On the Use of Observation Equivalence in Synthesis Abstraction

Sahar Mohajerani* Robi Malik† Simon Ware† 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,siw4}@cs.waikato.ac.nz

***Abstract*—In a previous paper we introduced the notion of *synthesis abstraction*, which allows efficient compositional synthesis of maximally permissive supervisors for large-scale systems of composed finite-state automata. In the current paper, *observation equivalence* is studied in relation to synthesis abstraction. It is shown that general observation equivalence is not useful for synthesis abstraction. Instead, we introduce additional conditions strengthening observation equivalence, so that it can be used with the compositional synthesis method. The paper concludes with an example showing the suitability of these relations to achieve substantial state reduction while computing a modular supervisor.**

I. Introduction

Modular approaches to supervisor synthesis are of great interest in *supervisory control theory* [1], [2], firstly in order to find more comprehensible supervisor representations, and secondly to overcome the problem of *state-space explosion* for systems with a large number of components. Many approaches studied so far, such as [3], [4], rely on structure to be provided by users and hence are hard to automate. Other early methods such as [5] only consider the synthesis of a least restrictive controllable supervisor, ignoring nonblocking. *Supervisor reduction* [6] greatly helps to simplify synthesised supervisors, yet it relies on a monolithic supervisor to be constructed first, and thus remains limited by its size.

More recently, abstraction based on *natural projection* has been studied for compositional supervisor synthesis. Natural projection with the *observer property* produces a nonblocking but not necessarily least restrictive supervisor; if *output control consistency* is added as an additional requirement, least restrictiveness can be ensured [7]. In [8], it is furthermore shown that output control consistency can be replaced by a weaker condition called *local control consistency*.

Supervisor synthesis and abstractions have also been studied in a nondeterministic setting. In [9], [10], *conflict-preserving* abstractions and *weak observation equivalence* are shown to be adequate for the synthesis of nonblocking supervisors, but least restrictiveness is only guaranteed if all observable events are retained in the abstraction. The methods in [11], [12] also allow for the abstraction of observable events through *hiding*.

In more recent work [13], the authors propose another means of abstraction called *synthesis abstraction*, which avoids hiding and some of the problems encountered in [11], [12]. This present paper builds on this work and investigates how automata can be simplified in the framework of synthesis abstraction. The focus is on *observation equivalence* and related methods.

After the preliminaries in Sect. II, the framework of synthesis abstraction is presented in Sect. III. Next, in Sect. IV observation equivalence-based abstractions are studied in detail. It is first shown that general observation equivalence is not suitable for synthesis abstraction, and then the stronger versions of *uncontrollable observation equivalence* and *synthesis observation equivalence* are shown to guarantee synthesis abstraction. It is also shown that synthesis observation equivalence can produce better abstraction than the projection-based method of [8]. Finally, Sect. V demonstrates observation equivalence-based abstraction using a practical example, and Sect. VI adds some concluding remarks. Most formal proofs are omitted for lack of space in this paper and can be found in [14].

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. A string $s \in \Sigma^*$ is called a *prefix* of $t \in \Sigma^*$, written $s \sqsubseteq t$, if $t = su$ for some $u \in \Sigma^*$. 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

Discrete system behaviours are typically modelled by deterministic automata, but notation in this paper is based on nondeterministic automata, which may arise as intermediate results during abstraction.

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 *state transition relation*, and $Q^\circ \subseteq Q$ is the set of *initial states*. G is *deterministic*, if $|Q^\circ| \leq 1$ and $x \xrightarrow{\sigma} y_1$ and $x \xrightarrow{\sigma} y_2$ always implies $y_1 = y_2$.

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

$x \in Q$, 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.

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, always is 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.

For a state or state set x, the *continuation language* is $\mathcal{L}(x) = \{\, s \in \Sigma^* \mid x \xrightarrow{s} \,\}$. The language of an automaton G is $\mathcal{L}(G) = \mathcal{L}(Q^\circ)$, and its marked language is $\mathcal{M}(G) = \{\, s \in \Sigma^* \mid s\omega \in \mathcal{L}(G) \,\}$.

When automata are brought together to interact, lock-step synchronisation in the style of [15] is used.

