Diagnosability under Weak Fairness

Vasileios Germanos[1] Stefan Haar[2] Victor Khomenko[1] Stefan Schwoon[2]

[1]School of Computing Science, Newcastle University, Newcastle upon Tyne, UK
e-mail: {V.Germanos,Victor.Khomenko}@ncl.ac.uk

[2]INRIA & LSV (ENS Cachan & CNRS), France
e-mail: Stefan.{Haar,Schwoon}@inria.fr

Abstract

In partially observed Petri nets, diagnosis is the task of detecting whether or not the given sequence of observed labels indicates that some unobservable fault has occurred. Diagnosability is an associated property of the Petri net, stating that in any possible execution an occurrence of a fault can eventually be diagnosed.

In this paper we consider diagnosability under the weak fairness (WF) assumption, which intuitively states that no transition from a given set can stay enabled forever — it must eventually either fire or be disabled. We show that a previous approach to WF-diagnosability in the literature has a major flaw, and present a corrected notion. Moreover, we present an efficient method for verifying WF-diagnosability based on a reduction to LTL-X model checking. An important advantage of this method is that the LTL-X formula is fixed — in particular, the WF assumption does not have to be expressed as a part of it (which would make the formula length proportional to the size of the specification), but rather the ability of existing model checkers to handle weak fairness directly is exploited.

Keywords: *Diagnosability, weak fairness, model checking, LTL-X, formal verification, Petri nets.*

1. Introduction

The *diagnosability* of systems has recently drawn the attention of many researchers in both artificial intelligence and control theory communities. *Diagnosis* is the process of explaining abnormal behaviours of a physical system, and diagnosability is an important property that determines the possibility of detecting faults given a set of observations. If a system is diagnosable, it is always possible to determine whether the fault has occurred by observing the system's behaviour for sufficiently long time, and then the diagnosis can find possible explanations for the given sequence of observations. Otherwise there are scenarios in which it is impossible to tell whether the fault has occurred or not, no matter for how long the system is observed. Non-diagnosability usually indicates that the system should be augmented with additional sensors monitoring it.

The seminal work [8] introduced a formal language framework for diagnosis and analysis of diagnosability properties of discrete event systems represented by finite automata. The proposed method for diagnosability verification was based on the construction of a *diagnoser* — an automaton with only observable transitions that allows one to estimate states of the system by observing its traces. Improvements based on the *twin plant* method have been introduced in [3, 9], where the basic idea was to build a *verifier* by constructing the synchronous product of the system with itself on observable transitions. The verifier compares every pair of executions in the system that have the same projection on the observable transitions. If the original system is given as a labelled Petri net, then the verifier can be constructed directly, by synchronising the original net with its replica at the Petri net level, and the problem reduces to model checking of a fixed LTL-X [4, 7] property of the verifier [5].

Recent work [2] presented a diagnosis method that encompasses *weak fairness*. There, concurrent systems are modelled by partially observable safe Petri nets, and diagnosis is carried out under the assumption that all executions of the Petri net are weakly fair, that is, the only infinite executions admitted are those in which any transition *enabled* at some stage will be *disabled* at some later stage, i.e. either it will actually fire later in that execution, or else some conflicting transition will fire. Under this assumption, a given finite observation diagnoses a fault if no finite execution yielding this observation can be extended to a weakly fair fault-free exe-

cution. The work in [2] gave a procedure for deciding this diagnosis problem. It remained open for which systems this procedure reliably diagnoses faults, i.e. how to determine whether a system is *diagnosable* under the weak fairness assumption. In this paper, we address this problem.

Note that a first definition of diagnosability under weak fairness was proposed in [1]. However, this definition is incompatible with the notion of diagnosis in [2] and contains a major flaw, as we shall point out below.

We make the following contributions in this paper:

- We develop a notion of weakly fair (WF) diagnosability, which corrects and supersedes the one from [1].
- We characterise executions that *witness* violations of WF-diagnosability.
- We further investigate the special case where fault transitions are not WF, i.e. a fault is a *possible* outcome in the system but not one that is *required* to happen. (Our examples in Sect. 5 suggest that this is a reasonable assumption in practice.) Under this assumption, the notion of a witness can be significantly simplified.
- We develop a method for verifying diagnosability in this case, and evaluate it experimentally.

The paper is organised as follows: Sect. 2 discusses existing notions of diagnosability and explains why they are problematic for concurrent systems. Sect. 3 develops our notion of WF-diagnosability and witnesses. Sect. 4 presents the construction of the verifier, which is evaluated in Sect. 5. We conclude in Sect. 6.

2. Petri nets and diagnosability

In this paper, we consider concurrent systems modelled as Petri nets. We use this section to explain why the standard notion of diagnosability, as well as the notion of WF-diagnosability developed in [1], are problematic, which motivates our new definition, to be presented later.

Throughout the paper we assume that the system is modelled as a labelled Petri net (LPN) $\mathcal{N}$, where each transition is labelled with the performed action. The actions are partitioned into *observable* and *silent,* i.e. there is a labelling function ℓ mapping the LPN's transitions to $O \cup \{\varepsilon\}$, where O is an alphabet of *observable* actions and $\varepsilon \notin O$ is the empty word denoting the *silent* action. (Intuitively, observable actions correspond to controller commands and sensor readings, while the silent action models some internal activity that is not recorded by sensors.) This labelling function ℓ can be naturally extended to finite and infinite executions of the LPN, projecting them to words in O^* or O^ω. We assume that the LPN is free from deadlocks and divergencies, i.e. every execution of the LPN can be extended to an infinite one, and every infinite execution of the LPN has infinitely many observable transitions. Some of the transitions are designated as *faults;* w.l.o.g., we assume that none of them is observable. An example in Fig. 1 shows an LPN with the observable transitions t_3, t_4 and t_5 with $\ell(t_3) = a$, $\ell(t_4) = b$ and $\ell(t_5) = tick$ (the other transitions are unobservable). Note that we draw faults as black boxes, and the observable transitions are shaded.

