"Polynomial Time Verification of Decentralized Diagnosability of Discrete Event Systems" Versus "Decentralized Failure Diagnosis of Discrete Event Systems": A Critical Appraisal

Marcos V. Moreira, João C. Basilio, *Member, IEEE*, and Felipe G. Cabral

***Abstract*—In [1], the authors claim that there is an oversight in [2], in the sense that the proposed verifier is, in general, nondeterministic and the computational complexity analysis is incorrect. The authors in [1] also claim that the complexity of the verification algorithm presented in [3] is reduced when considering the more restrictive setting of projection masks, in contrast to the more general non-projection masks case, and equals the complexity of the verification algorithm presented in [2]. In this note, we show that the computational complexity analysis of [2] is actually correct and that the complexity of the verification algorithm presented in [3] is not reduced without additional modification of the algorithm (not yet proposed in the literature) if projection masks are used, and, therefore, is not equal to the complexity of the algorithm presented in [2].**

***Index Terms*—Complexity analysis, discrete event systems, fault diagnosis.**

I. Introduction

In [2], a new polynomial time algorithm to verify the codiagnosability of a discrete event system is proposed for the decentralized diagnosis scheme presented in Protocol 3 of [4], which consists of a set of local diagnosers that do not communicate among each other, being a fault event diagnosed when at least one of the local diagnosers identifies its occurrence.

In [1], it is claimed that the computational complexity analysis carried out in [2] has an oversight since the authors of [2] overlooked the fact that the proposed verifier is in general nondeterministic. In addition, the authors of [1] claim that the computational complexity of the algorithm presented in [3] can be smaller when dealing with projection masks instead of non-projection masks and is equal to the computational complexity of [2].

In this note, we clarify all concerns raised in [1] regarding the algorithm proposed in [2]. In this regard, we show that: i) the verifier presented in [2] is deterministic in the sense that its transition function is deterministic and there is no transition labeled with the empty trace; ii) the computational complexity analysis of [2], *claimed* to be incorrect in [1], is actually correct, and the computational complexity of the algorithm proposed in [2] is $O(m|X|^{m+1} \times |\Sigma|)$; iii) the verifier proposed in [3] has complexity $O(|X|^{m+1} \times |\Sigma|^{m+1})$ even when projection masks are used.

Manuscript received December 27, 2013; revised July 18, 2014, November 24, 2014, and April 2, 2015; accepted April 14, 2015. Date of publication April 29, 2015; date of current version December 24, 2015. This work was supported in part by PETROBRAS, by FAPERJ under Grant E-26/110155/2014, and by the Brazilian Research Council (CNPq) under Grant 309084/2014-8 and Grant 306592/2010-0. Recommended by Associate Editor D. Hristu-Varsakelis.

The authors are with the COPPE-Programa de Engenharia Elétrica, Universidade Federal do Rio de Janeiro, 21949-900 Rio de Janeiro-RJ, Brazil (e-mail: moreira.mv@poli.ufrj.br; basilio@dee.ufrj.br; felipecabral@poli.ufrj.br).

Digital Object Identifier 10.1109/TAC.2015.2427711

II. Verifier Proposed in [2] Is Actually Deterministic

Instead of modeling the transitions associated with unobservable events with ε-transitions, another possibility is to partition the set of events as $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, where Σ_o and Σ_{uo} denote, respectively, the observable and unobservable event sets. In this case, assuming that the system automaton G has only one initial state and the transition function is deterministic, G is said to be a deterministic automaton with unobservable events [5].

In the construction of the verifier proposed in [2], unobservable events are created by renaming the unobservable events of the system. Thus, the set of events of the verifier is augmented and this fact is clearly presented in Algorithm 1 of [2], where the event set of verifier $G_V = G_{N,1} \| G_{N,2} \| G_F$ is given by $\Sigma_{R_1} \cup \Sigma_{R_2} \cup \Sigma$. Since $G_{N,1}$, $G_{N,2}$, and G_F are deterministic automata with unobservable events, verifier G_V is also a deterministic automaton with unobservable events.

It is worth remarking that the fact that the cardinality of the event set of verifier G_V be greater than the cardinality of the event set of G does not interfere in the complexity analysis of the verification algorithm in [2]. Indeed, as remarked in [1], the renamed events are considered in [2, Table 1], showing that these events have been actually taken into account.

