Modular specification of forbidden states for supervisory control [⋆]

P. Magnusson [*] **M. Fabian** [*] **K. Åkesson** [*]

[*] *Department of Signals and Systems Chalmers University of Technology, Gothenburg, Sweden (e-mail: patrikm, fabian, knut at chalmers.se)*

Abstract: A method for solving the forbidden state problem in the Supervisory Control Theory framework is presented. In many real-world applications both the plant and specification is given as a set of interacting automata or processes. In this work, we enable specification of forbidden states within such a modular structure. The aim with the method is to make each forbidden modular state combination uncontrollable. It is then possible to use efficient modular synthesis algorithms for calculation of a modular supervisor where the forbidden states are removed.

Keywords: discrete event systems, automata, modular control, supervisory control, formal specification, forbidden state problem

1. INTRODUCTION

Systems that may be abstracted to at all times occupy a single *state* from out of a finite set of states, and transit between these states on asynchronously and instantaneously occurring *events* are usefully modeled as *discrete event systems*, DESs. Examples include manufacturing systems and communication networks. The possible sequences of events may be described by *regular languages* and/or *finite state automata.*

The control of DESs arises since there is a possibility that not all sequences of events are desirable. The uncontrolled DES model, called the *plant*, needs to be restricted according to some *specification.* Synthesis of a *supervisor* that dynamically disables events in the plant based on a specification may be done according to the *supervisory control theory*, SCT, a formal framework presented in Ramadge and Wonham (1987b, 1989).

A subset of the events in the plant are not subject to disablement, these are said to be *uncontrollable.* A supervisor must never try to disable uncontrollable events, it must be *controllable.* It is known that for every plant and specification there exists a unique controllable supervisor that restricts the sequences of events as little as possible, this supervisor is said to be *minimally restrictive.*

The straight-forward *monolithic modeling*, and hence synthesis, is intractable for industrial systems as these typically encompass enormous state-spaces due to the combinatorial *state space explosion problem.* One favorable way to overcome this problem is to employ a *modular* approach; the model then consists of a number of interacting sub-plants and sub-specifications, Wonham and Ramadge (1988); Queiroz and Cury (2000). Each sub-specification typically specifies the desired or forbidden behavior of only a small part of the global plant, which is favorable for modular synthesis. This then typically results in a number of sub-supervisors, each controlling its own small, but possibly overlapping, part of the plant. Highly efficient algorithms that benefit from modular modeling do exist, see for instance Åkesson et al. (2002); Queiroz and Cury (2000); Flordal (2006).

An important sub-problem of the supervisory control theory is the *forbidden state problem.* The importance of this problem stems from the fact that it is a *safety* problem; it is concerned with bad things never happening. The forbidden state problem appears with different terminologies within the literature. *Supervision based on place invariants* is a term from the Petri net community for specifying an upper limit for the sum of tokens in a subset to the total set of places. An extensive survey is given in Iordache and Antsaklis (2006). A common characteristic of the Petri net approaches described by Iordache and Antsaklis (2006) is a monolithic specification, the L and b matrices. Though this may result in a modular supervisor, as in the decentralized case described by Iordache and Antsaklis (2006), the specification task may still be hampered. Place invariants are very similar to *predicate invariants* in the automata community. Predicate invariants specify how some state combinations from sub-systems are to remain invariant during the process. Ramadge and Wonham (1987a) prove that a predicate composed (through conjunction and/or disjunction) of sub-predicates lends itself to a modular synthesis. Though the above described approaches employ a modular synthesis approach to the forbidden state problem, they do not explicitly treat the problem of *modular specification* of forbidden states tailor made for a modular synthesis approach.

This paper deals with modular specification of forbidden states, explicitly relying on specification decomposition, Komenda et al. (2008), in the hope to reap the benefits of a modular synthesis algorithm. The modular supervisor

[⋆] Supported by the Swedish Research Council, grant number 90378601, and the European 7th Framework Programme, grant agreement number 213734 (FLEXA).

is typically an interaction of several sub-supervisors of manageable size.

A brute force method to calculate a minimally restrictive supervisor is to compose sub-plants and possible sub-specifications, remove forbidden state combinations from the composition, and finally perform synthesis by removing additional states of the composition. Another method is to create additional sub-specifications. These specifications are sequences of events that reach all forbidden state combinations, in order to forbid the last state in each sequence. The sequences should not restrict the plant behavior, only track events. Simple removal of sub-plant and sub-specification states in the initial sub-systems is then the same as removing all state combinations where any of the sub-plant or the sub-specification state exists. This will not necessarily give a minimally restrictive supervisor, Åkesson et al. (2002).

