Nonblocking and Safe Control of Discrete-Event Systems Modeled as Extended Finite Automata

Nonblocking and Safe Control of Discrete Event Systems modeled as Extended Finite Automata

Lucien Ouedraogo, *Member, IEEE* Ratnesh Kumar, *Fellow, IEEE* Robi Malik and Knut Åkesson

Abstract—**Extended Finite Automata (EFA), i.e., finite automata extended with variables, are a suitable modeling framework for discrete event systems owing to their compactness, resulting from the use of variables. In this paper, we propose a symbolic algorithm that efficiently synthesizes a supervisor for a plant modeled by an EFA and a specification defined by another EFA. The principle of the algorithm is to iteratively strengthen the guards of the plant EFA so that forbidden or blocking states become unreachable in the controlled plant. As a consequence of the algorithm, the controlled behavior is modeled by an EFA having the same structure as the plant EFA, having stronger guards and is shown to be maximally permissive. We illustrate our algorithm via a simple manufacturing example.**

Note to Practitioners: A compact way of modeling event-driven systems is to use state-variables, instead of an explicit enumeration of the states. This paper uses such a model for representing the system to be controlled as well as its desired behaviors, and develops a symbolic approach, that avoids explicit enumeration of the state-space, for control synthesis. The contribution is the symbolic computation of a safe (avoids reaching forbidden states) and nonblocking (avoids getting blocked at non final states) controller that is also maximal (permits all safe and nonblocking behaviors). The results are illustrated via a simple manufacturing system.

Index Terms—**Discrete event systems, Extended finite automata, Supervisory control.**

I. Introduction

IT is well known that automata-based approaches to discrete-event control suffer from state-space explosion. Prior works addressing this issue of complexity includes [1], where a controller is synthesized based on progressively finer abstractions of the plant. Another approach is to employ binary decision diagram (BDD) representation [2], [3].

The extended finite automata (EFA) framework, obtained by augmenting a standard finite state automaton (FSA) with variables and predicates over them [4]–[8], provides a compact representation of a DES. In this paper, we propose a symbolic approach for synthesizing the most permissive nonblocking and safe supervisor for DES modeled by EFA with data variables of finite domains. Our approach resolves some limitations of the existing approaches and is efficient in exploiting the model structure due to the symbolic representation and symbolic computations (ie., over sets of states, rather states). Moreover, our algorithm leads to more efficient representation of controllers (symbolic representation instead of state-transitions representation) and the symbolic computation of guards and predicates, that are boolean operations, can be efficiently implemented by BDDs [9].

Supervisory control methods that use the EFA framework are proposed in [7], [8], [10]–[15]. The method of [7] does not preserve the structure of the plant EFA in control computation, and does not consider blocking issues or nondeterminism. [8], [10], [15] propose methods for representing a supervisor synthesized in the FSA modeling framework by EFA. In [11], the supervisory control problem for EFA is solved by transforming the EFA into ordinary FSA, and [12] proposes a method for converting EFA into the model of the Symbolic Model Verification tool NuSMV, from which supervisory control properties can be verified. The contributions of [8], [10]–[12], [15] are therefore different from ours, as instead of using a FSA-based synthesis algorithm, we propose an EFA-based supervisor synthesis algorithm that exploits the EFA model compactness and leads to a reduced complexity as the synthesis is carried out over guards, ie., over sets of states, rather than over states. The control method of [13], [14] is also based on abstraction and hence not necessarily maximally permissive, and also doesn't consider blocking issues, while it requires the exploration of the entire state space of the plant EFA to determine states co-reachability; which our method avoids. [16] uses a similar approach for computing a supervisor for infinite state systems.

Our algorithm synthesizes a supervisor by associating new stronger guard conditions to the transitions of the plant EFA. The work reported here is based on the conference version [17], extended to include the complete proofs and more detailed examples and discussions.

The rest of the paper is organized as follows. In Section II, we give formal definitions related to the EFA model. In Section III, we state formally the supervisory control problem for EFA, and in Section IV, we present our synthesis algorithm and demonstrate its correctness and maximal permissiveness. In Section V, we give an example of our synthesis method, and Section VI contains our concluding remarks.

The research was supported in part by the National Science Foundation under the grants NSF-ECS-0601570, NSF-ECCS-0801763, NSF-CCF-0811541, and NSF-ECCS-0926029 and by the Fond québecois de la recherche sur la nature et les technologies.

L. Ouedraogo and R. Kumar are with the Dept. of Elect. & Comp. Eng., Iowa State University, Ames, IA, 50011, USA, (olucien, rkumar)@iastate.edu. R. Malik is with the Dept. of Comp. Science, The University of Waikato, Hamilton, New Zealand robi@cs.waikato.ac.nz. K. Åkesson is with the Dept. of Signals and Syst., Chalmers University of Technology, Göteborg, Sweden knut@chalmers.se

The authors would like to acknowledge Martin Fabian and Bengt Lennartson of Chalmers University of Technology for hosting Profs. Kumar and Malik at Chalmers where the initial research took shape, and also for their participation in the initial discussions.

II. Preliminaries

A. Predicates and notations

FSAs are extended with data variables to obtain EFAs. Let $D = D_1 \times \ldots \times D_i \times \ldots \times D_p$ be the domain of definition of p *one-dimensional* data variables. We use d to denote a *p-dimensional* variable (vector) of domain D, i.e. $d = [d(1), \ldots, d(i), \ldots, d(p)]$, where $d(i)$ is the ith data variable of domain D_i. We use *predicates* for describing various sets of elements of D. Let $\mathcal{G}(d)$ denote the collection of predicates defined using the data variable vector d, i.e., if $g \in \mathcal{G}(d)$, then it is a boolean valued map $g : D \to \{false, true\}$. $g \in \mathcal{G}(d)$ can also be seen as a subset of D, i.e. $g \subseteq D$ is the set of values $d \in D$ for which $g(d) = true$. We use the notations T for $true$ and F for $false$. Given a predicate $g \in \mathcal{G}(d)$, its *negation* is denoted by $\neg g$. Given an indexing set I such that $g_i \in \mathcal{G}(d)$ for each $i \in$ I, the *conjunction* and *disjunction* over I are denoted by $\bigwedge_{i \in \mathrm{I}} g_i$ and $\bigvee_{i \in \mathrm{I}} g_i$ respectively (see [18] for more detailed discussions and results on predicates).

B. Extended Finite Automaton

Extended Finite Automata (EFA) constitute a modeling framework which allows symbolic description of discrete event systems in the form of automata. An EFA can be seen as a finite state automaton (FSA) incorporating data variables defined over finite or infinite domains. The transitions of an EFA are augmented by guard formulas, which are predicates defined over the data variables, and data update functions, which are actions on the data variables. An EFA is formally defined as follows.

Definition 1 (Extended Finite Automaton)
An *Extended Finite Automaton* is a 7-tuple $A = (L, D, \Sigma, E, L_0, D_0, L_m)$ where: L is a finite set of locations; $D = D_1 \times \cdots \times D_p$ is a domain of p one-dimensional data variables; Σ is a finite set of events; $L_0 \subseteq L$ is a set of initial locations; $D_0 = D_0^1 \times \ldots \times D_0^p$ is a set of initial data values; $L_m \subseteq L$ is a set of marked (final) locations; and E is a finite set of edges (or transitions), each edge $e \in E$ being a 5-tuple $e = (o_e, t_e, \sigma_e, g_e, f_e)$ where:

- $o_e \in L$ is the origin location of e;
- $t_e \in L$ is the terminal location of e;
- $\sigma_e \in \Sigma$ is the transition label;
- $g_e \subseteq D$ is the enabling guard of e;
- $f_e : D \to D$ is the data update function. □

A transition $e = (o_e, t_e, \sigma_e, g_e, f_e)$ is *enabled* if the current location is o_e and guard condition g_e is evaluated to *true*. An enabled transition can be executed to update the current location as well as current data values. When the transition e is executed, location t_e is reached and the variables are updated by applying f_e to them.

Given two guards g and h, we say that g is a subguard of h, denoted $g \preceq h$, if g is stronger than h, namely, $g \wedge h = g$. Given two EFAs A and A', we say that A' is a subautomaton of A, denoted $A' \preceq A$, if A' is obtained from A by removing some locations of A as well as the transitions linked to these locations and/or removing some transitions of A and/or replacing the guards of some edges of A by subguards. When

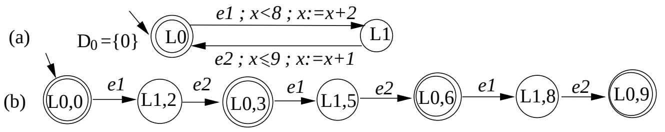


Figure 1. An EFA (top) and its equivalent FSA (bottom)

a transition e is executed resulting in "post-condition" $h(d)$ for the data, then the following "pre-condition" must be satisfied by the data prior to the execution of e: $g_e(d) \wedge h(f_e(d))$.

Figure 1(a) illustrates an example of an EFA with two locations L0 and L1 and two edges, where the labels of edges are in the form $\sigma_e; g_e; f_e$. For this example, the data variable vector d consists of a single variable x and we consider that $D = D_1 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ (domain of x). L0 is the only initial (indicated on figure by an arrow) and marked (indicated on figure by double circles) location, and the initial value of the variable x is 0, i.e., $D_0 = \{0\}$. The transition from location L0 to location L1 can be executed only if the guard $x < 8$ is evaluated to $true$, and after this transition is executed, the value of x is updated so that its new value is equal to its current value plus two: $x := x + 2$. Note that if $h(x) = [x > 6]$ holds after the execution of $e1$, then it must be the case that prior to the execution of $e1$, $[x < 8] \wedge [(x + 2) > 6]$ holds, where $[x < 8] = g_{e1}(x)$ and $[(x + 2) > 6] = h(f_{e1}(x)) = f_{e1}(x) > 6$.