III. Further Details on the Computational Complexity Analysis in [2]

Since all operations presented in the algorithm proposed in [2] are linear in the number of states and transitions of the automata constructed in the algorithm (including the search for strongly connected components that violate the codiagnosability condition), the computational complexity has been analyzed in terms of the size of the automata.

The maximum number of states and transitions of verifier G_V computed according to [2, Algorithm 1] are, respectively, $2 \times |X|^{m+1}$ and $2 \times |X|^{m+1}[|\Sigma| + m(|\Sigma| - |\Sigma_f|)]$, where m is the number of local diagnosers, X is the state space of the system, Σ is the set of events and Σ_f is the set of fault events. Let us now explain how the complexity presented in [2, Table 1] has been achieved

$$\begin{aligned} |\Sigma| + m\left(|\Sigma| - |\Sigma_f|\right) &= |\Sigma| - |\Sigma_f| + |\Sigma_f| + m\left(|\Sigma| - |\Sigma_f|\right) \\ &= (m+1)\left(|\Sigma| - |\Sigma_f|\right) + |\Sigma_f| \\ &\leq 2m\left(|\Sigma| - |\Sigma_f|\right) + |\Sigma_f|. \end{aligned}$$

Since, in general, the codiagnosability verification is carried out for each type of fault separately, then, without loss of generality, Σ_f can be expressed as a singleton, i.e., $\Sigma_f = \{\sigma_f\}$. In [2], the cardinality of Σ_f is assumed to be a constant independent of the size of the input. Thus, it is not difficult to see that

$$2m\left(|\Sigma| - |\Sigma_f|\right) + |\Sigma_f| \leq cm\left(|\Sigma| - |\Sigma_f|\right) \tag{1}$$

for

$$c \geq 2 + \frac{|\Sigma_f|}{m\left(|\Sigma| - |\Sigma_f|\right)}$$

where c is a positive constant. This implies that, for large inputs, we can always choose a constant c that satisfies inequality (1). Therefore, the complexity of [2, Algorithm 1] is indeed $O(m|X|^{m+1}(|\Sigma| - |\Sigma_f|))$.

Assume, now, that the number of fault events is not constant and can also grow. In this case, the following inequalities hold true:

$$\begin{aligned} |\Sigma| + m\left(|\Sigma| - |\Sigma_f|\right) &\leq |\Sigma| + m\left(|\Sigma| + |\Sigma_f|\right) \\ &\leq |\Sigma| + 2m|\Sigma| \\ &= (2m+1)|\Sigma|. \end{aligned}$$

It is not difficult to see that

$$(2m+1)|\Sigma| \leq cm|\Sigma|$$

for

$$c \geq 2 + \frac{1}{m}.$$

Thus, in this case, the complexity of [2, Algorithm 1] is $O(m|X|^{m+1} \times |\Sigma|)$.

We have, therefore, proven in this subsection that the complexity analysis presented in [2] is actually correct.

IV. Complexity Analysis of the Verification Algorithm Proposed in [3]

In [3], a verification algorithm is proposed for codiagnosability considering the more general case of non-projection masks. The algorithm is based on the construction of a testing automaton T and the search for cycles that violate the codiagnosability condition. Notice that, since the more restrictive setting of projection masks was not explicitly considered in [3] (although it is a special case of the non-projection mask case), it must be assumed that the verification algorithm proposed in [3] is the same if projection masks are used instead of non-projection masks. Thus, the complexity of the algorithm is independent of the type of mask used.

In [1] the authors claim, without proof, that "*when the local masks are projection type, there will be a smaller number of choices for a transition label in T.*" According to the authors, this is true for the following reasons:

R1. "*When the first-component is an event, the $(i+1)$th-component has only one choice depending on whether the events is locally observable or unobservable*;"

R2. "*when the first-component is ε (so the first-component doesn't evolve) one of the remaining m-components can execute a locally nonfaulty unobservable event and evolve (while the others don't evolve).*"

As we will illustrate in the sequel, the claimed reduction in the number of transitions of T is not observed if one simply applies the algorithm presented in [3], i.e., reasons **R1** and **R2** are not verified in the algorithm presented in [3].

