Nondeterminism Avoidance in Compositional Synthesis of Discrete Event Systems

Sahar Mohajerani* Robi Malik† Martin Fabian*

*Department of Signals and Systems
Chalmers University of Technology, Gothenburg, Sweden
Email: {mohajera,fabian}@chalmers.se

†Department of Computer Science
University of Waikato, Hamilton, New Zealand
Email: robi@cs.waikato.ac.nz

***Abstract*—This paper proposes a framework for compositional synthesis of least restrictive controllable and nonblocking supervisors for modular discrete event systems models. The problem of state-space explosion is mitigated by abstracting individual components using synthesis abstraction before computing too large synchronous products. The paper improves and generalises previous work by introducing renaming to avoid nondeterministic intermediate results, making it possible to use more means of abstraction. Four classes of abstraction rules are discussed in the generalised framework, and an example demonstrates the feasibility of the method for practical problems.**

I. Introduction

Supervisory control theory [1] provides a general framework to compute least restrictive control strategies to control a given *plant* such that its behaviour satisfies a given *specification*. Synthesis for systems with a large number of components is impeded by an inherent complexity problem known as *state-space explosion*. To overcome the problem of state-space explosion and also to find more comprehensible supervisors, it is desirable to reduce the complexity using *abstraction*. However, abstraction of automata often introduces nondeterminism, making it difficult to apply standard synthesis methods. This paper proposes a method of abstraction that avoids nondeterminism, making it possible to use more abstraction techniques for supervisor synthesis.

Supervisory control theory is generalised for nondeterministic models in [2]–[4]. In [2], [3], even though the plant may be nondeterministic, the specification must be deterministic. This condition is further relaxed in [4], where the plant and specification can be nondeterministic. All of these works synthesise deterministic or nondeterministic supervisors for nondeterministic systems. In contrast, this paper seeks to find a deterministic supervisor for a deterministic plant and specification.

Abstraction is used in [5]–[11] to remove unnecessary information and reduce the size of systems. [5], [6] use *natural projection*, while [7] uses *weak observation equivalence* for abstraction. The works [8], [9] propose automata equivalences tailored for supervisor synthesis, called *supervision equivalence* and *synthesis equivalence* respectively. In [7]–[9], synthesis is considered in a nondeterministic setting, which leads to some problems when interpreting results and ensuring maximal permissiveness. These problems are overcome to some extent in [10], [11], where *synthesis abstraction* is used to abstract automata. The methods of [5], [6], [10], [11] require all automata and their abstraction results to be deterministic, which makes some desirable abstraction impossible.

This paper adopts the idea of *distinguishing sensors* to avoid such nondeterminism. Distinguishing sensors are proposed in [12] as a modelling aid. By renaming certain events using different names in different contexts, it is possible to create models that are more suitable for synthesis. In this paper, nondeterministic automata are made deterministic by renaming. This improves the scope of abstraction over previous compositional synthesis methods [10], [11], and gives rise to a fully automatic method to synthesise nonblocking and least restrictive modular supervisors.

This paper is organised as follows. First, Sect. II introduces the relevant notation. Then Sect. III presents a framework of compositional synthesis, describes possible means abstraction and the use of renaming to avoid nondeterminism. Afterwards, Sect. IV applies the approach to a practical example, and Sect. V adds some concluding remarks.

II. Preliminaries and Notation

A. Events and Languages

Discrete event systems are modelled using events and languages [1]. Events are taken from a finite alphabet Σ, which is partitioned into two disjoint subsets, the set Σ_c of *controllable* events and the set Σ_u of *uncontrollable* events. The special event $\omega \in \Sigma_c$ denotes *termination*.

The set of all finite strings of elements of Σ, including the *empty string* ε, is denoted by Σ^*. A subset $L \subseteq \Sigma^*$ is called a *language*. The concatenation of two strings $s, t \in \Sigma^*$ is written as st. For $\Omega \subseteq \Sigma$, the *natural projection* $P_\Omega \colon \Sigma^* \to \Omega^*$ is the operation that removes from strings $s \in \Sigma^*$ all events not in Ω.

B. Nondeterministic Automata

This paper models discrete event systems using nondeterministic automata, in order to show how nondeterminism arises during abstraction and how it can be avoided.

Definition 1: A (nondeterministic) finite-state automaton is a tuple $G = \langle \Sigma, Q, \to, Q^\circ \rangle$, where Σ is a finite set of events, Q is a finite set of states, $\to \subseteq Q \times \Sigma \times Q$ is the *transition relation*, and $Q^\circ \subseteq Q$ is the set of *initial states*.

The transition relation is written in infix notation $x \xrightarrow{\sigma} y$, and is extended to strings in Σ^* by letting $x \xrightarrow{\varepsilon} x$ for all $x \in Q$,

This work was supported by the Swedish Research Council.

and $x \xrightarrow{s\sigma} z$ if $x \xrightarrow{s} y$ and $y \xrightarrow{\sigma} z$ for some $y \in Q$. Furthermore, $x \xrightarrow{s}$ means that $x \xrightarrow{s} y$ for some $y \in Q$, and $x \to y$ means that $x \xrightarrow{s} y$ for some $s \in \Sigma^*$. These notations also apply to state sets, $X \xrightarrow{s}$ for $X \subseteq Q$ means that $x \xrightarrow{s}$ for some $x \in X$, and to automata, $G \xrightarrow{s}$ means that $Q^\circ \xrightarrow{s}$, etc. The language of an automaton G is $\mathcal{L}(G) = \{\, s \in \Sigma^* \mid G \xrightarrow{s} \,\}$.

A special requirement is that states reached by the termination event ω do not have any outgoing transitions, i.e., if $x \xrightarrow{\omega} y$ then there does not exist $\sigma \in \Sigma$ such that $y \xrightarrow{\sigma}$. This ensures that the termination event, if it occurs, is always the final event of any trace. The traditional set of *marked states* is $Q^\omega = \{\, x \in Q \mid x \xrightarrow{\omega} \,\}$ in this notation. For graphical simplicity, states in Q^ω are shown shaded in the figures of this paper instead of explicitly showing ω-transitions.

