# Compositional Synthesis of Maximally Permissive Supervisors using Supervision Equivalence

Hugo Flordal*
Department of Signals and Systems
Chalmers University of Technology
Göteborg, Sweden

Robi Malik
Department of Computer Science
University of Waikato
Hamilton, New Zealand

Martin Fabian
Department of Signals and Systems
Chalmers University of Technology
Göteborg, Sweden

Knut Åkesson
Department of Signals and Systems
Chalmers University of Technology
Göteborg, Sweden

November 1, 2007

**Abstract**

This paper presents a general framework for efficient synthesis of supervisors for discrete event systems. The approach is based on compositional minimisation, using concepts of process equivalence. In this context, a large number of ways are suggested how a finite-state automaton can be simplified such that the results of supervisor synthesis are preserved. The proposed approach yields a compact representation of a least restrictive supervisor that ensures controllability and nonblocking. The method is demonstrated on a simple manufacturing example to significantly reduce the number of states constructed for supervisor synthesis.

## 1 Introduction

This paper proposes a solution to the problem of *supervisor synthesis* in *supervisory control theory of discrete-event systems* (Ramadge and Wonham, 1989),

*Corresponding author: `flordal@chalmers.se` focusing on very large synthesis problems. The standard (monolithic) way to synthesise a controllable and nonblocking supervisor is to build the synchronous composition of all components and search the state-space. This method is known to suffer from the *state-space explosion problem* and therefore is only feasible for small systems.

For larger systems, *modular* approaches to supervisor synthesis are of great interest and have long been studied in supervisory control theory (Ramadge and Wonham, 1989; Wong and Wonham, 1998). Most of the approaches studied so far rely on structure to be provided by users (Song and Leduc, 2006) and hence are hard to automate. Those that can be automated either do not consider nonblocking, or are guaranteed to produce a least restrictive supervisor only under certain constraints (Brandin, Malik and Malik, 2004; Åkesson, Flordal and Fabian, 2002; Lin and Wonham, 1990; Queiroz and Cury, 2000).

An interesting approach to create equivalent simpler supervisors from a given monolithic supervisor has been proposed in (Su and Wonham, 2004). While the results are very impressive, the method relies on a monolithic supervisor to be constructed first, and thus remains limited by its size.

An alternative approach is introduced in (Feng and Wonham, 2006; Hill and Tilbury, 2006), where language projection is used to simplify finite-state machines during synthesis and to construct modular supervisors. To ensure that nonblocking and maximal permissiveness are preserved, the *observer property* and *output-control consistency* are imposed as additional requirements on the projection. Unfortunately, these conditions are quite strong and allow for only little simplification.

Using ideas of process equivalence (De Nicola and Hennessy, 1984), this present paper proposes a different framework for *compositional* synthesis of least restrictive controllable and nonblocking supervisors, which can be fully automated. The method efficiently handles supervisory control problems given as a large set of small finite-state automata by compositionally calculating abstractions. Automata are simplified while the supervisor is computed, avoiding construction of an overall monolithic supervisor, and the limits of natural projection are overcome using nondeterministic automata to represent intermediate results.

This paper is an extended version of (Flordal and Malik, 2006), with more detailed proofs and an elaborate set of rewrite procedures for finite-state automata that leave the synthesis results unchanged. Section 2 demonstrates the proposed method using a simple example. Section 3 provides the formal notation for automata and supervisory control, and section 4 explains the synthesis framework in detail, with a formal proof of its soundness. Section 5 gives a set of reduction rules for making abstractions, section 6 gives some preliminary experimental results, and finally section 7 contains some concluding remarks.

## 2 Motivating Example

This section demonstrates the ideas of the new synthesis procedure using a simple manufacturing example. The following text assumes familiarity with supervisory control theory, (Ramadge and Wonham, 1989; Cassandras and Lafortune, 1999).

Figure 1 shows an automata model of an industrial transfer line, originally

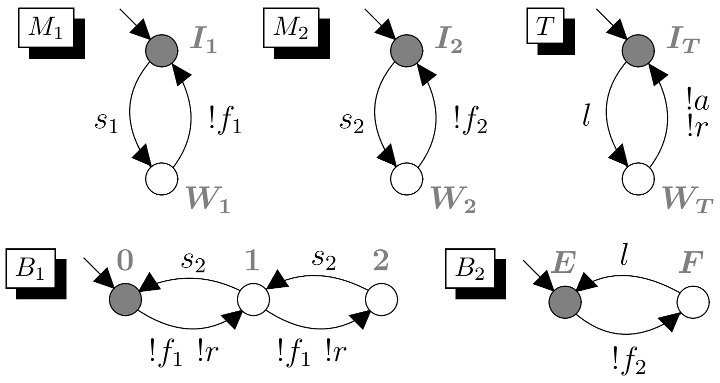


Figure 1: The transfer line example.

given in (Wonham, 2006). Automata $M_1$, $M_2$, and $T$ constitute the plant model, while $B_1$ and $B_2$ are specifications. The automata are assumed to interact in lock-step synchronisation, (Hoare, 1985). Uncontrollable events are prefixed by an exclamation mark (!), marked states are shaded.

This synthesis procedure presupposes the system model to be given as a set of *plant* models only. Therefore, the buffer *specifications* $B_1$ and $B_2$ are first transformed into plants $B_1^\perp$ and $B_2^\perp$. This is straightforward: wherever an uncontrollable event is disabled, a transition on that event to a dump state $\perp$ is added. The result is shown in figure 2. As shown later in section 3.4, this transformation produces an equivalent supervisory control problem if both controllability and nonblocking are considered.

The compositional synthesis is performed as a series of small steps that in the end result in a compact representation of the least restrictive supervisor. The intermediate steps strive to avoid the state-space explosion by simplifying parts of the system while preserving all information necessary for the synthesis. In the terminology to follow, the simplified part must be *supervision equivalent* to the original part.

For example, when composing the components $M_2$ and $B_2^\perp$, the uncontrollable event $f_2$ becomes local to the subsystem $M_2 \parallel B_2^\perp$. That is, transitions associated with $f_2$ can never be disabled in future compositions with other components. Therefore, the identity of event $f_2$ is no longer important, so all its occurrences can be replaced by $\tau_u$, an identity-less uncontrollable event. In

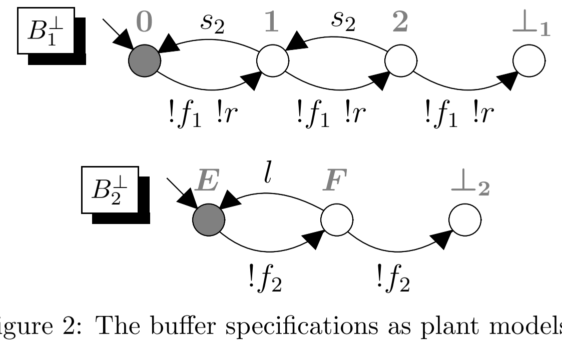


Figure 2: The buffer specifications as plant models.

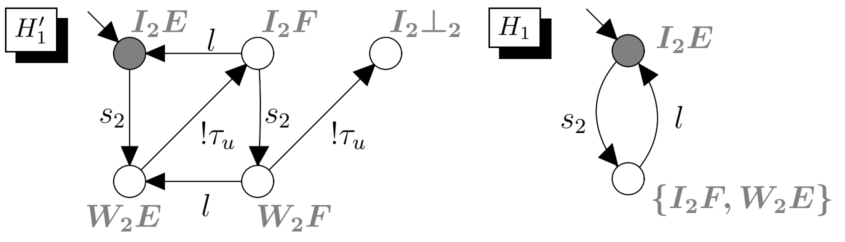


Figure 3: The simplification of $H_1' = (M_2 \parallel B_2^{\bot}) \setminus_! \{f_2\}$ (to the left) results in the automaton $H_1$ (to the right).

a similar fashion, all local controllable events are replaced by the identity-less controllable event $\tau_c$. The resultant automaton, $H_1' = (M_2 \parallel B_2^{\bot}) \setminus_! \{f_2\}$, where $\setminus_!$ denotes this special kind of event *hiding*, is shown in figure 3. For later use in supervision, the states of this automaton are labelled by pairs of states of the automata that were composed originally. For example, $I_2F$ represents a state where automaton $M_2$ is in state $I_2$, and $B_2^{\bot}$ is in state $F$.

After hiding, the five-state automaton $H_1'$ can be replaced by the supervision equivalent two-state automaton $H_1$, also shown in figure 3. To see that these automata are supervision equivalent, consider a supervisor that is to enforce controllability and nonblocking for the system. Clearly, all global states where this subsystem is in state $I_2\bot_2$ must be avoided since these are blocking. Also states involving $W_2F$ must be avoided, since there is an outgoing uncontrollable transition leading to a blocking state that cannot be disabled by the rest of the system. Thus, already at this point it is clear that the controllable transition from $I_2F$ to $W_2F$ must be (and can be) prevented by a supervisor—states $I_2\bot_2$ and $W_2F$ can be removed. Furthermore, states $I_2F$ and $W_2E$ can be merged into a single state since a supervisor that allows the plant to reach $W_2E$ cannot do anything but accept that the plant may uncontrollably transit to $I_2F$. The resulting state is labelled $\{I_2F, W_2E\}$ to reflect the fact that it has been constructed from these two state pairs.

Figures 4 and 5 show further simplification results, $H_2$ derives from the composition of $M_1$ and $B_1^{\bot}$, and $H_3$ derives from the composition of $H_1$ and $T$. State labels are propagated using the Cartesian product, to represent all possible combinations of states. For example, $\{I_2F, W_2E\} \times \{I_T\} = \{I_2FI_T, W_2EI_T\}$ represents two possible state combinations of the original automata $M_2$, $B_2^{\bot}$, and $T$.

Figure 6 shows the end result $H$, a simplified version of $(H_2 \parallel H_3) \setminus_! \{s_2, r\}$.

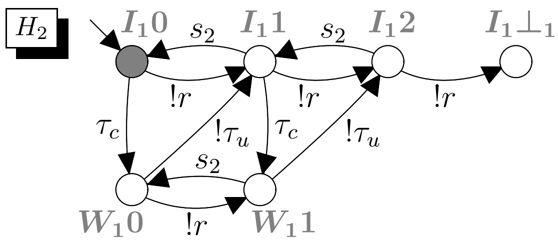


Figure 4: The result of the simplification of $(M_1 \parallel B_1^{\bot}) \setminus_! \{f_1, s_1\}$.

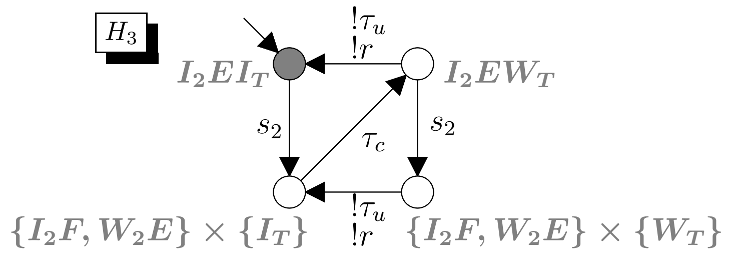


Figure 5: The result of the simplification of $(H_1 \parallel T) \setminus_! \{a, l\}$.

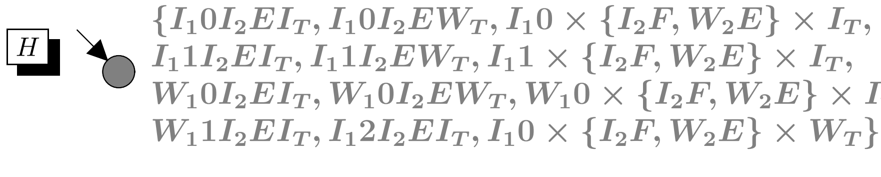
In the last step, there a no events left to be shared with other components, so all events can be hidden. The final result can always be reduced to an automaton with a single state if controllable and nonblocking supervision is possible. Otherwise the result is the null automaton, an automaton with no states.

In the process of producing the final result, the largest intermediate automaton, $H_2 \| H_3$, has 21 states and 45 transitions. These figures should be compared to the corresponding values for the monolithic approach, which calculates the supervisor directly from the composed system $G = M_1 \parallel B_1 \parallel M_2 \parallel B_2 \parallel T$, an automaton with 48 states and 120 transitions.

