Supervisory Control of Discrete Event Systems with State-dependent Observability

Peng Wang, Xiang-Yun Wang *and* **Kai-Yuan Cai**

School of Automation Science and Electrical Engineering, Beijing University of Aeronautics and Astronautics
Department of Electrical and Computer Engineering, University of Connecticut

This paper considers a new class of discrete event systems under partial observations. The problem is presented within the background of a manufacturing process where workpieces are loaded and transported, and this process is controlled with the partial information collected by sensors. The model extracted is novel because the observation of an event does not only depend on an event itself, but also the state where the system stays. Two standard problems are discussed in this paper: supervisor existence problem and supervisor synthesis problem. With a natural revision of observable languages, a necessary and sufficient condition is given for the existence of a supervisor. For supervisor synthesis problem, two algorithms are developed: one algorithm is to check the properties of a control specification given by a regular language, and the other one is to synthesize a supervisor if the properties hold. Within the background of manufacturing systems, an example is illustrated to show how the algorithms are applied to practical computing.

Keywords: Discrete event systems, supervisory control, state-dependent observability

1. Introduction

A discrete event system (DES) is commonly described by a state transition model such as a automaton, and it is controlled by disabling certain events that cause undesired state transitions [1,2,3]. If some events are not completely observable, control of a system is implemented by using partial information of the state transitions. As a result, an observation problem has been formed since 1980s [2, 3]. Different from linear models, the observation problem of DES is not a dual problem of its controllability study, and it has become an independent branch in the framework of supervisory control theory.

From the perspective of mathematics, observation of events is usually specified by a mapping function which is defined on the event set of DES, and two event sequences look the same if they are the same after mapped to another set of symbols. This function characterizes the ability on how much information the system provides to the outside, especially to the supervisor to be synthesized. In several branch fields related to the observation problem, such as the decentralized control [3,4], optimal control [5,6] and control of concurrent systems [7], the observation function follows the original form in [1,2], where the object and outcome of observation are both events and the outcome exclusively depends on the observed object.

In this paper the original form of an observation function is extended, where the object and outcome of observation are both events, but the outcome is not only dependent on the object, but also the state where the system stays. As a result, the observability of an event is changeable as the system states change form one to another, and it is called state-dependent observability.

To intuitively explain the state-dependent observability we present an example in manufacturing systems. In this example a container is to carry workpieces and a pressure sensor is placed at bottom of the container. The sensor can distinguish whether the container is empty or not, but it cannot differentiate how many workpieces are loaded. If we want to develop a controller to supervise the process of loading/unloading, one question is how much information we can obtain through the sensor. Based on the capability of the sensor, if the container is empty, the action of loading can be observed. If there are workpieces loaded, the action of loading will not be observed by the pressure sensor (See Figure 1). In other words, for the same action of loading, it is observable as the system stays at one state (the container is empty), while become unobservable at other states (the container is loaded). Such observability is thus state-dependent.

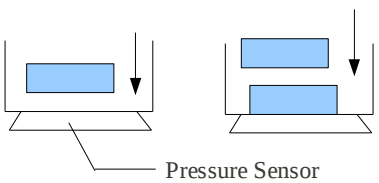

Figure. 1. A container with workpieces

The above loading/unloading workshop is a typical cell in a flexible manufacturing system, and it will be further extended in the later sections to be a computing example.

The rest of this paper is organized as follows. Section 2 reviews the preliminaries and notations of DES and a new model of partially observable system is defined with state-dependent observability. Section 3 studies the supervisor existence problem and derives a necessary and sufficient condition with respect to a given specification. Section 4 focuses on the supervisor synthesis problem and two algorithms are introduced. Section 6 illustrates examples, and concluding remarks are presented in Section 7.

2. Preliminaries and Notation

A. Discrete Event Systems and Supervisory Control

In the original framework by Ramadge and Wonham, a discrete-event system is characterized by a deterministic automaton. The system is event-driven, and the state of a system transits as events happen. The math description is

$$G=(Q,\Sigma,\delta,q_0)$$

where Q is the state set, Σ is the event set, and the state transition function is $\delta: Q\times\Sigma\to Q$. $q_0\in Q$ is the initial state. In general, δ is a partial function, i.e., $\delta(q,\sigma)$ is defined only on a subset of Σ for each $q\in Q$. In this paper we will write $\delta(q,\sigma)!$ if $\delta(q,\sigma)$ is defined, and $\neg\delta(q,\sigma)!$ if it is not.

Let Σ^* denote the set of all finite strings $s=\sigma_1\ldots\sigma_n$ of elements in Σ, including the empty string ε. A subset of Σ^* is a language over Σ. The set of all the prefixes of strings in a language *L* is denoted by $\bar{L}$, and we say *L* is (prefix-)closed if $L=\bar{L}$. The language generated by *G* is

$$L(G)=\{s\in\Sigma^*\mid\delta(q_0,s)!\}.$$

The automaton G is considered as a uncontrolled device that starts from q_0 and executes and generates a sequence of events permitted by δ. Events happen instantaneously and asynchronously, and they can be considered as atomic activities of a device such as "move a workpiece" or "send a message." If $s=\sigma_1\ldots\sigma_n$ belongs to *L*(*G*), it means that the device is capable of carrying out these activities in the order of $\sigma_1\ldots\sigma_n$.

