Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

A General Approach for Synthesis of Supervisors for Partially-Observed Discrete-Event Systems [⋆]

Xiang Yin [*] Stéphane Lafortune [*]

[*] *Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA. (e-mail:{xiangyin,stephane}@umich.edu)*

Abstract: We revisit the synthesis of supervisors for partially-observed discrete-event systems from a new angle, based on the construction of a new structure called the All Inclusive Controller (or AIC). We consider control problems for safety specifications, where the legal language is a prefix-closed sublanguage of the system language. We define the AIC as a bipartite transition system that embeds all safe supervisors and thus all controllable and observable sublanguages of the legal language. The structure of the AIC is that of a bipartite graph, reminiscent of a game between the supervisor and the system, with (i) control states, where all safe control decisions are enumerated, and (ii) system states, where all feasible observable system events are executed. The states of the AIC are information states, i.e., subsets of system states. We present an algorithm for the construction of the AIC. This algorithm exploits the pre-computation of the so-called extended specification, which makes the safety of a control state a function of the current information state alone, thereby allowing for on-the-fly construction of the AIC, if so desired. We discuss the properties of the AIC. We also describe how the AIC can be used for synthesis of supervisors that possess a desired maximality property.

Keywords: Discrete Event Systems; Supervisory Control and Automata; Partial Observation.

1. INTRODUCTION

This paper is concerned with the control of partially-observed Discrete Event Systems (DES). Control problems for DES arise in the study of complex automated systems where the behavior is inherently event-driven, as well as in the study of discrete abstractions of continuous, hybrid, and/or cyber-physical systems. The goal in this paper is to restrict the behavior of a DES within a prefix-closed legal language, while accounting for the presence of uncontrollable and unobservable events, due to the limited actuation and sensing capabilities in the plant. Unlike the fully-observed case, synthesizing a supervisor that achieves some specification in the partially-observed case is more difficult to deal with, since no supremal solution exists in general. Thus, more general synthesis methodologies for supervisors under safety specifications are needed.

A number of other works have considered this problem. The control of centralized partially-observed DES was initially studied by Lin and Wonham [1988], Cieslak et al. [1988]. The problem of synthesizing a safe and nonblocking partial observation supervisor was shown to be decidable [Inan, 1994] and solvable [Yoo and Lafortune, 2006]. To deal with computational complexity of off-line synthesis, online schemes for the computation of the supervisor were proposed in Heymann and Lin [1994] and Ben Hadj-Alouane et al. [1996]. In [Ben Hadj-Alouane et al., 1996], the authors gave an online algorithm, which computes a particular type of maximal solution for a prefix-closed language specification. In [Yoo and Lafortune, 2006], where the prefix-closed assumption was relaxed, the authors provided an algorithm that returns a nonblocking solution to this problem; however, the maximal property cannot be guaranteed. This problem was also studied from different angles by Overkamp and van Schuppen [2000] and Arnold et al. [2003]. An alternative approach to the partial-observation control problem was studied by allowing the supervisor to be nondeterministic. Control using nondeterministic supervisors was first advocated by Inan [1994] and extended by Kumar et al. [2005], in which a weaker notion of observability was studied.

Recently, a new structure called "most permissive observer" was studied in dynamic sensor activation problems in DES by Cassez and Tripakis [2008] and Dallal and Lafortune [To appear]. This structure captures all controllers (sensor activation policies) maintaining K-diagnosability. In this paper, motivated by this previous work, we define a new structure called *All Inclusive Controller* (AIC). However, what we aim to solve here is the standard supervisory control problem under partial observation. In this new structure, we enumerate all "safe" control decisions that are possible at each information state. This structure contains all possible control policies that generate controllable and observable sublanguages of the specification; this justifies naming this structure the "all inclusive controller". The AIC can be used for finding one "optimal" solution with respect to some cost criterion.

[⋆] This work was partially supported by the NSF Expeditions in Computing project ExCAPE: Expeditions in Computer Augmented Program Engineering (grant CCF-1138860).

Compared with previous approaches, the main contribution of this paper is that we present a general approach for synthesizing a supervisor by defining a new *deterministic* structure and provide an algorithm for constructing this structure. The structure of the AIC has the following properties: (i) The AIC is defined as a deterministic bipartite transition system. This structure provides a more readily interpretable solution to this problem, since the alternation of states in the AIC captures the alternation between control decisions and observations; (ii) The construction of the AIC can be achieved on-the-fly, if so desired, since we prove that safety of a control decision is fully determined by the current information state, rather than by looking forward or iterating; and (iii) The structure of the AIC contains *all* solutions to this problem. Thus, it can serve as a basis for finding one maximal and/or one "optimal" solution w.r.t. some cost criterion.

This paper is organized as follows. Section 2 describes the definitions related to the information state and defines the total controller. In Section 3, we define the recursive structure of the AIC for the safety control problem. In Section 4, we define the extended specification, establish a number of results on safety, and provide an algorithm to construct the AIC by using the extended specification. Section 5 establishes the correctness of the AIC and shows how to synthesize a supervisor from it. We illustrate the application of the AIC to a collision avoidance problem in Section 6. Finally, we conclude in Section 7. Due to space constraints, all proofs have been omitted.