The state of an EFA consists of its current location (as in an FSA) and its current value of data-variables, and so the set of states of A is given by $L \times D$. Given an EFA A with data variables of finite domains, we can construct an equivalent FSA by representing all its reachable states in $L \times D$. Figure 1(b) represents the equivalent FSA of the EFA of Figure 1(a). The location and the value of x are indicated in every states.

In the sequel, Σ^* denotes the set of all finite strings of the form $\sigma_1\sigma_2...\sigma_n$ of events from Σ, including the empty string ϵ. Let q_0 denote an initial state $(l_0, d_0) \in L_0 \times D_0$, Q^A the set of reachable states of A and $Q^A_m \subseteq Q^A$ the set of reachable marked states of A (states in $L_m \times D$). For a state $q = (l, d) \in Q^A$, $q \xrightarrow{\sigma} q'$ denotes that there exists an edge $e = (l, l', \sigma, g_e, f_e) \in E$ such that $g_e(d) = T$, $f_e(d) = d'$ and $q' = (l', d')$; and $q \xrightarrow{\sigma}$ denotes that there exists at least one state $q' = (l', d')$ such that $q \xrightarrow{\sigma} q'$. This notation is extended to every $s \in \Sigma^*$ as follows: $q \xrightarrow{\epsilon} q$ for every $q \in Q^A$ and $q \xrightarrow{s\sigma} q'$ if $q \xrightarrow{s} \bar{q}$ and $\bar{q} \xrightarrow{\sigma} q'$ for some $\bar{q} \in Q^A$. A run of A is a finite sequence $r = (l_0, d_0) \xrightarrow{\sigma_1} (l_1, d_1) \xrightarrow{\sigma_2} \ldots \xrightarrow{\sigma_n} (l_n, d_n)$ where $l_0 \in L_0$, $d_0 \in D_0$, $l_i \in L$, $d_i \in D$ and $\sigma_i \in \Sigma$, for $i = 1, 2, ..., n$. r is accepted by A if in every state (l_i, d_i), $g_{e_i}(d_i) = T$ and $d_{i+1} = f_{e_i}(d_i)$, where $e_i = (l_{i-1}, l_i, \sigma_i, g_{e_i}, f_{e_i}) \in E$. A state $q = (l, d)$ of A is said to be reachable if there exists a sequence s and an initial state q_0 such that $q_0 \xrightarrow{s} q$. Given a location $l \in L$, $\Sigma^A(l)$ denotes the set of events of the outgoing edges of location l of A. On the other hand, $\Sigma^A(q)$ denotes the set of events enabled at state $q \in Q^A$, i.e. $\Sigma^A(q) = \{\sigma \in \Sigma | q \xrightarrow{\sigma}\}$. In the same way, $Q^A(q)$ denotes the set of states reached from q through events in $\Sigma^A(q)$, i.e. $Q^A(q) = \{q' \in Q^A, \exists \sigma \in \Sigma^A(q) | q \xrightarrow{\sigma} q'\}$. In particular, for a $\sigma \in \Sigma^A(q)$, $Q^A(q, \sigma)$ denotes the set of states reached after the execution of σ from q, i.e. $Q^A(q, \sigma) = \{q' \in Q^A | q \xrightarrow{\sigma} q'\}$.

C. Parallel composition of EFA

In general a system can consist of multiple subsystems, each modeled as an EFA. Then their parallel composition, as defined below, can be used to obtain the EFA model of the entire system. For the parallel composition of two EFA to exist, they must have the same initial data values for all shared variables. For an update function f and two domains $D_1 = D'_1 \times D_s$ and $D_2 = D_s \times D'_2$, where D'_1, D_s and D'_2 are subdomains (D_s is shared by D_1 and D_2), let $D_1 \otimes D_2 = D'_1 \times D_s \times D'_2$. For an update function $f_i : D_i \to D_i$ ($i = 1, 2$), let $f_i|_{D_s}$ denote the projection of f_i on D_s. In the following we define the parallel composition of two EFA, in which the function composition $f_1 \oplus f_2 : D_1 \otimes D_2 \to D_1 \otimes D_2$ is defined for the functions $f_1 : D_1 \to D_1$ and $f_2 : D_2 \to D_2$ that map the shared data variables in D_s identically as either of the functions map, whereas it maps the non-shared data variables according to the functions whose domain they belong. I.e., $f_1 \oplus f_2 = f_1 \times f_2|_{D'_2} = f_1|_{D'_1} \times f_2$.

Definition 2 (Parallel composition of EFA)
Let $A_k = (L_k, D_k, \Sigma_k, E_k, L_0^k, D_0^k, L_m^k)$, $k = 1, 2$ be two EFA. The parallel composition of A_1 and A_2 is $A_1 \parallel A_2 = (L_1 \times L_2, D_1 \otimes D_2, \Sigma_1 \cup \Sigma_2, E, L_0^1 \times L_0^2, D_0^1 \otimes D_0^2, L_m^1 \times L_m^2)$ where the set of edges E is defined as follows:

- $\forall \sigma \in \Sigma_1 \cap \Sigma_2$, $\forall (l_1, l'_1, \sigma, g_1, f_1) \in E_1$, $\forall (l_2, l'_2, \sigma, g_2, f_2) \in E_2$, we have $((l_1, l_2), (l'_1, l'_2), \sigma, g_1 \wedge g_2 \wedge [f_1|_{D_s} = f_2|_{D_s}], f_1 \oplus f_2) \in E$.
- $\forall \sigma \in \Sigma_1 \setminus \Sigma_2$, $\forall (l_1, l'_1, \sigma, g_1, f_1) \in E_1$ we have $\forall l_2 \in L_2$, $((l_1, l_2), (l'_1, l_2), \sigma, g_1, f_1) \in E$.
- $\forall \sigma \in \Sigma_2 \setminus \Sigma_1$, $\forall (l_2, l'_2, \sigma, g_2, f_2) \in E_2$ we have $\forall l_1 \in L_1$, $((l_1, l_2), (l_1, l'_2, \sigma, g_2, f_2)) \in E$. □

Following Definition 2, an event can occur in the synchronized EFA if and only if it can occur in all EFA that share the event and all occurrences of the event involved in this synchronization update the data variables consistently.

III. Supervisory control of EFA

In general, the plant is given by an EFA $P = (L^P, D, \Sigma, E^P, L_0^P, D_0, L_m^P)$ and the specification by another EFA $R = (L^R, D, \Sigma, E^R, L_0^R, D_0, L_m^R)$. By refining P with respect to R we can obtain a refined plant model G with the same behaviors as P such that the executions not allowed in R end up in certain forbidden locations in G. The refined EFA $G = (L, D, \Sigma, E, L_0, D_0, L_m)$ is constructed as follows: $L_0 = L_0^P \times L_0^R$; $L = L^P \times (L^R \cup \{\phi\})$ (ϕ = forbidden location); $L_m = L_m^P \times L_m^R$; and E constructed as follows:

- $\forall e \in E^P, \forall l \in L^R \cup \{\phi\}, \forall e' \in E^R$ s.t. $(o_{e'} = l) \wedge (\sigma_{e'} = \sigma_e) : ((o_e, l), (t_e, t_{e'}), \sigma_e, g_e \wedge g_{e'} \wedge [f_e = f_{e'}], f_e) \in E$, $((o_e, l), (t_e, \phi), \sigma_e, g_e \wedge \neg[\bigvee_{\bar{e} \in E^R : o_{\bar{e}} = o_{e'}, \sigma_{\bar{e}} = \sigma_{e'}} g_{\bar{e}} \wedge [f_{\bar{e}} = f_e]], f_e) \in E$; and
- $\forall e \in E^P, \forall l \in L^R \cup \{\phi\}, \nexists e' \in E^R$ s.t. $(o_{e'} = l) \wedge (\sigma_{e'} = \sigma_e) : ((o_e, l), (t_e, \phi), \sigma_e, g_e, f_e) \in E$.

Figure 2 illustrates an example of refinement, where only locations reachable from an initial location are represented, and forbidden locations are shaded.

From now on we assume without loss of generality that the plant model is given as EFA G and the specification is given as a set of forbidden locations $L_f \subseteq L$, and $L_s = L - L_f$ is the set of safe locations of G. Recall that Q^G denotes the set of reachable states of G. A state $q = (l, d) \in Q^G$ is a forbidden state iff $l \in L_f$, otherwise q is a safe state. G is such that it is impossible to reach a safe state from a forbidden state, and no forbidden location is marked. In the sequel, G_s denotes the EFA obtained from G by assigning F to the guard of every edge e for which $t_e \in L_f$, i.e. the terminal location of e is a forbidden location. Following the way G_s is constructed, it holds that $G_s \preceq G$. We call G_s the *safe subautomaton of* G.