It is sometimes convenient to eliminate states that are not accessible from q_0. The accessible component of *G* is $A(G)=(Q_a,\Sigma,\delta_a,q_0)$. Here Q_a is the set of accessible states, and δ_a is given by

$$Q_a=\{q\mid\exists s\in\Sigma^*, s.t.\ q=\delta(q_0,s)\},$$

$$\delta_a(q,\sigma)=\begin{cases}\delta(q,\sigma) & \text{if } q\in Q_a \text{ and } \delta(q,\sigma)\in Q_a\\ \text{undefined} & \text{otherwise}\end{cases}$$

It is clear that *G* is accessible if *G*=*A*(*G*).

A control mechanism for G is as follows [1,2]. The event set Σ is partitioned into two disjoint part. Let Σ_c be the set of all controllable events, and Σ_{uc} is the set of all uncontrollable events.

$$\Sigma=\Sigma_c\cup\Sigma_{uc} \text{ and } \Sigma_c\cap\Sigma_{uc}=\emptyset.$$

A controllable event can be allowed to occur or prevented from possible occurring while no such control is possible for an uncontrollable event. Thus, a control pattern is γ such that $\Sigma_{uc}\subseteq\gamma\subseteq\Sigma$, and all control patterns are

$$\Gamma=\{\gamma\in 2^{\Sigma}\mid\Sigma_{uc}\subseteq\gamma\subseteq\Sigma\}.$$

An event σ is said to be enabled by γ if $\sigma\in\gamma$, or be disabled by γ if $\sigma\notin\gamma$. A supervisor is defined as a pair $\Phi=(S,\psi)$, where $S=(X,\Sigma,\xi,x_0)$ is a deterministic automaton and $\psi: X\to\Gamma$ is called a state feedback map that selects a control pattern $\psi(x)$ for each state x.

B. State-dependent Observability

As presented in Section 1, in some real situations, Φ cannot observe all transitions of G or cannot distinguish between certain transitions. These situations can be modeled by introducing an observation stage *M* between G and Φ specified by an observation or mask function. The original form of the function was $M:\Sigma\to\Theta\cup\{\varepsilon\}$ [1]. This paper will present a more general definition as below.

Definition 2.1: Let (G,M) denote a partially observable DES where *G* is a deterministic automaton, and *M* is the observation function. In this paper *M* is defined by

$$M: Q\times\Sigma\to\Theta\cup\{\varepsilon\}$$

where Θ represents a set of observation outcomes.

This definition implies that the outcome of observation depends on both the observed object and the state where the system stays. For example, $M(q,\sigma)=\beta$ implies that σ is observed as β at state *q*. Comparing our definition with the old one [1,2], we note that our definition makes an observation process dependent on the system state. In other words, an event can be observable in one state while be unobservable in another. For example, $M(q_1,\sigma)=\sigma$ means σ is observable at state q_1 while $M(q_2,\sigma)=\varepsilon$ implies it becomes unobservable at state q_2.

Furthermore, a sequence of events forms a string, and thus a string can be observed as its component events are sequentially observed. That is, $M: Q\times\Sigma\to\Theta\cup\{\varepsilon\}$ can be extended to $M: Q\times\Sigma^*\to\Theta^*$ by specifying

(1) $M(q,\varepsilon)=\varepsilon$

(2) $M(q,s\sigma)=M(q,s)M(\delta(q,s),\sigma)$.

Because a language is a set of strings, an observation function can also mask a language as each string in the language is masked, and the resulting language is given by

$$M(K)=\{d\in\Theta^*\mid\exists s\in K, s.t.\ M(q_0,s)=d\}.$$

Besides, because such observation process is related to the system state and the transition function of *G* is often a partial function, it is necessary to define *M* only on part of $Q\times\Sigma$. In brief, it is meaningless to define $M(q,\sigma)$ if there is no transition σ at state q. This paper will write $M(q,\sigma)!$ if $M(q,\sigma)$ is defined, and write $\neg M(q,\sigma)!$ if it is not defined. The domain of *M* is thus specified as

For $q\in Q$ and $\sigma\in\Sigma$, $M(q,\sigma)!$ iff $\delta(q,\sigma)!$.

To take into account the new function *M*, the preceding framework goes through with some minor modification. A

supervisor of (G,M) is a pair $\Phi=(S,\psi)$, where the automaton $S=(X,\Theta,\xi,x_0)$ has the input Θ; The state feedback map $\psi:X\to\Gamma$ is as before. As Φ is coupled to (G,M), a supervised discrete event system is given as

$$\Phi/G=A(X\times Q,\ \Sigma,(\xi\circ M)\times\delta_c,(x_0,q_0)).$$

where the transition function is defined by

$$((\xi\circ M)\times\delta_c)(x,q,\sigma)=\begin{cases}\xi(x,M(q,\sigma)),\delta(q,\sigma)\\ \text{if } \sigma\in\psi(x),M(q,\sigma)!,\delta(q,\sigma)!\\ \text{undefined}\qquad\text{otherwise}\end{cases}$$

As for a DES with state-dependent observability, a supervisor is called complete if

$$s\in L(\Phi/G)\wedge s\sigma\in L(G)\wedge\sigma\in\psi(\xi(x_0,M(q_0,s)))\Rightarrow s\sigma\in L(\Phi/G)$$

This definition implies that automaton in the supervisor will not control the state transition of *G*. Rather, it is the state feedback that controls the plant. The next question is how to find a required supervisor to meet a specification. Some specification can be satisfied while others are not. It is therefore necessary to investigate certain properties of a specification to discuss whether a supervisor exists.