The objective of supervision is to avoid "bad" states, i.e., states that are blocking or uncontrollable either by themselves or as a consequence of other states being "bad". Given the final result $H$ and the original automata $M_1$, $B_1$, $M_2$, $B_2$, and $T$, it is possible to calculate the supervisor function that yields the least restrictive behaviour that is controllable and nonblocking. The original automata are used to observe the system and determine the current global state, and $H$ can be used to determine whether any states that can be reached from the current state are "bad", which calls for event disablement.

To make it possible to identify "bad" states, at each stage in the construction of $H$, a clear correspondence between the states of the intermediate and the original automata is maintained. This correspondence is propagated and stored using *labels* in such a way that the single state of $H$ has labels representing all states that are reachable under a least restrictive supervisor.

For instance, assume the transfer line system is in the global state $I_12I_2EI_T$. An inspection of the modular model shows two possible transitions, associated with controllable events $s_1$ and $s_2$. Event $s_1$ would lead the system to state $W_12I_2EI_T$, but since its label *cannot* be found in $H$ (actually no label starting with "$W_12$" can be found there) the supervisor *disables* $s_1$. Event $s_2$, on the other hand, leads to state $I_11W_2EI_T$ whose label *can* be found in $H$ (on the second line in figure 6 if the last expression is unfolded). Therefore, the



$H$
$\{I_10I_2EI_T, I_10I_2EW_T, I_10 \times \{I_2F, W_2E\} \times I_T,$
$I_11I_2EI_T, I_11I_2EW_T, I_11 \times \{I_2F, W_2E\} \times I_T,$
$W_10I_2EI_T, W_10I_2EW_T, W_10 \times \{I_2F, W_2E\} \times I_T,$
$W_11I_2EI_T, I_12I_2EI_T, I_10 \times \{I_2F, W_2E\} \times W_T\}.$

Figure 6: The result of the simplification of $(H_2 \parallel H_3) \setminus_! \{s_2, r\}$.

supervisor *enables* $s_2$.

This may seem to imply an explosion in the amount of labels of the same size as the dreaded state-space explosion, but the labels can be stored efficiently without additional effort. Instead of propagating the labels explicitly throughout the process, at each intermediate step, a map can be stored that relates the labels of each component to those of its simplified version. For example, the subterm $\{I_2F, W_2E\} \times \{I_T\}$ that occurs three times in figure 6 can be replaced by a reference to the corresponding state in the intermediate automaton $H_3$ in figure 5.

In this way, each state of a simplified automaton is labelled by a set of state tuples of the intermediate results from which it was composed. These label sets do not need to be larger than the corresponding automata. In this way, a supervisor can be implemented using data structures that do not exceed the size of the automata constructed during the synthesis process.

## 3 Notation and Preliminaries

This section presents the notation and mathematical framework used in this paper. The notation in this paper is essentially standard supervisory control theory notation (Wonham, 2006) that has been enriched by some notation borrowed from process calculus (Milner, 1989).

### 3.1 Events and Strings

Event sequences and languages are a simple means to describe discrete system behaviours. Their basic building blocks are *events*, which are taken from a finite *alphabet* $\Sigma$. For the purpose of supervisory control, the event alphabet $\Sigma$ is partitioned into the set $\Sigma_c$ of *controllable* events and the set $\Sigma_u$ of *uncontrollable* events. There are two special events, the *silent* controllable event $\tau_c$ and the *silent* uncontrollable event $\tau_u$. These do not belong to either of $\Sigma$, $\Sigma_c$, and $\Sigma_u$. If they are to be included, the alphabets $\Sigma_\tau = \Sigma \cup \{\tau_c, \tau_u\}$, $\Sigma_{\tau,c} = \Sigma_c \cup \{\tau_c\}$ and $\Sigma_{\tau,u} = \Sigma_u \cup \{\tau_u\}$ are used instead.

$\Sigma^*$ denotes the set of all finite *strings* of the form $\sigma_1\sigma_2\cdots\sigma_k$ of events from $\Sigma$, including the *empty string* $\varepsilon$. The *catenation* of two strings $s, t \in \Sigma^*$ is written as $st$.

### 3.2 Nondeterministic Automata

System behaviours are represented using finite-state automata. Nondeterminism is used to support hiding, which is essential for the proposed synthesis approach.

**Definition 1** A (nondeterministic) *finite-state automaton* is a 5-tuple $G = \langle Q, \Sigma, \rightarrow, Q^i, Q^m \rangle$, where $\Sigma$ is a finite alphabet of events, $Q$ is a finite set of *states*, $\rightarrow \subseteq Q \times \Sigma_\tau \times Q$ is the *state transition relation*, $Q^i \subseteq Q$ is the set of *initial* states, and $Q^m \subseteq Q$ is the set of *marked* states. ■

Note that the silent events $\tau_c$ and $\tau_u$ are allowed in $\rightarrow$ even though they are never included in the alphabet of an automaton.

The transition relation is written in infix notation $p \xrightarrow{\sigma} q$, and is extended to strings in $\Sigma_\tau^*$ by letting

$$p \xrightarrow{\varepsilon} p \qquad \text{for all } p \in Q \ ; \tag{1}$$
$$p \xrightarrow{s\sigma} q \qquad \text{if } p \xrightarrow{s} r \text{ and } r \xrightarrow{\sigma} q \text{ for some } r \in Q \ . \tag{2}$$

For state sets $Q_1, Q_2 \subseteq Q$, $Q_1 \xrightarrow{s} Q_2$ denotes the existence of $q_1 \in Q_1$ and $q_2 \in Q_2$ such that $q_1 \xrightarrow{s} q_2$. Similarly, $p \rightarrow q$ means that there exists a string $s \in \Sigma_\tau^*$ such that $p \xrightarrow{s} q$. Finally, $p \xrightarrow{s}$ denotes that there exists a state $q$ such that $p \xrightarrow{s} q$, and for an automaton $G$, $G \xrightarrow{s} q$ means $Q^i \xrightarrow{s} q$.

For $P \subseteq Q$, the notation $P$, pronounced $\rightarrow$ restricted to (the state set) $P$ is defined as

$$P = \{ \langle p, \sigma, q \rangle \in \rightarrow \mid \text{such that } p, q \in P \} \ . \tag{3}$$

That is, $P$ is the portion of $\rightarrow$ for which both the source and the target states are in $P$.

A state $q$ is called *reachable* in an automaton $G$ if $G \rightarrow q$; if this holds for all $q \in Q$, then $G$ is called reachable. If $G$ is not reachable, it can easily be made so by removing all states that are not reachable. Therefore, in the following, all automata are assumed to be reachable.

**Definition 2** An automaton $G$ is said to be *deterministic* if $Q^i$ is a singleton, $p \xrightarrow{\sigma} q_1$ and $p \xrightarrow{\sigma} q_2$ implies $q_1 = q_2$, and $\rightarrow$ contains no transitions labelled $\tau_c$ or $\tau_u$. ■

Various operations can be used to modify or combine automata. For compositional synthesis, synchronous composition (Hoare, 1985) and event hiding are the most important.

**Definition 3** Let $G_1 = \langle Q_1, \Sigma, \rightarrow_1, Q_1^i, Q_1^m \rangle$ and $G_2 = \langle Q_2, \Sigma, \rightarrow_2, Q_2^i, Q_2^m \rangle$ be two automata using the same alphabet. The *synchronous product* of $G_1$ and $G_2$ is

$$G_1 \parallel G_2 \ = \ \langle Q_1 \times Q_2, \Sigma, \rightarrow, Q_1^i \times Q_2^i, Q_1^m \times Q_2^m \rangle \tag{4}$$

where

$$(p, q) \xrightarrow{\sigma} (p', q') \text{ if } \sigma \in \Sigma,\ p \xrightarrow{\sigma}_1 p' \text{ and } q \xrightarrow{\sigma}_2 q' \ ;$$
$$(p, q) \xrightarrow{\sigma} (p', q) \text{ if } \sigma \in \{\tau_c, \tau_u\} \text{ and } p \xrightarrow{\sigma}_1 p' \ ;$$
$$(p, q) \xrightarrow{\sigma} (p, q') \text{ if } \sigma \in \{\tau_c, \tau_u\} \text{ and } q \xrightarrow{\sigma}_2 q' \ .$$
■

Note that no generality is lost when assuming the alphabets to be equal. If the two automata to be combined do not use the same alphabet, they must first be *extended* to their united alphabet. An automaton using alphabet $\Sigma$ is extended to $\Sigma' \supseteq \Sigma$ by adding *selfloops* $q \xrightarrow{\sigma} q$ for all states $q \in Q$ and all events $\sigma \in \Sigma' \setminus \Sigma$, i.e., all new events not in $\Sigma$.

**Definition 4** Let $G = \langle Q, \Sigma, \rightarrow, Q^i, Q^m \rangle$ be an automaton, and let $\Upsilon \subseteq \Sigma$. The result of *controllability preserving hiding*, hiding henceforth, of $\Upsilon$ from $G$ is

$$G \setminus_! \Upsilon \ = \ \langle Q, \Sigma \setminus \Upsilon, \rightarrow_!, Q^i, Q^m \rangle \tag{5}$$

where $\rightarrow_!$ is obtained from $\rightarrow$ by replacing each transition $p \xrightarrow{\sigma} q$ such that $\sigma \in \Upsilon$ by $p \xrightarrow{\tau_c} q$ if $\sigma \in \Sigma_c$ or by $p \xrightarrow{\tau_u} q$ if $\sigma \in \Sigma_u$. ■

Hiding removes the identity of the events in $\Upsilon$ and in general produces a nondeterministic automaton.

By introducing concepts of *subautomata* and *union* of automata, the set of automata can be considered as a lattice.

**Definition 5** Let $G_1 = \langle Q_1, \Sigma, \to_1, Q^i, Q_1^m \rangle$ and $G_2 = \langle Q_2, \Sigma, \to_2, Q^i, Q_2^m \rangle$ be two automata with the same alphabet and set of initial states. $G_1$ is a *subautomaton* of $G_2$, written $G_1 \subseteq G_2$, if $Q_1 \subseteq Q_2$, $\to_1 \subseteq \to_2$, and $Q_1^m \subseteq Q_2^m$. ■

**Definition 6** Let $G_j = \langle Q_j, \Sigma, \to_j, Q^i, Q_j^m \rangle$, $j \in J$ be a family of automata all having the same alphabet and set of initial states. Define

$$\bigcup_{j\in J} G_j = \langle \bigcup_{j\in J} Q_j, \Sigma, \bigcup_{j\in J} \to_j, Q^i, \bigcup_{j\in J} Q_j^m \rangle \ . \qquad ■$$

This definition only makes sense if all the different $G_j$ are subautomata of the same automaton $G$, which will also be the case when it is used later.

## 3.3 Supervision and Synthesis

Supervisors are used to restrict the behaviour of systems represented by automata. A supervisor observes the sequence of events occurring in the system (the *plant*) and then enables or disables certain controllable events, but cannot disable any uncontrollable events. Formally, this can be considered as a map $S$, where $S(s)$ represents the set of events enabled by the supervisor after observing the system execute the string $s$.

**Definition 7** Let $G$ be an automaton with alphabet $\Sigma$. A *supervisor* for $G$ is a map $S\colon \Sigma^* \to 2^\Sigma$, such that $S(s) \supseteq \Sigma_u$ for all $s \in \Sigma^*$. ■

Given a plant $G$, and a sub-automaton $K$ of $G$ representing a *desired behaviour*, it is of interest to construct a supervisor for $G$ that produces exactly the same behaviour as $K$. Supervisory control theory shows that $K$ has to be *controllable* for such a supervisor to exist (Ramadge and Wonham, 1989; Cassandras and Lafortune, 1999). Below are two definitions of controllability used for the nondeterministic setting of this paper.