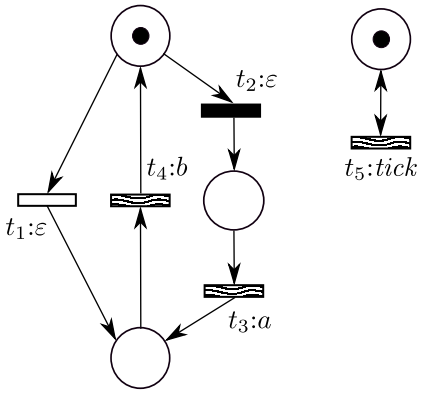

Figure 1. This LPN without t_5 would be diagnosable, but t_5 makes it undiagnosable. Making t_3 WF makes the LPN diagnosable.

2.1. Standard diagnosability

Given a finite execution σ of the LPN, the observer sees the outputs of the system $\ell(\sigma) \in O^*$, and needs to conclude whether some fault transition t has definitely occurred in σ. In a diagnosable system, once a fault has occurred, the observer is able to *eventually* detect this. That is, provided that the suffix of σ after the first occurrence of a fault in it is sufficiently long, the observer should be able to conclude that each execution with the same projection $\ell(\sigma)$ contains a fault, i.e. a fault has either already occurred or will definitely occur in the future. Let us first recall the standard definition of diagnosability:[1]

Definition 1 (Diagnosability)**.** *An LPN is diagnosable iff for all its infinite traces σ and ρ such that $\ell(\sigma) = \ell(\rho)$, σ contains a fault iff ρ contains a fault.*

[1] This definition is taken from [5]. It is subtly different from the original definition in [8], but equivalent for finite state systems, and simpler to use in practice. (An LPN has finitely many reachable markings iff it is bounded.)

In other words, a non-diagnosable LPN has two infinite executions having the same projection onto the observable actions and such that one of them contains a fault and the other does not; such a pair of traces constitutes a *witness* of diagnosability violation.

For example, the LPN in Fig. 1 is not diagnosable. Indeed, the diagnoser can only conclude that the fault has occurred after observing a. However, the infinite execution $t_2t_5^\omega$ contains a fault but never fires t_3. Nevertheless, if t_5 is removed, the LPN becomes diagnosable.

2.2. Weak fairness

The example from Fig. 1 exhibits a pathological property of this notion of diagnosability: a diagnosable system ceases to be such simply because some unrelated concurrent activity is added to the specification. In practice, it is often reasonable to assume that the system is keen to fire its enabled transitions, and *cannot perpetually ignore an enabled transition.* In other words, one can consider the LPN in Fig. 1 diagnosable, by declaring the infinite execution $t_2t_5^\omega$ impossible.

To capture this idea formally, the notion of *weak fairness* is helpful [10]. Suppose the designer wants to disallow some of the transitions to be perpetually ignored when enabled. We call such transitions *weakly fair* (WF). An infinite execution σ of the LPN is called *weakly fair* (WF) if for each WF transition t, if t is enabled after some prefix of σ then the rest of σ contains some transition in $({}^\bullet t)^\bullet$, see Fig. 2. All finite executions are regarded as WF. We now can use the set of WF executions as the semantics of the LPN, i.e. other executions are considered impossible. Coming back to the example in Fig. 1, if t_3 is WF then the execution $t_2t_5^\omega$ is not WF and thus impossible, and so the LPN becomes diagnosable.

It is tempting to derive the definition of WF-diagnosability simply by taking Def. 1 and restricting to WF executions. In fact, such an approach was taken in [1], where an LPN $\mathcal{N}$ was said to be WF-diagnosable iff for all its infinite WF executions σ and ρ such that $\ell(\sigma) = \ell(\rho)$, σ contains a fault iff ρ contains a fault.

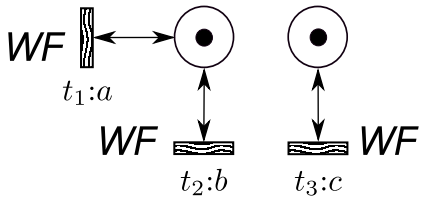

Figure 2. (i) The execution $(t_1t_2t_3)^\omega$ is WF as no enabled transition is perpetually ignored by it. (ii) The execution $(t_1t_2)^\omega$ is not WF as t_3 is enabled but all the transitions in $({}^\bullet t_3)^\bullet = \{t_3\}$ are perpetually ignored. (iii) The execution $(t_1t_3)^\omega$ is WF: even though t_2 is perpetually ignored, $t_1 \in ({}^\bullet t_2)^\bullet = \{t_1, t_2\}$ is fired.

Unfortunately, this definition contains a major flaw, demonstrated by the example in Fig. 3. This LPN would be said to be diagnosable, while it is not possible for the observer to detect a fault in finite time, as one would have to observe the infinite trace a^ω to positively conclude that the fault has occurred.

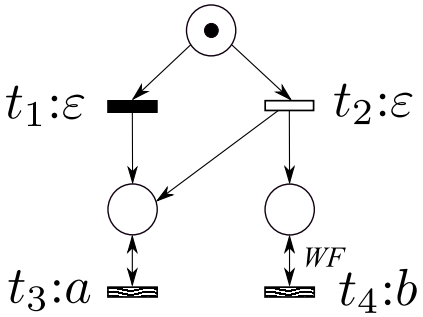

Figure 3. This LPN is WF-diagnosable according to the definition from [1], but not according to the corrected definition (Def. 2 and Lemma 1). Note that the observer cannot detect the fault in finite time.

