A Systematic Methodology for Actuator Augmentation in the Supervisory Control of Discrete Event Systems

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

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

and

Satya MOHANTY
Department of Computer Science and Engineering, University of California
Riverside, CA 92521, USA

ABSTRACT

Supervisory control theory of discrete event systems in the Ramade-Wonham paradigm addresses the problem of restricting the system evolution so that it conforms to certain predefined behavior commonly referred to as specifications. This theory states that a sequence of events that cause the plant to violate the specifications is suitably pruned or eliminated. However, in doing so, event traces, partial prefixes of which that actually meet control specification are eliminated as well. This happens for instance whenever a chain of uncontrollable event extensions render the plant behavior trajectory irrevocably outside the outlined specifications. Such partial conformance can be ensured if the capability of the system is augmented by additional actuators so that in the augmented plant there is a greater degree of control over uncontrollable events. It does not follow trivially where such actuators are to be placed. We propose an algorithm that enables us to identify states of the automaton where the new actuators need to be inserted, thus enlarging the scope of its applicability to system identification purposes as well.

Keywords: Actuator augmentation, Automata, Modeling, Supervisory Control Theory, Discrete Event Systems, Industrial Control.

1. INTRODUCTION

A Discrete Event System (DES) in the Supervisory Control Theory (SCT) framework of Ramadge and Wonham (RW) [4], is a dynamic system modeled as an automaton, which evolves from one state to the next via abrupt event transitions.

Besides the problem of handling a large state-space common to automata based SCT formalisms, one of the main challenges faced by system designers in the RW framework is the difficulty in identifying the precise locations in the system which is causing the specifications to become uncontrollable. Modeling and subsequent supervisor construction issues of DESs have been studied extensively in the literature [2, 3, 1]. However the issue of adding actuators in order to gain better control over the operation of a plant has not been addressed in the SCT framework. Whenever a certain combination of controllable (actuator) actuator at a certain state in the system causes an undesired sequence of uncontrollable (sensor) events to occur, the state from which the uncontrollable event trace originates is eliminated since the specification is uncontrollable. In this paper we will show a systematic way of identifying the precise states where the system would most benefit by the addition of an actuator such that any partial uncontrollable event traces can be achieved successfully.

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. Section 3 illustrates the problems encountered by a supervisor trying to control plant behavior, when under a certain configuration of controllable events a series of cascaded uncontrollable events can occur, making control over a partial subset of the uncontrollable events impossible. In Section 4 we provide an algorithm for adding actuators which will break the sequence of uncontrollable events, thereby granting more precise control to the supervisor for the plant. In Section 5, the sensor augmentation algorithm, and subsequent supervisor construction procedure is illustrated using an example drawn from process control. Section 6 provides conclusions of the work, and directions for future research. Finally, Section 7 contains bibliographic references.

2. SUPERVISORY CONTROL OF DESs

Language Models and Automata

In order to express the behavior of a system, a language model for it can be created by using the events in the system. The finite set of events is denoted by Σ, and different sequences of these events forms a *string* of events, also called a *trace*. A *language* is defined as a set of traces. Σ^* represents the set of all finite traces of events in Σ including the empty trace ε. 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.

The events occurring in a system can be partitioned into two sets Σ_c, the set of *controllable* events, and Σ_u, the set of *uncontrollable* events, depending on whether they can or cannot be disabled by an external agent (called a supervisor). The set Σ, can thus be represented as: $\Sigma = \Sigma_c \cup \Sigma_u$.

Automata [5] can be used for representing the untimed behavior of language models and of the qualitative pecifications needed for its control. Formally, an automaton G is represented as a 5-tuple $G = (X, \Sigma, \delta, x_0, X_m)$, $\delta: X \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^X$ is the partial state information function, and $x_0 \in X$ is the initial state, and X_m denotes the set of marked state. Starting from a single initial state the system evolves on the occurrence of events permitted at each state by the transition function.

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.