Definition 2: 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

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

Another common automaton operation is the *quotient* modulo an equivalence relation on the state set.

Definition 3: Let $G = \langle \Sigma, Q, \to, Q^\circ \rangle$ be an automaton and let $\sim\ \subseteq Q \times Q$ be an equivalence relation. The *quotient automaton* of G modulo $\sim$ is

$$G/{\sim} = \langle \Sigma, Q/{\sim}, {\to}/{\sim}, \tilde{Q}^\circ \rangle \ , \tag{2}$$

where ${\to}/{\sim} = \{\, [x] \xrightarrow{\sigma} [y] \mid x \xrightarrow{\sigma} y \,\}$ and $\tilde{Q}^\circ = \{\, [x^\circ] \mid x^\circ \in Q^\circ \,\}$. Here, $[x] = \{\, x' \in Q \mid x \sim x' \,\}$ denotes the *equivalence class* of $x \in Q$, and $Q/{\sim} = \{\, [x] \mid x \in Q \,\}$ is the set of all equivalence classes modulo $\sim$.

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 *least restrictive* controllable sublanguage

$$\sup\mathcal{C}_G(K) \subseteq \mathcal{L}(K) \tag{3}$$

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 [12], 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: [12] 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: [12] 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 $S \subseteq \sup\mathcal{CN}(G)$.

Therefore, $\sup\mathcal{CN}(G)$ is the unique synthesis result for a *plant* G. It is shown in [12] how $\sup\mathcal{CN}(G)$ can be computed using a fixpoint iteration. In order to apply this synthesis to control problems that also involve specifications, the transformation proposed in [11] 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: [11] Let $K = \langle \Sigma, Q, \to, Q^\circ \rangle$ be a specification. The *complete plant automaton* $K^\perp$ for K is

$$K^\perp \ = \ \langle \Sigma, Q \cup \{\perp\}, \to^\perp, Q^\circ \rangle \tag{4}$$

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

$$\to^\perp \ = \ \to \cup \ \{\, (x, \upsilon, \perp) \mid x \in Q, \upsilon \in \Sigma_u, x \stackrel{\upsilon}{\not\to} \,\} \ . \tag{5}$$

Proposition 2: [11] Let G, K, and K' be deterministic automata over the same alphabet Σ, and let K' be reachable. Then $K' \subseteq G \parallel K^\perp$ is nonblocking and controllable in $G \parallel K^\perp$ 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

$$\mathcal{L}\mathrm{sup}\mathcal{CN}(G \parallel K^{\perp}) = \mathrm{sup}\mathcal{C}_G(K) \ . \tag{6}$$

III. Compositional Synthesis

Many discrete event systems are *modular* in that they consist of a large number of interacting components. This modularity can be used to abstract components before composing them, in many cases avoiding state-space explosion. This section briefly describes the framework introduced in [13] to perform synthesis compositionally in this setting.

A. General Compositional Approach

A modular system consists of a modular specification $K = K_1 \parallel \cdots \parallel K_m$ and a modular plant $G = G_1 \parallel \cdots \parallel G_n$. As discussed in Sect. II-C, all the specifications can be translated to plants, so the synthesis problem is given as

$$G \parallel K^{\perp} = G_1 \parallel \cdots \parallel G_n \parallel K_1^{\perp} \parallel \cdots \parallel K_m^{\perp} \ . \tag{7}$$

In the compositional algorithm of [13], the modular system (7) is abstracted step by step. Each automaton G_i or $K_j^{\perp}$ in (7) may be replaced by an abstracted version, $\tilde{G}_i$ or $\tilde{K}_j^{\perp}$, until no more abstraction is possible. Then synchronous composition is computed step by step, abstracting each intermediate result again.

When abstracting an automaton G_i, this automaton will typically contain some events that do not appear in any other component G_i or $K_j^{\perp}$. These events are called *local events*. In the following, local events are denoted by the set Υ, and $\Omega = \Sigma \setminus \Upsilon$ denotes the non-local or *shared* events. Local events are helpful to find an abstraction.

Eventually, the procedure leads to a single automaton $\tilde{G}$, the abstract description of the system (7). After abstraction, $\tilde{G}$ has less states and transitions compared to the original system. Once $\tilde{G}$ is found, the final step is to use $\tilde{G}$ instead of the original system, to calculate a synthesis result $\mathrm{sup}\mathcal{CN}(\tilde{G})$, which leads to a solution for the original synthesis problem (7).