Example 1: Let G, shown in Fig. 1, be the automaton model of a system and consider that there exist two local diagnosers, G_{D_1} and G_{D_2}, whose observable event sets are, respectively, $\Sigma_{o_1} = \{a, b\}$ and $\Sigma_{o_2} = \{b, c\}$. The fault event set is given by $\Sigma_f = \{\sigma_f\}$. Let us now consider the computation of the testing automaton T. In this example, the specification is given by automaton H, shown in Fig. 2, that represents the nonfaulty behavior of the system. Following the steps of [3, Algorithm 1], we obtain the testing automaton T presented in Fig. 3.

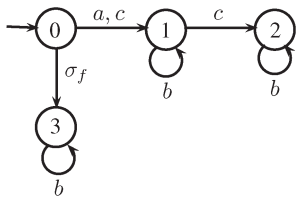

Fig. 1. Plant automaton G.

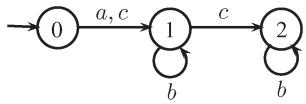

Fig. 2. Specification automaton H.

Let us show first why reason **R1** is not verified. Initially, notice that there are 10 transitions departing from state 0000 of testing automaton T, where one transition is labeled with aaa and another transition is labeled with $aa\varepsilon$. These transitions are created because a is feasible in state 0, and a is observable for local diagnoser G_{D_1} and is unobservable for local diagnoser G_{D_2}, i.e., $P_{o_1}(a) = a$ and $P_{o_2}(a) = P_{o_2}(\varepsilon) = \varepsilon$, where $P_{o_i} : \Sigma^\star \to \Sigma_{o_i}^\star$, $i = 1, 2$ denote projections. Thus, according to Step 2 of the algorithm proposed in [3], we have two possibilities for the third component when the first component is an event, which shows that reason **R1** was not verified in this example.

Let us now show that reason **R2** is also not verified in T. Notice that in state 0000 there exists a transition labeled with εca. This transition is possible because the events a and c are feasible in state 0, and event c is unobservable for local diagnoser G_{D_1} and event a is unobservable for local diagnoser G_{D_2}, i.e., $P_{o_1}(c) = P_{o_1}(\varepsilon) = \varepsilon$ and $P_{o_2}(a) = P_{o_2}(\varepsilon) = \varepsilon$. Thus, in accordance with Step 2 of the algorithm presented in [3], when the first component of the transition label is ε, then it is possible to have more than one component equal to a locally unobservable event. This contradicts reason **R2** that states that only one of the remaining components of the transition label can execute a locally unobservable event and evolve. Thus, reason **R2** is not verified either.

Based on the above considerations we can see that, in the worst-case, the complexity of the algorithm is indeed $O(|X|^{m+1} \times |\Sigma|^{m+1})$. In order to illustrate this fact, assume now that event a is unobservable for both local diagnosers G_{D_1} and G_{D_2}. Since $P_{o_1}(a) = P_{o_1}(\varepsilon) = \varepsilon$ and $P_{o_2}(a) = P_{o_2}(\varepsilon) = \varepsilon$, and a is feasible in state 0, then, following Step 2 of the algorithm presented in [3], seven transitions departing from state 0000 must be created, being labeled as $\varepsilon\varepsilon a$, $\varepsilon a\varepsilon$, $a\varepsilon\varepsilon$, $aa\varepsilon$, εaa, $a\varepsilon a$, aaa, i.e., $(2^3 - 1)$ transitions, which shows that, in the worst-case, there are $[(|\Sigma| + 1)^{m+1} - 1]$ transitions for each state in T.

In order to achieve a reduction in the number of transitions it is necessary to formally introduce reasons **R1** and **R2** as modifications in the algorithm presented in [3] and also provide the corresponding proof of correctness. Thus, the only available result regarding the complexity of the algorithm proposed in [3] is $O(|X|^{m+1} \times |\Sigma|^{m+1})$, whether masks are of projection type or not.