The approach presented in this paper takes advantage of the fact that state combinations where a specification disables an uncontrollable event possible in the plant will be removed by the synthesis algorithm. Each forbidden state is therefore turned into a controllability problem. This is done by introducing uncontrollable events in the concerned sub-plants and sub-specifications. The introduced events are self-looped at the respective states in the sub-plants (and possibly in some sub-specifications). Adding the events to the alphabets of the sub-specifications then creates a controllability problem so that the forbidden state combinations are removed with employment of a synthesis algorithm. This paper proofs the validity of the approach for forbidden state combinations with states from a subset of the sub-plants and zero or one sub-specification. Guidelines for how to handle states from a subset of both sub-plants and sub-specifications are given.

We assume dissimilar alphabets for all sub-automata, as among others pointed out by Åkesson et al. (2002). For clarity of presentation and without loss of generality, we assume that all given events are controllable. We will introduce new events that are uncontrollable. We do not address the problem of synthesizing a non-blocking supervisor.

The next section presents the modeling formalism used. Sections 3 and 4 describe the method, the latter with formal proofs. Illustrative examples are given in Section 5. The paper ends with some conclusions.

2. PRELIMINARIES

This section presents conventions and notations for the modeling formalism used in this paper.

Definition 1. **Deterministic Finite Automaton**
A *deterministic finite automaton* is a 4-tuple $A := \langle Q_A, \Sigma_A, \delta_A, i_A \rangle$ where Q_A is the nonempty finite set of *states*; Σ_A is the nonempty finite set of *events*, the *alphabet* for the automaton; $\delta_A : Q_A \times \Sigma_A \to Q_A$ is the partial transition function and $i_A \in Q_A$ is the initial state.

Let $\delta_A(q, \sigma)!$ ($\neg\delta_A(q, \sigma)!$) denote that an event σ is defined (undefined) from a state q for an automaton A. The set of all finite sequences of events over an alphabet Σ_A including the empty sequence, ϵ, is denoted Σ_A^*. An element $s \in \Sigma_A^*$ is called a *string*. A *language*, $L(A)$, is the set of strings accepted from the initial state, defined by an automaton A. $\delta_A(q, \sigma s)$ is equal to $\delta_A(\delta_A(q, \sigma), s)$. A state $p \in Q_A$ is *reachable* if $p = \delta_A(i_A, t)$ where $t \in L(A)$, otherwise *non-reachable*.

Definition 2. **Active event function**
The *active event function* returns the set of events defined from a state. $\Gamma(q) := \{\sigma \in \Sigma \mid \delta(q, \sigma)!\}$

Interaction of two automata may be modeled with full synchronous composition, FSC, Hoare (1985).

Definition 3. **Full synchronous composition** (FSC)
The *full synchronous composition* of two automata A and B is defined as $C := A||B$ where $Q_C := Q_A \times Q_B$; $\Sigma_C := \Sigma_A \cup \Sigma_B$; $i_C := \langle i_A, i_B \rangle$ and $\delta_C(\langle q_A, q_B \rangle, \sigma) :=$

$$\begin{cases} \langle \delta_A(q_A, \sigma), \delta_B(q_B, \sigma) \rangle & \sigma \in \Gamma_A(q_A) \cap \Gamma_B(q_B) \\ \langle \delta_A(q_A, \sigma), q_B \rangle & \sigma \in \Gamma_A(q_A) \setminus \Sigma_B \\ \langle q_A, \delta_B(q_B, \sigma) \rangle & \sigma \in \Gamma_B(q_B) \setminus \Sigma_A \\ \textit{undefined} & \textit{otherwise} \end{cases}$$

FSC models a way to supervise the behavior of an automaton P, *plant*, through an automaton S, *specification*. Automaton S disables events in automaton P. A common situation is that automaton P models some process and that automaton S models restrictions of this process.

Some of the events in an automaton P, $\Sigma_P^u \subseteq \Sigma_P$, are not subject to disablement. These are said to be *uncontrollable*. $L(S)\Sigma_P^u$ represents the *concatenation* of all strings in $L(S)$ with all events in Σ_P^u. S must never disable uncontrollable events, that is S must be controllable. This is captured by the notation of controllability.

Definition 4. **Controllability**
If $\Sigma_S \subseteq \Sigma_P$, S is controllable with respect to P and Σ_P^u if $L(P||S)\Sigma_P^u \cap L(P) \subseteq L(P||S)$

It follows directly from Definition 4 that some states may be regarded as uncontrollable.

Definition 5. **Uncontrollable state**
Let P be a plant and S a specification. $\exists t \in L(P||S)$. A state $p := \delta_{P||S}(i_{P||S}, t)$ is *uncontrollable* if there exists an uncontrollable event
$u \in \Sigma_P^u \; s.t. \; \delta_P(i_P, tu)! \wedge \neg\delta_{P||S}(i_{P||S}, tu)!$

Possible uncontrollable states in the composition $P||S$ prevents the specification $S = S_0$ from being a supervisor, it is not controllable. The plant may perform transitions that result in loss of synchronization between P and S_0. A controllable supervisor S_{n+1} may be synthesized through iterative removal of uncontrollable states in S_n, $S_1 = P||S_0$. It is known that every plant and specification has a *minimally restrictive* supervisor that restricts the sequences of events as little as possible, Ramadge and Wonham (1987b).