3. Existence of Supervisors

This section discusses under what condition there exists a supervisor for a given specification. In the sense of math, a specification is often given by a formal language. According to existing results of a fully observable DES [9], a supervisor exists if the specification is given as a invariant language. The definition of such a language is

$$K\text{ is }(\Sigma_{uc},L(G))-\text{invariant}\Leftrightarrow s\in\bar{K}\wedge\sigma\in\Sigma_{uc}\wedge s\sigma\in L(G)\to s\sigma\in\bar{K}$$

As for a DES with state-dependent observability, we will further look into the observability of a language. The definition of an observable language is given as below.

Definition 3.1: Let (G,M) denote a partially observable DES with $M:Q\times\Sigma\to\Theta\cup\{\varepsilon\}$. Let $K\subseteq L(G)\subseteq\Sigma^*$ and $\Omega\subset\Sigma$. language *K* is $(M,\Omega,L(G))-$observable if

$$s,s'\in K\wedge\sigma\in\Omega\wedge s\sigma\in K\wedge s'\sigma\in L(G)\wedge M(q_0,s)=M(q_0,s')\Rightarrow s'\sigma\in K.$$

The observability of a language indicates that if two event sequences look the same, they must be consistent in the sense that no conflict exists for one event continuable after one sequence while not continuable after the other. By the above definition, the supervisor existence problem is studied through a theorem as given below.

Theorem 3.1: Let (G,M) be a partially observable DES with observation function defined by $M:Q\times\Sigma\to\Theta\cup\{\varepsilon\}$. There exists a complete supervisor $\Phi=<S,\psi>$ such that $L(\Phi/G)=K$ if and only if (1) $K\subseteq L(G)$, (2) *K* is prefix-closed, (3) *K* is $(\Sigma_{uc},L(G))-$invariant, (4) *K* is $(M,\Sigma_c,L(G))-$observable.

Proof: (Only if). Suppose we have a complete supervisor $\Phi=<S,\psi>$ such that $L(\Phi/G)=K$, and we are to verify the properties of *K*.

(1), (2) are trivial. (3) holds since the supervisor can only disable controllable events.

To prove (4), we are given $s,s'\in K$, $\alpha\in\Sigma_c$ such that $M(q_0,s)=M(q_0,s')$, $s\alpha\in K$ and $s'\alpha\in L(G)$. By $s\in K$ and $s\alpha\in K$, we will know that event α is enabled after sequence *s* occurs, and thus $\alpha\in\psi(\xi(x_0,M(q_0,s)))$. Because $M(q_0,s)=M(q_0,s')$, *s* and *s*′ are the same as observed by the supervisor, and thus event α is enabled after *s*′ occurs, i.e., $\alpha\in\psi(\xi(x_0,M(q_0,s')))$. Because $s'\alpha\in L(G)$ and Φ is complete, it is clear that $s'\alpha\in K$, and thus *K* is $(M,\Sigma_c,L(G))-$observable by definition.

(If). To prove the sufficiency of the theorem a supervisor $\Phi=<S,\psi>$ is to be synthesized such that $L(\Phi/G)=K$. Based on property (a) and (b) in Lemma (See Appendix), a supervisor can be synthesized as below.

Let $S=(X,\Theta,\xi,x_0)$ denote the automaton of the supervisor where $X=M(K)$; $x_0=\varepsilon$;

$$\xi(d,\theta)=\begin{cases}d\theta & d\theta\in M(K)\\ \text{undefined} & d\theta\notin M(K)\end{cases}.$$

Synthesize state feedback map as

$$\sigma\in\psi(d)\ \text{ iff }\ \sigma\in\Sigma_{uc}\cup\Sigma(d).$$

Then a supervisor is synthesized and it satisfies property (a) and (b) in the lemma (See Appendix). Thus, $L(\Phi/G)=K$ and $\Phi=<S,\psi>$ is complete. Q.E.D.

The theorem above presents a necessary and sufficient condition for existence of a complete supervisor. Because the resulting DES under supervision is also an automaton, the theorem addresses equivalence of formal languages and automata. Besides, the theorem does not require the language to be regular, implying that the automaton is not necessarily finite. However, in practical computing finite state automata are usually required, and the corresponding languages are thus regular.

Definition 3.2: Let (G,M) denote a partially observable DES with $M:Q\times\Sigma\to\Theta\cup\{\varepsilon\}$. Let $K\subseteq L(G)\subseteq\Sigma^*$, and language *K* is $(M,L(G))-$recognizable if

$$s\in K\wedge s'\in L(G)\wedge M(q_0,s)=M(q_0,s')\Rightarrow s'\in K.$$

In the previous framework, a $(M,L(G))-$recognizable language implies that it is $(M,\Omega,L(G))-$observable. Is the proposition still valid for a DES with state-dependent observability? The following proposition gives an answer.

Proposition 3.1: Let (G,M) be a partially observable

DES with $M: Q \times \Sigma \to \Theta \cup \{\varepsilon\}$. Let $K_1, K_2 \subseteq L(G)$

(1) If K_1, K_2 are $(M, L(G))-$recognizable, then $K_1 \cap K_2$, $K_1 \cup K_2$ are $(M, L(G))-$recognizable

(2) If K_1, K_2 are $(M, \Omega, L(G))-$observable, $K_1 \cap K_2$ is $(M, \Omega, L(G))-$observable, but $K_1 \cup K_2$ is not

(3) If K_1 is $(M, L(G))-$recognizable, it does **NOT** imply that K_1 is $(M, \Omega, L(G))-$observable

(4) For a language *K*, its $(M, \Sigma_c, L(G))-$observability is independent of its $(\Sigma_{uc}, L(G))-$invariablity.

