# New Algorithms for Verification of Relative Observability and Computation of Supremal Relatively Observable Sublanguage

Marcos V. S. Alves, Lilian K. Carvalho, *Member, IEEE*, and João C. Basilio, *Senior Member, IEEE*

***Abstract*—In this technical note, we present a new property of relative observability, and based on this property, we propose two algorithms: the first one, that has polynomial complexity, verifies if a regular language is relatively observable; the second algorithm computes the supremal relatively observable sublanguage of a given regular language. Although the latter has exponential complexity, it is more efficient than a recently proposed algorithm, which has double exponential complexity. Moreover, the algorithm proposed here has polynomial complexity when the automaton that marks the specification language is state partition.**

***Index Terms*—Automaton, discrete event systems, observability, supervisory control, supremal sublanguage.**

## I. Introduction

The search for a more permissive observable sublanguage of a given language is a research topic that has been attracting the Discrete Event System (DES) community for some time now [1]–[8]. With a view to circumventing the deficiency of language observability regarding the non-existence of the supremal observable sublanguage a new definition of observability, called relative observability, has been recently proposed [9]. As shown in [9], relative observability is closed under set union operation, as opposed to observability that does not possess this property, and is also less conservative than normality [1], although both share the set union closure property. Therefore, the supremal relatively observable sublanguage always exists and is larger compared to supremal normal sublanguage.

Although it has been introduced only recently, relative observability has already received considerable attention in the DES community [10]–[14]. Two algorithms for computing the supremal relatively observable sublanguage of a given language $K$ with respect to ambient language $\overline{K}$ have been proposed in [9] and [15]; the former has double exponential complexity whereas the computational complexity of the algorithm proposed in [15] cannot be easily determined because it is difficult to infer how many iterations it needs to converge in the worst case, although, the computational complexity of each iteration is, at least, exponential.

In this technical note, we propose two new algorithms. The first algorithm verifies if a regular language, $K$, is relatively observable with respect to a given ambient language, $\overline{C}$, a plant, $G$, and a projection $P_o$. This algorithm has polynomial complexity, and, since relative observability is equivalent to observability when $\overline{K}$ is chosen as ambient language, it can be also applied to verify if a language $K$ is observable with respect to $G$ and $P_o$. The second algorithm computes the supremal relatively observable (with respect to $\overline{C}$, $G$ and $P_o$) sublanguage of a regular language $K$. This algorithm has exponential complexity, and is, therefore, considerably more efficient than that proposed in [9], which has double exponential complexity. The key to the success of this algorithm is a new property on relative observability which ensures that for any ambient language $\overline{C}$, there exists an equivalent reduced ambient language that is a subset of $\overline{C}$. It is worth remarking that the computational complexity of the algorithm proposed here for the computation of the supremal relatively observable sublanguage becomes polynomial when the automaton that marks $K$ is state partition [2], [3], [16]–[18].

This technical note is organized as follows: we present preliminary concepts and notations in Section II; we present and prove a new property of relative observability in Section III; we propose an algorithm for the verification of relative observability in Section IV, and an algorithm for the computation of the supremal relatively observable sublanguage of regular languages in Section V; we analyze the complexities of the proposed algorithms in Section VI; finally we draw some conclusions in Section VII. Preliminary results of this technical note are presented in [19]. The main differences between this technical note and [19] are that, here, we introduce a new proposition that has led to a reduction in the computational complexity of the proposed algorithms, and we present the proofs for all results.

## II. Background Preliminaries

Let $G = (X, \Sigma, f, x_0, X_m)$ denote a deterministic finite state automaton, where $X$ is the finite set of states, $\Sigma$ is the finite set of events, $f : X \times \Sigma \rightarrow X$ is the transition function, partially defined over its domain, $x_0$ is the initial state, and $X_m$ is the set of marked states. The transition function $f$ is also extended to $f : X \times \Sigma^* \rightarrow X$, where $\Sigma^*$ denotes the Kleene closure of $\Sigma$. We use the notations $f(x, s)!$ and $f(x, s)\not!$ to denote that $f(x, s)$ is defined and undefined, respectively. The generated and marked languages of $G$ will be denoted, respectively, as $L(G)$ and $L_m(G)$. We say that an automaton $G$ is nonblocking, if $L(G) = \overline{L_m(G)}$, where $\overline{L}$ denotes the prefix-closure of a language $L$. Throughout the text, $|B|$ (resp. $|s|$) denotes the cardinality (resp. length) of set $B$ (resp. string $s$).

We assume that $\Sigma$ can be partitioned as $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, where $\Sigma_o$ and $\Sigma_{uo}$ are, respectively, the set of observable and unobservable events. The natural projection [20] is defined as $P_o : \Sigma^* \rightarrow \Sigma_o^*$, with the following properties: *(i)* $P_o(\varepsilon) = \varepsilon$, where $\varepsilon$ denotes the empty string; *(ii)* $P_o(\sigma) = \sigma$, if $\sigma \in \Sigma_o$; *(iii)* $P_o(\sigma) = \varepsilon$, if $\sigma \in \Sigma_{uo}$; *(iv)* $P_o(s\sigma) = P_o(s)P_o(\sigma)$, for $s \in \Sigma^*$ and $\sigma \in \Sigma$. The inverse projection $P_o^{-1}$ is defined as $P_o^{-1}(t) = \{s \in \Sigma^* : P_o(s) = t\}$. Both, the projection and the inverse projection operations, can be extended to languages by applying $P_o(s)$ and $P_o^{-1}(s)$ to all strings $s$ in the language.

Throughout the text $Ac(G)$ and $CoAc(G)$ will denote, respectively, the accessible and coaccessible parts of $G$. For two automata $G_1 = (X_1, \Sigma_1, f_1, x_{0_1}, X_{m_1})$ and $G_2 = (X_2, \Sigma_2, f_2, x_{0_2}, X_{m_2})$, their

Manuscript received October 21, 2016; revised December 2, 2016 and December 15, 2016; accepted December 20, 2016. Date of publication December 23, 2016; date of current version October 25, 2017. This work was supported in part by the Brazilian Research Council (CNPq), grant number 310980/2013-5. Recommended by Associate Editor S. Takai.

The authors are with Universidade Federal do Rio de Janeiro, Electrical Engineering Program, 21949-900, Rio de Janeiro, Brazil (e-mail: mvalves@poli.ufrj.br; lilian@dee.ufrj.br; basilio@dee.ufrj.br).

Digital Object Identifier 10.1109/TAC.2016.2644640

product and parallel compositions are denoted by $G_1 \times G_2$ and $G_1 \| G_2$, respectively [21, p. 77]. If $L(G_1) \subseteq L(G_2), L_m(G_1) \subseteq L_m(G_2), f_1(x_{0_1}, s) = f_2(x_{0_2}, s), \forall s \in L(G_1)$, and $X_{m_1} = X_1 \cap X_{m_2}$, then $G_1$ is said to be a subautomaton of $G_2$, denoted as $G_1 \sqsubseteq G_2$. The observer automaton [21, chap. 2] of a deterministic automaton $G$ with respect to $\Sigma_o$, to be denoted by $Obs(G, \Sigma_o)$, is a deterministic automaton whose generated and marked languages are $P_o[L(G)]$ and $P_o[L_m(G)]$, respectively. Throughout the text $X_{obs}$ will denote the set of states of $Obs(G, \Sigma_o)$. An automaton $G$ is said to be state partition if for all $B, B' \in X_{obs}, B \neq B'$, implies that $B \cap B' = \emptyset$ [17]. If an automaton $H$ is not state partition, we can obtain an equivalent state partition automaton $H_{sp} = H \| Obs(H, \Sigma_o)$, such that $L(H_{sp}) = L(H)$ and $L_m(H_{sp}) = L_m(H)$ [18].

A language $K \subseteq L_m(G)$ is observable [22] with respect to $G$ and $P_o$ if, $\forall s, s' \in \Sigma^*$ such that $P_o(s) = P_o(s')$ and $\forall \sigma \in \Sigma$:

$$(s\sigma \in \overline{K}) \wedge (s' \in \overline{K}) \wedge (s'\sigma \in L(G)) \Rightarrow s'\sigma \in \overline{K}. \qquad (1)$$

Given a language $C \subseteq L_m(G)$, a language $K \subseteq C$ is relatively observable[1] with respect to $\overline{C}, G$ and $P_o$, or simply $\overline{C}$-observable, if $\forall s, s' \in \Sigma^*$ such that $P_o(s) = P_o(s')$, and $\forall \sigma \in \Sigma$:

$$(s\sigma \in \overline{K}) \wedge (s' \in \overline{C}) \wedge (s'\sigma \in L(G)) \Rightarrow s'\sigma \in \overline{K}. \qquad (2)$$

Language $\overline{C}$ is referred to as ambient language in the literature.

