Conservation of Normality for Master-Slave and Strict Discrete-Event System Composition

Florian Wenck[†] and Jan H. Richter[‡]

Abstract— **In this work the normality property of specification languages for discrete-event systems under partial observation is investigated in a modular context for large-scale systems-modelling. This is related to work on controllability done previously in a similar context [1]. The problem of preserving normality under master-slave (biased synchronous composition, BSC) and strict composition (strict product composition, SPC) operations is defined, and sufficient conditions for preserving normality under application of these operators are provided.**

I. INTRODUCTION

Discrete-event systems (DES) are characterized by the occurrence of discrete state transitions induced by events. The need to treat complex DES by using modular approaches arises for three reasons. First, reduction in computational complexity is desired. Second, for large-scale systems, modular plant modelling is the only feasible approach. Regarding their underlying physical structure, real complex systems in manufacturing should be treated as a composition of subsystems, such as shops or work cells. Finally, sometimes the overall supervisory task needs to be divided into subtasks to run on more than one supervisor.

The best known methods for supervisor synthesis are exponential in the number of modules constituting a global plant. Most likely, no polynomial time methods exist due to the computational complexity. The complexity issue has been examined in detail by Gohari and Wonham [2]. Inspired by this fact, much subsequent research work dealt with the refinement of structured approaches for supervisor design to decrease computational complexity. The original modular approach [3] introduces the concept of modular design to allow for modular specifications, defined on the global plant. Hence the global plant has to be expressed explicitly for the computation of supremal controllable sublanguages. The approach has been refined by de Queiroz and Cury, who introduce the concept of local modularity to avoid the explicit computation of the global plant [4], [5].

Regardless whether the monolithic or a modular approach is used, some of the events generated by the plant cannot be observed by the respective supervisory control. The cause is irrelevant from a theoretical point of view, however in an application-oriented perspective an event can be unobservable for several reasons. First, two events may be indistinguishable to a supervisory control, due to the presence of faulty sensors. Second, due to the lack of sensors, the observation of a particular known existing event (e.g. inside of a melting furnace) is impossible. Such events can be named as *virtual events*. Last, in case of distributed systems, the availability of sensor signals can be restricted to local areas.

Dealing with unobservable events leads to partial observation problems, first proposed by Lin and Wonham [6] and Cieslak et. al. [7]. In addition to the controllability property, observability and normality became important to solve supervisory control problems under partial observation.

Recently, we examined the conservation of controllability from a composition oriented view, providing results for the two composition operators considered here [1]. In that work, a broader perspective regarding composition operators was taken than here. This work is an extension of our previous results on controllability for the property of normality with prior restriction to SPC and BSC.

The notion of normality for specification languages is key to the solution of supervisory control problems under partial observation. Composition operators remain the basis of module-oriented system modelling. The focus of this work is on the conservation of normality under the application of two special compositional operators, namely biased synchronous composition and strict product composition.

The remaining part of this paper is structured as follows. Section II provides a brief review of the foundations for this study. Next, the two relevant composition operators are introduced and explained (Section III). The problem of conserving normality under composition is defined and results are provided for the aforementioned operators (Section IV). An example to illustrate our ideas follows (Section V). The paper concludes with a summary and outlook on future research ideas (Section VI).

II. PRELIMINARIES

In this work, DES are treated using the Supervisory Control Theory (SCT) framework introduced by Wonham et. al. [8]. Plants are modelled as deterministic finite automata (dfa).

Let Σ denote a finite *alphabet* of *events* and the Kleene-closure Σ^* the set of all finite strings, derived by concatenation of events from Σ, including the empty string ε. The *prefix closure* of a language $L \subseteq \Sigma^*$ is denoted by $\overline{L}$.

A plant model dfa is represented by a *generator* $G = (X, \Sigma, \delta, x_0)$ with state set X, event alphabet Σ, partial transition function $\delta : X \times \Sigma \to X$ and initial state x_0.

[†]Florian Wenck is with the Department of Process Automation, Technische Universität Hamburg-Harburg, Schwarzenbergstrasse 95, 21073 Hamburg, Germany wenck@tu-harburg.de

[‡]Jan H. Richter is with the Institute of Automation and Computer Control, Ruhr-Universität Bochum, Universitätsstrasse 150, 44805 Bochum, Germany richter@atp.rub.de

Marker states are not relevant in our context and thus omitted. The automaton generates the regular language $L(G)$, expressing uncontrolled plant behavior. The event alphabet is partitioned into the subsets of *controllable* and *uncontrollable* events $\Sigma = \Sigma_c \cup \Sigma_{uc}$, and into the subsets of *observable* and *unobservable* events $\Sigma = \Sigma_o \cup \Sigma_{uo}$. In general, the two partitions are independent from each other. When DES are used to model technical systems such as manufacturing cells, controllable events are in practice often also observable. Hence the assumption $\Sigma_c \subseteq \Sigma_o$ is valid for most technical systems, an often useful fact. A specification language $K \subseteq L(G)$ represents legal plant behavior.