2. SYSTEM MODEL, INFORMATION STATE AND TOTAL CONTROLLER

2.1 System Model

We assume basic knowledge of DES and common notations (see, e.g., [Cassandras and Lafortune, 2008]). We model a DES as a deterministic finite-state automaton $G = (X, E, f, x_0, X_m)$, where X is the finite set of states, E is the finite set of events, $f : X \times E \to X$ is the partial transition function where $f(x, e) = y$ means that there is a transition labeled by event e from state x to state y, x_0 is the initial state, and X_m is the set of marked states. f is extended to $X \times E^*$ in the usual way. The behavior of the system is described by the prefix-closed language $\mathcal{L}(G)$, generated by G. In this paper, we only deal with prefix-closed languages (i.e., blocking is not considered). Therefore, we assume that $X_m = X$.

In the supervisory control framework [Ramadge and Wonham, 1987], a sublanguage $K \subseteq \mathcal{L}(G)$ represents the desired or legal/safe behavior. A supervisor is imposed on G with the task of dynamically enabling/disabling events in order to achieve such a specification. The event set E is partitioned into two disjoint subsets: E_c, the subset of controllable events, and E_{uc}, the subset of uncontrollable events. Under the partial observation assumption [Lin and Wonham, 1988], E is also partitioned into the subset of observable events, E_o, and the subset of unobservable events, E_{uo}. The natural projection, $P : E^* \to E_o^*$, is defined in the usual manner (see, e.g., [Cassandras and Lafortune, 2008]). A partial observation supervisor is a function $S_P : E_o^* \to 2^E$, with the following constraints: $E_{uc} \subseteq S_P(s), \forall s \in E_o^*$. We say that a control decision is admissible if it satisfies the above constraint and define $\Gamma = \{\gamma \in 2^E : E_{uc} \subseteq \gamma\}$ as the set of admissible control decisions. The notation S_P/G is used to represent the controlled system and the language generated by S_P/G, denoted by $\mathcal{L}(S_P/G)$, is defined recursively in the usual manner.

For the sake of simplicity, we use the algorithm in the appendix of [Ben Hadj-Alouane et al., 1996] to pre-process the trim automaton that generates K, denoted by H where $\mathcal{L}(H) = K$, and the original system model G, if necessary, such that the resulting H and G satisfy the following properties: (i) H is a sub-automaton of G, denoted by $H \sqsubseteq G$ and its state space is denoted by X_H; and (ii) if $f(x, e) = y$ is defined in G and $x, y \in X_H$, then $f_H(x, e) = y$ is also defined in H. This means that the safety specification K on $\mathcal{L}(G)$ is fully captured by a partition of the state space X of G into legal states, i.e., those in X_H, and illegal states, i.e, those in $X \setminus X_H$. That is, any transition between legal states is also legal.

Before moving forward, we define the following operators that will be used in this paper. The *unobservable reach* of the subset of states $S \subseteq X$ under the subset of events $\gamma \subseteq E$ is given by, $\mathrm{UR}_\gamma(S) := \{x \in X : (\exists u \in S)(\exists e \in (E_{uo} \cap \gamma)^*) \text{ s.t. } x = f(u, e)\}$. The *extended unobservable reach* of the subset of states $S \subseteq X$ under the subset of events $\gamma \subseteq E$ is given by, $\mathrm{UR}^+_\gamma(S) := \mathrm{UR}_\gamma(S) \cup \{x \in X : (\exists u \in \mathrm{UR}_\gamma(S))(\exists e \in (E_o \cap \gamma)) \text{ s.t. } x = f(u, e)\}$. The *observation transition* of the subset of states $S \subseteq X$ under observable event $e \in E_o$ is given by, $\mathrm{Next}_e(S) := \{x \in X : \exists u \in S \text{ s.t. } x = f(u, e)\}$.

2.2 Information State and Total Controller

An information state (IS) is a subset $IS \subseteq X$ of states. We denote by $I = 2^X$ the set of information states. We will use two kinds of states, Y-states, $y \in Y$, and Z-states, $z \in Z$. Specifically, Y is the set of information states (i.e., $Y = I$), and Z is the set of information states augmented with enabled events (i.e., $Z = I \times 2^E$). We use the notation $I(z)$ and $C(z)$ denote z's information state and control decision components, respectively, so that $z = (I(z), C(z))$. For example, consider the automaton in Figure 1; a Y-state y is a state in the form of $\{1, 2\}$ and a Z-state z is a state in the form of $(\{1\}, \{a, c, uc\})$, where $I(z) = \{1\}$ and $C(z) = \{a, c, uc\}$. Also, we define the labeled transition relation $A \subseteq Y \times \Gamma \times Z \cup Z \times E_o \times Y$ between Y and Z-states as follows:

- A transition from a Y-state to a Z-state represents the unobservable reach and "remembers" the set of enabled events from the Y-state that leads to it. Formally, $(y, \gamma, z) \in A$ is a transition from y to z, labeled with $\gamma \in \Gamma$, if:
$$I(z) = \mathrm{UR}_\gamma(y) \quad \text{and} \quad C(z) = \gamma \tag{1}$$
This means that $I(z)$ is the set of states reachable from some state in y through some string of enabled unobservable events, and that $C(z)$ is the control decision made in y. We write $h_{YZ}(y, \gamma) = z$.
- A transition from a Z-state to Y-state represents the observation transition. Formally, $(z, e, y) \in A$ is a transition from z to y, labeled with $e \in C(z) \cap E_o$, if:

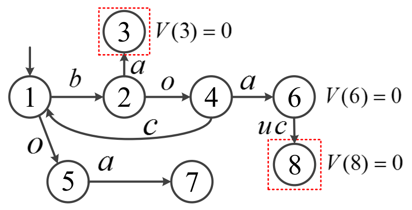

Fig. 1. A finite state automaton G. Events are classified as follows: $E_o = \{o, uc\}$, $E_{uo} = \{a, b, c\}$, $E_c = \{a, b, c, o\}$ and $E_{uc} = \{uc\}$. States 3 and 8 are illegal states.

$$y = \text{Next}_e(I(z)) \tag{2}$$

This means that y is the set of states reachable from some state in $I(z)$ through an enabled observable event e. We write $h_{ZY}(z, e) = y$.