It has been proved in [9] that relative observability is closed under set unions and, thus, the supremal relatively observable sublanguage always exists. Notice that, from the definition of relative observability, when $C = K$, conditions (1) and (2) become equivalent. However, that does not mean that $K_{sup}$, the supremal $\overline{C}$-observable (or equivalently, $\overline{K}$-observable) sublanguage, is the supremal observable sublanguage of $K$. Finally, it is not difficult to show that, the larger the ambient language, the stronger the relative observability property will be. However, as proved in [9], relative observability is weaker than normality.

## III. An Equivalent Reduced Ambient Language

We will consider in this section the problem of finding a language $C_s \subseteq C$ for which if $K$ is relatively observable with respect to $\overline{C}, G$ and $P_o$, it is also relatively observable with respect to $\overline{C_s}, G$ and $P_o$, and conversely. This result will play a key role in the algorithms proposed later on the technical note for the verification of relative observability and for the computation of the supremal relatively observable sublanguage.

*Lemma 1:* Let $K \subseteq C \subseteq L_m(G)$. Then, $K$ is relatively observable with respect to $\overline{C}, G$ and $P_o$ if, and only if, $K$ is relatively observable with respect to $\overline{C_s} = (\overline{K}\Sigma_{uo}^* \cap \overline{C}), G$ and $P_o$.

*Proof:* ($\Rightarrow$) It is straightforward and comes from the fact that $\overline{C_s} \subseteq \overline{C}$.

($\Leftarrow$) Assume, now, that $K$ is not relatively observable with respect to $\overline{C}, G$ and $P_o$. Then, there exist $s \in \overline{K}, s' \in \overline{C}$ and $\sigma \in \Sigma$ such that $s\sigma \in \overline{K}, s'\sigma \in L(G) \setminus \overline{K}$ and $P_o(s) = P_o(s')$. Without loss of generality, write $s' = s'_p s'_s$, where $s'_p$ is the longest prefix of $s'$ in $\overline{K}$ and $s'_s \in \Sigma^*$. Notice that, $s'_s$ must satisfy one of following conditions: *(i)* $s'_s \in \Sigma_{uo}^*$ or *(ii)* $s'_s \in \Sigma^* \setminus \Sigma_{uo}^*$, *i.e.* $s'_s$ has, at least, one observable event. Let us now consider each one of these possibilities:

i) $s'_s \in \Sigma_{uo}^*$. In this case, $s' = s'_p s'_s \in \overline{K}\Sigma_{uo}^* \cap \overline{C}$. Therefore, $K$ is not relatively observable with respect to $\overline{C_s}, G$ and $P_o$.

ii) $s'_s \in \Sigma^* \setminus \Sigma_{uo}^*$. Without loss of generality, write $s'_s = s'_{sp}\alpha s'_{ss}$, where $s'_{sp} \in \Sigma_{uo}^*, \alpha \in \Sigma_o$ and $s'_{ss} \in \Sigma^*$. Thus, $P_o(s') = P_o(s'_p s'_{sp})\alpha P_o(s'_{ss})$. Since $P_o(s) = P_o(s')$, we can write $s$ as $s = s_p \alpha s_s$, where $P_o(s_p) = P_o(s'_p s'_{sp})$ and $P_o(s_s) = P_o(s'_{ss})$. Defining, now, $t = s_p$ and $t' = s'_p s'_{sp}$, it can be seen that $t\alpha \in \overline{K}, t' \in (\overline{K}\Sigma_{uo}^* \cap \overline{C}), t'\alpha \in L(G) \setminus \overline{K}$ and $P_o(t) = P_o(t')$, which implies that $K$ is not relatively observable with respect to $\overline{C_s}, G$ and $P_o$. ∎

Therefore, in accordance with Lemma 1 instead of considering ambient language $\overline{C}$, we can equivalently consider the reduced ambient language $\overline{C_s} = (\overline{K}\Sigma_{uo}^* \cap \overline{C})$ in all computations regarding relative observability.

## IV. Verification of Relative Observability

An equivalent way to state the relative observability definition given in (2) is as follows: a language $K$ is $\overline{C}$-observable if,

$$(\forall (s, \sigma) \in \overline{K} \times \Sigma)\ s\sigma \in \overline{K} \Rightarrow (\nexists s' \in \overline{C_s})$$
$$\left[(s'\sigma \in L(G) \setminus \overline{K}) \wedge (P_o(s) = P_o(s'))\right]. \qquad (3)$$

Notice that, in Expression (3) $C$ has been replaced with $C_s$, as guaranteed by Lemma 1.

According to conditional statement (3), $\overline{C}$-observability is violated when there exist $s\sigma \in \overline{K}$ and $s'\sigma \in \overline{C_s}\Sigma \cap \overline{K}^C \cap L(G)$ (where $\overline{K}^C$ denotes the complement of $\overline{K}$ with respect to $\Sigma^*$), such that $P_o(s) = P_o(s')$. This observation suggests an algorithm for the verification of relative observability based on the comparison between the projections of languages $\overline{K}$ and $\overline{C_s}\Sigma \cap \overline{K}^C \cap L(G)$, like the algorithm proposed in [23] for the verification of codiagnosability.

Let $\Sigma_R = \{\sigma_R : \sigma \in \Sigma_{uo}\} \cup \Sigma_o$. Define the renaming function $R$, recursively, as follows: $R : \Sigma^* \longrightarrow \Sigma_R^*$, where: *(i)* $R(\sigma) := \sigma$, if $\sigma \in \Sigma_o$, *(ii)* $R(\sigma) := \sigma_R$, if $\sigma \notin \Sigma_o$, and *(iii)* $R(s\sigma) = R(s)R(\sigma)$ for $s \in \Sigma^*$ and $\sigma \in \Sigma$. The inverse renaming function is the mapping $R^{-1} : \Sigma_R^* \longrightarrow \Sigma^*$, where $R^{-1}(s_R) = s$, such that $R(s) = s_R$. Both the renaming and the inverse renaming functions can be extended to languages by applying $R(s)$ and $R^{-1}(s)$ to all strings $s$ in the language. The following result provides the basis for the algorithm we propose here.

*Lemma 2:* Consider automata $G_1 = (X_1, \Sigma, f_1, x_{0_1}, X_{m_1})$ and $G_2 = (X_2, \Sigma, f_2, x_{0_2}, X_{m_2})$, whose event sets are partitioned as $\Sigma = \Sigma_{uo} \dot{\cup} \Sigma_o$. Let $G_m^R := (X_1, R(\Sigma), f_R, x_{0_1}, X_1)$ where, $\forall x \in X_1, f_R(x, R(\sigma)) = f_1(x, \sigma)$, if $f_1(x, \sigma)!$ and, undefined, otherwise, and define $V = G_m^R \| G_2 = (X_v, \Sigma \cup R(\Sigma), f_v, x_{0_v}, X_{m_v})$. Then, for any event $\sigma \in \Sigma$ and a pair of strings $(s_v\sigma, s_v R(\sigma)) \in L_m(V) \times L(V)$ for which $f_v(x_{0_v}, s_v) = (x_1, x_2)$, there exists a pair of strings $(s_1\sigma, s_2\sigma) \in L(G_1) \times L_m(G_2)$ such that $P_o(s_1) = P_o(s_2), f_1(x_{0_1}, s_1) = x_1$ and $f_2(x_{0_2}, s_2) = x_2$, and conversely.

*Proof:* Define projections $P_R : [\Sigma \cup R(\Sigma)]^* \rightarrow R(\Sigma)^*$ and $P_\Sigma : [\Sigma \cup R(\Sigma)]^* \rightarrow \Sigma^*$.