B. Synthesis Abstraction

The general compositional approach explained above requires an appropriate notion of abstraction. The task is to find the least restrictive, nonblocking, and controllable supervisor, so each automaton should be abstracted in such a way that the behaviour of the supervised system is left unchanged.

Definition 9: [13] 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 $G \lesssim_{\mathrm{synth},\Upsilon} \tilde{G}$, 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,

$$\mathcal{L}(G \parallel T \parallel \mathrm{sup}\mathcal{CN}(\tilde{G} \parallel T)) = \mathcal{L}(G \parallel T \parallel \mathrm{sup}\mathcal{CN}(G \parallel T)) \tag{8}$$

Def. 9 requires that the synthesis result for G and its abstraction $\tilde{G}$ are the same in every possible context T. The synthesis results $\mathrm{sup}\mathcal{CN}(G \parallel T)$ and $\mathrm{sup}\mathcal{CN}(\tilde{G} \parallel T)$ are composed with the original plant $G \parallel T$, and the resultant behaviours must be equal. The following theorem shows how synthesis abstraction is applied to a control problem such as (7).

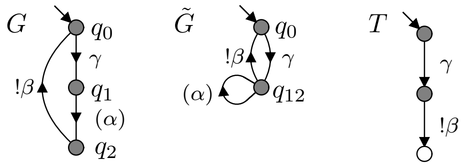

Fig. 1. $\tilde{G}$ is observation equivalent to G, but not a synthesis abstraction.

Theorem 3: Let $H_i = \langle \Sigma_i, Q_i, \rightarrow_i, Q_i^{\circ} \rangle$, $i = 1, \ldots, k$, be deterministic automata, and let $\Upsilon \subseteq \Sigma_1$ such that $H_1 \lesssim_{\mathrm{synth},\Upsilon} \tilde{H}_1$ and $\Upsilon \cap \Sigma_2 = \cdots = \Upsilon \cap \Sigma_k = \emptyset$. Then

$$\begin{aligned} &\mathcal{L}(H_1 \parallel \cdots \parallel H_k \parallel \mathrm{sup}\mathcal{CN}(H_1 \parallel H_2 \parallel \cdots \parallel H_k)) \\ &= \mathcal{L}(H_1 \parallel \cdots \parallel H_k \parallel \mathrm{sup}\mathcal{CN}(\tilde{H}_1 \parallel H_2 \parallel \cdots \parallel H_k)) \ . \end{aligned} \tag{9}$$

Proof: The claim follows directly from Def. 9 by considering $H_2 \parallel \cdots \parallel H_k$ as T. ■

Theorem 3 is applied several times when simplifying (7). It can be shown by induction that, if (7) is composed and simplified to a single automaton $\tilde{G}$, then the synthesis result $\tilde{S} = \mathrm{sup}\mathcal{CN}(\tilde{G})$ composed with the original system (7) is equal to the monolithic synthesis result for (7). A least restrictive *modular supervisor* can be constructed as $\tilde{S} \| K_1 \| K_2 \| \cdots \| K_m$.

Note that the modular supervisor $\tilde{S} \parallel K_1 \parallel K_2 \parallel \cdots \parallel K_m$ never needs to be calculated. It can be represented in its modular form, and synchronisation can be performed on-line, tracking the synchronous product states as the system evolves. In this way, synchronous product computation and state-space explosion can be avoided.

IV. Methods of Abstraction

This section discusses some possible methods to compute synthesis abstractions. While observation equivalence does not in general yield synthesis abstractions, it can be strengthened to do so.

A. Observation Equivalence

Observation equivalence or *weak bisimilarity* is a well-known general abstraction method for nondeterministic automata [16]. It seeks to merge *observation equivalent* states, i.e., states with the same future behaviour.

