On the Design of Adaptive Supervisors for Discrete Event Systems

Vigyan CHANDRA
Department of Technology, Eastern Kentucky University
Richmond, KY 40475, USA

and

Siddhartha BHATTACHARYYA
Division of Computer and Technical Sciences, Kentucky State University
Frankfort, KY 40601, USA

ABSTRACT

The supervised control of complex event-driven Discrete Event Systems (DESs) such as those present in manufacturing systems, or the communication processes involved therein, continue to pose a challenge to system designers. To a certain extent this complexity can be reduced by applying existing modular control approaches to large-scale DES design. These solutions divide the system into different sections in such a way that its overall behavior is given by a suitable arrangement of the different sections. However, if the system is reconfigured frequently, the overall plant models and control specifications computed earlier would no longer be valid. Thus a new controlled system will have to be computed. We propose a new methodology, for ensuring that the new controlled plant will meet any valid control specifications taken from the existing modules. Being built on the framework of Supervisory control theory, this method is guaranteed to work even as the system is being dynamically reconfigured.

Keywords: Supervisory Control Theory, Adaptive supervisors, Discrete Event System, Automata, Control specifications.

1. INTRODUCTION

In many complex systems, such as those found in manufacturing and communication, the evolution of the system behavior is characterized by abrupt changes corresponding to the occurrences of events. Such systems are commonly termed as Discrete Event Systems (DESs) [8, 9]. Of the event transitions occurring in the system, also called the *plant*, certain events can be disabled by a supervisor controlling the plant behavior. Such events are called *controllable events*, and all other events are called *uncontrollable events*. This behavior of the system can be described by a language model and equivalently represented by an automaton as well. Prior to controlling the system a model which accurately represents all possible behavior starting from an initial state is required. The overall plant behavior is obtained by tracking a succession of events, both controllable and uncontrollable ones, originating from a single initial state. A supervisor acting in conjunction with the plant restricts its behavior in order to achieve some control objectives. In doing so the supervisor may delete controllable events. Such deletions may make certain states of the plant unreachable from the initial state. Hence the reachability of any state in the plant is usually checked with reference to the initial state as part of the supervision procedure.

Instead of creating a large and complex plant model, frequently the system might be split into sub-models and the interaction of these sub-models provides an overall model of the system. In this paper we discuss scenarios wherein a complex system composed of several sub-models is reconfigured, and it is required that a new supervisor for the reconfigured plant be obtained adaptively based on the changes made to the system structure. All the events in the sub-models are regarded as controllable, and the overall plant is obtained by including all possible interactions of the sub-models. As the system is being reconfigured some of the sub-models will no longer be needed whereas other new sub-models might need to be included.

We propose an automata theory based solution to the above problem and provide an algorithm which uses the power of existing supervisory control theory (SCT) [1, 6, 10] for making sure that events from sub-systems which are no longer part of the reconfigured system are eliminated in such a way as to maintain linkages between original states of the control specification. If the events which are no longer part of the reconfigured plant are simply deleted in the control specification, this could cause various parts of the supervised plant to become unreachable, and this in turn this might lead to the control objective no longer being feasible.

A two-part solution to the above problem is described in this paper: one part applied to the plant sub-models, and the second part applied to the control specification. Since the overall model of the plant is formed by the interaction of sub-models, the events from the sub-model which is dropped can be eliminated either by deleting them in the overall model followed by finding a reachability of all states from the initial state; or by finding a new overall interaction model for the plant after eliminating the sub-model which has been dropped.

For reconfiguring the control specification which now should no longer contain events which belong to the sub-model that has been dropped, we replace these events by silent or ε events. The ε event trace represents the absence of any event, or alternately signifies that if an event exists its occurrence cannot be observed. Instead of causing a break where the event from the dropped sub-model formerly was, it now gets seamlessly connected via the ε event to the next state in the control specification. The notion of ε is especially important when dealing with such automata models which can abruptly change state apparently without the occurrence of an event, thereby causing non-determinism in the system. This non-determinism can be removed easily by creating an equivalent deterministic version of the specification automaton. Once the reconfigured plant and control specifications are available, a new supervisor for the plant can be obtained using existing SCT algorithms. Doing so will ensure that the reconfigured supervisor does indeed meet all the specifications.