Let P denote the *natural projection*, which is a map $P : \Sigma^* \to \Sigma_o^*$ deleting all unobservable events from a given string, preserving the order of all remaining events. A full definition of the natural projection is available in [9]. The corresponding *inverse natural projection* $P^{-1} : \Sigma_o^* \to 2^{\Sigma^*}$ of a given string t, produces the set of all strings $T = P^{-1}(t)$ with the property $P(T) = t$. Note that P^{-1} is *not* an inverse function in the sense that $P^{-1}(P(s)) = s$ for $s \in \Sigma^*$. P and P^{-1} are extended to be defined over languages by applying them to all the strings in the considered language.

The standard definition for *normality* of a pair (K,G) with respect to the natural projection P, with $L(G)$ prefix-closed, is used: K is $(L(G),P)$-normal if and only if

$$\overline{K} = P^{-1}[P(\overline{K})] \cap L(G), \tag{1}$$

that is, if $\overline{K}$ can be exactly reconstructed from its natural projection $P(\overline{K})$. In other words: For K to be $(L(G),P)$-normal, $\overline{K}$ must be a union of some subsets of $L(G)$ which each consists of those strings in $L(G)$ that are indistinguishable from each other regarding their natural projection. A graphical illustration of the normality property can be found in [10].

In general, normality is both stricter and better suited for analytic manipulation than *observability*, a related property not presented here. The paper assumes that all controllable events are observable. It is known under this assumption that normality and observability are equivalent, if K is in addition controllable. The reduction of observability to normality is valid in the course of most application problems, as mentioned before. This explains the choice of normality for investigation in this work.

Concerning the *composition* of automata representing DES, only the two operators relevant for this work are presented in the following section. For a more general treatment, the reader is referred to [1] and [11].

III. COMPOSITION OF DES

Fundamentally, composition of DES can be seen as the replacement of several influencing subsystems by a composed system under some given constraints. The resulting composed system must reflect the mutual influence of the subsystems.

In the SCT context, composition means synchronization of deterministic finite automata. Regarding modelling activity, the degree of synchronization of e.g. two dfa depends on how the subsets of *private events* (occurrence of such an event in one subsystem is independent from the other) and *common events* (such events can only occur in both subsystems simultaneously) are preset by the system modeler. This subdivision reflects the influence of the subsystems. Here, this is done by defining the corresponding transition function for each presented composition operator, thus event subdivision is not necessary a-priori. The basis for the definition of all following composition operators are two Generators $G_1 = (X_1, \Sigma_1, \delta_1, x_{01})$ and $G_2 = (X_2, \Sigma_2, \delta_2, x_{02})$ with disjoint state sets $X_1 \cap X_2 = \emptyset$ but generally overlapping event sets. The result of any composition is a generator $G = G_1 ||_{(\cdot)} G_2 = (X, \Sigma, \delta, x_0)$ with the state set $X = X_1 \times X_2$, event set $\Sigma \subseteq \Sigma_1 \cup \Sigma_2$ and initial state $x_0 = (x_{01}, x_{02})$, where $(\cdot)$ is one of $\{$BSC, SPC$\}$, defined below. Each operator is defined by a distinct transition function presented below, with $\sigma \in \Sigma$ a single event, and $x \in X$ a state. The notation $\delta(x,\sigma)!$ denotes that $\delta(x,\sigma)$ is defined. The notation $\neg\delta(x,\sigma)!$ embodies the opposite.

Definition 1 (Biased Synchronous Comp. (BSC)): BSC assigns the two input DES different roles. The generator G_2 is called master, G_1 is called follower.

$$\delta(x,\sigma) = \begin{cases} \delta_1(x_1,\sigma) \times \delta_2(x_2,\sigma) & \text{if } \delta_1(x_1,\sigma)! \\ & \quad \wedge \delta_2(x_2,\sigma)! \\ \{x_1\} \times \delta_2(x_2,\sigma) & \text{if } \neg\delta_1(x_1,\sigma)! \\ & \quad \wedge \delta_2(x_2,\sigma)! \\ \text{undefined} & \text{otherwise.} \end{cases}$$

In BSC, introduced by Lafortune and Chen [12], the master may execute state transitions whenever it can, completely independent from the follower. The follower moves synchronously, if possible. The language generated by G is exactly the master generator language: $L(G) = L(G_2)$ [12]. In [12], Lafortune and Chen introduced the biased synchronous composition in the context of control. Generally, BSC can be used for plant modelling as well.

Consider the model of a simple operator desk for a valve, containing only two push-buttons to open and close the valve, respectively. In addition, consider the model of the valve consisting of two states representing the status (open, close) and two state transitions representing somehow the opening and closing. Defining the operator desk model as master and applying BSC on those subsystems lead to a composed system representation reflecting exactly the characteristic of the push-buttons, namely they can be pushed any time. The case were the valve is already closed and the button to close the valve is pressed, is an instance of a situation where the follower cannot follow its master.