Example 1. Consider the automaton of Figure 1. For the initial Y-state $y_0 = \{1\}$, given control decision $\gamma_0 = \{a, b, o\}$, by definition, the successor Z-state is $z_0 = h_{YZ}(y_0, \gamma_0) = (\text{UR}_{\gamma_0}(y_0), \gamma_0) = (\{1, 2, 3\}, \gamma_0)$. Here, there is only one observable event o in $C(z_0)$. Once event o occurs, the next Y-state is given by $y_1 = h_{ZY}(z_0, o) = \text{Next}_o(I(z_0)) = \{4, 5\}$, since $f(1, o) = 5, f(2, o) = 4$ and $1, 2 \in I(z_0)$. However, the control decision γ_0 is not a safe control decision at the initial state, because illegal state 3 could be reached before a new event is observed.

Definition 2.1. (Total Controller). The Total Controller (TC) is defined as a bipartite transition system $\mathcal{TC}(G) = (Y, Z, A, \Gamma, E, y_0)$, Here, Y and Z are the Y and Z-state sets, respectively and, A is the set of labeled transitions defined above where Γ and E are the respective label sets. The set A contains (i) all transitions from Y-states to Z-states (all admissible control decisions) and (ii) all transitions from Z-states to Y-states (all observable events). The initial state is the Y-state corresponding to the initial information state, i.e., $y_0 = \{x_0\}$. □

Since the control decision for a Y-state may not be unique, we use the notation $C(y)$ to denote the set of control decisions from a Y-state y. Note that, $C(y), y \in Y$, is a set of event sets (set of control decisions) and $C(z), z \in Z$, is a single event set (one control decision).

Remark 1. Because the total controller contains all admissible control decisions after each observation and all possible event observations after each control decision, it contains all strings in $\mathcal{L}(G)$ and also every admissible supervisor. As a consequence, this structure contains all possible supervisors and languages generated by some controlled system, no matter safe or unsafe. Also, we observe that the a run of the bipartite structure (alternation between control and observation) also results in an alternation between Y and Z-states.

Remark 2. There is a special kind of Z-state in TC, which has no Y-state successors. Formally, we say that a Z-state z is *terminal* if $(\forall x \in I(z))(\forall e \in (E_o \cap C(z)))[f(x, e)$ is not defined]. By definition, we know that the total controller only ends up with a *terminal* Z-state.

Definition 2.2. (Y-state Supervisor Induced Information State Evolution). Given a supervisor S_P, we define $IS^Y_{S_P}(y, s)$ to be the Y-state that results from the occurrence of string s, when starting in Y-state y. This can be computed recursively as follows:

$$IS^Y_{S_P}(y, \epsilon) := y$$

$$IS^Y_{S_P}(y, s\sigma) := \begin{cases} h_{ZY}(h_{YZ}(IS^Y_{S_P}(y, s), S_P(P(s))), \sigma), \\ \qquad \text{if } \sigma \in E_o \cap S_P(P(s)) \\ IS^Y_{S_P}(y, s), \\ \qquad \text{if } \sigma \in E_{uo} \cap S_P(P(s)) \\ \text{undefined}, \quad \text{otherwise} \end{cases} \tag{3}$$

For brevity, we define $IS^Y_{S_P}(y_0, s) := IS^Y_{S_P}(s)$.

Also, $IS^Z_{S_P}(z, s)$ is defined analogously, with $IS^Z_{S_P}(s) := IS^Z_{S_P}(z_0, s)$, where $z_0 = h_{YZ}(y_0, S_P(\epsilon))$. □

Lemma 1. Given a supervisor S_P, for any string $s \in \mathcal{L}(S_P/G)$, we have $I(IS^Z_{S_P}(s)) = \{v \in X : \exists s' \in \mathcal{L}(S_P/G) \text{ s.t. } P(s) = P(s') \wedge v = f(s')\}$.

This lemma implies that the information state reached by supervisor S_P upon the occurrence of string s is only determined by its projection. Thus, strings that have the same projection should lead to the same information state.

3. ALL INCLUSIVE CONTROLLER FOR SAFETY

In this section, we define the safety control problem we need to solve and define the recursive structure of the AIC, that contains all solutions to this problem.

Definition 3.1. We say an information state $i \in I$ violates safety if there exist a state $x \in i$ such that $x \notin X_H$. Thus, we define the safety binary function $D_I : I \to \{0, 1\}$ as:

$$D_I(i) = \begin{cases} 1, & \text{if } \forall x \in i : x \in X_H \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

□

Theorem 2. Supervisor S_P maintains safety if and only if $D_I(I(z)) = 1$, for all reachable Z-states. Mathematically:

$$\exists s \in \mathcal{L}(S_P/G) : z = IS^Z_{S_P}(s) \wedge D_I(I(z)) = 0$$
$$\Leftrightarrow \exists s' \in \mathcal{L}(S_P/G) : f(s') \notin X_H$$

This theorem allows us to transfer the partially-observed safety control problem to the problem of finding a subgraph of the total controller, in which all reachable states are safe. To formally describe this transformation, let us first define the notion of safety for Y and Z-states.