Definition 10: Let $G = \langle \Sigma, Q, \rightarrow, Q^{\circ} \rangle$ be an automaton with $\Sigma = \Omega \,\dot{\cup}\, \Upsilon$. An equivalence relation $\approx \ \subseteq Q \times Q$ is called an *observation equivalence* on G with respect to Υ, if the following holds for all $x_1, x_2 \in Q$ such that $x_1 \approx x_2$: if $x_1 \xrightarrow{t_1 \sigma u_1} y_1$ for some $\sigma \in \Sigma$ and $t_1, u_1 \in \Upsilon^*$, then there exist $y_2 \in Q$ and $t_2, u_2 \in \Upsilon^*$ such that $x_2 \xrightarrow{t_2 P_{\Omega}(\sigma) u_2} y_2$ and $y_1 \approx y_2$.

Observation equivalence is known to preserve all temporal logic properties [16] including *conflict equivalence* [17]. However, it does not always produce a synthesis abstraction, and the following counterexample shows this.

Example 1: Consider automata G, $\tilde{G}$, and T in Fig. 1. Uncontrollable events are prefixed with !, and local events in Υ

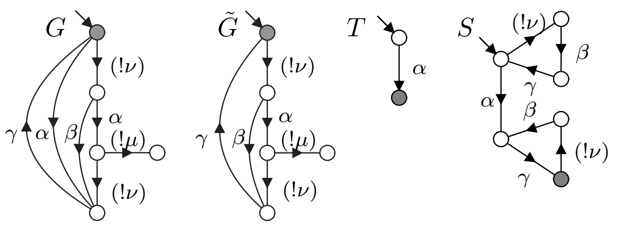

Fig. 2. $\tilde{G}$ is observation equivalent to G with only uncontrollable local events. Nevertheless it is not a synthesis abstraction.

are marked with parentheses around them. With $\alpha \in \Upsilon$, states q_1 and q_2 in G are observation equivalent, and merging them produces the abstraction $\tilde{G}$. However, $\gamma \in \mathcal{L}\mathrm{sup}\mathcal{CN}(G \parallel T)$ but $\gamma \notin \mathcal{L}\mathrm{sup}\mathcal{CN}(\tilde{G} \parallel T)$, because in G, the local controllable event α can be disabled to prevent the state q_2 and thus the undesirable uncontrollable $!\beta$, but this is no longer possible in $\tilde{G}$. Thus, $\mathcal{L}(G \parallel T \parallel \mathrm{sup}\mathcal{CN}(\tilde{G} \parallel T)) \neq \mathcal{L}(G \parallel T \parallel \mathrm{sup}\mathcal{CN}(G \parallel T))$, and $\tilde{G}$ is not a synthesis abstraction of G.

This counterexample seems to contradict results in [9], [10], where observation equivalence is used in synthesis abstraction. However, the above mentioned papers only allow *unobservable* events to be considered as local, while in this paper *observable* events can also be local.

B. Bisimulation

One simple way to restrict observation equivalence such that it implies synthesis abstraction is by not permitting any local events. This leads to *bisimulation equivalence* [16], one of the strongest known process equivalences.

Definition 11: Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ be an automaton. An equivalence relation $\approx \subseteq Q \times Q$ is called a *bisimulation* on G, if the following holds for all $x_1, x_2 \in Q$ such that $x_1 \approx x_2$: if $x_1 \xrightarrow{\sigma} y_1$ for some $\sigma \in \Sigma$, then there exists $y_2 \in Q$ such that $x_2 \xrightarrow{\sigma} y_2$ and $y_1 \approx y_2$.

Theorem 4: Let G be an automaton, and let $\approx$ be a bisimulation on G. Then $G \lesssim_{\mathrm{synth},\emptyset} G/{\approx}$.

C. Uncontrollable Observation Equivalence

While bisimulation ensures synthesis abstraction, not permitting any local events is highly restrictive, and it is desirable to relax the condition. In example 1, the local event, α, is controllable; if it was uncontrollable, merging the states would result in a synthesis abstraction. This suggests to restrict the set of local events to be uncontrollable, yet the following counterexample shows that this is still not enough.

Example 2: In Fig. 2, the local events $!\mu$ and $!\nu$ are both uncontrollable, and $\tilde{G}$ is observation equivalent to G. The figure also shows $S = G \parallel T \parallel \mathrm{sup}\mathcal{CN}(G \parallel T)$. However, $\mathcal{L}\mathrm{sup}\mathcal{CN}(\tilde{G} \parallel T) = \emptyset$, because in $\tilde{G}$ there is no way to permit event α without also permitting the deadlock after the uncontrollable $!\mu$. Thus, $\tilde{G}$ is not a synthesis abstraction of G.

