2024 WODES Workshop on Discrete Event Systems Rio de Janeiro, Brazil

17th IFAC Workshop on Discrete Event Systems
April 29-May 1, 2024. Rio de Janeiro, Brazil

Supervisor synthesis under partial observation of uncontrollable events using full observation synthesis ⋆

Martijn Goorden * **Michel Reniers** **

* *Department of Computer Science, Aalborg University, Denmark (e-mail: mgoorden@cs.aau.dk)*
** *Department of Mechanical Engineering, Eindhoven University of Technology, The Netherlands (e-mail: m.a.reniers@tue.nl)*

Abstract: This paper presents an approach towards the synthesis of maximally permissive, safe, controllable, control consistent and nonblocking supervisors where the supervisor is assumed to have partial observation of the events in the plant that is modeled as a discrete event system. In contrast to existing work, the presented approach avoids the use of concepts such as observability and normality and only relies on a transformation of the plant into another discrete event system to which subsequently supervisory control synthesis with full observation is applied.

Keywords: discrete event systems, supervisory control synthesis, partial observation.

1. INTRODUCTION

Supervisory control theory of Ramadge and Wonham (1987) provides a formal framework to analyze the problem of controlling discrete event systems (DES). Based on a model of the uncontrolled plant and a model of the control specification, a correct-by-construction supervisor can be synthesized. This supervisor ensures that the controlled system is safe and nonblocking while also being controllable and maximally permissive.

In the basic control problem setting, all events specified in the plant model are observable. In practice though, not everything might be observable, for example due to a limited number of sensors. Supervisory control for partially observed systems was initially studied in Cieslak et al. (1988); Lin and Wonham (1988). To deal with partial observability, the notion of observability of a specification language is introduced to establish if such a specification can be exactly realized by means of a supervisor. The condition entails that if a continuation with a certain event is prohibited by the specification, then from all observationally equivalent observations the event must also be prohibited.

In practice, it is often not possible to exactly realize the specification language. In this situation, as much safe behavior as possible is desired. Since observability is not closed under union, no supremal solution exists in general. Various approaches have been proposed to resolve this situation. A stronger condition that is closed under union, normality, is introduced in Cieslak et al. (1988); Lin and Wonham (1988) as a work around, but this comes at the price of sacrificing maximal permissiveness. In Takai and Ushio (2003) a class of observable sublanguages has been identified that is closed under the union operation of strict subautomata. More recently, Cai et al. (2015) proposed the notion of relative observability that is stronger than observability but weaker than normality. Neither Takai and Ushio (2003) nor Cai et al. (2015) may always obtain a maximally permissive supervisor in general. In Hu et al. (2023) a supervisor is allowed to use quiescent information such that a maximally permissive nonblocking supervisor can be obtained. Effectively, a supervisor is able to time-out when no observable event has been registered for a while, thus being able to identify dead states.

The first paper that solves the problem of synthesizing a maximally permissive, safe, controllable, nonblocking supervisor for a plant and a specification is Yin and Lafortune (2016). It is assumed that the specification is controllable and observable. In the paper, such a supervisor is synthesized in a number of steps. First, an All Inclusive Controller (AIC) is generated from the plant that represents all safe controllers. This AIC is a bipartite transition system which represents a game between a controller (proposing a set of enabled events) and a plant (selecting one such proposed event). From the AIC a NonBlocking All Inclusive Controller (NB-AIC) is composed that represent all nonblocking safe controllers. On the NB-AIC a dedicated synthesis algorithm is applied that achieves an information-state-based supervisor (i.e., a supervisor that each time when confronted with the same information state makes the same control decision) and removes livelocks by iteratively unfolding states that represent a livelock.

In this paper, we propose a different solution for synthesizing supervisors for partially observed DES using supervisor synthesis for fully observable plants. In our set-up, there are no assumptions on the specification, so controllability and observability are not assumed. For the case where all controllable events are observable, in Section 4, we transform the original partially observable plant automaton into

⋆ This research is partly funded by the Digital Research Centre Denmark (DIREC), Innovation Fund Denmark.

a fully observable (plant) automaton and some additional information (to be elaborated later) in such a way that we can apply full observation supervisory control synthesis to obtain a supervisor for the partially observable plant. Before we elaborate on this transformation, in Section 3, we analyze the reasons why projection of the partially observable plant onto the observable events does not provide us with a full observable plant that can be used for fully observable synthesis. The cause is that the projection loses information about blocking. We circumvent this problem by collecting nonblocking providing events for each state before performing projection. This information allows us to perform synthesis on the fully observable, projected plant.