Let us now compute the verifier automaton according to Algorithm 1 presented in [2]. Following the first and second steps of [2, Algorithm 1], automata G_N and G_F, shown, respectively, in Figs. 4 and 5, are computed. In the sequel, the unobservable events of G_N are renamed according to each local observation, generating automata G_{N_1} and G_{N_2}. Finally, the verifier automaton $G_V = G_{N_1} \| G_{N_2} \| G_F$ is computed. The verifier automaton G_V, presented in Fig. 6, has

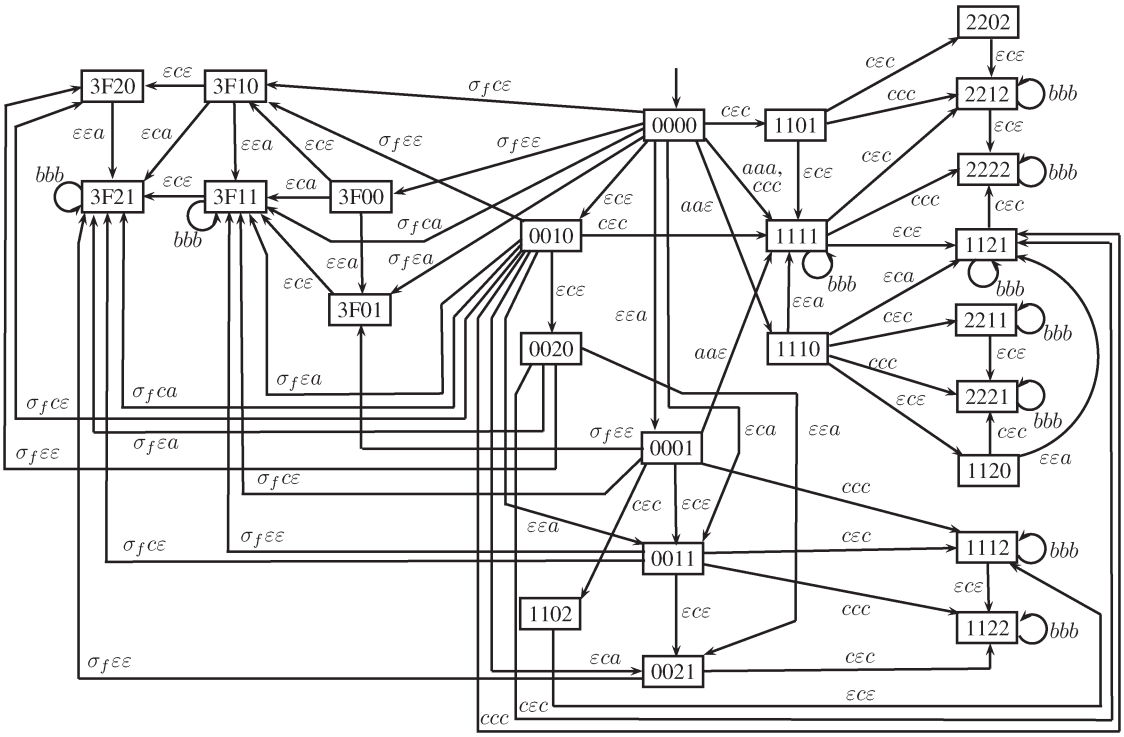

Fig. 3. Testing automaton T.

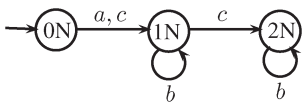

Fig. 4. Nonfaulty automaton G_N.

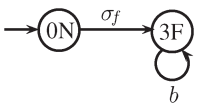

Fig. 5. Faulty automaton G_F.

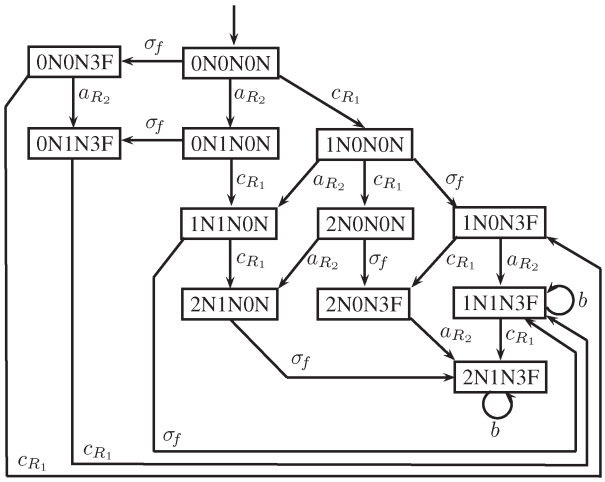

Fig. 6. Verifier G_V.

only 12 states and 22 transitions whereas the testing automaton T has 25 states and 75 transitions.

