An Efficient Approach for Online Diagnosis of Discrete Event Systems

Francesco Basile, *Member, IEEE*, Pasquale Chiacchio, and Gianmaria De Tommasi, *Member, IEEE*

***Abstract*—A novel approach to fault diagnosis of discrete event systems is presented in this paper. The standard approach is based on the offline computation of the set of fault events that may have occurred at each reachable state, providing a fast online diagnosis at a price of excessive memory requirements. A different approach is here adopted, which is based on the online computation of the set of possible fault events required to explain the last observed event. This is efficiently achieved by modelling the plant by Petri nets, since their mathematical representation permits to formulate the fault diagnosis problems in terms of mathematical programming, which is a standard tool. Moreover, the graphical representation of the net allows the diagnoser agent to compute off-line reduced portions of the net in order to improve the efficiency of the online computation, without a big increase in terms of memory requirement.**

***Index Terms*—Discrete event systems (DES), fault diagnosis, Petri nets (PNs).**

I. Introduction

TODAY the safety issue plays an important role for the reliability of complex systems. The fault detection is crucial for the safety of both systems and operators. Once a fault has been detected and identified, the control law can be modified in order to safely continue the operations, although with degraded performance. As a result the fault diagnosis issue has been studied a lot in the field of discrete event systems (DES) field since the early 90s [1].

DES based methodologies for fault diagnosis are applicable not only to systems normally modeled as DES, but also to systems that traditionally are treated as continuous-time dynamic systems. In general, DES approaches to fault diagnosis are suitable for failures that cause a distinct change in the state of system components but do not bring the system to a halt: examples are equipment failures (stuck failure of valves, stalling of actuators, controller failures, etc.) usual in flight control systems or heating and air conditioning systems, and process failures (buffer overflow) usual in manufacturing systems.

A. Position of the Work

A standard approach to fault diagnosis is to build a DES, called *compiled diagnoser*. At each state transition this fault diagnoser provides the set of faults that could have happened [1], or a set of fault states that the system could have reached [2]. In general, building the compiled diagnoser is very computation demanding, even if it can be performed offline, often resulting in a state space that is too big. However, the computational effort to run a diagnoser is very low. Hence, this approach provides a fast online diagnosis at a price of excessive memory requirements, since the diagnoser state space must be fully available. It follows that such an approach may be applied only when the state space is bounded, or not too large in practice.

This is the usual approach when the plant is modeled as finite state automata [1]–[3]. In this framework, the diagnoser is a finite state automaton and many theoretical results have been devised: conditions to establish if the system is diagnosable or not, i.e., all faults can be detected [1], [2]; it has been shown that a diagnoser coupled to a controller could drive a system from nominal to degraded behavior and vice versa, positively affecting the robustness of a control system [4]; in [5] the authors deal with a decentralized implementation of the diagnoser. An extension of this approach to stochastic automata has been presented in [6].

Another approach is to write an algorithm, called *interpreted diagnoser*, which online, after each observed event, computes the set of faults that could have happened, or a set of fault states the system could have reached. In this case, the computational effort to run a diagnoser is higher than in the case of the compiled diagnoser, and it is difficult to derive a diagnosability test. On the other hand, the memory requirement is much less since there is no need to precompute any state space. A first interpreted diagnoser based on automata can be found in [7], even if the interpreted diagnoser are used mainly with Petri Nets (PNs), since the online computation can take advantage of their twofold representation—graphical and mathematical.

Several works based exclusively on a PN model of the plant have been presented recently. Since the approach proposed in this paper is based on PNs, below some related works are briefly recalled.

In [8] faults are not explicitly taken into account in the model, and two types of faults have been defined: a place fault that corrupts the net marking, and a transition fault that causes an incorrect update of the marking after events occurrences. The authors propose an identification methodology based on algebraic decoding techniques and redundant places which are added to the plant. Although the approach is very general and it leads to an interpreted diagnoser, it is assumed that the net marking is periodically observable even if unobservable events (transitions) – i.e., events whose occurrence cannot be directly detected by sensors—are admitted.

More usually faults are associated to unobservable transitions (unobservable transitions not associated to fault events are also considered) as in the works recalled below.

Manuscript received October 26, 2007; revised May 21, 2008. Current version published April 08, 2009. Recommended by Associate Editor S. Haar.

F. Basile and P. Chiacchio are with the Dip. Ing. dell'Informazione e Ing. Elettrica, Università di Salerno, Salerno 84084, Italy (e-mail: fbasile@unina.it).

G. De Tommasi is with the Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II, Napoli 80125, Italy.

Digital Object Identifier 10.1109/TAC.2009.2014932

In [9] the authors propose a novel compiled diagnoser approach based on a PN (similar to a colored PN), which associates to each net marking the different estimated states the plant may be in. Such estimated states are computed offline. Substantially, as for the state explosion problem, this diagnoser does not improve the previous ones based on automata when it is implemented in a centralized way, i.e., the entire plant is monitored by a single diagnoser. However, a great improvement can be obtained when the plant can be modeled as a collection of PN modules coupled through common places [10], since the diagnosis can be achieved by a set of local PN diagnosers communicating among each other.

In [11] a set of fundamental vectors is used to characterize the firing count vectors associated to the set of minimal sequences of unobservable transitions that can explain the firing of an observable transition; such sequences are called minimal explanations. These fundamental vectors can be computed by a tabular algorithm and they are used to build a deterministic automaton, called basis reachability tree (BRT), which provides a compact characterization of the net reachability set. Such an approach permits us to perform fault diagnosis and it can be regarded as an efficient compiled diagnoser. In [12] a similar approach was proposed but backward analysis on the net structure—by using the precondition of the observed event, the preconditions of the precondition and so on—is used to compute minimal explanations. The work [12] requires that the PN model is bounded, while this assumption is not required in this paper. On the other hand, as in [11] and [12] in this paper it is required that the unobservable portion of the PN model is acyclic.

A net unfolding approach for online asynchronous diagnosis was presented in [13]. This approach leads to an interpreted diagnoser where the state explosion is well kept under control, but the online computational effort may increase significantly due to the online building of PN structures—called diagnosis nets—via unfolding. Such nets are used to detect faults under the current net marking. Moreover, this approach can be applied only to safe PN.

In [14] an interpreted diagnoser has been devised only for safe PNs with an output function that associates an output vector to each net marking (interpreted PNs).

B. Contribution of the Paper

In this paper an interpreted diagnoser based on the online solution of programming problems is proposed. The problem of diagnosability, i.e., to decide a priori if a given fault can be isolated, is not addressed here. Hence, it is here assumed that all the unobservable events can be detected. In particular, diagnosability of the fault events is sufficient to distinguish between "a fault has occurred for sure" and "a fault may be not occurred". Moreover detectability of the unobservable events, either faulty and non-faulty, allows the proposed algorithm to distinguish between "a fault may have occurred" and "a fault has not occurred for sure".

To do that *g-markings* are introduced. If the firing of an observable transition t_o is observed and such a transition was enabled, then the previous net marking is updated according to PN state equation. In the presence of unobservable transitions, it may happen that t_o was not enabled under the current net marking since, in order to enable it, a sequence of unobservable transitions must have fired before. Hence, by using the PN state equation a unique marking having negative components is obtained from the firing of t_o, and this marking is here called g-marking. Since the sequence of unobservable transitions required to explain the firing of t_o is not always unique, to perform diagnosis other approaches require the estimation of the net marking, which in general is not a vector, but a set of vectors representing all the possible markings where the net can be in according to the observed sequences.

This problem is avoided here, since it is shown that linear programming problems based on g-markings can be used to compute online the firing count vectors associated to unobservable sequences that can explain the firing of t_o, and consequently also fault occurrences. If such a firing count vector is unique the g-marking is updated, otherwise more observations are needed to update it. It follows that the online implementation of the algorithm presented in this work is very efficient in terms of memory requirement.

A good compromise to speed up the online diagnosis is to precompute something offline. Thanks to the graphical representation of PN, it is shown that the programming problems to be solved by the diagnoser can be formulated on reduced portions of the net properly computed offline, without a big increase in terms of memory requirement. This is very interesting when the diagnoser must be implemented on embedded systems.

A discussion regarding the computational complexity of the online solution of the programming problems used in the proposed algorithm is also presented.

The following assumptions are used throughout this work:

- two different observable transitions cannot be associated to the same event;
- the subnet induced by the unobservable transitions is *acyclic*;
- the prefix-closed language associated with the net is *detectable*.

NOTATIONS AND ASSUMPTIONS

This section introduces the basic Petri net formalism, together with some additional notation, and with the assumptions exploited throughout the paper. For a complete review on Petri nets refer to [15].

C. Basic Petri Net Notation

A *Place/Transition* net (*P/T* net) is a 4-tuple $N = (P, T, \mathbf{Pre}, \mathbf{Post})$, where P is a set of m places (represented by circles), T is a set of n transitions (represented by empty boxes and each one associated to an event), $\mathbf{Pre} : P \times T \mapsto \mathbb{N}$ ($\mathbf{Post} : P \times T \mapsto \mathbb{N}$) is the *pre-* (*post-*) *incidence* matrix. $\mathbf{Pre}(p,t) = w$ ($\mathbf{Post}(p,t) = w$) means that there is an arc with weight w from p to t (from t to p); $\boldsymbol{C} = \mathbf{Post} - \mathbf{Pre}$ is the incidence matrix. The symbols ${}^\bullet p$ (${}^\bullet t$) and $p^\bullet$ ($t^\bullet$) are used for the *pre-set* and *post-set* of a place $p \in P$ (transition $t \in T$), respectively, e.g., ${}^\bullet t = \{p \in P \mid \mathbf{Pre}(p,t) \neq 0\}$.

A *marking* is a function $\boldsymbol{m} : P \mapsto \mathbb{N}$ that assigns to each place of a net a nonnegative integer number of tokens, drawn as black

dots. It is useful to represent the marking of a net with a vector $\boldsymbol{m} \in \mathbb{N}^m$. A *net system* $\mathcal{S} = \langle N, \boldsymbol{m}_0 \rangle$ is a net N with an initial marking $\boldsymbol{m}_0$. A transition t is enabled at $\boldsymbol{m}$ iff $\boldsymbol{m} \geq \mathbf{Pre}(\cdot, t)$ and this is denoted as $\boldsymbol{m}[t\rangle$. An enabled transition t may fire, yielding the marking $\boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{C}(\cdot, t)$, and this is denoted as $\boldsymbol{m}[t\rangle\boldsymbol{m}'$.

A *firing sequence* from $\boldsymbol{m}$ is a sequence of transitions $\sigma = t_1 t_2 \ldots t_k$ such that $\boldsymbol{m}[t_1\rangle\boldsymbol{m}_1[t_2\rangle\boldsymbol{m}_2 \ldots [t_k\rangle\boldsymbol{m}_k$, and this is denoted as $\boldsymbol{m}[\sigma\rangle\boldsymbol{m}_k$. An enabled sequence σ is denoted as $\boldsymbol{m}[\sigma\rangle$, while $t_i \in \sigma$ denotes that the transition t_i belongs to the sequence σ. The *empty* sequence is denoted as ν.