3. Weakly fair diagnosability

To fix the problems exhibited in Sect. 2, we present a corrected definition of WF-diagnosability, where the possibility of detecting a fault in finite time is imposed. Intuitively, it states that each infinite WF execution containing a fault must have a finite prefix after which it is possible to conclude unambiguously that the fault has either occurred or will inevitably occur in future. Below we denote by '$<$' the prefix relation on sequences.

Definition 2 (WF-diagnosability)**.** *An LPN is WF-diagnosable iff each infinite WF execution σ containing a fault has a finite prefix $\hat{\sigma}$ such that every infinite WF execution ρ with $\ell(\hat{\sigma}) < \ell(\rho)$ contains a fault.*

The LPN in Fig. 3 is not WF-diagnosable according to Def 2, as for each finite prefix (say, $t_1t_3^n$ for some $n \in \mathbb{N}$) of the infinite WF execution $t_1t_3^\omega$ containing a fault, there is a finite execution $(t_2t_3^n)$ with the same projection to observable actions, that can be extended to an infinite WF execution without a fault (e.g. $t_2t_3^n(t_3t_4)^\omega$).

In this example one can also identify a fault-free *infinite* execution $t_2t_3^\omega$ that is in itself not WF, but each of

its finite prefixes can be extended to a fault-free WF execution. As we shall see, such an execution can always be found in a *bounded* LPN that is not WF-diagnosable.

Definition 3 (Witness for a bounded LPN)**.** *Let $\mathcal{N}$ be a bounded LPN. A pair of infinite executions (σ,ρ) with $\ell(\sigma)=\ell(\rho)$ is called* witness *(of WF-diagnosability violation) if σ is WF and contains a fault, and every prefix of ρ can be extended to a fault-free WF execution.*

Lemma 1 (WF-diagnosability of a bounded LPN)**.** *A bounded LPN $\mathcal{N}$ is WF-diagnosable iff no witness of its WF-diagnosability violation satisfying the conditions of Def. 3 exists.*

Proof. If a witness satisfying Def. 3 exists then the condition of Def. 2 is violated, as for any prefix of σ one can choose a prefix of ρ with the same projection, which can be extended to a fault-free WF execution, i.e. the $\mathcal{N}$ is not WF-diagnosable.

In the reverse direction: Suppose $\mathcal{N}$ is not WF-diagnosable. Then, according to Def. 2, there exists an infinite, WF, faulty execution σ such that for every finite prefix $\hat{\sigma}<\sigma$ there exists some infinite, WF, fault-free execution ρ with $\ell(\hat{\sigma})<\ell(\rho)$. From σ, we shall construct a pair of executions (σ',ρ') constituting a witness according to Def. 3.

Let K be the number of states (i.e. reachable markings) of $\mathcal{N}$. Let $m(\sigma,i)$ denote the marking generated by the i-th observable transition in σ; since $\mathcal{N}$ has no divergencies, it is well-defined for all $i\geq 1$. Moreover, let $s(\sigma,i,j)$ denote the interval of σ starting after i-th observable transition and ending at j-th observable transition, for all $0<i<j$. Furthermore, let k be the number of observable transitions in σ before the first occurrence of a fault.

By the pigeonhole principle, some marking m must satisfy $m=m(\sigma,i)$ for infinitely many i, and thus one can construct an infinite, strictly ascending sequence of indices $(i_j)_{j\geq 0}$ such that $i_0>k$, and all $j\geq 0$ satisfy (i) $m(\sigma,i_j)=m$, and (ii) $s(\sigma,i_j,i_{j+1})\cap({}^\bullet t)^\bullet\neq\emptyset$ for every WF transition t enabled in m (such a subsequence exists since σ is WF and m appears infinitely often). Let $\hat{\sigma}$ be the prefix of σ with $|\ell(\hat{\sigma})|=i_K$.

By the pigeonhole principle, there must be two indices $0\leq j_1<j_2\leq K$ with $m(\rho,i_{j_1})=m(\rho,i_{j_2})=:m'$.

We are now ready to conclude. Consider the execution σ', identical to σ up to $m(\sigma,i_{j_1})$ and then executing $s(\sigma,i_{j_1},i_{j_2})^\omega$. This execution is infinite, contains a fault, and is WF by construction. Moreover, let ρ' be an infinite execution identical to ρ up to $m(\rho,i_{j_1})$ and then executing $s(\rho,i_{j_1},i_{j_2})^\omega$. By construction, $\ell(\sigma')=\ell(\rho')$ but ρ' does not contain a fault. Also, every prefix of ρ' can be extended to a WF fault-free execution by going to the next occurrence of m' and then continuing as in ρ. Thus, (σ',ρ') constitutes a witness. □

We note that in certain practical cases, the witness definition can be simplified. In particular, we consider the case when no fault transition is WF: Then one can simplify the requirements imposed on ρ in Def. 3.

Definition 4 (Special case for witness)**.** *Let $\mathcal{N}$ be a bounded LPN where no fault transition is WF. Then a pair of infinite executions (σ,ρ) with $\ell(\sigma)=\ell(\rho)$ is called* witness *iff σ is WF and contains a fault, and ρ contains no fault.*

Note also that this definition is quite similar to the definition from [1], but with the following important differences: (i) it is correct only for bounded LPNs without WF faults, and (ii) ρ is not required to be WF.

As an example, a witness of WF-diagnosability violation for the LPN in Fig. 3 would be $(t_1t_3^\omega,t_2t_3^\omega)$; note that the latter trace is not WF, but any its prefix can be extended to a WF trace.

It should be noted that the assumption that the faults are not WF is essential for the above definition. Indeed, consider the LPN in Fig. 4. This LPN is trivially WF-diagnosable, as every its infinite WF execution will contain the WF fault transition. However, $(t_2t_1^\omega,t_1^\omega)$ would constitute a witness of WF-diagnosability violation had the assumption about the absence of WF faults been dropped in Def. 4.