In Su et al. (2008), an abstraction operator is proposed (that operates on a network of automata) that preserves the nonblocking property and may also be used to remove the unobservable events from the plant. Nevertheless, as the abstraction may result in a nondeterministic plant (and with an additional new event τ) we cannot directly apply standard synthesis algorithms. Inan (1994) also uses projection and nondeterminism to show the existence of a solution, but, based on this work, the algorithm of Yoo and Lafortune (2006) might not result in a maximally permissive supervisor.

2. PRELIMINARIES

An automaton is a five-tuple $G = (Q, \Sigma, \rightarrow, q^0, Q^m)$, where Q is the (finite) nonempty state set, Σ is the alphabet of events, $\rightarrow \subseteq Q \times \Sigma \times Q$ the partial transition relation, $q^0 \in Q$ the initial state, and $Q^m \subseteq Q$ the set of marked states. Such an automaton is called *deterministic* in case $(q, \sigma, q') \in \rightarrow$ and $(q, \sigma, q'') \in \rightarrow$ implies that $q' = q''$ for all $q, q', q'' \in Q$ and $\sigma \in \Sigma$. In the remainder of this paper, it is assumed that all automata are deterministic.

Σ^* represents the Kleene closure of Σ, i.e. the set of all finite strings of events in Σ, including the empty string ε. We use the infix notation for transitions: we write $q \xrightarrow{\sigma} q'$ to indicate that $(q, \sigma, q') \in \rightarrow$. We extend this infix notation to words $w \in \Sigma^*$ in the usual way. Furthermore, we write $q \xrightarrow{w}$ iff there exists $q' \in Q$ such that $q \xrightarrow{w} q'$. The events enabled from a state q in an automaton G are denoted $En(q, G)$ and defined by $En(q, G) = \{\sigma \in \Sigma \mid q \xrightarrow{\sigma}\}$. The *language* generated by the automaton G is $L(G) = \{w \in \Sigma^* \mid q^0 \xrightarrow{w}\}$ and the *language marked* by the automaton G is $L_m(G) = \{w \in \Sigma^* \mid \exists_{q' \in Q^m} q^0 \xrightarrow{w} q'\}$.

A state q of an automaton is called *reachable* if there exists a word $w \in \Sigma^*$ such that $q_0 \xrightarrow{w} q$. The set of all reachable states of an automaton G is denoted by $Reach(G)$. An automaton G is called *reachable* if every state $q \in Q$ is reachable, i.e., $Reach(G) = Q$. A state q is *coreachable* if there exist a word $w \in \Sigma^*$ and a marked state $q' \in Q^m$ such that $q \xrightarrow{q} q'$. The set of all coreachable states of an automaton G is denoted by $Coreach(G)$. An automaton G is called *coreachable* if every state $q \in Q$ is coreachable., i.e., $Coreach(G) = Q$. A state is called *nonblocking* if it is reachable and coreachable. An automaton is called *nonblocking* if every reachable state is coreachable. An automaton is called *trim* if it is reachable and coreachable. Notice that a trim automaton is nonblocking, but a nonblocking automaton may not be trim, since it may have unreachable states. In the remainder of this paper, it is assumed that every automaton is (made) reachable and hence a state is coreachable iff it is nonblocking.

The alphabet $\Sigma = \Sigma_c \cup \Sigma_u$ is partitioned into two disjoint sets containing the *controllable* events (Σ_c) and the *uncontrollable* events (Σ_u). Some events may not be observable to the environment of an automaton. Therefore, the alphabet $\Sigma = \Sigma_o \cup \Sigma_{uo}$ is partitioned into two disjoint sets containing the *observable* events (Σ_o) and the *unobservable* events (Σ_{uo}).

Natural projection, also simply called projection, from a set of events to a smaller set of events erases events that do not belong in the smaller set. In supervisory control synthesis literature projection is typically defined on the level of languages, e.g., Cassandras and Lafortune (2009), or on the automata-level by replacing the unobservable events by an empty transition (denoted ε or λ) and a subsequent determinization step (Ware and Malik, 2008). In this paper, we provide a mathematical definition that results in a deterministic projection automaton.

Definition 1. (E-closure). For a given automaton G and a given set of events $E \subseteq \Sigma$, we define the E-closure of a set of states $Q' \subseteq Q$, notation $cl_E(Q')$, as the smallest subset of Q that satisfies (1) $Q' \subseteq cl_E(Q')$ and (2) for all $q, q' \in Q$ and $e \in E$, whenever $q \xrightarrow{e} q'$ and $q \in cl_E(Q')$, then $q' \in cl_E(Q')$.