**Definition 8** Let $G$ and $K$ be two automata using the same alphabet $\Sigma$. $K$ is *controllable with respect to* $G$ if, for every string $s \in \Sigma^*$, every state $q$ of $K$, and every uncontrollable event $\upsilon \in \Sigma_u$ such that $K \xrightarrow{s} q$ and $G \xrightarrow{s\upsilon}$, it holds that $q \xrightarrow{\upsilon}$ in $K$. ■

**Definition 9** Let $G$ be an automaton, and let $K \subseteq G$ be a subautomaton of $G$. $K$ is *controllable in* $G$ if, for every string $s \in \Sigma_\tau^*$, every uncontrollable event $\upsilon \in \Sigma_{\tau,u}$, and all states $p, q \in Q$ such that $K \xrightarrow{s} p$ and $G \xrightarrow{s} p \xrightarrow{\upsilon} q$, it holds that $K \xrightarrow{s} p \xrightarrow{\upsilon} q$. ■

For deterministic systems, definition 8 corresponds to the original controllability definition from (Ramadge and Wonham, 1989). The two definitions above coincide when $K$ is a subautomaton of a deterministic automaton $G$.

In the nondeterministic case, definition 9 assumes that a supervisor can disable each (controllable) *transition* individually unlike in traditional supervisory control. For example, consider a controllable string $s$ that can take $G$ to two different states $p_1$ and $p_2$ but can take only $K$ to $p_1$. That is, $K$ has disabled a nondeterministic choice somewhere. Then, assuming there is no other way to reach $p_2$, $p_2$ does not matter for controllability according to the definition.

Here, definition 9 makes sense because nondeterministic automata always derive from deterministic automata, and the supervisor is assumed to distinguish different transitions using its knowledge about the global state. In the following, be aware of the difference between controllable *with respect to* and controllable *in*.

In addition to controllability, the behaviour of a supervised system is typically also required to be nonblocking (Ramadge and Wonham, 1989; Malik, Streader and Reeves, 2006).

**Definition 10** An automaton $G$ is called *nonblocking* if, for every state $q$ such that $G \rightarrow q$ it holds that $q \rightarrow Q^m$. ■

Similar to traditional supervisory control theory (Ramadge and Wonham, 1989; Fabian, 1995), it can be shown that the union of controllable and nonblocking subautomata of a given automaton is again controllable and nonblocking. This justifies the following definition.

**Definition 11** Let $G$ be an automaton. The supremal controllable and nonblocking subautomaton of $G$ is

$$\sup\mathcal{CN}(G) = \bigcup \{\, G' \subseteq G \mid G' \text{ is controllable in } G \text{ and nonblocking} \,\} . \quad ■$$

The supremal controllable and nonblocking subautomaton can be calculated by iteratively identifying uncontrollable and blocking states until a *fixed point* is reached. Given an automaton $G = \langle Q, \Sigma, \rightarrow, Q^i, Q^m \rangle$ and a state set $P \subseteq Q$, each step in the iteration calculates a subset of $P$ that has some uncontrollable and blocking states removed. The function that does this will be denoted $\Theta_G(P) = \Theta_G^{\mathrm{cont}}(P) \cap \Theta_G^{\mathrm{nonb}}(P)$ where

$$\Theta_G^{\mathrm{cont}}(P) = \{\, q \in P \mid \forall \sigma \in \Sigma_{\tau,u},\ q \xrightarrow{\sigma} p \text{ implies } p \in P \,\} ; \quad (6)$$

$$\Theta_G^{\mathrm{nonb}}(P) = \{\, q \in P \mid \exists p \in Q^m,\ qPp \,\} . \quad (7)$$

The first of these functions considers controllability problems caused by single uncontrollable transitions, and the second function considers states that are blocking when the transition function is restricted to states in $P$. Since the supervisor can distinguish transitions of nondeterministic automata based on its knowledge of the deterministic model, these synthesis operators disable individual transitions and not just events.

Given an automaton $G$, let $\hat{P}$ be the fixed point of the iteration $P^0 = Q$, $P^{i+1} = \Theta_G(P^i)$. Then the synthesis result $\sup\mathcal{CN}(G)$ for $G$ is the portion of $G$ that remains when the transition function is restricted to $\hat{P}$ (the non-reachable parts can be removed), or the null automaton if some of the initial states have been removed:

$$\sup\mathcal{CN}(G) = \begin{cases} \langle \hat{P}, \Sigma, \hat{P}, Q^i, Q^m \cap \hat{P} \rangle & \text{if } Q^i \subseteq \hat{P} ; \\ \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle & \text{otherwise} . \end{cases} \quad (8)$$

Calculating $\sup\mathcal{CN}(G)$ for some $G$ is called *synthesis*. Unlike traditional synthesis (Ramadge and Wonham, 1989), $\sup\mathcal{CN}(G)$ merely describes a controllable and nonblocking sub-behaviour of the given *plant G*. *Specifications* are not considered here since, as will be shown shortly, they can easily be translated into plants if both controllability and nonblocking is considered in the synthesis.

## 3.4 Translation of Specifications into Plants

The compositional procedure can be simplified considerably if it has only one kind of automata to consider. Therefore, this section suggests a way how general supervisory control problems can be expressed using plants only. The resultant models always require synthesis for nonblocking in addition to controllability, which may be more difficult than synthesis for controllability only. Therefore, this method is of most interest for synthesis problems that involve nonblocking from the start. Modular synthesis involving only controllability is discussed in (Brandin et al., 2004; Åkesson et al., 2002).

A specification automaton can be transformed into a plant by adding, for every uncontrollable event that is not enabled in a state, a transition to a new blocking state $\perp$. The result of synthesis remains the same after this transformation.

**Definition 12** Let $K = \langle Q, \Sigma, \rightarrow, Q^i, Q^m \rangle$ be a specification. The *complete plant automaton* $K^\perp$ for $K$ is

$$K^\perp = \langle Q \cup \{\perp\}, \Sigma, \rightarrow^\perp, Q^i, Q^m \rangle \tag{9}$$

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

$$\rightarrow^\perp = \rightarrow \cup \{ \langle q, \upsilon, \perp \rangle \mid q \in Q, \upsilon \in \Sigma_u, q \overset{\upsilon}{\not\rightarrow} \} . \qquad \blacksquare$$

Whenever the specification disallows an uncontrollable event, the corresponding state in the complete plant automaton has an uncontrollable transition to a blocking state. In a controllable model, these transitions are removed by synchronous composition with the original plants; in an uncontrollable model, they cause the synchronous product to be blocking. Thus, all controllability problems are replaced by blocking problems.

**Proposition 1** Let $G$, $K$, and $K'$ be deterministic automata with the same alphabet $\Sigma$. Then the following two statements are equivalent.

(i) $K' \subseteq G \parallel K^\perp$ is nonblocking and controllable in $G \parallel K^\perp$.

(ii) $K' \subseteq G \parallel K$ is nonblocking and controllable with respect to $G$. $\square$

**Proof.** First, assume that (i) holds. Since, by the assumption, $K'$ is nonblocking, it holds that $K' \not\rightarrow (q, \perp)$ for every state $q$ in $G$. Thus, since $K^\perp$ is the complete plant automaton for $K$, $K' \subseteq G \parallel K^\perp$ implies $K' \subseteq G \parallel K$.

It remains to show that $K'$ is controllable with respect to $G$. Let $s \in \Sigma^*$ and $\upsilon \in \Sigma_u$ such that $G \xrightarrow{s} p_G \xrightarrow{\upsilon} q_G$ and $K' \xrightarrow{s} (p_G, p_K)$. Since $K' \subseteq G \parallel K^\perp$ it holds that $K^\perp \xrightarrow{s} p_K$. Since $\upsilon \in \Sigma_u$ and since $K^\perp$ is the complete plant automaton for $K$, there exists a state $q^\perp$ such that $K^\perp \xrightarrow{s} p_K \xrightarrow{v} q^\perp$. This implies $G \parallel K^\perp \xrightarrow{s} (p_G, p_K) \xrightarrow{v} (q_G, q^\perp)$. Since $K'$ is controllable in $G \parallel K^\perp$, it holds that $(p_G, p_K) \xrightarrow{v}$ in $K'$.

Second, assume that (ii) holds. Clearly, since $K \subseteq K^\perp$, it follows that $K' \subseteq G \parallel K \subseteq G \parallel K^\perp$. Also, $K'$ is nonblocking by assumption. It remains to show that $K'$ is controllable in $G \parallel K^\perp$. Let $s \in \Sigma^*$ and $v \in \Sigma_u$ be such that $K' \xrightarrow{s} p$ and $G \parallel K^\perp \xrightarrow{s} p \xrightarrow{v} q$. By the definition of $\parallel$, it is clear that $G \xrightarrow{sv}$. Thus, since $K'$ is controllable with respect to $G$, it follows that $K' \xrightarrow{sv}$. Since $K'$ is deterministic, this implies $K' \xrightarrow{s} p \xrightarrow{v} q$. ■

An immediate consequence of this result is that synthesis of the least restrictive nonblocking and controllable behaviour allowed by a specification $K$ with respect to a plant $G$—both deterministic—can be achieved by computing

$$\sup\mathcal{CN}(G \parallel K^\perp)\ . \tag{10}$$

The result can be used to implement a supervisor, enabling precisely the events enabled in $\sup\mathcal{CN}(G \parallel K^\perp)$.

However, as a result of hiding and/or abstractions, $\sup\mathcal{CN}$ may be applied to nondeterministic automata. Then it is not immediately clear which events the supervisor should enable (and, indeed, in which states). To solve this problem, state labels are introduced to convey the necessary information.

## 3.5 Kripke-Structures

The intermediate results in the construction of the supervisor need to carry state labels to establish a correspondence to the global system states. To this end, labelled automata or *Kripke-structures* are used.

**Definition 13** An (extended) *Kripke-structure* is a 7-tuple

$$G \;=\; \langle Q, \Sigma, \rightarrow, Q^i, Q^m, \Lambda, B \rangle \tag{11}$$

where $\langle Q, \Sigma, \rightarrow, Q^i, Q^m \rangle$ is an automaton, $\Lambda$ is a set of *state labels*, and $B : Q \rightarrow 2^\Lambda$ is a map that associates each state with a set of labels. ■

A Kripke-structure can be considered as an automaton, simply by ignoring its labels. Conversely, an unlabelled automaton $G = \langle Q, \Sigma, \rightarrow, Q^i, Q^m \rangle$ can be extended to a Kripke-structure $G'$ by using the states themselves as their labels,

$$G' \;=\; \langle Q, \Sigma, \rightarrow, Q^i, Q^m, \Lambda, B \rangle \tag{12}$$

where $\Lambda = Q$ and $B(q) = \{q\}$. Simplification may result in states with more than one label associated to them. In the following, the letter $G$ is used to represent both an automaton and its Kripke-structure. All concepts and notations that can be applied to automata, such as transitions and controllability, are extended to Kripke-structures in the straightforward way.

Synchronous composition produces state tuples as labels and is extended to Kripke-structures using

$$\Lambda_{G_1 \parallel G_2} \;=\; \Lambda_{G_1} \times \Lambda_{G_2}\ ; \tag{13}$$

$$B_{G_1 \parallel G_2}\big((p, q)\big) \;=\; B_{G_1}(p) \times B_{G_2}(q)\ . \tag{14}$$

Here, the resulting labels should not depend on the order in which automata are composed; $\|$ should be commutative. Therefore pairs $(p, q)$ and $(q, p)$, e.g., are considered as equivalent when occurring as labels. It is also assumed that all states in all automata are unique.

Synthesis is also extended to Kripke-structures,

$$\sup\mathcal{CN}(G) \;=\; \begin{cases} \langle \hat{P}, \Sigma, \hat{P}, Q^i, Q^m \cap \hat{P}, \Lambda, B\hat{P} \rangle & \text{if } Q^i \subseteq \hat{P} \; , \\ \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle & \text{otherwise} \; ; \end{cases} \tag{15}$$

where $\hat{P}$ is the fixed point of the synthesis iteration $P^0 = Q$, $P^{i+1} = \Theta_G(P^i)$ for the automaton $G$. $B\hat{P}$ is the labelling function restricted to the state set $\hat{P}$.

Sometimes it is of interest to know the set of all reachable labels in a Kripke-structure $G$, which is defined as

$$B(G) \;=\; \{\, l \in \Lambda \mid \exists q \in Q \colon G \to q,\; l \in B(q) \,\} \; . \tag{16}$$

# 4 Compositional Synthesis

A modular supervisory control problem consists of a plant $G = G_1 \| \cdots \| G_n$ and a specification $K = K_1 \| \cdots \| K_m$, each composed of *deterministic* automata. The task is to find the supremal controllable and nonblocking sub-behaviour of

$$G \| K \;=\; G_1 \| \cdots \| G_n \| K_1 \| \cdots \| K_m \; , \tag{17}$$

or, equivalently, the largest subautomaton of $G \| K$ that is nonblocking and controllable with respect to $G$.