We end with defining some terms to simplify the further discussion.

Definition 6. **Configuration**
A *configuration* is a finite set of automata and is denoted by $A_\sqcup := \{A_1, ..., A_n\}$. A sub-configuration $B_\sqcup$ comprises a subset of automata from $A_\sqcup$, i.e., $B_\sqcup \subseteq A_\sqcup$. The FSC of a configuration is interpreted as $A := A_1||...||A_n$.

Definition 7. **Global, local, and sub-states**
With the term *local-state* we will refer to a state in a

single automaton. With the term *global-state* we will refer to a state in the FSC of a configuration. With the term *sub-state* we will refer to a state in the FSC of a sub-configuration.

Let a configuration $B_\sqcup \subseteq A_\sqcup$. Assume there exists an automaton B_i in $B_\sqcup$. A global-state in $B_\sqcup$ is a sub-state in $A_\sqcup$ and a local-state $q_{B_i} \in Q_{B_i}$ is a sub-state in both $B_\sqcup$ and $A_\sqcup$.

3. MODULAR SPECIFICATION AND SYNTHESIS

A modular supervisory control problem may now be described in terms of plant and specification configurations, $P_\sqcup := \{P_1, ..., P_n\}$ and $S_\sqcup := \{S_1, ..., S_m\}$, respectively. Add to this a specification that refers to local-states from different automata that should not appear together in the closed loop system of $P_\sqcup$ and $S_\sqcup$. We denote each specified combination of local-states as a *forbidden sub-state*. The terms local-, sub- and global-state refer to the FSC of configuration $P_\sqcup \cup S_\sqcup$ if nothing else is written. $P_1, ..., P_n$ and $S_1, ..., S_m$ are denoted sub-plants and sub-specifications, respectively.

The problem is then to synthesize the minimally restrictive supervisor with respect to these conditions. There is a high interest to keep the modularity of the system. A set of local-states from different automata is per definition a sub-state. It is most likely that this sub-state exists in many global-states. All these global-states should then be non-accessible in the supervised configuration $P_\sqcup \cup S_\sqcup$.

Modular synthesis algorithms can be very efficient to remove uncontrollable states, Åkesson et al. (2002). The algorithms make use of the modular structure of the systems in order to return the minimally restrictive supervisor. Some algorithms return sub-supervisors where the FSC of these is minimally restrictive. In-depth discussion about how a modular synthesis works is outside the scope of this paper. We will only assume that the algorithms work better with higher degree of system modularity.

The above gives the line of arguments for the proposed method. The synthesis algorithms remove uncontrollable states, so we make the forbidden states uncontrollable and use already existing algorithms. Definition 5 shows that a sub-state composed from one local plant state and one local specification state is uncontrollable if the local plant state may execute an uncontrollable event and this event is disabled in the local-state of the specification, i.e., the specification cannot follow the plant. It is therefore desirable to extend the automata for a sub-state with uncontrollable events so that the described situation occurs.

To conclude, we propose a pre-step to any already existing modular synthesis algorithm in order to guarantee that the algorithm removes all forbidden global-states and returns a minimally restrictive supervisor.

4. MODULAR AUTOMATA EXTENSION

The basic idea of the method is to make each forbidden sub-state in $P_\sqcup \cup S_\sqcup$ uncontrollable, so that the synthesis algorithm will remove it. This is achieved with introduction of an event α^i connected to each forbidden sub-state. The superscript i is unique for each forbidden sub-state. The event α^i is uncontrollable and models the linking between the local-states, and only those local-states, that compose each forbidden sub-state.

We introduce the definition α^i-uncontrollability for strings that lead to states that are uncontrollable because of α^i.

Definition 8. **α^i-uncontrollable**
Let P be a plant and S a specification. $\exists t \in L(P||S)$. A string t is *α^i-uncontrollable* if
$\delta_P\left(i_P, t\alpha^i\right)! \;\wedge\; \delta_{P||S}\left(i_{P||S}, t\right)! \;\wedge\; \neg\delta_{P||S}\left(i_{P||S}, t\alpha^i\right)!$

The significance of a state reached with an α^i-uncontrollable string is that it is uncontrollable and therefore removed in the synthesis algorithm. We will show that only the forbidden states are reached with α^i-uncontrollable strings.