Definition 2 (Strict Product Comp. (SPC)): SPC treats the two input DES symmetrically. An event may occur in the output DES if it can occur simultaneously in both input DES.

$$\delta(x,\sigma) =$$

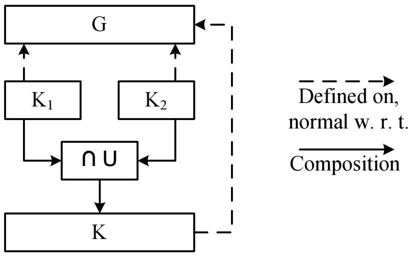


Fig. 1. Established normality check of unions and intersection of normal specifications

$$\begin{cases} \delta_1(x_1,\sigma)\times\delta_2(x_2,\sigma) & \text{if } \delta_1(x_1,\sigma)! \\ & \wedge\,\delta_2(x_2,\sigma)! \\ \text{undefined} & \text{otherwise.}\end{cases}$$

SPC, mentioned in Heymann [13] and often named *Cross Product*, forces the generators G_1, G_2 into complete lock-step operation. The language generated by G is simply the intersection of the source languages: $L(G)=L(G_1)\cap L(G_2)$. It is equivalent to Wonham's meet operator [9] if $\Sigma_1=\Sigma_2$. Note that $\Sigma_1\cap\Sigma_2=\emptyset$ results in an empty generator, thus $\Sigma_1\cap\Sigma_2\neq\emptyset$ is assumed.

With reference to the valve example, the application of SPC on the same subsystems results in a composed system representation not reflecting the above mentioned button characteristics. In this composed system representation the operator desk behaves as a switch.

IV. CONSERVATION OF NORMALITY UNDER COMPOSITION

In this Section, the main results are presented. First, the problem of conserving normality under the application of compositional operators is further specified (Subsections IV-A and IV-B). Sufficient conditions are subsequently provided for the normality of the global plant specification (Subsections IV-C and IV-D).

A. Problem Definition

It is known from the general SCT that the set of all $(L(G),P)$-normal sublanguages $\mathcal{N}(K,L(G))$ of a given language $L(G)$ is closed under arbitrary unions: $(K_1\in\mathcal{N}(K,L(G)))\wedge(K_2\in\mathcal{N}(K,L(G)))\Rightarrow K_1\cup K_2\in\mathcal{N}(K,L(G))$ [9]. The same holds for intersections. In both cases, the original specification languages and their unions and intersections are considered normal with respect to a common, distinct language $L(G)$, representing a monolithic plant. This prior knowledge is depicted in Fig. 1.

Now consider the situation where G is a composed system, consisting of two components ($n=2$). K_1 and K_2 are defined on modules G_1 and G_2, respectively. One possible way of analyzing the normality of unions and intersections of modular specifications in that context would be to directly check the normality of the modular specifications against the composed plant. In case of successful testing the above results about unions and intersections apply immediately. This approach however involves the explicit computation of the composed plant.

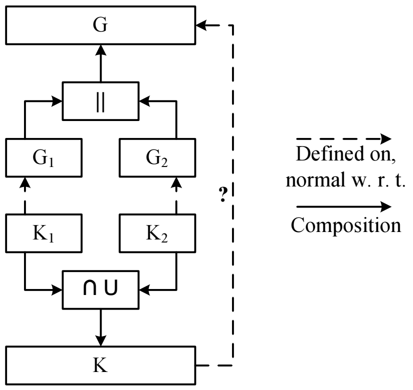


Fig. 2. Illustration of this works key idea

The objective of the approach presented here is to avoid, when possible, the explicit computation of both the composed plant and the combined specification in order to judge the normality of the union or intersection of the modular specifications. For this purpose the following common framework is introduced.

Let G_1, G_2 be generator modules as defined in Section III, and G be the composed generator. Let $K_1\subseteq L(G_1)$, $K_2\subseteq L(G_2)$ be normal specifications local to G_1 and G_2:

$$\overline{K}_1 = P^{-1}[P(\overline{K}_1)]\cap L(G_1) \quad (2)$$
$$\overline{K}_2 = P^{-1}[P(\overline{K}_2)]\cap L(G_2) \quad (3)$$

Let possible combinations of modular specifications be intersection $K_\cap=K_1\cap K_2$ and union $K_\cup=(K_1\cup K_2)\cap L(G)$. Intersecting two different plants' local specifications means that when considered together, they may only perform common tasks. Taking the union respectively means that they may each perform any specified operation individually and independently from each other. The restriction to $L(G)$ for $K_\cup$ is necessary for the specification to be consistent with the context of the composed system.