($\Rightarrow$) Assume that there exists a pair of strings $(s_v\sigma, s_v R(\sigma)) \in L_m(V) \times L(V)$ such that $f_v(x_{0_v}, s_v) = (x_1, x_2)$. Notice that $L(V) = P_R^{-1}[L(G_m^R)] \cap P_\Sigma^{-1}[L(G_2)]$ and $L_m(V) = P_R^{-1}[L_m(G_m^R)] \cap P_\Sigma^{-1}[L_m(G_2)]$, since $V = G_m^R \| G_2$. Define $s_{1R} = P_R(s_v)$ and $s_2 = P_\Sigma(s_v)$, then: *(i)* $s_v R(\sigma) \in P_R^{-1}[L(G_m^R)] \Rightarrow P_R[s_v R(\sigma)] \in L(G_m^R) \Rightarrow s_{1R}R(\sigma) \in L(G_m^R)$, and; *(ii)* $s_v\sigma \in P_\Sigma^{-1}[L_m(G_2)] \Rightarrow P_\Sigma(s_v\sigma) \in L_m(G_2) \Rightarrow s_2\sigma \in L_m(G_2)$.

Since $G_m^R$ is obtained from $G_1$ by applying the renaming function $R$ and marking all of its states, it is easy to check that $L(G_m^R) = L_m(G_m^R) = R[L(G_1)]$. Therefore, by defining $s_1 = R^{-1}(s_{1R})$, and

[1]In [9], there is an additional condition for relative observability: $\forall s, s' \in \Sigma^*$ if $P_o(s) = P_o(s')$, then $(s \in K) \wedge (s' \in \overline{C} \cap L_m(G)) \Rightarrow s' \in K$. However, this second condition is necessary only if *marking nonblocking supervisors* are applied; therefore this condition will be omitted in this work.

since $s_{1R}R(\sigma) \in L(G_m^R)$, we have that $s_1\sigma \in L(G_1)$. Notice that, $P_o(s_1) = P_\Sigma(s_{1R})$ and $P_o(s_2) = P_R(s_2)$, and since $s_{1R} = P_R(s_v)$ and $s_2 = P_\Sigma(s_v)$, we can conclude that $P_o(s_1) = P_\Sigma[P_R(s_v)]$ and $P_o(s_2) = P_R[P_\Sigma(s_v)]$. Finally, as $P_\Sigma[P_R(s_v)] = P_R[P_\Sigma(s_v)]$, then $P_o(s_1) = P_o(s_2)$.

Since $G_m^R$ is obtained from $G_1$ by renaming its unobservable events, the renamed unobservable events, $R(\Sigma_{uo})$, and the unobservable events, $\Sigma_{uo}$, become private events of $G_1^R$ and $G_2$, respectively, in the parallel composition $V = G_1^R \| G_2$. Then, by the construction of $V$, it can be seen that, if a transition labeled by an unobservable (resp. a renamed unobservable) event occurs, the first (resp. second) component of state $V$ does not modify. Therefore, we may conclude that $x_1 = f_R(x_{0_1}, s_{1R}) = f_1(x_{0_1}, s_1)$ and $x_2 = f_2(x_{0_2}, s_2)$.

($\Leftarrow$) Take now a pair of strings $(s_1\sigma, s_2\sigma) \in L(G_1) \times L_m(G_2)$ such that $P_o(s_1) = P_o(s_2)$, $f_1(x_{0_1}, s_1) = x_1$ and $f_2(x_{0_2}, s_2) = x_2$. Since $P_o(s_1) = P_o(s_2)$, we have that $s_1$ and $s_2$ are different only in the unobservable events. Therefore, $P_R^{-1}[R(s_1)] \cap P_\Sigma^{-1}(s_2) \neq \emptyset$, which implies that there exists $s_v \in P_R^{-1}[R(s_1)] \cap P_\Sigma^{-1}(s_2)$. Notice that, as $L(G_m^R) = L_m(G_m^R) = R[L(G_1)]$, then $L(V) = P_R^{-1}\{R[L(G_1)]\} \cap P_\Sigma^{-1}[L(G_2)]$ and $L_m(V) = P_R^{-1}\{R[L(G_1)]\} \cap P_\Sigma^{-1}[L_m(G_2)]$. Initially, consider the case when $\sigma \in \Sigma_o$. In this case, it can be seen that $s_v\sigma = s_vR(\sigma) \in P_R^{-1}[R(s_1)R(\sigma)] \cap P_\Sigma^{-1}(s_2\sigma)$. Therefore, as $s_1\sigma \in L(G_1)$, $s_2\sigma \in L_m(G_2)$, then $s_v\sigma = s_vR(\sigma) \in L_m(V)$. Consider, now, the case when $\sigma \in \Sigma_{uo}$. In this case, it is not difficult to see that $s_v\sigma \in P_R^{-1}[R(s_1)] \cap P_\Sigma^{-1}(s_2\sigma)$ and $s_vR(\sigma) \in P_R^{-1}[R(s_1)R(\sigma)] \cap P_\Sigma^{-1}(s_2)$. Therefore, as $s_1 \in L(G_1)$ and $s_2\sigma \in L_m(G_2)$, then $s_v\sigma \in L_m(V)$, and, as $s_1\sigma \in L(G_1)$ and $s_2 \in L(G_2)$, then $s_vR(\sigma) \in L(V)$.

Finally, since the first (resp. second) component of the states of $V$ does not modify if a transition labeled by an event in $\Sigma_{uo}$ (resp. $R(\Sigma_{uo})$) occurs and $P_o(s_1) = P_o(s_2)$, then $f_v(s_{0_v}, s_v) = (x_1, x_2)$. ■

*Algorithm 1:* (Verification of relative observability)

*Inputs:*

- $G = (X_g, \Sigma, f_g, x_{0_g}, X_{m_g})$: automaton whose marked language is $L_m(G)$;
- $A = (X_a, \Sigma, f_a, x_{0_a}, X_{m_a})$: nonblocking automaton whose marked language is $C$;
- $H = (X_h, \Sigma, f_h, x_{0_h}, X_{m_h})$: nonblocking automaton whose marked language is $K$.

*Output:* $K$ is relatively observable wrt $\overline{C}, G$ and $P_o$: true/false.

*Step 1*: Compute automaton $G_m$ by marking all states of $G$, *i.e.* $G_m := (X_g, \Sigma, f_g, x_{0_g}, X_g)$.

*Step 2*: From automaton $A$, construct automaton $M := (X_a \cup \{x_d\}, \Sigma, f_m, x_{0_a}, X_a \cup \{x_d\})$, where *(i)* $\forall (x,\sigma) \in X_a \times \Sigma$, $f_m(x,\sigma) = f_a(x,\sigma)$, if $f_a(x,\sigma)!$ and $f_m(x,\sigma) = x_d$, otherwise; and *(ii)* $f_m(x_d,\sigma)$ is undefined $\forall \sigma \in \Sigma$.

*Step 3*: Compute automaton $M_g := M \times G_m$.

*Step 4*: From automaton $H$, construct automaton $N := (X_h \cup \{x_{d1}, x_{d2}\}, \Sigma, f_n, x_{0_h}, \{x_{d1}, x_{d2}\})$, where *(i)* $\forall (x,\sigma) \in X_h \times \Sigma$: $f_n(x,\sigma) = f_h(x,\sigma)$, if $f_h(x,\sigma)!$, $f_n(x,\sigma) = x_{d1}$, if $((\sigma \in \Sigma_{uo}) \wedge f_h(x,\sigma)\not!)$ and $f_n(x,\sigma) = x_{d2}$, if $((\sigma \in \Sigma_o) \wedge f_h(x,\sigma)\not!)$; *(ii)* $f_n(x_{d1},\sigma) = x_{d1}$, if $\sigma \in \Sigma_{uo}$ and $f_n(x_{d1},\sigma) = x_{d2}$, if $\sigma \in \Sigma_o$; and *(iii)* $f_n(x_{d2},\sigma)$ is undefined $\forall \sigma \in \Sigma$.

*Step 5*: Compute automaton $H_c := CoAc(M_g \times N)$.

*Step 6*: Construct automaton $H_m^R := (X_h, R(\Sigma), f_R, x_{0_h}, X_h)$, where $f_R(x, R(\sigma)) = f_h(x,\sigma)$.

*Step 7*: Compute the verifier automaton $V := H_m^R \| H_c = (X_v, \Sigma \cup \Sigma_R, f_v, x_{0_v}, X_{m_v})$.