Proof. (1) and (2) are trivial, and they are similar to the traditional results. As for (3), $M(q_0, s) = M(q_0, s')$ does NOT imply $M(q_0, s\sigma) = M(q_0, s'\sigma)$ because an event σ can be observable at $q_1 = \delta(q_0, s)$ while unobservable at $q_2 = \delta(q_0, s')$. Thus, (3) holds. For (4), it holds because $\Sigma_c \cap \Sigma_{uc} = \emptyset$. Thus, invariability on Σ_{uc} is not related to the observability on Σ_c.

By Proposition 2.1, the supremal sublanguage which is prefix-closed, invariant, observable and within the given specification may not exists. Such a difficulty also existed in the previous framework where the observation function was specified by $M: \Sigma \to \Theta \cup \{\varepsilon\}$. In the previous work [1], this problem was solved by finding the supremal recognizable language because recognizable languages are closed under union, and if a language is recognizable, it is an observable language. However, this approach cannot be followed because of (3) in Proposition 3.1. Thus, we must find a new approach to solve this problem.

4. Supervisor Synthesis Problem

After the supervisor existence problem, this section will focus on the supervisor synthesis problem. Two specific problems are investigated. The first problem is to judge whether there exists a supervisor for a specification as given by a formal language. The next problem is to synthesize a supervisor if there exists one for the specification. Two algorithms are given as solutions to the problems.

Based on Theorem 3.1, we need to verify the necessary properties of a given specification. In practical problems, a specification is often given as a set of a state-transition rules (See the example in Section 4). The corresponding language of a state-transition machine is naturally prefix-closed. Also, in this paper a specification is considered to control an existing system, but not to extend the behaviors of the existing system. Thus, the specification language *K* is a subset of the original language, i.e., $K \subseteq L(G)$. As a result, a control specification in practical computing often meets the first two properties in Theorem 3.1[1], and the key features to be checked are the controllability and observability of the specification language.

Furthermore, in particular, this section will consider the specification to be a regular language because in practical computing only finite state automata are considered, and the corresponding languages are regular. The problem is next addressed as below.

Problem 1: Let (G, M) be a partially observable DES with observation function defined by $M: Q \times \Sigma \to \Theta \cup \{\varepsilon\}$. Consider *K* as a regular language such that $K \subseteq L(G)$ and $K = \bar{K}$, test whether *K* satisfies

(1) *K* is $(\Sigma_{uc}, L(G))-$invariant;

(2) *K* is $(M, \Sigma_c, L(G))-$observable.

Algorithm 1:

Step 1:

Let *K* be a regular language such that $K \subseteq L(G)$ and $K = \bar{K}$, synthesize a finite deterministic automaton $T = (Y, \Sigma, \eta, y_0)$ such that $L(T) = K$ and *T* refines *G*. Thus, there exists a unique function $h: Y \to Q$ such that for any $s \in K$, $h \circ \eta(y_0, s) = \delta(q_0, s)$ holds [1].

Let *X* denote the set of all nonempty subsets of *Y* and define the automaton $S = A(X, \Theta, \xi, x_0)$ where

$$x_0 = \{\eta(y_0, s) \mid M(h(y_0), s) = \varepsilon, s \in \Sigma^*\},$$

$$\xi(x, \theta) = \begin{cases} \{\eta(y, s) \mid y \in x, M(h(y), s) = \theta, s \in \Sigma^*\} & \text{if it is nonempty} \\ \text{undefined} & \text{otherwise} \end{cases}.$$

Construct $\hat{G} = A(T \times S \times G)$

$$= A(Y \times X \times Q,\ \Sigma,\ \hat{\delta},\ <y_0, x_0, q_0>)$$

where

$$\hat{\delta}(<y_0, x_0, q_0>, s) = \begin{cases} <\eta(y_0, s), \xi(x_0, M(q_0, s)), \delta(q_0, s)> & \text{if all defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Step 2: Check invariability of *K*.

Let $(Y \times X \times Q)_a$ denote the state set of $\hat{G}$. Check each $<y, x, q> \in (Y \times X \times Q)_a$ and each $\sigma \in \Sigma$ such that $\delta(q, \sigma)!$ and $\neg\hat{\delta}(<y, x, q>, \sigma)!$, if $\sigma \in \Sigma_c$, then *K* is $(\Sigma_{uc}, L(G))-$invariant; otherwise *K* is not.

Step 3: Check observability of *K*.

Let $(Y \times X \times Q)_a$ be the state set of $\hat{G}$. Check each $\sigma \in \Sigma_c$ and $<y', x', q'>, <y, x, q> \in (Y \times X \times Q)_a$, such that $x = x'$, $\delta(q, \sigma)!$ and $\delta(q', \sigma)!$, if

$$\neg\hat{\delta}(<y, x, q>, \sigma)! \text{ implies } \neg\hat{\delta}(<y', x', q'>, \sigma)!$$

then *K* is $(M, \Sigma_c, L(G))-$observable; otherwise *K* is not.

To prove Algorithm 1, propositions are given as below.