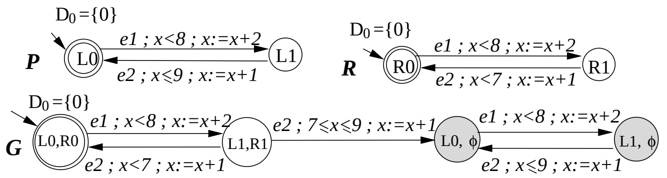


Figure 2. Refinement of P w.r.t. R (top) yields G (bottom)

Let $\Sigma_c \subseteq \Sigma$ and $\Sigma_u = \Sigma - \Sigma_c$ be the set of controllable and uncontrollable events of G respectively. The objective of control is to satisfy nonblockingness and safety while satisfying controllability requirement.

Definition 3 (Nonblocking, Safety, Controllability)
Let G be an EFA, L_f its set of forbidden locations, and G_s its safe subautomaton. A state $q \in Q^G$ is: (a) *nonblocking* if there exists a sequence s and a state $q' \in Q_m^G$ such that $q \xrightarrow{s} q'$; (b) *safe* if $q \in Q^{G_s}$; and (c) (G, L_f, Σ_u)*-controllable* (or simply *controllable* when clear from context) if q is safe and $\forall \sigma \in \Sigma^G(q) \cap \Sigma_u$, we have $Q^G(q, \sigma) \subseteq Q^{G_s}$. The EFA G_s is respectively nonblocking, safe and controllable if every reachable state of G_s is respectively nonblocking, safe and controllable. □

A supervisor is a function that assigns a stronger guard to each controllable edge.

Definition 4 (Supervisor)
Given a plant modeled by an EFA $G = (L, D, \Sigma, E, L_0, D_0, L_m)$, a *supervisor* $\mathcal{S}$ for G is a function $\mathcal{S} : E \to \mathcal{G}(d)$ which maps each plant-edge $e = (o_e, t_e, \sigma_e, g_e, f_e)$ to a guard such that $\mathcal{S}(e) \preceq g_e$ if $\sigma_e \in \Sigma_c$, and $\mathcal{S}(e) = g_e$ if $\sigma_e \in \Sigma_u$. □

Let $G^{\mathcal{S}}$ denote the subautomaton obtained from G by replacing its guards by those provided by $\mathcal{S}$. $\mathcal{S}$ is said to be nonblocking if $G^{\mathcal{S}}$ is nonblocking and safe if $G^{\mathcal{S}}$ is safe. Theorem 1 gives the existence condition of a nonblocking and safe supervisor $\mathcal{S}$ for G.

Theorem 1 (Supervisor existence)
Given a plant modeled by an EFA G and a specification defined by $G_s \preceq G$, there exists a nonblocking and safe supervisor $\mathcal{S}$ such that $Q^{G^{\mathcal{S}}} = Q^{G_s}$ iff G_s is nonblocking and controllable. □

Proof (If) Let E be the set of edges of G and $\mathcal{S}$ be a supervisor defined as follows: $\forall e = (o_e, t_e, \sigma_e, g_e, f_e) \in E$: $\mathcal{S}(e) = g_e$ if $t_e \in L_s$ or $\sigma_e \in \Sigma_u$, and $\mathcal{S}(e) = F$ otherwise. From the fact that it is impossible to reach a safe location from a forbidden location in G, it follows that every safe state of G remains reachable in $G^{\mathcal{S}}$ and thus, we have $Q^{G_s} \subseteq Q^{G^{\mathcal{S}}}$ as every reachable safe state of G is in Q^{G_s}. For the converse, let $q \in Q^{G^{\mathcal{S}}}$. G_s is controllable implies that every edge e of G linking a safe location to a forbidden location is controllable and $g_e = F$ in G_s. Then, $\mathcal{S}(e) = F$ from the above definition of $\mathcal{S}$. Added to the fact that it is impossible to reach a safe location from a forbidden location in G, it follows that only safe state are reachable in $G^{\mathcal{S}}$ and thus $Q^{G^{\mathcal{S}}} \subseteq Q^{G_s}$ meaning that $\mathcal{S}$ is safe. The above two inclusions imply that $Q^{G^{\mathcal{S}}} = Q^{G_s}$. For the nonblockingness: G_s is nonblocking implies every state in Q^{G_s} is nonblocking, and so is every state in $Q^{G^{\mathcal{S}}}$ (from the above equality) and then $\mathcal{S}$ is nonblocking.

(Only if) Let $\mathcal{S}$ be a nonblocking and safe supervisor such that $Q^{G^{\mathcal{S}}} = Q^{G_s}$. Let $q \in Q^{G^{\mathcal{S}}}$ and $q' \in Q^G$ such that $q \xrightarrow{\sigma} q'$ in G and $\sigma \in \Sigma_u$. From the definition of $\mathcal{S}$ (no change of guards of uncontrollable edges), it follows that $q' \in Q^{G^{\mathcal{S}}}$. Then for every state $q \in Q^{G^{\mathcal{S}}}$ and $\forall \sigma \in \Sigma^G(q) \cap \Sigma_u$, we have $Q^G(q, \sigma) \subseteq Q^{G^{\mathcal{S}}}$ and q is safe. Then G_s is controllable as $Q^{G^{\mathcal{S}}} = Q^{G_s}$. For the nonblockingness: $\mathcal{S}$ is nonblocking implies every state in $Q^{G^{\mathcal{S}}}$ is nonblocking, and so is every state in Q^{G_s} as $Q^{G^{\mathcal{S}}} = Q^{G_s}$ and thus G_s is nonblocking. ■

In case Theorem 1 conditions are not satisfied, we try to find a safe and nonblocking supervisor $\mathcal{S}$ such that $G^{\mathcal{S}} \preceq G_s$. Given two supervisors $\mathcal{S}^1$ and $\mathcal{S}^2$ for G, we say that $\mathcal{S}^2$ is *more permissive* than $\mathcal{S}^1$, denoted $\mathcal{S}^1 \preceq \mathcal{S}^2$, if for every edge e of G, $\mathcal{S}^1(e) \preceq \mathcal{S}^2(e)$. It follows that if $\mathcal{S}^1 \preceq \mathcal{S}^2$, then $G^{\mathcal{S}^1} \preceq G^{\mathcal{S}^2}$. If $\mathcal{S}(G, L_f)$ denotes the set of nonblocking and safe supervisors of G, then the *most permissive nonblocking and safe supervisor* of G, denoted $\mathcal{S}^{\uparrow} := sup\mathcal{S}(G, L_f)$, is the supervisor which is more permissive than any other supervisor in $\mathcal{S}(G, L_f)$ when the latter is nonempty. We call $G^{\mathcal{S}^{\uparrow}}$ the *supremal controllable and nonblocking subautomaton* of G_s.

IV. Supervisory synthesis for EFA

A. Computation of maximally permissive supervisor for EFA

Algorithm 1, denoted SSEFA (for Supervisory Synthesis for EFA), computes stronger, maximally permissive, guards for the edges of G such that the obtained EFA is nonblocking, safe and controllable. To compute the stronger guards for the controllable transitions we use two predicates associated to every location l: a *nonblocking predicate*, denoted N_l, and a *bad location predicate*, denoted B_l.

Let us explain the intuition of Algorithm 1. The outer iteration over the variable j successively strengthens the guard condition for each edge $e \in E$ to g_e^j, where $g_e^0 := g_e$. In the jth iteration, the nonblocking predicate specifies for location l the set of data that are nonblocking with respect to the current guards g_e^j. That is, for a state (l, d), if $N_l^j(d) = T$, then the state (l, d) is flagged nonblocking at iteration j, otherwise it is blocking. This predicate is computed iteratively with its initial valued $N_l^{j,0}$ assigned to T (resp. F) for marked (resp. unmarked) locations (line 3). It is then updated (line 4) such that if from a state (l, d), an enabled edge e leads to a state already flagged as nonblocking ($g_e^j(d) = T$ and $N_{t_e}^{j,k}(f_e(d)) = T$), then (l, d) is flagged nonblocking ($N_l^{j,k+1}(d) = T$). Note that during the computation, $N_l^{j,k}(d)$ is expressed by considering d as a variable, i.e. a predicate that defines a subset of data that characterize nonblocking states of location l. If for example $D = D_1$ and given the variable $d = [d(1)]$ (of domain D_1), if $g_e^j(d) = [d > 2]$, $f_e(d) = d + 1$, $N_{t_e}^{j,k}(d) = [d \leq 5]$ and $N_l^{j,k}(d) = F$, then we obtain $N_l^{j,k+1}(d) = F \vee [[d > 2] \wedge [d + 1 \leq 5]] = [d > 2] \wedge [d \leq 4]$.