Problem 1: Let $n=2$ local specifications $K_1\subseteq L(G_1)$, $K_2\subseteq L(G_2)$ be given, each locally normal with respect to their $n=2$ generator modules G_1, G_2 as indicated by Eqns. 2 and 3. Decide if their intersection $K_\cap=K_1\cap K_2$ and union $K_\cup=(K_1\cup K_2)\cap L(G)$ are $(L(G),P)$-normal with respect to the composed plant $G=G_1||_{(\cdot)}G_2$, with the composition operator $||_{(\cdot)}\in\{BSC,SPC\}$.

This problem is illustrated in Fig. 2 to support the intuitive understanding of the problem.

Some notes on the underlying natural projection P together with a lemma necessary for subsequent proofs follow.

B. The Natural Projection P

Some observations concerning the natural projection P are necessary. The same natural projection is assumed to be valid for both plant modules. This implies that the modular plants have to agree on the observability of common events. Mixtures, as they were allowed for the deduction of the global controllable event subalphabet in [1], are not possible for our approach to analyzing normality under composition, due to the use of a common natural projection.

Formally, this amounts to $\Sigma_o \subseteq (\Sigma_1 \cup \Sigma_2)$ and $P : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_o^*$.

The following lemma is used in subsequent proofs as well as a property relative to the inverse natural projection mentioned afterwards. Otherwise, all subsequent proofs are based on basic set operation properties like distributivity of concatenation, union and intersection and the use of identities like $\overline{K_1 \cap K_2} \subseteq \overline{K}_1 \cap \overline{K}_2$.

Lemma 1 (Supporting Lemma): Let $A \subseteq \Sigma_1^*$, $B \subseteq \Sigma_2^*$ be languages on the alphabets Σ_1, Σ_2. Let $\Sigma_o \subseteq (\Sigma_1 \cup \Sigma_2)$ and $P : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_o^*$ the corresponding natural projection. Then

$$P(A \cap B) \subseteq P(A) \cap P(B). \tag{4}$$

Proof: Since $A \cap B \subseteq A$, then $P(A \cap B) \subseteq P(A)$. Similarly, $A \cap B \subseteq B$ leads to $P(A \cap B) \subseteq P(B)$. Thus

$$P(A \cap B) \subseteq P(A) \cap P(B)$$

■

The following property is taken from de Queiroz and Cury [4].

Property 1 (Inv. Natural Projection Identity): Let $A,B \subseteq \Sigma_o^*$ be languages on the alphabet Σ_o and let $P : \Sigma^* \to \Sigma_o^*$ denote the natural projection related to Σ_o^*. Then

$$P^{-1}(A) \cap P^{-1}(B) = P^{-1}(A \cap B). \tag{5}$$

Proof: See [4]. ■

For subsequent proofs it should be noted that $P^{-1}[P(\overline{K})] \cap L(G) \supseteq \overline{K}$ is a tautology.

Sufficient conditions to ensure the normality of the combined specifications for the SPC and BSC operators follow.

C. Results On Strict Product Composition (SPC)

As mentioned in Definition 2, $G = G_1 ||_{\mathrm{SPC}} G_2$, $L(G) = L(G_1) \cap L(G_2)$.

Proposition 1 ($K_\cap = K_1 \cap K_2$ for SPC): K is $(L(G),P)$-normal ($\overline{K} = P^{-1}[P(\overline{K})] \cap L(G)$) if

1) $\overline{K_1 \cap K_2} = \overline{K}_1 \cap \overline{K}_2$, i. e. K_1, K_2 are nonconflicting

Proof:

$$\begin{aligned}
& P^{-1}[P(\overline{K})] \cap L(G) \\
= \;& P^{-1}[P(\overline{K_1 \cap K_2})] \cap L(G_1) \cap L(G_2) \\
= \;& P^{-1}[P(\overline{K}_1 \cap \overline{K}_2)] \cap L(G_1) \cap L(G_2) \\
& \text{if } K_1,\ K_2 \text{ nonconflict} \\
\subseteq \;& P^{-1}[P(\overline{K}_1) \cap P(\overline{K}_2)] \cap L(G_1) \cap L(G_2) \\
= \;& P^{-1}[P(\overline{K}_1)] \cap P^{-1}[P(\overline{K}_2)] \cap L(G_1) \cap L(G_2) \\
= \;& P^{-1}[P(\overline{K}_1)] \cap L(G_1) \cap P^{-1}[P(\overline{K}_2)] \cap L(G_2) \\
= \;& \overline{K}_1 \cap \overline{K}_2 \\
= \;& \overline{K_1 \cap K_2}, \text{ if } K_1,\ K_2 \text{ nonconflict} \\
= \;& \overline{K}.
\end{aligned} \tag{6}$$

The condition in Propos. 1 is used for the second and the last but one equation. Lemma 1 is used for the third step. The fourth step requires Property 1. $P(\overline{K_1})$,$P(\overline{K_2}) \subseteq \Sigma_o^*$, due to the agreement on observable common events mentioned before. All other steps are based on well-known identities. ■

The nonconflicting condition of Propos. 1 unsurprisingly requires that the modular specifications do not lead to contradicting behavior in the context of the composed plant.