Definition 2: An automaton G is called *deterministic*, if $|Q^\circ| \leq 1$ and $x \xrightarrow{\sigma} y_1$ and $x \xrightarrow{\sigma} y_2$ always implies $y_1 = y_2$.

When automata are brought together to interact, lock-step synchronisation in the style of [13] is used.

Definition 3: Let $G_1 = \langle \Sigma_1, Q_1, \to_1, Q_1^\circ \rangle$ and $G_2 = \langle \Sigma_2, Q_2, \to_2, Q_2^\circ \rangle$ be two automata. The *synchronous composition* of G_1 and G_2 is defined as

$$G_1 \parallel G_2 = \langle \Sigma_1 \cup \Sigma_2, Q_1 \times Q_2, \to, Q_1^\circ \times Q_2^\circ \rangle \tag{1}$$

where

$$\begin{aligned}
&(x,y) \xrightarrow{\sigma} (x',y') \text{ if } \sigma \in (\Sigma_1 \cap \Sigma_2),\ x \xrightarrow{\sigma}_1 x',\ y \xrightarrow{\sigma}_2 y' ;\\
&(x,y) \xrightarrow{\sigma} (x',y) \text{ if } \sigma \in (\Sigma_1 \setminus \Sigma_2),\ x \xrightarrow{\sigma}_1 x' ;\\
&(x,y) \xrightarrow{\sigma} (x,y') \text{ if } \sigma \in (\Sigma_2 \setminus \Sigma_1),\ y \xrightarrow{\sigma}_2 y' .
\end{aligned}$$

C. Supervisory Control Theory

Given a *plant* automaton G and a *specification* automaton K, *supervisory control theory* [1] provides a method to synthesise a supervisor that restricts the behaviour of the plant such that the specification is always fulfilled. Two common requirements for the supervisor are *controllability* and *nonblocking*.

Definition 4: Let G and K be two automata using the same alphabet Σ. K is *controllable* with respect to G if, for every string $s \in \Sigma^*$, every state x of K, and every uncontrollable event $\upsilon \in \Sigma_u$ such that $K \xrightarrow{s} x$ and $G \xrightarrow{s\upsilon}$, it holds that $x \xrightarrow{\upsilon}$ in K.

Definition 5: Let $G = \langle \Sigma, Q, \to, Q^\circ \rangle$. A state $x \in Q$ is called *reachable* in G if $G \to x$, and *coreachable* if $x \xrightarrow{t\omega}$ for some $t \in \Sigma^*$. G is called reachable or coreachable, if every state $x \in Q$ has the respective property. G is called *nonblocking* if every reachable state is coreachable.

For a deterministic plant G and specification K, it is shown in [1] that there exists a unique *least restrictive* controllable sublanguage

$$\sup\mathcal{C}_G(K) \subseteq \mathcal{L}(K) \tag{2}$$

such that $\sup\mathcal{C}_G(K)$ is controllable with respect to G and nonblocking, and this language can be computed using a fixed-point iteration.

In [9], this result is generalised to nondeterministic automata. For nondeterministic automata, synthesis produces a *subautomaton* instead of a language, and the controllability condition is modified accordingly.

Definition 6: [9] Let $G_1 = \langle \Sigma, Q_1, \to_1, Q_1^\circ \rangle$ and $G_2 = \langle \Sigma, Q_2, \to_2, Q_2^\circ \rangle$ be two automata. G_1 is a *subautomaton* of G_2, written $G_1 \subseteq G_2$, if $Q_1 \subseteq Q_2$, $\to_1 \subseteq \to_2$, and $Q_1^\circ \subseteq Q_2^\circ$.

Definition 7: [9] Let $G = \langle \Sigma, Q_G, \to_G, Q_G^\circ \rangle$ and $K = \langle \Sigma, Q_K, \to_K, Q_K^\circ \rangle$ be automata such that $K \subseteq G$. Then K is called *controllable* in G if, for all states $x \in Q_K$ and $y \in Q_G$ and for every uncontrollable event $\upsilon \in \Sigma_u$ such that $x \xrightarrow{\upsilon}_G y$, it also holds that $x \xrightarrow{\upsilon}_K y$.

The upper bound of controllable and nonblocking subautomata is again controllable and nonblocking, and this implies the existence of a least restrictive synthesis result.

Theorem 1: Let $G = \langle \Sigma, Q, \to, Q^\circ \rangle$ be an automaton. There exists a unique subautomaton $\sup\mathcal{CN}(G) \subseteq G$ such that $\sup\mathcal{CN}(G)$ is nonblocking and controllable in G, and such that for every subautomaton $S \subseteq G$ that is also nonblocking and controllable in G, it holds that $S \subseteq \sup\mathcal{CN}(G)$.

Thus, $\sup\mathcal{CN}(G)$ is the unique synthesis result for a *plant* G. In order to apply this synthesis to control problems that also involve specifications, the transformation proposed in [8] is used. A specification automaton is transformed into a plant by adding, for every uncontrollable event that is not enabled in a state, a transition to a new blocking state $\perp$. This essentially transforms all potential controllability problems into potential blocking problems.

Definition 8: [8] Let $K = \langle \Sigma, Q, \to, Q^\circ \rangle$ be a specification. The *complete plant automaton* $K^\perp$ for K is

$$K^\perp = \langle \Sigma, Q \cup \{\perp\}, \to^\perp, Q^\circ \rangle \tag{3}$$

where $\perp \notin Q$ is a new state and

$$\to^\perp = \to \cup \{\, (x, \upsilon, \perp) \mid x \in Q, \upsilon \in \Sigma_u, x \not\xrightarrow{\upsilon} \,\} . \tag{4}$$

Proposition 2: [8] Let G, K, and K' be deterministic automata over the same alphabet Σ, and let K' be reachable. Then $K' \subseteq G \| K^\perp$ is nonblocking and controllable in $G \| K^\perp$ if and only if $K' \subseteq G \parallel K$ is nonblocking and controllable with respect to G.