The situation in example 2 can be avoided by requiring that the trace matching a controllable transition (such as the α-transition in the example) does not contain any more local events *after* the controllable event.

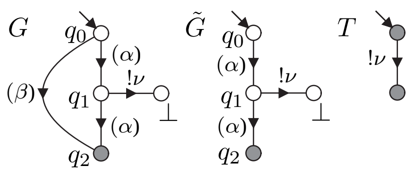

Fig. 3. $\tilde{G}$ is observation equivalent to G, but not a synthesis abstraction.

Definition 12: Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ be an automaton with $\Sigma = \Omega \dot\cup \Upsilon$ and $\Upsilon \subseteq \Sigma_u$. An equivalence relation $\sim \subseteq Q \times Q$ is called an *uncontrollable observation equivalence* on G with respect to Υ, if the following conditions hold for all $x_1, x_2 \in Q$ such that $x_1 \sim x_2$:

(i) $\forall \sigma \in \Sigma_c$, if $x_1 \xrightarrow{\sigma} y_1$ then $\exists t_2 \in \Upsilon^*$ such that $x_2 \xrightarrow{t_2 P_\Omega(\sigma)} y_2$ and $y_1 \sim y_2$;

(ii) $\forall \sigma \in \Sigma_u$, if $x_1 \xrightarrow{\sigma} y_1$ then $\exists t_2, u_2 \in \Upsilon^*$ such that $x_2 \xrightarrow{t_2 P_\Omega(\sigma) u_2} y_2$ and $y_1 \sim y_2$.

Condition (ii) is like observation equivalence (Def. 10), but (i) imposes a stronger requirement for controllable events.

Theorem 5: Let $\sim$ be an uncontrollable observation equivalence on G with respect to Υ. Then $G \lesssim_{\mathrm{synth},\Upsilon} G/{\sim}$.

D. Synthesis Observation Equivalence

This section shows that the conditions of uncontrollable observation equivalence can be relaxed, permitting controllable local events under certain conditions.

Example 3: Automata G and $\tilde{G}$ in Fig. 3 are observation equivalent with controllable local events α and β, because the local controllable β-transition is redundant according to observation equivalence. In both G and $\tilde{G}$, the controllable event α must be disabled to prevent the undesired uncontrollable $!\nu$. By disabling α in $\tilde{G}$, termination no longer can be achieved, yet termination is still possible in G using the β-transition. Therefore, $\tilde{G}$ is not a synthesis abstraction of G.

The situation in example 3 can be avoided by imposing an additional requirement as follows: a local controllable transition $x \xrightarrow{\sigma} y$ in G needs to have a matching sequence of local transitions $[x] \xrightarrow{s} [y]$ in $\tilde{G}$ such that every state along this path, reached by a local controllable transition, is equivalent to x. In the example, the transition $q_0 \xrightarrow{\beta} q_2$ in G can only be matched by the transition sequence $q_0 \xrightarrow{\alpha} q_1 \xrightarrow{\alpha} q_2$ in $\tilde{G}$, but the state q_1 in this sequence is not equivalent to q_0 in G. This idea leads to the following definition.

Definition 13: Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ be an automaton with $\Sigma = \Omega \dot\cup \Upsilon$. An equivalence relation $\sim \subseteq Q \times Q$ is called a *synthesis observation equivalence* on G with respect to Υ, if the following conditions hold for all $x_1, x_2 \in Q$ such that $x_1 \sim x_2$:

(i) $\forall \sigma \in \Sigma_c$, if $x_1 \xrightarrow{\sigma} y_1$ then $\exists t_2 \in \Upsilon^*$ such that $x_2 \xrightarrow{t_2 P_\Omega(\sigma)} y_2$ and $y_1 \sim y_2$ and for all strings $p_2 \sqsubseteq t_2$ such that $x_2 \xrightarrow{p_2} z_2$ and $p_2 \in \Sigma^* \Sigma_c$ it holds that $x_1 \sim z_2$;

(ii) $\forall \sigma \in \Sigma_u$, if $x_1 \xrightarrow{\sigma} y_1$ then $\exists t_2, u_2 \in (\Upsilon \cap \Sigma_u)^*$ such that $x_2 \xrightarrow{t_2 P_\Omega(\sigma) u_2} y_2$ and $y_1 \sim y_2$.