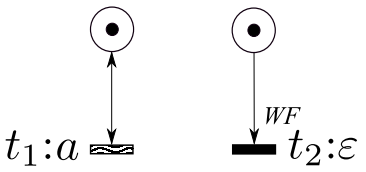

Figure 4. A bounded LPN illustrating that the assumption about faults being non-WF is essential: This LPN is trivially WF-diagnosable, as the fault must occur in every infinite WF execution, but $(t_2t_1^\omega,t_1^\omega)$ would constitute a witness of WF-diagnosability violation had this assumption been dropped in Def. 4.

Lemma 2 (special case for WF-diagnosability)**.** *Let $\mathcal{N}$ be a bounded LPN where no fault transition is WF. Then $\mathcal{N}$ is WF-diagnosable iff no witness satisfying the conditions of Def. 4 exists.*

Proof. If $\mathcal{N}$ is not WF-diagnosable then Lemma 1 provides a witness satisfying also the less restrictive conditions in Def. 4.

For the other direction, suppose that a witness (σ,ρ) according to Def. 4 exists. Take any finite prefix

$\hat{\sigma}$ of σ and let $\hat{\rho}\rho'$ be a decomposition of ρ satisfying $\ell(\hat{\sigma}) = \ell(\hat{\rho})$. To get a contradiction, it is enough to construct an infinite, WF, fault-free continuation of $\hat{\rho}$. If ρ itself is WF then we are done. Otherwise there exists some WF transition t that is enabled at some point in ρ after which ρ contains no more transition from $({}^\bullet t)^\bullet$; note that t is not a fault by the assumption. But this means that firing t cannot disable any transition in the rest of the execution, so we can insert it anywhere into ρ' without disabling the rest of this execution. The repeated application of this insertion process yields the required continuation of $\hat{\rho}$, and it is always can be done in such a way that no enabled WF transition is perpetually ignored by the insertion process, and no transition from ρ' is indefinitely delayed by the newly inserted transitions. □

This result is central for the WF-diagnosability verification method proposed in the next section.

4. Checking WF-diagnosability

In this section we show how checking WF-diagnosability can be re-formulated in terms of LTL-X [4, 7] model checking.

Our approach works for a bounded LPN $\mathcal{N}$. We perform various operations on $\mathcal{N}$ to obtain another bounded LPN $\mathcal{V}$, called the *verifier*, which we check against a *fixed* LTL-X formula (in particular, its size does not depend on $\mathcal{N}$). To achieve this, we exploit the ability of many existing model checkers to handle weak fairness directly.[2]

We first introduce the operations on $\mathcal{N}$ needed to obtain $\mathcal{V}$ (Sect. 4.1), then recall the approach for non-WF diagnosability (Sect. 4.2), and finally present the modifications necessary to handle WF-diagnosability for the special case where no fault transition is WF (Sect. 4.3).

We use the net in Fig. 5 as a running example.

4.1. Net operations

In this paper we are concerned with the state-based LTL-X verification. However, the definition of diagnosability in Sect. 3 is action-based, and thus has to be reformulated in terms of states. The first two operations are defined for this purpose.

Fault monitor We will need to keep track whether some execution contains a fault transition. Given $\mathcal{N}$, the

[2]The algorithm looking for an accepting (lasso-shaped) execution of a Büchi automaton can be modified in such a way as to ignore non-WF executions.

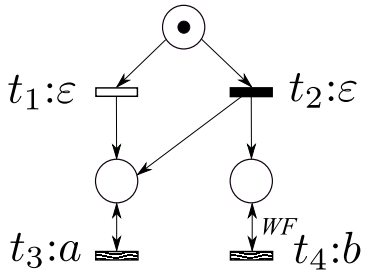

Figure 5. An LPN similar to that in Fig. 3, but with a different choice of a fault transition. It is not diagnosable but WF-diagnosable, as an occurrence of a fault enables t_4, which can be perpetually ignored under the non-WF semantics, but must eventually fire — thus diagnosing the fault — under the WF semantics.

net $\mathcal{N}^{ft}$ denotes $\mathcal{N}$ extended with two additional places $\overline{p_f}$ and p_f of which $\overline{p_f}$ is initially marked, indicating that no fault has happened so far. Then we make every fault transition move a token from $\overline{p_f}$ to p_f, indicating that a fault has occurred. Also, since a fault transition may fire several times in $\mathcal{N}$, another transition f' is added for each fault transition f, in order to simulate these subsequent firings in $\mathcal{N}^{ft}$. The construction is illustrated in Fig. 6, where it is applied to Fig. 5.

In terms of behaviour, $\mathcal{N}$ and $\mathcal{N}^{ft}$ are equivalent in a strong sense. Suppose that the transitions of $\mathcal{N}$ are injectively labelled, and the transitions of $\mathcal{N}^{ft}$ retain these labels, with the label of f and f' being the same. Then these two nets are strongly bisimilar. Moreover, if $\overline{p_f}$ in $\mathcal{N}^{ft}$ is unmarked then a fault occurred in the past.

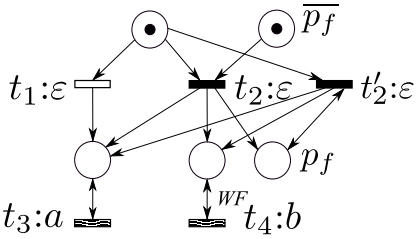

Figure 6. Fault tracking net $\mathcal{N}^{ft}$ for the LPN in Fig. 5.

Stubs We will want to know whether an infinite execution perpetually enables certain transitions. Given a subset of $\mathcal{N}$'s transitions and a 'fresh' initially marked place *stub_monitor*, we can turn these transitions into *stubs* by removing all their outgoing arcs and adding *stub_monitor* to their presets.