According to this result, synthesis of the least restrictive nonblocking and controllable behaviour allowed by a specification K with respect to a plant G can be achieved by computing $\sup\mathcal{CN}(G \parallel K^\perp)$. If G and K are both deterministic, it can be shown that

$$\mathcal{L}(\sup\mathcal{CN}(G \parallel K^\perp)) = \sup\mathcal{C}_G(K) . \tag{5}$$

III. Renaming in Compositional Synthesis

The compositional approach proposed in [10], [11] exploits the modular structure present in many discrete event systems models to avoid state-space explosion and synthesise modular supervisors. Unlike previous approaches [8], [9], the compositional method in [10], [11] can synthesise least restrictive modular supervisors, but it requires deterministic automata throughout all abstraction steps. This section briefly outlines the compositional method in [10], [11], and then introduces *renaming* to avoid nondeterminism and extend the scope of applicable abstractions.

A. General Compositional Approach

A modular system consists of a modular specification $K = K_1 \parallel \cdots \parallel K_m$ and a modular plant $G = G_1 \parallel \cdots \parallel G_l$. As shown in Sect. II-C, the specifications can be translated into plants, so the synthesis problem consists of finding the least restrictive controllable and nonblocking supervisor for a set of plants,

$$G = G_1 \parallel \cdots \parallel G_n. \tag{6}$$

In the compositional algorithm of [10], [11], the modular system (6) is abstracted step by step. Each automaton G_i may be replaced by an abstracted version $\tilde{G}_i$. When no more abstraction is possible, synchronous composition is computed step by step, and each intermediate result is abstracted again.

Eventually, the procedure leads to a single automaton $\tilde{G}$, the abstract description of the original system. Once $\tilde{G}$ is found, the final step is to use $\tilde{G}$ instead of the original system, to synthesise $\sup\mathcal{CN}(\tilde{G})$, which leads to a synthesis result for the original system (6).

The abstraction steps to simplify the individual components G_i must satisfy certain conditions to guarantee that the synthesis result obtained from the final abstraction $\tilde{G}$ is a correct supervisor for the original system. The sufficient condition of *synthesis abstraction* is introduced in [10].

Definition 9: [10] Let G and $\tilde{G}$ be two deterministic automata with alphabet Σ. Then $\tilde{G}$ is a *synthesis abstraction* of G with respect to the *local events* $\Upsilon \subseteq \Sigma$, written $G \lesssim_{\text{synth},\Upsilon} \tilde{G}$, if for every deterministic automaton $T = \langle \Sigma_T, Q_T, \rightarrow_T, Q_T^\circ \rangle$ such that $\Sigma_T \cap \Upsilon = \emptyset$ the following holds,

$$\mathcal{L}(G \parallel T \parallel \sup\mathcal{CN}(\tilde{G} \parallel T)) = \mathcal{L}(G \parallel T \parallel \sup\mathcal{CN}(G \parallel T)) \tag{7}$$

Synthesis abstraction requires that the abstracted automaton $\tilde{G}$ yields the same synthesised language as the original automaton G, no matter what the behaviour of the remainder of the system T is. The abstraction is done with respect to a set Υ of *local events*, which are known to be used only in the automaton G being abstracted. Several suitable abstraction rules are described in [10], [11]. However, to guarantee that the synthesis abstraction is preserved after applying these rules, the abstracted automata are required to be deterministic in all the abstraction steps. Therefore these rules cannot be applied when the merging of the states results in nondeterminism which makes some abstractions impossible.

Example 1: Consider automata G, $\tilde{G}$, and T in Fig. 1. All events are controllable, and events α and β in G are local events, so states q_0 and q_1 in G are *synthesis observation equivalent* [11]. Merging these states results in nondeterministic automaton $\tilde{G}$ such that $G \lesssim_{\text{synth},\{\alpha,\beta\}} \tilde{G}$. Fig. 1 also shows $S = \sup\mathcal{CN}(\tilde{G} \| T)$. Since this supervisor enables event γ after both α and β, the closed-loop system is blocking, so S is not a correct supervisor.

B. Renaming

In [12], distinguishing sensors are proposed to replace a single event by different events to develop more suitable models. This idea can be used to remove nondeterminism as it arises in example 1.

Example 2: The abstraction $\tilde{G}$ in Fig. 1 is nondeterministic, because there are two transitions from q_{01} with event γ. Now consider automaton H in Fig. 1. Its alphabet $\{\alpha, \beta, \gamma_1, \gamma_2\}$ is obtained by replacing one of the γ-transitions with γ_1 and the other one with γ_2. Fig. 1 also shows $\tilde{H}$, the abstracted version of H, which is a deterministic automaton. Having replaced events in one component, any automata in the remainder of the system also need to be modified. Fig. 1 also shows automaton T', which is the adjusted version of automaton T, where all occurrences of γ are replaced by both γ_1 and γ_2.

The key to avoid nondeterminism in this example is to introduce new events that are linked to the original nondeterministic transitions by renaming. This idea is formalised in the following definition.

Definition 10: Let Σ_1 and Σ_2 be two sets of events. A *renaming* $\rho\colon \Sigma_2 \rightarrow \Sigma_1$ is a controllability-preserving map, i.e., a map such that $\rho(\sigma)$ is controllable if and only if σ is controllable.

This definition is extended to languages over Σ_2^* and to automata with alphabet Σ_2 in the standard way.

When introducing distinguishing sensors or renaming, some events are replaced by new events that do not appear in the original plant model. Then it is no longer clear how a supervisor synthesised from the renamed model can control the original plant. To make this possible, a *distinguisher* is introduced that enables the final supervisor to choose the correct transitions.