On the other hand, in the jth iteration, the bad location predicate $B_l^j(d)$ specifies for location l the set of data that are undesirable (blocking, forbidden or uncontrollable). That is, if for a state (l, d), $B_l^j(d) = T$, then (l, d) is flagged as an undesirable state. This predicate is also computed iteratively with its initial valued $B_l^{j,0}$ assigned to T for forbidden locations and to $\neg N_l^j(d) \vee B_l^{j-1}(d)$ for safe locations ($\neg N_l^j(d)$ when $j = 0$) (line 12). $B_l^j(d)$ is then updated (line 13) such that if from a state (l, d), an enabled uncontrollable edge e

Algorithm 1 : Supervisory Synthesis for EFA (SSEFA)

```
Input: EFA $G = (L, D, \Sigma, E, L_0, D_0, L_m)$ with set of forbidden locations
       $L_f \subset L$
 1. Initialize iterators: $i := 0$, $j := 0$, $k := 0$
 2. Transitions guards are initially those of $G$: $\forall e \in E : g_e^0(d) = g_e(d)$ for
    every $d \in D$
 3. Initialize the nonblocking predicate of every location $l \in L$ as follows:
        $\forall d \in D,\ N_l^{j,0}(d) = \begin{cases} T, & \text{if } l \in L_m; \\ F, & \text{if } l \notin L_m. \end{cases}$        (1)
 4. Update the nonblocking predicate of every location $l \in L$ as follows:
        $\forall d \in D,\ N_l^{j,k+1}(d) = N_l^{j,k}(d) \vee \bigvee_{\{e|o_e=l\}} [g_e^j(d) \wedge N_{t_e}^{j,k}(f_e(d))]$        (2)
 5. if there exists $l \in L$ and $d \in D$ such that $N_l^{j,k}(d) \neq N_l^{j,k+1}(d)$
    then
 6.     $k := k + 1$
 7.     Go to 4
 8. else
 9.     for all $l \in L$ and $d \in D$ : $N_l^j(d) = N_l^{j,k}(d)$
10.     $k := 0$
11. end if
12. Initialize the bad location predicate of every location $l \in L$ as follows:
        $\forall d \in D,\ B_l^{j,0}(d) = \begin{cases} T, & \text{if } l \in L_f; \\ \neg N_l^j(d), & \text{if } l \notin L_f \text{ and } j = 0; \\ \neg N_l^j(d) \vee B_l^{j-1}(d), & \text{if } l \notin L_f \text{ and } j > 0. \end{cases}$        (3)
13. Update the bad location predicate of every location $l \in L$ as follows:
        $\forall d \in D,\ B_l^{j,i+1}(d) = B_l^{j,i}(d) \vee \bigvee_{\{e|o_e=l,\sigma_e \in \Sigma_u\}} [g_e^j(d) \wedge B_{t_e}^{j,i}(f_e(d))]$        (4)
14. if there exists $l \in L$ and $d \in D$ such that $B_l^{j,i+1}(d) \neq B_l^{j,i}(d)$ then
15.     $i := i + 1$
16.     Go to 13
17. else
18.     for all $l \in L$ and $d \in D$ : $B_l^j(d) = B_l^{j,i}(d)$
19.     $i := 0$
20. end if
21. Update the guard of every edge $e \in E$ as follows:
        $\forall d \in D,\ g_e^{j+1}(d) = \begin{cases} g_e^j(d) \wedge \neg B_{t_e}^j(f_e(d)), & \text{if } \sigma \in \Sigma_c; \\ g_e^j(d), & \text{if } \sigma \in \Sigma_u. \end{cases}$        (5)
22. if there exists $l \in L$ and $d \in D$ such that $g_e^{j+1}(d) \neq g_e^j(d)$ then
23.     $j := j + 1$
24.     Go to 3
25. else
26.     Stop
27. end if
```

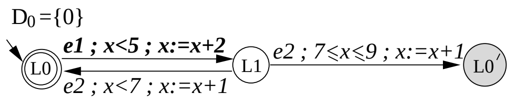


Figure 3. EFA SSEFA(G) for G of Figure 2

leads to an undesirable state ($g_e^j(d)=T$ and $B_{t_e}^{j,i}(f_e(d))=T$), then (l,d) becomes undesirable ($B_l^{j,i+1}(d)=T$). At line 21, the guard of every controllable edge is updated so that controllable transitions leading to undesirable states are disabled: if $B_l^j(d)=T$ for a state (l,d), then every controllable edge that has (l,d) as destination state has F as its guard following Line 21, making (l,d) unreachable through this edge.

Remark 1 In the above computation, only safe and "border forbidden" locations and the edges linking them need to be considered, where $l \in L_f$ is a border forbidden location if there exists an edge $e \in E$ such that $t_e = l$ and $o_e \in L_s$. The forbidden locations that are not border forbidden locations are irrelevant in the process of verifying the controllability and nonblockingness of G_s or subautomata. □

Let us illustrate Algorithm 1 by applying it to the simple example of Figure 2 (without location $(L1,\phi)$ as it is not a border forbidden location). We consider that $e1$ is controllable and $e2$ is uncontrollable. For simplification we rename the locations as follows: $(L0,R0) = L0$, $(L1,R1) = L1$ and $(L0,\phi) = L0'$. Here are the results of the algorithm (note that the variable d is equal to x here, given that D is 1-dimensional). For $j = 0$, we have for $k = 0$: $N_{L0}^{0,0}(x) = T$ and $N_{L1}^{0,0}(x) = N_{L0'}^{0,0}(x) = F$; and for $k = 1$: $N_{L0}^{0,1}(x) = T$, $N_{L1}^{0,1}(x) = F \vee \big[[[x<7] \wedge T] \vee [[7 \le x \le 9] \wedge F]\big] = [x<7]$, and $N_{L0'}^{0,1}(x) = F \vee \big[[x<8] \wedge F\big] = F$. For $k = 2$, we obtain the same result as for $k = 1$, so we stop the iteration on k and reset k to 0. For $i = 0$, we have: $B_{L0}^{0,0}(x) = \neg T = F$, $B_{L1}^{0,0}(x) = \neg[x<7] = [7 \le x \le 9]$ (following the domain of x), and $B_{L0'}^{0,0}(x) = T$. For $i = 1$, we obtain the same result as for $i = 0$, so the iterations stops. The new guard of the edge from $L0$ to $L1$ labeled by the controllable event $e1$ is: $g_{L0\to L1}^1(x) = [x<8] \wedge \neg[7 \le x+2 \le 9] = [x<5]$. For $j = 1$, no guard is modified, so the iteration on j stops.

The new EFA obtained after the application of Algorithm 1 is illustrated in Figure 3. The new guard of the edge from L0 to L1 ensures nonblocking and safety of the controlled system. Indeed, in L1, if the guard $[7 \le x \le 9]$ of the edge leading to L0' is satisfied, then the edge can be executed ($e2$ is uncontrollable) and this leads to a forbidden and blocking state. The new guard $[x<5]$ of the edge from L0 to L1 ensures that the guard $[7 \le x \le 9]$ is never satisfied in L1, whereas the guard $[x<7]$ of the edge from L1 to L0 is always satisfied when L1 is reached, ensuring nonblocking and safety. The supervisor ensures that $e1$ is executed only if the new guard is satisfied and equivalently, $e1$ is disabled when the original guard $[x<8]$ is satisfied but the new guard $[x<5]$ is unsatisfied, i.e. when $[5 \le x < 8]$.

B. Correctness of the supervisory synthesis algorithm

Given G, let SSEFA(G) denote the EFA obtained from G by applying Algorithm 1. SSEFA(G) has the same structure as G but has stronger guard conditions. The correctness is established through Proposition 1 and Theorems 2-3. We need to introduce the following notations. A state (l,d) of SSEFA(G) is said to be a *bad state* if $B_l^N(d) = T$, where N is the last iteration of j when the execution of Algorithm 1 stops. Given a set X, let $|X|$ denote the cardinality of X.

Proposition 1 (Termination of execution)
Given an EFA $G = (L, D, \Sigma, E, L_0, D_0, L_m)$ with data variables of finite domains and a set $L_f \subset L$ of forbidden locations, the following statements hold in the computation of SSEFA(G)*: in each iteration of j, the iterations over k and i both terminate in $O(|L||D|)$ steps, and the iteration over j itself terminates in $O(|L||D|)$ steps. The complexity of Algorithm 1 is $O(|L|^2|D|^2)$.* □

PROOF : Let us consider that each data variable $d(i)$, for $i = 1,...,p$ has $|d(i)|$ (finite) possible values. Then, the maximum number of possible values in D is $|D| = \prod_{i=1}^{p} |d(i)|$, and the number of reachable states of G is bounded by $|L||D|$. Let us show that every loop of Algorithm 1 terminates necessarily.

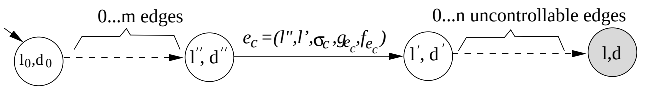
Inside every iteration over j, the iteration over k loops until no change of the nonblocking predicate for a state occurs (lines 4-11). Following Eq. 2, the nonblocking predicate of a state can only switch from F to T inside the same iteration of j due to the disjunction, and consequently, within each iteration over k, the set of nonblocking states increases by at least one state (except for the last iteration). Then, we can have at most $|L||D|$ possible changes of this predicate as the set of nonblocking states is bounded by $|L||D|$. Therefore, the iteration over k terminates in $O(|L||D|)$ steps.

Inside every iteration over j, the iteration over i loops until no change of the bad location predicate for a state occurs (lines 13-20). Following Eq. 4, the bad location predicate of a state can only switch from F to T due to the disjunction, and consequently, within each iteration over i, the set of bad states increases by at least one state (except for the last iteration). Therefore, the iteration over i terminates in $O(|L||D|)$ steps, as the set of bad states is bounded by $|L||D|$.

The iteration over j loops until no change of the guard of a controllable edge occurs (lines 21-25). Following Eq. (5), the guard of a controllable edge changes if and only if the bad location predicate of its terminal state changes. Therefore, the iteration over j loops until no change of the bad location predicate of a state occurs, i.e. until the set of bad states does not increase following the above Item (b). Therefore, the iteration over j terminates in $O(|L||D|)$ steps, as the set of bad states is bounded by $|L||D|$.