Stubs are not meant to be executed: in fact, our

LTL-X formulae will make such executions 'irrelevant' by demanding that *stub_monitor* remains always marked. Then, a 'relevant' WF execution that keeps *stub_monitor* marked cannot enable a stub forever.

Removing transitions We can remove a given subset of transitions from an LPN, together with their incoming and outgoing arcs.

Synchronising Let $\mathcal{N}$ and $\mathcal{N}'$ be two LPNs with disjoint sets of places and transitions, whose transition sets are T and T', respectively. Intuitively, the *synchronisation* of $\mathcal{N}$ and $\mathcal{N}'$ w.r.t. $T_s \subseteq T \uplus T'$ puts $\mathcal{N}$ and $\mathcal{N}'$ side-by-side, and then each transition t of $\mathcal{N}$ is fused with each transition t' of $\mathcal{N}'$ that has the same label (each fusion produces a new transition), provided that t and t' are both in T_s (t and t' remain in the result). Thus the synchronised net has three types of transitions: those from $\mathcal{N}$, those from $\mathcal{N}'$, and the fused ones.

4.2. Verifying ordinary diagnosability

We recall the verification of (non-WF) diagnosability from [5] and show that it is unsuitable for WF-diagnosability. Let $\mathcal{N}$ be the original LPN. The construction works in the following steps:

1. Let $\mathcal{N}^{ft}$ be the fault tracking net corresponding to $\mathcal{N}$.
2. Let $\mathcal{N}'$ be a copy of $\mathcal{N}$.
3. Synchronise $\mathcal{N}^{ft}$ and $\mathcal{N}'$ on the observable transitions in both nets, yielding the net $\mathcal{N}_s$.
4. Remove from $\mathcal{N}_s$ all observable transitions of $\mathcal{N}^{ft}$.
5. Remove from $\mathcal{N}_s$ all observable and fault transitions of $\mathcal{N}'$.
6. Call the resulting net $\mathcal{V}$.

Note that after $\mathcal{V}$ has been built, it is no longer necessary to remember which actions are visible and which are not, and so we can disregard all the labelling and treat $\mathcal{V}$ as an unlabelled PN. This construction is illustrated in Fig. 7.

It turns out [5] that $\mathcal{N}$ is diagnosable iff the following LTL-X property holds for all traces of $\mathcal{V}$:

$$diag \stackrel{\text{df}}{=} \Box \overline{p_f},$$

i.e. we simply require that there is no infinite trace in $\mathcal{V}$ containing an occurrence of a fault.

Conversely, a counterexample satisfying $\Diamond \neg \overline{p_f}$ is an infinite execution of $\mathcal{V}$ containing a fault; when projected to the parts corresponding to $\mathcal{N}^{ft}$ and $\mathcal{N}'$, it gives a witness of (non-WF) diagnosability violation, i.e. two infinite executions of $\mathcal{N}$ that have the same projection on the set of observable actions but the first contains a fault while the second does not. Similarly, such a pair of executions corresponds to an infinite trace of $\mathcal{V}$, with the first being executed by the part of $\mathcal{V}$ corresponding to $\mathcal{N}^{ft}$, and the second (which has no occurrences of faults) being executed by the part of $\mathcal{V}$ corresponding to $\mathcal{N}'$.

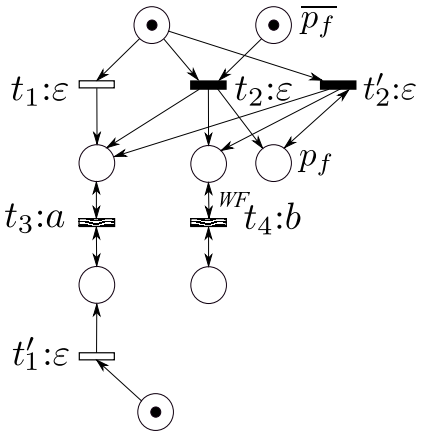

Figure 7. The (non-WF) verifier for the LPN in Fig. 5.

Unfortunately, this construction is not appropriate for WF-diagnosability, even if the executions of the verifier are restricted to be WF. For example, consider the net in Fig. 5. The verifier proposed in [5] is shown in Fig. 7. It has an infinite execution containing a fault, $t_2 t_1' t_3^\omega$, which, when projected to $\mathcal{N}^{ft}$ and $\mathcal{N}'$, yields a pair of traces constituting a witness of diagnosability violation. However, this verifier cannot be used for checking WF-diagnosability simply by restricting its executions to be WF, as the same execution $t_2 t_1' t_3^\omega$ is actually WF, since t_4 is not permanently enabled by it (in fact, it is a dead transition in the verifier). Therefore, this execution is a false negative (the original LPN is in fact WF-diagnosable). Note that when this WF execution of the verifier is projected to $\mathcal{N}^{ft}$ and $\mathcal{N}'$, the resulting pair of traces will not constitute a witness of WF-diagnosability, as the former projection will be a non-WF execution of $\mathcal{N}^{ft}$ that perpetually ignores an enabled transition t_4.

Below, we amend $\mathcal{V}$ to fix this problem for bounded LPNs with no WF faults.

4.3. Verifier for non-WF fault transitions

Let $\mathcal{N}$ be a bounded LPN, in which no fault transition is WF. We keep the basic idea of the verifier con-

struction from Sect. 4.2, i.e. our verifier $\mathcal{V}_{WF}$ will be the synchronisation of two nets, and a counterexample to our LTL-X formula will give a faulty execution σ in one net, and a fault-free execution ρ in the other net, such that (σ, ρ) is a witness.