Our notion of closure is a (trivial) generalization of the notion of unobservable reach (denoted UR) from Cassandras and Lafortune (2009) to an arbitrary set of events E, i.e., for E the set of unobservable events the notions coincide.

Using the notion of closure, the projection of an automaton to a given set of events O, typically observing at least the observable events (in this paper), can be defined as follows.

Definition 2. (Projection). Given automaton G and $O \subseteq \Sigma$. Then the projection automaton $\pi_O(G) = (Q_\pi, \Sigma_\pi, \rightarrow_\pi, q^0_\pi, Q^m_\pi)$ is defined as follows:

- $Q_\pi = \{cl_{\overline{O}}(Q') \mid Q' \subseteq Q\}$,
- $\Sigma_\pi = O$,
- $\rightarrow_\pi$ is the smallest relation that satisfies, for $Q_s \in Q_\pi$ and $\sigma \in \Sigma_\pi$: if $q_s \xrightarrow{\sigma} q_t$ for some $q_s \in Q_s$ and $q_t \in Q$, then $Q_s \xrightarrow{\sigma}_\pi cl_{\overline{O}}(\{q' \in Q \mid \exists_{q_s \in Q_s}\ q_s \xrightarrow{\sigma} q'\})$,
- $q^0_\pi = cl_{\overline{O}}(\{q_0\})$,
- $Q^m_\pi = \{Q' \in Q_\pi \mid Q' \cap Q^m \neq \varnothing\}$.

When applied with O the set of observable events, the reachable part [1] of $\pi_O(G)$ and $Obs(G)$ (from Cassandras and Lafortune (2009)) are identical.

Note that the automaton that results from projection is deterministic and has the same language and marked language as the original automaton (when restricting words to events from O). Similar properties are provided in Cassandras and Lafortune (2009) for their observer automaton. Also note that the projection defined here when applied with the set of all events coincides with determinization

[1] The reachable part of an automaton $G = (Q, \Sigma, \rightarrow, q^0, Q^m)$ is defined to be the automaton $(Reach(G), \Sigma, \rightarrow \cap (Reach(G) \times \Sigma \times Reach(G)), q^0, Q^m \cap Reach(G))$.

as presented for automata throughout literature, the so-called subset-construction.

Proposition 3. (Projection). For arbitrary automaton G with $O \subseteq \Sigma$ we have

(1) $\pi_O(G)$ is deterministic,
(2) $L(\pi_O(G)) = P_O(L(G))$, and
(3) $L_m(\pi_O(G)) = P_O(L_m(G))$,

where P_O denotes the *natural projection* of a language on the set of events O as in Cassandras and Lafortune (2009), for example.

A supervisor can be represented as a (strict) subautomaton of the plant, though in literature also different representations exists. E.g., in Ramadge and Wonham (1989) a supervisor is represented as a mapping from states of the plant to sets of events (that are allowed to occur). In supervisory control theory with full observation it is assumed that the alphabet of the supervisor is contained in that of the plant (Wonham and Cai, 2019), while in the setting with partial observability, it is assumed that the alphabet of the supervisor Σ_S is contained in the sets of observable events and controllable events of the plant, i.e., $\Sigma_S \subseteq \Sigma_o \cup \Sigma_c$ (Wonham and Cai, 2019; Cassandras and Lafortune, 2009). In this paper, we use the notion of strict subautomaton, which is also used in Yin and Lafortune (2016), as this format naturally arises in synthesis with full observability and synthesis with normality (Cho and Marcus, 1989).

Definition 4. (Strict subautomaton). An automaton $S = (Q_S, \Sigma_S, \rightarrow_S, q_S^0, Q_S^m)$ is a strict subautomaton of an automaton $G = (Q, \Sigma, \rightarrow, q^0, Q^m)$ iff $Q_S \subseteq Q$, $\Sigma_S = \Sigma$, $\rightarrow_S = \rightarrow \cap (Q_S \times \Sigma_S \times Q_S)$, $q_S^0 = q^0$, and $Q_S^m = Q^m \cap Q_S$.

The closed-loop behaviour of a plant and a supervisor is defined using parallel composition, denoted by $\|$, see Cassandras and Lafortune (2009) for definition.