A marking $\boldsymbol{m}'$ is said to be *reachable* from $\boldsymbol{m}_0$ iff there exists a sequence σ such that $\boldsymbol{m}_0[\sigma\rangle\boldsymbol{m}'$. $R(N, \boldsymbol{m}_0)$ denotes the set of reachable markings of the net system $\langle N, \boldsymbol{m}_0 \rangle$.

The function $\boldsymbol{\sigma} : T \mapsto \mathbb{N}$, where $\boldsymbol{\sigma}(t)$ represents the number of occurrences of t in σ, is called *firing count vector* of the firing sequence σ. As it has been done for the marking of a net, the firing count vector is often denoted as a vector $\boldsymbol{\sigma} \in \mathbb{N}^n$. The notation $\boldsymbol{\sigma} = \pi(\sigma)$ is used to denote that $\boldsymbol{\sigma}$ is the firing count vector of σ, thus $\pi(\nu) = \boldsymbol{0}\boldsymbol{u} = \boldsymbol{0}$. Note that, if a sequence is made by a single transition, i.e., $\sigma = t_i$, then the corresponding firing count vector is the i-th canonical basis vector denoted as $\boldsymbol{e}_{t_i}$.

If $\boldsymbol{m}_0[\sigma\rangle\boldsymbol{m}$, then it is possible to write the vector equation

$$\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \tag{1}$$

which is called the *state equation* of the net system.

Theorem 1 ([15]): Consider a net system $\langle N, \boldsymbol{m}_0 \rangle$ and let N be an acyclic[1] net. It results $\boldsymbol{m} \in R(N, \boldsymbol{m}_0)$ iff there exists a nonnegative integer solution $\boldsymbol{\sigma}$ satisfying (1). ■

The set T can be partitioned into the disjoint sets of *observable* (represented by empty boxes) and *unobservable* transitions (represented by filled boxes), named respectively T_o and T_{uo}, with $|T_{uo}| = n_{uo} \leq n$. In this paper the fault events $t \in T_f$ are supposed to be unobservable, i.e., $T_f \subseteq T_{uo}$, with $|T_f| = n_f \leq n_{uo}$.

For the sake of simplicity the same name is used for each transition and the associated event, i.e., t denotes both the transition t and the corresponding event. Thus, T_{uo} denotes both the set of unobservable transitions and the set of unobservable events. The notation T_{uo}^* is used to denote the set of all possible sequences of unobservable events.

D. Induced Nets and Marking Projections

Definition 1 (T'-Induced Subnet): Given a net $N = (P, T, \mathbf{Pre}, \mathbf{Post})$, and a subset $T' \subseteq T$, the T'-induced subnet on N, denoted with $N' \prec_{T'} N$, is the 4-tuple $N' = (P', T', \mathbf{Pre}', \mathbf{Post}')$, where $P' = {}^\bullet T' \cup T'^\bullet$ while $\mathbf{Pre}'$ and $\mathbf{Post}'$ are the restrictions of $\mathbf{Pre}$ and $\mathbf{Post}$ to P' and T'.

Thus, the subnet $N' \prec_{T'} N$ can be obtained from N removing all the places which are not connected with any transition in T', and all the transitions in $T \setminus T'$. ◊

Definition 2 (Marking Projection on $P' \subseteq P$): Let $N = (P, T, \mathbf{Pre}, \mathbf{Post})$ and $P' \subseteq P$ a subset of places of N. The marking projection on P' is the restriction of the marking function to the places in P' and it is denoted with $\boldsymbol{m}_{|P'} : P' \mapsto \mathbb{N}$. ◊

[1]A net is acyclic if it does not include a directed circuit.

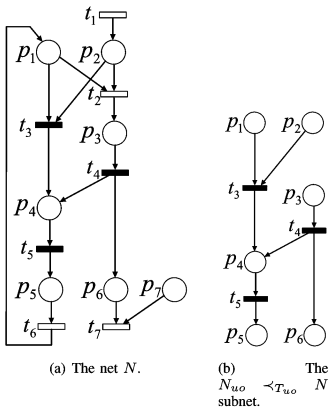


Fig. 1. Example of Petri net model.

To avoid awkward notation if $P' = \{\bar{p}\}$ is a singleton the marking projection on P' is denoted with $\boldsymbol{m}_{|\bar{p}}$ instead of $\boldsymbol{m}_{|P'}$. It is immediate to extend Definition 2 to the firing count vectors projection on $T' \subseteq T$, denoted as $\boldsymbol{\sigma}_{|T'}$.

Definition 3 (T'-Induced Net System): Consider net system $\mathcal{S} = \langle N, \boldsymbol{m}_0 \rangle$ and $T' \subseteq T$. The T'-induced net system on $\mathcal{S}$ is the net system $\mathcal{S}' = \langle N', \boldsymbol{m}_{0|P'} \rangle$, with $N' \prec_{T'} N$. ◊

Example 1: The net N shown in Fig. 1(a) has $T_o = \{t_1, t_2, t_6, t_7\}$ and $T_{uo} = \{t_3, t_4, t_5\}$. Fig. 1(b) shows the $N_{uo} \prec_{T_{uo}} N$ subnet, with $P_{uo} = \{p_1, p_2, p_3, p_4, p_5, p_6\}$.

Let $\boldsymbol{m}_0 = [4 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1]^T$ and $P' = \{p_1, p_3, p_7\}$, then the projection of the initial marking on P' is $\boldsymbol{m}_{0|P'} = [4 \;\; 0 \;\; 1]^T$, while the projection on P_{uo} is $\boldsymbol{m}_{0|P_{uo}} = [4 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 0]^T$.

Consider the firing count vector $\boldsymbol{\sigma} = [1 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0]^T$, its projection on T_{uo} is $\boldsymbol{\sigma}_{|T_{uo}} = [0 \;\; 1 \;\; 0]^T$. ◊

E. Diagnosability and Detectability

The notion of detectable prefix-closed and live language is given starting from the definition of diagnosability given in [1].

Consider a net $N = (P, T, \mathbf{Pre}, \mathbf{Post})$ with $T = T_{uo} \cup T_o$, and $T_f = \{t_{f_1}, t_{f_2}, \ldots, t_{f_k}\} \subseteq T_{uo}$.

Let $\overline{s}$ be the prefix-closure of any trace $s \in T^*$. We denote by L/s the post-language of L after s, i.e.,

$$L/s = \{q \in T^* \text{ s.t. } sq \in L\}.$$

Let $Pr : T^* \mapsto T_o^*$ be the usual projection [16], which "erases" the unobservable events in a trace s. The inverse projection operator Pr_L^{-1} is defined as

$$Pr_L^{-1}(r) = \{s \in L \text{ s.t. } Pr(s) = r\}.$$

Let $\dot{t}$ be the final event of trace s and define

$$\Psi(t_{f_i}) = \{s\dot{t} \in L \text{ s.t. } \dot{t} = t_{f_i}\}.$$

Under the assumptions that no cycles of unobservable events exists, and that the language L generated by N is live, it is possible to give the following definition of diagnosability.

Definition 4 (Diagnosable Prefix-Closed Live Language): A prefix-closed and live language L is said to be diagnosable w.r.t. Pr and T_f if

$$\begin{aligned}&\forall\, t_{f_i}\ \exists\, h_i \in \mathbb{N} \text{ such that the following holds}\\ &\forall\, s \in \Psi(t_{f_i}) \text{ and } \forall\, q \in L/s\\ &\|q\| \geq h_i \Rightarrow D\end{aligned}$$

where $\|q\|$ is the length of trace q, and the diagnosability condition is

$$r \in Pr_L^{-1}(Pr(sq)) \Rightarrow t_{f_i} \in r.$$

$\Diamond$

The above definition of diagnosability can be explained as follows. Let s be any trace generated by the system that ends in a failure event t_{f_i}, and let q be any sufficiently long continuation of s. Condition D implies that along every continuation q of s it is possible to detect the occurrence of t_{f_i} with a finite delay, specifically in at most h_i transitions of the system after s.

The notion of detectable language is derived from Definition 4 as follows:

Definition 5 (Detectable Prefix-Closed Live Language): A prefix-closed and live language L is said to be detectable w.r.t. Pr and T_{uo} if it is diagnosable w.r.t. Pr and T_{uo}. $\Diamond$

Note that detectability implies diagnosability, while undetectability does not necessarily imply undiagnosability. Furthermore undiagnosability implies undetectability.

F. Basic Assumptions

The assumptions exploited throughout the paper are presented in this section.

Assumption 1: Two different observable transitions cannot share the same event. $\Diamond$

The approach used in this paper is based on the fact that the firing of an observable transition requires a proper marking of its input places. If the same event is associated to more than one transition, the observation of an event could not necessarily correspond to the firing of a single transition. Taking into account this problem complicates too much the approach, and this is why Assumption 1 is made. A similar assumption is not required for unobservable transitions, which can be assumed to be associated to an arbitrary event, since such event is unobservable.

Assumption 2: Given a net N with $T = T_o \cup T_{uo}$ and $T_o \cap T_{uo} = \emptyset$, $N_{uo} \prec_{T_{uo}} N$ is *acyclic*, with $N_{uo} = (P_{uo}, T_{uo}, \mathbf{Pre}_{uo}, \mathbf{Post}_{uo})$. In words, the subnet induced by the unobservable transitions is *acyclic*. $\Diamond$

Assumption 2 is commonly adopted in the field of fault detection with PN models, and it corresponds to the fact that cycles of unobservable transitions are not admitted.

Under this assumption, it is well known that the PN state equation applied to N_{uo} does not have spurious solutions. Therefore it can be used to compute the firing count vectors associated to unobservable sequences that can explain the firing of observable transitions. Such an assumption is not necessary if N_{uo} belongs to some net subclasses—e.g., live *Marked Graphs* and *State Machine*. However, for the sake of generality, from now on this assumption is assumed to be true in this paper.

Assumption 3: Given a net N, the associated prefix-closed and live language L is supposed to be *detectable*. $\Diamond$

As it will be shown in Section IV, Assumption 3 is exploited to allow the proposed algorithm to distinguish both between "a fault has occurred for sure" and "a fault may be not occurred," and between "a fault may have occurred" and "a fault has not occurred for sure." It is worth noticing that only the hypothesis of diagnosable language is needed to decide if a fault is occurred for sure.

II. G-Marking

This section introduces the notion of *g-marking* of a Petri net, which extends the classic net marking presented in the previous section.

A. Why g-Marking?

Let $\boldsymbol{m}$ be the current net marking. If a sequence σ fires, a new marking $\boldsymbol{m}'$ is reached. From the state equation it follows that for the firing count vector $\boldsymbol{\sigma}$ it is possible to write

$$\boldsymbol{m} + \boldsymbol{C\sigma} = \boldsymbol{m}' \geq \mathbf{0}.$$

Suppose that $\sigma = \epsilon t$ where ϵ is a sequence of unobservable transitions and $t \in T_o$. Let $\boldsymbol{\mu} = \boldsymbol{m} + \boldsymbol{Ce}_t = \boldsymbol{m} + \boldsymbol{C}(\cdot, t)$. It may happen that $\boldsymbol{\mu}$ has negative components, since t may not be enabled under the marking $\boldsymbol{m}$. The negative components in $\boldsymbol{\mu}$ mean that the unobservable sequence ϵ must have fired in order to explain the firing of t, which is the unique observed event. A marking that may have negative components is called *g-marking*[2], and it will be formally defined in Section III-B.