Proposition 4.1: By Step 1 and Step 2 in Algorithm 1,

K is $(\Sigma_{uc}, L(G))-$invariant $\Leftrightarrow$

[1]If needed, $K \subseteq L(G)$ and $K = \bar{K}$ can be checked by a method in [8].

$$(\forall <y,x,q>\in (Y\times X\times Q)_a)(\forall \sigma\in\Sigma)$$
$$(\neg\hat{\delta}(<y,x,q>,\sigma)!\wedge\delta(q,\sigma)!\rightarrow\sigma\in\Sigma_c)$$

Proof: By the definition of $\hat{G}$ we have $L(\hat{G})=K$ and $\hat{G}$ refines G [1]. Then the proposition is restated as

$$K \text{ is } (\Sigma_{uc},L(G))-\text{invariant}\Leftrightarrow$$
$$(\forall s\in K)(\forall\sigma\in\Sigma)(s\sigma\notin K\wedge s\sigma\in L(G)\rightarrow\sigma\in\Sigma_c).$$

The right side of the above statement is equivalent to

$$(\forall s\in K)(\forall\sigma\in\Sigma)(\sigma\in\Sigma_{uc}\wedge s\sigma\in L(G)\rightarrow s\sigma\in K),$$

Thus, the proposition holds by the definition of invariant language. Q.E.D.

Proposition 4.2: By Step 1 and Step 3 in Algorithm 1, we have K is $(M,\Sigma_c,L(G))-\text{observable}\Leftrightarrow$

$$(\forall <y,x,q>,<y',x',q'>\in(Y\times X\times Q)_a)(\forall\sigma\in\Sigma_c)$$
$$(\delta(q,\sigma)!\wedge\delta(q',\sigma)!\wedge(x=x')\wedge\neg\hat{\delta}(<y,x,q>,\sigma)!\rightarrow$$
$$(\neg\hat{\delta}(<y',x',q'>,\sigma)!)$$

Proof: By $L(\hat{G})=K$ and $\hat{G}$ refines G, the proposition is restated as follows. For $(\forall s\in K)(\forall s'\in K)(\forall\sigma\in\Sigma_c)$

$$K \text{ is } (M,\Sigma_c,L(G))-\text{observable}\Leftrightarrow$$
$$s\sigma\notin K\wedge s\sigma\in L(G)\wedge M(q_0,s)=M(q_0,s')\rightarrow s'\sigma\notin K$$

Consider the right side is equivalent to

$$(s'\sigma\in K\wedge s\sigma\in L(G)\wedge M(q_0,s)=M(q_0,s')\rightarrow s\sigma\in K),$$

Then the proposition holds by the definition of $(M,\Sigma_c,L(G))-\text{observable}$ language. Q.E.D.

Based on (4) in Proposition 3.1, invariability on Σ_{uc} is independent of the observability on Σ_c. Thus, Step 2 and Step 3 of Algorithm 1 are also independent, and they can be swapped in sequence if needed.

Problem 2: Let (G,M) be a partially observable DES with observation function defined by $M:Q\times\Sigma\rightarrow\Theta\cup\{\varepsilon\}$. The specification is a regular language K which satisfies

(1) $K\subseteq L(G)$,
(2) K is prefix-closed,
(3) K is $(\Sigma_{uc},L(G))-\text{invariant}$,
(4) K is $(M,\Sigma_c,L(G))-\text{observable}$.

Synthesize a complete supervisor such that $L(\Phi/G)=K$.

The proof of Theorem 3.1 has presented a method to synthesize a required supervisor, but the supervisor as synthesized is always an infinite one (the automaton has infinite state set). Thus, the method cannot be applied to practical problems, and it is necessary to find an approach such that a finite supervisor can be synthesized, especially for the specification as given by a regular language.

Algorithm 2:

Step1: Synthesize automaton S.

Since K is prefix-closed, construct $T=(Y,\Sigma,\eta,y_0)$ such that $L(T)=K$ and T refines G [1]. Then there exists a unique function $h:Y\rightarrow Q$ such that for arbitrary $s\in K$, $h\circ\eta(y_0,s)=\delta(q_0,s)$ holds [1]. Let X denotes the set of all nonempty subsets of Y and define an automaton $S=A(X,\Theta,\xi,x_0)$ where

$$x_0=\{\eta(y_0,s)|M(h(y_0),s)=\varepsilon,s\in\Sigma^*\},$$
$$\xi(x,\theta)=\begin{cases}\{\eta(y,s)|y\in x,M(h(y),s)=\theta,s\in\Sigma^*\} & \\ \qquad\qquad\text{if it is nonempty} & \\ \text{undefined}\qquad\text{otherwise}\end{cases}.$$

Step2: Synthesize state feedback map ψ.

For $x\in X$, $f(x)$ is defined by $f(x)=\{\sigma\in\Sigma_c|\exists y\in x,\ s.t.\ \eta(y,\sigma)!\}$. Then the feedback map is defined by $\sigma\in\psi(x)$ iff $\sigma\in f(x)\cup\Sigma_{uc}$.

The correctness of Algorithm 2 can be checked based on the lemma and propositions in Appendix. If needed, the resulting supervisor can be further reduced in its state set by a method in [9].

Further, if K is a regular language, then Y can be a finite set, and thus X is finite through Step 1 in Algorithm 2. As a result, for a specification as given by a regular language, the supervisor can be synthesized in a finite form.

To summarize the work of this section, usual procedures for synthesizing supervisors are given based on the two algorithms introduced above. First we use Algorithms 1 to check whether there exists a required supervisor for the given control specification; if there exists one, Algorithm 2 is applied to synthesize it.

5. Illustrative Examples

Examples are illustrated in this section to show how to apply the algorithms to practical computing problems. A transport system is considered as a model to be supervised within the background of manufacturing systems.

As presented in Section 1, a container with workpieces can be characterized as a event-driven system with state-dependent observability. In this section such a container is placed on transport line so that it moves between a loading site and an unloading site. A machine will put workpieces into the container at the loading site, and another machine remove them at the unloading site. The entire system is illustrate in Figure 2.