Proposition 2 ($K_\cup = (K_1 \cup K_2) \cap L(G)$ for SPC): K is $(L(G),P)$-normal ($\overline{K} = P^{-1}[P(\overline{K})] \cap L(G)$) if

1) $L(G) = \overline{L(G)}$, i. e. $L(G)$ is prefix-closed
2) $\overline{K_1 \cap L(G)} = \overline{K}_1 \cap \overline{L}(G)$, i. e. K_1, $L(G)$ are nonconflicting
3) $\overline{K_2 \cap L(G)} = \overline{K}_2 \cap \overline{L}(G)$, i. e. K_2, $L(G)$ are nonconflicting

Note that K can be written as $K = (K_1 \cap L(G)) \cup (K_2 \cap L(G))$ and condition 1 of Propos. 2 is trivial since $L(G)$ is always prefix-closed by its definition as a generated language of G.

Proof:

$$\begin{aligned}
& P^{-1}[P(\overline{K})] \cap L(G) \\
= \;& P^{-1}[P(\overline{(K_1 \cap L(G)) \cup (K_2 \cap L(G))})] \\
& \cap L(G_1) \cap L(G_2) \\
= \;& P^{-1}[P(\overline{K_1 \cap L(G)} \cup \overline{K_2 \cap L(G)})] \\
& \cap L(G_1) \cap L(G_2) \\
= \;& P^{-1}[P(\overline{K_1 \cap L(G)}) \cup P(\overline{K_2 \cap L(G)})] \\
& \cap L(G_1) \cap L(G_2) \\
= \;& (P^{-1}[P(\overline{K_1 \cap L(G)})] \cup P^{-1}[P(\overline{K_2 \cap L(G)})]) \\
& \cap L(G_1) \cap L(G_2) \\
= \;& (P^{-1}[P(\overline{K_1 \cap L(G)})] \cap L(G_1) \cap L(G_2)) \\
& \cup (P^{-1}[P(\overline{K_2 \cap L(G)})] \cap L(G_1) \cap L(G_2)) \\
\subseteq \;& (\overline{K}_1 \cap L(G_1) \cap L(G_2)) \\
& \cup (\overline{K}_2 \cap L(G_1) \cap L(G_2) \\
= \;& (\overline{K}_1 \cap L(G)) \cup (\overline{K}_2 \cap L(G)) \\
= \;& (\overline{K}_1 \cap \overline{L(G)}) \cup (\overline{K}_2 \cap \overline{L(G)}) \\
& \text{if } L(G) \text{ is prefix-closed} \\
= \;& \overline{(K_1 \cap L(G)) \cup (K_2 \cap L(G))} \\
& \text{if } (L(G),K_{\mathrm{i}}) \text{ nonconflict, } i = 1,2 \\
= \;& \overline{(K_1 \cup K_2) \cap L(G)} \\
= \;& \overline{K}.
\end{aligned} \tag{7}$$

Together with the aforementioned tautology, this establishes $(L(G),P)$-normality for K. The conditions of Propos. 2 are used in steps 8 and 9. ■

The nonconflicting test in Propos. 1 involves the language K explicitly, however not the global plant. The nonconflicting test in Propos. 2 involves the language $L(G)$ explicitly, however not the combined specification. Both propositions have a benefit: The computation of K for intersection is easy, and the global plant need not be computed. For union, the computation of the global plant is necessary but the combined specification need not be computed in addition.

The result is not surprising in the light of the intuitive interpretation of the nonconflicting conditions on both modular specifications.

D. Results On Biased Synchronous Composition (BSC)

Here, $G = G_1 ||_{\mathrm{BSC}} G_2$, $L(G) = L(G_2)$ by Definition 1.

Proposition 3 ($K_\cap = K_1 \cap K_2$ for BSC): K is $(L(G),P)$-normal ($\overline{K} = P^{-1}[P(\overline{K})] \cap L(G)$) if

1) $P^{-1}[P(\overline{K}_1)] \cap L(G_2) \subseteq \overline{K}_1$, i. e. K_1 is $(L(G_2),P)$-normal.
2) $\overline{K_1 \cap K_2} = \overline{K}_1 \cap \overline{K}_2$, i. e. K_1, K_2 nonconflict