The objective of supervisory control theory (Ramadge and Wonham, 1987, 1989; Cassandras and Lafortune, 2009; Komenda, 2013; Wonham and Cai, 2019) is to design an automaton called a supervisor (with an event set that does not contain events that are both uncontrollable and unobservable) that has the function to dynamically disable controllable events so that the closed-loop system of the plant and the supervisor obeys some specified behavior. Different ways exist in literature to specify allowed behaviors, such as by means of automata, event conditions and state invariants (Markovski et al., 2010). In this paper, however, we use the specification of a set of forbidden plant states. More formally, given a plant model $G = (Q, \Sigma, \rightarrow_G, q^0, Q^m)$ and given a set of forbidden states $F \subseteq Q$, the goal is to synthesize supervisor S (with state set Q_S) that adheres to the following control objectives.

- *Safety*: the closed-loop system $G \parallel S$ may never reach a state in which the plant would be in a forbidden state, i.e., $Reach(G \parallel S) \cap (F \times Q_S) = \varnothing$.
- *Controllability*: uncontrollable events may never be disabled by the supervisor, i.e., for all reachable states $(q_G, q_S) \in Reach(G \parallel S)$, if $q_G \xrightarrow{u}_G$ for some $u \in \Sigma_u$, then $(q_G, q_S) \xrightarrow{u}$ (where $\rightarrow$ denotes the transition relation of $G \parallel S$).

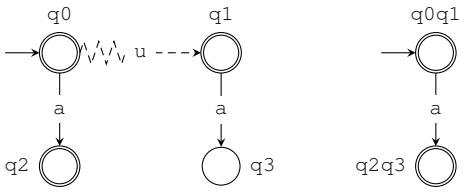

Fig. 1. Plant automaton and its observer automaton. States with different markings are merged. The observer automaton cannot be used directly for synthesis.

- *Control consistency*: the supervisor S is control consistent w.r.t. the plant G, i.e., for any two observationally equivalent states q_1 and q_2 from the initial state q_0 in the closed-loop system $G \parallel S$ (i.e., states such that $q_0 \xrightarrow{w_1} q_1$ and $q_0 \xrightarrow{w_2} q_2$ for some $w_1, w_2 \in \Sigma^*$ with $P_{\Sigma_o}(w_1) = P_{\Sigma_o}(w_2)$) it is the case that $q_1 \xrightarrow{\sigma}$ iff $q_2 \xrightarrow{\sigma}$ for any $\sigma \in \Sigma_c$.
- *Nonblockingness*: the closed-loop system $G \parallel S$ should be nonblocking.

A supervisor that is safe, controllable, control consistent, and nonblocking is called *proper*. A proper supervisor S is called *maximally permissive* when there does not exist a proper supervisor S' such that $L(G \parallel S) \subset L(G \parallel S')$.

Note that for fully observable systems and for partially observable systems where all controllable events are observable, there can be many (automaton-based) maximally permissive supervisors. However, they all result in the same closed-loop behavior.

3. INADEQUACY OF OBSERVER AUTOMATA

It is not possible to only use the observer automaton obtained by projection on the observable events for synthesizing a supervisor for the partially observable plant. This is illustrated by the following example.

Example 5. Consider the plant automaton with unobservable and uncontrollable event `u` as depicted in the left part of Fig. 1 (unobservable transitions are depicted with a zigzag arrow, uncontrollable ones with a dashed arrow). Applying supervisory control synthesis to this plant should result in the event `a` being disabled (in states `q0` and `q1`). However, in the projection automaton, shown in the figure on the right, there seems to be no reason to disable the event `a` as there are no blocking states in the projection automaton.

As mentioned in Example 5, the main problem with using projection to obtain a fully observable plant from a partially observable plant is that the property of blocking is not invariant w.r.t. projection. The partially observed plant is blocking, but its fully observable projection version is not! Consequently, a subsequent synthesis procedure will fail to remove those blocking states that are observationally indistinguishable from a nonblocking state (in the example, `q3` is a blocking state that is indistinguishable from nonblocking state `q2`). Note that if a partially observed plant is nonblocking, then also its fully observable projection is nonblocking.

Example 6. The example from Fig. 1 shows that merging states with different marking is problematic with the

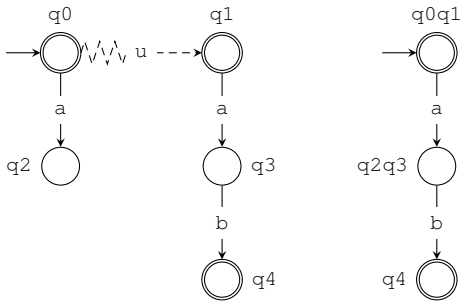

Fig. 2. Plant automaton and its observer automaton. No states with different marking are merged; nevertheless, the observer automaton cannot be used directly for synthesis.

traditional projection. But even if all merged states agree on their marking, blocking states in the partially observed plant can become nonblocking in the fully observable projection. Fig. 2 illustrates this. In the partially observed plant, state q2 is blocking, while state q3 is nonblocking, even though both states are not marked. In the partially observed system, event a should be disabled from state q0. After applying the traditional projection, it seems that event a is not required to be disabled from state q0q1.