The *generated language* of G is the set of all traces that it can execute starting from its initial state, i.e., $L(G) = \{s \in \Sigma^* \mid \delta(x_0, s) \text{ is defined}\}$. The *marked language* of automaton G contains those generated traces which signify task completion, i.e., the set of marked states X_m. $L_m(G) = \{s \in L(G) \mid \delta(x_0, s) \in X_m\}$. For a language K, the notation $\overline{K}$, called the prefix closure of K, is the set of all prefixes of traces in K. K is said to be prefix closed if $K = \overline{K}$. K is said to be relatively closed or $L_m(G)$*–closed* if $K = \overline{K} \cap L_m(G)$. A proper prefix t of a trace s is denoted as $t < s$. For any language K, the notation $K\backslash s$, where $s \in \overline{K}$, denotes the set $\{t \in \Sigma^* \mid st \in K\}$, i.e., the set of extensions t of s such that $st \in$ K.

The $Re\{\Sigma_u\}(StateLabel)$ or *uncontrollable reachability* of any state in an automaton, which might be representing either the plant or the control specification, is defined as the set of states which can be reached by the execution of only uncontrollable events. In a similar way the $Re\{\Sigma_c\}(StateLabel)$ or *controllable reachability* of any state in an automaton can be defined as well.

Supervisory Control Theory

Given a language K, a plant G with a generated language $L(G)$ over a set of events Σ, the controllability of K with respect to $L(G)$ requires that the extension of any prefix of K by an uncontrollable event that is feasible in the plant should also be a prefix of K. If K is controllable and relative-closed with respect to $L(G)$ then the automata representing $\overline{K}$ is the required supervisor, but if this is not the case, then one computes the language $K^{\uparrow}$, the *supremal controllable and relatively closed sublanguage* of K with respect to G [4].

3. THE CASCADED UNCONTROLLABLE EVENT OCCURRENCE PROBLEM

Consider the DES shown in Figure 1, having as controllable events $\{a\}$ and as uncontrollable events $\{y, z\}$. The overall language $L(G)$ of the plant is given by the traces within the outer closed curve in Figure 1(i) and which originate at X0, the *initial state*. The initial specification language $K = L(S)$ is given by the traces within the inner closed curve in Figure 1(i) which originate at *X0*. K is to be tested for controllability using SCT, and if necessary modified such that the remaining language K is controllable. As stated before, the uncontrollable events occurring at any state cannot be prevented from occurring by a supervisor, which can disable only controllable events. So for example y cannot be disabled at state *X1*.

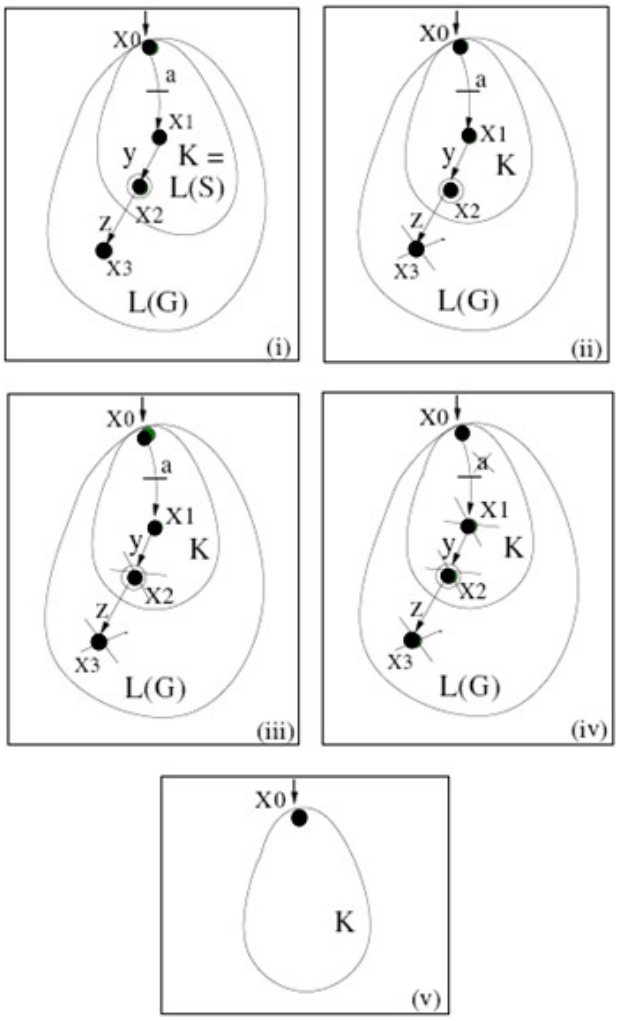

Figure 1: Limitations in supervisory control theory

The *marked states* of the specification, representing completion of some tasks in the plant, are indicated by double circles, and it is required that that the supervised plant always either stop in a marked state or be able to reach one through a finite sequence of events, i.e., it be *non-blocking*.