Theorem 6: Let $\sim$ be a synthesis observation equivalence on G with respect to Υ. Then $G \lesssim_{\mathrm{synth},\Upsilon} G/{\sim}$.

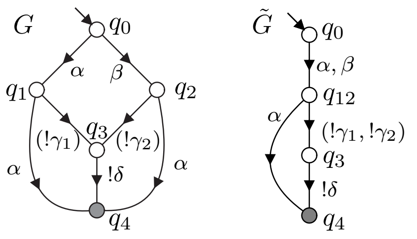

Fig. 4. Uncontrollable observation equivalence has more abstraction potential than observer projection.

E. Relationship to Projection

In related work [7], [8], natural projection is used to simplify subsystems and perform modular synthesis. It is well-known that, in general, natural projection of local events in a subsystem cannot ensure the preservation of a global synthesis result. In [7], it is shown that the synthesis result is preserved if the projection satisfies two additional requirements known as the *observer property* and *output control consistency*. The condition of output control consistency is relaxed to *local control consistency* in [8].

In the following, it is shown that observation equivalence-based abstractions have a higher abstraction potential than methods based on natural projection, and that every natural projection that satisfies the observer property and local control consistency leads to an abstraction that can also be achieved using synthesis observation equivalence.

Definition 14: Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ be an automaton, and $\Sigma = \Omega \,\dot\cup\, \Upsilon$. The natural projection $P_\Omega \colon \Sigma^* \rightarrow \Omega^*$ is

- an *observer* for G, if for all $s, s', t \in \Sigma^*$ and all states $x \in Q$ such that $P_\Omega(s) = P_\Omega(s')$, $G \xrightarrow{st\omega}$, and $G \xrightarrow{s'} x$, there exists $t' \in \Sigma^*$ such that $P_\Omega(t') = P_\Omega(t)$ and $x \xrightarrow{t'\omega}$; [18]
- *locally control-consistent (LCC)* for G, if for all $s \in \Sigma^*$, all $\upsilon \in \Omega \cap \Sigma_u$, and all states $x \in Q$ such that $G \xrightarrow{s} x$ and $P_\Omega(s)\upsilon \in P_\Omega\mathcal{L}(G)$, if there exists $t \in \Upsilon^*$ such that $x \xrightarrow{t\upsilon}$ then there also exists $u \in (\Upsilon \cap \Sigma_u)^*$ such that $x \xrightarrow{u\upsilon}$. [8]

Natural projection is a language-theoretic operation, which can be applied to automata using the standard algorithms of *subset construction* and *minimisation* [19]. Alternatively, natural projection can be seen to induce a state equivalence relation on nondeterministic automata using *Nerode equivalence* [1].

Definition 15: Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ and $\Omega \subseteq \Sigma$. The natural projection $P_\Omega \colon \Sigma^* \rightarrow \Omega^*$ induces the *Nerode equivalence* modulo Ω on the state set Q:

$$x \equiv_\Omega y \quad \text{if and only if} \quad P_\Omega\mathcal{L}(x) = P_\Omega\mathcal{L}(y) \ . \qquad (10)$$

It is known that Nerode equivalence implies observation equivalence if the projection satisfies the observer property [18], [20]; in this case, the quotient $G/{\equiv_\Omega}$ is a candidate for abstraction of G. On the other hand, not every observation equivalence abstraction can be expressed using projection.

Example 4: Consider automaton G in Fig. 4. Hiding the local uncontrollable events $!\gamma_1$ and $!\gamma_2$ does not yield an observer projection, because event α is enabled in the source states q_1 and q_2, but not in the target state q_3 of the local transitions $!\gamma_1$ and $!\gamma_2$. Nevertheless, states q_1 and q_2 are uncontrollable observation equivalent and can be merged, producing the abstraction $\tilde{G}$ such that $G \lesssim_{\{!\gamma_1, !\gamma_2\}} \tilde{G}$.