Suppose that a fault event is associated to the unobservable transition t_f, and that one wants to know if t_f has occurred prior to the observation of t. Essentially, the problem is to check if the minimum number of occurrences of t_f required to explain the firing of t is greater than zero. Note that it is not necessary to compute explicitly ϵ or its firing count vector $\boldsymbol{\epsilon}$, but simply to check if $\boldsymbol{\epsilon}(t_f)$ is greater than zero.

How to compute $\boldsymbol{\epsilon}(t_f)$? Since the net marking is positive by definition, it follows that at least one firing count vector $\boldsymbol{\sigma}$ must exist such that

$$\boldsymbol{\mu} + \boldsymbol{C\hat{\sigma}} \geq \mathbf{0}. \tag{2}$$

Moreover, Theorem 1 says that, if $\boldsymbol{C}$ is the incidence matrix of an acyclic net, then any integer vector $\boldsymbol{\sigma} \geq 0$ which verifies (2) can be a solution to our problem.

Note that more than one solution may exist, since more than one sequence of unobservable transitions may explain the firing of t.

As an example consider the net in Fig. 1(a), and suppose that $\boldsymbol{m} = [1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]^T$ is the current marking of the net. If the firing of t_6 is observed, then a sequence of unobservable transitions must have fired to explain the firing of t_6. However, the firing of t_6 may be explained both by the sequences

[2] *g-marking* stands for *generalized marking*.

$\hat{\sigma}_1 = t_3 t_5$ and $\hat{\sigma}_2 = t_4 t_5$[3]. If t_3 models a fault, since $\boldsymbol{\sigma}_1(t_3) = 1$ and $\boldsymbol{\sigma}_2(t_3) = 0$, it is not possible to establish if the fault has occurred or not.

All the possible firing count vectors associated to sequences of unobservable transitions explaining t are given by

$$\boldsymbol{\Sigma} = \{\boldsymbol{\epsilon} \in \mathbb{N}^n \mid \boldsymbol{C}\boldsymbol{\epsilon} \geq -\boldsymbol{\mu} \text{ and } \boldsymbol{\epsilon}(t) = 0 \,\forall\, t \in T_o\}.$$

Hence, if the solution of the integer programming problem

$$\begin{aligned} \min \quad & \boldsymbol{\epsilon}(t_f) \\ \text{s.t.} \quad & \boldsymbol{\epsilon} \in \boldsymbol{\Sigma} \end{aligned} \tag{3}$$

is different from zero, it is possible to establish if t_f has occurred or not before the firing of t. From now on, a compact notation is used to denote ILP problems. For example, problem (3) will be denoted as $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}} \boldsymbol{\epsilon}(t_f)$.

The main contribution of this paper is to use an algorithm based on programming problems such as (3) to detect occurrences of faults associated to unobservable transitions, every time an observable transition occurs. Then, an interpreted diagnoser is obtained which relies on programming problems which is a standard mathematical tool unlike net unfolding, and which requires the plant net—as a matter of fact the unobservable portion only—to be acyclic. It may be argued that the computational effort of solving online ILPs such as (3) makes the proposed approach not effective from a practical point of view. However, exploiting the graphical representation of PNs, the computational effort can be reduced. In particular:

1) The complexity of programming problems can be significantly reduced by formulating it on proper subnets of plant model. A first reduction can be achieved considering only the unobservable portion of the plant net. Moreover, it will be shown that a further reduction can be obtained by considering only the portion of unobservable subnet influencing the specific fault transition. In this way, a trade off between online computation and memory requirement is obtained, since such subnets have to be precomputed.
2) If the ILP is formulated on a subnet which is TS1 or TS2 net,[4] then its solution has a closed form which consists of logical predicates, as it has been proved in [17]. Thus, at low cost of storing the logical predicates, a significant improvement of online computational effort is achieved, since the logical predicates are simple algebraic expressions.
3) If the ILP is formulated on a subnet whose incidence matrix is totally unimodular, it results to be a linear programming problem, which has polynomial complexity [18]. If such a subnet is a *Marked Graph*, then its incidence matrix is totally unimodular. Moreover, if the subnet is a *Backward-Conflict Free-Choice Net*[5], then it can be decomposed in marked graph components and, even in this case, the solution can be obtained as solutions of linear programming problems [19].

[3]Note that in the considered example there are 8 other sequences of unobservable transitions that can explain the firing of t_6.

[4]A PN is said to have a generalized tree structure if it is acyclic and the places on one level of PN graph acts as sources for the transition on the next level. These transitions then act as the exclusive sources of tokens for the places in the next level. Places on one level receive tokens exclusively from the transitions one level back and deliver tokens to the transitions one level forward. A net N has a *tree structure of type 1* TS1 if it has a generalized tree structure and each transition has at most one output arc. A net N has a *tree structure of type 2* TS2 if it has a generalized tree structure and each place has at most one output arc.

B. Formal Definition

Definition 6 (g-Marking): A *g-marking* is a function $\boldsymbol{\mu} : P \mapsto \mathbb{Z}$ that assigns an integer number to each place of a net. Contrary to the standard marking introduced in Section II-A, a transition t is enabled at $\boldsymbol{\mu}$ iff one of the following two conditions is true:

ia) $t \in T_o$,

iia) $t \in T_{uo}$ and $\exists\, \sigma \in T_{uo}^*$ s.t. $\boldsymbol{\mu}' = \boldsymbol{\mu} + \boldsymbol{C}\boldsymbol{\sigma} \geq \mathbf{0}$, $t \in \sigma$, with $\boldsymbol{\sigma} = \pi(\sigma)$.

The notation $\boldsymbol{\mu}[t\rangle$ denotes that t is enabled at $\boldsymbol{\mu}$.

A transition t may fire if:

ib) $t \in T_o$ is enabled[6] and its firing has been observed.

iib) $t \in T_{uo}$ is enabled,

When a transition t fires, it yields the g-marking $\boldsymbol{\mu}' = \boldsymbol{\mu} + \boldsymbol{C}(\cdot, t)$, this is denoted as $\boldsymbol{\mu}[t\rangle\boldsymbol{\mu}'$.

A firing sequence from $\boldsymbol{\mu}$ is a sequence of transitions $\sigma = t_1 t_2 \ldots t_k$ such that $\boldsymbol{\mu}[t_1\rangle\boldsymbol{\mu}_1[t_2\rangle\boldsymbol{\mu}_2 \ldots [t_k\rangle\boldsymbol{\mu}_k$ and this is denoted as $\boldsymbol{\mu}[\sigma\rangle\boldsymbol{\mu}_k$. If the sequence σ is enabled at $\boldsymbol{\mu}$ this is denoted as $\boldsymbol{\mu}[\sigma\rangle$.

Given a net system $\mathcal{S} = \langle N, \boldsymbol{m}_0\rangle$, the corresponding *g-net system* $\mathcal{S} = \langle N, \boldsymbol{\mu}_0\rangle$ is a net N with the initial g-marking $\boldsymbol{\mu}_0 = \boldsymbol{m}_0$.

The marking projection previously defined is naturally extended to the g-markings (see Example 2). ◊

The conditions **ia)** and **iia)** are the *g-enabling conditions*, while **ib)** and **iib)** are the *g-firing conditions*. It is worth to notice that checking the condition **iia)** is not straightforward, as it is for the standard enabling rule given in Section II-A. Nevertheless Sections IV–VI will show how this rule is exploited to perform fault diagnosis.

The following example shows how the evolution rules for the g-marking given above can lead to negative marking components.

Example 2: Consider the net N in Fig. 1(a) and let $\boldsymbol{\mu}_0 = [1 \;\; 1 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 0]^T$ be the initial g-marking of the net. If the firing of $t_6 \in T_o$ is observed, then t_6 may fire, since an observable transition is always enabled under any g-marking. The firing of t_6 yields the g-marking $\boldsymbol{\mu}_1 = \boldsymbol{m}u_0 + \boldsymbol{C}(\cdot, t_6) = [2 \;\; 1 \;\; 1 \;\; 0 - 1 \;\; 0 \;\; 0]^T$. The projection of $\boldsymbol{\mu}_1$ on P_{uo} is $\boldsymbol{\mu}_{1|P_{uo}} = [2 \;\; 1 \;\; 1 \;\; 0 - 1 \;\; 0]^T$. The negative marking $\boldsymbol{\mu}_{1|p_5} = -1$ means that an unobservable sequence $\sigma \in T_{uo}^*$ must have fired to explain the firing of t_6. In this simple example it is immediate to observe that $\sigma_1 = t_3 t_5$, $\sigma_2 = t_4 t_5$, $\sigma_3 = t_3 t_4 t_5$, $\sigma_4 = t_3 t_5 t_4$, $\sigma_5 = t_4 t_3 t_5$, $\sigma_6 = t_4 t_5 t_3$, $\sigma_7 = t_3 t_5 t_4 t_5$, $\sigma_8 = t_3 t_4 t_5 t_5$,

[5]Backward-Conflict Free-Choice Nets (BCFCNs) are ordinary nets that admits a free-choice relationship (each arc from a place to a transition is either the unique outgoing arc from the place or the unique coming arc of the transition), together with a backward-conflict relationship (each place has only one incoming arc—no join places are allowed). BCFCNs can model not only concurrency and synchronization of activities—as the Marked Graph subclass — but also decisions.

[6]This condition is superfluous since $t \in T_o$ is always enabled, but it is stated for completeness.

$\sigma_9 = t_4t_5t_3t_5$, and $\sigma_{10} = t_4t_3t_5t_5$, is the set of all the sequences of unobservable transitions σ such that $\boldsymbol{\mu}_1 + \boldsymbol{C\sigma} \geq \mathbf{0}$, with $\pi(\sigma) = \boldsymbol{\sigma}$. Thus the unobservable transitions t_3, t_4, and t_5 are enabled at $\boldsymbol{\mu}_1$, since they belong to unobservable sequences $\sigma \in T_{uo}^*$ s.t. $\boldsymbol{\mu} = \boldsymbol{\mu}_1 + \boldsymbol{C\sigma} \geq \mathbf{0}$ with $\pi(\sigma) = \boldsymbol{\sigma}$. It follows that one of these enabled transitions can fire at $\boldsymbol{\mu}_1$. In particular the firing of t_3 yields the g-marking $\boldsymbol{\mu}_2 = \boldsymbol{\mu}_1 + \boldsymbol{C}(\cdot, t_3) = [1 \quad 0 \quad 1 \quad 1-1 \quad 0 \quad 0]^T$, with t_4 and t_5 still enabled at $\boldsymbol{\mu}_2$, while the firing of t_5 yields $\boldsymbol{\mu}_3 = \boldsymbol{\mu}_1 + \boldsymbol{C}(\cdot, t_5) = [2 \quad 1 \quad 1-1 \quad 0 \quad 0 \quad 0]^T$, with t_3, t_4, and t_5 enabled at $\boldsymbol{\mu}_3$. ◊

Throughout this paper, the negative components of a g-marking represent the tokens that are needed to explain either the firing of an observed transition, or the firing of an unobservable transition that must have fired.