The rest of the paper is arranged as follows: Section 2 provides an overview of several important concepts used in the modeling and supervisory control of DESs, as well as a review of the literature related to adaptive supervisory control. Section 3 illustrates the plant-control specification reconfiguration problem through the use of an example drawn from a DES. In Section 4 we provide an adaptive supervisory control solution for dealing with the reconfiguration problem of DESs in the presence of uncertainty regarding changes in system or control specifications. Section 5 provides conclusions of the work and Section 6 contains bibliographic references.

2. OVERVIEW OF AUTOMATA AND SUPERVISORY CONTROL THEORY

In order to express the behavior of a system, a language model and equivalently automata may be used. These models are created using the events in the system.

Automata
Automata [5, 10] can be used for representing the un-timed behavior of language models and of the qualitative specifications needed for its control. Formally, an automaton G is represented as a 5-tuple: $G = (X, \Sigma, \delta, x_0, X_m)$, where X denotes the set of states, Σ denotes the finite set of events, $\delta: X \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^X$ is the partial state information function, and $x_0 \in X$ is the initial state, and X_m denotes the set of marked states.

G is called a *deterministic* automaton if the transition function is a partial function of the form: $\delta: X \times \Sigma \rightarrow X$, signifying that from any state in the automaton, the transition function of any event defined at that state may take it to exactly one state. G is is called a *non-deterministic* automaton if the partial transition function is of the form: $\delta: X \times \Sigma \rightarrow 2^X$, signifying that within the automaton there are states, from which the transition function of some events defined at that state take it to more than one state. Not all events might be possible at all the states of the automaton and the transition function defines which events are possible at which states. A triple formed by $(x, \sigma, x') \in X \times (\Sigma \cup \{\varepsilon\}) \times X$ is called a *transition* of G where $x' \in \delta(x, \sigma)$. A transition is said to be an epsilon-transition if $\sigma = \varepsilon$, that is, the transition can occur on the ε (or unobservable) event.

Removal of non-determinism
The finite set of events is denoted by Σ, and different arrangements of these events forms a string of events, also called a *trace*. A *language* is defined as a set of traces. If we let Σ^* represent the set of all finite traces of events in Σ including the empty trace ε, then it can be seen that any language over the same event set, is a subset of Σ^*. The ε event trace represents the absence of event, or alternately signifies that if an event exists its occurrence cannot be observed. When only some of the events in a system can be observed, the system is said to be operating under *partial observability*.

If we let a subset of events, $\Sigma_S \subset \Sigma$, be the set of events which can be observed, then given a trace $s \in \Sigma^*$, the projection of s on Σ_S, denoted $s\uparrow\Sigma_S$, is the trace obtained by erasing the events not belonging to Σ_S from s. In such a trace the unobservable events are mapped to ε.

Non-determinism may be caused either by ε events or by one event at a state leading to two or more states. In such cases an equivalent deterministic version of the automaton can be obtained using the following steps modified from [10]:

Algorithm 1 *Removal of non-determinism*

<u>Step 1:</u> The initial state of the deterministic automaton corresponds to all the states which are reachable on ε events from the initial state of the non-deterministic automaton.

<u>Step 2:</u> Add a new state labeled Φ to the deterministic automaton. If there are no event transitions on a particular event leading out of a state in the non-deterministic automaton, in the deterministic version this is signified by such events leading to the Φ state.

<u>Step 3:</u> Consider each state in turn from among the set of states which form the new initial state in the deterministic automaton. Find the union of the states which can be reached by an event followed by an arbitrary number of ε events in the non-deterministic automaton. All the events in the plant are evaluated in turn, and events which do not have outgoing transitions from any of the set of states that constitute the initial state in the deterministic automaton are directed to the Φ state.

<u>Step 4:</u> Repeat Step 3 for all the new states added, and proceed until no further new states can be added. At most the number of states in the deterministic automaton will be number of states in the power set based on the number of states in the non-deterministic automaton.

Synchronous composition
When a complex discrete event system is modeled using sub-models, the overall behavior of the plant can be obtained using an interaction scheme termed as synchronous composition. Synchronous composition models the overall behavior of two interacting DESs which share events. Given two deterministic automata $G = (X, \Sigma_G, \delta_G, x_0, X_m)$, $X = (Y, \Sigma_S, \delta_S, y_0, Y_m)$, the synchronous composition [4] of G and S is denoted $R = G \parallel S = (Z, \Sigma, \delta, z0, Zm)$, where $Z = X \times Y$, $\Sigma = \Sigma G \cup \Sigma S$, $z_0 := (x_0, y_0)$, $Z_m := X_m \times Y_m$, and the transition function δ: $Z \times \Sigma \rightarrow Z$ is defined as: $\forall z = (x, y) \in Z, \sigma \in \Sigma$,