Definition 11: Let $\rho\colon \Sigma_2 \rightarrow \Sigma_1$ be a renaming. An automaton G_2 with alphabet Σ_2 is a *ρ-distinguisher* if, for all traces $s, t \in \mathcal{L}(G_2)$ such that $\rho(s) = \rho(t)$, it holds that $s = t$.

To guarantee the ρ-distinguisher property, each transition labelled by events from $\Sigma_1 \setminus \Sigma_2$ is renamed with a unique, not already existing label. In example 2, the renaming ρ is such that $\rho(\gamma_1) = \rho(\gamma_2) = \gamma$, and $\rho(\sigma) = \sigma$ for all other events. This renaming ensures $\rho(H) = G$, and H is a ρ-distinguisher.

The idea of a ρ-distinguisher is to enable a renamed supervisor to communicate correctly with the original unrenamed plant. If the plant executes an event γ, then the distinguisher is in a state that enables either γ_1 or γ_2, but not both, so the supervisor can execute the correct one among its renamed transitions.

After applying a renaming on component G_1 in a composed system such as (6), new events are introduced, so the remaining components need to be modified to use the new events. When G_1 is replaced by H_1 such that $\rho(H_1) = G_1$, all other components G_j are replaced by $\rho^{-1}(G_j)$ according to the following definition.

Definition 12: Let $G = \langle \Sigma_1, Q, \rightarrow, Q^\circ \rangle$ be an automaton, and let $\rho\colon \Sigma_2 \rightarrow \Sigma_1$ be a renaming. Then $\rho^{-1}(G) = \langle \Sigma_2, Q, \rho^{-1}(\rightarrow), Q^\circ \rangle$ where $\rho^{-1}(\rightarrow) = \{\, (x, \sigma, y) \mid x \xrightarrow{\rho(\sigma)} y \,\}$.

Example 3: Automaton T' in Fig. 1 is equal to $\rho^{-1}(T)$.

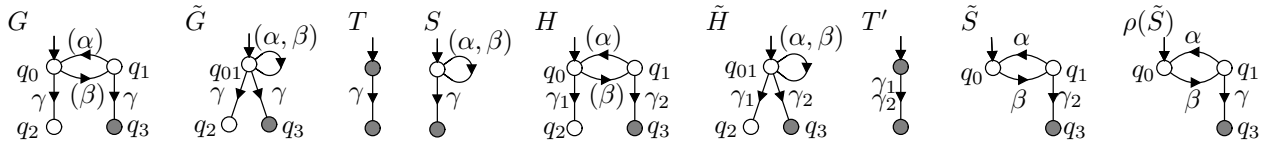

Fig. 1. Abstraction of G results in the nondeterministic automaton $\tilde{G}$, which is unsuitable as synthesis abstraction. Renaming event γ into γ_1 and γ_2 gives H, which leads to the suitable deterministic abstraction $\tilde{H}$.

C. Renaming and Compositional Synthesis

In order to support renaming in the compositional method, synthesis problems can no longer be represented only as a set of plants as suggested in (6). There are also renamings, supervisors, and distinguishers to be taken into account. Each step in the modified synthesis procedure considers:

- a set $\mathcal{G} = \{G_1, \ldots, G_n\}$ of uncontrolled plants;
- a set $\mathcal{S} = \{S_1, \ldots, S_m\}$ of collected supervisors and distinguishers;
- a renaming $\rho\colon \Sigma_{\mathcal{G}} \cup \Sigma_{\mathcal{S}} \to \Sigma$ from the combined alphabet of $\mathcal{G}$ and $\mathcal{S}$ into the alphabet Σ of the original system before renaming.

Definition 13: A *synthesis triple* is a triple $(\mathcal{G}; \mathcal{S}; \rho)$, where $\mathcal{G}$ and $\mathcal{S}$ are sets of automata and $\rho\colon \Sigma_{\mathcal{G}} \cup \Sigma_{\mathcal{S}} \to \Sigma$ is a renaming, such that $\mathcal{S}$ is a ρ-distinguisher.

The following definitions provide an appropriate notion of abstraction for synthesis triples.

Definition 14: Let $(\mathcal{G}; \mathcal{S}; \rho)$ be a synthesis triple. Then

(i) $\mathcal{L}(\mathcal{G}; \mathcal{S}; \rho) = \mathcal{L}(\rho(\mathcal{G} \parallel \mathcal{S})) = \rho\mathcal{L}(\mathcal{G} \parallel \mathcal{S})$;

(ii) $\sup\mathcal{CN}(\mathcal{G}; \mathcal{S}; \rho) = \rho(\sup\mathcal{CN}(\mathcal{G}) \parallel \mathcal{S})$.

Here, sets of automata represent the synchronous composition of their elements, e.g., $\mathcal{S} = \{S_1, \ldots, S_m\}$ stands for $\|_{i=1}^{m} S_i$.

Definition 15: Let $(\mathcal{G}_1; \mathcal{S}_1; \rho_1)$ and $(\mathcal{G}_2; \mathcal{S}_2; \rho_2)$ be two synthesis triples. Then $(\mathcal{G}_2; \mathcal{S}_2; \rho_2)$ is *synthesis equivalent* to $(\mathcal{G}_1; \mathcal{S}_1; \rho_1)$, written $(\mathcal{G}_1; \mathcal{S}_1; \rho_1) \simeq_{\text{synth}} (\mathcal{G}_2; \mathcal{S}_2; \rho_2)$, if $\mathcal{L}(\sup\mathcal{CN}(\mathcal{G}_1; \mathcal{S}_1; \rho_1)) = \mathcal{L}(\sup\mathcal{CN}(\mathcal{G}_2; \mathcal{S}_2; \rho_2))$.