As far as the fault diagnosis is concerned, the g-markings allow the fault diagnosis agent to store in a compact way all the needed information about the state space estimation, as it will be shown in the next.

III. Unobservable Explanations and Related Results

Unobservable explanations, together with the related results, are presented in this section. Given a g-marking $\boldsymbol{\mu}$, the unobservable explanations are the sequences of unobservable transitions which are enabled at $\boldsymbol{\mu}$.

The following definitions are given for a net $N = (P, T, \mathbf{Pre}, \mathbf{Post})$, with $T = T_o \cup T_{uo}$, $N_{uo} \prec_{T_{uo}} N$ and $T_f \subseteq T_{uo}$.

Definition 7: Given a g-marking $\boldsymbol{\mu} \in \mathbb{Z}^m$

$$\Sigma(N, \boldsymbol{\mu}) = \{\epsilon \in T_{uo}^* \mid \boldsymbol{\mu}[\epsilon\rangle \boldsymbol{\mu}' \text{ s.t. } \boldsymbol{\mu}' \geq 0\}$$

is the set of all the *unobservable explanations* enabled at $\boldsymbol{\mu}$ and

$$\boldsymbol{\Sigma}(N, \boldsymbol{\mu}) = \{\boldsymbol{\epsilon} \in \mathbb{N}^n \mid \exists\, \epsilon \in \Sigma(N, \boldsymbol{\mu}) \text{ s.t. } \pi(\epsilon) = \boldsymbol{\epsilon}\}$$

is the corresponding set of firing count vectors. ◊

The notation $\Sigma(N, \boldsymbol{\mu})$ makes clear the dependence of the unobservable explanations on the net structure.

In order to detect a fault transition t_f, it is sufficient to consider a specific subset of $\Sigma(N, \boldsymbol{\mu})$. In particular the subset of unobservable sequences containing the given fault transition t_f has to be considered.

Definition 8: Given a g-marking $\boldsymbol{\mu} \in \mathbb{Z}^m$ and a fault transition $t_f \in T_f$

$$\Sigma_f(N, \boldsymbol{\mu}, t_f) = \{\epsilon \in T_{uo}^* \mid \boldsymbol{\mu}[\epsilon\rangle \boldsymbol{\mu}' \text{ s.t. } \boldsymbol{\mu}' \geq 0 \text{ and } \boldsymbol{\epsilon}(t_f) \neq 0, \text{ with } \boldsymbol{\epsilon} = \pi(\epsilon)\}$$

is the set of all the *faulty unobservable explanations* of t_f enabled at $\boldsymbol{\mu}$ and

$$\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f) = \{\boldsymbol{\epsilon} \in \mathbb{N}^n \mid \exists\, \boldsymbol{\epsilon} \in \Sigma_f(N, \boldsymbol{\mu}, t_f) \text{s.t. } \pi(\epsilon) = \boldsymbol{\epsilon}\}$$

is the corresponding set of firing count vectors. ◊

From the definitions given above it follows that $\Sigma_f(N, \boldsymbol{\mu}, t_f) \subseteq \Sigma(N, \boldsymbol{\mu})$ and $|\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)| \leq |\boldsymbol{\Sigma}(N, \boldsymbol{\mu})|$.

Remark 1: It is important to remark the following properties of the empty sequence and its firing count vector:

- the empty sequence ν belongs to $\Sigma(N, \boldsymbol{\mu})$ only if $\boldsymbol{\mu} \geq \mathbf{0}$, since if $\boldsymbol{\mu} \ngeq \mathbf{0}$ the firing of the empty sequence does not change the actual g-marking, i.e., $\boldsymbol{\mu}[\nu\rangle \boldsymbol{\mu}$, and therefore it is not possible to explain any firing;
- the empty sequence ν does not belong to $\Sigma_f(N, \boldsymbol{\mu}, t_f)$ since $\boldsymbol{\nu}(t_f) = 0$.

◊

The following theorem follows directly from the definitions of g-marking and of $\boldsymbol{\Sigma}(N, \boldsymbol{\mu})$.

Theorem 2: Consider a net N with $T = T_o \cup T_{uo}$, and a g-marking $\boldsymbol{\mu}$. If $\boldsymbol{\mu}$ has at least one negative component, i.e $\boldsymbol{\mu} \ngeq \mathbf{0}$, then $\boldsymbol{\Sigma}(N, \boldsymbol{\mu}) \neq \emptyset$, i.e., $|\boldsymbol{\Sigma}(N, \boldsymbol{\mu})| \geq 1$, but $\boldsymbol{\nu} \notin \boldsymbol{\Sigma}(N, \boldsymbol{\mu})$.

Proof: If $\boldsymbol{\mu} \ngeq \mathbf{0}$, since all the negative components of a g-marking must be explained by the firing of an unobservable sequence, there must be at least one non-empty unobservable explanation, thus $|\boldsymbol{\Sigma}(N, \boldsymbol{\mu})| \geq 1$. ■

As it has been noticed in Section III-A, more than one possible explanation could exist, each one leading to a different marking estimation. Thus fault diagnosis approaches based on conventional state observers may suffer from problems related to the explosion of the state space estimation.

Remark 2: If $|\boldsymbol{\Sigma}(N, \boldsymbol{\mu})| = 1$, it means that all the unobservable explanations have the same firing count vector, thus they all yield the same g-marking $\boldsymbol{\mu}' = \boldsymbol{\mu} + \boldsymbol{C\epsilon}'$, with $\boldsymbol{\Sigma}(N, \boldsymbol{\mu}) = \{\boldsymbol{\epsilon}'\}$. ◊

The following theorem states a sufficient condition under which $|\boldsymbol{\Sigma}(N, \boldsymbol{\mu})| = 1$.

Theorem 3 ([20]): If $\boldsymbol{\mu} \ngeq \mathbf{0}$ is a g-marking, and the subnet $N_{uo} \prec_{T_{uo}} N$ is *Backward Conflict-Free*[7], then $|\boldsymbol{\Sigma}(N, \boldsymbol{\mu})| = 1$. ■

The theorems 4 and 5 presented further on are the main results of this section, and they are used in the proposed fault diagnosis algorithm in Section V.

Theorem 4: Consider a net N, with $T = T_o \cup T_{uo}$. Let $\boldsymbol{\mu} \ngeq \mathbf{0}$ be a g-marking, and $t_f \in T_f \subseteq T_{uo}$ a fault transition, then

$$|\boldsymbol{\Sigma}(N, \boldsymbol{\mu})| = |\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)| > 0 \iff \min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N, \boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) \neq 0.$$

Proof: (if). Suppose that $|\boldsymbol{\Sigma}(N, \boldsymbol{\mu})| = |\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)| > 0$. If $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N, \boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) = 0$, since $\boldsymbol{\mu} \ngeq \mathbf{0}$, it follows that exists a non-empty unobservable explanation ϵ such that $\boldsymbol{\epsilon} = \pi(\epsilon)$, and

$$\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N, \boldsymbol{\mu}) \text{ with } \boldsymbol{\epsilon}(t_f) = 0$$

which implies that $\boldsymbol{\epsilon} \notin \boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)$, hence $|\boldsymbol{\Sigma}(N, \boldsymbol{\mu})| \geq |\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)| + 1$, contradicting the initial assumption.

(only if). Suppose that $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N, \boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) \neq 0$. If $|\boldsymbol{\Sigma}(N, \boldsymbol{\mu})| \neq |\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)|$, then $|\boldsymbol{\Sigma}(N, \boldsymbol{\mu})| > |\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)|$, since $\Sigma_f(N, \boldsymbol{\mu}, t_f) \subseteq \Sigma(N, \boldsymbol{\mu})$. If $\boldsymbol{\mu} \ngeq \mathbf{0}$, then there exists a non-empty unobservable explanation ϵ such that $\boldsymbol{\epsilon} = \pi(\epsilon)$, and

$$\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N, \boldsymbol{\mu}) \text{ and } \boldsymbol{\epsilon} \notin \boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)$$

which implies $\boldsymbol{\epsilon}(t_f) = 0$, and therefore $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N, \boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) = 0$, contradicting the initial hypothesis. ■

Note that, if $\boldsymbol{\mu} \geq \mathbf{0}$, then $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N, \boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) = 0$ since $\boldsymbol{\nu}$ belongs to $\boldsymbol{\Sigma}(N, \boldsymbol{\mu}, t_f)$ (see Remark 1).

[7]A net is Backward Conflict-Free, if each place has only one incoming arc.

Theorem 5: Consider a net N, with $T = T_o \cup T_{uo}$. Let $\boldsymbol{\mu}$ be a g-marking, and $t_f \in T_f \subseteq T_{uo}$ a fault transition, then

$$|\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)| \neq 0 \iff \max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) \neq 0.$$

Proof: (if). Suppose that $|\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)| \neq 0$. If $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) = 0$, then

$$\forall\, \boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N, \boldsymbol{\mu}),\ \boldsymbol{\epsilon}(t_f) = 0 \Longrightarrow \boldsymbol{\epsilon} \notin \boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)$$

since $\Sigma_f(N, \boldsymbol{\mu}, t_f) \subseteq \Sigma(N, \boldsymbol{\mu})$, it follows that $|\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)| = 0$, which contradicts the hypothesis.

(only if). Suppose that $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) \neq 0$. If $|\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)| = 0$ then

$$\Sigma_f(N, \boldsymbol{\mu}, t_f) = \emptyset \Longrightarrow \forall\, \boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N, \boldsymbol{\mu}), \quad \boldsymbol{\epsilon}(t_f) = 0$$

which implies $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) = 0$, contradicting the initial assumption. ∎

Theorem 6: Consider a net N, with $T = T_o \cup T_{uo}$. It holds

$$\overline{\boldsymbol{\Sigma}}(N, \boldsymbol{\mu}) = \{\boldsymbol{\epsilon} \in \mathbb{N}^n \mid \boldsymbol{C}_{uo}\boldsymbol{\epsilon}_{|T_{uo}} \geq -\boldsymbol{\mu}_{|P_{uo}} \text{ and } \boldsymbol{\epsilon}_{|T_o} = \mathbf{0}\}.$$

Proof: The proof follows from Theorem 1. ∎

Thus $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f)$ and $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f)$ can be computed by solving programming problems on the $N_{uo} \prec_{T_{uo}} N$ subnet, which in general are integer linear programming problems. However, as it has been discussed in Section III-A, there are several cases in which such ILPs can be reduced either to linear programming problems, or to the evaluation of logical predicates, as it will be shown in Section VI.

The following corollary is a direct consequence of Theorem 6.