We choose to describe the introduction of the α^i events into the automata in $P_\sqcup \cup S_\sqcup$ within three subsections for ease of understanding. The first subsection concerns sub-states of the type $(k > 0, l = 0)$, where k (l) is number of local sub-plant (sub-specification) states. The second and third subsection concern $(k > 0, l = 1)$ and $(k > 0, l > 1)$, respectively. The observant reader will see that the method modifies all sub-states into the type $(k > 0, l = 1)$. We neglect sub-states of the type $(k = 0, l > 0)$ as these only concern forbidden sub-specification state combinations.

4.1 Sub-states of the type $(k > 0, l = 0)$

A forbidden sub-state $\langle p_1, p_2, ..., p, ..., p_k\rangle$ in $P_\sqcup$ concerns k local-states in k sub-plants, $p \in Q_P$ and $P \subseteq P_\sqcup$. We propose that a unique event α^i is created for the forbidden sub-state. Each of the k local plant states are linked to this event through extension of the k sub-plants.

Definition 9. **Plant extension**
Let P be a plant. A local-state $p \in Q_P$ is *linked* to an event α^i through *extension* of P, denoted $\mathring{P}$, such that $Q_{\mathring{P}} := Q_P$, $\Sigma_{\mathring{P}} := \Sigma_P \dot\cup \{\alpha^i\}$, $i_{\mathring{P}} := i_P$ and $\delta_{\mathring{P}} := \delta_P \dot\cup \left\{\langle p, \alpha^i, p\rangle\right\}$

An extended sub-plant $\mathring{P}$ replaces an initial given sub-plant P in a configuration $P_\sqcup$. An extension of an extended sub-plant replaces the extended sub-plant in a configuration and so forth. This latter concerns sub-plants that link to many α^i. We continue with composition of two extended automata.

Lemma 10. FSC of k sub-plants with and without extension according to Definition 9 will at most differ with a self-loop, event α^i, at the sub-state combined from the local sub-plant states with this self-loop.

Proof. The proof follows from the definition of FSC, Definition 3. Let $P := P_1||P_2$ and $\mathring{P} := \mathring{P}_1||\mathring{P}_2$ then
$Q_{\mathring{P}} = Q_P$, $\Sigma_{\mathring{P}} \backslash \Sigma_P = \left\{\alpha^i\right\}$, $i_{\mathring{P}} = i_P$ and $\delta_{\mathring{P}}\left(\langle p_1, p_2\rangle, \sigma\right) :=$

$$\begin{cases} \langle p_1, p_2\rangle & \sigma = \alpha^i \wedge \alpha^i \in \Gamma_{\mathring{P}_1}\left(p_1\right) \cap \Gamma_{\mathring{P}_2}\left(p_2\right) \\ \textit{undefined} & \sigma = \alpha^i \wedge \alpha^i \notin \Gamma_{\mathring{P}_1}\left(p_1\right) \cap \Gamma_{\mathring{P}_2}\left(p_2\right) \\ \delta_P\left(\langle p_1, p_2\rangle, \sigma\right) & \textit{otherwise} \end{cases}$$

Recall that FSC may make the forbidden sub-state non-reachable, no difference will then exist between P and $\mathring{P}$.

A unique specification S^i_α is created for the event α^i, as in Definition 11. Uncontrollability is enabled as S^i_α always disables the event α^i. The created specification is added to the configuration $S_\sqcup$.

Definition 11. **Specification for an event α^i**
$S^i_\alpha := \langle \{q_{S^i_\alpha}\}, \{\alpha^i\}, \delta_{S^i_\alpha}, q_{S^i_\alpha} \rangle$ where $\neg\delta_{S^i_\alpha}(q_{S^i_\alpha}, \alpha^i)!$

The remainder part of this section is devoted to uncontrollable states in the composition of $\mathring{P}$ and S^i_α, where $\mathring{P}$ is the composition of the k, extended, sub-plants.

Lemma 12. FSC of $\mathring{P}$ and S^i_α allows all transitions in $\mathring{P}$ besides the self-loop, event α^i, at the state $\langle p_1, p_2, ..., p_k \rangle$.
Proof. The proof is immediate from the definition of FSC, see Definition 3. The composition
$\mathring{P}||S^i_\alpha := \left\langle Q_{\mathring{P}} \times \{q_{S^i_\alpha}\}, \mathring{\Sigma}, \delta_{\mathring{P}||S^i_\alpha}, \langle i_{\mathring{P}}, q_{S^i_\alpha} \rangle \right\rangle$ where

$$\delta_{\mathring{P}||S^i_\alpha}\left(\langle p_{\mathring{P}}, q_{S^i_\alpha} \rangle, \sigma\right) := \begin{cases} \left\langle \delta_P\left(p_{\mathring{P}}, \sigma\right), q_{S^i_\alpha} \right\rangle & \sigma \in \Gamma_P\left(p_{\mathring{P}}\right) \\ \mathit{undefined} & \mathit{otherwise} \end{cases}$$

Note that $\delta_{\mathring{P}||S^i_\alpha}(\langle p, q \rangle, \sigma)$ is undefined for all events not in $\Gamma_P(p)$; specifically, $\delta_{\mathring{P}||S^i_\alpha}(\langle p, q \rangle, \sigma)$ is undefined for $\sigma = \alpha^i$, and thus for any string ending with α^i.

k sub-plants and one sub-specification have the event α^i in their alphabet. For clarity of presentation and without loss of generality, we assume $\Sigma^u_P = \emptyset$. Thus, the uncontrollable events to check for in the controllability theorem are $\{\alpha^i\}$.