The above results demonstrate that the two inner loops over k and i as well as the outer loop over j of Algorithm 1 terminate in $O(|L||D|)$ steps and thus, the computational complexity of Algorithm 1 is $O(|L|^2|D|^2)$. ■

Remark 2 SSEFA often converges faster than worst-case complexity obtained in Proposition 1. For the example of Section V, the worst-case complexity is 5992704 iterations (16 locations, and cardinality of D of 9x17), but our algorithm terminates in a total of 9 iterations (for $j = 0$ there are 5 and 2 iterations over k and i respectively, and for $j = 1$ there



0...m edges
0...n uncontrollable edges
$e_c = (l'', l', \sigma_c, g_{e_c}, f_{e_c})$
l_0, d_0
l'', d''
l', d'
l, d

Figure 4. Illustration for proof of Lemma 1

is 1 iteration over both k and i). For the same reason, our algorithm is also efficient in space complexity, i.e. memory usage. Indeed, the entire state space of the system need not to be stored in memory but only the EFA model (compressed state space) and the computation is carried out directly on the compressed state space. Moreover, our algorithm can be implemented efficiently as the symbolic computation of predicates (nonblocking and bad location predicates and guards) can be realized efficiently using OBDDs. □

Lemma 1 *: Given an EFA $G = (L, D, \Sigma, E, L_0, D_0, L_m)$ and a set $L_f \subset L$ of forbidden locations such that* $\text{SSEFA}(G)$ *is computed in N iterations of iterator j, every state $(l, d) \in Q^G$ for which $B_l^N(d) = T$ is either unreachable in* $\text{SSEFA}(G)$ *or there exists an initial state $(l_0, d_0) \in (L_0 \times D_0)$ for which $B_{l_0}^N(d_0) = T$. Formally: $\forall (l, d) \in Q^G : B_l^N(d) = T \Rightarrow [(l, d) \notin Q^{\text{SSEFA}(G)}] \vee [\exists (l_0, d_0) \in L_0 \times D_0 \; s.t \; B_{l_0}^N(d_0) = T]$.* □

PROOF : We use figure 4 to illustrate our proof. $q = (l, d)$ is the state satisfying the condition $B_l^N(d) = T$. Without loss of generality, let us suppose that q is reached in G through a sequence s starting in (l_0, d_0). Generalization can be done by applying the same procedure below to every sequence in this general form, that is to every sequence $(l_0, d_0) \xrightarrow{s} (l, d)$. Suppose that $B_l^{j,i}(d)$ switches from F to T at iteration $j = U, i = V$, with $U < N$ (this predicate remains T until the last iteration N of j following (3)-(4). Following (4), $B_l^{j,i}(d)$ remains T for every subsequent iteration $j = U, i$. Then, other subsequent iterations on i will be done in the execution of Algorithm 1 because the test at line 14 will fail at iteration $j = U, i = V$ (the subsequent iteration will be executed even if $U = V = 0$ because line 13 is executed at least once). In the subsequent iterations $j = U, i > V$, the following holds.

(a) If there exists at least one controllable edge in the sequence s (represented by e_c in Fig 4 where e_c is the last controllable edge in the sequence s) and if there is no uncontrollable edge between (l', d') and (l, d) (i.e. $(l', d') = (l, d)$ on Figure 4), we obtain at line 21 (4) the following result in the execution: $g_{e_c}^{U+1}(d') = g_{e_c}^U(d') \wedge \neg B_l^U(d) = g_{e_c}^U(d') \wedge F = F$. Following (5), we have $g_{e_c}^N(d') = F$. Thus, the state (l, d) is unreachable in $\text{SSEFA}(G)$.

(b) If there exists at least one controllable edge in the sequence s (represented by e_c in Fig 4 where e_c is the last controllable edge in the sequence s) and if there are $n > 0$ uncontrollable edges between (l', d') and (l, d), let γ be the part of run $\gamma = (l'', d'') \xrightarrow{\sigma_c} (l', d') \xrightarrow{\sigma_{u_1}} \cdots \xrightarrow{\sigma_{u_n}} (l, d)$. Let $q_p = (l_p, d_p)$, $p = 1, ..., n+1$ be the state reached in γ after $\sigma_{u_{p-1}}$ with $q_1 = (l', d')$ and $q_{n+1} = (l, d)$, i.e. $q_p = (q_1, \mu\sigma_{u_{p-1}})$ where $\mu = \sigma_{u_1} \cdots \sigma_{u_{p-2}}$, and $e_p = (q_p, q_{p+1}, g_{e_p}, f_{e_p})$ be the edge from state q_p to q_{p+1}. By definition and following the fact that every e_p is an uncontrollable edge, we have $g_{e_1}(d') = T$, $f_{e_1}(d') = d_2$, $g_{e_p}(d_p) = T$ and $f_{e_p}(d_p) = d_{p+1}$. At line 13 of Algorithm 1, we obtain: following (4), the bad location predicate of every state (l_p, d_p), $p = 1, ..., n$ which is not evaluated to T at iteration $j = U, i = V$ will switch from F to T at an iteration $j = U, i > V$. For example, if at iteration $j = U, i = V$, $B_{l_n}^{U,V}(d_n) = F$ (predecessor state of (l, d)), then at iteration $j = U, i = V+1$, we will have following (4): $B_{l_n}^{U,V+1}(d_n) = F \vee [T \wedge T] = T$. every pair of states (l_p, d_p) and (l_{p+1}, d_{p+1}), for $p = 1, ..., n$ is in the same configuration as (l', d') and (l, d) and thus, after a number $V' \leq n$ subsequent iterations of i from iteration $j = U, i = V$, we will have: $B_{l_p}^{U,V+V'}(d_p) = T$ for $p = 1, ..., n$ and in particular, $B_{l'}^{U,V+V'}(d') = T$. We obtain at line 21 the following result: $g_{e_c}^{U+1}(d') = g_{e_c}^U(d') \wedge \neg B_{l'}^U(d') = g_{e_c}^U(d') \wedge F = F$. Following (5), we have $g_{e_c}^N(d') = F$. Thus, every state (l_p, d_p), for $p = 1, ..., n+1$, is unreachable in $\text{SSEFA}(G)$.

(c) If there exists no controllable edge in the sequence s (e_c on Figure 4 does not exist), then the states (l_0, d_0) and (l, d) are in the same configuration as (l', d') and (l, d) considered in Item (b) above. Thus, we will have after a given number V' subsequent iterations of i: $B_{l_0}^{U,V+V'}(d_0) = T$ and following (3)-(4), it holds that $B_{l_0}^N(d_0) = T$.

(d) Items (a)-(c) imply that if $B_l^N(d) = T$, then either (l, d) is unreachable in $\text{SSEFA}(G)$ (when the conditions of Items (a)-(b) hold) or there exists an initial state (l_0, d_0) such that $B_{l_0}^N(d_0) = T$ (when the conditions of Item (c) hold). ■

Theorem 2 (Nonblockingness and Controllability) *Given an EFA $G = (L, D, \Sigma, E, L_0, D_0, L_m)$ and a set $L_f \subset L$ of forbidden locations,* $\text{SSEFA}(G)$*, as computed by Algorithm 1, is nonblocking and controllable if none of its initial states is a bad state.* □

PROOF : **Proof of controllability**: To prove that $\text{SSEFA}(G)$ is controllable, we have to prove that every reachable state q of $\text{SSEFA}(G)$ is controllable, i.e. q is safe and $\forall \sigma \in \Sigma^G(q) \cap \Sigma_u$, it holds that $Q^G(q, \sigma) \subseteq Q^{\text{SSEFA}(G)}$. Let $q = (l, d)$ be a reachable state of $\text{SSEFA}(G)$. Then $\forall \sigma \in \Sigma^G(q) \cap \Sigma_u$, $Q^G(q, \sigma)$ are reachable in $\text{SSEFA}(G)$ because following (5), the guards of uncontrollable edges are never modified by Algorithm 1. On the other hand, q reachable in $\text{SSEFA}(G)$ implies by Lemma 1 that either q is not a bad state or there exists an initial state of $\text{SSEFA}(G)$ that is bad, i.e. either $B_l^N(d) = F$ or $\exists (l_0, d_0) \in L_0 \times D_0 \; s.t \; B_{l_0}^N(d_0) = T$. If q is not a bad state (no bad initial state exists), then q is safe because following (3)-(4), if $l \in L_f$, then necessarily $B_l^N(d) = T$. In conclusion, every reachable state of $\text{SSEFA}(G)$ is safe and controllable ($\text{SSEFA}(G)$ is safe and controllable) if none of the initial states of $\text{SSEFA}(G)$ are bad states.