The first important change is to check the formula only against WF executions. As seen in Sect. 4.2, this alone is not enough: The false counterexample obtained for Fig. 5 comes from the fact that $\mathcal{V}_{WF}$ allows σ to perpetually ignore a transition (here: t_4) if ρ does not enable it. We use stubs to prevent this from happening. More precisely, $\mathcal{V}$ is constructed as follows:

1. Obtain the net $\mathcal{N}_s$ as in Sect. 4.2; its fused transitions are declared non-WF.
2. Turn in $\mathcal{N}_s$ the observable WF transitions of $\mathcal{N}^{ft}$ into stubs; they remain WF.
3. Remove from $\mathcal{N}_s$ all observable and fault transitions of $\mathcal{N}'$.
4. In $\mathcal{N}_s$, make the remaining transitions of $\mathcal{N}'$ non-WF.
5. Call the resulting net $\mathcal{V}_{WF}$.

Fig. 8 shows the verifier $\mathcal{V}_{WF}$ for the LPN in Fig. 5.

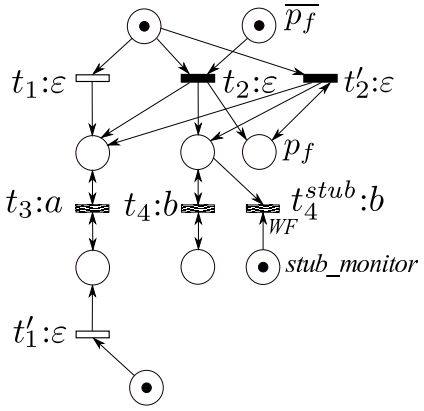

Figure 8. The WF verifier for the LPN in Fig. 5.

Now we can formulate WF-diagnosability of the original $\mathcal{N}$ as a fixed LTL-X formula on $\mathcal{V}_{WF}$ that has to be checked for infinite WF executions only:

$$diag_{WF} \stackrel{\text{df}}{=} \Box\overline{p_f} \vee \Diamond\neg stub\_monitor.$$

Thus a counterexample is an infinite WF execution containing a fault but no stubs.

Theorem 1 (Correctness of specialised WF Verifier)**.** *Let $\mathcal{N}$ be a bounded LPN where no fault transition is WF. Then $\mathcal{N}$ is WF-diagnosable iff all infinite WF executions of $\mathcal{V}_{WF}$ satisfy $diag_{WF}$.*

Proof. According to Lemma 2, $\mathcal{N}$ is WF-diagnosable iff no witness satisfying Def. 4 exists.

First, suppose $diag_{WF}$ is false, i.e. $\mathcal{V}_{WF}$ has an infinite WF execution τ that contains a fault and no stubs. Let σ and ρ be the projections of τ to $\mathcal{N}^{ft}$ and $\mathcal{N}'$, respectively. We claim that (σ, ρ) is a witness. Indeed, since $\mathcal{N}$ has no divergencies, τ must contain infinitely many observable transitions. Thus, both σ and ρ are infinite, and $\ell(\sigma) = \ell(\rho)$ holds; moreover, σ contains a fault but ρ does not. Finally, σ must be WF because τ is and no stubs are fired.

For the reverse direction, it is fairly straightforward to see that any witness (σ, ρ) from Def. 4 gives rise to an execution τ of $\mathcal{V}_{WF}$ violating $diag_{WF}$. Moreover, τ is WF because σ is. The fact that ρ is not necessarily WF does not play a role, as ρ is executed in the part of the verifier corresponding to $\mathcal{N}'$ and so contains no WF transitions by construction. □

5. Experimental results

In this section we present experimental results for the proposed WF diagnosability approach. Furthermore, we demonstrate that the proposed approach can easily be lifted from low-level Petri nets to high-level ones: both the used benchmarks and the corresponding verifiers were modelled using high-level PNs.

For the verification, we used the MARIA (modular reachability analyser) tool [6]. Since MARIA supports modular verification, it was possible to exploit the modular structure of the verifier (recall that it is built by synchronising two LPNs, see Sect. 4) to significantly speed up the verification.

It should be noted that finding interesting benchmarks was a challenging task: Despite a lot of theoretical work done in the area of diagnosability, rather few practical experiments have been conducted. Moreover, we wanted benchmarks where weak fairness is essential, i.e. removing some transitions from the WF set would make the system undiagnosable. Hence, we designed the following two new families of scalable benchmarks, available from the authors upon request.

COMMBOX (n) Fig. 9 shows a high-level Petri net modelling the system comprising commutator boxes and an inspector, together with the verifier. It models n boxes commuting telephone calls. Normally, a box just handles telephone calls (the *normal_execution* transition), but occasionally it may register a fault (the *fault* transition) in a telephone line. Such an event, however, does not take the box out of action, and it still continues to commute calls (the *normal_execution* transition) and register further faults (the *fault_* transition). Never-