*Step 8*: For all $(x,\sigma) \in X_v \times \Sigma$ such that $f_v(x,\sigma) \in X_{m_v}$, verify if the following conditions hold true

(a) $\sigma \in \Sigma_o$;
(b) $(\sigma \notin \Sigma_o) \wedge f_v(x, R(\sigma))!$.

If there exists a transition $f_v(x,\sigma)$ such that either condition *(a)* or *(b)* holds, then $K$ is not relatively observable with respect to $\overline{C}, G$ and $P_o$. Otherwise, $K$ is relatively observable with respect to $\overline{C}, G$ and $P_o$. □

In Step 1 of Algorithm 1, we obtain automaton $G_m$ from $G$ that marks the language generated by $G$, *i.e.*, $L_m(G_m) = L(G)$. In Step 2, we construct automaton $M$ from the nonblocking automaton $A$, whose marked language is $L_m(M) = \overline{C}\Sigma \cup \{\varepsilon\}$. In Step 3, we build automaton $M_g = M \times G_m$ that marks language $(\overline{C}\Sigma \cup \{\varepsilon\}) \cap L(G)$. In Step 4, we construct automaton $N$ from the nonblocking automaton $H$ for which $L_m(N) = \overline{K}\Sigma_{uo}^*\Sigma \cap \overline{K}^C$. In Step 5, we obtain automaton $H_c = M_g \times N$, which has the following property whose proof is straightforward and follows directly from the construction of automaton $H_c$.

*Lemma 3:* $L_m(H_c) = (\overline{K}\Sigma_{uo}^* \cap \overline{C})\Sigma \cap \overline{K}^C \cap L(G)$.

In Step 6, we compute automaton $H_m^R$ by applying the renaming function to the events of $H$ and marking all of its states. In Steps 7 and 8 we construct the verifier automaton $V = H_m^R \| H_c$, and check if $K$ is $\overline{C}$-observable. The following theorem demonstrates the correctness of Algorithm 1.

*Theorem 1:* Let $H, A$ and $G$ denote the automata whose marked languages are, respectively, $K, C$ and $L_m(G)$ such that $K \subseteq C \subseteq L_m(G)$, and consider the verifier automaton $V = (X_v, \Sigma \cup \Sigma_R, f_v, x_{0_v}, X_{m_v})$ computed using Algorithm 1 with the inputs $G, A$ and $H$. Then, $K$ is not relatively observable with respect to $\overline{C}, G$ and $P_o$ if and only if there exists $(x,\sigma) \in X_v \times \Sigma$ that satisfies $f_v(x,\sigma) \in X_{m_v}$ with either $(\sigma \in \Sigma_o)$ or $[(\sigma \notin \Sigma_o) \wedge f_v(x, R(\sigma))!]$.

*Proof:* ($\Rightarrow$) Assume that $K$ is not $\overline{C}$-observable wrt $G$ and $P_o$. Then, according to Lemma 1, $K$ is not $(\overline{K}\Sigma_{uo}^* \cap \overline{C})$-observable wrt $G$ and $P_o$. Therefore, there exist $s \in \overline{K}$, $s' \in (\overline{K}\Sigma_{uo}^* \cap \overline{C})$, and $\sigma \in \Sigma$, such that $P_o(s) = P_o(s')$, $s\sigma \in \overline{K}$ and $s'\sigma \in L(G) \setminus \overline{K}$.

Notice that $s\sigma \in L(H)$, since $L(H) = \overline{K}$. On the other hand, since $s' \in (\overline{K}\Sigma_{uo}^* \cap \overline{C})$ and $s'\sigma \in L(G) \setminus \overline{K}$, it can be concluded that $s'\sigma \in (\overline{C} \cap \overline{K}\Sigma_{uo}^*)\Sigma \cap \overline{K}^C \cap L(G) = L_m(H_c)$. Therefore, from the construction of verifier automaton $V$ and according to Lemma 2, there exists a pair of strings $(s_v\sigma, s_vR(\sigma)) \in L_m(V) \times L(V)$. Finally, defining $x = f_v(x_{0_v}, s_v)$, then $f_v(x,\sigma) \in X_{m_v}$. Moreover, since $s_vR(\sigma) \in L(V)$ and $f_v(x, R(\sigma))!$, it is possible to conclude that either $(\sigma \in \Sigma_o)$ or $[(\sigma \notin \Sigma_o) \wedge f_v(x, R(\sigma))!]$.

($\Leftarrow$) Assume, now, that there exists $(x,\sigma) \in X_v \times \Sigma$ such that $f_v(x,\sigma) \in X_{m_v}$ and $(\sigma \in \Sigma_o) \vee [(\sigma \notin \Sigma_o) \wedge f_v(x, R(\sigma))!]$. Then, from the construction of automaton $V$, there exists $s_v \in L(V)$ such that $f_v(x_{0_v}, s_v) = x$ and $s_v\sigma \in L_m(V)$. Moreover, since $R(\sigma) = \sigma$, if $\sigma \in \Sigma_o$, and $f_v(x, R(\sigma))!$, if $\sigma \notin \Sigma_o$, then $s_vR(\sigma) \in L(V)$. Consequently, according to Lemma 2, there exists $(s\sigma, s'\sigma) \in L(H) \times L_m(H_c)$ such that $P_o(s) = P_o(s')$. Notice that, $s\sigma \in \overline{K}$, $s' \in (\overline{K}\Sigma_{uo}^* \cap \overline{C})$, and $s'\sigma \in L(G) \setminus \overline{K}$, which implies that $K$ is not $(\overline{K}\Sigma_{uo}^* \cap \overline{C})$-observable with respect to $G$ and $P_o$. ■

*Remark 1:* (Verification of language observability) Algorithm 1 can be also applied to verify if a language $K$, marked by a nonblocking automaton $H$, is observable with respect to $G$ and $P_o$, just by making $A = H$. □

*Remark 2:* As it will be shown in Section V, the use of the reduced ambient language is crucial for the application of Algorithm 1 in the computation of the supremal $\overline{C}$-observable sublanguage of a language $K$. However, when we want only to verify if a language $K$ is $\overline{C}$-observable (or observable, by making $C = K$), we can use ambient

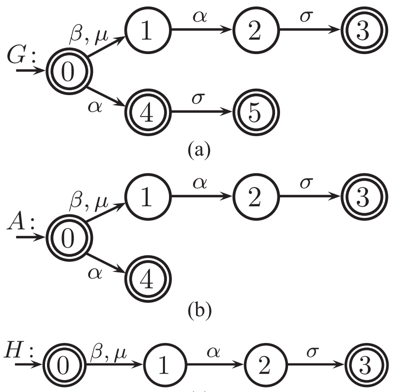
Fig. 1. System automaton $G$ (a), automaton $A$ that marks ambient language $C$ (b), and automaton $H$ whose marked language is $K$ (c).

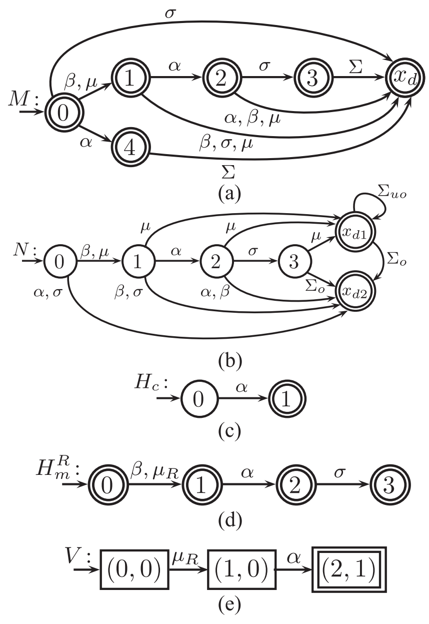
Fig. 2. Automata obtained in Example 1 by Algorithm 1: $M$ (a), $N$ (b), $H_c$ (c), $H_m^R$ (d) and $V$ (e).

language $\overline{C}$ instead of $\overline{C_s}$, and thus, Step 4 of Algorithm 1 should be modified so as to compute automaton $N$ such that $L_m(N) = \overline{K}^C$. However, this change does not improve the computational complexity of the algorithm, since the original and the modified versions of automaton $N$ have $|X_h| + 2$ and $|X_h| + 1$ states, respectively, but with the same number of transition $(|X_h| + 1)|\Sigma|$. ■

The following example illustrates Algorithm 1.