Proposition 1 shows that this can be represented equivalently by a set of plant automata. Therefore, in the following it is assumed without loss of generality that the synthesis problem consists of finding a nonblocking and controllable supervisor for a modular deterministic plant

$$G \;=\; G_1 \| \cdots \| G_n \; , \tag{18}$$

represented as Kripke-structures using the states themselves as their labels. This is the starting point for compositional synthesis.

## 4.1 Compositional Minimisation

The supervisor should result in the least restrictive nonblocking and controllable sub-behaviour of the system $G$ in (18). The simplest way to achieve this is a *monolithic supervisor* that can be defined by

$$S_G(s) \;=\; \{\, \sigma \in \Sigma \mid \sup\mathcal{CN}(G) \xrightarrow{s\sigma} \,\} \; . \tag{19}$$

To avoid monolithic synthesis and to be able to compute this supervisor compositionally, the system $G$ of plant automata is transformed into a simpler system

$$H \;=\; H_1 \| \cdots \| H_m \tag{20}$$

that should be related to the original system $G$ in an appropriate way. The simplified system $H$ is assumed to use the same state labels as the original system $G$, i.e., it is labelled by the states in $G$. Using these labels, the simplified system can be used to define an alternative supervisor to $S$, the *compositional supervisor*,

$$S'_H(s) \;=\; \{\, \sigma \in \Sigma \mid G \overset{s\sigma}{\to} q \text{ and } q \in B(\sup\mathcal{CN}(H))\,\} \; . \tag{21}$$

To determine whether an event $\sigma$ should be enabled after executing a string $s$, the compositional supervisor first determines whether the original system $G$ can execute string $s\sigma$. If this is the case, it checks whether the state reached by $G$ is a reachable label of $\sup\mathcal{CN}(H)$. If this also is the case, then the event $\sigma$ is enabled, otherwise it is disabled.

To be useful, the supervisor constructed in this way should produce exactly the same behaviour as a monolithic supervisor computed for the original system. Clearly, this can only be guaranteed if the simplified system $H$ stands in a certain relationship to the original system $G$.

Three operations are proposed that can be applied to a system of plant automata in such a way that the resultant supervisor yields the same behaviour.

*Synchronous Composition.* Any two plant automata can be replaced by their synchronous product.

*Hiding.* If an event $\sigma$ is used by only one automaton $G_i$, then $G_i$ can be replaced by $G_i \setminus\!\{\sigma\}$.

*Simplification.* A plant automaton can be replaced by a simplified automaton, provided that the simplified automaton is *supervision equivalent* to the original.

The simplification step clearly relies on an appropriate notion of equivalence to guarantee that the resultant supervisor remains unchanged. The following definition introduces a general equivalence that relates two automata with equivalent synthesis results in combination with any other system. Synthesis results for Kripke-structures can be considered as equivalent if they result in the same sets of state labels.

**Definition 14** Two Kripke-structures $G$ and $H$ are said to be *supervision equivalent*, denoted $G \simeq_{\sup} H$, if, for any automaton $T$,

$$B(\sup\mathcal{CN}(G \parallel T)) \;=\; B(\sup\mathcal{CN}(H \parallel T)) \; . \qquad \blacksquare$$

Given this, the three possible ways to rewrite systems of plant automata are now defined formally, using the notation of term rewrite systems (Dershowitz and Jouannaud, 1990). The set of automata on top of the horizontal line can be replaced by the one below, provided that the conditions listed are satisfied.

**Definition 15** The following rules can be used to rewrite sets of Kripke-structures $G_1, \ldots, G_n$.

$$\frac{\{G_1, \ldots, G_n\}}{\{G_1 \parallel G_2, G_3, \ldots, G_n\}}$$

$$\frac{\{G_1, \ldots, G_n\}}{\{G_1 \setminus\!\Upsilon, G_2, \ldots, G_n\}} \quad \text{if events } \Upsilon \subseteq \Sigma \text{ are unused in } G_2, \ldots, G_n;$$

$$\frac{\{G_1, \ldots, G_n\}}{\{H_1, G_2, \ldots, G_n\}} \quad \text{if } G_1 \simeq_{\sup} H_1.$$

If $\{G_1, \ldots, G_n\}$ can be rewritten into $\{H_1, \ldots, H_m\}$ using one of the above rules, this is denoted by $\{G_1, \ldots, G_n\} \succ \{H_1, \ldots, H_m\}$. The reflexive and transitive closure of the rewrite relation $\succ$ is denoted by $\succ^*$. ■

## 4.2 Main Result

In order to construct the compositional supervisor, the system of plants (18) is repeatedly rewritten and simplified using the rewrite rules given in definition 15. This process naturally terminates when all automata have been composed and all events have been hidden. As it turns out, the final result can always be reduced to a one-state or zero-state automaton.

The following result shows that this method is sound. The sequence of rewrite steps always leads to a supervisor that is equivalent to the monolithic supervisor for the original system, and therefore yields the least restrictive behaviour that can be achieved by supervision.

**Proposition 2** Let $G = G_1 \parallel \cdots \parallel G_n$ be a system of deterministic plant automata, and let $H = H_1 \parallel \cdots \parallel H_m$ be a system of Kripke-structures such that

$$\{G_1, \ldots, G_n\} \succ^* \{H_1, \ldots, H_m\} \ . \tag{22}$$

Then their synthesised supervisors yield the same behaviour, that is, $S_G = S'_H$. □

**Proof.** The claim is proved by induction on the number of rewrite steps used to transform $G$ into $H$.

*Base case.* First assume that $G = H$, that is, no rewrite steps have been used. It needs to be shown that the two supervisors obtained from $G$ are equal, i.e., that $S_G = S'_G$. This is the case because, for arbitrary $s \in \Sigma^*$,

$$\begin{aligned} S_G(s) &= \{\, \sigma \in \Sigma \mid \sup\mathcal{CN}(G) \xrightarrow{s\sigma} \} \\ &= \{\, \sigma \in \Sigma \mid G \xrightarrow{s\sigma} q, q \text{ reachable in } \sup\mathcal{CN}(G) \,\} \\ &= \{\, \sigma \in \Sigma \mid G \xrightarrow{s\sigma} q, q \in B(\sup\mathcal{CN}(G)) \,\} \\ &= S'_G(s) \ . \end{aligned}$$

*Inductive step.* Assume that $G$ is rewritten into $H$ in $k+1$ rewrite steps as follows

$$G = G^0 \succ \cdots \succ G^k \succ G^{k+1} = H \ . \tag{23}$$

By inductive assumption, the supervisor for $G^k$ behaves like the monolithic supervisor for $G$, i.e., $S_G = S'_{G^k}$. It remains to be shown that

$$S'_{G^k} = S'_H \ , \tag{24}$$

where $H$ is obtained from $G^k$ using one of the rewrite rules from definition 15.

*Case 1.* $H$ is obtained by synchronous composition from $G^k$. Let $G^k = \{G_1, G_2, \ldots, G_n\}$, and let $H = \{G_1 \parallel G_2, G_3, \ldots, G_n\}$. By definition of synchronous composition and synthesis, and using the assumption that state labels are constructed in such a way that the order of composition does not matter, it follows that

$$\begin{aligned}
B(\sup\mathcal{CN}(G^k)) &= B(\sup\mathcal{CN}(G_1 \parallel G_2 \parallel \cdots \parallel G_n)) \\
&= B(\sup\mathcal{CN}((G_1 \parallel G_2) \parallel \cdots \parallel G_n)) \\
&= B(\sup\mathcal{CN}(H)) \ .
\end{aligned}$$

*Case 2.* $H$ is obtained by hiding from $G^k$. Let $G^k = \{G_1, G_2, \ldots, G_n\}$ and $H = \{G_1 \setminus_! \Upsilon, G_2, \ldots, G_n\}$ where all events in $\Upsilon$ are unused in $G_2 \parallel \cdots \parallel G_n$. Since synthesis treats the silent events $\tau_c$ and $\tau_u$ in the same way as ordinary controllable and uncontrollable events, it follows that

$$\begin{aligned}
B(\sup\mathcal{CN}(G^k)) &= B(\sup\mathcal{CN}(G_1 \parallel \cdots \parallel G_n)) \\
&= B(\sup\mathcal{CN}((G_1 \parallel \cdots \parallel G_n) \setminus_! \Upsilon)) \\
&= B(\sup\mathcal{CN}((G_1 \setminus_! \Upsilon) \parallel G_2 \parallel \cdots \parallel G_n)) \\
&= B(\sup\mathcal{CN}(H)) \ .
\end{aligned}$$

*Case 3.* $H$ is obtained by simplification from $G^k$. Let $G^k = \{G_1, G_2, \ldots, G_n\}$ and $H = \{H_1, G_2, \ldots, G_n\}$ where $G_1 \simeq_{\text{sup}} H_1$. By letting $T = G_2 \parallel \cdots \parallel G_n$ in definition 14,

$$\begin{aligned}
B(\sup\mathcal{CN}(G^k)) &= B(\sup\mathcal{CN}(G_1 \parallel G_2 \parallel \cdots \parallel G_n)) \\
&= B(\sup\mathcal{CN}(H_1 \parallel G_2 \parallel \cdots \parallel G_n)) \\
&= B(\sup\mathcal{CN}(H)) \ .
\end{aligned}$$

Thus, the equation $B(\sup\mathcal{CN}(G^k)) = B(\sup\mathcal{CN}(H))$ holds in all three cases. This implies, for arbitrary $s \in \Sigma^*$,

$$\begin{aligned}
S_G(s) &= S'_{G^k}(s) \\
&= \{\, \sigma \in \Sigma \mid G \xrightarrow{s\sigma} q, q \in B(\sup\mathcal{CN}(G^k)) \,\} \\
&= \{\, \sigma \in \Sigma \mid G \xrightarrow{s\sigma} q, q \in B(\sup\mathcal{CN}(H)) \,\} \\
&= S'_H(s) \ . \qquad \blacksquare
\end{aligned}$$

## 5 Rewrite Operations

Simplification is the only step that reduces the size of intermediate automata, and is therefore crucial for the performance of the method. Since the state-space tends to grow exponentially with the number of components, even a small reduction, particularly at an early stage, can greatly reduce effort later in the process.

In this section a number of rules for how an automaton can be rewritten to a simpler supervision equivalent version are presented. The rules presented below are in general not sufficient to find a minimal supervision equivalent representation of an automaton. However, the authors believe that they can be used heuristically to provide significant reduction for many examples of practical relevance.

## 5.1 Removal of $\tau$-selfloops

All selfloops associated with silent events can be removed. This is possible because a selfloop never causes the automaton to change its state, and the states of other automata are not affected by silent transitions. By itself, this is not a very important reduction, yet it may be useful for fulfilling the prerequisites for other reduction rules.

## 5.2 Kripke Bisimulation

*Bisimulation* (Milner, 1989), is a strong equivalence of nondeterministic automata that considers only states with the same future behaviour as equivalent. Here, Kripke-structures are considered, and the bisimulation relation must also take the state labels into account when comparing states for equivalence, see figure 7.

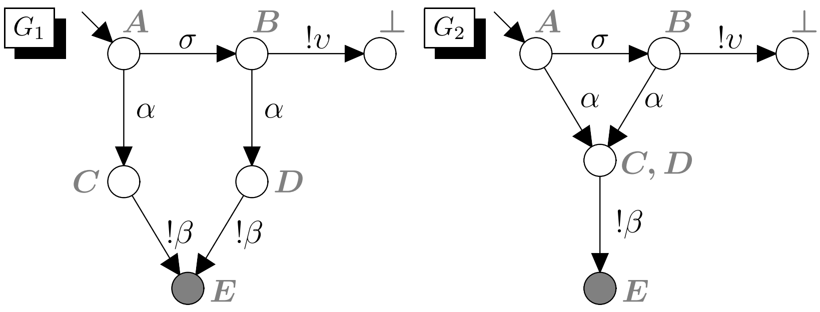


Figure 7: Even though the states labelled $C$ and $D$ in $G_1$ are bisimilar, they cannot be collapsed as in $G_2$. Consider as a test an automaton that allows everything. In $G_1$, the path to $D$ will be removed in synthesis, so $D$ is not an approved label. However, in $G_2$, the label $D$ would still be reachable.