Definition 3.2. (Safety binary function for Y and Z-state). We say that a Y-state is safe if it is currently safe and there *exists* some supervisor that maintains the safety property for any future behaviors. Since we cannot choose event occurrences, we say a Z-state is safe if it is currently safe and *all* of its successor Y-states are safe. Thus, we define two safety binary functions, $D_Y : Y \to \{0, 1\}$ and $D_Z : Z \to \{0, 1\}$ as follows:

$$D_Y(y) = \begin{cases} 1, & \begin{array}{l}\text{if } D_I(y) = 1 \text{ and} \\ \exists \gamma \in \Gamma : D_Z(h_{YZ}(y, \gamma)) = 1\end{array} \\ 0, & \text{else} \end{cases} \tag{5}$$

$$D_Z(z) = \begin{cases} 1, & \begin{array}{l}\text{if } D_I(I(z)) = 1 \text{ and} \\ \forall e \in (C(z) \cap E_o) : D_Y(h_{ZY}(z, e)) = 1\end{array} \\ 0, & \text{else} \end{cases} \tag{6}$$

□

From these definitions, we see that there exists a safe supervisor for G if and only if $D_Y(y_0) = 1$, i.e., $D_I(y_0) = 1$

and $\exists \gamma_0 : D_I(I(z_0)) = 1$ and $\forall e_0 \in \gamma_0 \cap E_o : D_I(y_1) = 1$, and so forth, where $z_0 = h_{YZ}(y_0, \gamma_0)$ and $y_1 = h_{ZY}(z_0, e_0)$. By using information state evolution, we can say that G can be safely controlled if and only if $\exists S_P$ such that $\forall s \in \mathcal{L}(S_P/G)$, $D_I(IS^Y_{S_P}(s)) = 1$ and $D_I(I(IS^Z_{S_P}(s))) = 1$. This is the same conclusion that is reached from Thm. 2. Using the above definition, we say that a control decision γ is safe from Y-state y if $D_Z(h_{YZ}(y, \gamma)) = 1$, since we know that there exists a sequence of safe control decisions in the future. Now, we are ready to define the structure of the All Inclusive Controller that contains all solutions to safety control problem.

Definition 3.3. (All Inclusive Controller). The All Inclusive Controller (AIC) is defined as the largest safe subgraph of the total controller that is reachable from state y_0. By the safe subgraph, we mean the subsystem of the total controller consisting only of safe Y and Z-states, and the transitions between them. The notation $\mathcal{AIC}(G)$ is used to denote the AIC of automaton G. □

Remark 3. It is equivalent to define the AIC as the *trim* of the total controller when removing Y-states such that $D_Y(\cdot) = 0$ and Z-states such that $D_Z(\cdot) = 0$. By Def. 3.2 $D_Y(\cdot)$ and $D_Z(\cdot)$ are defined recursively, therefore the AIC structure (only) captures all safe control decisions. Therefore, any supervisor included in the AIC (will be well defined by Def. 5.1) generates a legal language. It is not difficult to see that the terminal states of the AIC also consists of terminal Z-states.

Example 2. Let G be the automaton shown on Figure 1. By our definition, the resulting AIC of G is shown in Figure 2 (a formal construction algorithm will be given later). In the diagram of the AIC, rectangular states correspond to Y-states and oval states correspond to Z-states. For initial Y-state $\{1\}$, we cannot make control decision $\{a, b\}$, because it will unobservably lead to illegal state 3 before a new event is observed. For Y-state $\{4, 5\}$, it is necessary to disable event a, for otherwise it will lead us to state, which will uncontrollably lead to an illegal state.

Remark 4. Note that, in Figure 2, at the initial Y-state $y_0 = \{1\}$, we can also take control decision $\{o, c\}$. However, event c will never be executed within the resulting unobservable reach, thus we regard event c as a "redundant" event, and eliminate it from the control decisions. This situation also appears in other states. Formally, we say that a control decision $\gamma \in \Gamma$ is *irredundant* at $i \in I$ if, for all $e \in \gamma$, there exists $x \in \mathrm{UR}_\gamma(i)$ such that $f(x, e)$ is defined. To make our structure as compact as possible, we only show the irredundant control decisions in Figure 2.

4. THE EXTENDED SPECIFICATION AND CONSTRUCTION OF THE AIC

We first present the extended specification, which can be used to determine the safety of a Y or Z-state. Then we provide a construction algorithm for the AIC.

4.1 Extended Specification

Definition 4.1. (Extended Safety Specification). The extended specification is defined as the set of states that should never be reached because even if all events in E_c are disabled forever thereafter, there still exists some sequence

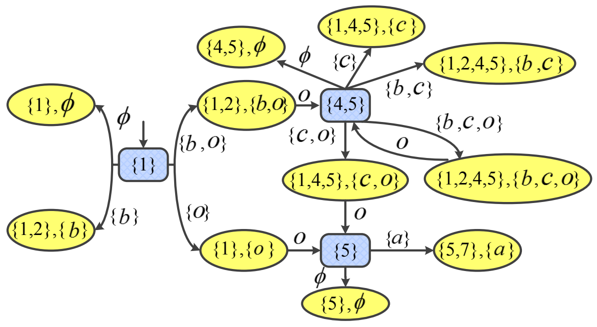

Fig. 2. The corresponding AIC for Figure 1.

of events such that some illegal state will be reached. To describe the extended specification, we define a binary function $V : X \to \{0, 1\}$ [1], where in the case of prefix-closed languages, 0 is assigned to a state if and only if it can uncontrollably reach an illegal state. Mathematically, $\exists s \in (E_{uc})^* : f(x, s) \notin X_H \Leftrightarrow V(x) = 0$ □

For example, in Figure 1, we have $V(3) = V(8) = V(6) = 0$, since 3 and 8 are illegal states and 6 can uncontrollably lead to illegal state 8. Note that the extend specification can be calculated offline all at once or online if so desired (see e.g. [Ben Hadj-Alouane et al., 1994]). In the remainder of this paper, we assume that the extended specification has already been pre-calculated.