The desired controlled language is specified to be $\{a.y\}$, a trace in the subset of the plant language. It ends at state *X2*. It can be seen in Figure 1(i) that after the supervised plant executes the controllable event a, it must execute the uncontrollable event y. In order for the controlled system to stay non-blocking after the y event, the event z should be *blocked*. This is because after the plant executes z it can no longer reach a marked state. However z cannot be disabled as it is an uncontrollable event.

In Figure 1(ii) using SCT we verify and as needed modify the supervised language so that it is controllable. During this process we will mark all states which are not within the specification as *bad* states, in particular *X3* reached on the uncontrollable event *z*. This state is indicated by the cross drawn across it.

In Figure 1(iii) and (iv) any state which is itself connected to a bad state by uncontrollable event transitions is itself marked as a *bad* state. Further, any outgoing controllable event leading to a *bad* state are disabled whenever such controllable event exist. Hence transition *a* is crossed out as well. Outgoing controllable events from a *bad* state are also disabled, whenever such transitions exist.

Finally in Figure 1(v) only one state, the initial on is left in the system which is not marked as bad. Hence the overall supervised language of the plant is Φ.

This is unfortunate, since even though the plant has the capability to perform several sub-traces which are within the specification, for example the trace {*a, a.y*} but the supervisor disables every event in the plant owing to controllability and non-blocking constraints. This clearly indicates the need for modifying the plant behavior, by adding controllable and/or uncontrollable events in such a way that at least partial traces can be successfully completed in the supervised plant.

4. ACTUATOR AUGMENTATION AND SUPERVISORY CONTROL

Algorithm 1 explained next and also shown in Figure 2, computes the potential list of states where actuator addition would lead to execution of sub-traces of uncontrollable events thereby granting a better degree of control over the overall system behavior. Only an automaton based model of the plant is required for this computation, and essentially it proceeds by identifying those states in the plant from where a cascaded set of uncontrollable events originate. It is at the states which lead to such states from controllable event transitions where we can attempt to add additional actuators.

Algorithm 1 *Identification of states requiring actuator event augmentation*

Step 1: Create a list Plant States List or PSL containing all the states of the plant automaton, with the initial state at the head of the PSL.

Step 2: Starting from the initial state of the PSL sequentially compute the set of states which are in its 1-step controllable event reachability, i.e., those states which are reachable from the initial state by the execution of only controllable events taken one at a time; and record them in a list denoted the $\mathrm{Re}_{\Sigma}^{1-c}$ (*StateLabel*) *List*.

If at the initial state, either no event is possible, or only uncontrollable events are possible, or a mixture of both controllable and uncontrollable events are possible, place a Φ in the $\mathrm{Re}_{\Sigma}^{1-c}$ (*StateLabel*) *List*, corresponding to the initial state in the *PSL*. Similarly process all the states in *PSL*.

1. $G := (Q, \Sigma, \delta, q_0, Q_m)$ is plant automaton:
   $\forall q \in Q$ in G/S do:
   $Re_{\Sigma}^{1-c}(q) := \phi,\ L := \phi$
2. $\forall q \in Q$ in G do:
   If $\delta(q, \Sigma) \neq \phi\ \wedge\ \delta(q, \Sigma_u)$ is not defined
   $Re_{\Sigma}^{1-c}(q) := \delta(q, \Sigma)$
   else
   $Re_{\Sigma}^{1-c}(q) := \phi$
3. $\forall q \in Q\ s.t\ Re_{\Sigma}^{1-c}(q) \neq \phi$ do:
   $\forall v \in Re_{\Sigma}^{1-c}(q)$ do:
   If $|Re_{\Sigma_u^*}(v)| \leq 1$
   $Re_{\Sigma}^{1-c}(q) := Re_{\Sigma}^{1-c}(q) - v;$
   else
   If $q \notin L,\ L := L \cup q$
4. L is the list of all states where actuator events can be introduced.