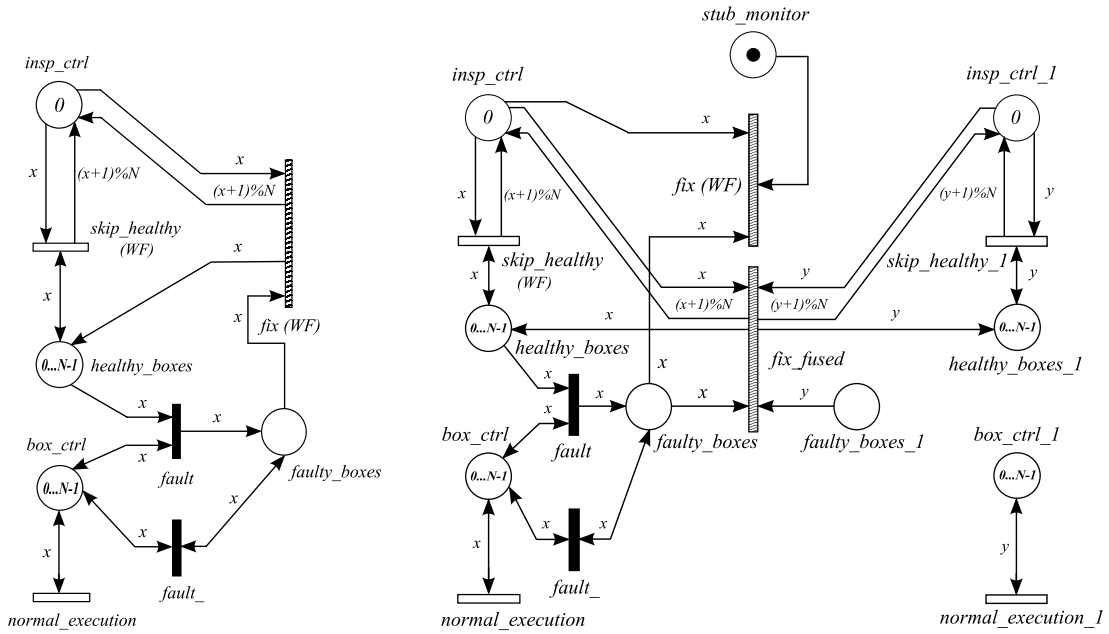

Figure 9. The COMMBOX (*n*) benchmark (left) and the corresponding verifier (right).

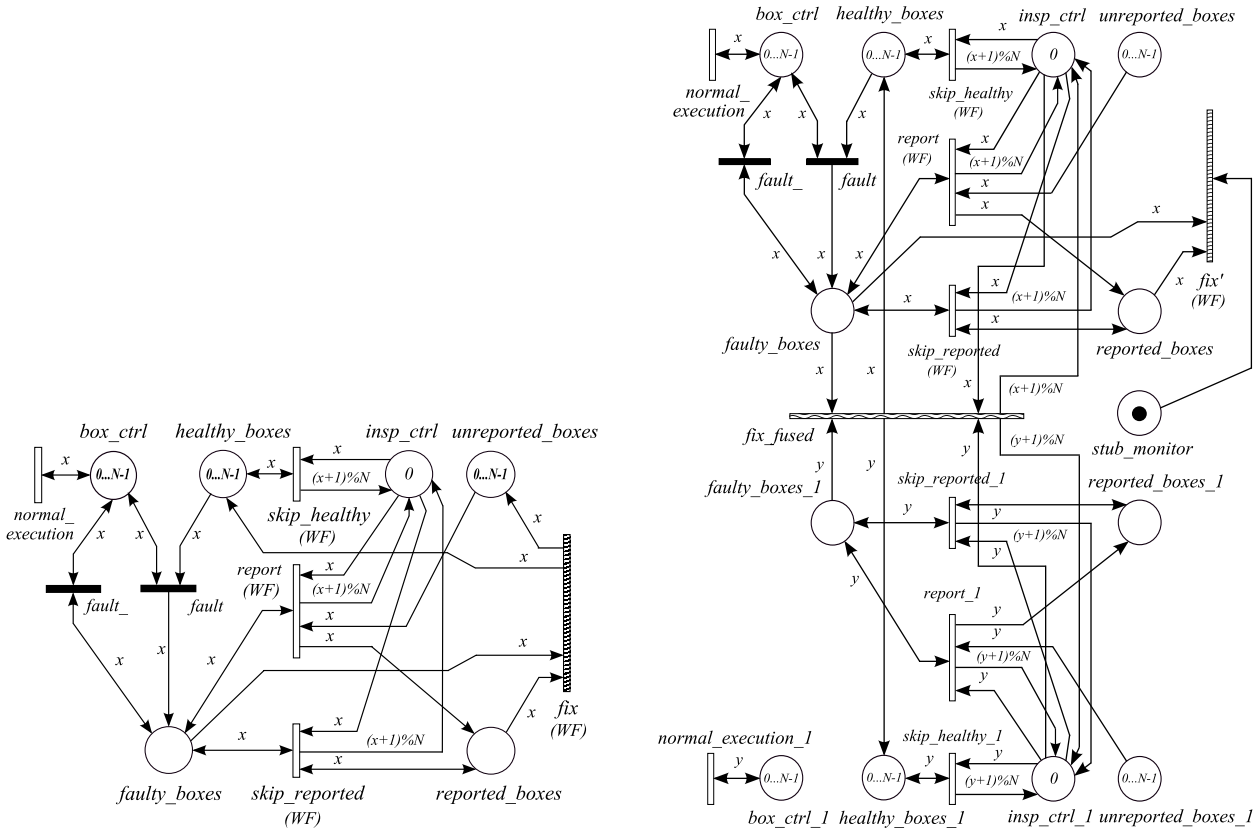

Figure 10. The COMMBOXTECH (*n*) benchmark (left) and the corresponding verifier (right).

theless, the registered faults have to be considered and fixed, and so there is an inspector visiting the boxes on a round trip and fixing them if necessary (the *skip_healthy* and *fix* transitions). It is assumed that *fix* is the only observable transition, and one can be sure that a fault has occurred once it fires. Nevertheless, it is possible that the inspector indefinitely postpones visiting the boxes (i.e. its transitions are always preempted by, e.g., *normal_execution* which is always enabled), and so the system is undiagnosable. However, if the transitions modelling the inspector are WF, the system becomes diagnosable, as after a fault the *fix* transition is eventually executed.

Benchmarks	Vrf Time	Vrf Modular Time
CommBox (4)	<1	<1
CommBox (5)	4	1
CommBox (6)	12	4
CommBox (7)	38	14
CommBoxTech (4)	17	6
CommBoxTech (5)	101	33
CommBoxTech (6)	561	162
CommBoxTech (7)	2995	Bug

Table 1. Experimental results for CommBox and CommBoxTech benchmarks (all nets are diagnosable).