The forbidden state $\langle p_1, p_2, ..., p_k \rangle$ in $\mathring{P}$ is equal to $\langle p_1, p_2, ..., p_k, q_{S^i_\alpha} \rangle$ in the synchronization $\mathring{P}||S^i_\alpha$, because $Q_{S^i_\alpha} = \{q_{S^i_\alpha}\}$. We are now ready to connect the forbidden state with α^i-uncontrollability.

Theorem 13. Given $\mathring{P}$ and S^i_α as defined in Definition 9 and 11, a string $t \in L(\mathring{P}||S^i_\alpha)$ is α^i-uncontrollable if and only if $\delta_{\mathring{P}||S^i_\alpha}\left(\langle i_{\mathring{P}}, q_{S^i_\alpha} \rangle, t\right) = \langle p_1, p_2, ..., p_k, q_{S^i_\alpha} \rangle$.
Proof. Only strings t to the forbidden state, $\langle p_1, p_2, ..., p_k \rangle$, are α^i-uncontrollable candidates, from Lemma 10. t is α^i-uncontrollable because $\neg\delta_{\mathring{P}||S^i_\alpha}\left(\langle i_{\mathring{P}}, q_{S^i_\alpha} \rangle, t\alpha^i\right)!$, from Lemma 12.

All states reached with α^i-uncontrollable strings are removed in the succeeding synthesis algorithm.

4.2 Sub-states of the type $(k > 0, l = 1)$

A forbidden sub-state $\langle p_1, p_2, ...p, ..., p_k, q \rangle$ in $P_\sqcup \cup S_\sqcup$ concerns k local-states in k sub-plants and one local-state in one sub-specification, $p \in Q_P$ and $P \subseteq P_\sqcup$, $q \in Q_S$ and $S \subseteq S_\sqcup$. We propose that a unique event α^i is created for the forbidden sub-state. All k sub-plants are extended as in Definition 9. The initially given sub-specification S is also linked to the event α^i through extension of S.

Definition 14. **Specification extension**
Let S be a specification. A local-state $q \in Q_S$ is linked to an event α^i through extension of S, denoted $\mathring{S}$, such that $\mathring{S} := \langle Q_S, \Sigma_{\mathring{S}}, \delta_{\mathring{S}}, i_S \rangle$ where $\Sigma_{\mathring{S}} = \Sigma_S \dot\cup \{\alpha^i\}$ and

$$\delta_{\mathring{S}}(p, \sigma) := \begin{cases} p & \sigma = \alpha^i \wedge p \neq q \\ \mathit{undefined} & \sigma = \alpha^i \wedge p = q \\ \delta_S(p, \sigma) & \mathit{otherwise} \end{cases}$$

An extended sub-specification $\mathring{S}$ replaces an initial given sub-specification S in a configuration $S_\sqcup$. We continue by comparing a specification S^i_α and a specification $\mathring{S}$.

A specification S^i_α may be seen as a special case of a specification $\mathring{S}$ where $Q_{\mathring{S}} = \{q_{S^i_\alpha}\}$ and $\Gamma_{S^i_\alpha}\left(q_{S^i_\alpha}\right) = \emptyset$. Hence, we have a similar issue as in the former section, k sub-plants and one sub-specification with the event α^i in their alphabet.

Lemma 15. FSC of $\mathring{P}$ and $\mathring{S}$ allows all α^i events in $\mathring{P}||S$ besides in sub-states where it is not permitted in the local-state of the sub-specification.
Proof. The proof follows from Definition 3. $\mathring{P}||\mathring{S} := \left\langle Q_{\mathring{P}} \times Q_{\mathring{S}}, \mathring{\Sigma}, \delta_{\mathring{P}||\mathring{S}}, \langle i_{\mathring{P}}, i_{\mathring{S}} \rangle \right\rangle$ where
$\delta_{\mathring{P}||\mathring{S}}\left(\langle q_{\mathring{P}}, q_{\mathring{S}} \rangle, \sigma\right) :=$