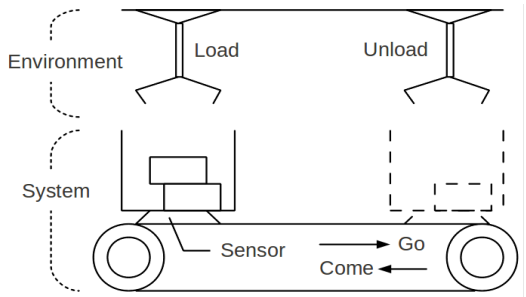

Figure. 2. Transport system and its environment

The physical system is abstracted to a block diagram as shown in Figure 3. Here the pressure sensor masks off some information, and thus the supervisor will implement control based on the partial information that the senor provides. The environment, which the system reacts to, is conducted by the behaviors of the loading machine and unloading machine[2], and it generates the discrete events that drive the transport system to respond. Thus, the action of loading and unloading are uncontrollable events to the transport line, but are partially observable to the system. In contrast, how to transport the container between the two sites are behaviors inside the system, and they can be observed and controlled by the supervisor.

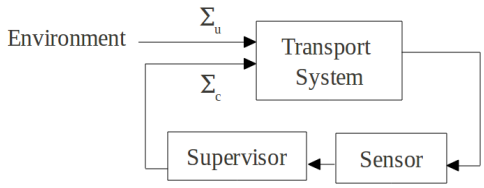

Figure. 3. Block diagram of the system components.

Let $\{L, U\}$ denote the location set of the container, where L and U represent the loading and unloading site, respectively. Let $\{0,1,2\}$ denote the storage index of the container, where the integers represent the number of workpieces inside the container. Let $G=(Q, \Sigma, \delta, q_0, Q)$ denote the system. As a result, the state set of G is given by $Q=\{L, U\}\times\{0,1,2\}$. The event set of the system G is $\Sigma=\{load, unload, go, come\}$, and the meaning of such symbols are listed as below.

Table 1. Event Explanation

Load	Load workpieces into the container
Unload	UnLoad workpieces from the container
Go	Go to the unloading site
Come	Come to the loading site

$\Sigma_c=\{go, come\}$ are the internal events within the system, and $\Sigma_{uc}=\{load, unload\}$ are the external events that are generated by the environment. The transition function is defined as shown in Figure 4. And the initial state is *L0*.

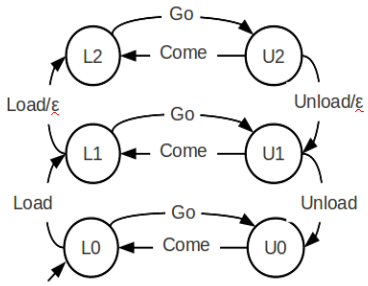

Figure. 4. State transition diagram of the original system

Let $(location, storage)$ denote an arbitrary state of G. By the feature of the pressure sensor, the observation function M is defined by Table 1.

Table 2. Observation Function of Machine Actions

	Loading Site	Unloading Site
Storage = 0	Observable	None
Storage = 1	Unobservable (ε)	Observable
Storage > 1	Unobservable (ε)	Unobservable (ε)

It is noted that event *load* is observable in state *L0* while unobservable in *L1*, and event *unload* is observable in *U1* while unobservable in *U2*. The partially observable DES is thus modeled with state-dependent observability.

Example 1: As to maximize efficiency of the transport system, the system is required to be controlled as below. When the container is fully loaded, it will be moved to the unloading site; and when the container is unloaded to be empty, it will return to the loading site. This gives the specification as shown in Figure 5.

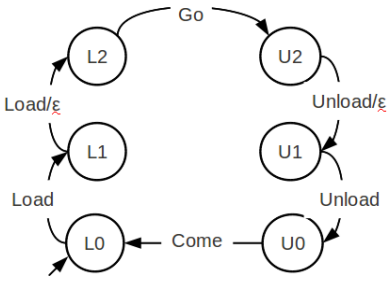

Figure. 5. Specification of Example 1

Since the specification is given in the state-transition form, $T=(Y, \Sigma, \eta, y_0, Y)$ is obtained as shown in Figure 5. Let the specification language be $K=L(T)$. Then it is clear that $K\subseteq L(G)$ and K is prefix-closed. Therefore,

[2] Here we refer to the concepts of systems and environments in the reactive systems theory [10].

Algorithm 1 can be applied to check the controllability and observability of language *K*.

Construct *S* from *T* by Step 1 in Algorithm 1, and *S* is presented as shown in Table 3, and $\hat{G}$ is further given in Table 4. Here the finite automaton is represented by its transition matrix. For example, the transition matrix of $\hat{G}$ is a finite square matrix Δ, whose columns and rows are both indexed by the state set of $\hat{G}$. The entries $\Delta(q_1,q_2)$ is an event, which means the system state transits from q_1 to q_2 when the event occurs. By following Step2 and Step 3 in Algorithm 1, it can be checked by $\hat{G}$ that *K* is $(\Sigma_{uc},L(G))-$invariant. However, the gray cell in the Table 4 shows that K is not $(M,\Sigma_c,L(G))-$observable. Thus, there does not exist a supervisor to satisfy the given specification.

Example 2: The control specification is then revised as illustrated in Figure 7.