Corollary 1: Consider a net N, with $T = T_o \cup T_{uo}$. It holds $|\boldsymbol{\Sigma}(N, \boldsymbol{\mu})| = 1$ iff $\boldsymbol{C}_{uo}\boldsymbol{\epsilon}_{|T_{uo}} \geq -\boldsymbol{\mu}_{|P_{uo}}$ admits only one solution $\boldsymbol{\epsilon}^*$, such that $\boldsymbol{\epsilon}^*_{|T_o} = \mathbf{0}$. ∎

It is worth remarking that the Gauss elimination method can be used to check if a linear system has a solution. This method has polynomial complexity, therefore it is suitable for online computation [18].

The next two propositions make clear the need of Assumption 3 to distinguish both between "a fault has occurred for sure" and "a fault may be not occurred," and between "a fault may have occurred" and "a fault has not occurred for sure." In particular, the proposition stated below proves that diagnosability is sufficient to decide if a fault is occurred for sure.

Proposition 1: Let L be diagnosable w.r.t. Pr and T_f. If $s \in \Psi(t_{f_i})$ then exists $q \in L/s$, such that

$$\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_{f_i}) > 0$$

with $\boldsymbol{\mu}_0[z\rangle\boldsymbol{\mu}$, and $z = Pr(sq)$.

Proof: Let L be diagnosable w.r.t. Pr and T_f. If $s \in \Psi(t_{f_i})$, and consider $s = t_{f_i} \in \Psi(t_{f_i})$. Suppose, *ad absurdum*, that for all $q \in L/s$ is

$$\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_{f_i}) = 0$$

where

$$\boldsymbol{\mu} = \boldsymbol{\mu}_0 + \boldsymbol{C}\boldsymbol{z}$$

and $\boldsymbol{z} = \pi(z)$, with $z = Pr(sq)$.

It follows that $\forall\ h \in \mathbb{N}$, $\|q\| > h$, it should exist at least one sequence $\hat{\sigma} \in T^*_{uo}$ which explains the firing of z, and such that $t_{f_i} \notin \hat{\sigma}$. Hence L is not diagnosable, which contradicts the hypothesis. ∎

Proposition 2 proves that detectability is sufficient to decide if a fault may have occurred.

Proposition 2: Let L be detectable w.r.t. Pr and T_{uo}. If s is a sequence which enables the firing of t_{f_i} and $t_{f_i} \notin s$, then it exists $h \in \mathbb{N}$ such that for all sequences $q \in L/s$ whose firing does not enable t_{f_i}, and $\|q\| > h$, $t_{f_i} \notin q$, it holds that

$$\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu}'')} \boldsymbol{\epsilon}(t_{f_i}) = 0$$

with $\boldsymbol{\mu}_0[\beta\rangle\boldsymbol{\mu}''$, and $\beta = Pr(sq)$.

Proof: Consider a sequence s which enables the firing of t_{f_i}, and $t_{f_i} \notin s$. By Theorem 5 it holds that

$$\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu}')} \boldsymbol{\epsilon}(t_{f_i}) > 0$$

with $\boldsymbol{\mu}_0[\alpha\rangle\boldsymbol{\mu}'$, and $\alpha = Pr(s)$.

Suppose, *ad absurdum*, that for all $h \in \mathbb{N}$, and for all sequences $q \in L/s$ whose firing disable t_{f_i}, with $t_{f_i} \notin q$, $\|q\| > h$, it is

$$\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu}'')} \boldsymbol{\epsilon}(t_{f_i}) > 0$$

with $\boldsymbol{\mu}[\beta\rangle\boldsymbol{\mu}''$, and $\beta = Pr(sq)$. Although t_{f_i} has been disabled by the firing of q, for every $\boldsymbol{\mu}''$ there is at least one negative component that can be explained either by the firing of t_{f_i}, or by the firing of another unobservable transition $\bar{t}$ (which should have fired since t_{f_i} is disabled). Hence, for every possible β there exist at least two sequences $\hat{\epsilon}_1, \hat{\epsilon}_2 \in \Sigma(N, \boldsymbol{\mu}'')$ which could explain the firing of the observable transitions in β, and such that $t_{f_i} \in \hat{\epsilon}_1$, $t_{f_i} \notin \hat{\epsilon}_2$, and $\bar{t} \notin \hat{\epsilon}_1$, $\bar{t} \in \hat{\epsilon}_2$. Therefore it is not possible to find any continuation of q which permits to detect the occurrence of $\bar{t}$. It follows that L is undetectable, contradicting the initial hypothesis. ∎

IV. Fault Detection Algorithm

As it has been discussed in the previous section, the cardinalities of $\boldsymbol{\Sigma}(N, \boldsymbol{\mu})$ and $\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)$ permit to perform the diagnosis of the fault t_f. In particular, it is possible to establish the occurrences of a fault t_f in terms of conditions on both $|\boldsymbol{\Sigma}(N, \boldsymbol{\mu})|$ and $|\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)|$. Such conditions can be equivalently expressed in terms of conditions on the values of $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f)$ and $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f)$, as it is claimed by the following proposition.

Proposition 3 (Fault Detection): Consider a net N, with $T = T_o \cup T_{uo}$. Let $t_f \in T_f \subseteq T_{uo}$, and $\boldsymbol{\mu}$ be a g-marking. In order to perform the detection of the fault t_f, the following three conditions have to be checked:

1a) $\boldsymbol{\mu} \not\geq \mathbf{0}$ and $|\boldsymbol{\Sigma}(N, \boldsymbol{\mu})| = |\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)| > 0 \iff t_f$ has occurred;

2a) $|\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)| = 0 \iff t_f$ has not occurred;

Require: $\boldsymbol{C}, \boldsymbol{m}_0, T_o, T_{uo}, T_f$

1 $\boldsymbol{\mu} = \boldsymbol{\mu}_0 = \boldsymbol{m}_0$ (* Initialization *)
2 **for all** $t_{f_i} \in T_f$ **do**
2.1 **if** $\boldsymbol{\mu} \not\geqslant \mathbf{0}$,
then (* if the g-marking has at least one negative component *)
2.1.1 **if** $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_{f_i}) = F \neq 0$,
then (* t_{f_i} has occurred F times *)
2.1.1.1 report that t_{f_i} **has occurred**
2.1.1.2 $\boldsymbol{\mu}_{|P_{uo}} = \boldsymbol{\mu}_{|P_{uo}} + \boldsymbol{C}_{uo}(\cdot, t_{f_i})F$ (* Update $\boldsymbol{\mu}$ *)
2.1.1.3 **go to** Step **2** (* Restart the **for** cycle *)
2.2 **if** $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_{f_i}) = G \neq 0$,
then report that t_{f_i} **may have occurred**
(* t_{f_i} may be occurred G times *)
2.3 **else** report that t_{f_i} **has not occurred yet**
3 **end for**
4 **if** $\boldsymbol{C}_{uo}\boldsymbol{\epsilon}_{|T_{uo}} \geq -\boldsymbol{\mu}_{|P_{uo}}$ admits only one solution $\boldsymbol{\epsilon}^*$ s.t. $\boldsymbol{\epsilon}^*_{|T_o} = \mathbf{0}$,
then $\boldsymbol{\mu}_{|P_{uo}} = \boldsymbol{\mu}_{|P_{uo}} + \boldsymbol{C}_{uo}\boldsymbol{\epsilon}^*_{|T_{uo}}$ (* Update $\boldsymbol{\mu}$ *)
5 **wait for** a new observed transition $\bar{t} \in T_o$
6 $\boldsymbol{\mu} = \boldsymbol{\mu} + \boldsymbol{C}(\cdot, \bar{t})$ (* Update $\boldsymbol{\mu}$ *)
7 **go to** Step **2**

Fig. 2. Fault detection algorithm.

3a) $|\boldsymbol{\Sigma}_f(N, \boldsymbol{\mu}, t_f)| \neq 0 \iff t_f$ may have occurred.

The three conditions listed above are equivalent to the following ones respectively:

1b) $\boldsymbol{\mu} \not\geqslant \mathbf{0}$ and $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) \neq 0 \iff t_f$ has occurred;

2b) $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) = 0 \iff t_f$ has not occurred;

3b) $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) \neq 0 \iff t_f$ may have occurred.

If condition **1b)** holds and $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) = F$, then t_f has occurred at least F times. While if condition **3b)** holds and $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) = G$, then the t_f may fire at most G times. Moreover if $F = G$, then the fault has occurred exactly $F = G$ times.

Proof: The proof is a direct consequence of Theorem 4, Theorem 5, Proposition 1 and Proposition 2. ■

Thus, the solutions of the programming problems $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f)$ and $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f)$ allow the diagnosis agent to perform the fault diagnosis. Moreover the proposed approach allows us to determine if a fault could happen or not and, if yes, at most how many times. In this sense this approach performs *fault prognosis*, i.e., what could have happened or is about to happen in the future, as stated in [12].

Exploiting the results given so far, it is now possible to introduce the algorithm for fault detection and identification reported in Fig. 2.

In order to perform fault detection, the proposed algorithm uses an estimation of the g-marking. In particular

- if the new available g-marking estimation has at least one negative component, then $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f)$ is computed for each fault transition, so as to establish if the considered fault has occurred (step **2.1**).
- $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f)$ is computed for every new g-marking estimation, and for each fault transition, so as to perform fault prognosis (steps **2.2** and **2.3**). Note that $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f)$ has to be computed only if we are interested also in fault prognosis. Thus, if this is not the case, the number of programming problems to be solved reduces significantly.

The g-marking estimation is updated when:

- a fault transition $t \in T_{uo}$ is diagnosed (step **2.1.1.2**);
- all the unobservable explanations have the same firing vector, i.e., $|\boldsymbol{\Sigma}(N, \boldsymbol{\mu})| = 1$ (step **4**);
- the firing of $t \in T_o$ is observed (step **6**).

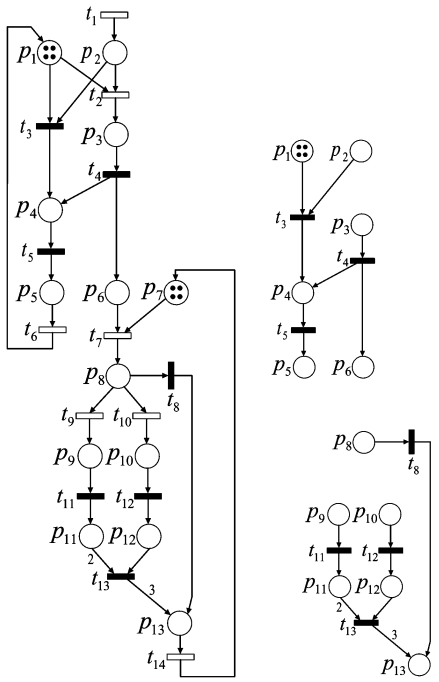
The example in the next subsection shows how the proposed algorithm works on a manufacturing system.