From the previous two examples, we can conclude that a fully observable version of the plant that we can use for synthesis of the partially observed plant needs to preserve the blocking status of states. Note that the (reachable) blocking states of automaton G can be identified as follows: $B(G) = \mathit{Reach}(G) \setminus \mathit{Coreach}(G)$.

Example 7. Considering the plant automaton G in Fig. 3. One can observe that only state q5 is blocking, all other states are nonblocking. This means that a supervisor is expected to prevent the controlled system from reaching state q5 and it can achieve this by disabling the event b from states q3 (because enabling b on that state would allow the system to reach state q5) and q2 (because it is observationally equivalent to state q3). The interesting situation that now arises is that the disabling of event b from state q2 has the effect that state q2 becomes blocking. So simply remembering whether states are blocking or nonblocking in the original plant is insufficient.

From the examples introduced in this section it can be concluded that (1) the observer automaton obtained by traditional projection is not sufficient for synthesizing a supervisory controller for the partially observable plant, and (2) incorporating the blocking status of states from the plant alone is also not sufficient. The last example shows that this blocking status may change as a consequence of the desire to achieve control consistency.

4. SYNTHESIS UNDER PARTIAL OBSERVATION OF UNCONTROLLABLE EVENTS

In this section, we formulate the synthesis of a maximally permissive proper supervisor for a plant under partial observation of uncontrollable events. This means that it is assumed that all controllable events (of the plant) are observable (to the supervisor), i.e., $\Sigma_c \subseteq \Sigma_o$. Note that in this case, the event set of the supervisor only considers observable events as $\Sigma_S \subseteq \Sigma_o \cup \Sigma_c = \Sigma_o$. This assumption is not restrictive in practice, and is also common in several other approaches, such as Hu et al. (2023); Tai et al. (2023).

Our solution to obtain a maximally permissive proper supervisor for a partially observable plant is provided in Algorithm 2, $\mathtt{SCS}^{\mathtt{po}}$, and informally introduced as follows:

(1) Compute the projection automaton w.r.t. the observable events.
(2) Compute for the plant for each state whether it is nonblocking or not, and if it is nonblocking and not marked, compute the set of all enabled events from such a state that connect to a nonblocking successor state. We call this set the *nonblocking providing events*, denoted by *NB*. Algorithm 1, NBP, a variant of a standard nonblocking computation algorithm, is used for this purpose.
(3) Define the set of forbidden states to be those multi-states that contain a blocking or forbidden state from the plant. These are the non-marked states for which none of the observable nonblocking providing events are enabled anymore.
(4) Perform synthesis on this projection plant such that the forbidden states are avoided.
(5) Repeat the previous two steps until synthesis results in an automaton that does not change anymore.

In Algorithm 1 for each non-marked state a set of events is computed that connect that state to a coreachable state. The algorithm performs a backward reachability analysis much the same as in algorithms for computing nonblocking states (see e.g., Ouedraogo et al. (2011)). Instead of simply recording whether a state is nonblocking it collects all events for which a transition labeled with that event is available to a nonblocking state. This information will later be used in synthesizing a supervisor. Note that all marked states and states for which $NB(q)$ is nonempty are nonblocking. The blocking states are the non-marked states for which this set is empty.

Proposition 8. (Termination). Algorithm 1 terminates. [2]

Theorem 9. (Correctness). Let a plant G be given. For all $q \in Q \setminus Q^m$ and $\sigma \in \Sigma$: $\sigma \in \mathtt{NBP}(G)(q)$ iff $q \xrightarrow{\sigma} q'$ for some nonblocking state $q' \in Q$.

In Algorithm 2 an algorithm for supervisory control under full observation $\mathtt{SCS}^{\mathtt{fo}}$ is used. It is assumed that for an input automaton, say G, and a set of forbidden states F, this algorithm results in an automaton, say S, such that (1) S is a strict subautomaton of G that is achieved by removing zero or more states from G, and (2) S is a maximally permissive proper supervisor for G. Such algorithms are readily available in literature (Cassandras and Lafortune, 2009) and in tools such as CIF (van Beek et al., 2014; Fokkink et al., 2023) and Supremica (Åkesson et al., 2006). In Algorithm 2 we use the notation $Q_{\pi,i}$ to indicate the state set of G_i^{fo}. We explicitly include π in this notation, as individual states of G_i^{fo} are sets of original states from G due to the projection on line 1.