The following definition extends the standard notion of bisimulation (Milner, 1989) to Kripke-structures, by adding the requirement that bisimilar states must have the same marking and label sets.

**Definition 16** Let $G_1 = \langle Q_1, \Sigma, \rightarrow_1, Q_1^i, Q_1^m, \Lambda_1, B_1 \rangle$ and $G_2 = \langle Q_2, \Sigma, \rightarrow_2, Q_2^i, Q_2^m, \Lambda_2, B_2 \rangle$ be two Kripke-structures. A relation $\sim \in Q_1 \times Q_2$ is said to be a *Kripke-bisimulation* for $G_1$ and $G_2$ if $p \sim q$ implies that for any $\sigma \in \Sigma_\tau$

$$\begin{aligned}
&\text{if } p \xrightarrow{\sigma}_1 p' \text{ then } \exists q' \text{ such that } q \xrightarrow{\sigma}_2 q' \text{ and } p' \sim q' \ ; \\
&\text{if } q \xrightarrow{\sigma}_2 q' \text{ then } \exists p' \text{ such that } p \xrightarrow{\sigma}_1 p' \text{ and } p' \sim q' \ ; \\
&p \in Q_1^m \text{ if and only if } q \in Q_2^m \ ; \\
&B_1(p) = B_2(q) \ .
\end{aligned}$$

Two Kripke-structures $G_1$ and $G_2$ are *Kripke-bisimilar* if there exists a Kripke-bisimulation $\sim$ such that for each initial state $q_1^i \in Q_1^i$ there exists an initial state $q_2^i \in Q_2^i$ such that $q_1^i \sim q_2^i$, and vice versa. ■

**Proposition 3** If $G_1$ and $G_2$ are Kripke-bisimilar then $G_1 \simeq_{\text{sup}} G_2$. □

**Proof (Sketch).** $G_1$ and $G_2$ clearly have the same reachable labels. In each step of synthesis, if a state is removed, so are all states that are bisimilar to that state. ■

In the case of $H_2$ in figure 4, the states $I_1 1$ and $W_1 0$ and also $I_1 2$ and $W_1 1$ are bisimilar but *not* Kripke bisimilar. However, the problem in figure 7 cannot occur in this special case unless we disable $\tau_c$ transitions in state $I_1 0$ or $I_1 1$ for which there is no reason. So, $H_2$ can actually be reduced to just four states. However, other reduction methods are needed, and no proof of this is given here.

## 5.3 Solitary Outgoing $\tau_u$

When a state only has one outgoing transition and it is $\tau_u$, a supervisor that allows the system to reach that state must also allow the system to reach the next state. Therefore, this transition can be abstracted away.

**Proposition 4** Let $G_1 = \langle Q, \Sigma, \rightarrow_1, Q^i, Q_1^m, \Lambda, B_1 \rangle$. If there exists a state $x \in Q$, $x \notin Q^m$ such that $x$ has *only* one outgoing transition and it is $x \xrightarrow{\tau_u}_1 y$ for some $y \in Q$, $y \neq x$. Then $G_1 \simeq_{\sup} G_2$ where $G_2 = \langle Q, \Sigma, \rightarrow_2, Q^i, Q_2^m, \Lambda, B_2 \rangle$ with

$$\begin{aligned}
\rightarrow_2 &= (\rightarrow_1 \setminus \{\langle x, \tau_u, y \rangle\}) \cup \{\langle x, \sigma, z \rangle \mid y \xrightarrow{\sigma}_1 z\} \ ; \\
Q_2^m &= \begin{cases} Q_1^m \cup \{x\} & \text{if } y \in Q_1^m \\ Q_1^m & \text{if } y \notin Q_1^m \ ; \end{cases} \\
B_2(q) &= \begin{cases} B_1(q) & \text{if } q \neq x \\ B_1(x) \cup B_1(y) & \text{if } q = x \ . \end{cases}
\end{aligned}$$
□

The proof can be found in appendix A. An example of the application of this rule can be found in figure 8. This rule, as well as the following, is of most use when $y$ has no incoming transitions other than the required $\tau_u$-transition. In such cases $y$ becomes unreachable and can be removed altogether.

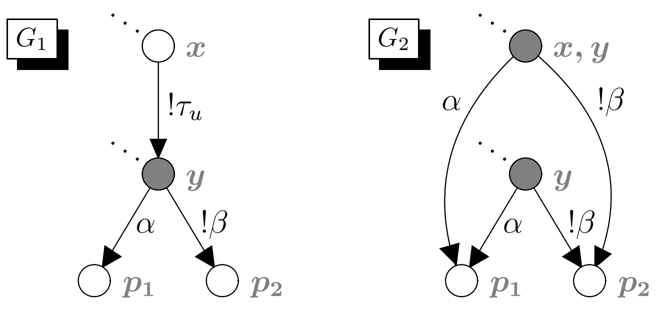


Figure 8: Example of solitary $\tau_u$-removal, in $G_1$ the $\tau_u$-transition from $x$ to $y$ can be removed. All outgoing transitions and labels from $y$ must be copied to $x$. The dots represent incoming transitions to $x$ and $y$ that are left unchanged.

## 5.4 Bypass of $\tau_u$

If a $\tau_u$-transition is followed by another $\tau_u$-transition, then the state in between can be bypassed. The second $\tau_u$-transition guarantees that the abstraction will keep any blocking situations that the original automaton had.

**Proposition 5** Let $G_1 = \langle Q, \Sigma, \rightarrow_1, Q^i, Q_1^m, \Lambda, B_1 \rangle$. If there exists a state $x \in Q$ such that there exists two transitions $x \xrightarrow{\tau_u}_1 y \xrightarrow{\tau_u}_1 z$ for some $y, z \in Q$. Then $G_1 \simeq_{\sup} G_2$ where $G_2 = \langle Q, \Sigma, \rightarrow_2, Q^i, Q_2^m, \Lambda, B_2 \rangle$ with

$$\begin{aligned}
\rightarrow_2 &= (\rightarrow_1 \setminus \{\langle x, \tau_u, y \rangle\}) \cup \{\langle x, \sigma, z \rangle \mid y \xrightarrow{\sigma}_1 z\} ; \\
Q_2^m &= \begin{cases} Q_1^m \cup \{x\} & \text{if } y \in Q_1^m \\ Q_1^m & \text{if } y \notin Q_1^m ; \end{cases} \\
B_2(q) &= \begin{cases} B_1(q) & \text{if } q \neq x \\ B_1(x) \cup B_1(y) & \text{if } q = x . \end{cases}
\end{aligned} \qquad \square$$

Note the difference between the "solitary outgoing $\tau_u$"-rule and the "bypass of $\tau_u$"-rule. In the former, $x$ must have no other outgoing transitions and no marking, in the latter, $y$ must have an outgoing $\tau_u$-transition.

The proof is similar to the proof for proposition 4 and is left out. An example of the application of this rule can be found in figure 9.

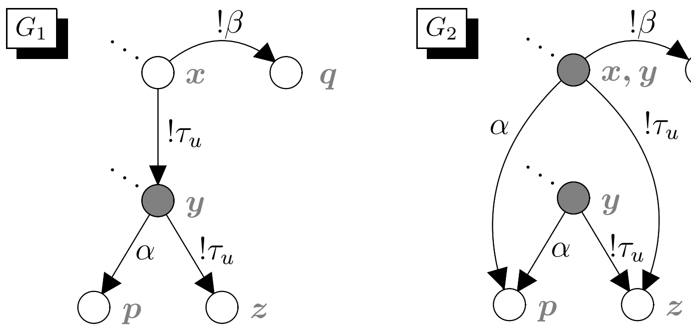


Figure 9: Example of $\tau_u$-bypass, in $G_1$ the $\tau_u$-transition from $x$ to $y$ can be bypassed. All transitions outgoing from $y$ are copied to $x$. The dots represent incoming transitions to $x$ and $y$ that are left unchanged.

## 5.5 $\tau_u$-saturation

Since $\tau_u$ can never become disabled, a supervisor must be designed to cope with all situations that can be reached by $\tau_u$-transitions. A transition that can be executed after a $\tau_u$ transition might as well be considered for execution in the current state. Adding transitions in this way is called $\tau_u$-saturation.

**Proposition 6** Let $G_1 = \langle Q, \Sigma, \rightarrow_1, Q^i, Q^m, \Lambda, B \rangle$. Whenever transitions exists such that $x \xrightarrow{\tau_u}_1 y \xrightarrow{\sigma}_1 z$ for some $x, y, z \in Q$, $\sigma \in \Sigma_\tau$. Then $G_1 \simeq_{\sup} G_2$ where $G_2 = \langle Q, \Sigma, \rightarrow_2, Q^i, Q^m, \Lambda, B \rangle$ with

$$\rightarrow_2 = \rightarrow_1 \cup \{\langle x, \sigma, z \rangle\} . \qquad \square$$

This rule can also be used for removal of transitions, of course, which may be more useful. If a transition exists out of the current state and "the same" transition exists in a state reachable by a $\tau_u$-transition, the first transition can be removed.

**Proof (Sketch).** This follows from the $\tau_u$-bypass rule and the $\tau$-selfloop rule. Consider $G_1$ and let $y = z$ in proposition 5, i.e., there is a $\tau_u$-selfloop in state $y$ (if there is none, add one). The result is that a number of transitions are added (and the added selfloop can be removed). The exact same result, however, is reached when starting from $G_2$. Thus $G_1 \simeq_{\text{sup}} G_2$. ■

**Remark 1** It follows from the argument in the proof that also *marking* and *labels* can be propagated backwards over $\tau_u$-transitions. Applying proposition 5 will show that the results are supervision equivalent.

An example of $\tau_u$-saturation can be found in figure 10.

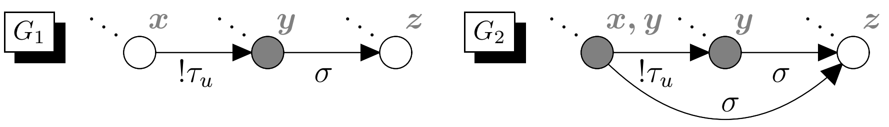


Figure 10: Example of $\tau_u$-saturation, transitions on $\sigma$ are added in $G_2$. The marking and the labels are propagated backwards. The dots represent incoming transitions that are left unchanged.

## 5.6 Collapsing $\tau_u$-loops

Let $G$ be an automaton, and let $P$ be a set of states such that for all states $p, p' \in P$ it holds that $p \xrightarrow{\tau_u^*} p'$. Then the states in $P$ can be collapsed meaning that all transitions to/from the set are redirected to/from a single state which has the union of the labels and the marking.

This follows from the $\tau_u$-saturation rule and the bisimulation rule.

## 5.7 Solitary Outgoing $\tau_c$

A similar rule as the rule for solitary $\tau_u$-transitions can be derived for $\tau_c$-transitions. However, the target state of the $\tau_c$ transition must not have any outgoing uncontrollable transitions. The reason for this is that every state with an outgoing uncontrollable transition may become a "bad" state in some environment, therefore it needs to be distinguished from other states until the status of its outgoing uncontrollable transitions is known.

**Proposition 7** Let $G_1 = \langle Q, \Sigma, \rightarrow_1, Q^i, Q_1^m, \Lambda, B_1 \rangle$. Assume there exists a state $x \in Q$, $x \notin Q_1^m$ such that $x$ has *only* one outgoing transition and it is $x \xrightarrow{\tau_c}_1 y$ for some $y \in Q$, $y \neq x$, and $y \xrightarrow{\sigma}$ implies that $\sigma \in \Sigma_{\tau,c}$. Then $G_1 \simeq_{\text{sup}} G_2$ where $G_2 = \langle Q, \Sigma, \rightarrow_2, Q^i, Q_2^m, \Lambda, B_2 \rangle$ with

$$\begin{aligned}
\rightarrow_2 &= (\rightarrow_1 \setminus \{\langle x, \tau_c, y \rangle\}) \cup \{\langle x, \sigma, z \rangle \mid y \xrightarrow{\sigma}_1 z\} \; ; \\
Q_2^m &= \begin{cases} Q_1^m \cup \{x\} & \text{if } y \in Q_1^m \\ Q_1^m & \text{if } y \notin Q_1^m \; ; \end{cases} \\
B_2(q) &= \begin{cases} B_1(q) & \text{if } q \neq x \\ B_1(x) \cup B_1(y) & \text{if } q = x \; . \end{cases}
\end{aligned}$$ □

Note that state $y$ must not have any outgoing uncontrollable transitions.