$$\delta((x,y), \sigma) = \begin{cases} (\delta_G(x, \sigma), \delta_S(y, \sigma)) & \\ \quad \text{if } \delta_G(x, \sigma), \delta_S(y, \sigma) \text{ defined} & \\ \quad \Sigma \in \Sigma_G \cap \Sigma_S & \\ (\delta_G(x, \sigma), y) & \\ \quad \text{if } \delta_G(x, \sigma) \text{ defined, } \sigma \in \Sigma_G - \Sigma_S & \\ (x, \delta_S(y, \sigma)) & \\ \quad \text{if } \delta_S(y, \sigma) \text{ defined, } \sigma \in \Sigma_S - \Sigma_G & \\ \text{undefined} \quad \text{otherwise} & \end{cases}$$

When G and S are composed through synchronous composition, the common events occur synchronously, while the other events occur asynchronously.

Supervisory Control of DESs
A supervisor, denoted S, may be defined as a map $S : L(G) \rightarrow 2^{\Sigma-\Sigma_S}$ that determines the set of events $Q(s) \subseteq (\Sigma\text{-}\Sigma_S)$ to be disabled after the occurrence of trace $s \in L(G)$.

Since restriction in the behavior of a plant can also be achieved by synchronous composition, the action of control may also be achieved by taking a synchronous composition of plant automaton G and supervisor automaton S.

When the DES being controlled includes uncertainty about the exact model of the system to use or if the design of the system is subject to frequent change, the computation of supervisors becomes increasingly complex to design. Extensions of supervisory control in such areas include robust and adaptive schemes [2, 7, 11]. Robust control attempts to supervise system behavior without resolving uncertainty about the model, and adaptive control proceeds by incorporates new information about the plant obtained by taking successive observations, thereby helping reduce uncertainty. This is particularly useful in cases when systems are modeled as sub-systems and if one of the sub-systems goes offline. Using an algorithm for learning in [3] the events related to the failed subsystem can be deleted and the original supervised system patched.

3. PLANT-SPCIFICATION RECONFIGURATION PROBLEM

Consider the overall plant automaton model of DES shown in Figure 1 obtained by the synchronous composition of three sub-models which are shown in Figure 2.

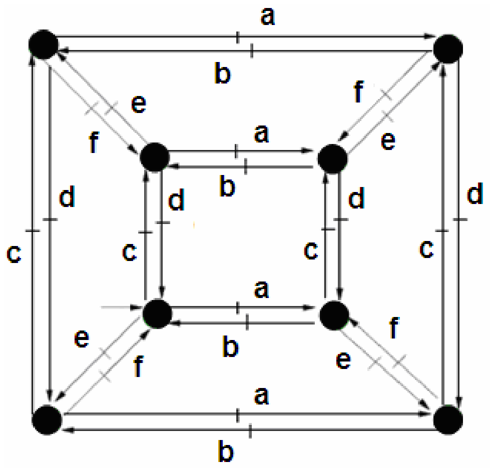

Figure 1: Automaton for overall plant model

The behavior of the plant is to be controlled using a sample specification given in Figure 3. The initial state of the automaton is indicated by the state which has a single ended arrow leading into it from no other state. Controllable event transitions are represented by marking them with small perpendicular dashes, indicating that these events can be pruned as needed by a supervisor synthesized for the plant using supervisory control theory. Filled double circles in the automaton represent goal or target states, which designate that certain tasks in the system have been performed.