Example 10. In Fig. 3, besides plant G, also $\pi_{\Sigma_o}(G) = G_0^{fo}$, G_1^{fo}, and $\mathtt{SCS}^{\mathtt{po}}(G) = G_2^{fo}$ are given. Note that state

[2] Due to page limitations, no proofs are provided in this conference paper. They are though available upon request.

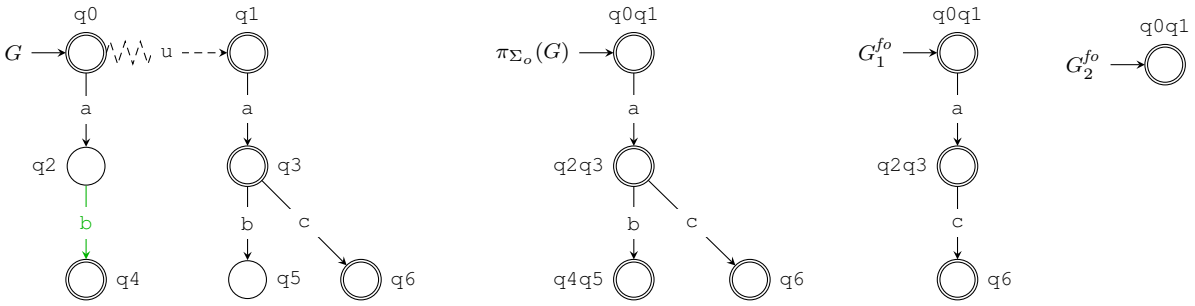

Fig. 3. Plant automaton G, its projection $\pi_{\Sigma_o}(G)$, and the results of the first two iterations of $\texttt{SCS}^{\texttt{po}}$. In G, the nonblocking providing events (for the non-marked states) are indicated by green outgoing transitions.

Algorithm 1 Nonblocking providing events: NBP.

Require: plant $G = (Q, \Sigma, \rightarrow, q^0, Q^m)$
Ensure: mapping $NB : Q \setminus Q^m \mapsto 2^\Sigma$ such that for all $q \in Q \setminus Q^m$ and $\sigma \in \Sigma$: $\sigma \in NB(q)$ iff $q \xrightarrow{\sigma} q'$ for some coreachable state $q' \in Q$

$i \leftarrow 0$
for $q \in Q \setminus Q^m$ **do**
  $NB_0(q) \leftarrow \varnothing$
end for
repeat
  for $q \in Q \setminus Q^m$ **do**
    $A \leftarrow \varnothing$
    for $\sigma \in En(q, G)$ **do**
      Let q' be the state reached after σ: $q \xrightarrow{\sigma} q'$.
      if $q' \neq q \wedge (q' \in Q^m \vee NB_i(q') \neq \varnothing)$ **then**
        $A \leftarrow A \cup \{\sigma\}$
      end if
    end for
    $NB_{i+1}(q) \leftarrow NB_i(q) \cup A$
  end for
  $i \leftarrow i + 1$
until $NB_i = NB_{i-1}$
$NB \leftarrow NB_i$

Algorithm 2 Supervisory Control Synthesis: $\texttt{SCS}^{\texttt{po}}$.

Require: plant $G = (Q, \Sigma, \rightarrow, q^0, Q^m)$ with observable events $\Sigma_o \subseteq \Sigma$, controllable events $\Sigma_c \subseteq \Sigma_o$, and forbidden states $F \subseteq Q$
Ensure: Maximally permissive proper supervisor S for G

$G_0^{fo} \leftarrow \pi_{\Sigma_o}(G)$
$S \leftarrow \texttt{SCS}^{\texttt{po}}_{\texttt{core}}(G, G_0^{fo})$

function $\texttt{SCS}^{\texttt{po}}_{\texttt{core}}(G,\ G_0^{fo})$
  $N \leftarrow \texttt{NBP}(G)$
  $i \leftarrow 0$
  repeat
    $F_i \leftarrow \varnothing$
    for $Q' \in Q_{\pi,i}$ **do**
      for $q \in Q' \setminus Q^m$ **do**
        if $N(q) \cap En(Q', G_i^{fo}) = \varnothing$ or $q \in F$ **then**
          $F_i \leftarrow F_i \cup \{Q'\}$
        end if
      end for
    end for
    $G_{i+1}^{fo} \leftarrow \texttt{SCS}^{\texttt{fo}}(G_i^{fo}, F_i)$
    $i \leftarrow i + 1$
  until $G_i^{fo} = G_{i-1}^{fo}$
  return G_i^{fo}
end function

q2 has only one nonblocking providing event, namely b, and that state q5 has no nonblocking providing events. Because for state q5 in aggregate state q4q5 no nonblocking providing events are available, state q4q5 is removed in iteration 1. As a consequence, the only nonblocking proving event for state q2 in aggregate state q2q3 is also removed. Therefore, in iteration 2, the aggregate state q2q3 is removed and only the initial state remains.