The proof is similar to the proof for proposition 4 and is left out.

## 5.8 $\tau_c$-saturation

A weaker kind of saturation than the one for $\tau_u$-transitions can be derived for $\tau_c$-transitions from the solitary $\tau_c$ rule.

It is not possible to derive something like a $\tau_c$-bypass rule. Controllable transitions have to be retained until it is absolutely certain that the supervisor does not need to disable them, see figure 11.

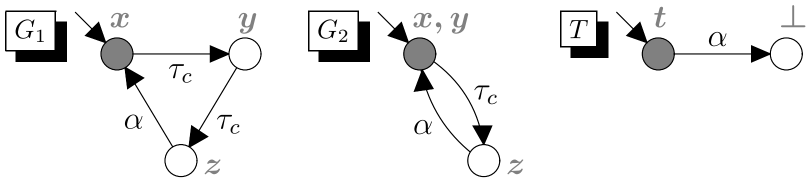


Figure 11: Illustration of the problems involved with developing a "$\tau_c$-bypass" rule. Consider bypassing the state labelled $y$ in $G_1$. The result of this would be $G_2$ since the state with $y$ would become unreachable. However, the automaton $T$ would distinguish $G_1$ from $G_2$. Synthesis on $G_2 \parallel T$ would allow the label $(y, t)$ which is really a blocking state. Note that everything is controllable.

## 5.9 Halfway Synthesis

A simple but powerful simplification method is called *halfway synthesis*. The idea is to perform synthesis on a subsystem but, to guarantee that the end result is least restrictive, the synthesis must take into account that all uncontrollable events except $\tau_u$ may actually become disabled by other plants in the system. Thus, transitions associated with such events are not sure to cause uncontrollability and must be retained to guarantee least restrictiveness.

The halfway synthesis result can be calculated using a fixed point algorithm similar to standard synthesis. Let $G = \langle Q, \Sigma, \rightarrow, Q^i, Q^m, \Lambda, B\rangle$, and let $\Theta_G^{\tau_u}(P) = \Theta_G^{\mathrm{cont},\tau_u}(P) \cap \Theta_G^{\mathrm{nonb}}(P)$ with

$$\Theta_G^{\mathrm{cont},\tau_u}(P) \;=\; \{\, q \in P \mid q \xrightarrow{\tau_u} p \text{ implies } p \in P \,\} \;. \tag{25}$$

Let $\hat{P}$ be the fixed point of the iteration $P^0 = Q$, $P^{i+1} = \Theta_G^{\tau_u}(P^i)$. The halfway synthesis result is the portion of $G$ that remains when all controllable transitions that leave $\hat{P}$ and all transitions that leave states outside $\hat{P}$ have been removed,

$$\sup\mathcal{CN}_h(G) \;=\; \langle Q, \Sigma, \rightarrow_h, Q^i, Q^m \cap \hat{P}, \Lambda, B\rangle \tag{26}$$

where

$$\rightarrow_h \;=\; \{\, \langle p, \sigma, q\rangle \in \rightarrow \mid p \in \hat{P} \text{ and, if } q \notin \hat{P} \text{ then } \sigma \in \Sigma_{\tau,u} \,\} \;. \tag{27}$$

Technically, this construction only deletes transitions from the automaton $G$, while retaining all its states. This is necessary, because some states not in $\hat{P}$ remain reachable by uncontrollable events; these states are made blocking in the process so they are treated as "bad" states by subsequent synthesis steps. Of course, all states that are unreachable after the completion of halfway synthesis can be removed.

**Proposition 8** Let $G$ be a Kripke-structure and let $G_h = \sup\mathcal{CN}_h(G)$. Then

$$G \simeq_{\sup} G_h \ . \qquad \square$$

The proof can be found in appendix B. An example of halfway synthesis can be found in figure 12.

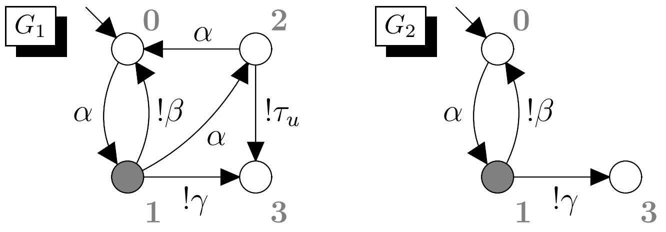


Figure 12: Example of halfway synthesis, state 2 is bad since it can reach the blocking state 3 on $\tau_u$. That is, $\hat{P} = \{0, 1\}$ and all controllable transitions leading to 2 and 3 can be removed. 2 becomes unreachable and is removed.

## 6 Experimental results

As of yet, there is just a rudimentary implementation of the compositional supervision equivalence algorithm, in the DES tool Supremica, (Supremica, 2007). The only rule that has been implemented is halfway synthesis. Since halfway synthesis never collapses states, only removes them, maximally permissiveness implies that the end result in this case is actually nothing other than an automaton with the same state-space as the traditional monolithic supervisor. However, this end result can sometimes be reached without exploring more than just a small fraction of the entire global state-space. In effect, the halfway synthesis enables the synthesis process to take a "shortcut" straight to the monolithic end result as it can sometimes cut off huge chunks of the state-space by identifying and removing transitions to blocking and uncontrollable states at a very early stage. That is, the blocking and uncontrollable parts of the global state-space may never have to be examined explicitly if the problems can be identified before the state-space explosion occurs.

Table 1 shows some statistics for three examples where this effect is particularly prominent; that is, where the system being analysed has many blocking and

Table 1: Test case examples for compositional controllability and nonblocking synthesis in Supremica. "Aut." is the number of automata. The "States", "Trans.", and "Time" columns present the number of states and transitions explored and the time for monolithic and compositional synthesis, respectively. The "Result" columns show the size of the monolithic supervisor.

| **Example** | | **Monolithic** | | | **Compositional** | | | **Result** | |
|---|---|---|---|---|---|---|---|---|---|
| **Name** | **Aut.** | **States** | **Trans.** | **Time** | **States** | **Trans.** | **Time** | **States** | **Trans.** |
| `IPC` | 12 | 11469 | 36660 | 1 s | 11916 | 42708 | 1 s | 9216 | 33592 |
| `3transfer` | 15 | 127764 | 426465 | 11 s | 22916 | 125242 | 4 s | 15352 | 81422 |
| `FMS` | 16 | 309024 | 683280 | 25 s | 57003 | 259580 | 11 s | 45505 | 200124 |

uncontrollable states. The models are "manufacturing system" models found in the literature.

**Intertwined Product Cycles—`IPC`.** Two types of products are produced in a system with two machines such that the products must move back and forth between the two machines in opposite directions (Lin and Wonham, 1990). Only the plant models and the buffer specifications are considered here.

**Transfer Line—`3transfer`.** Three serially connected transfer line cells. The individual cells are the same as in figure 1 except that the first buffer has a capacity of three.

**Flexible Manufacturing System—`FMS`.** A manufacturing system consisting of a several machining devices, a robot, and a set of buffers and conveyors. The system produces two kinds of products. The example is adapted from (Queiroz, Cury and Wonham, 2005).

The monolithic synthesis algorithm used in the table also avoids exploring the full state-space. It does not explore the successor states of confirmed "bad" states. Nevertheless the compositional algorithm is significantly faster and constructs less states in all cases except for the small `IPC` example. Much better results can be expected if more of the reduction rules presented in this paper are implemented.

## 7 Conclusions

A general method for compositional synthesis of controllable and nonblocking supervisors for discrete-event systems has been proposed. The monolithic representation of the state-space is avoided by the use of simplified automata at the intermediate stages of the algorithm, in this way the modularity is exploited. The resulting supervisor is represented efficiently using a symbolic mapping of state labels. A set of test cases are presented where a rudimentary implementation of the synthesis procedure is shown to be able to avoid building the full synchronous composition.

The proposed framework can be extended and enhanced in different ways. In the future, the authors would like to study and evaluate additional algorithms for the minimisation of automata in a way that preserves supervision equivalence. The order in which automata are to be composed needs to be studied in more detail, since the performance of the method depends on it. Another problem with the present approach is the need to consider state labels throughout the synthesis process, which can make some desirable reductions impossible, and which may be avoided when using a different approach. Furthermore, it is interesting to consider coarser equivalences than supervision equivalence that take some aspects of the rest of the system considered into account.

## References

Brandin, B. A., Malik, R. and Malik, P.: 2004, Incremental verification and synthesis of discrete-event systems guided by counter examples, *Transactions on Control System Technology* **12**(3), 387–401.

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

De Nicola, R. and Hennessy, M. C. B.: 1984, Testing equivalences for processes, *Theoretical Computer Science* **34**(1–2), 83–133.

Dershowitz, N. and Jouannaud, J.-P.: 1990, Rewrite systems, *in* J. van Leeuwen (ed.), *Handbook of Theoretical Computer Science*, Vol. B, Elsevier, pp. 243–320.

Fabian, M.: 1995, *On Object Oriented Nondeterministic Supervisory Control*, PhD thesis, Control Engineering Laboratory, Chalmers University of Technology, Göteborg, Sweden.

Feng, L. and Wonham, W. M.: 2006, Computationally efficient supervisor design: Abstraction and modularity, *Proceedings of the 8th International Workshop on Discrete Event Systems, WODES '06*, Ann Arbor, MI, USA, pp. 3–8.

Flordal, H. and Malik, R.: 2006, Supervision equivalence, *Proceedings of the 8th International Workshop on Discrete Event Systems, WODES '06*, Ann Arbor, MI, USA, pp. 155–160.

Hill, R. C. and Tilbury, D. M.: 2006, Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction, *Proceedings of the 8th International Workshop on Discrete Event Systems, WODES '06*, Ann Arbor, MI, USA, pp. 399–406.

Hoare, C. A. R.: 1985, *Communicating sequential processes*, Series in Computer Science, Prentice-Hall.

Lin, F. and Wonham, W. M.: 1990, Decentralized control and coordination of discrete-event systems with partial observation, *IEEE Transactions on Automatic Control* **35**(12), 1330–1337.

Malik, R., Streader, D. and Reeves, S.: 2006, Conflicts and fair testing, *International Journal of Foundations of Computer Science* **17**(4), 797–813.

Milner, R.: 1989, *Communication and concurrency*, Series in Computer Science, Prentice-Hall.

Queiroz, M. H. d. and Cury, J. E. R.: 2000, Modular supervisory control of large scale discrete event systems, *in* R. Boel and G. Stremersch (eds), *Discrete Event Systems, Analysis and Control*, Kluwer, pp. 103–110.

Queiroz, M. H. d., Cury, J. E. R. and Wonham, W. M.: 2005, Multitasking supervisory control of discrete-event systems, *Discrete Event Dynamic Systems* **15**(4), 375–395.

Ramadge, P. J. and Wonham, W. M.: 1989, The control of discrete event systems, *Proceedings of the IEEE* **77**(1), 81–98.

Song, R. and Leduc, R. J.: 2006, Symbolic synthesis and verification of hierarchical interface-based supervisory control, *Proceedings of the 8th International Workshop on Discrete Event Systems, WODES '06*, Ann Arbor, MI, USA, pp. 419–426.

Su, R. and Wonham, W. M.: 2004, Supervisor reduction for discrete-event systems, *Discrete Event Dynamic Systems* **14**(1), 31–53.

Supremica: 2007, `www.supremica.org`. The official website for the Supremica project.

Wong, K. C. and Wonham, W. M.: 1998, Modular control and coordination of discrete-event systems, *Discrete Event Dynamic Systems* **8**(3), 247–297.

Wonham, W. M.: 2006, Supervisory control of discrete event systems, *Technical report*, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada.

Åkesson, K., Flordal, H. and Fabian, M.: 2002, Exploiting modularity for synthesis and verification of supervisors, *Proceedings of the 15th IFAC World Congress*, Barcelona, Spain.

## A Proof of Proposition 4

To simplify the proof, the following lemma is used.