Proof of nonblocking: Let a *path* $p_{q \to q'}^{\text{SSEFA}(G)}$ of $\text{SSEFA}(G)$ be a sequence of consecutive edges linking the states q and q' of $\text{SSEFA}(G)$. For a state $q \in Q^{\text{SSEFA}(G)}$, let $\mathcal{P}^{\text{SSEFA}(G)}(q) = \{p_{q \to q'}^{\text{SSEFA}(G)} | q' \in Q^{\text{SSEFA}(G)}\}$. For a path $p_{q \to q'}^{\text{SSEFA}(G)}$, let $\mathcal{L}(p_{q \to q'}^{\text{SSEFA}(G)})$ denote the set of states linked by $p_{q \to q'}^{\text{SSEFA}(G)}$, including q and q'. By extension, $\mathcal{L}(\mathcal{P}^{\text{SSEFA}(G)}(q))$ denotes the set of states linked by all the paths in $\mathcal{P}^{\text{SSEFA}(G)}(q)$. Suppose that there exists a reachable state $q = (l, d)$ of $\text{SSEFA}(G)$ that is blocking, i.e. no state in $\mathcal{L}(\mathcal{P}^{\text{SSEFA}(G)}(q))$ is marked. Algorithm 1 stops after iteration $j = N$ when there is no change of a guard of a transition between iterations $j = N-1$ and $j = N$ and this means also that there is no change of the bad location predicate of a state between iterations $j = N-1$ and $j = N$ (following (5)). This implies that there will be no change in SSEFA(G) if we run Algorithm 1 one more iteration of j, i.e. for $j = N+1$ (every state reachable in SSEFA(G) will remain reachable after the $(N+1)$th iteration of j). If we run Algorithm 1 on SSEFA(G) for $j = N+1$, we obtain the following results.

(a) Equation 1 of SSEFA (for j=N+1, k=0) and the fact that no state in $\mathcal{L}(\mathcal{P}^{\text{SSEFA}(G)}(q))$ is marked imply that $\forall (l', d') \in \mathcal{L}(\mathcal{P}^{\text{SSEFA}(G)}(q))$, $N_{l'}^{N+1,0}(d') = F$.

(b) Item (a) and (2) imply that $\forall (l', d') \in \mathcal{L}(\mathcal{P}^{\text{SSEFA}(G)}(q))$, $N_{l'}^{N+1,k}(d') = F$ at every subsequent iteration $j = N+1, k$. Indeed, $N_{l'}^{N+1,k+1}(d') = F \bigvee_{\{e|o_e=l'\}} \left[g_e^j(d') \wedge F\right] = F$. This is justified by the fact that even if the guards are satisfied, it takes that the nonblocking predicate of a reachable successor state of an unmarked state be true initially so it can impact recursively the nonblocking predicate of its predecessor states (switch from F to T).

(c) Item (b) and Line 9 of SSEFA imply that $\forall (l', d') \in \mathcal{L}(\mathcal{P}^{\text{SSEFA}(G)}(q))$, $N_{l'}^{N+1}(d') = F$.

(d) Item (c) and Equation 3 of SSEFA imply that $\forall (l', d') \in \mathcal{L}(\mathcal{P}^{\text{SSEFA}(G)}(q))$, $B_{l'}^{N+1,0}(d') = \neg F = T$.

(e) Items (d) and Equation 4 of SSEFA imply that $\forall (l', d') \in \mathcal{L}(\mathcal{P}^{\text{SSEFA}(G)}(q))$, $B_{l'}^{N+1,i}(d') = T$ at every subsequent iteration $j = N+1, i$. Indeed, $\forall (l', d') \in \mathcal{L}(\mathcal{P}^{\text{SSEFA}(G)}(q))$, we have $B_{l'}^{0,i+1}(d') = T \bigvee_{\{e|o_e=l', \sigma_e \in \Sigma_u\}} \left[g_e^j(d') \wedge T\right] = T$.

(f) Item (e) and Line 18 of SSEFA imply that $\forall (l', d') \in \mathcal{L}(\mathcal{P}^{\text{SSEFA}(G)}(q))$, $B_{l'}^{N+1}(d') = T$. In particular, we have $B_l^{N+1}(d) = T$ (recall that $q = (l, d)$).

(g) Lemma 1 and the result of Item (f) imply that either q is made unreachable in SSEFA(G) or there exists an initial state (l_0, d_0) such that $B_{l_0}^{N+1}(d_0) = T$. This means that if $\forall (l_0, d_0) \in (L_0 \times D_0) : B_{l_0}^{N+1}(d_0) = F$, there will be a change in SSEFA(G) ((l, d) is reachable in SSEFA(G) after $j = N$ and unreachable after $j = N+1$). This contradicts the fact that there must be no change in G between iterations $j = N$ and $j = N+1$. Therefore, our initial assumption that there exists a state q of SSEFA(G) that is blocking holds only if there exists an initial state (l_0, d_0) such that $B_{l_0}^{N}(d_0) = T$, otherwise SSEFA(G) is nonblocking. ■

Theorem 3 (supremal controllable and nonblocking EFA) *Given an EFA $G = (L, D, \Sigma, E, L_0, D_0, L_m)$ and a set $L_f \subset L$ of forbidden locations, if* SSEFA(G) *is nonblocking and controllable, then it is the supremal controllable and nonblocking subautomaton of G.* □

PROOF : We suppose that $\forall (l_0, d_0) \in (L_0 \times D_0) : B_{l_0}^{N}(d_0) = F$. Let G'' be a controllable and nonblocking subautomaton of G. Then, $Q^{\text{SSEFA}(G)} \subseteq Q^G$ and $Q^{G''} \subseteq Q^G$ and if $Q^{G''} \subseteq Q^{\text{SSEFA}(G)}$, then $G'' \preceq \text{SSEFA}(G)$. Our aim is thus to prove that $Q^{G''} \subseteq Q^{\text{SSEFA}(G)}$. Let $q = (l, d) \in Q^{G''}$. G'' is controllable implies $\forall \sigma \in \Sigma^G(q) \cap \Sigma_u$, $Q^G(q, \sigma) \subseteq Q^{G''}$. Note that $(l, d) \in Q^{G''}$ and G'' controllable imply $l \notin L_f$. G'' is nonblocking implies $\mathcal{L}(\mathcal{P}^{G''}(q)) \cap Q_m^{G''} \neq \emptyset$. Let $q_m = (l_m, d_m) \in \mathcal{L}(\mathcal{P}^{G''}(q)) \cap Q_m^{G''}$. We suppose without loss of generality that among all the marked states reachable from q, q_m is one among those reached from q after the smallest number of edges. Let n be this number of edges and $\gamma = (l, d) \overset{\sigma_1}{\to} (l_1, d_1) \cdots (l_{n-1}, d_{n-1}) \overset{\sigma_n}{\to} (l_m, d_m)$, where each state (l_p, d_p), for $p = 1, ..., n$ denotes the pth state in γ (excluding (l, d)) and $e_p = ((l_{p-1}, d_{p-1}), (l_p, d_p), \sigma_p, g_{e_p}, f_{e_p})$ denotes the edge linking states (l_{p-1}, d_{p-1}) and (l_p, d_p). By definition and following the fact that all the states in γ are reachable, we have that $g_{e_p}(d_{p-1}) = T$ and $f_{e_p}(d_{p-1}) = d_p$. SSEFA(G) is obtained from G by making unreachable some (blocking, forbidden and uncontrollable) states of G. To prove that $q \in Q^{\text{SSEFA}(G)}$, we have to prove that q is not made unreachable in the computation of SSEFA(G). In the computation of SSEFA(G), we obtain the following results.

(a) For $j = k = 0$, (1) gives: $N_l^{0,0}(d) = F$ (if $(l, d) \neq (l_m, d_m)$), $N_{l_m}^{0,0}(d_m) = T$ and $\forall (l_p, d_p)$ in γ (for $p = 1, ..., n-1$), $N_{l_p}^{0,0}(d_p) = F$.

(b) For $j = 0, k = 1$, (2) gives: $N_{l_m}^{0,1}(d_m) = T$, $N_{l_{n-1}}^{0,1}(d_{n-1}) = F \vee [T \wedge T] = T$ and $\forall (l_p, d_p)$ in γ (for $p = 1, ..., n-2$), $N_{l_p}^{0,1}(d_p) = F \vee [T \wedge F] = F$.

(c) At iteration $j = 0, k$ for $k = 2, ..., n+1$ (n=number of edges between q and q_m), (l_{n+2-k}, d_{n+2-k}) and (l_{n+3-k}, d_{n+3-k}) are in the same configuration as (l_{n-1}, d_{n-1}) and (l_m, d_m) at iteration $j = 0, k = 1$, and thus the nonblocking predicate of (l_{n+2-k}, d_{n+2-k}) will be changed from F to T, i.e. $N_{l_{n+2-k}}^{0,k}(d_{n+2-k}) = T$ and remains unchanged in every subsequent iteration j, k. We obtain at iteration $j = 0, k = n+1$ that $N_l^{0,n+1}(d) = T$ and remains unchanged in every subsequent iteration $j = 0, k$. That is, at Line 11 of SSEFA(G), we obtain $N_l^0(d) = T$. Every state of G'' is in the same configuration as (l, d) (G'' is nonblocking), so we have that in general, $\forall (l', d') \in Q^{G''} : N_{l'}^0(d') = T$.

(d) For $j = i = 0$, (3) gives: $B_l^{0,0}(d) = \neg T = F$ and in general, following Item (c), $\forall (l', d') \in Q^{G''} : B_{l'}^{0,0}(d') = F$.

(e) The fact that G'' is controllable implies that in every state $(l', d') \in Q^{G''}$, every uncontrollable event enabled at (l', d') in G is also enabled in G'' and leads to a state that belongs to $Q^{G''}$. Following this and (4) and given that $\forall (l', d') \in Q^{G''}$: $B_{l'}^{0,0}(d') = F$ (see Item (d) above), it results that the bad location predicate of a state $(l', d') \in Q^{G''}$ cannot be changed from F to T at any iteration j, i at Lines 13-17 of SSEFA. That is, $\forall (l', d') \in Q^{G''} : B_{l'}^0(d') = F$ and this result remains unchanged in every subsequent iteration of iterator j, as all the same conditions described in Items (a)-(d) hold at $j \neq 0$ as at $j = 0$.