*Example 1:* Consider automata $G, A$ and $H$ depicted in Fig. 1, where $\Sigma = \{\alpha, \beta, \sigma, \mu\}$ and $\Sigma_o = \{\alpha, \beta, \sigma\}$. Automaton $M$, constructed in Step 2, and automaton $N$, constructed in Step 4, are shown in Figs. 2(a) and 2(b), respectively. Automata $H_c$ and $H_m^R$ obtained in Steps 5 and 6, respectively, are depicted in Figs. 2(c,d). Finally, the verifier automaton $V$ computed in Step 7 is depicted in Fig. 2(e), from where, it can be checked that transition $((1,0), \alpha, (2,1))$ satisfies condition *(a)* of Step 8. Therefore, $K$ is not $\overline{C}$-observable with respect to $G$ and $P_o$. Notice that, since $(2,1) \in X_{m_v}$, the following strings can be obtained from $V$: $s = \mu \in \overline{K}, s' = \varepsilon \in \overline{C}, s\alpha = \mu\alpha \in \overline{K}$ and $s'\alpha = \alpha \in L(G) \setminus \overline{K}$, and $P_o(s) = P_o(s') = \varepsilon$. ■

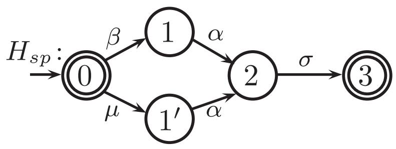
Fig. 3. Automaton $H_{sp} = H \| Obs(H, \Sigma_o)$ obtained from the automaton $H$ depicted in Fig. 1(c).

## V. Computation of the Supremal Relatively Observable Sublanguage

Consider nonblocking automata $A$ and $H$ such that $L_m(A) = C$ and $L_m(H) = K$, and let $f_g, f_a$ and $f_h$ (resp. $x_{0_g}, x_{0_a}$ and $x_{0_h}$) denote the transition functions (resp. initial states) of automata $G, A$ and $H$. We make the following assumptions.

**A1.** For all $s, s' \in L(A)$ such that $f_a(x_{0_a}, s) = f_a(x_{0_a}, s')$, then $f_g(x_{0_g}, s) = f_g(x_{0_g}, s')$.

**A2.** For all $s, s' \in L(H)$ such that $f_h(x_{0_h}, s) = f_h(x_{0_h}, s')$, then $f_a(x_{0_a}, s) = f_a(x_{0_a}, s')$.

Notice that, if $A$ (resp. $H$) does not satisfy Assumption **A1** (resp. **A2**), then another automaton $A$ (resp. $H$) that satisfies Assumption **A1** (resp. **A2**) can be obtained by computing the completely synchronous composition $A \times G$ (resp. $H \times A$). It is worth remarking that Assumptions **A1** and **A2** are less restrictive than assuming that $H$ and $A$ are subautomata of $G$. Finally, when $K = C$, Assumptions **A1** and **A2** reduces to a single one, equivalent to that made in [9].

We will propose an algorithm to obtain a deterministic automaton that marks the supremal $\overline{C}$-observable sublanguage of $K$ with respect to $G$ and $P_o$, whose main idea is to remove, from an automaton that marks $K$, all transitions that correspond to transitions in verifier automaton $V$ that violate the relative observability condition according to Theorem 1 and Algorithm 1. It is worth remarking that not every automaton $H$ has the correct structure to prevent that other strings, besides those which actually violates the relative observability condition, are removed from $K$. This fact is illustrated by the following example.

*Example 2:* Let us consider languages $L(G), C$ and $K$, generated, respectively, by automata $G, A$ and $H$ of Example 1, shown in Figs. 1(a, b, c). In verifier automaton $V$ computed in Example 1, and depicted in Fig. 2(e), transition $((1,0), \alpha, (2,1))$ satisfies condition $(a)$ of Step 8 of Algorithm 1. This is due to string $s = \mu \in \overline{K}$ that violates relative observability since $s' = \varepsilon \in (\overline{K}\Sigma_{uo}^* \cap \overline{C}), P_o(s) = P(s')$ but $s\alpha \in \overline{K}$ and $s'\alpha \in L(G) \setminus \overline{K}$. However, if we remove transition $(1, \alpha, 2)$ of $H$ (Fig. 1(c)), we not only remove string $\mu\alpha$, but also exclude string $\beta\alpha$ which must not be removed. In such case, the calculated supremal relatively observable sublanguage will be $\varepsilon$, which is incorrect. However, if we use the state partition automaton $H_{sp} = H \| Obs(H, \Sigma_o)$, depicted in Fig. 3, we successfully eliminate string $\mu\alpha$ and preserve string $\beta\alpha$ when we exclude transition $(1', \alpha, 2)$; therefore leading to the correct supremal relatively observable sublanguage. ■

Let us define the state partition automaton $H_{sp} = H \| Obs(H, \Sigma_o)$. It is not difficult to see that $L(H_{sp}) = L(H) = \overline{K}, L_m(H_{sp}) = L_m(H) = K$, and that $H_{sp}$ also satisfies Assumption **A2**. Let us define the *uncertainty set* [9] of an automaton $H$ after the occurrence of a string $s \in L(H)$ as: $U_h(s) := \{x \in X_h : (\exists s' \in L(H))[(x = f_h(x_{0,h}, s')) \wedge (P_o(s') = P_o(s))]\}$. The following results can be stated.

*Lemma 4:* Let $H = (X_h, \Sigma, f_h, x_{0_h}, X_{m_h})$ and $H_{sp} = H \| Obs(H, \Sigma_o) = (X_{sp}, \Sigma, f_{sp}, x_{0_{sp}}, X_{m_{sp}})$. Then, $f_{sp}(x_{0_{sp}}, s) = (f_h(x_{0_h}, s), U_h(s)), \forall s \in L(H_{sp})$.

*Proof:* In accordance with the construction of $H_{sp}$, $f_{sp}(x_{0_{sp}}, s) = (f_h(x_{0_h}, s), f_{obs}(x_{0_{obs}}, P_o(s)))$, for all $s \in L(H_{sp})$, where $x_{0_{obs}}$ and $f_{obs}$ denote the initial state and the transition function of $Obs(H, \Sigma_o)$, respectively. Thus, using the definition of the estimate of possible states of $H$ after string $s$ proposed in [24], which is equivalent to the definition of uncertainty set $U_h(s)$, we conclude that $f_{obs}(x_{0_{obs}}, P_o(s)) = U_h(s)$. ∎

*Lemma 5:* Let $H = (X_h, \Sigma, f_h, x_{0_h}, X_{m_h})$, $H_{sp} = H \| Obs(H, \Sigma_o) = (X_{sp}, \Sigma, f_{sp}, x_{0_{sp}}, X_{m_{sp}})$ and $H_s = (X_s, \Sigma, f_s, x_{0_s}, X_{m_s})$ such that $H_s \sqsubseteq H_{sp}$ and $L_m(H_s) = K_s \subseteq K$. Assume that there exist $s, s' \in \overline{K_s}$ and $\sigma \in \Sigma$ such that $s\sigma, s'\sigma \in \overline{K_s}$ and $f_s(x_{0_s}, s) = f_s(x_{0_s}, s')$. If $\exists s'' \in (\overline{K_s}\Sigma_{uo}^* \cap \overline{C})$ such that $P_o(s'') = P_o(s')$ and $s''\sigma \in L(G) \setminus \overline{K_s}$, then $\exists s_c \in \overline{C}$ such that $P_o(s_c) = P_o(s)$ and $s_c\sigma \in L(G) \setminus \overline{K_s}$.

*Proof:* Assume that there exist $s, s' \in \overline{K_s}, \sigma \in \Sigma$ and $s'' \in (\overline{K_s}\Sigma_{uo}^* \cap \overline{C})$ such that $s\sigma, s'\sigma \in \overline{K_s}$, $f_s(x_{0_s}, s) = f_s(x_{0_s}, s')$, $P_o(s'') = P_o(s')$ and $s''\sigma \in L(G) \setminus \overline{K_s}$.