**Lemma 1** Assume Kripke-structures $G_1$ and $G_2$ with states $x$ and $y$ as in proposition 4. Furthermore, let $T$ be an arbitrary automaton with a state $t \in Q_T$, and let $\hat{P}_2$ be the fixed point of the synthesis calculation for $G_2 \parallel T$. Then $(x,t) \in \hat{P}_2$ implies $(y,t) \in \hat{P}_2$. □

**Proof.** Let $(x,t) \in \hat{P}_2$ and assume that $(y,t) \notin \hat{P}_2$. Note that this only makes sense if $x \neq y$. Then $(y,t)$ must have been removed during synthesis, i.e., $\exists k \in \mathbb{N}$ such that $(y,t) \in P_2^k$ but $(y,t) \notin P_2^{k+1} = \Theta_{G_2\parallel T}(P_2^k) = \Theta^{\text{cont}}_{G_2\parallel T}(P_2^k) \cap \Theta^{\text{nonb}}_{G_2\parallel T}(P_2^k)$. That is, either ($\alpha$) it holds that $(y,t) \notin \Theta^{\text{cont}}_{G_2\parallel T}(P_2^k)$ or ($\beta$) it holds that $(y,t) \notin \Theta^{\text{nonb}}_{G_2\parallel T}(P_2^k)$.

($\alpha$) This implies that $\exists \upsilon \in \Sigma_{\tau,u}$ such that $(y,t) \xrightarrow{\upsilon}_{G_2\parallel T} q \notin P_2^k$. Then, by construction of $G_2$ and since the test clearly can execute $\upsilon$, it holds that $(x,t) \xrightarrow{\upsilon}_{G_2\parallel T} q$. Thus $(x,t) \notin \Theta^{\text{cont}}_{G_2\parallel T}(P_2^k) \supseteq \hat{P}_2$, in contradiction to the assumption $(x,t) \in \hat{P}_2$.

($\beta$) This implies that $\forall q \in Q \times Q_T$ it holds that $(y,t)G_2 \parallel TP_2^k q \Rightarrow q \notin Q_2^m \times Q_T^m$. Now, consider a trace $s$ such that $(x,t)[s]G_2 \parallel TP_2^k p$. If $s = \varepsilon$, then $p = (x,t)$ and, since by construction $(x,t)$ is marked only if $(y,t)$ is marked, $p \notin Q_2^m \times Q_T^m$. Otherwise, by construction of $G_2$ and since it is known that $(y,t) \in P_2^k$ it holds that $(y,t)[s]G_2 \parallel TP_2^k p$. Thus $p \notin Q_2^m \times Q_T^m$ and since this holds for any trace $s$ it is clear that $(x,t) \notin \Theta^{\text{nonb}}_{G_2\parallel T}(P_2^k) \supseteq \hat{P}_2$, which leads to a contradiction. ■

Now, the proof of proposition 4 is quite straightforward, but lengthy.

**Proof (of proposition 4).** It needs to be shown that, for any automaton $T$, $B(\sup\mathcal{CN}(G_1 \parallel T)) = B(\sup\mathcal{CN}(G_2 \parallel T))$. To show this it is enough to show that (i) the fixed points of the synthesis calculations of $G_1 \parallel T$ and $G_2 \parallel T$, $\hat{P}_1$ and $\hat{P}_2$, are the same, and (ii) that the set of reachable labels are identical.

(i) This can be proven by induction. Let us show (A) that $\hat{P}_2 \subseteq P_1^j$ and (B) that $\hat{P}_1 \subseteq P_2^j$ for all $j \geq 0$.

(A) *Base case, $j = 0$.* By definition $\hat{P}_2 \subseteq Q \times Q_T = P_1^0$.
*Inductive step.* Assuming that the property holds for $j$ it needs to be shown that it also holds for $j+1$. Let $p \in \hat{P}_2$. By the inductive assumption it holds that $p \in P_1^j$. Assume that $p \notin P_1^{j+1}$. That is, $p \notin \Theta_{G_1\|T}(P_1^j) = \Theta^{\text{cont}}_{G_1\|T}(P_1^j) \cap \Theta^{\text{nonb}}_{G_1\|T}(P_1^j)$. This implies that either ($\alpha$) it holds that $p \notin \Theta^{\text{cont}}_{G_1\|T}(P_1^j)$ or ($\beta$) it holds that $p \notin \Theta^{\text{nonb}}_{G_1\|T}(P_1^j)$.

($\alpha$) Then $\exists \upsilon \in \Sigma_{\tau,u}$ such that $p \xrightarrow{\upsilon}_{G_1\|T} q \notin P_1^j$. If this is not a transition such that $(x,t) \xrightarrow{\tau_u}_{G_1\|T} (y,t)$ for some $t \in Q_T$ then the same transition exists in $G_2 \| T$. So $p \xrightarrow{\upsilon}_{G_2\|T} q \notin P_1^j \supseteq \hat{P}_2$. But then $p \notin \Theta^{\text{cont}}_{G_2\|T}(\hat{P}_2) = \hat{P}_2$ which is a contradiction. Now, if $p \xrightarrow{\upsilon}_{G_1\|T} q$ *is* a transition $(x,t) \xrightarrow{\tau_u}_{G_1\|T} (y,t)$ then $(y,t) \notin P_1^j \supseteq \hat{P}_2$. By lemma 1 this implies that $p = (x,t) \notin \hat{P}_2$ which is a contradiction.

($\beta$) Then $pG_1 \| TP_1^j q$ implies $q \notin Q_1^m \times Q_T^m$ for all states $q \in Q \times Q_T$. Now, consider a trace $s$ such that $p[s]G_2 \| T\hat{P}_2 r$. Then a modified trace $s'$ can be found that takes $G_1 \| T$ to the same state. The modification concerns passing through $(y,t)$ for some $t \in Q_T$ if the trace in $G_2 \| T$ uses any of the introduced transitions. The trace moreover uses the exact same states as the trace in $G_2 \| T$ except possibly for $(y,t)$ in the mentioned case. However, every time the trace needs to be redirected over $(y,t)$ that means that the trace has already passed through $(x,t)$. Thus by lemma 1, the redirection is possible without leaving $\hat{P}_2$. So, $p[s']G_1 \| T\hat{P}_2 r$ and, since $\hat{P}_2 \subseteq P_1^j$, also $p[s']G_1 \| TP_1^j r$. This implies that $r \notin Q_1^m \times Q_T^m$ but, since a state can be marked in $G_2$ without being marked in $G_1$ it is not immediately clear that $r \notin Q_2^m \times Q_T^m$.
It may be the case that $(x,t) \in Q_2^m \times Q_T^m$ although $(x,t) \notin Q_1^m \times Q_T^m$. So, consider the case $r = (x,t) \in Q_2^m \times Q_T^m$. Since $r$ can be reached without leaving $\hat{P}_2$ it holds that $r = (x,t) \in \hat{P}_2$, and by lemma 1 this implies that $(y,t) \in \hat{P}_2$. Thus $p[s'\tau_u]G_1 \| T\hat{P}_2(y,t)$ and $p[s'\tau_u]G_1 \| TP_1^j(y,t)$ by inductive assumption. This implies $(y,t) \notin Q_1^m \times Q_T^m$ and, since $t \in Q_T^m$, also $y \notin Q_1^m$. Then by construction $Q_1^m = Q_2^m$, i.e., $r \notin Q_2^m \times Q_T^m$.
Since the trace $s$ was chosen arbitrarily this means that $p \notin \Theta^{\text{nonb}}_{G_2\|T}(\hat{P}_2) = \hat{P}_2$. This contradicts the initial assumption.

(B) *Base case, $j = 0$.* By definition $\hat{P}_1 \subseteq Q \times Q_T = P_2^0$.
*Inductive step.* Assuming that the property holds for $j$ it needs to be shown that it also holds for $j+1$. Let $p \in \hat{P}_1$. By the inductive assumption it holds that $p \in P_2^j$. Assume that $p \notin P_2^{j+1}$. That is, $p \notin \Theta_{G_2\|T}(P_2^j) = \Theta^{\text{cont}}_{G_2\|T}(P_2^j) \cap \Theta^{\text{nonb}}_{G_2\|T}(P_2^j)$. This implies that either ($\alpha$) it holds that $p \notin \Theta^{\text{cont}}_{G_2\|T}(P_2^j)$ or ($\beta$) it holds that $p \notin \Theta^{\text{nonb}}_{G_2\|T}(P_2^j)$.

($\alpha$) Then $\exists \upsilon \in \Sigma_{\tau,u}$ such that $p \xrightarrow{\upsilon}_{G_2\|T} q \notin P_2^j$. If this is none of the introduced transitions, the same transition can be found in $G_1 \parallel T$ which leads to a contradiction (like (i.A.$\alpha$) above). Otherwise, this must be a transition such that for some $z \in Q$ and some $t, t' \in Q_T$ it holds that $p = (x,t) \xrightarrow{\upsilon}_{G_2\|T} (z,t') = q$. By construction it then holds that $(y,t) \xrightarrow{\upsilon}_{G_1\|T} q \notin P_2^j \supseteq \hat{P}_1$. Thus, $(y,t) \notin \Theta^{\text{cont}}_{G_1\|T}(\hat{P}_1) = \hat{P}_1$, and since $(x,t) \xrightarrow{\tau_u}_{G_1\|T} (y,t) \notin \hat{P}_1$, it also holds that $p = (x,t) \notin \Theta^{\text{cont}}_{G_1\|T}(\hat{P}_1) = \hat{P}_1$, which is a contradiction.

($\beta$) Then $pG_2 \parallel TP_2^j q$ implies $q \notin Q_2^m \times Q_T^m$ for all states $q \in Q \times Q_T$. Now, consider a trace $s$ such that $p[s]G_1 \parallel T\hat{P}_1 r$. If the last transition in this trace is not $(x,t) \xrightarrow{\tau_u}_{G_1\|T} (y,t)$ for some $t \in Q_T$, then there exists a slightly modified trace $s'$ in $G_2 \parallel T$ such that $p[s']G_2 \parallel T\hat{P}_1(y,t) = r$. Since, by the inductive assumption, $\hat{P}_1 \subseteq P_2^j$, it holds that $p[s']G_2 \parallel TP_2^j r$, which implies $r \notin Q_2^m \times Q_T^m$. Otherwise, for some $t \in Q_T$, it holds that $p[s']G_1 \parallel T\hat{P}_1(x,t)[\tau_u]G_1 \parallel T\hat{P}_1(y,t) = r$. For $(x,t)$ this is the same situation as previously, so $(x,t) \notin Q_2^m \times Q_T^m$. But, by construction, $(x,t) \in Q_2^m \times Q_T^m$ is marked if and only if $(y,t) \in Q_2^m \times Q_T^m$. Thus $(y,t) = r \notin Q_2^m \times Q_T^m \supseteq Q_1^m \times Q_T^m$. Since $s$ was chosen arbitrarily, it follows in both cases that $p \notin \Theta^{\text{nonb}}_{G_1\|T}(\hat{P}_1) = \hat{P}_1$, which is a contradiction.

(ii) It needs to be shown that $B(\sup\mathcal{CN}(G_1 \parallel T)) = B(\sup\mathcal{CN}(G_2 \parallel T))$.

$\subseteq$ Let $l \in B(\sup\mathcal{CN}(G_1 \| T))$. This means that $\exists q \in Q \times Q_T$ and $s \in \Sigma_\tau^*$ such that $G_1 \parallel T[s]G_1 \parallel T\hat{P}_1 q$ and $l \in B_{G_1\|T}(q)$. If this path does not end with the transition $(x,t) \xrightarrow{\tau_u}_{G_1\|T} (y,t)$ for some $t \in Q_T$, then it is possible to find a modified trace $s'$ such that $G_2 \parallel T[s']G_2 \parallel T\hat{P}_1 q$. Since $\hat{P}_1 = \hat{P}_2$, this implies $G_2 \parallel T[s']G_2 \parallel T\hat{P}_2 q$. By construction it holds that $B_{G_1\|T}(q) \subseteq B_{G_2\|T}(q)$ for any $q$ and so $l \in B_{G_2\|T}(q)$ and $l \in B(\sup\mathcal{CN}(G_2\|T))$. Otherwise, $G_1 \parallel T[s']G_1 \parallel T\hat{P}_1(x,t)[\tau_u]G_1 \parallel T\hat{P}_1(y,t) = q$. For $(x,t)$ it is possible to find a modified trace $s''$ such that $G_2 \parallel T[s'']G_2 \parallel T\hat{P}_1(x,t)$ and $G_2 \parallel T[s'']G_2 \parallel T\hat{P}_2(x,t)$. Now, by construction $l \in B_{G_1\|T}(q) = B_{G_1\|T}((y,t)) \subseteq B_{G_2\|T}((x,t))$, which implies $l \in B(\sup\mathcal{CN}(G_2 \parallel T))$.