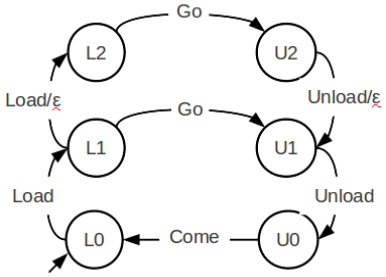

Figure 7. Specification of Example 2

Figure 7 presents T where the specification language is $K=L(T)$. By the same steps, S is obtained as shown in Table 3, and $\hat{G}$ is given by Table 5. Compared Table 5 with Table 4, it is noted that event "Go" is enabled at the state L1,(L1,L2),L1. Thus, it is checked by Algorithm 1 that specification *K* is both $(\Sigma_{uc},L(G))-$invariant and $(M,\Sigma_c,L(G))-$observable, and thus Algorithm 2 can be applied to synthesize the supervisor. The automaton in the supervisor is directly obtained by *S*, and the feedback map is synthesized in Table 5. The supervisor is finally given as shown in Figure 8.

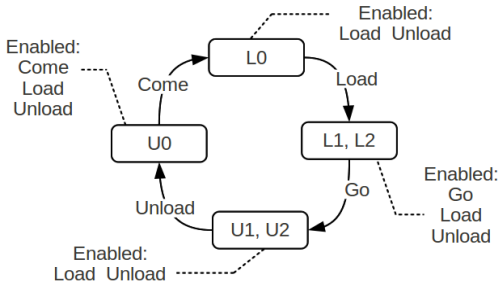

Figure. 8. Supervisor

It is shown from Table 6 that the supervisor synthesized can be realized by a policy of if-and-then rules. Such a control policy can easily be implemented in practice.

Table 6. The control policy as synthesized in Example 2.

If the state is:	then disabled:
L0	Go
U1,U2	Come

Table 3. Automaton S in Example 1 and 2.

	L0	L1,L2	U1,U2	U0
L0		Load		
L1,L2			Go	
U1,U2				Unload
U0	Come			

Table 4. Transition Matrix of $\hat{G}$ in Example 1

	L0, (L0), L0	L1, (L1,L2), L1	L2, (L1,L2), L2	U2, (U1,U2), U2	U1, (U1,U2), U1	U0, (U0), U0
L0, (L0), L0		Load				~~Go~~
L1,(L1,L2),L1			Load		~~Go~~	
L2,(L1,L2),L2				Go		
U2,(U1,U2),U2			~~Come~~		Unload	
U1,(U1,U2),U1		~~Come~~				Unload
U0,(U0),U0	Come					

Table 5. Transition Matrix of $\hat{G}$ in Example 2

	L0, (L0), L0	L1, (L1,L2), L1	L2, (L1,L2), L2	U2, (U1,U2), U2	U1, (U1,U2), U1	U0, (U0), U0	Disabled
L0, (L0), L0		Load				Go	Go
L1,(L1,L2),L1			Load		Go		None
L2,(L1,L2),L2				Go			None
U2,(U1,U2),U2			Come		Unload		Come
U1,(U1,U2),U1		Come				Unload	Come
U0,(U0),U0	Come						None

6. Conclusions

This paper presents a new model of discrete-event systems with state-dependent observability, and our study aims to be of help to system verification and redesign.

A new observation function is first presented to be state-dependent. Next, given a control specification as a formal language, a necessary and sufficient condition is derived for the existence of the supervisor, and it refers to closure, invariability and observability of the specification language. The supervisor synthesis problem is solved by two algorithms. One algorithm is to verify the properties of the specification language, and the other one is to synthesize the required supervisor if there exists one. Our work is mainly an extension of the preceding framework by Lin and Wonham.

Several follow-up research topics can be approached in future. One topic is to advance our method for a more complex system such as a multi-computer system within a network [11]. Besides, another possible topic is to explore implementing our algorithms by using the state-flow in MATLAB [12] so that a system can be synthesized in a computationally efficient manner.

Appendix

Lemma: Let (G,M) be a partially observable DES with observation function defined by $M:Q\times\Sigma\to\Theta\cup\{\varepsilon\}$. Let K represent a language such that (1) $K\subseteq L(G)$, (2) K is prefix-closed, (3) K is $(\Sigma_{uc},L(G))-$invariant, (4) K is $(M,\Sigma_c,L(G))-$observable. For $d\in\Theta^*$, Let

$\Sigma(d)=\{\sigma\in\Sigma_c|\exists s\in K, s.t.\ M(q_0,s)=d,\ s\sigma\in K\}$.

If the supervisor $\Phi=<S,\psi>$ satisfies

(a) $\xi(x_0,d)$ is defined iff $d\in M(K)$,

(b) $\sigma\in\psi(\xi(x_0,d))$ iff $\sigma\in\Sigma_{uc}\cup\Sigma(d)$.

Then $L(\Phi/G)=K$ and $\Phi=<S,\psi>$ is complete.

Proof: First we prove $K\subseteq L(\Phi/G)$ by induction of $|s|$.

For $|s|=0$, namely, $s=\varepsilon$, we have $s=\varepsilon\in K\Rightarrow s=\varepsilon\in L(\Phi/G)$.

For $|s|=j$, suppose $s\in K\Rightarrow s\in L(\Phi/G)$ holds. Let $|\sigma|=1$ and $s\sigma\in K$, and four items are derived as below.

(1) $K\subseteq L(G)$ and $s\sigma\in K$ implies $s\sigma\in L(G)$.

(2) $s\in L(\Phi/G)$ holds by the induction hypothesis.

(3) Because $s\sigma\in K$, $\xi(x_0,M(q_0,s\sigma))$ is defined by property (a).