Without loss of generality, write $s''$ as $s'' = s_p'' s_s''$, where $s_p''$ is the longest prefix of $s''$ in $\overline{K_s}$ and $s_s'' \in \Sigma_{uo}^*$. Since $f_s(x_{0_s}, s) = f_s(x_{0_s}, s')$, according to Lemma 4, $U_h(s) = U_h(s') = U_h(s_p'')$, where the last equality is a consequence of the fact that $P_o(s') = P_o(s'') = P_o(s_p'')$. According to the definition of uncertainty set, $f_h(x_{0_h}, s_p'') \in U_h(s_p'')$, and, since $U_h(s_p'') = U_h(s)$, there exists $s_{cp} \in \overline{K}$ such that $P_o(s_{cp}) = P_o(s)$ and $f_h(x_{0_h}, s_{cp}) = f_h(x_{0_h}, s_p'')$. Thus, using Lemma 4, $f_{sp}(x_{0_{sp}}, s_{cp}) = (f_h(x_{0_h}, s_p''), U_h(s_p''))$, *i.e.*, strings $s_p''$ and $s_{cp}$ reach the same state of $H_{sp}$ and also $H_s$, which implies that these strings are continued in $H_s$ with the same strings. Therefore, $s_{cp}s_s''\sigma \notin \overline{K_s}$, since $s_p''$ is not continued with $s_s''\sigma$ in $\overline{K_s}$. Moreover, according to Assumptions **A2** and **A1**, since $s_p''$ and $s_{cp}$ reach the same state of $H$, they also reach the same states of $A$ and $G$, which, together with the fact that $s_p''s_s'' \in \overline{C}$ and $s_p''s_s''\sigma \in L(G)$, imply, respectively, that $s_{cp}s_s'' \in \overline{C}$ and $s_{cp}s_s''\sigma \in L(G)$. Finally, defining $s_c = s_{cp}s_s''$, we have that $s_c \in \overline{C}$, $P_o(s) = P_o(s_c)$ and $s_c\sigma \in L(G) \setminus \overline{K_s}$. ∎

Lemma 5 shows that, when $H, A$ and $G$ satisfy Assumptions **A1** and **A2**, then for an automaton $H_s (H_s \sqsubseteq H_{sp} = H \| Obs(H, \Sigma_o))$ where $L_m(H_s) = K_s$, if a string $s'\sigma \in \overline{K_s}$ violates $(\overline{K_s}\Sigma_{uo}^* \cap \overline{C})$-observability, then all strings $s \in \overline{K_s}$ that reach the state of $H_s$ reached by $s'$ are such that $s\sigma$ also violates $\overline{C}$-observability. As a consequence, if we remove transitions of $H_s$ associated with the strings of $\overline{K_s}$ that violate $(\overline{K_s}\Sigma_{uo}^* \cap \overline{C})$-observability, we only eliminate from $\overline{K_s}$ those strings that violate $\overline{C}$-observability. This fact suggests the following algorithm for computing the supremal $\overline{C}$-observable sublanguage of a language $K$.

*Algorithm 2:* (Computation of the supremal relatively observable sublanguage)

*Inputs:*
- $G = (X_g, \Sigma, f_g, x_{0_g}, X_{m_g})$: automaton whose marked language is $L_m(G)$;
- $A = (X_a, \Sigma, f_a, x_{0_a}, X_{m_a})$: nonblocking automaton whose marked language is $C$;
- $H = (X_h, \Sigma, f_h, x_{0_h}, X_{m_h})$: nonblocking automaton whose marked language is $K$.

*Output:* $H_{sup}$: nonblocking automaton whose marked language is the supremal $\overline{C}$-observable sublanguage of $K$ with respect to $G$ and $P_o$.

*Step 1*: Compute $H_{sp} := H \| Obs(H, \Sigma_o) = (X_{sp}, \Sigma, f_{sp}, x_{0_{sp}}, X_{m_{sp}})$;
*Step 2*: Set $H_s = H_{sp}$;
*Step 3*: Compute verifier automaton $V = (X_v, \Sigma \cup \Sigma_R, f_v, x_{0_v}, X_{m_v})$, by using Algorithm 1 with inputs $G, A$ and $H_s$;
*Step 4*: If $V$ is not an empty automaton, then form the following set:

$$X\Sigma = \{(x_v, \sigma) \in X_v \times \Sigma : (f_v(x_v, \sigma) \in X_{m_v}) \wedge ((\sigma \in \Sigma_o) \vee ((\sigma \notin \Sigma_o) \wedge (f_v(x_v, R(\sigma))!)))\}.$$

*Step 5*: If $X\Sigma \neq \emptyset$, then:
- *5.1:* For all $(x_v, \sigma) \in X\Sigma$, exclude from $H_s$ transition $(x, \sigma, f_s(x, \sigma))$, where $x$ is the state of $H_s$ equal to the first component of $x_v$;
- *5.2:* $H_s \leftarrow Trim(H_s)$;
- *5.3:* Return to Step 3;

*Step 6*: $H_{sup} \leftarrow H_s$. ∎

Notice that in Algorithm 2, after the computation of the state partition automaton $H_{sp}$, we execute Steps 3 to 5 iteratively. For each iteration, we compute the verifier automaton $V$ in Step 3 by using Algorithm 1 with the inputs $G, A$ and $H_s$. In Step 4, we form set $X\Sigma$, which represents all pairs $(x_v, \sigma), x_v \in X_v$ and $\sigma \in \Sigma$, responsible for the loss of relative observability according to Theorem 1. Notice that, when $X\Sigma = \emptyset$, language $L_m(H_s)$ is $\overline{C}$-observable with respect to $G$ and $P_o$. If $X\Sigma \neq \emptyset$, then, according to Lemma 2, for each $(x_v, \sigma) \in X\Sigma$, there exists a string $s$ that reaches state $x$, equal to the first component of $x_v$, and is continued by $\sigma$ such that $s\sigma \in L(H_s)$ violates $\overline{C}$-observability; in Step 5.1, we remove transition $(x, \sigma, f_s(x, \sigma))$ of $H_s$ and, in order to remove possible non-accessible and/or non-coaccessible states, we apply $Trim()$ operator in Step 5.2. When we remove transitions in Step 5.1, it is necessary to verify if the language marked by the new $H_s$ is $\overline{C}$-observable, therefore, after carrying out Step 5.2 we return to Step 3. Notice also that, at each iteration of Algorithm 2, we remove at least one transition from automaton $H_s \sqsubseteq H_{sp}$, which implies that the number of iterations is at most equal to the number of transitions of automaton $H_{sp}$. Therefore, Algorithm 2 terminates in finite steps.

*Theorem 2:* Consider automaton $G$ and nonblocking automata $A$ and $H$ such that $L_m(A) = C, L_m(H) = K$ and $K \subseteq C \subseteq L_m(G)$, and assume that automata $G, A$ and $H$ satisfy Assumptions **A1** and **A2**. Then, automaton $H_{sup}$ obtained by Algorithm 2 with inputs $G, A$ and $H$ marks the supremal $\overline{C}$-observable sublanguage of $K$ with respect to $G$ and $P_o$.

*Proof:* Let $K'_{sup}$ denote the supremal $\overline{C}$-observable sublanguage of $K$ with respect to $G$ and $P_o$. Algorithm 2 finishes when $X\Sigma = \emptyset$. Therefore, according to Theorem 1, $L_m(H_{sup})$ is $\overline{C}$-observable with respect to $G$ and $P_o$, which implies that $L_m(H_{sup}) \subseteq K'_{sup}$. Then, we only need to prove that $K'_{sup} \subseteq L_m(H_{sup})$. The proof will be done using mathematical induction over the (finite) set of iterations in Algorithm 2.

i) *Basis step.* At the beginning of the first iteration, $H_s = H_{sp}$. Therefore, $K'_{sup} \subseteq L_m(H_s) = K$;
ii) *Induction hypothesis.* Suppose that $K'_{sup} \subseteq L_m(H_s)$, up to the beginning of the i-th iteration;
iii) *Inductive step.* Let us now consider the (i+1)-st iteration. Notice that in the i-th iteration, we have removed transitions from $H_s$ in Steps 5.1 and 5.2. Since *Trim* operator applied in Step 5.2 does not modify the marked language of an automaton, strings are only removed from $L_m(H_s)$ in Step 5.1. Thus we need only to analyze Step 5.1.