A. Example: A Manufacturing System

Let us consider the manufacturing system whose Petri net model is shown in Fig. 3(a), with $T_o = \{t_1, t_2, t_6, t_7, t_9, t_{10}, t_{14}\}$ and $T_{uo} = \{t_3, t_4, t_5, t_8, t_{11}, t_{12}, t_{13}\}$.

This manufacturing system consists of two machines, $\mathcal{M}_1$ and $\mathcal{M}_2$, which process parts conveyed on pallets. Machine $\mathcal{M}_1$ processes single parts, and it is modeled by the transition set $\mathcal{T}_1 = \{t_2, t_3, t_4, t_5, t_6\}$. When $\mathcal{M}_1$ ends its operations the pallets must be washed before they are made available again. The parts coming from $\mathcal{M}_1$ are then processed by machine $\mathcal{M}_2$, which is modeled by the set $\mathcal{T}_2 = \{t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}\}$. Two different jobs, named J_1 and J_2, can be executed by $\mathcal{M}_2$. Machine $\mathcal{M}_2$ outputs a finished part after having executed two times J_1 and one time J_2. In particular:

- t_1 fires when a new part to be processed by machine $\mathcal{M}_1$ is available;
- t_2 (t_7) fires when $\mathcal{M}_1$ ($\mathcal{M}_2$) picks up a pallet, and starts to work a part;
- t_3 (t_8) models a faulty behavior of machine $\mathcal{M}_1$ ($\mathcal{M}_2$)—hence $T_f = \{t_3, t_8\}$;
- t_4 models an unobservable manufacturing step of machine $\mathcal{M}_1$;



(a) The net system $\mathcal{S} = \langle N, \boldsymbol{m}_0 \rangle$. (b) The $N_{uo} \prec_{T_{uo}} N$ subnet.

Fig. 3. Petri net model of the manufacturing system.

- t_5 is an unobservable transition which fires when a pallet is washed after being processed by $\mathcal{M}_1$;
- t_6 (t_{14}) fires when $\mathcal{M}_1$ ($\mathcal{M}_2$) ends its operation;
- t_9, t_{10} model the two different manufacturing jobs, J_1 and J_2, of machine $\mathcal{M}_2$;
- t_{11} and t_{12} model two unobservable manufacturing steps of machine $\mathcal{M}_2$;
- t_{13} is an unobservable transition which fires when a finished part is produced by $\mathcal{M}_2$;
- the initial number of tokens in p_1 and p_7 model the fact that each machine can process up to four parts simultaneously.

Some execution steps of the proposed algorithm are reported in Fig. 4. In this case, if the estimated g-marking has negative components, then the two ILPs $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_3)$ and $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_8)$ are solved so as to perform fault detection and identification. The remaining two ILPs, $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_3)$ and $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_8)$, are solved for each new estimation, so as to perform fault prognosis.

V. Complexity Reduction Using Subnets

This section deals with the problem of reducing the computational effort needed to solve the ILPs in the fault detection algorithm proposed in Section V.

The main idea to improve the computational efficiency, is to solve each ILP on a subnet which is smaller than the whole plant model, since the complexity of ILPs strongly depends on the number of variables [18], i.e., the number of places.

Hence, it is necessary to store in memory a number of subnets equal to the number of different ILPs that have to be evaluated. Since these subnets are calculated offline, the only price to be paid in order to improve the online efficiency of the proposed algorithm, is the increase of memory needed to store these additional subnets. However this increase is not comparable to the memory requirement of a typical compiled approach to fault diagnosis, where the whole diagnoser state space must be stored.

A. Pre-Net and Post-Net of a Fault Transition

In the proposed fault detection algorithm, two ILPs, $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f)$ and $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f)$, must be solved for each fault transition t_f. In order to reduce the complexity of each of these ILPs, only a specific part of net can be considered, i.e., only the portion of unobservable subnet that influences the specific fault transition.

A first reduction can be achieved considering only the $N_{uo} \prec_{T_{uo}} N$ subnet, as it has been done in the previous section. An additional reduction can be made considering only the portion of the $N_{uo} \prec_{T_{uo}} N$ subnet that influences a given fault transition t_f. At this aim, it is very efficient to use the graphical nature of PNs. This approach was initiated by Holloway and Krogh which gave a computationally efficient solution for the forbidden-state problem of marked graphs on the basis of the influence paths of forbidden places [21].

As an example, for the manufacturing system introduced in Section V-A, it can be noticed that the ILPs $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_3)$ and $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_3)$ can be solved considering the subnet induced by $T' = \{t_3, t_4, t_5\}$, since the firing of t_3 does not depend on the firing of t_8, t_{11}, t_{12} and t_{13}. Similarly, the ILPs $\min_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_8)$ and $\max_{\boldsymbol{\epsilon} \in \boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_8)$ can be solved on the subnet induced by $T'' = \{t_8, t_{11}, t_{12}, t_{13}\}$.

Given a fault transition t_f, a further increase of the computational efficiency can be attained considering the two subnets defined below.

Definition 9 (Pre-Net of t_f): Given a net N with $T = T_o \cup T_{uo}$. Let $t_f \in T_f \subseteq T_{uo}$ and consider the following set of unobservable transitions:

$$T \downarrow (t_f) = \{t \in T_{uo} \mid \exists\, p \in {}^\bullet t_f \text{ such that a path from } t \text{ to } p \text{ exists in } N_{uo}\} \cup \{t_f\}.$$

The *pre-net* of t_f, denoted as $\mathcal{N} \downarrow (t_f)$, is equal to the subnet $\hat{N} \prec_{T\downarrow(t_f)} N_{uo}$, if ${}^\bullet(t_f^\bullet) \neq \{t_f\}$ in $\hat{N}$. If ${}^\bullet(t_f^\bullet) = \{t_f\}$ in $\hat{N}$, then $\mathcal{N} \downarrow (t_f)$ is obtained from $\hat{N}$ removing the places in $t_f^\bullet$. $\Diamond$

Definition 10 (Post-Net of t_f): Given a net N with $T = T_o \cup T_{uo}$. Let $t_f \in T_f \subseteq T_{uo}$ and consider the following set of unobservable transitions:

$$T \uparrow (t_f) = \{t \in T_{uo} \mid \exists\, p \in t_f^\bullet \text{ such that a path from } t \text{ to } p \text{ exists in } N_{uo}\} \cup \{t_f\}.$$

The *post-net* of t_f, denoted as $\mathcal{N} \uparrow (t_f)$, is equal to the subnet $\hat{N} \prec_{T\uparrow(t_f)} N_{uo}$, if $({}^\bullet t_f)^\bullet \neq \{t_f\}$ in $\hat{N}$. If $({}^\bullet t_f)^\bullet = \{t_f\}$ in $\hat{N}$, then $\mathcal{N} \uparrow (t_f)$ is obtained from $\hat{N}$ removing the places in ${}^\bullet t_f$. $\Diamond$

Action	$\boldsymbol{\mu}$	$\min_{\boldsymbol{\epsilon}\in\boldsymbol{\Sigma}(N,\boldsymbol{\mu})}\boldsymbol{\epsilon}(t_3)$	$\max_{\boldsymbol{\epsilon}\in\boldsymbol{\Sigma}(N,\boldsymbol{\mu})}\boldsymbol{\epsilon}(t_3)$	$\min_{\boldsymbol{\epsilon}\in\boldsymbol{\Sigma}(N,\boldsymbol{\mu})}\boldsymbol{\epsilon}(t_8)$	$\max_{\boldsymbol{\epsilon}\in\boldsymbol{\Sigma}(N,\boldsymbol{\mu})}\boldsymbol{\epsilon}(t_8)$	Comment
Initialization	$[4\ 0\ 0\ 0\ 0\ 0\ 4\ 0\ 0\ 0\ 0\ 0\ 0]^T$	not computed	0	not computed	0	Under the initial marking both t_3 and t_8 cannot fire
t_1 fires	$[4\ 1\ 0\ 0\ 0\ 0\ 4\ 0\ 0\ 0\ 0\ 0\ 0]^T$	not computed	1	not computed	0	t_3 may have fired once, while t_8 cannot fire
t_1 fires	$[4\ 2\ 0\ 0\ 0\ 0\ 4\ 0\ 0\ 0\ 0\ 0\ 0]^T$	not computed	2	not computed	0	t_3 may have fired up to 2 times, while t_8 cannot fire
t_2 fires	$[3\ 1\ 1\ 0\ 0\ 0\ 4\ 0\ 0\ 0\ 0\ 0\ 0]^T$	not computed	1	not computed	0	t_3 may have fired once, while t_8 cannot fire
t_6 fires	$[4\ 1\ 1\ 0\ -1\ 0\ 4\ 0\ 0\ 0\ 0\ 0\ 0]^T$	0	1	0	0	t_3 may have fired once, while t_8 cannot fire
t_7 fires	$[4\ 1\ 1\ 0\ -1\ -1\ 3\ 1\ 0\ 0\ 0\ 0\ 0]^T$	0	1	0	1	Both t_3 and t_8 may have fired once
t_1 fires	$[4\ 2\ 1\ 0\ -1\ -1\ 3\ 1\ 0\ 0\ 0\ 0\ 0]^T$	0	2	0	1	t_3 may have fired up to 2 times, while t_8 may have fired once
t_2 fires	$[3\ 1\ 2\ 0\ -1\ -1\ 3\ 1\ 0\ 0\ 0\ 0\ 0]^T$	0	1	0	1	Both t_3 and t_8 may have fired once
t_6 fires	$[4\ 1\ 2\ 0\ -2\ -1\ 3\ 1\ 0\ 0\ 0\ 0\ 0]^T$	0	1	0	1	Both t_3 and t_8 may have fired once
t_7 fires	$[4\ 1\ 2\ 0\ -2\ -2\ 2\ 2\ 0\ 0\ 0\ 0\ 0]^T$	0	1	0	2	t_3 may have fired once, while t_8 may have fired up to 2 times
t_6 fires	$[5\ 1\ 2\ 0\ -3\ -2\ 2\ 2\ 0\ 0\ 0\ 0\ 0]^T$	1	1	0	2	t_3 has fired once, while t_8 may have fired up to 2 times
Update $\boldsymbol{\mu}$ (Step **2.1.1.2**)	$[4\ 0\ 2\ 1\ -3\ -2\ 2\ 2\ 0\ 0\ 0\ 0\ 0]^T$	0	0	0	2	t_3 cannot fire, while t_8 may have fired up to 2 times
t_9 fires	$[4\ 0\ 2\ 1\ -3\ -2\ 2\ 1\ 1\ 0\ 0\ 0\ 0]^T$	0	0	0	1	t_3 cannot fire, while t_8 may have fired once
t_1 fires	$[4\ 1\ 2\ 1\ -3\ -2\ 2\ 1\ 1\ 0\ 0\ 0\ 0]^T$	0	1	0	1	Both t_3 and t_8 may have fired once
t_2 fires	$[3\ 0\ 3\ 1\ -3\ -2\ 2\ 1\ 1\ 0\ 0\ 0\ 0]^T$	0	0	0	1	t_3 cannot fire, while t_8 may have fired once
t_{14} fires	$[3\ 0\ 3\ 1\ -3\ -2\ 3\ 1\ 1\ 0\ 0\ 0\ -1]^T$	0	0	1	1	t_8 has fired once, while t_3 cannot fire
Update $\boldsymbol{\mu}$ (Step **2.1.1.2**)	$[3\ 0\ 3\ 1\ -3\ -2\ 3\ 0\ 1\ 0\ 0\ 0\ 0]^T$	0	0	0	0	Both t_3 and t_8 cannot fire
t_7 fires	$[3\ 0\ 3\ 1\ -3\ -3\ 2\ 1\ 1\ 0\ 0\ 0\ 0]^T$	0	0	0	1	t_3 cannot fire, while t_8 may have fired once
t_{10} fires	$[3\ 0\ 3\ 1\ -3\ -3\ 2\ 0\ 1\ 1\ 0\ 0\ 0]^T$	0	0	0	0	Both t_3 and t_8 cannot fire
t_1 fires	$[3\ 1\ 3\ 1\ -3\ -3\ 2\ 0\ 1\ 1\ 0\ 0\ 0]^T$	0	1	0	0	t_3 may have fired once, while t_8 cannot fire
t_2 fires	$[2\ 0\ 4\ 1\ -3\ -3\ 2\ 0\ 1\ 1\ 0\ 0\ 0]^T$	0	0	0	0	Both t_3 and t_8 cannot fire
t_7 fires	$[2\ 0\ 4\ 1\ -3\ -4\ 1\ 1\ 1\ 1\ 0\ 0\ 0]^T$	0	0	0	1	t_3 cannot fire, while t_8 may have fired once
t_9 fires	$[2\ 0\ 4\ 1\ -3\ -4\ 1\ 0\ 2\ 1\ 0\ 0\ 0]^T$	0	0	0	0	Both t_3 and t_8 cannot fire
t_{14} fires	$[2\ 0\ 4\ 1\ -3\ -4\ 2\ 0\ 2\ 1\ 0\ 0\ -1]^T$	0	0	0	0	Both t_3 and t_8 cannot fire
t_6 fires	$[3\ 0\ 4\ 1\ -4\ -4\ 2\ 0\ 2\ 1\ 0\ 0\ -1]^T$	0	0	0	0	Both t_3 and t_8 cannot fire
t_6 fires	$[4\ 0\ 4\ 1\ -5\ -4\ 2\ 0\ 2\ 1\ 0\ 0\ -1]^T$	0	0	0	0	Both t_3 and t_8 cannot fire, and $\boldsymbol{C}_{uo}\boldsymbol{\epsilon}_{\mid T_{uo}} \geq -\boldsymbol{\mu}_{\mid P_{uo}}$ admits only one solution $\boldsymbol{\epsilon}^*$ s.t. $\boldsymbol{\epsilon}^*_{\mid T_o} = \mathbf{0}$
Update $\boldsymbol{\mu}$ (Step **4**)	$[4\ 0\ 0\ 0\ 0\ 0\ 2\ 0\ 0\ 0\ 0\ 0\ 2]^T$	not computed	0	not computed	0	Both t_3 and t_8 cannot fire