$$\begin{cases} \langle \delta_{\mathring{P}}(q_{\mathring{P}}, \sigma), \delta_{\mathring{S}}(q_{\mathring{S}}, \sigma) \rangle & \sigma \in \Gamma_{\mathring{P}}\left(q_{\mathring{P}}\right) \cap \Gamma_{\mathring{S}}\left(q_{\mathring{S}}\right) \\ \langle \delta_{\mathring{P}}(q_{\mathring{P}}, \sigma), q_{\mathring{S}} \rangle & \sigma \in \Gamma_{\mathring{P}}\left(q_{\mathring{P}}\right) \setminus \Gamma_{\mathring{S}}\left(q_{\mathring{S}}\right) \\ \mathit{undefined} & \mathit{otherwise} \end{cases}$$

and explicitly for $\sigma = \alpha^i$, $\delta_{\mathring{P}||\mathring{S}}\left(\langle q_{\mathring{P}}, q_{\mathring{S}} \rangle, \alpha^i\right) :=$

$$\begin{cases} \langle q_{\mathring{P}}, q_{\mathring{S}} \rangle & \alpha^i \in \Gamma_{\mathring{P}}\left(q_{\mathring{P}}\right) \cap \Gamma_{\mathring{S}}\left(q_{\mathring{S}}\right) \\ \mathit{undefined} & \mathit{otherwise} \end{cases}$$

We end with an extension of the proof in Theorem 13 in order to show the linking between α^i-uncontrollability and the forbidden state.

Theorem 16. Given $\mathring{P}$ and $\mathring{S}$ as defined in Definition 9 and 14, a string $t \in L(\mathring{P}||\mathring{S})$ is α^i-uncontrollable if and only if $\delta_{\mathring{P}||\mathring{S}}\left(\langle i_{\mathring{P}}, i_{\mathring{S}} \rangle, t\right) = \langle p_1, p_2, ..., p_k, q \rangle$.
Proof. $\delta_{\mathring{P}||\mathring{S}}\left(\langle i_{\mathring{P}}, i_{\mathring{S}} \rangle, t\right) = \langle p_1, p_2, ..., p_k, q \rangle$ if q disables α^i, from Theorem 13. There exist such states q in $\mathring{P}||\mathring{S}$, from Lemma 15. q is the single state in $\mathring{S}$ that disables α^i, from Definition 14.

The subsequent synthesis algorithm removes all states reached with α^i-uncontrollable strings.

4.3 Sub-states of the type $(k > 0, l > 1)$

We believe that the previous two subsections handle the corpus of problems with forbidden states, but the most general sub-state still remains. The most general forbidden sub-state $\langle p_1, ..., p, ..., p_k, q_1, ..., q, ..., q_l \rangle$ in $P_\sqcup \cup S_\sqcup$ concerns k local-states in k sub-plants and l local-states in l sub-specifications, $p \in Q_P$ and $P \subseteq P_\sqcup$, $q \in Q_S$ and $S \subseteq S_\sqcup$. We propose that a unique event α^i is created for the forbidden sub-state. All k sub-plants are extended as in Definition 9.

The proposed automaton extension method for a single local specification state, Definition 14, is not applicable for sub-states with more than one local sub-specification state. FSC removes the self-loop, event α^i, from all sub-states where α^i is not in all local-states. This is the same as making all sub-state combinations uncontrollable where any of the forbidden local sub-specification states exists. This will most certainly not give a minimally restrictive supervisor.

For this reason we propose two alternative approaches for sub-states with more then one local sub-specification states that to some extent answer our intention to represent forbidden sub-states as uncontrollable states and still preserve modularity.

In a first approach, each local-state $q_1, ..., q_l$ is linked to an event α^i as in Definition 14. The extended sub-specifications, $\mathring{S}_1, ..., \mathring{S}_l$, are composed to a single sub-specification $\mathring{S}$ with prioritized synchronous composition, see Heymann (1990). All events are prioritized besides the event α^i. We will then have a forbidden sub-state of the type $(k > 0, l = 1)$. Sub-states of that type are processed in section 4.2.

A second approach requires modifications within the controllability verification part of the synthesis algorithm. The concerned sub-specifications are extended as in Definition 14, but not with the same event α^i as before, instead with a unique event α^i_j $j := 1, ..., l$. All sub-states, besides one (zero), in the FSC of extended sub-specifications will always comprise at least one self-loop, event α^i_j. There is one (zero) sub-state without self-loops if our forbidden sub-state, $\langle q_1, ..., q_l \rangle$, is reachable (non-reachable) in the FSC. The controllability verification part is implemented not to distinguish between events with and without index j. The sub-state of our concern will be the single possible uncontrollable sub-state for the uncontrollable events α^i and α^i_j. Section 4.2 is then a special case where only one value is needed for index j.

To summarize the second approach, the forbidden sub-state with $k + l$ local-states is reflected in $1 + l$ events in FSC and a single event in controllability verification.

5. EXAMPLES

The proposed method is illustrated with three examples. Two three state users from Ramadge and Wonham (1987b) compose sub-plants.