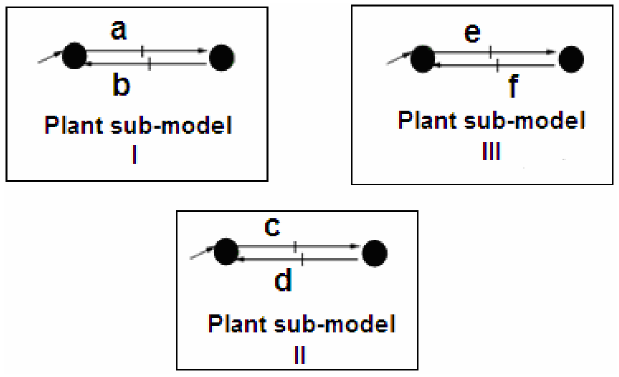

Figure 2: Automata for plant sub-models

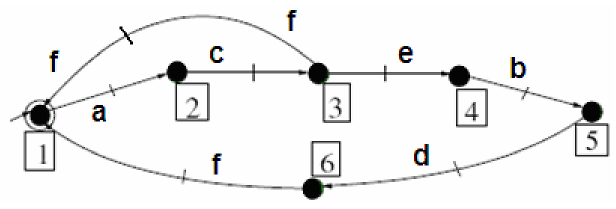

Figure 3: Automaton for control specification

As is often the case with modern manufacturing systems, different products are often produced using a few core pieces of equipment, aided by auxiliary equipment, specific to different product processing-line setups, which are frequently shuffled. The new system or specification after reconfiguration should contain only events belonging to the sub-models in use, else it might deadlock waiting for events which never occur, as would happen when the events are from modules that have been dropped. For example if sub-model II is dropped during reconfiguration based on this the events $\{c, d\}$ are simply deleted in the plant and control specification. In Figure 3 after executing the event a at state 1 and reaching state 2, the system would halt waiting for event c which cannot be executed. A method for connecting the events which have become disjointed in the specification after reconfiguration is needed. Regardless of the algorithm used to heal the specification, the automatically reconfigured specification should also be verified by the designer in the context of the control it will have on the behavior of the plant.

4. ADAPTIVE SUPERVISORY CONTROL

We provide algorithms for obtaining a reconfigured overall model of the plant and of the specification after sub-models which constitute either are dropped.

Algorithm 2 computes the reconfigured overall plant automaton.

Algorithm 2 *Computation of reconfigured plant automaton*

Step 1: In the initial overall plant model delete the events taken from the sub-model which has been dropped.

Step 2: Compute the states which are reachable from the initial state on the execution of events from existing sub-models, resulting in the new reconfigured overall plant automaton.

This algorithm is applied to obtain the reconfigured overall plant model shown in Figure 4. It is extracted from the initial system model shown in Figure 2 after dropping sub-model II and its associated events $\{c, d\}$.

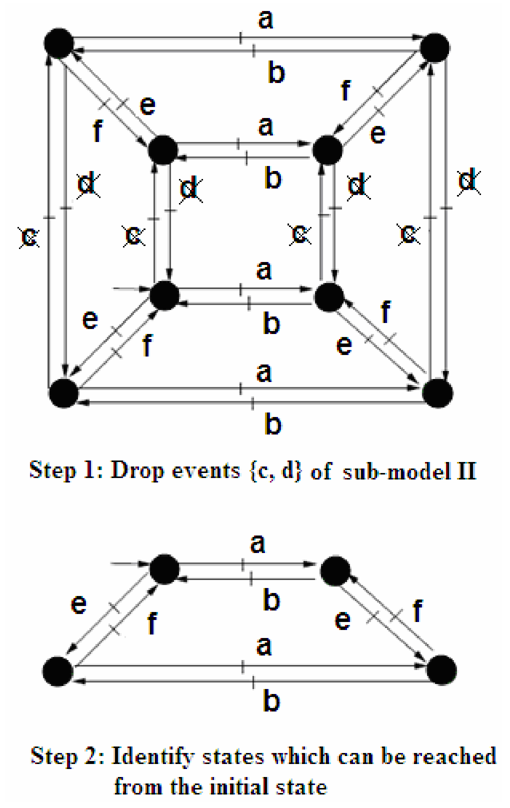
Step 1: Drop events {c, d} of sub-model II
Step 2: Identify states which can be reached from the initial state

Figure 4: Reconfigured overall plant model

Algorithm 3 computes the reconfigured control specification.

Algorithm 3 *Computation of reconfigured specification*

Step 1: Replace all the events taken from the sub-model which has been dropped with ε events. This step in general will result in a non-deterministic automaton.

Step 2: Remove the ε events using Algorithm 1. This step will make specification automaton a deterministic one.

Step 3: Since the specifications we are considering consists only of controllable events, all controllable event transitions to the Null or Φ state can be eliminated, resulting in the new reconfigured specification automaton.

This algorithm is applied to obtain the reconfigured control specification shown in Figure 5. It is extracted from the original control specifications shown in Figure 3, after replacing the events $\{c, d\}$ associated with sub-model II, with ε.