The compositional synthesis algorithm is designed to calculate a modular supervisor for a modular system like (6). The initial synthesis triple is $(\mathcal{G}_0; \{\}; \mathrm{id})$, where $\mathcal{G}_0 = \{G_1, \ldots, G_n\}$ is the uncontrolled plant, and $\mathrm{id}\colon \Sigma \to \Sigma$ is the identical renaming, i.e., $\mathrm{id}(\sigma) = \sigma$ for all $\sigma \in \Sigma$. To construct a supervisor compositionally, the initial triple is abstracted repeatedly such that synthesis equivalence is preserved. The algorithm terminates once $\mathcal{G} = \{\tilde{G}\}$ consists of a single automaton representing the abstracted and renamed description of the original system. The following result shows that the supervisor calculated from $\tilde{G}$, together with the supervisors in $\mathcal{S}$ yields the same behaviour as a monolithic supervisor for the original system.

Proposition 3: Let $\mathcal{G} = \{G_1, \ldots, G_n\}$ be a set of automata, and let $(\mathcal{G}_0; \{\}; \mathrm{id}) \simeq_{\text{synth}} (\mathcal{G}_k; \mathcal{S}_k; \rho_k)$. Then

$$\mathcal{L}(\sup\mathcal{CN}(\mathcal{G}_k; \mathcal{S}_k; \rho_k)) = \mathcal{L}(\sup\mathcal{CN}(\mathcal{G}))\ . \tag{8}$$

Proof: As $(\mathcal{G}_k; \mathcal{S}_k; \rho_k) \simeq_{\text{synth}} (\mathcal{G}_0; \{\}; \mathrm{id})$, it follows from Def. 15 and 14 that

$$\begin{aligned} &\mathcal{L}(\sup\mathcal{CN}(\mathcal{G}_k; \mathcal{S}_k; \rho_k)) \;=\; \mathcal{L}(\sup\mathcal{CN}(\mathcal{G}_0; \{\}; \mathrm{id})) \\ =\; &\mathcal{L}(\mathrm{id}(\sup\mathcal{CN}(\mathcal{G}) \parallel \{\})) \;=\; \mathcal{L}(\sup\mathcal{CN}(\mathcal{G}))\ . \end{aligned}$$

■

This result shows that a least restrictive supervisor can be obtained by repeated abstraction of the initial synthesis triple $(\mathcal{G}_0; \{\}; \mathrm{id})$. The requirement that $\mathcal{S}_k$ is a ρ_k-distinguisher in Def. 13 ensures that this supervisor can be implemented to control the original unrenamed plant.

Note that the final supervisor never needs to be calculated explicitly. It can be represented in its modular form $\{\sup\mathcal{CN}(\tilde{G})\} \cup \mathcal{S}_k$, and synchronisation can be performed online, tracking the synchronous product states as the system evolves. In this way, synchronous product computation and state-space explosion can be avoided.

Example 4: Consider again automata G and T in Fig. 1, let $\mathcal{G}_0 = \{G, T\}$, and consider the initial synthesis triple $(\mathcal{G}_0; \{\}; \mathrm{id})$. As suggested in example 2, automaton G is replaced by H in Fig. 1, using renaming $\rho\colon \{\alpha, \beta, \gamma_1, \gamma_2\} \to \{\alpha, \beta, \gamma\}$ with $\rho(\alpha) = \alpha$, $\rho(\beta) = \beta$, and $\rho(\gamma_1) = \rho(\gamma_2) = \gamma$. It can be shown that $(\mathcal{G}_0; \{\}; \mathrm{id}) \simeq_{\text{synth}} (\mathcal{G}_1; \mathcal{S}; \rho)$ where $\mathcal{G}_1 = \{H, \rho^{-1}T\}$ and $\mathcal{S} = \{H\}$. After this renaming, synthesis observation equivalence [11] can be applied, which results in automaton $\tilde{H}$ shown in Fig. 1. This changes the synthesis triple to $(\mathcal{G}_2; \mathcal{S}; \rho)$ where $\mathcal{G}_2 = \{\tilde{H}, \rho^{-1}T\}$. The figure also shows $\tilde{S} = \sup\mathcal{CN}(\mathcal{G}_2) \parallel \mathcal{S}$ and $\rho(\tilde{S}) = \sup\mathcal{CN}(\mathcal{G}_2; \mathcal{S}; \rho)$, which is the least restrictive nonblocking and controllable behaviour.

D. Abstractions Preserving Synthesis Equivalence

Compositional synthesis requires suitable means to rewrite synthesis triples into equivalent simpler ones. Therefore, this section describes *synchronous composition* and *abstraction* as proposed in [10], [11], and then adds to this *renaming* to avoid nondeterminism, and *selfloop removal* to remove certain events from a model.

The simplest method to rewrite a synthesis triple is by *synchronous composition*. It is always possible to compose any two automata in the set $\mathcal{G}$ of uncontrolled plants, and the result is a synthesis equivalent triple.

Proposition 4: Let $\mathcal{G}_1 = \{G_1, \ldots, G_n\}$ and $\mathcal{G}_2 = \{G_1 \parallel G_2, G_3, \ldots, G_n\}$, let ρ be a renaming, and let $\mathcal{S}$ be a ρ-distinguisher. Then $(\mathcal{G}_1; \mathcal{S}; \rho) \simeq_{\text{synth}} (\mathcal{G}_2; \mathcal{S}; \rho)$.

The second method of rewriting synthesis triples is by *synthesis abstraction* [10], [11]. An automaton G_1 can be replaced by a synthesis abstraction H_1 provided that the original automaton G_1 is added as a supervisor component to the set $\mathcal{S}$.

Proposition 5: Let $\mathcal{G} = \{G_1, \ldots, G_n\}$ and $\mathcal{G}' = \{H_1, G_2, \ldots, G_n\}$, let Υ be a set of events not used in $G_2, \ldots, G_n$ such that $G_1 \lesssim_{\text{synth},\Upsilon} H_1$, let ρ be a renaming, and let $\mathcal{S}$ be a ρ-distinguisher. Then $(\mathcal{G}; \mathcal{S}; \rho) \simeq_{\text{synth}} (\mathcal{G}'; \mathcal{S} \cup \{G_1\}; \rho)$.