Proposition 11. (Termination). Algorithm 2 terminates.

The following theorem states that the supervisor that results from Algorithm 2 is indeed a maximally permissive proper supervisor for the plant under partial observation.

Theorem 12. (Correctness). Let a plant G with set of observable events $\Sigma_o \subseteq \Sigma$ and forbidden states $F \subseteq Q$ be given. $\texttt{SCS}^{\texttt{po}}(G)$ is a maximally permissive proper supervisor for G under partial observation.

Theorem 13. The worst case time complexity of $\texttt{SCS}^{\texttt{po}}$ of Algorithm 2 is $\mathcal{O}(|\Sigma| \cdot 2^{3|Q|})$.

The time complexity of the algorithm proposed by Yin and Lafortune (2016) is $\mathcal{O}((|Q|^3 \cdot 2^{|Q|} + |\Sigma|) \cdot |Q| \cdot 2^{2|Q|+|\Sigma|})$. Note that this algorithm handles the more general case of arbitrary unobservable events. Although the precise complexities of theirs and ours differ, both suffer from exponential blow-up. This is probably unavoidable, as Tsitsiklis (1989) has shown that no polynomial algorithm exists for synthesis of a partial observation supervisor.

Next, we compare the complexities of these two algorithms in two different situations:

(1) Assuming that Σ is either larger or approximately of the same size as Q, the complexity of the algorithm of Yin and Lafortune (2016) can be considered to be $\mathcal{O}((|Q|^3 \cdot 2^{|Q|} + |\Sigma|) \cdot |Q| \cdot 2^{2|Q|+|\Sigma|}) \approx \mathcal{O}((|Q|^3 \cdot 2^{|Q|} + |Q|) \cdot |Q| \cdot 2^{3|Q|}) = \mathcal{O}(|Q|^4 \cdot 2^{4|Q|})$.
(2) Assuming that the size of Σ is small compared to the size of Q, we obtain $\mathcal{O}((|Q|^3 \cdot 2^{|Q|} + |\Sigma|) \cdot |Q| \cdot 2^{2|Q|+|\Sigma|}) = \mathcal{O}(|Q|^3 \cdot 2^{|Q|} \cdot |Q| \cdot 2^{2|Q|}) = \mathcal{O}(|Q|^4 \cdot 2^{3|Q|})$ for the algorithm from Yin and Lafortune (2016).

Observe that in both cases our worst case time complexity of $\mathcal{O}(|\Sigma| \cdot 2^{3|Q|})$ is better.

The all inclusive controller used in the algorithm of Yin and Lafortune (2016) has a state space size of $\mathcal{O}(2^{|Q|} \cdot (2^{|\Sigma_c|} + 1))$. The projection automaton used in our approach has a state space size $\mathcal{O}(2^{|Q|})$. Additionally, note that in the algorithm of Yin and Lafortune (2016) an additional expansion may take place to solve livelocks. In our approach, this is not needed.

5. CONCLUSION

In this paper, an algorithm has been proposed for synthesizing a maximally permissive proper supervisor for a given partially observed plant where all controllable events are assumed to be observable. The approach relies upon the construction of an observer automaton equipped with extra information about the nonblocking providing events for each state. Using this information iteratively using full observation supervisory control synthesis (on the observed plant), we obtain a desired supervisor. Our approach avoids the use of complicated constructions as in known algorithms (in our opinion), and shows a better worst-case time complexity.

In future work, we have the intention to extend this work towards unobservable controllable events, to implement it in the tool ESCET, and to apply it to benchmark cases in literature in order to also compare approaches on (realistic) case studies.

REFERENCES

Cai, K., Zhang, R., and Wonham, W.M. (2015). Relative observability of discrete-event systems and its supremal sublanguages. *IEEE TAC*, 60(3), 659–670. doi:10.1109/TAC.2014.2341891.

Cassandras, C.G. and Lafortune, S. (2009). *Introduction to discrete event systems*. Springer Science & Business Media.

Cho, H. and Marcus, S.I. (1989). On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation. *Mathematics of Control, Signals and Systems*, 2(1), 47–69. doi:10.1007/BF02551361.