Proof:

$$\begin{array}{rc}
 & P^{-1}[P(\overline{K})] \cap L(G) \qquad (8)\\
= & P^{-1}[P(\overline{K_1 \cap K_2})] \cap L(G_2)\\
= & P^{-1}[P(\overline{K}_1 \cap \overline{K}_2)] \cap L(G_2)\\
 & \text{if } K_1,\ K_2 \text{ nonconflict}\\
\subseteq & P^{-1}[P(\overline{K}_1) \cap P(\overline{K}_2)] \cap L(G_2)\\
= & P^{-1}[P(\overline{K}_1)] \cap P^{-1}[P(\overline{K}_2)] \cap L(G_2)\\
= & P^{-1}[P(\overline{K}_1)] \cap L(G_2) \cap P^{-1}[P(\overline{K}_2)] \cap L(G_2)\\
\subseteq & \overline{K}_1 \cap \overline{K}_2 \text{ if } K_1 \text{ is } (L(G_2),P)\text{-normal}\\
= & \overline{K_1 \cap K_2} \text{ if } K_1,\ K_2 \text{ nonconflict}\\
= & \overline{K}.
\end{array}$$

The steps rely on the same basic identities and Lemma 1 as for Propos. 1 and 2, except for step 6, which requires the new condition 1 of Propos. 3. This condition requires that the follower specification be normal with respect to the master plant. The nonconflicting condition intuitively requires that both specifications do not lead to a contradiction for the composed plant. ■

Proposition 4 ($K_\cup = (K_1 \cup K_2) \cap L(G)$ for BSC): K is $(L(G),P)$-normal ($\overline{K} = P^{-1}[P(\overline{K})] \cap L(G)$) if

1) $P^{-1}[P(\overline{K}_1)] \cap L(G_2) \subseteq \overline{K}_1$, i. e. K_1 is $(L(G_2),P)$-normal.
2) $\overline{K_1 \cap L(G_2)} = \overline{K}_1 \cap \overline{L(G_2)}$, i. e. K_1, $L(G_2)$ nonconflict
3) $\overline{K_2 \cap L(G_2)} = \overline{K}_2 \cap \overline{L(G_2)}$, i. e. K_1, $L(G_2)$ nonconflict

Note for the following proof that here $K = (K_1 \cap L(G)) \cup (K_2 \cap L(G))$.

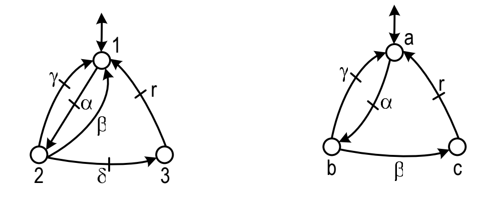

(a) Generator G_1 (b) Generator G_2

Fig. 3. Plant modules used to obtain the composed plant

Proof:

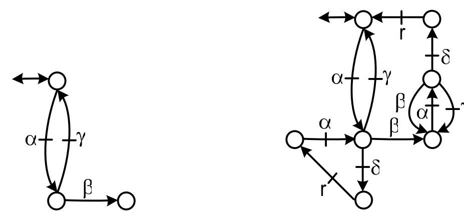
$$\begin{array}{rc}
 & P^{-1}[P(\overline{K})] \cap L(G) \qquad (9)\\
= & P^{-1}[P(\overline{(K_1 \cap L(G_2)) \cup (K_2 \cap L(G_2))})]\\
 & \cap L(G_2)\\
= & P^{-1}[P(\overline{K_1 \cap L(G_2)} \cup \overline{K_2 \cap L(G_2)})] \cap L(G_2)\\
= & (P^{-1}[P(\overline{K_1 \cap L(G_2)})]\\
 & \cup P^{-1}[P(\overline{K_2 \cap L(G_2)})]) \cap L(G_2)\\
\subseteq & (P^{-1}[P(\overline{K_1} \cap \overline{L(G_2)})] \cap L(G_2))\\
 & \cup (P^{-1}[P((\overline{K_2} \cap \overline{L(G_2)})] \cap L(G_2))\\
\subseteq & (\underbrace{P^{-1}[P(\overline{K}_1)] \cap L(G_2)}_{\overset{!}{\subseteq}\overline{K}_1} \cap \underbrace{P^{-1}[P(\overline{L(G_2)})] \cap L(G_2)}_{\subseteq \overline{L(G_2)}})\\
\cup & (\underbrace{P^{-1}[P(\overline{K}_2)] \cap L(G_2)}_{=\overline{K}_2} \cap \underbrace{P^{-1}[P(\overline{L(G_2)})] \cap L(G_2)}_{\subseteq \overline{L(G_2)}})\\
\subseteq & (\overline{K}_1 \cap \overline{L(G_2)}) \cup (\overline{K}_2 \cap \overline{L(G_2)})\\
= & \overline{(K_1 \cap L(G_2)) \cup (K_2 \cap L(G_2))}\\
= & \overline{(K_1 \cap K_2) \cap L(G_2)}\\
= & \overline{K}.
\end{array}$$

In addition to the requirement that K_1 be $(L(G_2),P)$-normal, both modular specifications must nonconflict with the master plant language $L(G_2)$ according to step 8, conditions 2 and 3 of Propos. 4. ■