5.1 Mutual exclusion with two sub-plants

The task is to create a supervisor that prevents two machines from using a shared resource at the same time, a normal safety issue. Each machine is modeled as the automaton in figure 1. FSC of these two sub-plants gives the total plant. Both machines using the shared resource is then equal to the state UU.

The problem may be solved as described in section 4.1. A self-loop with an uncontrollable event α^i is added to the U state for each sub-plant. A specification automaton S^i_α is created for the α^i event, $Q_{S^i_\alpha} = \{M\}$.

The synchronous composition of the sub-plants with and without the specification S^i_α is seen in figure 2. The two cases are the same besides the self-loop with event α^i that is present (disabled) in the case without (with) S^i_α. The state UUM is uncontrollable, thus removed in a succeeding synthesis algorithm.

5.2 Inclusion of local-state from a specification

A similar but somewhat harder problem arises when the machines may operate in two modes; manual and auto. Both machines may access the shared resource in the manual mode but not in the auto mode. Switching between the modes is only feasible in the idle state of a machine. This latter may be modeled as a three state user modified according to figure 3. A specification S for the switching may be modeled as in figure 4.

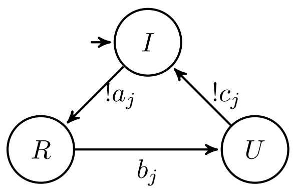

Fig. 1. Sub-plant with states; I: idle, R: request, U: use. The supervisor may neither disable access request event a_j nor leaving event c_j, so these are modeled as uncontrollable events. Access event b_j is modeled as controllable.

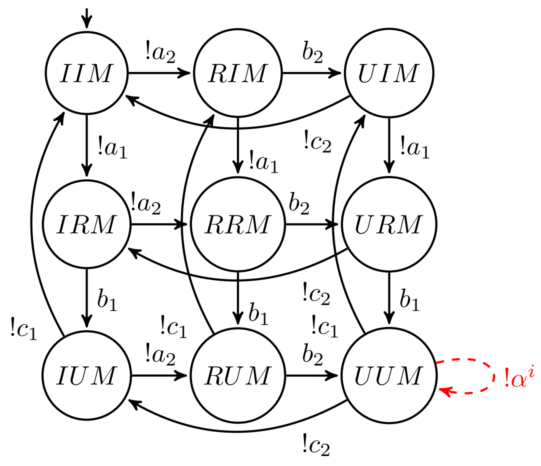

Fig. 2. Synchronization of two sub-plant automata as of figure 1 and one specification automaton for the α^i event. The dashed transition is disabled as cause of synchronization with the specification. Thus, the state UUM is uncontrollable.

The task is to create a supervisor that prevents two machines from using a shared resource at the same time in auto mode and permits the two machines to use the shared resource at the same time in manual mode.

The problem may be solved as described in section 4.2. The sub-plants undergo the same extensions as in the last subsection. The α^i event is added as a self-loop to the manual state, the allowed state, in the specification.

Synchronization of the sub-plants and the specification $\mathring{S}$ is seen in figures 5 and 2. The state UUA is uncontrollable in relation to the synchronization of the two sub-plants. The self-loop with event α^i in state UUM may be purged from the synthesized supervisor.

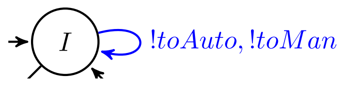

Fig. 3. Modification of automaton in figure 1 in order to only allow mode switch in state I.

5.3 Arithmetic example with many users

The strength of the proposed method is colorfully illustrated with an arithmetic continuation of the example in section 5.1. What is the cost of synthesizing a supervisor that prevents all sub-states $\langle U_i, U_j \rangle$ $s.t.$ $i \neq j$ in N users? We assert that a monolithic and modular synthesis

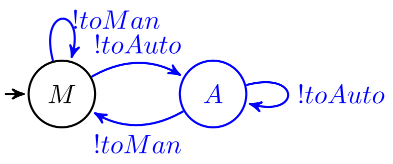

Fig. 4. Specification for how mode switch is allowed to occur. State; M: Manual, A: Auto. None of the events may be controlled from the supervisor, thus uncontrollable.