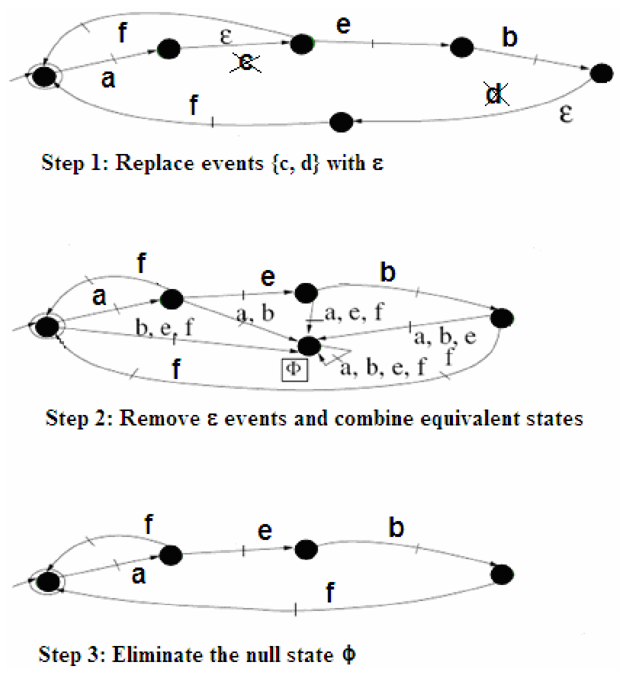
Step 1: Replace events {c, d} with ε
Step 2: Remove ε events and combine equivalent states
Step 3: Eliminate the null state Φ

Figure 5: New control specification

Using supervisory control algorithms, the supervised plant which meets these specifications is computed and is shown in Figure 6. It should be noted that the event f following the event trace $\{a.c\}$ in the original control specification has been deleted in the supervised plant. This is because it violates the sub-model behavior which requires that the event e be executed before event f (refer Figure 2).

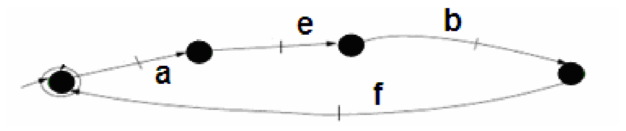

Figure 6: Reconfigured supervised plant

This adaptive method of dynamically incorporating information regarding reconfigurations in the plant can be used for computing an updated supervisor for the plant. The supervisor for the reconfigured plant obtained next using SCT is guaranteed to reach a target state in the supervised plant without violating any of the newly reconfigured control specifications.

5. CONCLUSIONS

In this paper we develop self-healing algorithms which readily adapt to changing configurations of a discrete event system. These algorithms can be used for obtaining a new overall model of the plant and control specification after reconfiguration. It is performed in a way which safely extracts events which are no longer part of the reconfigured system. The use of the algorithms is illustrated with an example drawn from a discrete event system composed of interacting sub-systems.

6. REFERENCES

[1] C. G. Cassandras, **Discrete Event Systems: Modeling and Performance Analysis**, Boston, MA: Aksen Associates, 1993.

[2] D. Gordon and K. Kiriakidis, "Adaptive supervisory control of interconnected discrete event systems", **IEEE Conference on Control Applications**, Anchorage, AK, Sept. 2000.

[3] D. Gordon and K. Kiriakidis, "Design of adaptive supervisors for discrete event systems via learning", **ASME Dynamic Systems and Control Division, International Mechanical Engineering Congress and Exposition**, Orlando, FL, Nov. 2000.

[4] C. A. R. Hoare, **Communicating Sequential Processes**, Englewood Cliffs, NJ: Prentice-Hall Pub., 1985.

[5] J. E. Hopcroft and J. D. Ullman, **Introduction to Automata Theory, Languages and Computation**, Reading, MA: Addison-Wesley, 1979.

[6] R. Kumar and V. K. Garg, **Modeling and Control of Logical Discrete Event Systems**, Boston, MA: Kluwer Academic Pub., 1995.

[7] F. Lin, "Robust and adaptive supervisory control of discrete event systems", **IEEE Transactions on Automatic Control**, Vol. 38, No. 12, 1993, pp. 1842-1852.

[8] P. J. Ramadge and W. M. Wonham, "On the supremal controllable sublanguage of a given language", **SIAM Journal of Control and Optimization**, Vol. 25, No. 3, 1987, pp. 637-659.

[9] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes", **SIAM Journal of Control and Optimization**, Vol. 25, No. 1, 1987, pp. 206-230.

[10] M. Sipser, **Introduction to the Theory of Computation**, Boston, MA: Brooks Cole, Inc., 1996.

[11] S. Young and V. K. Garg, "Model uncertainty in discrete event systems", **SIAM Journal of Control and Optimization**, Vol. 33, No. 1, 1995, pp. 208-226.