V. ILLUSTRATIVE EXAMPLE

The ideas developed above are now illustrated by an example. This example uses the same automata as in [1]. Consider two plant generators $G_1 = (X_1, \Sigma_1, \delta_1, x_{01})$, $G_2 = (X_2, \Sigma_2, \delta_2, x_{02})$ with $\Sigma_1 = \{\alpha, \beta, \gamma, \delta, r\}$, $\Sigma_2 = \{\alpha, \beta, \gamma, r\}$, $\Sigma_{\mathrm{uo}} = \{\beta\}$, state set and transition structures as in Fig. 3. The corresponding natural projection is $\{\alpha, \beta, \gamma, \delta, r\}^* \rightarrow \{\alpha, \gamma, \delta, r\}^*$. Given are locally normal specifications $K_1 = \{\alpha\beta, \alpha\delta r\}$ and $K_2 = \{\alpha\beta, \alpha\gamma\}$. This results in $K_\cap = K_1 \cap K_2 = \{\alpha\beta\}$, $K_\cup = K_1 \cup K_2 = \{\alpha\beta, \alpha\gamma, \alpha\delta r\}$.

The composed generator G_{SPC} is shown in Fig. 4(a). Observe that G_{SPC} is blocking, but would not be so if state c of G_2 were marked. The intersected specification $K_\cap = K_1 \cap K_2 = \{\alpha\beta\}$ is $(L(G_{\mathrm{SPC}}),P)$-normal if the condition of Propos. 1 holds: 1 – (1) K_1, K_2 are nonconflicting as $\overline{K}_1 \cap \overline{K}_2 = \{\varepsilon, \alpha, \alpha\beta, \alpha\delta, \alpha\delta r\} \cap \{\varepsilon, \alpha, \alpha\beta, \alpha\gamma\} = \{\varepsilon, \alpha, \alpha\beta\} = \overline{\{\alpha\beta\}} = \overline{K_1 \cap K_2}$ ✓. Since K_1, K_2 are each



(a) Generator G_{SPC} (b) Generator G_{BSC}

Fig. 4. Automata graphs of composed plants

$(L(G_{\mathrm{i}}),P)$-normal w. r. t. their own plant i, by Propos. 1 their intersection $K_\cap$ is $(L(G_{\mathrm{SPC}}),P)$-normal. This is easily verified by inspection of Fig. 4(a). Note that the composed plant was not used for this decision.

For $K_\cup = (K_1 \cup K_2) \cap L(G_{\mathrm{SPC}}) = \{\alpha\beta, \alpha\gamma\}$, Propos. 2 is relevant. Condition 2 – (1) is naturally met, because $L(G_{\mathrm{SPC}})$ is generated by an automaton. By condition 2 – (2) $(K_1, L(G_{\mathrm{SPC}}))$ is nonconflicting because $\overline{K}_1 \cap \overline{L}(G_{\mathrm{SPC}}) = \{\varepsilon,\alpha,\alpha\beta,\alpha\delta,\alpha\delta r\} \cap \overline{L}(G_{\mathrm{SPC}}) = \{\varepsilon,\alpha,\alpha\beta\} = \overline{\{\alpha\beta,\alpha\delta r\} \cap L(G_{\mathrm{SPC}})} = \overline{K_1 \cap L(G_{\mathrm{SPC}})}$ ✓. 2 – (2) $(K_2, L(G_{\mathrm{SPC}}))$ is nonconflicting because $\overline{K}_2 \cap \overline{L}(G_{\mathrm{SPC}}) = \{\varepsilon,\alpha,\alpha\beta,\alpha\gamma\} \cap \overline{L}(G_{\mathrm{SPC}}) = \{\varepsilon,\alpha,\alpha\beta,\alpha\gamma\} = \overline{\{\alpha\beta,\alpha\gamma\} \cap L(G_{\mathrm{SPC}})} = \overline{K_2 \cap L(G_{\mathrm{SPC}})}$ ✓. $K_\cup$ is $(L(G_{\mathrm{SPC}}),P)$-normal, which is easily verified by inspection. Observe that in this case, the composed plant language was required for the last two conditions, but the combined specification language was not used.

Finally the composed generator G_{BSC}, with G_1 master and G_2 follower, is shown in Fig. 4(b). Checking the conditions of Propos. 3 yields: 3 – (1) $P^{-1}[P(\overline{K}_2)] \cap L(G_1) = \{\varepsilon,\alpha,\alpha\beta,\alpha\gamma\} \subseteq \{\varepsilon,\alpha,\alpha\beta,\alpha\gamma\}$ ✓, 3 – (2) the same as in 1 – (1). Since K_1, K_2 are each $(L(G_{\mathrm{i}}),P)$-normal w. r. t. their own plant i, by Propos. 3, their intersection $K_\cap$ is $(L(G_{\mathrm{BSC}}),P)$-normal, which is easily verified by inspection of Fig. 4(b) as well.