V. Concluding Remarks

Although the verification methods presented in [2] and [3] are based on the construction of different automata, in both methods, as in all verification methods presented in the literature (see, for instance, [6]–[8]), the main idea is to search for the faulty and the corresponding nonfaulty traces of the language generated by the system that have the same local observations. In [2], the verifier is computed by making the parallel composition of G_F and m renamed copies of G_N. Thus, a state of G_V, x_V, is composed of a state of G_F, x_F, and states of G_{N_i}, x_{N_i}, for $i = 1, \ldots, m$, i.e., $x_V = (x_{N_1}, \ldots, x_{N_m}, x_F)$.

Notice, initially, that for a given state x_V, a non-renamed event σ is feasible if one of the following conditions is satisfied: i) σ is unobservable for all local diagnosers and σ is feasible for x_F; ii) σ is observable for at least one local diagnoser, and σ is feasible for x_F and for all components x_{N_i} of x_V whose associated local diagnoser G_{D_i} observes σ. Therefore, a transition labeled with the non-renamed event σ in G_V represents the synchronization of G_F with all copies of G_N for which σ is observable. It is not difficult to see that reason **R1** implements a similar idea in the algorithm proposed in [3], in the sense that when the first component of the transition label of T is an event, the $(i+1)$th-component is forced to be equal to the event or ε. Notice, now, that a renamed event σ_{R_i} is feasible in a given state x_V, if it is feasible in its ith component x_{N_i}. After the occurrence of σ_{R_i}, only the ith component of x_V evolves. This is the same idea implemented by reason **R2** that states that when the first component of the transition label of T is ε only one of the remaining m-components is allowed to execute a locally nonfaulty unobservable event and evolve.

As illustrated in Example 1, the non-verification of **R1** and **R2** in the algorithm proposed in [3] is the main reason why the computational cost of the method proposed in [3] is larger than the computational cost of the method presented in [2] when projection masks are used. Since there are similarities between reasons **R1** and **R2** and what is actually implemented in [2], it is likely that the computational complexity of the algorithm proposed in [3] can be reduced, as claimed in [1], by

the applications of reasons **R1** and **R2**. However, as shown in the above analysis and in Example 1, reasons **R1** and **R2** are not direct consequences of the replacement of general masks with projection masks, as claimed in [1], but require a formal modification of the algorithm proposed in [3] and the corresponding proof of correctness, which are not included in [1].

REFERENCES

[1] R. Kumar and S. Takai, "Comments on "Polynomial time verification of decentralized diagnosability of discrete event systems" vs. "Decentralized failure diagnosis of discrete event systems": Complexity clarification," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1391–1392, May 2014.

[2] M. V. Moreira, T. C. Jesus, and J. C. Basilio, "Polynomial time verification of decentralized diagnosability of discrete event systems," *IEEE Trans. Autom. Control*, vol. 56, pp. 1679–1684, 2011.

[3] W. Qiu and R. Kumar, "Decentralized failure diagnosis of discrete event systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 36, no. 2, pp. 384–395, Mar. 2006.

[4] R. Debouk, S. Lafortune, and D. Teneketzis, "Coordinated decentralized protocols for failure diagnosis of discrete event systems," *Discrete Event Dynam. Syst.: Theory and Applic.*, vol. 10, no. 1, 2000.

[5] C. Cassandras and S. Lafortune, *Introduction to Discrete Event System*. Secaucus, NJ, USA: Springer-Verlag, 2008.

[6] S. Jiang, Z. Huang, V. Chandra, and R. Kumar, "A polynomial algorithm for testing diagnosability of discrete-event systems," *IEEE Trans. Autom. Control*, vol. 46, no. 8, pp. 1318–1321, Aug. 2001.

[7] T.-S. Yoo and S. Lafortune, "Polynomial-time verification of diagnosability of partially observed discrete-event systems," *IEEE Trans. Autom. Control*, vol. 47, no. 9, pp. 1491–1495, Sep. 2002.

[8] Y. Wang, T.-S. Yoo, and S. Lafortune, "Diagnosis of discrete event systems using decentralized architectures," *Discrete Event Dynam. Syst.: Theory And Applic.*, vol. 17, pp. 233–263, 2007.