Definition 4.2. Similar to the safety binary function, we define the extended safety binary function for information states $D^e_I : I \to \{0, 1\}$ as:

$$D^e_I(i) = \begin{cases} 1, & \forall x \in i : V(x) = 1 \\ 0, & \text{else} \end{cases} \tag{7}$$

We are now ready to prove two key theorems show that how we can determine the values of $D_Y(\cdot)$ and $D_Z(\cdot)$ by using the extended specification.

Theorem 3. For any Y-state y, we have $D_Y(y) = D^e_I(y)$.

Theorem 4. For any Z-state z, we have
$D_Z(z) = D^e_I(\mathrm{UR}^+_{C(z)}(I(z)))$.

Remark 5. Theorems 3 and 4 imply that the safety of a Y or Z-state solely depends on itself. We note that the safety of a Z-state is not solely information state based, but also depends on its associated control decision, since adding a new observable event in the control list may lead to some illegal state, in which case even if its information state is safe, the extended unobservable reach can still be unsafe. That is why we also need to check the safety of its extended unobservable reach, i.e., all reachable states it could be in before the next control decision is issued.

Corollary 5. (Monotonicity Properties).
(i) If Y-state y_1 is safe then so is any Y-state $y_2 \subseteq y_1$.
(ii) Any control decision that is safe in Y-state y_1 is also safe in Y-state $y_2 \subseteq y_1$.
(iii) If control decision γ_1 is safe in Y-state y, then so is any control decision $\gamma_2 \subseteq \gamma_1$.

4.2 Construction of the AIC

Definition 3.2 provides us an obvious means for constructing the AIC. Since the safety of a Z-state is defined

[1] This binary function was expressed as a cost function $V : X \to \{0, \infty\}$ in [Ben Hadj-Alouane et al., 1994].

Algorithm 1

```
procedure DoDFS($G, y, V, sl, E$)
    for all $el \subseteq E.c$ do
        $Act \leftarrow el \cup E.uc$
        $eur \leftarrow \mathrm{UR}^{+}_{Act}(y)$
        if $[(\forall x \in eur) V(x) = 1]$ then
            Add $(y, Act)$ to $sl$
            Mark $y$ as "safe"
        end if
    end for
    if $y$ is not marked "safe" then
        Mark $y$ as "unsafe"
    end if
    for all $el \subseteq E.c$ s.t $(y, el \cup E.uc) \in sl$ do
        $Act \leftarrow el \cup E.uc$
        $ur \leftarrow \mathrm{UR}_{Act}(y)$
        for all $e \in Act \cap E.o$ do
            $next \leftarrow \mathrm{Next}_e(ur)$
            if $next$ not marked then
                DoDFS($G, next, V, sl, E$)
            end if
        end for
    end for
end procedure
```

universally, to determine whether it is safe or not, one way is searching through the whole (Y and Z) state space until all remaining reachable states are either known to be unsafe, or have already been visited. This approach does work, but is worse in computational complexity. Recalling the previous results, Thm. 3 and Thm. 4 allow us to use the extended specification to simplify this problem by replacing this whole search by a *single* calculation each time we would like to determine the safety of a particular information state. At any Y-state, we can determine whether or not taking control decision γ is safe through three steps: first, we compute $z = h_{YZ}(y, \gamma)$; second, we compute the extend unobservable reach of $I(z)$ under γ, $eur = \mathrm{UR}^{+}_{\gamma}(I(z))$ and, third, we verify whether or not the information state eur satisfies the extended specification. Since $\mathrm{UR}^{+}_{\gamma}(y) = \mathrm{UR}^{+}_{\gamma}(I(z))$ for $z = h_{YZ}(y, \gamma)$, steps 1 and 2 can be collected as one step.

The construction algorithm for the AIC is shown in Algorithm DoDFS, which is based on a depth-first search. The parameters in this algorithm are as follows: G represents the system for which we want to construct the AIC, y represents a Y-state, V represents the extended specification which has been pre-computed, and E contains the sets of events E_c, E_{uc}, E_o and E_{uo}. Lines 2-12 are used to find the safe control decisions. This is done by considering each subset of events $el \subseteq E.c$, and determining whether it is safe or not to choose to enable events $Act = el \cup E.uc$. For each control decision, we compute its extended unobservable reach, and determine the value of $D^{e}_{I}(\mathrm{UR}^{+}_{Act}(y))$. If the control decision is safe, then we add y to the state list sl and we mark y as safe. Lines 13-22 are used to traverse the space of Y-states. This is done by considering all safe control decisions of the current Y-state, determining the next Z-states and, for each such Z-state, computing all possible Y-state successors and making a recursive call. Since the number of information states and the number of events are both finite, the algorithm will eventually terminate. Note that the final output of this Algorithm is sl and the initial call is DoDSF$(G, y_0, V, \emptyset, E)$.

Theorem 6. DoDFS correctly constructs the AIC.

Proposition 7. The running time of DoDFS is in $O([2^{(|X|+|E_c|)}]|X||E|)$. The number of states in the AIC, in the worst case, is $[2^{|X|} + 2^{(|X|+|E_c|)}]$.