Cieslak, R., Desclaux, C., Fawaz, A., and Varaiya, P. (1988). Supervisory control of discrete-event processes with partial observations. *IEEE TAC*, 33(3), 249–260. doi:10.1109/9.402.

Fokkink, W.J., Goorden, M.A., Hendriks, D., van Beek, D.A., Hofkamp, A.T., Reijnen, F.F.H., Etman, L.F.P., Moormann, L., van de Mortel-Fronczak, J.M., Reniers, M.A., Rooda, J.E., van der Sanden, L.J., Schiffelers, R.R.H., Thuijsman, S.B., Verbakel, J.J., and Vogel, J.A. (2023). Eclipse ESCET™: The Eclipse Supervisory Control Engineering Toolkit. In *TACAS*, 44–52. Springer Nature Switzerland. doi:10.1007/978-3-031-30820-8_6.

Hu, Y., Ma, Z., and Li, Z. (2023). Design of supervisors for partially observed discrete event systems using quiescent information. *IEEE TASE*. doi:10.1109/TASE.2023.3301997. Early access.

Inan, K. (1994). Nondeterministic supervision under partial observations. In *11th Int. Conf. on Analysis and Optimization of Systems: Discrete Event Systems*, LNCIS, 39–48. Springer. doi:10.1007/BFb0033530.

Komenda, J. (2013). Supervisory control with partial observations. In *Control of Discrete-Event Systems: Automata and Petri Net Perspectives*, LNCIS, 65–84. Springer. doi:10.1007/978-1-4471-4276-8_4.

Lin, F. and Wonham, W.M. (1988). On observability of discrete-event systems. *Information Sciences*, 44(3), 173–198. doi:10.1016/0020-0255(88)90001-1.

Markovski, J., van Beek, D.A., Theunissen, R.J.M., Jacobs, K.G.M., and Rooda, J.E. (2010). A state-based framework for supervisory control synthesis and verification. In *IEEE CDC*, 3481–3486. doi:10.1109/CDC.2010.5717095.

Ouedraogo, L., Kumar, R., Malik, R., and Akesson, K. (2011). Nonblocking and safe control of discrete-event systems modeled as extended finite automata. *IEEE TASE*, 8(3), 560–569. doi:10.1109/TASE.2011.2124457.

Åkesson, K., Fabian, M., Flordal, H., and Malik, R. (2006). Supremica – An integrated environment for verification, synthesis and simulation of discrete event systems. In *WODES*, 384–385. IEEE. doi:10.1109/WODES.2006.382401.

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

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

Su, R., van Schuppen, J.H., and Rooda, J.E. (2008). Supervisor synthesis based on abstractions of nondeterministic automata. In *WODES*, 412–418. IEEE. doi:10.1109/WODES.2008.4605981.

Tai, R., Lin, L., Zhu, Y., and Su, R. (2023). Synthesis of the supremal covert attacker against unknown supervisors by using observations. *IEEE TAC*, 68(6), 3453–3468. doi:10.1109/TAC.2022.3191393.

Takai, S. and Ushio, T. (2003). Effective computation of an Lm(G)-closed, controllable, and observable sublanguage arising in supervisory control. *Systems & Control Letters*, 49(3), 191–200. doi:10.1016/S0167-6911(02)00322-5.

Tsitsiklis, J.N. (1989). On the control of discrete-event dynamical systems. *Mathematics of Control, Signals and Systems*, 2(2), 95–107. doi:10.1007/BF02551817.

van Beek, D.A., Fokkink, W., Hendriks, D., Hofkamp, A., Markovski, J., Van De Mortel-Fronczak, J., and Reniers, M.A. (2014). CIF 3: Model-based engineering of supervisory controllers. In *TACAS*, 575–580. Springer. doi:10.1007/978-3-642-54862-8_48.

Ware, S. and Malik, R. (2008). The use of language projection for compositional verification of discrete event systems. In *WODES*, 322–327. IEEE. doi:10.1109/WODES.2008.4605966.

Wonham, W.M. and Cai, K. (2019). *Supervisory Control of Discrete-Event Systems*. Communications and Control Engineering. Springer International Publishing. doi:10.1007/978-3-319-77452-7.

Yin, X. and Lafortune, S. (2016). Synthesis of maximally permissive supervisors for partially-observed discrete-event systems. *IEEE TAC*, 61(5), 1239–1254. doi:10.1109/TAC.2015.2460391.

Yoo, T.S. and Lafortune, S. (2006). Solvability of centralized supervisory control under partial observation. *DEDS*, 16(4), 527–553. doi:10.1007/s10626-006-0023-7.