(f) For $j = 0$, (5) and the above Item (e) imply that for every edge $e = ((r, x), (r', x'), \sigma_e, g_e, f_e)$ of G'', we have $g_e^1(x) = T$. Furthermore, following the above Item (e), we deduce that for every edge $e = ((r, x), (r', x'), \sigma_e, g_e, f_e)$ of G'', we have $g_e^j(x) = T$ after every iteration of j. That is, no state of $Q^{G''}$ is made unreachable due to a guard that should become unsatisfied after the application of SSEFA(G). In particular, the state $q = (l, d)$ (considered for the proof) remains reachable in SSEFA(G), i.e. $q \in Q^{\text{SSEFA}(G)}$ and therefore, we have $Q^{G''} \subseteq Q^{\text{SSEFA}(G)}$ and thus $G'' \preceq \text{SSEFA}(G)$. This proves that SSEFA$(G)$ is the supremal controllable and

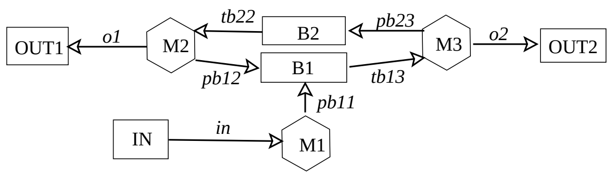


Figure 5. Work cycle system

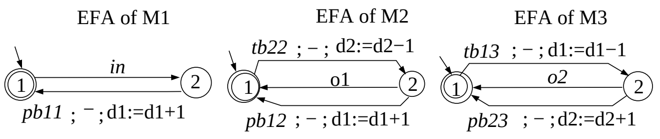


Figure 6. EFAs of the three machines

nonblocking subautomaton of G. ■

Following Theorem 3, if $\mathrm{SSEFA}(G)$ is nonblocking and controllable, i.e. if none of its initial states is a bad state (following Theorem 2), then $G^{S^{\uparrow}} = \mathrm{SSEFA}(G)$. Algorithm 1 solves thus our supervisory control problem stated in Section III by computing the EFA of the supremal controllable and nonblocking subautomaton of G if the latter exists.

V. Example

We consider a system consisting of three machines M1, M2 and M3, working on parts stored in two buffers B1 and B2 of size 16 and 8 respectively. Parts are supplied through an input buffer IN (of infinite size) and stored after being processed in two output buffers OUT1 and OUT2 (of infinite size). Figure 5 illustrates the system which operates as follows: M1 supplies B1 with parts taken from the input buffer IN; M2 takes a part from B2 and after processing puts it either in OUT1 or in B1; and M3 takes a part from B1 and after processing puts it either in OUT2 or in B2.

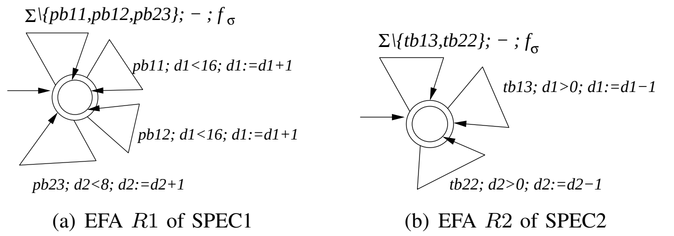
Figures 6 represents the EFA of M1, M2 and M3, where:

- event $tb22$ (resp. $tb13$) means that machine M2 (resp. M3) takes a part from B2 (resp. B1);
- event $pb11$ (resp. $pb12$ and $pb23$) means that machine M1 (resp. M2 and M3) puts a part in B1 (resp. B1 and B2);
- event in means M1 takes a part from IN; and
- event $o1$ (resp. $o2$) means that machine M2 (resp. M3) puts a part in OUT1 (resp. OUT2).

The events $pb11, o1, o2$ are the only uncontrollable events. The variables space is $D = D_1 \times D_2$, which record the number of parts in the two buffers, and following the size of the buffers, we have that $D_1 = \{0, 1, ..., 16\}$ and $D_2 = \{0, 1, ..., 8\}$. We suppose that B1 and B2 initially contain no part, and so $D_0 = \{(0, 0)\}$. In figures and for simplification, if no guard is present, then *true* is treated as the guard, and if the update function is not explicitly defined for a given data variable, it is assumed that the variable is updated to its current value. Moreover, when it holds that $f_e(d(i)) \notin D_i$, then $f_e(d(i))$ is implicitly replaced by the identity function. For example, if $f_e(d1) = d1 + 1$, then for $d1 = 16$, we obtain $f_e(d1) = 16$ instead of $f_e(d1) = 17$.

We consider the following two specifications:

SPEC1: buffers B1 and B2 must not overflow, i.e. a machine must not try to put a part in a buffer when it is full, i.e. when d1=16 or d2=8.



(a) EFA $R1$ of SPEC1 (b) EFA $R2$ of SPEC2

Figure 7. EFAs $R1$ and $R2$ of the specifications SPEC1 SPEC2

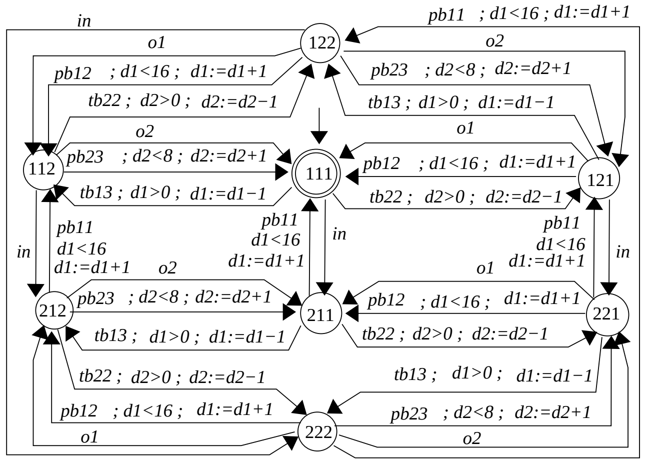


Figure 8. Safe part of the EFA G of the plant refined with the specification

SPEC2: buffers B1 and B2 must not underflow, i.e. a machine must not try to take a part from a buffer when it is empty, i.e. when d1=0 or d2=0. The EFAs $R1$ and $R2$ of SPEC1 and SPEC2 are represented in Figures 7(a) and 7(b) respectively, where for a transition on an event σ in $\Sigma \setminus \{pb11, pb12, pb23\}$ (for $R1$) or in $\Sigma \setminus \{tb13, tb22\}$ (for $R2$), f_σ is the same update function associated to the transition on σ in the models of Figure 6. Note the effect of SPEC1 is to add the guard $[d1 < 16]$ (resp., $[d2 < 8]$) to edges labeled $pb11$ and $pb12$ (resp., $pb23$), whereas the effect of SPEC2 is to add the guard $[d1 > 0]$ (resp., $[d2 > 0]$) to edges labeled $tb13$ (resp., $tb22$).

We compute the EFA G (plant $P = M1\|M2\|M3$ refined with the specification $R = R1\|R2$) as described in Section III. G is essentially the union of the two EFAs shown in Figures 8 and 9, where Figure 8 shows only the safe part of G and Figure 9 shows only the border forbidden locations of G along with the edges leading to them. It is clear from Figure 9 that the only uncontrollable transitions that cause the violation of safety are the transitions on $pb11$ under the condition that M2 is in its second location and the buffer B1 is full ($d1 = 16$). As computed below, our maximally permissive controller avoids this situation by ensuring that whenever M2 is in its second location, the buffer B1 is not full ($d1 < 16$).

Let us now apply the controller synthesis algorithm to the problem. We have that $\Sigma_u = \{pb11, o1, o2\}$. We use d in place of $[d1, d2]$, i.e. $d = [d1, d2]$ (two dimensional vector). It holds that for every forbidden location l, $N_l^j(d) = F$ and $B_l^j(d) = T$ at every iteration of j. Here are the results of applying the algorithm SSEFA to the model G.

j=0,k=0: $N_{111}^{0,0}(d) = T$ and $N_{L_i}^{0,0}(d) = F \; \forall L_i \neq 111$.

j=0,k=1: $N_{111}^{0,1}(d) = T$.

