure diagnosability when some failure transitions are observable. Interesting cases occur when $\mathbf{z}(t) = 0$ for one or more unobservable transition.

Example 2: ***Constructing the Petri net** $\mathcal{N}'$*

Consider Petri net $\mathcal{N}$ of Fig. 1 of our running example. A special case of RFT failure is described using equation (3) of section V. By Lemma 2, we can create a new Petri net $\mathcal{N}'$ which has failure events corresponding to the violation RFT as formulated in equation (3). However, construction of $\mathcal{N}'$ by applying the procedure in [9] will result in very large Petri nets. We followed the procedure described in section 6 of [9] and constructed the Petri net $\mathcal{N}'$ ended up with over 40 extra transitions and 7 extra places. This makes the execution of diagnosers impossible from practical point of view. Fortunately, for the Petri net of example 1 and Yen's logic formula described in the equation (3) there is a much smaller Petri net $\mathcal{N}'$ that satisfies Lemma 2. This Petri net is depicted in Fig. 3.

Checking that $\mathcal{N}'$ satisfies Lemma 2 is straightforward. The initial marking of $\mathcal{N}'$ (Fig. 3) is such that $\forall p \in P$, $M_0'(p) = M_0(p)$. In addition $M_0'(D) = 0$ and $M_0'(C) = 3$. As a result, condition (1) of Lemma 2 is satisfied. t_{end} is the only added transition to the set of transitions of $\mathcal{N}$. As t_{end} is unobservable, condition (2) of Lemma 2 is satisfied. Firing t_{end}, after a violation has happened, will produce the initial marking M_0'. We have created the *reachability graph* (Fig. 4 with initial state marked by thick border line) of $\mathcal{N}'$ and showed that $\mathcal{N}'$ is deadlock free. At the same time, it satisfies conditions (3) and (4) of Lemma 2.

Reachability graph in Fig. 4 shows that the Petri net $\mathcal{N}'$ of Fig. 3 is bounded and has no cycle of unobservable events. Hence, we can use Theorem 1 of [1] to verify if it is diagnosable or not. For this end, we need to produce a diagnoser from the *Reachability graph* following the method of [11]. In fact, the *reachability graph* and diagnoser have been produced using the tool described in [12].

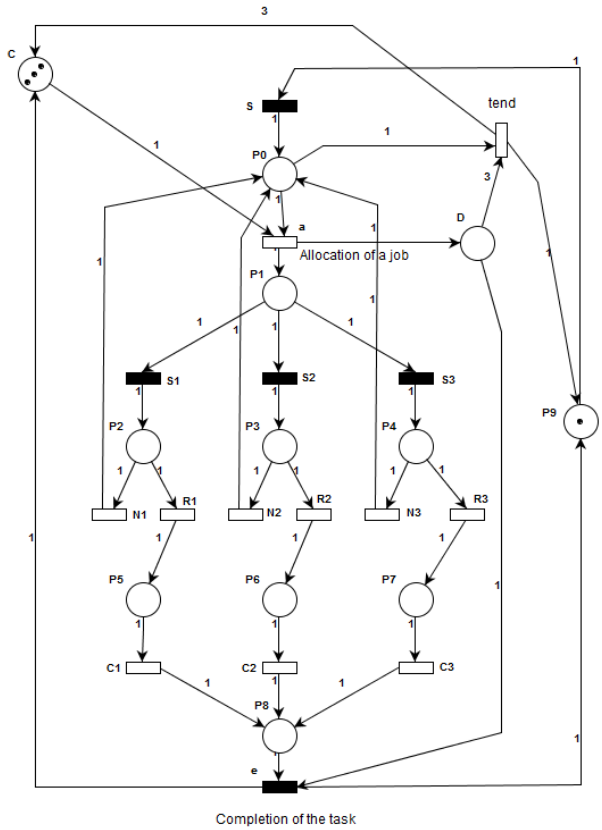

Fig. 3. Petri net $\mathcal{N}'$ of the example 1

Fig. 5 depicts a part of the diagnoser of the Petri net $\mathcal{N}'$ of Fig. 3. The rest of this diagnoser has not been included