CommBoxTech (*n*) Fig. 10 shows an elaborated version of the above system, together with its verifier: The inspector reports the faults to a technician, who then fixes them. Again, the inspector's and technician's transitions must both be WF to make the system diagnosable.

The experimental results are summarised in Table 1, where the meaning of the columns is as follows (from left to right): name of the benchmark, verification time, and verification time using the modular representation of the verifier. (The time is measured in seconds.) All experiments were conducted on a PC with 64-bit Windows 7 operating system, an Intel Core i7 2.8GHz Processor with 8 cores and 4GB RAM (no parallelisation was used for the results in this table). The Maria tool has confirmed that the diagnosability property holds for these benchmarks. We also discovered a bug in Maria: for the CommBoxTech (7) benchmark there is a mismatch between the verification outcomes in the standard and modular modes.

We also wanted to check that the WF constraint is essential for diagnosability, i.e. that if even one transition is removed from the WF set then the system becomes undiagnosable. These results are summarised in Tables 2 and 3. The Maria tool confirmed that this is the case for the transitions *skip_healthy* and *fix* for the CommBox family, and for the transitions *skip_healthy, report* and *fix* for the CommBoxTech family. However, to our surprise, the CommBoxTech benchmarks remain diagnosable even when the *skip_reported* transition is removed from the WF set: This is in fact correct, as *skip_reported* can be enabled only after some fault has been reported, i.e. some fault will be diagnosed due to the *fix* transition even if *skip_reported* never fires.

6. Conclusions

In this paper we have identified a major flaw in the previous definition of WF-diagnosability in the literature, and proposed a corrected notion. Moreover, under a simplifying assumption that the fault transitions are non-WF, we have presented an efficient technique for verifying WF-diagnosability based on a reduction to LTL-X model checking. An important advantage of this method is that the LTL-X formula is fixed — in particular, the WF assumption does not have to be expressed as a part of it (which would make the formula length proportional to the size of the specification), but rather the ability of existing model checkers to handle weak fairness directly is exploited.

We also created two families of scalable benchmarks, where the weak fairness is essential for diagnosability. The proposed WF-diagnosability verification method has been tested on these benchmarks, and the experimental results demonstrate its feasibility in practice.

Lemma 1 indicates that WF-diagnosability in the general settings in which any transition may be WF, is feasible in principle; we are already pursuing ideas for the construction and optimisation of adequate verifiers. Other possible directions of future work include developing a theory that would allow one to cope with strong fairness.

Acknowledgements. This research was supported by the Epsrc grant EP/K001698/1 (Uncover) and by the project ImpRo ANR-2010-BLAN-0317.

References

[1] A. Agarwal, A. Madalinski, and S. Haar. Effective verification of weak diagnosability. In *Proc. SAFEPROCESS'12*. IFAC, 2012.

[2] S. Haar, C. Rodríguez, and S. Schwoon. Reveal your faults: It's only fair! In *Proc. ACSD'13*, pages 120–129. IEEE Computer Society Press, 2013.

Benchmarks	WF enabled *skip_healthy*	WF enabled *fix*	Vrf Time	Vrf Modular Time	Diagnosable
CommBox (4)	✓	×	1	<1	×
	×	✓	1	<1	×
CommBox (5)	✓	×	1	1	×
	×	✓	2	1	×
CommBox (6)	✓	×	2	1	×
	×	✓	2	2	×
CommBox (7)	✓	×	2	2	×
	×	✓	3	2	×

Table 2. Experimental results for CommBox with reduced WF set.

Benchmarks	WF enabled *skip_healthy*	WF enabled *report*	WF enabled *skip_reported*	WF enabled fix	Vrf Time	Vrf Modular Time	Diagnosable
CommBoxTech (4)	✓	✓	✓	×	2	1	×
	✓	✓	×	✓	17	6	✓
	✓	×	✓	✓	1	1	×
	×	✓	✓	✓	8	3	×
CommBoxTech (5)	✓	✓	✓	×	3	2	×
	✓	✓	×	✓	102	30	✓
	✓	×	✓	✓	2	1	×
	×	✓	✓	✓	42	14	×
CommBoxTech (6)	✓	✓	✓	×	3	2	×
	✓	✓	×	✓	560	147	✓
	✓	×	✓	✓	2	1	×
	×	✓	✓	✓	6	61	×
CommBoxTech (7)	✓	✓	✓	×	4	3	×
	✓	✓	×	✓	2853	Bug	✓
	✓	×	✓	✓	3	2	×
	×	✓	✓	✓	1099	4	×

Table 3. Experimental results for CommBoxTech with reduced WF set.

[3] S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial algorithm for testing diagnosability of discrete event systems. In *IEEE Trans. on Autom. Control*, 2001.

[4] L. Lamport. What good is temporal logic? In *Proc. IFIP Congr.'83*, pages 657–668. Elsevier, 1983.

[5] A. Madalinski and V. Khomenko. Diagnosability verification with parallel LTL-X model checking based on Petri net unfoldings. In *Proc. SysTol'10*, pages 398–403. IEEE Computer Society Press, 2010.

[6] MARIA: The modular reachability analyzer, 2005. URL: `http://www.tcs.hut.fi/Software/maria/index.en.html`.

[7] A. Pnueli. The temporal logic of programs. In *Proc. FOCS'77*, pages 46–57, 1977.

[8] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of Discrete Events Systems. *IEEE Trans. on Autom. Control*, 40(9):1555–1575, 1995.

[9] A. Schumann and Y. Pencolé. Scalable diagnosability checking of event-driven systems. In *Proc. IJCAI'07*, pages 575–580, 2007.

[10] W. Vogler. Fairness and partial order semantics. *Inf. Process. Lett.*, 55(1):33–39, 1995.