Fig. 4. Execution of the fault detection algorithm on the net of Fig. 3(a).

Require: $t_f \in T_f$, T_{uo}, P_{uo}, $\mathbf{Pre}_{uo}$, $\mathbf{Post}_{uo}$

1 $T_{old} = t_f$, $T_{new} = \emptyset$, $\widetilde{P} = {}^\bullet t_f$, $P_{c_{old}} = P_{c_{new}} = \emptyset$ (* Initialization *)
2 $\widetilde{T} = \left\{t \in T_{uo} \mid \exists\ p \in \widetilde{P} \text{ s.t. } \mathbf{Pre}_{uo}(p,t) \neq 0 \text{ or } \mathbf{Post}_{uo}(p,t) \neq 0\right\} \setminus \left\{t_f\right\}$
(* $\widetilde{T}$ is the set of transitions in T_{uo} which are linked to at least one place in $\widetilde{P}$, except t_f *)
5 $T_{new} = T_{old} \cup \widetilde{T}$ (* update T_{new} *)
6 $T_{old} = T_{new}$ (* update T_{old} *)
7 **if** $\widetilde{T}$ is empty **then**
$T{\downarrow}(t_f) = T_{new}$
8 **else**
8.1 $P_{c_{new}} = P_{c_{old}} \cup \widetilde{P}$ (* update the set of places already checked *)
8.2 $P_{c_{old}} = P_{c_{new}}$ (* update $P_{c_{old}}$ *)
8.3 $\widetilde{P} = \left\{p \in P_{uo} \mid \exists\ t \in \widetilde{T} \text{ s.t. } \mathbf{Pre}_{uo}(p,t) \neq 0 \text{ or } \mathbf{Post}_{uo}(p,t) \neq 0\right\} \setminus P_{c_{new}}$
(* $\widetilde{P}$ is the set of places in P_{uo} which are linked to at least one transition in $\widetilde{T}$, except the places already checked *)
8.4 **go to** Step **2**

Fig. 5. Algorithm to compute the transition set $T \downarrow (t_f)$.

The algorithm to compute the set $T \downarrow (t_f)$ is reported in Fig. 5. The set $T \uparrow (t_f)$ can be obtained by the same algorithm defining $\widetilde{P} = t_f^\bullet$.

Let us denote $\mathcal{N} \downarrow (t_f) = (P \downarrow (t_f), T \downarrow (t_f), \mathbf{Pre} \downarrow (t_f), \mathbf{Post} \downarrow (t_f))$, and $\mathcal{N} \uparrow (t_f) = (P \uparrow (t_f), T \uparrow (t_f), \mathbf{Pre} \uparrow (t_f), \mathbf{Post} \uparrow (t_f))$.

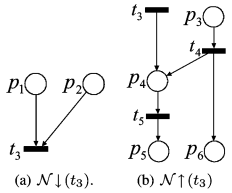


Fig. 6. Pre and post-net of t_3.

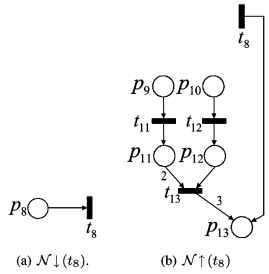


Fig. 7. Pre and post-net of t_8.

Example 3: Consider the $N_{uo} \prec_{T_{uo}} N$ subnet of the manufacturing system shown in Fig. 3(b). Applying the definitions given in this section, it follows that:

- $T\downarrow(t_3) = \{t_3\}$
- $T\uparrow(t_3) = \{t_3, t_4, t_5\}$
- $T\downarrow(t_8) = \{t_8\}$
- $T\uparrow(t_8) = \{t_8, t_{11}, t_{12}, t_{13}\}$

The *pre* and *post-nets* for t_3 and t_8 are shown in Figs. 6 and 7, respectively. In particular, since in the $\hat{N} \prec_{T\downarrow(t_3)} N_{uo}$ subnet one has ${}^\bullet(t_f^\bullet) = \{t_3\}$, the place p_4 is removed from $\hat{N}$, and $P\downarrow(t_3) = \{p_1, p_2\}$. $\Diamond$

The following theorems show that it is possible to solve ILPs on the *pre* and *post-nets* of a given fault, instead of solve them on the whole $N_{uo} \prec_{T_{uo}} N$ subnet.

Theorem 7: Consider a net N, with $T = T_o \cup T_{uo}$. Let $\boldsymbol{\mu}$ be a g-marking, and $t_f \in T_f \subseteq T_{uo}$ a fault transition, then

$$\min_{\boldsymbol{\epsilon}\in\boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) = \min_{\boldsymbol{\epsilon}'\in\boldsymbol{\Sigma}(\mathcal{N}\uparrow(t_f),\boldsymbol{\mu}_{|P\uparrow(t_f)})} \boldsymbol{\epsilon}'(t_f).$$

Proof: Let us denote with

$$\bar{\boldsymbol{\epsilon}} = \arg \min_{\boldsymbol{\epsilon}\in\boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f)$$
$$\bar{\bar{\boldsymbol{\epsilon}}} = \arg \min_{\boldsymbol{\epsilon}'\in\boldsymbol{\Sigma}(\mathcal{N}\uparrow(t_f),\boldsymbol{\mu}_{|P\uparrow(t_f)})} \boldsymbol{\epsilon}'(t_f)$$

with $\bar{\boldsymbol{\epsilon}}(t_f) \geq 0$ and $\bar{\bar{\boldsymbol{\epsilon}}}(t_f) \geq 0$. Since $\mathcal{N}\uparrow(t_f)$ is a subnet of N, it follows that some of the negative markings explained by the firing of t_f in $\mathcal{N}\uparrow(t_f)$, may be explained by the firing of a transition which belongs to N, but it is not included in $\mathcal{N}\uparrow(t_f)$. Then, by definition of the post-net, it holds

$$\bar{\bar{\boldsymbol{\epsilon}}}(t_f) \geq \bar{\boldsymbol{\epsilon}}(t_f).$$

Suppose, *ad absurdum*, that

$$\bar{\bar{\boldsymbol{\epsilon}}}(t_f) > \bar{\boldsymbol{\epsilon}}(t_f).$$

It follows that there exists a transition $t \in T_{uo} \setminus T\uparrow(t_f)$, such that $\bar{\boldsymbol{\epsilon}}(t) > \bar{\bar{\boldsymbol{\epsilon}}}(t)$, and whose firing adds at least one token to at least one place p in $t_f^\bullet$. Hence there is a path from t to at least one place p in $t_f^\bullet$, which implies that $t \in T\uparrow(t_f)$, and $\bar{\boldsymbol{\epsilon}}(t) = \bar{\bar{\boldsymbol{\epsilon}}}(t)$. It turns out that it is not possible to find any $t \in T_{uo} \setminus T\uparrow(t_f)$ whose firing in $\bar{\boldsymbol{\epsilon}}$ can replace the firing of t_f in $\bar{\bar{\boldsymbol{\epsilon}}}$, thus $\bar{\bar{\boldsymbol{\epsilon}}}(t_f) = \bar{\boldsymbol{\epsilon}}(t_f)$. ■

Theorem 8: Consider a net N, with $T = T_o \cup T_{uo}$. Let $\boldsymbol{\mu}$ be a g-marking, and $t_f \in T_f \subseteq T_{uo}$ a fault transition, then

$$\max_{\boldsymbol{\epsilon}\in\boldsymbol{\Sigma}(N,\boldsymbol{\mu})} \boldsymbol{\epsilon}(t_f) = \max_{\boldsymbol{\epsilon}''\in\boldsymbol{\Sigma}(\mathcal{N}\downarrow(t_f),\boldsymbol{\mu}_{|P\downarrow(t_f)})} \boldsymbol{\epsilon}''(t_f).$$

Proof: This proof can be done using similar arguments as in the proof of Theorem 7. ■

B. Fault Detection Algorithm With Pre and Post-Nets

It is possible to exploit the results presented in the previous subsection, when solving the ILPs in the proposed fault detection algorithm.