This example shows that uncontrollable observation equivalence can perform abstractions that are not possible using natural projection. In addition to removing events, under certain conditions events can also be identified and merged, and synthesis observation equivalence provides some conditions under which this is possible.

The following result shows that every abstraction obtained by a projection with the observer and LCC properties induces a synthesis observation equivalence abstraction, for *nonblocking* automata. Blocking states can always be merged while ensuring synthesis abstraction, but this cannot be done via observation equivalence. For blocking automata, the relationship to projection is similar to the results in [18].

Theorem 7: Let $G = \langle \Sigma, Q, \rightarrow, Q^\circ \rangle$ be a reachable and nonblocking automaton, and let $\Sigma = \Omega \,\dot\cup\, \Upsilon$ such that $P_\Omega \colon \Sigma^* \rightarrow \Omega^*$ is an observer and LCC for G. Then $\equiv_\Omega$ is a synthesis observation equivalence on G with respect to Υ.

In combination with example 4, this result confirms that synthesis observation equivalence can perform more abstraction than the projection-based method of [8].

V. Example

In this section, the proposed method is applied to a manufacturing example previously studied in [21]. Fig. 5 gives an overview of the system, and Fig. 6 shows automata models. The manufacturing system consists of two machines (M_1 and M_2) for processing workpieces and four subsystems ($\mathbf{Sub}_1, \ldots, \mathbf{Sub}_4$) for moving and buffering workpieces in transit between the machines. Each subsystem $\mathbf{Sub}_i$ consists of a buffer (B_i) that can store up to two workpieces, and a handler (H_i) that fetches a workpiece from a machine and puts it into the buffer.

The manufacturing system can produce two types of workpieces. Type I workpieces are first processed by M_1 (action input_1). Then they are passed through $\mathbf{Sub}_1$: they are fetched by H_1 (fetch_1) and placed into B_1 (put_1). Next, they are processed by M_2 (get_1), fetched by H_4 (fetch_4) in $\mathbf{Sub}_4$ and placed into B_4 (put_4). Finally, they are processed by M_1

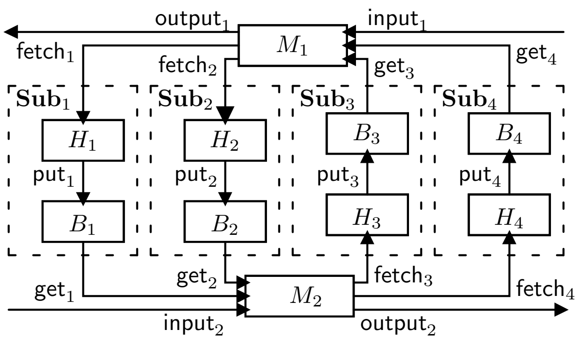

Fig. 5. Manufacturing system overview.

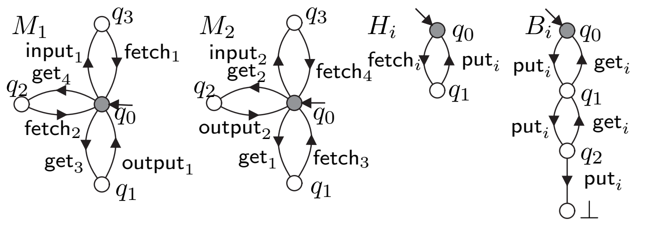

Fig. 6. Automata for manufacturing example.

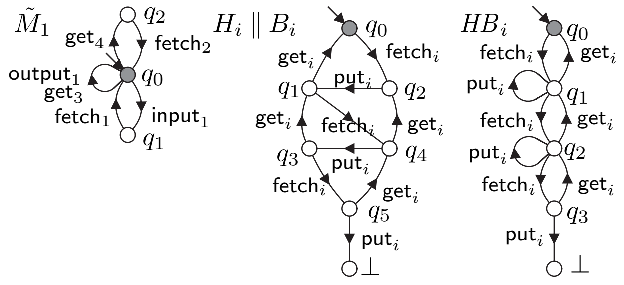

Fig. 7. Abstractions of manufacturing system.

once more (get$_4$), and released (output$_1$). Similarly, type II workpieces are first processed by M_2, passed through $\mathbf{Sub}_3$, further processed by M_1, passed through $\mathbf{Sub}_2$, and finally processed by M_2.