5. PROPERTIES OF THE AIC

In this section, we first prove the correctness of the AIC, i.e., that this structure contains all possible control policies that generate the controllable and observable sublanguages of the specification. Then we discuss how to synthesize a supervisor from the AIC and study a special kind of supervisor which is solely information state based.

5.1 Correctness of the AIC

We start by revisiting the well-known properties of controllability and observability. We say that a prefix-closed sublanguage K is controllable if, $(\forall s \in K, \sigma \in E_{uc})(s\sigma \in \mathcal{L}(G) \Rightarrow s\sigma \in K)$; and K is observable if, $(\forall s, s' \in K, \sigma \in E_c)(P(s) = P(s') \wedge s\sigma \in K \wedge s'\sigma \in \mathcal{L}(G) \Rightarrow s'\sigma \in K)$. Then, we first define the notion of the AIC included supervisor and then use this concept to define the notion of the AIC included language.

Definition 5.1. (AIC Included Supervisor). A supervisor $S_P : E_o^* \to \Gamma$ is said to be included in $\mathcal{AIC}(G)$ if,

$$(\forall s \in \mathcal{L}(S_P/G))[S_P(P(s)) \in C(IS^{Y}_{S_P}(s))]$$

We use the notation $\mathcal{S}(\mathcal{AIC}(G))$ to denote the set of supervisors included in $\mathcal{AIC}(G)$. □

Definition 5.2. (AIC Generated Language). A language L is said to be generated by $\mathcal{AIC}(G)$ if,

$$(\exists S_P \in \mathcal{S}(\mathcal{AIC}(G)))[\mathcal{L}(S_P/G) = L]$$

We use the notation $\mathcal{L}_{AIC}(\mathcal{AIC}(G))$ to denote the set of languages generated by $\mathcal{AIC}(G)$. □

Remark 6. By the Controllability and Observability Theorem (see, e.g., [Cassandras and Lafortune, 2008]), for a prefix-closed language L, controllability and observability are necessary and sufficient conditions for the existence of a supervisor S_P such that $\mathcal{L}(S_P/G) = L$. Thus, the statement of synthesizing a controllable and observable sublanguage and the statement of synthesizing a supervisor to restrict the behavior are equivalent. We can define either notion first, and it will lead to the same result.

Theorem 8. There exists a supervisor S_P for system G such that $\mathcal{L}(S_P/G) \subseteq \mathcal{L}(H)$ *iff* $\mathcal{AIC}(G)$ is non-empty.

We are now ready to prove the correctness of the AIC. The next results shows that the AIC structure (only) contains *all* solutions to the partially-observation control problem.

Theorem 9. If $\mathcal{AIC}(G)$ is non-empty, then

$$(L = \overline{L} \subseteq \mathcal{L}(H) \wedge L \text{ is observable} \wedge L \text{ is controllable})$$
$$\Leftrightarrow L \in \mathcal{L}_{AIC}(\mathcal{AIC}(G))$$

5.2 State Based Property of the AIC

Theorem 9 provides us with a clear way for synthesizing a supervisor. By Def. 5.1, since safety is the only issue we consider, then for any string s, the control decision made based on the projection $P(s)$ must be within the control decision set of its corresponding information state $IS^Y_{S_P}(s)$ in the AIC, i.e., for each Y-state y, we can arbitrarily choose a control decision which is defined in the AIC, and following the same step after a new observation, and so forth, until reaching a terminal Z-state.

As discussed above, if we solely consider safety, then randomly picking a control decision at each information state does satisfies the requirement; however, in most of the cases, we would like to do more. There are two issues to consider. First, randomly choosing a control decision can only guarantee safety; however, in many situations, we want to find a *maximal* solution. Second, by Def. 5.1, we know that an AIC included supervisor is string based, not information state based. Thus, to synthesize a supervisor from the AIC, it is possible to take different control decisions upon different visits to the same state. However, for many purposes (e.g., saving the memory of the controller), it is desirable to restrict attention to information-state-based supervisors, rather than remembering all the histories.

Definition 5.3. (IS-Based Supervisor) A partially-observed supervisor $S_P : P(\mathcal{L}(G)) \to 2^E$ is said to be information-state-based (IS-based) if $(\forall s, t \in \mathcal{L}(S_P/G))[IS^Y_{S_P}(s) = IS^Y_{S_P}(t) \Rightarrow S_P(P(s)) = S_P(P(t))]$. □

The above definition means the following. If a supervisor is IS-based, it implies that it will take the same control decision every time it visits the same information state. Thus, an IS-based supervisor can be defined as $S_I : I \to 2^E$. We define $\mathcal{S}_I(\mathcal{AIC}(G)) \subseteq \mathcal{S}(\mathcal{AIC}(G))$ as the set of IS-based supervisors included in the AIC. Clearly, the cardinality of $\mathcal{S}_I(\mathcal{AIC}(G))$ is finite.

It is well known that the property of observability is more difficult to deal with than the property of controllability, because it is not preserved under union in general, i.e., the operation $^{\uparrow O}$ does not always exist. As stated above, we usually wish to find a *maximal* (w.r.t. set inclusion) solution. Unfortunately, in general, not all maximal solutions can be expressed by IS-based supervisors. However, in some special cases, IS-based supervisors provide a simple way for us to find maximal solutions. We present three results in this regard.

Proposition 10. There exists at least one IS-based supervisor $S_I \in \mathcal{S}_I(\mathcal{AIC}(G))$ such that $\mathcal{L}(S_I/G)$ is a maximal controllable and observable sublanguage.