By using Lemma 5, it can be concluded that for each string $s_m \in L_m(H_s)$ that is removed from $L_m(H_s)$ in Step 5.1, there exist $s \in \overline{\{s_m\}}, s' \in \overline{C}$ and $\sigma \in \Sigma$ such that $s\sigma \in \overline{\{s_m\}}, s'\sigma \in L(G) \setminus L(H_s)$ and $P_o(s) = P_o(s')$, *i.e.*, $s\sigma$ violates $\overline{C}$-observability. In accordance with the induction hypothesis, $K'_{sup} \subseteq L_m(H_s)$ at the beginning of the i-th iteration, and thus, at the beginning of the i-th, $(L(G) \setminus L(H_s)) \subseteq$

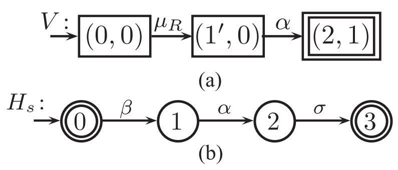

Fig. 4. Automata obtained in the first iteration of Algorithm 2: $V$ (a) and $H_s$ (b).

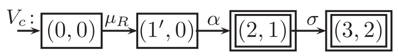

Fig. 5. Verifier obtained with Algorithm 1 by using $\overline{C}$ instead of $\overline{C_s}$.

$(L(G) \setminus \overline{K'_{sup}})$, which implies that $s'\sigma \in L(G) \setminus \overline{K'_{sup}}$, and thus, since $K'_{sup}$ is $\overline{C}$-observable, $s\sigma \notin \overline{K'_{sup}}$, we conclude that $s_m \notin K'_{sup}$. Therefore, all string $s_m$ removed from $L_m(H_s)$ does not belong to $K'_{sup}$,which implies that $K'_{sup} \subseteq L_m(H_s)$ at the beginning of the (i+1)-st iteration.

Finally, since $H_{sup}$ is equal to the $H_s$ obtained in the last iteration of Algorithm 2, then $K'_{sup} \subseteq L_m(H_{sup})$, which concludes the proof. ■

We will now illustrate the application of Algorithm 2.

*Example 3:* Let us consider automata $G, A$ and $H$ of Example 1, shown in Figs. 1(a, b, c). When we apply Algorithm 2 with inputs $G, A$ and $H$, we obtain, in Step 1, automaton $H_{sp}$, which is depicted in Fig. 3, and, in the first iteration, we obtain the verifier automaton $V$ depicted in Fig. 4(a). Therefore, according to Step 4, we obtain $X\Sigma = \{((1', 0), \alpha)\} \neq \emptyset$. This implies that Steps 5.1 to 5.3 must be performed: in Step 5.1 we remove transition $(1', \alpha, 2)$ of $H_s (H_s = H_{sp})$, because this transition is associated with string $\mu\alpha$ that violates $\overline{C}$-observability. In Step 5.2, we exclude state $1'$ of $H_s$ by applying the *Trim* operation, because this state becomes a blocking one after the exclusion of transition $(1', \alpha, 2)$. Finally, at the end of the first iteration, we obtain automaton $H_s$ depicted in Fig. 4(b). In the second iteration, we obtain $X\Sigma = \emptyset$. Therefore automaton $H_{sup}$ is equal to automaton $H_s$, shown in Fig. 4(b), and marks the supremal $\overline{C}$-observable sublanguage of $K$ with respect to $G$ and $P_o$. ■

*Remark 3:* It could be argued that instead of using $\overline{C_s} = \overline{K}\Sigma^*_{uo} \cap \overline{C}$, we could use $\overline{C}$ to compute $V$ in order to identify the transitions that must be removed. However, in doing so, we could eliminate strings that do not violate $\overline{C}$-observability. In order to illustrate this fact consider verifier $V_c$, shown in Fig. 5, computed by using, in Algorithm 1, $\overline{C}$ in the place of $\overline{C_s}$. Notice that, transition $((2, 1), \sigma, (3, 2))$ of $V_c$ satisfies condition $(a)$ in Step 8 of Algorithm 1, since string $s = \mu\alpha \in \overline{K}$ is such that $s\sigma \in \overline{K}$ violates the $\overline{C}$-observability, because $s' = \alpha \in \overline{C}, s'\sigma \in L(G) \setminus \overline{K}$ and $P_o(s) = P_o(s')$. However, if we remove transition $(2, \sigma, 3)$ from $H_{sp}$ (depicted in Fig. 3), we also eliminate string $\beta\alpha\sigma$, which must remain. On the other hand, if we use the reduced ambient language $\overline{C_s}$, string $s\sigma$ does not violate $\overline{C_s}$-observability, but its prefix $s$ does, which leads to the exclusion of transition $(1', \alpha, 2)$, and consequently the removal of string $s\sigma$ from $K$, as expected from the proof of Lemma 1. ■

## VI. Analysis of Computational Complexity of Proposed Algorithms

### A. Computational Complexity of Algorithm 1

Steps 1 to 5 of Algorithm 1 are employed to construct automaton $H_c$ by the product composition of $N, M$ and $G_m$ that have $(|X_h| + 2), (|X_a| + 1)$ and $|X_g|$ states, respectively. Then, $H_c$ has $(|X_h| + 2) \cdot (|X_a| + 1) \cdot |X_g|$ states at most. Since $V = H^R_m \| H_c$ and $H^R_m$ has $|X_h|$ states, then $V$ has, at most, $|X_h| \cdot (|X_h| + 2) \cdot (|X_a| + 1) \cdot |X_g|$ states. The search for transitions of $V$ executed in Step 8 of Algorithm 1 can be done with linear complexity with respect to the number of transitions of $V$, therefore, the computational complexity of Algorithm 1 is equal to $|X_h| \cdot (|X_h| + 2) \cdot (|X_a| + 1) \cdot |X_g| \cdot |\Sigma|$, *i.e.*, $O(|X_h|^2 \cdot |X_a| \cdot |X_g| \cdot |\Sigma|)$.

Consider now the following proposition.

*Proposition 1:* Let $G_1 = (X_1, \Sigma_1, f_1, x_{0_1}, X_{m_1})$ and $G_2 = (X_2, \Sigma_2, f_2, x_{0_2}, X_{m_2})$ such that $L(G_1) \subseteq L(G_2), L_m(G_1) \subseteq L_m(G_2)$. In addition, assume that for every $s, s' \in L(G_1)$ such that $f_1(x_{0_1}, s) = f_1(x_{0_1}, s')$, then $f_2(x_{0_2}, s) = f_2(x_{0_2}, s')$. Construct two automata $G'_1$ and $G'_2$ from $G_1$ and $G_2$ by adding $n_1$ and $n_2$ new states, respectively, and only adding new transitions from the states of $G_1$ and $G_2$ to the new states and between the new states. Then, automaton $G'_1 \times G'_2$ has at most $|X_1| + n_1(|X_2| + n_2)$ states.

*Proof:* In accordance with the construction of $G'_1$ and $G'_2$, we conclude that $(G_1 \times G_2) \sqsubseteq (G'_1 \times G'_2)$ and every state of $G'_1 \times G'_2$ outside $G_1 \times G_2$ has the first component equal to one of the new states added to $G_1$. Therefore, $G'_1 \times G'_2$ has, at most, $|X_{1\times 2}| + n_1(|X_2| + n_2)$ states, where $X_{1\times 2}$ denotes the set of states of automaton $G_1 \times G_2$. Define function $\Theta : X_{1\times 2} \to X_1$ as the mapping $\Theta((x_1, x_2)) = x_1, \forall (x_1, x_2) \in X_{1\times 2}$. Function $\Theta$ is bijective since, for every $s, s' \in L(G_1), f_1(x_{0_1}, s) = f_1(x_{0_1}, s') \Rightarrow f_2(x_{0_2}, s) = f_2(x_{0_2}, s')$ and $L(G_1) \subseteq L(G_2)$. Therefore, $|X_{1\times 2}| = |X_1|$, which concludes the proof. ■

In the verification of observability, $H = A$ and, thus, applying Proposition 1 with $G_1 = H, G_2 = A = H, G'_1 = N$ and $G'_2 = M$, we conclude that automaton $N \times M$ has at most $3|X_h| + 2$ states, and, consequently, automaton $V = H^R_m \| (N \times M \times G_m)$ has $|X_h| \cdot (3|X_h| + 2) \cdot |X_g|$ states, at most. Therefore, the complexity of the verification of language observability by applying Algorithm 1 is $O(|X_h|^2 \cdot |X_g| \cdot |\Sigma|)$, which is equal to the complexity of the algorithm for the verification of observability proposed by [25].