(4) Suppose $M(q_0,s)=d$. If $\sigma\in\Sigma_c$, there exists $s\in K$ such that $s\sigma\in K$ and $M(q_0,s)=d$, and therefore $\sigma\in\Sigma(d)$ holds. If $\sigma\in\Sigma_{uc}$, $\sigma\in\Sigma_{uc}\cup\Sigma(d)$ holds. Thus we have $\sigma\in\psi(\xi(x_0,d))$ by property (b).

(1)-(4) as above together imply that $s\sigma\in L(\Phi/G)$, and $K\subseteq L(\Phi/G)$ is thus proved.

Next, we prove $K\supseteq L(\Phi/G)$ by induction.

For $|s|=0$, $s=\varepsilon\in L(\Phi/G)\Rightarrow s=\varepsilon\in K$ holds.

For $|s|=j$, suppose that $s\in L(\Phi/G)\Rightarrow s\in K$. Let $|\sigma|=1$ and $s\sigma\in L(\Phi/G)$. Then $s\sigma\in K$ is to be proved.

For $M(q_0,s)=d$, $s\in L(\Phi/G)$ and $s\sigma\in L(\Phi/G)$ together imply that $\sigma\in\psi(\xi(x_0,d))$ holds, and therefore $\sigma\in\Sigma_{uc}\cup\Sigma(d)$ by property (b). If $\sigma\in\Sigma_{uc}$, then $s\sigma\in K$ because K is $(\Sigma_{uc},L(G))-$invariant. If $\sigma\in\Sigma(d)$, there exists $s'\in K$ such that $M(q_0,s')=d$ and $s'\sigma\in K$, and then $s\sigma\in K$ since K is $(M,\Sigma_c,L(G))-$observable.

Because $L(\Phi/G)=K$, it is checked from property (a) that the supervisor $\Phi=<S,\psi>$ is complete. Q.E.D.

Proposition 1: Synthesize $S=A(X,\Theta,\xi,x_0)$ by Step 1 of Algorithm 2. For $d\in\Theta^*$,

$$\xi(x_0,d)=\begin{cases}\{\eta(y_0,s)|M(h(y_0),s)=d,s\in\Sigma^*\} & \\ \qquad\qquad\text{if it is nonempty} & \\ \text{undefined}\qquad\text{otherwise} & \end{cases}.$$

Proof: Trivial.

By $M(K)=\{d\in\Theta^*|\exists s\in K,s.t.\ M(q_0,s)=d\}$ and $L(T)=K$, the proposition indicates that $\xi(x_0,d)$ is defined if and only if $d\in M(K)$. Thus, S as synthesized by Step 1 satisfies property (a).

Proposition 2: For $x \in X$, $\Sigma(x)$ is defined by

$$\Sigma(x)=\{\sigma | \sigma \in \Sigma_c, \exists y \in x, s.t. \eta(y,\sigma)!\},$$

If $\xi(x_0, d)=x$, then $\Sigma(x)=\Sigma(d)$ holds.

Proof: Trivial. It is clear that map ψ meets property (b).

References

[1] F. Lin and W.M. Wonham, "On Observability of Discrete-Event systems", Information Sciences, vol. 44, 1988, pp. 173-198.

[2] R. Cieslak, C. Desclaux, A. S. Fawaz and P. Varaiys, "Supervisory Control of Discrete-Event Processes with Partial Observation," IEEE Transactions on Automatic Control, vol. 33, no. 3, 1988, pp. 249-260.

[3] W. M. Wonham, "Supervisory Control of Discrete-Event Systems", ECE 1636F/1637S 2007-08, Updated 2011.07.01, Dept. of Electrical & Computer Engineering, University of Toronto.

[4] F. Lin, W. M. Wonham, "Decentralized Control and Coordination of Discrete-Event Systems with Partial Observation", IEEE Transactions On Automatic Control, vol. 35, no. 12, 1990, pp. 1330-1337.

[5] H. Marchant, O. Boivineaub, S. Lafortune, "On Optimal Control of a Class of Partially Observed Discrete Event System", Automatica, vol. 38, 2002, pp. 1935-1942.

[6] R. Kumar, V. Garg, "Optimal Supervisory Control of Discrete Event Dynamical Systems", SIAM Journal of Control and Optimization, vol. 33, 1995, pp. 419-439.

[7] S. Takai, T. Ushio , "Supervisory Control of a Class of Concurrent Discrete Event Systems Under Partial Observation ," Discrete Event Dynamic Systems: Theory and Applications, Vol. 15, 2005, pp. 7–32.

[8] P. Wang and K. Y. Cai, "Supervisory control of discrete event systems with state-dependent controllability,"International Journal of Systems Science, vol. 40, No. 4, 2009, pp. 357-366. 2009.

[9] R. Su, W. M. Wonham , "Supervisor Reduction for Discrete-Event Systems ," Discrete Event Dynamic Systems: Theory and Applications, Vol. 14, 2004, pp. 31-53.

[10] D. Harel and M. Politi, "Modeling Reactive Systems With Statecharts: The STATEMATE Approach", New York, McGraw-Hill, 1998.

[11] Cai, K. Y., Cangussu, J. W., DeCarlo R. A., Mathur, A. P. An Overview of Software Cybernetics. Proceedings of the 11th Annual International Workshop on Software Technology and Engineering Practice, IEEE Computer Society Press. 2004, pp. 77-86.

[12] "Stateflow User Guide, For state diagram modeling," Math Works, Inc., 2001.