Figure 2: Algorithm for event augmentation state identification

Step 3: Perform an uncontrollable event reachability analysis for each of the states that are in the $\mathrm{Re}_{\Sigma}^{1-c}$ (*StateLabel*) *List*, and if the number of states reached is less than two, remove that state from the $\mathrm{Re}_{\Sigma}^{1-c}$ (*StateLabel*) *List*.

The uncontrollable event analysis is used to identify states which are reachable from any state on the execution of only uncontrollable events. The requirement that there be at least two different states in the uncontrollable event reachability ensures that either there is a cascaded sequence of uncontrollable events originating from that state, or there are multiple uncontrollable events possible at that state.

Step 4: Identify the entries which have at least 2 states in the revised $\mathrm{Re}_{\Sigma}^{1-c}$ (*StateLabel*) *List*. Perform a reverse lookup for these entries under the *PSL*, to obtain the set of states which will benefit most by the addition of new actuator events, since they will be introduced at location of the plant which are 1 controllable event away from a potentially cascaded sequence of uncontrollable events.

5. INDUSTRIAL APPLICATION: ACTUATOR AUGMENTATION IN A PROCESS CONTROL SYSTEM

The functioning algorithm is illustrated through an example from a process control. The schematic of a tank is shown in Figure 3. This system has the following sensors and tap connected:

- A tap, *t1*, for filling, with the events corresponding to it being open/closed denoted by *t1on/t1off* respectively.
- A magnetic nominal level sensor, *n*, for sensing fluid height in tank, with the events for measuring changes in the nominal level signal denoted by *nup*, *ndn*.

- A magnetic high level sensor, *n*, for sensing fluid height in tank, with the events for measuring changes in the high level signal denoted by *hup*, *hdn*.

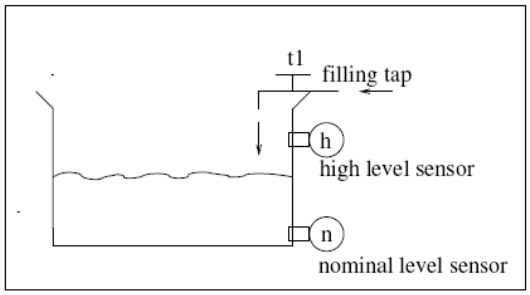

Figure 3: Tank system

Assuming that initially the tank is empty and the filling tap is off the initial state of the system is represented by the actuator-sensor events {*t1off, ndn, hdn*}. This is shown as state *X0* in Figure 4(i). The only event possible at this state is that of switching the filling tap on, *t1on*, which will eventually cause the nominal level sensor to indicate that the nominal level has been reached, *nup*. If filling is continued, then eventually the high level sensor will be reached as well, *hup*. When the tap *t1* is on, it can be switched off at will and vice versa. The automaton representing the behavior of the plant is shown in Figure 4(i).

The control specification for the plant is shown in Figure 4(ii), and the similarity it has with that of Figure 1 discussed in Section 3 can be observed. It is required that only partial filling of the tank be carried out, i.e., after the *nup* event occurs the system should stop at state *X3*.