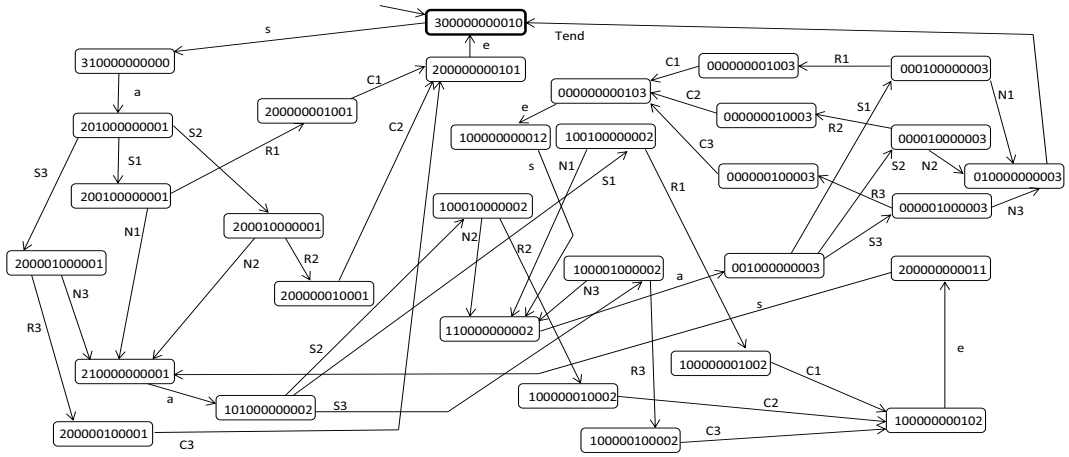

Fig. 4. Reachability graph of the Petri net $\mathscr{N}'$

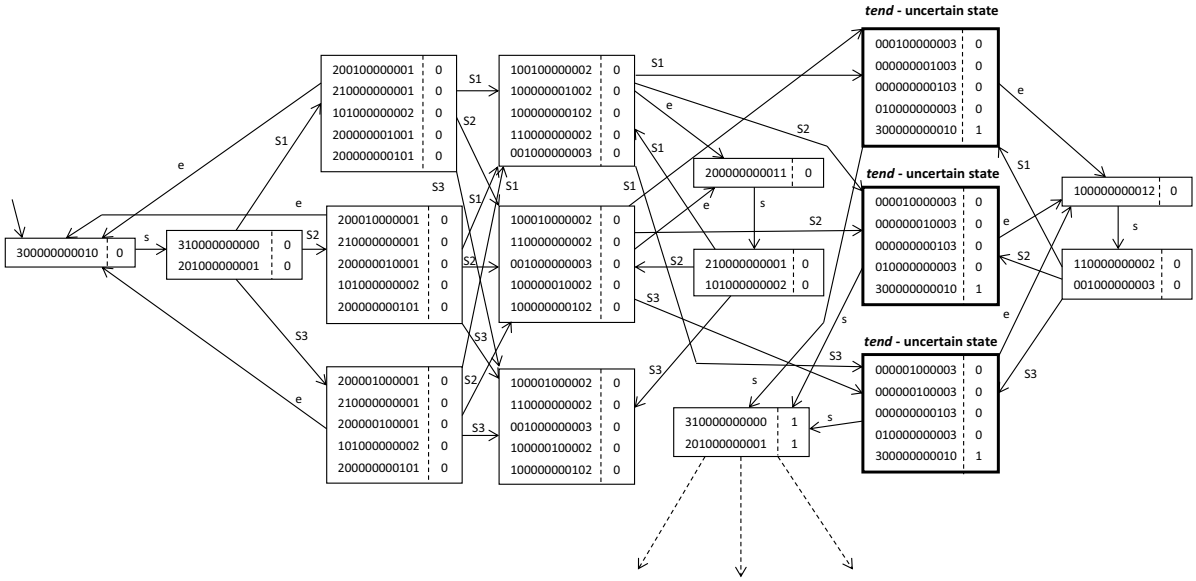

Fig. 5. The diagnoser of the Petri net $\mathscr{N}'$

due to space constraints, and dotted arrows show that. Each diagnoser state includes a number of rows. Each row has two parts: first part represents marking of $\mathscr{N}'$, which is a vector of dimension $|P'| = 12$. The second part, which is separated from the first part, decodes the failure information. Each column of this part represents failure type represented by a label. This label is either 0, in case that there is no failure, or 1 which indicates failure. When all values of one column that correspond to one failure type are 0(1), that means the diagnoser is *certain* that the failure must (must not) have happened. On the other hand, if the column entries include both 1 and 0, this refers to that the diagnoser state is *uncertain*, i.e. a failure may have happened.