When $K_\cup = (K_1 \cup K_2) \cap L(G_{\mathrm{BSC}}) = \{\alpha\beta,\alpha\gamma,\alpha\delta r\}$, Propos. 4 is applied, with condition 4 – (1) the same as in Proposition 3. 4 – (2) $(K_2, L(G_{\mathrm{BSC}}))$ is nonconflicting because $\overline{K}_2 \cap \overline{L}(G_1) = \{\varepsilon,\alpha,\alpha\beta,\alpha\gamma\} \cap \overline{L}(G_1) = \{\varepsilon,\alpha,\alpha\beta,\alpha\gamma\} = \overline{\{\alpha\beta,\alpha\gamma\}} = \overline{K_2 \cap L(G_1)}$ ✓. 4 – (3) $(K_1, L(G_{\mathrm{BSC}}))$ is nonconflicting because $\overline{K}_1 \cap \overline{L}(G_1) = \{\varepsilon,\alpha,\alpha\beta,\alpha\delta,\alpha\delta r\} \cap \overline{L}(G_1) = \{\varepsilon,\alpha,\alpha\beta,\alpha\delta,\alpha\delta r\} = \overline{\{\alpha\beta,\alpha\delta r\}} = \overline{K_1 \cap L(G_1)}$ ✓. Thus, $K_\cup$ is $(L(G_{\mathrm{BSC}}),P)$-normal, as easily verified using Fig. 4(b).

VI. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

The notion of normality was investigated within the context of composed systems. Sufficient conditions on the conservation of normality under the application of two compositional operators of master-slave and strict type were presented and illustrated by an example. The approach extends prior work on controllability to partial observation problems where all controllable events are observable.

B. Future Works

Directions for further research include the analysis of other compositional operators mentioned in [1] with focus on the most general operator Prioritized Synchronous Composition (PSC). The derivation of conditions for all remaining operators, which are specializations of PSC, from the results on PSC should be possible.

Furthermore, the quantification of the computational benefit of the presented methods and expansion to multi-component systems ($n > 2$) remain to be investigated in detail.

In addition, the conditions found should be weakened from sufficient to necessary and sufficient conditions.

Finally, investigations regarding the conservation of other properties related to supervisory control design and DES analysis under composition, such as nonblocking, separability or optimality, represent potentially useful expansion of this ongoing line of research.

REFERENCES

[1] F. Wenck and J. H. Richter, "A Composition Oriented Perspective on Controllability of Large Scale DES," in *Proceedings of the 7th International Workshop on Discrete Event Systems (WODES)*, Reims, France, September 2004.

[2] P. Gohari and W. M. Wonham, "On the Complexity of Supervisory Control Design in the RW Framework," *IEEE Trans. Syst., Man, Cybern. B*, vol. 30, no. 5, pp. 643–652, October 2000.

[3] P. J. Ramadge and W. M. Wonham, "Modular Supervisory Control of Discrete-Event Systems," *Mathematics of Control, Signals and Systems*, vol. 1, no. 1, pp. 13–30, 1988.

[4] M. H. de Queiroz and J. E. R. Cury, "Modular Control of Composed Systems," in *Proceedings of the American Control Conference*, Chicago, USA, June 2000.

[5] ——, "Modular Supervisory Control of Large Scale Discrete Event Systems," in *Discrete Event Systems: Analysis and Control*, R. Boel and G. Stremersch, Eds. Kluwer Academic Publishers, 2000, pp. 103–110.

[6] F. Lin and W. M. Wonham, "On Observability of Discrete-event Systems," *Information Sciences*, vol. 44, no. 3, pp. 173–198, 1988.

[7] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya, "Supervisory Control of Discrete Event Processes with Partial Observation," *IEEE Trans. Automat. Contr.*, vol. 33, pp. 249–260, March 1988.

[8] P. J. Ramadge and W. M. Wonham, "The Control of Discrete Event Systems," *Proc. IEEE, Special Issue on Discrete Event Dynamic Systems*, vol. 77, no. 1, pp. 81–98, January 1989.

[9] W. M. Wonham, *Notes on Control of Discrete-Event Systems*. Dept. of Elec. and Comp. Eng., University of Toronto, Toronto, Canada, 2002.

[10] R. Kumar and V. K. Garg, *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic Publishers, 1995.

[11] S. Kowalewski, *Modulare diskrete Modellierung verfahrenstechnischer Anlagen zum systematischen Steuerungsentwurf.* Shaker Verlag, Aachen, Germany, 1996.

[12] S. Lafortune and E. Chen, "The Infimal Closed Controllable Superlanguage and Its Application in Supervisory Control," *IEEE Trans. Automat. Contr.*, vol. 35, no. 4, pp. 398–405, April 1990.

[13] M. Heymann, "Concurrency and Discrete Event Control," *IEEE Control Syst. Mag.*, vol. 10, no. 4, pp. 103–112, June 1990.