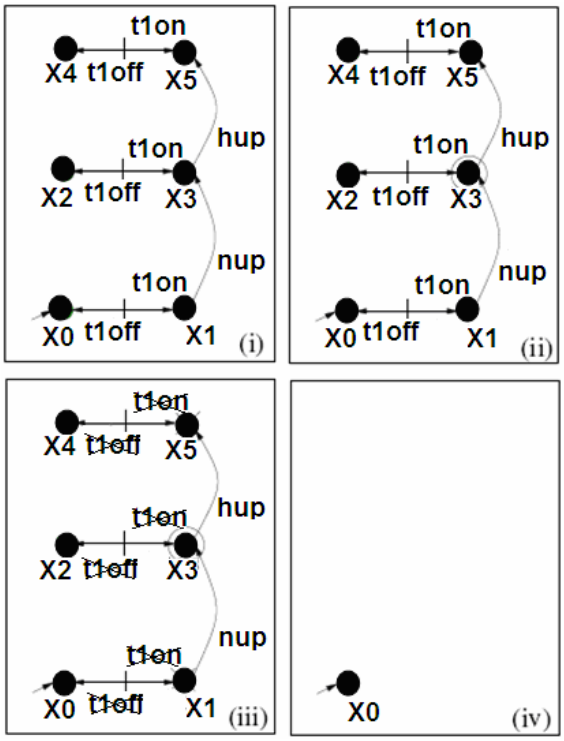

Figure 4: Supervisor construction for tank system

Based on the control specification, while it is possible to reach *X3,* the partially filled tank state, supervisory control constraints prohibit disablement of uncontrollable event at any state, and thus cannot stop the *hup* event at state *X3* from occurring. Following, a similar line of reasoning as discussed earlier for Figure 1, a supervisor for the system based on the control specification is computed in steps Figure 4(iii) and (iv) yielding Φ.

Clearly, this is not acceptable, and in order to achieve partial filling of the tank additional actuators (taps) will need to be added to make just nominal filling possible. Use is made of Algorithm 1 for this.

Refer to the different steps of Algorithm 1 for determining the states in Figure 4 which will benefit from actuator addition. The results at each step of the algorithm are shown in Table 1.

Step 1	Step 2	Step 3	Step 4
X0	X1	X3, X5	X3, X5
X1	Φ	Φ	Φ
X2	X3	Φ	Φ
X3	Φ	Φ	Φ
X4	X5	Φ	Φ
X5	X4	Φ	Φ

Table 1: Determining states for actuator addition

Based on this analysis it can be observed that an actuator will need to be added at the initial state *X0*, and enabled, after which only filling should be initiated. This implies a draining tap *t2* positioned between the nominal and high level sensors as shown in Figure 5.

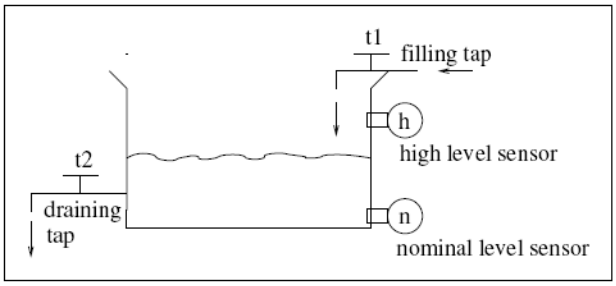

Figure 5: Modified tank with draining tap added

The tap, *t2*, for draining, has the events corresponding to it being open/closed by the events *t2on/t2off* respectively. The rate of filling the tank is assumed to be equal to that of draining.

The automaton model of the plant, with the tap *t2* in place, is shown in Figure 6(i). The initial state is when all the actuators are (*t1off, t2off*) and the tank is completely drained (*ndn*). Starting from the initial state, *X0*, all feasible trajectories of the system are shown in the twelve state automaton model of the plant.

The filling operation in the tank is regulated by its control specifications, which requires partial filling of the tank, until the nominal level is sensor n is reached. This is indicated by the

double circle drawn around the states $\{B\}$ as shown in Figure 6(ii). As part of the computation of the supervisor for the plant an additional state is added to the specification designated as state $\{D\}$ in Figure 6(iii). Events which are not already part of the specification lead into the state $\{D\}$, and the self-loop on all events taken around this state indicate once the plant reaches the state $\{D\}$ it stays there. This step is needed to track deviations of plant behavior from that required by the control specifications.