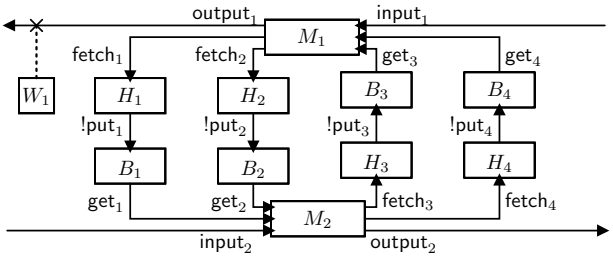

Fig. 2. Manufacturing system overview.

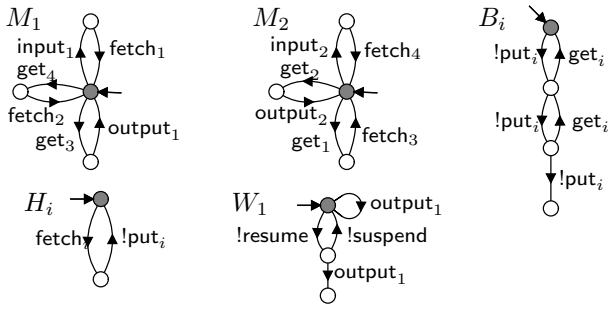

Fig. 3. Automata for manufacturing example.

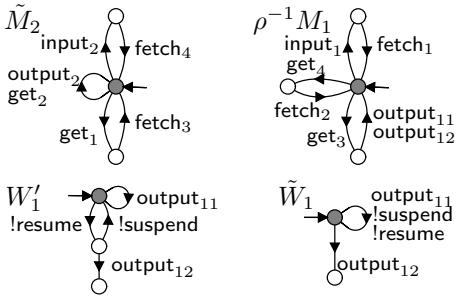

Fig. 4. Abstraction results of some of the automata of Fig. 3.

As explained above in Sect. III-B, an automaton G_1 can be rewritten by *renaming* into H_1, where the renaming ρ is such that $\rho(H_1) = G_1$ and H_1 is a ρ-distinguisher. Then H_1 is added to the set $\mathcal{S}$ of supervisors as a distinguisher, and the renaming ρ is composed with the previous renamings.

Proposition 6: Let $\rho\colon \Sigma_2 \to \Sigma_1$ and $\rho_1\colon \Sigma_1 \to \Sigma$ be renamings, let $\mathcal{G}_1 = \{G_1, \ldots, G_n\}$, let $\mathcal{G}_2 = \{H_1, \rho^{-1}G_2, \ldots, \rho^{-1}G_n\}$ where H_1 is a ρ-distinguisher such that $\rho(H_1) = G_1$, and let $\mathcal{S}$ be a ρ_1-distinguisher. Then $(\mathcal{G}_1; \mathcal{S}; \rho_1) \simeq_{\text{synth}} (\mathcal{G}_2; \rho^{-1}(\mathcal{S}) \cup \{H_1\}; \rho_1 \circ \rho)$.

It is a feature of the compositional approach in [10], [11] that it avoids hiding and the associated problems of nondeterminism. However, this makes it impossible to remove any events from automata, which eventually may make it difficult to apply other abstraction steps. With synthesis triples, it becomes clear that events can be removed once they only appear on selfloops in all components. This is formalised by the following rewrite method of *selfloop removal*.

Proposition 7: Let $(\mathcal{G}; \mathcal{S}; \rho)$ be a synthesis triple with $\mathcal{G} = \{G_1, \ldots, G_n\}$, and let $\Lambda \subseteq \Sigma$ be a set of events that only appear in selfloops in $\mathcal{G}$, i.e., if $G_i = \langle \Sigma_i, Q_i, \to_i, Q_i^\circ \rangle$ and $x \xrightarrow{\lambda}_i y$ for some $\lambda \in \Lambda$ then $x = y$. Then $(\mathcal{G}; \mathcal{S}; \rho) \simeq_{\text{synth}} (\mathcal{G}_{|\Sigma\setminus\Lambda}; \mathcal{G} \cup \mathcal{S}; \rho)$, where $\mathcal{G}_{|\Sigma\setminus\Lambda}$ is obtained from $\mathcal{G}$ by removing Λ from the alphabets of all automata in $\mathcal{G}$, and deleting all transitions labeled by events in Λ.

In Prop. 7, the supervisor set after abstraction, $\mathcal{G} \cup \mathcal{S}$, is larger than necessary. In fact, it is enough to include as additional supervisors only those automata from $\mathcal{G}$ that disable in some reachable state a controllable event contained in Λ.

IV. Example

In this section, compositional synthesis is applied to a manufacturing system consisting of two machines (M_1 and M_2) and four pairs of handlers (H_i) and buffers (B_i) for transferring workpieces between the machines. This is a modified version of a system studied previously in [14], [15]: a switch (W_1) has been added. Fig. 2 gives an overview of the system, and an automata model is shown in Fig. 3.

The manufacturing system can produce two types of workpieces. Type I workpieces are first processed by M_1 (input_1). Then they are fetched by H_1 (fetch_1) and placed into B_1 ($\mathsf{!put}_1$). Next, they are processed by M_2 (get_1), fetched by H_4 (fetch_4) and placed into B_4 ($\mathsf{!put}_4$). Finally, they are processed by M_1 once more (get_4), and released (output_1). Using a switch W_1, production can be suspended (!suspend) or resumed (!resume) by the user. Similarly, type II workpieces are first processed by M_2, passed through H_3 and B_3, further processed by M_1, passed through H_2 and B_2, and finally processed by M_2. Uncontrollable event are prefixed by !, all other events are controllable.