Proposition 11. Assume that $E_c \subseteq E_o$. If $L \subseteq \mathcal{L}(H)$ is a maximal controllable and observable sublanguage, then there exists an IS-based supervisor $S_I \in \mathcal{S}_I(\mathcal{AIC}(G))$ such that $\mathcal{L}(S_I/G) = L$.

Proposition 12. There exists an IS-based supervisor $S_I \in \mathcal{S}_I(\mathcal{AIC}(G))$ such that $\mathcal{L}(H)^{\uparrow(CN)} \subseteq \mathcal{L}(S_I/G)$, where $^{\uparrow(CN)}$ is the supremal controllable and normal sublanguage operation.

The supervisor in Proposition 10 can be constructed by taking a fixed control decision $c \in C(y)$, such that $\forall c' \in C(y) : c \not\subset c'$, for any reachable Y-state y. Proposition 12 could be regarded as a corollary of Theorem 5 in [Ben Hadj-Alouane et al., 1996]. Such a supervisor can be constructed as follows. For each Y-state $y \in Y$, pick a fixed control decision $c \in C(y)$, such that: (i) $\forall c' \in C(y) : c \not\subset c'$ and, (ii) $\forall c'' \in C(y)$ satisfying (i): $P_{uo}(c) \not\subset P_{uo}(c'')$, where $P_{uo}(c)$ means eliminate all observable events in c.

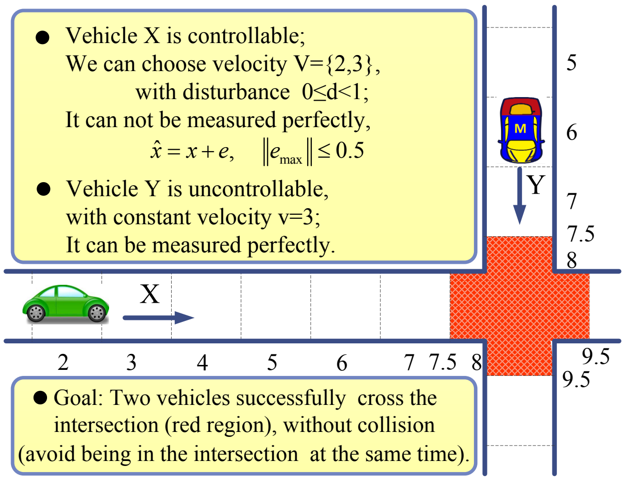
Vehicle X is controllable;
We can choose velocity V={2,3},
with disturbance 0≤d<1;
It can not be measured perfectly,
$\hat{x} = x + e, \quad \|e_{\max}\| \le 0.5$
Vehicle Y is uncontrollable,
with constant velocity v=3;
It can be measured perfectly.
M
Y
X
5
6
7
7.5
8
9.5
2
3
4
5
6
7
7.5
8
9.5
Goal: Two vehicles successfully cross the intersection (red region), without collision (avoid being in the intersection at the same time).

Fig. 3. An vehicle control problem

Example 3. Consider again the example we studied in Figures 1 and 2. We want to synthesize the IS-based supervisor of Prop. 12. For initial Y-state $\{1\}$, we can pick control decision $\{b, o\}$ and take $\{b, c, o\}$ at $\{4, 5\}$. Since the only successor Y-state of the Z-state reached by taking $\{b, c, o\}$ from $\{4, 5\}$ is itself, we finish the synthesis procedure and the closed-loop language is $\overline{\{(boc)^*o\}}$. Here, the supremal and normal language is empty, clearly $\mathcal{L}(H)^{\uparrow(CN)} \subseteq \mathcal{L}(S_I/G)$. Furthermore, we see that $\mathcal{L}(S_I/G)$ is also a maximal language.

6. ILLUSTRATIVE EXAMPLE

This section is devoted to an example of a collision avoidance problem in vehicular networks. The example is to illustrate the application of the proposed AIC structure. For more details (modeling and analysis) about the supervisory control of vehicular networks, the reader is referred to [Dallal et al., 2013]. Here, we will only briefly introduce and use results from that work.

Each vehicle is modeled by $\dot{x} = v + d$, where $x \in X \subset \mathbb{R}$ is the state, $v \in V \subset \mathbb{R}$ is the control input, and $d \in D \subset \mathbb{R}$ is the disturbance input. Moreover, we assume that the measurement is imperfect, i.e., $\hat{x} = x + e$, $||e||_\infty \le e_{max}$, where $\hat{x} \in X \subset \mathbb{R}$ is the measurement value. In our particular problem, as shown in Figure 3, we assume that vehicle X is controllable and can take velocity $v = 2$ or 3 with disturbance $0 \le d < 1$ and measurement uncertainty $e_{max} = 0.5$; vehicle Y is uncontrollable, with constant velocity $v = 3$, however, its position can be measured perfectly, i.e., $e_{max} = 0$. Our goal is to guarantee that all vehicles successfully cross the intersection safely, i.e., by avoiding being in the intersection at the same time. It is shown in [Dallal et al., 2013] that the problem of synthesis a safe supervisor for the continuous system can be transferred into a DES control problem by proper abstraction. Furthermore, we assume that the initial state of the continuous system is $(2, 6)$ and at discrete event level, the prior information for X is $\{1, 2, 3\}$.