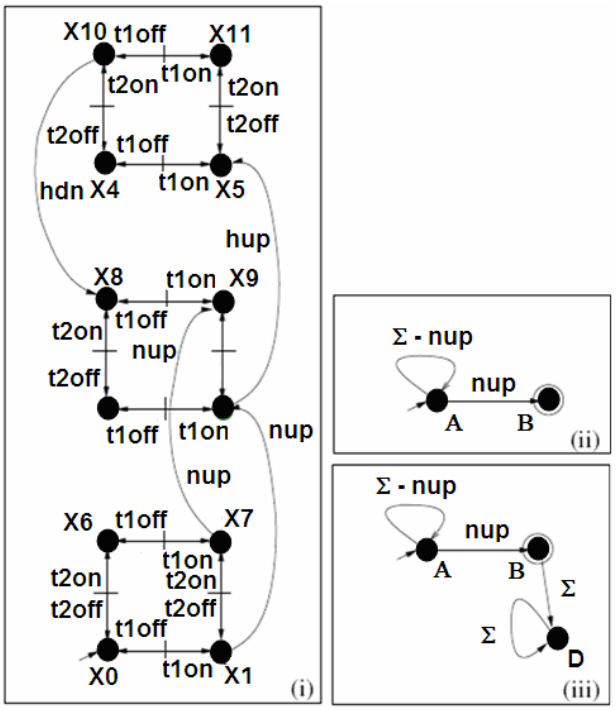

Figure 6: Modified plant model and control specifications

The supervised plant which meets this specification is computed in Figure 7(i) by taking a synchronous composition of the plant shown in Figure 6(i) with the augmented specification shown in Figure 6(iii). The resultant supervisor is shown in Figure 7(ii), after deleting the states which are not reachable from the initial state *X0A* of the supervised plant.

Finally, in Figure 8, a controller is extracted from the supervisor by enabling at most one control at each state. This controller is capable of carrying out partial filling operations in the tank system, and completely meets the control specifications

6. CONCLUSIONS

The design of a supervisor for a discrete event system can be rendered extremely difficult when cascaded uncontrollable events (those not under the direct control of a supervisor, such as sensor events) are possible in the system, but only a subtrace of the uncontrollable events is part of the control specification. In an untimed model of a DES it is not possible for a supervisor to break a trace containing cascaded uncontrollable event. Under such circumstances, the number of actuator events will need to be expanded at locations where the uncontrollable event sequences can indeed be broken, thereby providing precise control over the uncontrollable events in the system. We provide a algorithm for isolating the precise locations where actuator events will need to be added. The usefulness of actuator augmentation algorithm in identifying the states in a sample process control plant which require actuator augmentation so that partial uncontrollable event traces can be controller has been demonstrated.

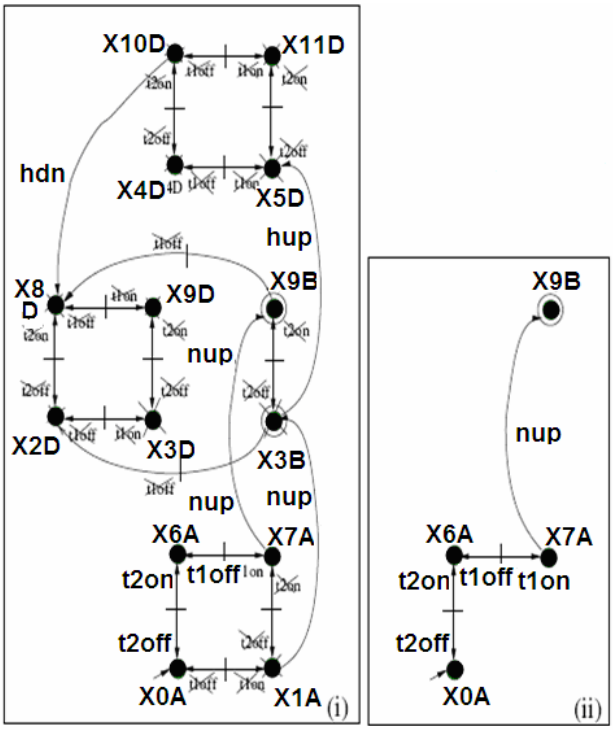

Figure 7: Supervisor construction using SCT

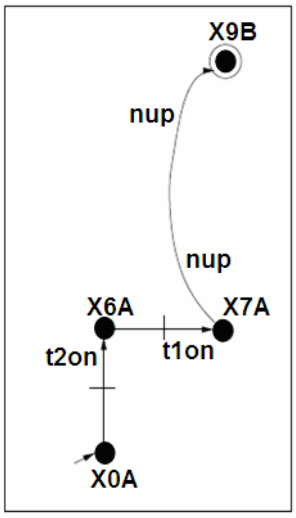

Figure 8: Final supervisor and controller

7. REFERENCES

[1] R. Alur and D. Dill, "Automata for modeling realtime systems", **International Colloquium on Automata,**

Languages and Programming, pp. 322-335, Warwick, UK, 1990.

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

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

[4] 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.

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