The initial synthesis triple is $(\mathcal{G}_0; \{\}; \mathrm{id})$ where $\mathcal{G}_0 = \{M_1, W_1, M_2, H_1, B_1, \ldots, H_4, B_4\}$. In this model, output_2 is a controllable local event, so synthesis observation equivalence [11] can be used to replace M_2 by the abstraction result $\tilde{M}_2$ such that $M_2 \lesssim_{\text{synth},\{\mathsf{output}_2\}} \tilde{M}_2$, shown in Fig. 4. The new synthesis triple is $(\mathcal{G}_1; \mathcal{S}_1; \mathrm{id})$ where $\mathcal{G}_1 = \{M_1, W_1, \tilde{M}_2, H_1, B_1, \ldots, H_4, B_4\}$ and $\mathcal{S}_1 = \{M_2\}$. Now event output_2 only appears in the selfloop in $\tilde{M}_2$, so selfloop removal can be applied, resulting in the new synthesis triple $(\mathcal{G}_2; \mathcal{S}_2; \mathrm{id})$ where $\mathcal{G}_2 = \{M_1, W_1, \tilde{M}_{2|\Sigma\setminus\{\mathsf{output}_2\}}, H_1, B_1, \ldots, H_4, B_4\}$ and $\mathcal{S}_2 = \{M_2, \tilde{M}_2\}$.

Events !suspend and !resume are uncontrollable local events in W_1, so uncontrollable observation equivalence is applicable. However, this abstraction causes nondeterminism, so a renaming is applied first. Let output_{11} and output_{12} be new events, and let ρ be a renaming such that $\rho(\mathsf{output}_{11}) = \rho(\mathsf{output}_{12}) = \mathsf{output}_1$ and $\rho(\sigma) = \sigma$ for $\sigma \in \Sigma \setminus \{\mathsf{output}_1\}$. Fig. 4 shows a ρ-distinguisher W_1' such that $\rho(W_1') = W_1$. Automaton M_1 also uses event output_1 and therefore is replaced by $\rho^{-1}M_1$, also shown in Fig. 4; the other automata in $\mathcal{G}_2$ are unchanged when ρ^{-1} is applied. The distinguisher W_1' ensures that only one of the events output_1 or output_2 is enabled in every state. Therefore, when the plant later sends event output to the synthesised supervisor, the distinguisher can replace this by output_1 or output_2 as appropriate.

The new synthesis triple after renaming is $(\mathcal{G}_3; \mathcal{S}_3; \rho)$ where

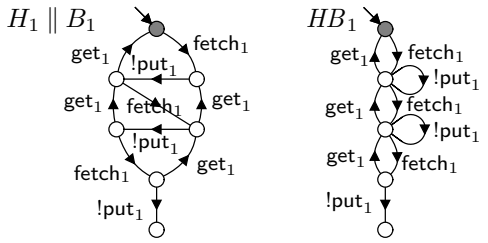

Fig. 5. $H_1 \parallel B_1$ and its abstraction result.

$\mathcal{G}_3 = \{\rho^{-1}M_1, W_1', \tilde{M}_{2|\Sigma\setminus\{\mathsf{output}_2\}}, H_1, B_1, \ldots, H_4, B_4\}$ and $\mathcal{S}_3 = \{M_2, \tilde{M}_2, W_1'\}$. Now uncontrollable observation equivalence [11] leads to the deterministic abstraction result $\tilde{W}_1$ such that $W_1' \lesssim_{\mathrm{synth},\{!\mathsf{suspend},!\mathsf{resume}\}} \tilde{W}_1$, shown in Fig. 4. Abstraction leads to the new synthesis triple $(\mathcal{G}_4; \mathcal{S}_4; \rho)$ where $\mathcal{G}_4 = \{\rho^{-1}M_1, \tilde{W}_1, \tilde{M}_{2|\Sigma\setminus\{\mathsf{output}_2\}}, H_1, B_1, \ldots, H_4, B_4\}$ and $\mathcal{S}_4 = \mathcal{S}_3 \cup \{W_1'\} = \mathcal{S}_3$. Now events !suspend and !resume only appear in selfloops in $\tilde{W}_1$, so selfloop removal can be applied and gives the new synthesis triple $(\mathcal{G}_5; \mathcal{S}_5; \rho)$ where $\mathcal{G}_5 = \{\rho^{-1}M_1, \tilde{W}_{1|\{!\mathsf{suspend},!\mathsf{resume}\}}, \tilde{M}_{2|\Sigma\setminus\{\mathsf{output}_2\}}, H_1, B_1, \ldots, H_4, B_4\}$ and $\mathcal{S}_5 = \mathcal{S}_4 \cup \{\tilde{W}_1\} = \{M_2, \tilde{M}_2, W_1', \tilde{W}_1\}$.

At this point, no further abstraction is possible and some automata need to be composed. Composing H_1 and B_1 results in the new synthesis triple $(\mathcal{G}_6; \mathcal{S}_6; \rho)$ where $\mathcal{G}_6 = \{\rho^{-1}M_1, \tilde{W}_{1|\{!\mathsf{suspend},!\mathsf{resume}\}}, \tilde{M}_{2|\Sigma\setminus\{\mathsf{output}_2\}}, H_1 \parallel B_1, H_2, B_2, \ldots, H_4, B_4\}$ and $\mathcal{S}_6 = \mathcal{S}_5$. Now event $!\mathsf{put}_1$ is an uncontrollable local event in $H_1 \parallel B_1$, and thus $H_1 \parallel B_1$ can be abstracted to HB_1 using uncontrollable observation equivalence [11]. Fig. 5 shows the composition $H_1 \parallel B_1$ and its abstraction result HB_1. The modified synthesis triple is $(\mathcal{G}_7; \mathcal{S}_7; \rho)$ where $\mathcal{G}_7 = \{\rho^{-1}M_1, \tilde{W}_{1|\{!\mathsf{suspend},!\mathsf{resume}\}}, \tilde{M}_{2|\Sigma\setminus\{\mathsf{output}_2\}}, HB_1, H_2, B_2, \ldots, H_4, B_4\}$ and $\mathcal{S}_7 = \mathcal{S}_6 \cup \{H_1 \parallel B_1\} = \{M_2, \tilde{M}_2, W_1', \tilde{W}_1, H_1 \parallel B_1\}$. The abstraction steps leading to $(\mathcal{G}_6; \mathcal{S}_6; \rho)$ and $(\mathcal{G}_7; \mathcal{S}_7; \rho)$ are repeated for the remaining subsystems, resulting in the final synthesis triple $(\mathcal{G}_{13}; \mathcal{S}_{13}; \rho)$ where $\mathcal{G}_{13} = \{\rho^{-1}M_1, \tilde{W}_{1|\{!\mathsf{suspend},!\mathsf{resume}\}}, \tilde{M}_{2|\Sigma\setminus\{\mathsf{output}_2\}}, HB_1, \ldots, HB_4\}$ and $\mathcal{S}_{13} = \{M_2, \tilde{M}_2, W_1', \tilde{W}_1, H_1 \parallel B_1, \ldots, H_4 \parallel B_4\}$.