According to Theorem 1 of [1], in the presence of this diagnoser, the problem of verifying the diagnosability can be reduced to the problem of finding the *indeterminate cycles* (cycles of *uncertain* states) in this diagnoser. As there are no *indeterminate cycles*, Petri net $\mathscr{N}'$ is diagnosable. By Theorem 2, $\mathscr{N}$ is diagnosable too and the diagnoser of Fig. 5 can detect violation of the constraint expressed in equation (3). We shall exemplify this next.

Consider the system shown in Fig. 1. Assume that we have two sequences of observable events; $\mathbf{s}_1 = sS_1S_1S_1s$ and $\mathbf{s}_2 = sS_1S_1es$. The former sequence satisfies equation (3), where a close inspection of this sequence in Fig. 4 reveals that the difference between number of executions of a and e is greater than $\delta = 2$. The other sequence dose not satisfy this equation because there are only two appearances of the event S_1 which means there are two executions of the unobservable event a against one execution of the event e; hence the difference value is less than the value of δ.

Let us trace running the diagnoser, shown in Fig. 5, for these two sequences to know what would be its decisions in both cases. Regarding sequence $\mathbf{s}_1$, when the diagnoser

observes the event s this takes the diagnoser to a normal state. The same thing happens for the second appearance of S_1 whereas observing the third S_1 enters the diagnoser into an *uncertain* state. Finally, the event s makes the diagnoser certain about the occurrence of the violation. On the other hand, projection of the sequence $\mathbf{s}_2$ on the diagnoser shows normal states for all the events in this sequence.

VIII. Related work

A wide range of SLA and QoS statements related to ratio of events, such as error rate, percentage of availability can be expressed in the Yen's logic. For example, consider the ratio of message loss in communication channel. In this example, let us assume that t_1 represents sending of a message to a channel and t_2 represents arrival of the message at the other end. It is required that the ratio of the loss be $\frac{\#(t_1,\mathbf{s})}{\#(t_2,\mathbf{s})} \leq \frac{p}{q}$. This means $q \times \#(t_1,\mathbf{s}) - p \times \#(t_2,\mathbf{s}) \leq 0$. The following formula represents Yen's logic expression for this constraint

$$\exists M_1,M_2 \exists \mathbf{s}_1,\mathbf{s}_2 \; ((M_0 \xrightarrow{\mathbf{s}_1} M_1 \xrightarrow{\mathbf{s}_2} M_2) \wedge \\ ((\#(t_2,\mathbf{s}_1) = 0) \wedge (q \times \#(t_1,\mathbf{s}_2) - p \times \#(t_2,\mathbf{s}_2) > 0))),$$

where the ratio $\frac{p}{q} > 1$ and $q \neq 0$.

In our approach, we adopted the extended method in [11] of what has been suggested in [1] to generate the diagnoser and verify the diagnosability. Unfortunately, this method has the problem of exhaustive enumeration of reachable markings. To overcome this problem, several methods [3], [13]–[16] have been introduced to reduce the enumeration of the markings of Petri net and as a result reduce the size of the diagnoser. One of the works suggested in this regards has been presented by [13]. In this work, the notion of *basis markings* has been proposed that relies on finding the set of markings consistent with some observable sequence. Then, a set of markings are enumerated reached from *basis markings* set by firing unobservable transitions. This set of *basis markings* is used to build a deterministic automaton which is called *basis reachability tree* used as a diagnoser. A similar idea can be found in [15], but to deal with general Petri net models compared with the previous idea. Adopting any of these ideas to create a diagnoser, rather than the idea suggested in [11], reduces the size of the diagnoser. In addition to this, creating the diagnoser will be in one step and without the need to create an automaton (*reachability graph*) from the Petri net and then produce another one which represents the diagnoser.