$\supseteq$ Let $l \in B(\sup\mathcal{CN}(G_2 \parallel T))$. Then there exist $q \in Q \times Q_T$ and $s \in \Sigma_\tau^*$ such that $G_2 \parallel T[s]G_2 \parallel T\hat{P}_2 q$ and $l \in B_{G_2\|T}(q)$. By construction, there is a modified trace $s'$ such that $G_1 \parallel T[s']G_1 \parallel T\hat{P}_2 q$ and $G_1 \parallel T[s']G_1 \parallel T\hat{P}_1 q$. If $l \in B_{G_1\|T}(q)$ then it is clear that $l \in B(\sup\mathcal{CN}(G_1 \parallel T))$, otherwise $q = (x,t)$ for some $t \in Q_T$. Since $(x,t) = q \in \hat{P}_2$ and by lemma 1 it is then clear that also $(y,t) \in \hat{P}_2$ and thus $q = (x,t)[\tau_u]G_1 \parallel T\hat{P}_2(y,t)$. By construction $l \in B_{G_1\|T}((y,t))$, and since $\hat{P}_1 = \hat{P}_2$, it is clear that $G_1 \parallel T[s'\tau_u]G_1 \parallel T\hat{P}_1(y,t)$. Thus $l \in B(\sup\mathcal{CN}(G_1 \parallel T))$. ■

## B Proof of Proposition 8

To simplify the proof, the following lemma is used.

**Lemma 2** Consider two automata $G$ and $T$, let $\hat{P}$ be the fixed point of $\Theta_{G\|T}(\cdot)$ applied to $Q \times Q_T$, and let $\hat{H}$ be the fixed point of $\Theta_G^{\tau_u}(\cdot)$ applied to $Q$. Then $\hat{P} \subseteq \hat{H} \times Q_T$. □

Note that the fixed points $\hat{P}$ and $\hat{H}$ are found for different state sets.

**Proof.** This is proved using induction on the fixed point iterations. It is enough to show that $\forall j \in \mathbb{N}$, $P^j \subseteq H^j \times Q_T$.

*Base case.* $P^0 = Q \times Q_T = H^0 \times Q_T$.

*Inductive step.* Assuming that $P^j \subseteq H^j \times Q_T$, it needs to be shown that $P^{j+1} = \Theta_{G\|T}(P^j) \subseteq \Theta_G^{\tau_u}(H^j) \times Q_T = H^{j+1} \times Q_T$. Let $(x,t) \in P^{j+1} = \Theta_{G\|T}(P^j) = \Theta_{G\|T}^{\text{cont}}(P^j) \cap \Theta_{G\|T}^{\text{nonb}}(P^j)$, then it must be shown that $x \in H^{j+1} = \Theta_G^{\tau_u}(P^j) = \Theta_G^{\text{cont},\tau_u}(H^j) \cap \Theta_G^{\text{nonb}}(H^j)$. For this to be true it must hold ($\alpha$) that $x \in \Theta_G^{\text{cont},\tau_u}(H^j)$ and ($\beta$) that $x \in \Theta_G^{\text{nonb}}(H^j)$.

($\alpha$) Let $x \xrightarrow{\tau_u}_G y$. Then, since $\tau_u$ is silent, also $(x,t) \xrightarrow{\tau_u}_{G\|T} (y,t)$. Since $(x,t) \in P^{j+1}$ this implies that $(y,t) \in P^j \subseteq H^j \times Q_T$. So, $y \in H^j$ and since this holds for all states $y$ reachable by $\tau_u$-transitions, it is clear that $x \in \Theta_G^{\text{cont},\tau_u}(H^j)$.

($\beta$) Since $(x,t) \in \Theta_{G\|T}^{\text{nonb}}(P^j)$, for some $s \in \Sigma_\tau^*$ it holds that $(x,t)[s]G \parallel TP^jQ^m \times Q_T^m$. Since, by the inductive assumption, $P^j \subseteq H^j \times Q_T$, it is clear that $G$ only uses states in $H^j$ in this path. Then it must be possible to find a trace $s'$, derived from $s$ by possibly removing some silent events corresponding to transitions in $T$, such that $x[s']GH^jQ^m$. Thus $x \in \Theta_G^{\text{nonb}}(H^j)$. ■

**Proof (of proposition 8).** Let $\hat{H}$ be the fixed point of $\Theta_G^{\tau_u}(\cdot)$ applied to $Q$, recall that $G_h$ is derived from $G$ based on $\hat{H}$ in halfway synthesis. It needs to be shown that, for any automaton $T$, $B(\sup\mathcal{CN}(G \parallel T)) = B(\sup\mathcal{CN}(G_h \parallel T))$. It is assumed that the synthesis iterations for $G \parallel T$ and $G_h \parallel T$, resulting in fixed points $\hat{P}$ and $\hat{P}_h$, start from the complete state set $Q \times Q_T$, which may contain unreachable states. To prove the claim, it is enough to show that (i) $\hat{P} = \hat{P}_h$, and that (ii) the set of reachable labels are identical.

(i) This can be proven by induction. Let us show (A) that $\hat{P}_h \subseteq P^j$ and (B) that $\hat{P} \subseteq P_h^j$ for all $j \geq 0$.

(A) *Base case,* $j = 0$. By definition $\hat{P}_h \subseteq Q \times Q_T = P^0$.

*Inductive step.* Assuming that the property holds for $j$ it needs to be shown that it also holds for $j+1$. Let $p \in \hat{P}_h$. By the inductive assumption it holds that $p \in P^j$. Assume that $p \notin P^{j+1}$. That is, $p \notin \Theta_{G\|T}(P^j) = \Theta_{G\|T}^{\text{cont}}(P^j) \cap \Theta_{G\|T}^{\text{nonb}}(P^j)$. This implies that either ($\alpha$) it holds that $p \notin \Theta_{G\|T}^{\text{cont}}(P^j)$ or ($\beta$) it holds that $p \notin \Theta_{G\|T}^{\text{nonb}}(P^j)$.

($\alpha$) Then $p \xrightarrow{\upsilon}_{G\|T} q \notin P^j$ for some $\upsilon \in \Sigma_{\tau,u}$. Let $p = (p_h, t)$ and $q = (q_h, t')$. Then, from $(p_h, t) = p \in \hat{P}_h$ a marked state can be reached in $G_h \parallel T$. Thus, $p_h \rightarrow_{G_h} Q_{G_h}^m$. It follows that $p_h \in \hat{H}$, because states not in $\hat{H}$ are blocking after halfway-synthesis.

Therefore, the uncontrollable transition $p_h \xrightarrow{v}_G q_h$ remains in $G_h$ by equation (27) in the definition of halfway-synthesis. Thus, $p \xrightarrow{v}_{G_h \| T} q \notin P^j \supseteq \hat{P}_h$ by inductive assumption. But then $p \notin \Theta^{\text{cont}}_{G_h \| T}(\hat{P}_h) = \hat{P}_h$, which is a contradiction.

($\beta$) Then $pG \parallel TP^j q$ implies $q \notin Q^m \times Q^m_T$ for all states $q \in Q \times Q_T$. Now, consider a trace $s$ such that $p[s]G_h \parallel T\hat{P}_h r$. Since, by construction, $G_h$ has a subset of the transitions in $G$, the same trace can be found in $G \parallel T$, using the same states, $p[s]G \parallel T\hat{P}_h r$ and, since $\hat{P}_h \subseteq P^j$ by inductive assumption, also $p[s]G \parallel TP^j r$. This implies that $r \notin Q^m \times Q^m_T$. Since the trace $s$ was chosen arbitrarily this means that $p \notin \Theta^{\text{nonb}}_{G_h \| T}(\hat{P}_h) = \hat{P}_h$ which is a contradiction.

(B) *Base case, $j = 0$.* By definition $\hat{P} \subseteq Q \times Q_T = P^0_h$.
*Inductive step.* Assuming that the property holds for $j$ it needs to be shown that it also holds for $j+1$. Let $p \in \hat{P}$. By the inductive assumption it holds that $p \in P^j_h$. Assume that $p \notin P^{j+1}_h$. That is, $p \notin \Theta_{G_h \| T}(P^j_h) = \Theta^{\text{cont}}_{G_h \| T}(P^j_h) \cap \Theta^{\text{nonb}}_{G_h \| T}(P^j_h)$. This implies that either ($\alpha$) it holds that $p \notin \Theta^{\text{cont}}_{G_h \| T}(P^j_h)$ or ($\beta$) it holds that $p \notin \Theta^{\text{nonb}}_{G_h \| T}(P^j_h)$.

($\alpha$) Then $\exists v \in \Sigma_{\tau,u}$ such that $p \xrightarrow{v}_{G_h \| T} q \notin P^j_h$. Every transition in $G_h$ is also in $G$ so $p \xrightarrow{v}_{G \| T} q \notin P^j_h \supseteq \hat{P}$. But then $p \notin \Theta^{\text{cont}}_{G \| T}(\hat{P}) = \hat{P}$ which is a contradiction.

($\beta$) Then $pG_h \parallel TP^j_h q$ implies $q \notin Q^m_h \times Q^m_T$ for all states $q \in Q \times Q_T$. Consider a trace $s$ such that $p[s]G \parallel T\hat{P}r = (x,t)$. Note that this implies $r \in \hat{P}$. By lemma 2 it is clear that $\hat{P} \subseteq \hat{H} \times Q_T$ and so $p[s]G \parallel T\hat{H} \times Q_T r$. This implies that the path from $p$ to $r$ only uses transitions that are also in $G_h$. So, $p[s]G_h \parallel T\hat{P}r$ and since by the inductive assumption $\hat{P} \subseteq P^j_h$ it also holds that $p[s]G_h \parallel TP^j_h r$. This implies $(x,t) = r \notin Q^m_h \times Q^m_T$. Since $(x,t) = r \in \hat{P} \subseteq \hat{H} \times Q_T$ it is clear that $x \in \hat{H}$. Hence, $x \notin Q^m_h = Q^m \cap \hat{H}$ implies $x \notin Q^m$. So, it is clear that $r = (x,t) \notin Q^m \times Q^m_T$ and since $s$ was chosen arbitrarily this means that $p \notin \Theta^{\text{nonb}}_{G \| T}(\hat{P}) = \hat{P}$, which is a contradiction.

(ii) It needs to be shown that $B(\sup\mathcal{CN}(G \parallel T)) = B(\sup\mathcal{CN}(G_h \parallel T))$. It has been shown that the fixed points $\hat{P}$ and $\hat{P}_h$ are the same and it is clear that the labels of all states in $G$ and $G_h$ are the same. To show that the *reachable* labels are also the same it is enough to show that $\rightarrow_{G \| T | \hat{P}} = \rightarrow_{G_h \| T | \hat{P}_h}$. Clearly, $\rightarrow_{G_h \| T} \subseteq \rightarrow_{G \| T}$ and if both are restricted to the same state set $\hat{P} = \hat{P}_h$ it must also hold that $\rightarrow_{G_h \| T | \hat{P}_h} \subseteq \rightarrow_{G \| T | \hat{P}}$. The inclusion in the other direction remains. Assume that, for some $\sigma \in \Sigma_\tau$ it holds that $(p,t)[\sigma]G \parallel T\hat{P}(p',t')$. It must be shown that $(p,t)[\sigma]G_h \parallel T\hat{P}_h(p',t')$. By lemma 2 it is clear that $\hat{P} \subseteq \hat{H} \times Q_T$ and so $(p,t)[\sigma]G \parallel T\hat{H} \times Q_T(p',t')$. Then $p \xrightarrow{\sigma}_{G\,|\hat{H}} p'$ which, by construction of $G_h$, implies that $p \xrightarrow{\sigma}_{G_h} p'$. Since $T$ is the same $(p,t) \xrightarrow{\sigma}_{G_h \| T} (p',t')$ and, since both $(p,t)$ and $(p',t')$ must be in $\hat{P} = \hat{P}_h$, it is clear that $(p,t)[\sigma]G_h \parallel T\hat{P}_h(p',t')$. ■