The final step is to calculate the supervisor for the final synthesis triple $(\mathcal{G}_{13}; \mathcal{S}_{13}; \rho)$,

$$\mathrm{sup}\mathcal{CN}(\mathcal{G}_{13}; \mathcal{S}_{13}; \rho) = \rho(\mathrm{sup}\mathcal{CN}(\mathcal{G}_{13}) \parallel \mathcal{S}_{13}) \ . \qquad (9)$$

The final synthesis step to compute $\mathrm{sup}\mathcal{CN}(\mathcal{G}_{13})$ explores the state space of $\mathcal{G}_{13}$, which has 14600 states. This is in contrast to monolithic synthesis and modular synthesis without renaming [10], [11], which explore state spaces of 61776 and 21900 states respectively.

The number of states of the final supervisor (9) is 18432, but there is no need to construct this synchronous product. The resultant supervisor can be represented modularly using the components $\mathrm{sup}\mathcal{CN}(\mathcal{G}_{13})$ and the members of $\mathcal{S}_{13}$. The largest of these automata is $\mathrm{sup}\mathcal{CN}(\mathcal{G}_{13})$ with 919 states.

The supervisor resulting from (9) improves on the solution in [14] because it is nonblocking, and unlike [15], it can be computed fully automatically and is guaranteed to be least restrictive.

V. Conclusions

The compositional synthesis approach based on synthesis abstraction [10], [11] is generalised to support *renaming*. It is shown how renaming can avoid abstraction steps that would result in nondeterministic automata, so renaming makes it possible to apply certain abstraction steps in cases where it would not have been possible in previous work. Abstraction rules proposed in previous work are generalised in the new framework, and abstraction rules for renaming and selfloop removal are added to the framework. In future work, the authors would like to further generalise the framework to take the original plant model fully into account, and to support *halfway synthesis* [8].

References

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

[2] R. Kumar and M. A. Shayman, "Centralized and decentralized supervisory control of nondeterministic systems under partial observation," *SIAM J. Control and Optimization*, vol. 35, no. 2, pp. 363–383, Mar. 1997.

[3] M. Heymann and F. Lin, "Discrete event control of nondeterministic systems," 1997.

[4] C. Zhou and R. Kumar, "A small model theorem for bisimilarity control under partial observation," in *Proc. American Control Conf. 2005*, Portland, OR, USA, Aug. 2005, pp. 3937–3942.

[5] L. Feng and W. M. Wonham, "Computationally efficient supervisor design: Abstraction and modularity," in *Proc. 8th Int. Workshop on Discrete Event Systems, WODES '06*, Ann Arbor, MI, USA, Jul. 2006, pp. 3–8.

[6] K. Schmidt and C. Breindl, "On maximal permissiveness of hierarchical and modular supervisory control approaches for discrete event systems," in *Proc. 9th Int. Workshop on Discrete Event Systems, WODES '08*, Göteborg, Sweden, May 2008, pp. 462–467.

[7] R. Su, J. H. van Schuppen, and J. E. Rooda, "Model abstraction of nondeterministic finite-state automata in supervisor synthesis," *IEEE Trans. Automat. Contr.*, vol. 55, no. 11, pp. 2527–2541, Nov. 2010.

[8] H. Flordal, R. Malik, M. Fabian, and K. Åkesson, "Compositional synthesis of maximally permissive supervisors using supervision equivalence," *Discrete Event Dyn. Syst.*, vol. 17, no. 4, pp. 475–504, 2007.

[9] R. Malik and H. Flordal, "Yet another approach to compositional synthesis of discrete event systems," in *Proc. 9th Int. Workshop on Discrete Event Systems, WODES '08*, Göteborg, Sweden, May 2008, pp. 16–21.

[10] S. Mohajerani, R. Malik, S. Ware, and M. Fabian, "Compositional synthesis of discrete event systems using synthesis abstraction," in *Proc. 23rd Chinese Control and Decision Conf., CCDC 2011*, Mianyang, China, 2011, to appear.

[11] ——, "Three variations of observation equivalence preserving synthesis abstraction," Working Paper 01/2011, Dept. of Computer Science, University of Waikato, Hamilton, New Zealand, 2011.

[12] G. Bouzon, M. H. de Queiroz, and J. E. R. Cury, "Exploiting distinguishing sensors in supervisory control of DES," in *Proc. 7th Int. Conf. Control and Automation, ICCA '09*, Christchurch, New Zealand, Dec. 2009, pp. 442–447.

[13] C. A. R. Hoare, *Communicating Sequential Processes*. Prentice-Hall, 1985.

[14] F. Lin and W. M. Wonham, "Decentralized control and coordination of discrete-event systems with partial observation," *IEEE Trans. Automat. Contr.*, vol. 35, no. 12, pp. 1330–1337, Dec. 1990.

[15] P. Malik, R. Malik, D. Streader, and S. Reeves, "Modular synthesis of discrete controllers," in *Proc. 12th IEEE Int. Conf. Engineering of Complex Computer Systems, ICECCS '07*, Auckland, New Zealand, 2007, pp. 25–34.