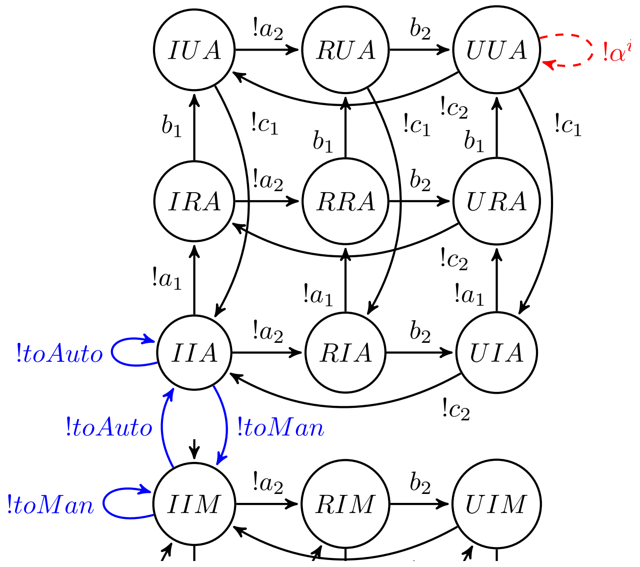

Fig. 5. Synchronization of two sub-plant automata as of figure 3 and one specification automaton as of figure 4 with addition of a α^i self-loop in the manual state. Lower part of automaton as in figure 2 but with a solid self-loop in state UUM. The dashed transition from state UUA is disabled as cause of synchronization with the specification, this gives an uncontrollable state.

algorithm can be compared with respect to the number of states that it needs to take into consideration for calculation of the minimally restrictive supervisor. Thus, the cost is lower with fewer states.

N users have $N \cdot (N-1)/2$ forbidden sub-states $\langle U_i, U_j \rangle$. We assume that a monolithic algorithm requires a specification where each sub-state is removed. This gives $3^N - N \cdot (N-1)/2$ number of states to consider in the algorithm. A modular algorithm composes all sub-plants and sub-specifications that share uncontrollable events, Åkesson et al. (2002). One specification S^i_α is created for each sub-state. This gives $N \cdot (N-1)/2 \cdot 3^2$ number of states to consider in the algorithm. See table 1.

We have used the implementation of modular synthesis from Åkesson et al. (2002), but any modular synthesis algorithm would benefit from our approach to modular specification of forbidden states.

6. CONCLUSIONS

We have shown how to specify forbidden state combinations within a set of modular plants and specifications. Each specified state combination is made uncontrollable.

Table 1. Number of states to consider in synthesis algorithms in order to synthesize supervisor for N users and no sub-states $\langle U_i, U_j\rangle$ $s.t.\ i \neq j$

N	3	4	10	50
Monolithic	24	75	$\sim 6e3$	$\sim 7e23$
Modular	27	54	405	$\sim 1e3$

These uncontrollable states are removed by a modular synthesis algorithm, Åkesson et al. (2002). The algorithm returns a modular supervisor where the forbidden state combinations are removed. The focus in this paper has been on many modular plant states and zero or one modular specification state in the forbidden state combinations. Our modeling experience tells us that this constitutes the corpus of problems with forbidden states.

Future research concerns decreasing the number of events that make the forbidden state combinations uncontrollable, in order to minimize computer memory use. Hence, two forbidden state combinations (a, b) and (a, c) may use the same event α^i if $a \in A$ and $b, c \in B$, where A and B are two sub-plants. Future research also concerns a formalization of the ideas for the most general forbidden state type, given in this paper. See *www.supremica.org* for the latest implementation.

REFERENCES

Åkesson, K., Flordal, H., and Fabian, M. (2002). Exploiting Modularity for Synthesis and Verification of Supervisors. In *15th Triennial World Congress of the International Federation of Automatic Control.*

Flordal, H. (2006). *Compositional Approaches in Supervisory Control with Application to Automatic Generation of Robot Interlocking Policies.* PhD Thesis, Chalmers.

Heymann, M. (1990). Concurrency and Discrete Event Control. *IEEE Control Systems Magazine*, 10(4), 103–112.

Hoare, C.A.R. (1985). *Communicating Sequential Processes.* Prentice-Hall International Series in Computer Science.

Iordache, M.V. and Antsaklis, P.J. (2006). Supervision Based on Place Invariants: A Survey. *Discrete Event Dynamic Systems*, 16(4), 451–492.

Komenda, J., van Schuppen, J., Gaudin, B., and Marchand, H. (2008). Supervisory control of modular systems with global specification languages. *Automatica*, 44(4), 1127–1134.

Queiroz, M.H. and Cury, J.E.R. (2000). *Modular Supervisory Control of Large Scale Discrete Event Systems*, 103–110. Kluwer Academic Publishers.

Ramadge, P.J. and Wonham, W.M. (1987a). Modular Feedback Logic for Discrete Event Systems. *SIAM Journal on Control and Optimization*, 25(5), 1202–1218.

Ramadge, P.J. and Wonham, W.M. (1987b). Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1), 206–230.

Ramadge, P.J. and Wonham, W.M. (1989). The Control of Discrete Event Systems. *Proc. of IEEE*, 77(1), 81–89.

Wonham, W.M. and Ramadge, P.J. (1988). Modular Supervisory Control of Discrete-Event Systems. *Mathematics of Control Signals and Systems*, 1(1), 13–30.