The DES model G of this system by abstraction is shown in Figure 4, where: events in layer Λ represent measurements,

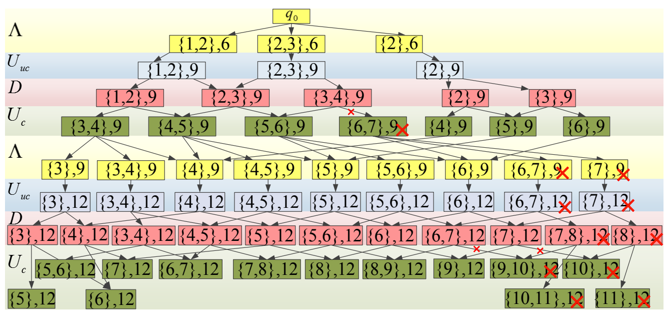

Fig. 4. The DES model G for Figure 3

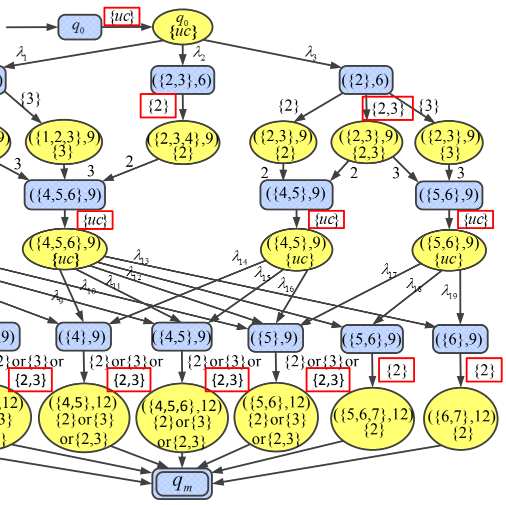

Fig. 5. The All Inclusive Controller for G

which are uncontrollable but observable; events in layer U_{uc} represent the actions of uncontrollable vehicles, which are uncontrollable and unobservable; events in layer D represent the disturbances, which are uncontrollable and unobservable; and events in layer U_c represent the actions of controllable vehicles, which are controllable and observable. By the safety criteria provided in [Dallal et al., 2013], we know that states $(\{9, 10\}, 12)$, $(10, 12)$, $(\{10, 11\}, 12)$ and $(11, 12)$ are illegal, because two vehicles are possibly colliding in the intersection.

By our construction algorithm, the AIC for G is shown in Fig. 5. For simplicity, we use a single state q_m to represent the set of states in which at least one vehicle has already crossed the intersection. Following the properties discussed in the previous sections, the AIC contains all possible safe control policies for this problem. Note that in this model the event set satisfies $E_c \subseteq E_o$. By Prop. 11, we see that the unique (IS-based) maximally permissive solution exists; it consists of the decisions highlighted in Fig. 5. The intuition of the solution is that, in the worst case (largest measurement uncertainty and largest disturbance), vehicle X should keep the slowest velocity, waiting for vehicle Y to cross the intersection first, in order to guarantee no collision will ever happen.

7. CONCLUSION

We revisited the problem of synthesizing a controllable and observable safe sublanguage for a partially observed discrete event system. We defined the All Inclusive Controller, whose structure contains all the solutions to the problem. Also, we defined the extended specification, established several results on safety, and provided an algorithm to effectively construct the AIC by making use of the extended specification. Finally, the correctness of the AIC was proved and how to synthesize a supervisor based on the information state was discussed. In the future, we will investigate: 1) improving the efficiency of our construction algorithms; 2) extending to decentralized structure; 3) dealing with non-prefix-closed languages, i.e., blocking (deadlock and livelock) should be considered; and 4) finding a single optimal controller, w.r.t. some cost criterion.

REFERENCES

A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 303(1):7–34, 2003.

N. Ben Hadj-Alouane, S. Lafortune, and F. Lin. Variable lookahead supervisory control with state information. *IEEE Trans. Autom. Control*, 39(12):2398–2410, 1994.

N. Ben Hadj-Alouane, S. Lafortune, and F. Lin. Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation. *Discrete Event Dyn. Syst.: Theory Appl.*, 6(4):379–427, 1996.

C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, 2nd edition, 2008.

F. Cassez and S. Tripakis. Fault diagnosis with static and dynamic observers. *Fund. Inform.*, 88(4):497–540, 2008.

R. Cieslak, C. Desclaux, A.S. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE Trans. Autom. Control*, 33(3):249–260, 1988.

E. Dallal and S. Lafortune. On most permissive observers in dynamic sensor activation problems. *IEEE Trans. Autom. Control*, To appear.

E. Dallal, A. Colombo, D. Del Vecchio, and S. Lafortune. Supervisory control for collision avoidance in vehicular networks with imperfect measurements. In *Proc. 52th IEEE Conf. Decision Contr.*, 2013.

M. Heymann and F. Lin. On-line control of partially observed discrete event systems. *Discrete Event Dyn. Syst.: Theory Appl.*, 4(3):221–236, 1994.

K. Inan. Nondeterministic supervision under partial observations. In *11th International Conference on Analysis and Optimization of Systems: Discrete Event Systems*, pages 39–48. Springer, 1994.

R. Kumar, S. Jiang, C. Zhou, and W. Qiu. Polynomial synthesis of supervisor for partially observed discrete-event systems by allowing nondeterminism in control. *IEEE Trans. Autom. Control*, 50(4):463–475, 2005.

F. Lin and W.M. Wonham. On observability of discrete-event systems. *Inform. Sciences*, 44(3):173–198, 1988.

A. Overkamp and J.H. van Schuppen. Maximal solutions in decentralized supervisory control. *SIAM J. Control Optim.*, 39(2):492–511, 2000.

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

T.-S. Yoo and S. Lafortune. Solvability of centralized supervisory control under partial observation. *Discrete Event Dyn. Syst.: Theory Appl.*, 16(4):527–553, 2006.