Proposition 4 (Fault Detection Using Pre and Post-Nets): Consider a net N, with $T = T_o \cup T_{uo}$. Let $t_f \in T_f \subseteq T_{uo}$, and $\boldsymbol{\mu}$ a g-marking. In order to perform the detection of the fault t_f, the following three conditions have to be checked:

1c) $\boldsymbol{\mu}_{|P\uparrow(t_f)} \not\geq \mathbf{0}$ and $\min_{\boldsymbol{\epsilon}'\in\boldsymbol{\Sigma}(\mathcal{N}\uparrow(t_f),\boldsymbol{\mu}_{|P\uparrow(t_f)})} \boldsymbol{\epsilon}'(t_f) \neq 0 \iff t_f$ has occurred

2c) $\max_{\boldsymbol{\epsilon}''\in\boldsymbol{\Sigma}(\mathcal{N}\downarrow(t_f),\boldsymbol{\mu}_{|P\downarrow(t_f)})} \boldsymbol{\epsilon}''(t_f) = 0 \iff t_f$ has not occurred

3c) $\max_{\boldsymbol{\epsilon}''\in\boldsymbol{\Sigma}(\mathcal{N}\downarrow(t_f),\boldsymbol{\mu}_{|P\downarrow(t_f)})} \boldsymbol{\epsilon}''(t_f) \neq 0 \iff t_f$ may have occurred

■

Example 4: Consider the *pre* and *post-nets* of Figs. 6 and 7. As it has been discussed at the beginning of this section, by solving the ILPs on these four nets, it is possible to improve the efficiency of the fault detection algorithm, at a price of a little increase in memory requirement, that is the storing of four additional nets.

Note that, although the whole N_{uo} subnet does not have any particular structure, some of the *pre* and *post-nets* may exhibit particular properties, which permit to attain a further improvement in terms of computational effort. In particular, for the considered manufacturing system, it can be easily shown that

$$\max_{\boldsymbol{\epsilon}''\in\boldsymbol{\Sigma}(\mathcal{N}\downarrow(t_8),\boldsymbol{\mu}_{|P\downarrow(t_8)})} \boldsymbol{\epsilon}''(t_8) = \max(0, \boldsymbol{\mu}_{|p_8}).$$

Moreover $\mathcal{N}\downarrow(t_3)$ shown in Fig. 6(a) is TS1 [17], while $\mathcal{N}\uparrow(t_3)$ and $\mathcal{N}\uparrow(t_8)$, shown in Figs. 6(b) and 7(b) respectively, are TS2, thus

$$\min_{\boldsymbol{\epsilon}'\in\boldsymbol{\Sigma}(\mathcal{N}\uparrow(t_3),\boldsymbol{\mu}_{|P\uparrow(t_3)})} \boldsymbol{\epsilon}'(t_3) = \max(0, -\boldsymbol{\mu}_{|p_3} - \boldsymbol{\mu}_{|p_4} - \boldsymbol{\mu}_{|p_5})$$
$$\max_{\boldsymbol{\epsilon}''\in\boldsymbol{\Sigma}(\mathcal{N}\downarrow(t_3),\boldsymbol{\mu}_{|P\downarrow(t_3)})} \boldsymbol{\epsilon}''(t_3) = \max(0, \min(\boldsymbol{\mu}_{|p_1}, \boldsymbol{\mu}_{|p_2}))$$
$$\min_{\boldsymbol{\epsilon}'\in\boldsymbol{\Sigma}(\mathcal{N}\uparrow(t_8),\boldsymbol{\mu}_{|P\uparrow(t_8)})} \boldsymbol{\epsilon}'(t_8) =$$
$$= \max\left(0, -\boldsymbol{\mu}_{p_{13}} - 3\cdot\max\left(0, \min\left(\boldsymbol{\mu}_{p_{10}} + \boldsymbol{\mu}_{p_{12}}, \left\lfloor \frac{\boldsymbol{\mu}_{p_9} + \boldsymbol{\mu}_{p_{11}}}{2} \right\rfloor\right)\right)\right).$$

It follows that, when applying the proposed algorithm to perform the fault detection for the manufacturing system consid-

ered in Section V-A, there is no need to solve any ILP online, while only the four expressions above must be stored and evaluated each time the firing of a new transition is observed. ◊

VI. Conclusion

This paper proposes a novel approach to fault diagnosis for DES, which requires the online computation of the set of possible fault events explaining the last observed event. In order to achieve this result, *g-markings* are introduced in this paper. G-markings are net markings that may have negative components and whose estimation is always unique. The online computation consists of solving programming problems formulated on net structure and based on g-markings. In many cases of practical interest, such programming problems have closed-form solutions or LP forms, even if in general they are ILPs. However, it has been shown that such ILPs can be rewritten into equivalent ones, which are formulated on proper subnets that influence the occurrence of the observed event. These subnets are computed offline and permit to reduce the computational effort to solve the programming problems at the price of a small memory increase.

References

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

[2] S. H. Zad, R. H. Kwong, and W. M. Wonham, "Fault diagnosis in discrete-event systems: Framework and model reduction," *IEEE Trans. Automat. Control*, vol. 48, no. 7, pp. 1199–1212, Jul. 2003.

[3] M. Sampath, R. Sengupta, S. Lafortune, and K. Sinnamohideen, "Failure diagnosis using discrete event models," *IEEE Trans. Control Syst. Technol.*, vol. 4, no. 2, pp. 105–124, Mar. 1996.

[4] M. Sampath, S. Lafortune, and D. Teneketzis, "Active diagnosis of discrete event systems," *IEEE Trans. Automat. Control*, vol. 43, no. 7, pp. 908–929, Jul. 1998.

[5] R. Debouk, S. Lafortune, and D. Teneketzis, "Coordinated decentralized protocols for failure diagnosis of discrete event systems," *Discrete Event Dyn. Syst.: Theory Appl.*, vol. 10, no. 1, pp. 33–86, Jan. 2000.

[6] J. Lunze and J. Schröder, "State observation and diagnosis of discrete-event systems described by stochastic automata," *Discrete Event Dyn. Syst.*, vol. 11, no. 4, pp. 319–369, Oct. 2001.

[7] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella, "Diagnosis of a class of distributed discrete-event systems," *IEEE Trans. Syst., Man, Cybern. A*, vol. 30, no. 6, pp. 731–752, Nov. 2000.

[8] Y. Wu and C. N. Hadjicostis, "Algebraic approaches for fault identification in discrete-event systems," *IEEE Trans. Automat. Control*, vol. 50, no. 12, pp. 2048–2053, Dec. 2005.

[9] S. Genc and S. Lafortune, "Distributed diagnosis of discrete-event systems using Petri nets," in *Proc. Int. Conf. Appl. Theory Petri Nets*, Eindhoven, The Netherlands, Jun. 2003, vol. 2679, pp. 316–336.

[10] S. Genc and S. Lafortune, "Distributed diagnosis of place-bordered Petri nets," *IEEE Trans. Automat. Sci. Eng.*, vol. 4, no. 2, pp. 206–219, Apr. 2007.

[11] A. Giua and C. Seatzu, "Fault detection for discrete event systems using Petri net with unobservable transitions," in *Proc. 44th IEEE Conf. Decision Control*, Seville, Spain, Dec. 2005, pp. 6323–6328.

[12] R. K. Boel and G. Jiroveanu, "Contextual distributed diagnosis for very large systems," in *Proc. 16th Int. Symp. Math. Theory Networks Syst.*, Leuven, Belgium, Jul. 2004, [CD ROM].

[13] A. Benveniste, E. Fabre, S. Haar, and C. Jard, "Diagnosis of asynchronous discrete-event systems: A net unfolding approach," *IEEE Trans. Automat. Control*, vol. 48, no. 5, pp. 714–727, May 2003.

[14] A. Trevino, E. Ruiz-Beltran, I. Rivera-Rangel, and E. Lopez-Mellado, "Online fault diagnosis of discrete event systems. A Petri net-based approach," *IEEE Trans. Automat. Science Eng.*, vol. 4, no. 1, pp. 31–39, Jan. 2007.

[15] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.

[16] P. Ramadge and W. Wonham, "The control of vector discrete-event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, Jan. 1989.

[17] Y. Li and W. M. Wonham, "Control of vector discrete-event systems II—Controller synthesis," *IEEE Trans. Automat. Control*, vol. 39, no. 3, pp. 512–531, Mar. 1994.

[18] A. Schrijver, *Theory of Linear and Integer Programming*. New York: Wiley, 1986.

[19] F. Basile, P. Chiacchio, and C. Carbone, "Feedback control logic for backward conflict free choice nets," *IEEE Trans. Automat. Control*, vol. 52, no. 3, pp. 387–400, Mar. 2007.

[20] D. Corona, A. Giua, and C. Seatzu, "Marking estimation of Petri nets with silent transitions," in *Proc. 43rd IEEE Conf. Decision Control*, Atlantis, Bahamas, Dec. 2004, vol. 1, pp. 966–971.

[21] L. Holloway and B. Krogh, "Synthesis of feedback control logic for a class of controlled Petri nets," *IEEE Trans. Automat. Control*, vol. 35, no. 5, pp. 514–523, May 1990.

Francesco Basile (S'95–M'03) was born in Naples, Italy, in 1971. He received the Laurea degree in electronic engineering and the Ph.D. degree in electronic and computer engineering from the University of Naples, Naples, Italy, in 1995 and 1999, respectively.

In 1999, he was a Visiting Researcher for six months at the Departamento de Ingenieria Informatica y Systems, University of Saragoza, Spain. He is currently Assistant Professor of automatic control at the Dipartimento di Ingegneria dell'Informazione e Ingegneria Elettrica, University of Salerno, Salerno, Italy. He is a member of the editorial board of the *International Journal of Robotics and Automation*. His current research interest are: modelling and control of discrete event systems, automated manufacturing, and robotics.

Dr. Basile is a member of the IEEE Control System Society Conference Editorial Board.

Pasquale Chiacchio was born in Naples, Italy, on September 7, 1963. He received the Laurea and the Research Doctorate degrees in electronics engineering from the University of Naples, in 1987 and 1992, respectively.

He is currently a Full Professor of automatic control in the Engineering Faculty of the University of Salerno, Salerno, Italy. He has published 23 international journal papers, 59 international conference papers, three textbooks, and is co-editor of the book *Complex Robotic Systems* (New York: Springer-Verlag, 1998). His research interests include manipulator inverse kinematics techniques, redundant manipulator control, cooperative robot systems, and analysis and control of discret event systems.

Gianmaria De Tommasi (M'06) received the laurea degree (with highest honors) in electronic engineering and the Ph.D. degree in computer and automatic engineering from the University of Naples Federico II, Naples, Italy, in 2001 and 2005, respectively.

Since 2002, he has been with the Department of Computer and Systems Engineering, University of Naples Federico II, where he is currently an Assistant Professor. He has been a Visiting Researcher at the JET tokamak, where he has participated in the implementation of the eXtreme Shape Controller project. His current research interests include control of nuclear fusion devices, fault detection for discrete event systems, and stability of hybrid systems.