### B. Computational Complexity of Algorithm 2

In the first step of Algorithm 2, we compute automaton $H_{sp}$, that has, at most, $2^{|X_h|} \cdot |X_h|$ states and $2^{|X_h|} \cdot |X_h| \cdot |\Sigma|$ transitions. Since we execute Steps 3 to 5 iteratively, and, for each iteration, at least one transition is removed from automaton $H_{sp}$, then, the number of iterations is at most equal to the number of transitions of $H_{sp}$. In addition, at each iteration, we compute verifier $V$ using Algorithm 1 with inputs $H_s$ (subautomaton of $H_{sp}$), $A$ and $G$. Since $H_s$ and $A$ satisfy Assumption **A1**, by applying Proposition 1 with $G_1 = H_s, G_2 = G'_2 = A$ and $G'_1 = N$, we conclude that automaton $N \times A$ has at most $|X_s| + 2|X_a|$ states, where $|X_s| \leq 2^{|X_h|} \cdot |X_h|$. Moreover, since $A$ and $G$ satisfy Assumption **A2**, by using Proposition 1 with $G_1 = N \times A, G_2 = G'_2 = G_m$ and $G'_1 = N \times M$, we conclude that automaton $H_c = N \times M \times G_m$ has at most $|X_s| + 2(|X_a| + |X_g|)$ states and, consequently, automaton $V$ has, at most, $|X_s|^2 + |X_s| \cdot 2(|X_a| + |X_g|)$ states and, thus, the computational complexity of one iteration of Algorithm 2 is $O([|X_s|^2 + |X_s| \cdot (|X_a| + |X_g|)] \cdot |\Sigma|)$. Therefore, the complexity of Algorithm 2 is $O\left(\left[2^{3|X_h|} \cdot |X_h|^3 + 2^{2|X_h|} \cdot |X_h|^2 \cdot (|X_a| + |X_g|)\right] \cdot |\Sigma|^2\right)$.

When Algorithm 2 is applied for the computation of the supremal $\overline{K}$-observable sublanguage of $K$ with respect to $G$ and $P_o$, then $A = H$, and, thus, the complexity of Algorithm 2 becomes $O\left(\left[2^{3|X_h|} \cdot |X_h|^3 + 2^{2|X_h|} \cdot |X_h|^2 \cdot |X_g|\right] \cdot |\Sigma|^2\right)$, which is smaller than that of the algorithm proposed in [9], that is doubly exponential, *i.e.*, $O(2^{[(2^{|X_h|} \cdot |X_h| + 1)|X_g| + |X_h|]} \cdot |X_h| \cdot |\Sigma|)$.

*Remark 4:* It is important to remark that, when $H$ is already a state partition automaton, *i.e.* when $H_{sp} = H$, the complexity of Algorithm 2 becomes $O\left((|X_h|^3 + |X_h|^2 \cdot |X_g|) \cdot |\Sigma|^2\right)$, being therefore polynomial. Notice that the state partition assumption was made in [9], but the complexity of the algorithm proposed in [9] is still exponential, being $O(2^{(|X_h|+1)|X_g|} \cdot |X_h| \cdot |\Sigma|)$. Therefore, even in this situation, the algorithm proposed here performs better. ∎

## VII. Conclusion

In this technical note, we presented a new property of relative observability which was leveraged in order to derive two new algorithms: the first one for the verification of relative observability and the second one for the computation of the supremal relative observable sublanguage; the former has polynomial time complexity, whereas the latter, although, in general, has exponential complexity, it will have polynomial complexity when the automaton that marks the specification language is state partition.

## References

[1] F. Lin and W. M. Wonham, "On observability of discrete-event systems," *Information Science*, vol. 44, no. 3, pp. 173–198, 1988.

[2] H. Cho and S. I. Marcus, "Supremal and maximal sublanguages arising in supervisor synthesis problems with partial observations," *Mathematical Systems Theory*, vol. 22, no. 1, pp. 177–211, 1989.

[3] H. Cho and S. I. Marcus, "On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation," *Mathematics of Control, Signals and Systems*, vol. 2, no. 1, pp. 47–69, 1989.

[4] J. H. Fa, X. J. Yang, and Y. P. Zheng, "Formulas for a class of controllable and observables sublanguages larger than the supremal controllable and normal sublanguage," *Syst. Control Lett.*, vol. 20, no. 1, pp. 11–18, 1993.

[5] M. Heymann and F. Lin, "On-line control of partially observed discrete event systems," *Discrete Event Dynamic Systems*, vol. 4, no. 3, pp. 221–236, 1994.

[6] N. Hadj-Alouane, S. Lafortune, and F. Lin, "Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation," *Discrete Event Dynamic Systems*, vol. 6, no. 4, pp. 379–427, 1996.

[7] J. Prosser, M. Kam, and H. G. Kwatny, "Online supervisor synthesis for partially observed discrete-event systems," *IEEE Trans. Autom. Control*, vol. 43, no. 11, pp. 1630–1634, Nov. 1998.

[8] S. Takai and T. Ushio, "Effective computation of an $L_m(G)$-closed, controllable, and observable sublanguage arising in supervisory control," *Syst. Control Lett.*, vol. 49, no. 3, pp. 191–200, 2003.

[9] K. Cai, R. Zhang, and W. M. Wonham, "Relative observability of discrete-event systems and its supremal sublanguages," *IEEE Trans. Autom. Control*, vol. 60, no. 3, pp. 659–670, Mar. 2015.

[10] J. Komenda, T. Masopust, and J. H. van Schuppen, "Relative observability in coordination control," *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, Gothenburg, Sweden: IEEE, 2015, pp. 75–80.

[11] K. Cai, R. Zhang, and W. M. Wonham, "On relative observability of timed discrete-event systems," *Preeprints 12th Workshop Discrete Event Syst.*, Cachan, France, 2014, pp. 208–213.

[12] K. Cai, R. Zhang, and W. M. Wonham, "On relative coobservability of discrete-event systems," *Proc. American Control Conf.*, Chicago, IL, USA 2015, pp. 371–376.

[13] K. Cai, R. Zhang, and W. M. Wonham, "Relative observability and coobservability of timed discrete-event systems," *IEEE Trans. Autom. Control*, vol. 61, no. 11, pp. 3382–3395, 2016.

[14] R. Zhang and K. Cai, "On supervisor localization based distributed control of discrete-event systems under partial observation," *Proc. American Control Conf.*, Boston, MA, USA, 2016, pp. 764–769.

[15] K. Cai and W. M. Wonham, "A new algorithm for computing the supremal relatively observable sublanguage," in *Proc. 13th Int. Workshop Discrete Event Syst.*, Xi'an, China, 2016, pp. 8–13.

[16] G. Jirásková and T. Masopust, "On properties and state complexity of deterministic state-partition automata," *International Conference on Theoretical Computer Science*, Amsterdam, The Netherlands, Springer, 2012, pp. 164–178.

[17] J. Komenda, "Supervisory control with partial observations," in *Control Discrete-Event Syst.*, C. Seatzu, M. Silva, and J. van Schuppen, Eds. Springer, 2013, vol. 433, pp. 65–84.

[18] M. Schutzenberger, "On the definition of a family of automata," *Inform. Control*, vol. 4, no. 2–3, pp. 245–270, 1961.

[19] M. V. S. Alves, L. K. Carvalho, and J. C. Basilio, "New algorithms for verification of relative observability and computation of supremal relatively observable sublanguage," in *Proc. IEEE Conf. Control Appl.*, Buenos Aires, Argentina, Sep. 2016, pp. 526–531.

[20] P. J. Ramadge and W. M. Wonham, "The control of discrete-event systems," *Proc. IEEE*, vol. 77, pp. 81–98, 1989.

[21] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. New York: Springer, 2008.

[22] W. M. Whonham, "Supervisory control of discrete-event systems," 2013, http://www.control.utoronto.ca/cgi-bin/dldes.cgi

[23] 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, no. 7, pp. 1679–1684, 2011.

[24] S. Shu, F. Lin, and H. Ying, "Detectability of discrete event systems," *IEEE Trans. Autom. Control*, vol. 52, no. 12, pp. 2356–2359, 2007.

[25] J. N. Tsitsiklis, "On the control of discrete-event dynamical systems," *Math. Control, Signals Syst.*, vol. 2, no. 2, pp. 95–107, 1989.