In the first step of compositional synthesis, events output$_1$ and output$_2$ are controllable local events in M_1 and M_2. Since both conditions of synthesis observation equivalence are fulfilled, it can be applied to M_1 and M_2, resulting in abstractions $\tilde{M}_1$ and $\tilde{M}_2$. Fig. 7 shows $\tilde{M}_1$; $\tilde{M}_2$ is similar.

The remaining automata cannot be abstracted, so the next step is to compose the automata in each subsystem. After composing H_i and B_i, event put$_i$ becomes an uncontrollable local event, and uncontrollable observation equivalence becomes applicable. The composition $H_i \parallel B_i$ and the resulting abstracted automaton, HB_i, are shown in Fig. 7. HB_i is obtained by merging q_1 with q_2 and q_3 with q_4.

The final step of compositional synthesis is to compute a supervisor for $\tilde{M}_1$, $\tilde{M}_2$, and HB_i, $i = 1, \ldots, 4$. The calculated supervisor $\tilde{S}$ has 685 states, and the modular supervisor for the system is $\tilde{S} \parallel B_1 \parallel B_2 \parallel B_3 \parallel B_4$. Composing this supervisor with the system results in the least restrictive monolithic supervisor for the system, an automaton with 9216 states.

VI. Conclusions

Three variations of observation equivalence are investigated for their applicability in the compositional synthesis framework of *synthesis abstraction* [13], which allows the synthesis of least restrictive modular supervisors for discrete event systems. While standard observation equivalence is not useful for synthesis abstraction, the stronger conditions of *bisimulation*, *uncontrollable observation equivalence*, and *synthesis observation equivalence* can be shown to preserve synthesis abstraction and guarantee the construction of a correct modular supervisor. It is also shown that observation equivalence based synthesis abstraction provides more abstraction than natural projection using local control consistency [8], and an example demonstrates the potential of state-space reduction using the proposed abstractions.

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] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Kluwer, Sept. 1999.

[3] K. C. Wong and W. M. Wonham, "Modular control and coordination of discrete-event systems," *Discrete Event Dyn. Syst.*, vol. 8, no. 3, pp. 247–297, Oct. 1998.

[4] R. Song and R. J. Leduc, "Symbolic synthesis and verification of hierarchical interface-based supervisory control," in *Proc. 8th Int. Workshop on Discrete Event Systems, WODES '06*, Ann Arbor, MI, USA, July 2006, pp. 419–426.

[5] K. Åkesson, H. Flordal, and M. Fabian, "Exploiting modularity for synthesis and verification of supervisors," in *Proc. 15th IFAC World Congress on Automatic Control*, Barcelona, Spain, 2002.

[6] R. Su and W. M. Wonham, "Supervisor reduction for discrete-event systems," *Discrete Event Dyn. Syst.*, vol. 14, no. 1, pp. 31–53, Jan. 2004.

[7] 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, July 2006, pp. 3–8.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] S. Mohajerani, R. Malik, S. Ware, and M. Fabian, "Three variations of observation equivalence preserving synthesis abstraction," Working Paper 01/2011, Dept. of Computer Science, University of Waikato, Hamilton, New Zealand, 2011.

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

[16] R. Milner, *Communication and concurrency*, ser. Series in Computer Science. Prentice-Hall, 1989.

[17] R. Malik, D. Streader, and S. Reeves, "Fair testing revisited: A process-algebraic characterisation of conflicts," in *Proc. 2nd Int. Symp. Automated Technology for Verification and Analysis, ATVA 2004*, ser. LNCS, F. Wang, Ed., vol. 3299. Taipei, Taiwan: Springer, Oct.–Nov. 2004, pp. 120–134.

[18] R. Malik, H. Flordal, and P. N. Pena, "Conflicts and projections," in *Proc. 1st IFAC Workshop on Dependable Control of Discrete Systems, DCDS '07*, Paris, France, June 2007, pp. 63–68.

[19] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2001.

[20] K. C. Wong and W. M. Wonham, "On the computation of observers in discrete-event systems," *Discrete Event Dyn. Syst.*, vol. 14, no. 1, pp. 55–107, 2004.

[21] 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.