As mentioned in section VII, regarding creating a new Petri net from another, the creation procedure described in [9] produces a large Petri net. As result, that leads to creating a diagnoser with large state space. Implementing this diagnoser in order to detect violations of constraints is not possible from practical point of view. Therefore, it is there is a clear scope for presenting methods of producing Petri net $\mathcal{N}'$ such that the size does not grow inhibitively. This remains an area for future research.

IX. Conclusions

A New definition of diagnosability in partially observable Discrete-Event Systems modelled using Petri nets has been presented in this paper. This new definition adopts a new formalism to model a failure as a violation of logical constraints. In order to explain the notion of this definition, an example that represents Right-First Time (RFT) failure has been described and studied in this context. We have proved that the classic definition of diagnosability represents a special case of this new definition. The suggested method consists of the following steps. Firstly, we create a new Petri net from the original one with failure transitions which its firing corresponds to violations of constraints. Constraints in here are written in Yen's logic. Secondly, we use the existing approaches to diagnose these failures in the new Petri net. In other words, the diagnoser that can diagnosis these failures in the new Petri net has the same ability to diagnosis a violation of constraints in the original model.

References

[1] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1555–1575, 1995.

[2] S. Genc and S. Lafortune, "Distributed diagnosis of Place-Bordered Petri nets," *IEEE Transactions on Automatic Science and Enginnering*, vol. 4, no. 2, pp. 206–219, 2007.

[3] M. P. Cabasino, A. Giua, S. Lafortune, and C. Seatzu, "Diagnosability analysis of unbounded Petri nets," in *48th IEEE Conference on Decision and Control*, Shanghai, China, December 2009, pp. 1267–1272.

[4] F. Basile, P. Chiacchio, and G. D. Tommasi, "Sufficient conditions for diagnosability of Petri nets," in *Proceedings of the 9th International Workshop on Discrete Event Systems*, Gteborg, Sweden, May 2008.

[5] M. Dotoli, M. P. Fanti, A. M. Mangini, and W. Ukovich, "On-line fault detection of discrete event systems by Petri nets and integer linear programming," *Automatica*, vol. 45, no. 11, pp. 2665–2672, 2009.

[6] M. Alodib and B. Bordbar, "A model-based approach to fault diagnosis in service oriented architectures," in *Proceedings of the IEEE European Conference on Web Services (ECOWS)*, Netherlands, 2009, pp. 129–138.

[7] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, April 1989.

[8] H.-C. YEN, "A unified approach for deciding the exitence of certain Petri net paths," *Information and Computation*, vol. 96, pp. 119–137, 1992.

[9] M. F. Atig and P. Habermehl, "On Yen's path logic for Petri nets," *International Journal of Foundations of Computer Science*, vol. 22, no. 4, pp. 783–799, 2011.

[10] W. van der Aalst, "Verification of workflow nets," *Application and Theory of Petri Nets*, vol. 1248, pp. 407–426, 1997.

[11] S. Genc and S. Lafortune, "Distributed diagnosis of discrete-event systems using Petri nets," in *Applications and Theory of Petri Nets*, vol. 2679, Eindhoven, The Netherlands, June 2003, pp. 316–336.

[12] L. Ricker, S. Lafortune, and S. Genc, "DESUMA: A tool integrating giddes and umdes," in *Discrete Event Systems, 2006 8th International Workshop on*, July 2006, pp. 392–393.

[13] D. Corona, A. Giua, and C. Seatzu, "Marking estimation of Petri nets with silent transitions," in *IEEE Conference on Decision and Control*, Atlantis, Paradise Island, Bahamas, December 2004.

[14] A. Giua and C. Seatzu, "Fault detection for discrete event systems using Petri nets with unobservable transitions," in *Proceeding of the 44th IEEE Conference on Decision and Control, and the Eurpean Control Conference*, Seville, Spain, December 2005, pp. 6323–6328.

[15] G. Jiroveanu, R. K. Boel, and B. Bordbar, "On-line monitoring of large Petri net models under partial observation," *Discrete Event Dynamic Systems*, vol. 18, pp. 323–354, 2008.

[16] G. Jiroveanu and R. K. Boel, "The diagnosability of Petri net models using minimal explanations," *IEEE Transaction on Automatic Control*, vol. 55, no. 7, pp. 1663–1668, 2010.