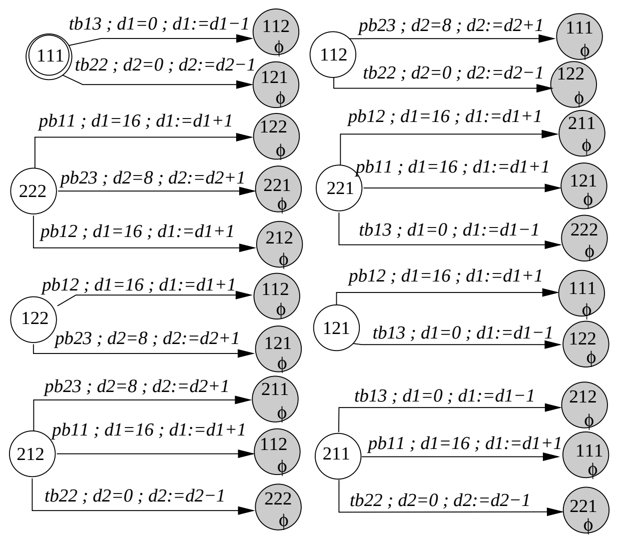


Figure 9. Forbidden locations of G reached from safe locations

$N^{0,1}_{112}(d) = F \vee [(d2<8)\wedge T] \vee [(d2>0)\wedge F]$
$\quad \vee [T \wedge T] = T.$
$N^{0,1}_{121}(d) = F \vee [(d1<16)\wedge T] \vee [(d1>0)\wedge F]$
$\quad \vee [T \wedge T] = T.$
$N^{0,1}_{211}(d) = F \vee [(d1<16)\wedge T] \vee [(d1>0)\wedge F] \vee$
$\quad [(d2>0)\wedge F] = [d1<16].$
$N^{0,1}_{212}(d) = F \vee [(d2<8)\wedge F] \vee [T\wedge F]$
$\quad \vee [(d1<16)\wedge F] = F.$
$N^{0,1}_{221}(d) = F \vee [(d1<16)\wedge F] \vee [T\wedge F]$
$\quad \vee [(d1<16)\wedge F] = F.$
$N^{0,1}_{122}(d) = F \vee [(d1<16)\wedge F] \vee [(d2<8)\wedge F]$
$\quad \vee [T\wedge F] \vee [T\wedge F] \vee [T\wedge F] = F.$
$N^{0,1}_{222}(d) = F \vee [(d1<16)\wedge F] \vee [(d1<16)\wedge F]$
$\quad \vee [(d2<8)\wedge F] \vee [T\wedge F] = F.$

j=0,k=2: $N^{0,2}_{111}(d) = T$; $N^{0,2}_{112}(d) = T$; and $N^{0,2}_{121}(d) = T$.
$N^{0,2}_{211}(d) = [d1<16] \vee [(d1<16)\wedge T] \vee [(d1>0)$
$\quad \wedge F] \vee [(d2>1)\wedge F] = [d1<16].$
$N^{0,2}_{212}(d) = F \vee [(d1<16)\wedge T] \vee [(d2<8)\wedge$
$\quad (d1<16)] \vee [(d2>0)\wedge F] = [d1<16].$
$N^{0,2}_{221}(d) = F \vee [(d1<16)\wedge T] \vee [(d1<16)$
$\quad \wedge (d1+1<16)] \vee [(d1>0)\wedge F]$
$\quad = [d1<16].$
$N^{0,2}_{122}(d) = [T\wedge T] = T$ and $N^{0,2}_{222}(d) = F$.

j=0,k=3: $N^{0,3}_{111}(d) = T$; $N^{0,3}_{112}(d) = T$; and $N^{0,3}_{121}(d) = T$.
$N^{0,3}_{211}(d) = [d1<16] \vee [(d1<16)\wedge T]$
$\quad \vee [(d1>0)\wedge (d1-1<16)]$
$\quad \vee [(d2>0)\wedge (d1<16)]$
$\quad = [d1<16] \vee [d1>0] = T.$
$N^{0,3}_{212}(d) = [d<16] \vee [(d1<16)\wedge T]$
$\quad \vee [(d2<8)\wedge (d1<16)]$
$\quad \vee [(d2>0)\wedge F] = [d1<16].$
$N^{0,3}_{221}(d) = [d1<16] \vee [(d1<16)\wedge T]$
$\quad \vee [(d1<16)\wedge (d1+1<16)]$
$\quad \vee [(d1>0)\wedge F] = [d1<16].$
$N^{0,3}_{122}(d) = T.$
$N^{0,3}_{222}(d) = F \vee [(d1<16)\wedge T] \vee [(d1<16)$
$\quad \wedge (d1+1<16)] \vee [(d2<8)$
$\quad \wedge (d1<16)] = [d1<16].$

j=0,k=4: $N^{0,4}_{111}(d) = N^{0,4}_{112}(d) = N^{0,4}_{121}(d) = N^{0,4}_{211}(d) = T.$
$N^{0,4}_{212}(d) = [d1<16] \vee [T\wedge T] = T.$
$N^{0,4}_{221}(d) = [d1<16] \vee [T\wedge T] = T.$
$N^{0,4}_{122}(d) = T.$
$N^{0,4}_{222}(d) = [d1<16] \vee [T\wedge T] = T.$

j=0,k=5: no change compared to $k=4$, then: $N^0_l(d) = T$ for every safe location and we reset k to 0.

j=0,i=0: $B^{0,0}_l(d) = \neg T = F$ for every $l \notin L_f$ and $B^{0,0}_{l'}(d) = T$ for every $l' \in L_f$

j=0,i=1: $B^{0,1}_{111}(d) = F.$
$B^{0,1}_{112}(d) = F \vee [T\wedge F] = B^{0,1}_{121}(d) = F.$
$B^{0,1}_{211}(d) = F \vee [(d1<16)\wedge F] \vee [(d1=16)\wedge T]$
$\quad = [d1=16].$
$B^{0,1}_{212}(d) = F \vee [(d1<16)\wedge F] \vee [(d1=16)\wedge T]$
$\quad = [d1=16].$
$B^{0,1}_{221}(d) = F \vee [(d1<16)\wedge F] \vee [(d1=16)\wedge T]$
$\quad = [d1=16].$
$B^{0,1}_{122}(d) = F \vee [T\wedge F] = F.$
$B^{0,1}_{222}(d) = F \vee [(d1<16)\wedge F] \vee [(d1=16)\wedge T]$
$\quad = [d1=16].$

j=0,i=2: no change compared to $i=1$, then:
$B^0_{111}(d) = B^0_{112}(d) = B^0_{211}(d) = B^0_{122}(d) = F$; and $B^0_{211}(d) = B^0_{212}(d) = B^0_{221}(d) = B^0_{222}(d) = [d1 = 16]$; and we reset i to 0.

New guards (only those that change), in the form g_{o_e,σ,t_e}:
$g_{111,in,211}(d) = T \wedge \neg[d1=16] = [d1<16].$
$g_{112,in,212}(d) = T \wedge \neg[d1=16] = [d1<16].$
$g_{121,in,221}(d) = T \wedge \neg[d1=16] = [d1<16].$
$g_{212,pb23,211}(d) = [d2<8] \wedge \neg[d1=16] = [d2<8] \wedge [d1<16].$
$g_{212,tb22,222}(d) = [d2>0] \wedge \neg[d1=16] = [d2>0] \wedge [d1<16]$
$g_{221,pb12,211}(d) = [d1<16] \wedge \neg[d1+1=16] = [d1<15].$
$g_{221,tb13,222}(d) = [d1>0] \wedge \neg[d1-1=16] = [d1>0].$
$g_{211,tb13,212}(d) = [d1>0] \wedge \neg[d1-1=16] = [d1>0]$
$g_{211,tb22,221}(d) = [d2>0] \wedge \neg[d1=16] = [d2>0] \wedge [d1<16]$
$g_{122,in,222}(d) = T \wedge \neg[d1=16] = [d1 \neq 16] = [d1<16].$
$g_{222,pb12,212}(d) = [d1<16] \wedge \neg[d1+1=16] = [d1<15].$
$g_{222,pb23,221}(d) = [d2<8] \wedge \neg[d1=16] = [d2<8] \wedge [d1<16].$

The guard of every edge labeled by a controllable event and leading to a forbidden location is equal to F. For $j=1$, there is no change of the guards, so the algorithm stop.

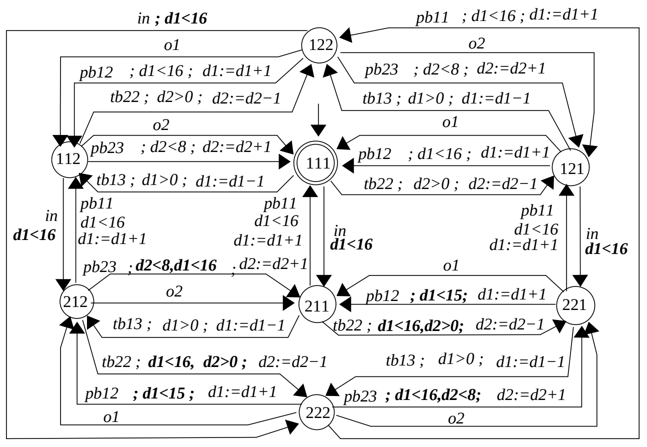


Figure 10. Reachable part of $\mathrm{SSEFA}(G)$

Figure 10 shows the reachable part of SSEFA(G). In particular the forbidden locations of Figure 9 are no longer reachable since those guards of controllable transitions become false where those guards of uncontrollable transitions can never be satisfied.

VI. Conclusion

We presented in this paper a new algorithm for synthesis of supervisors for discrete event systems (DES) modeled by Extended Finite Automata (EFA). The algorithm operates by strengthening the guards of controllable edges of the plant EFA so that undesirable states, i.e. blocking, forbidden or uncontrollable states, become unreachable. The obtained EFA is the supremal controllable, safe and nonblocking subautomaton of the plant EFA. Our algorithm benefits from the efficiency of the EFA modeling framework and the solution of the control problem can be interpreted intuitively.

References

[1] R. St-Denis, “Designing reactive systems: integration of abstraction techniques into a synthesis procedure,” *Journal of Systems and Software*, vol. 60, no. 2, pp. 103 – 112, 2002.

[2] A. Vahidi, M. Fabian, and B. Lennartson, “Efficient supervisory synthesis of large systems,” *Control Engineering Practice*, vol. 14, no. 10, pp. 1157 – 1167, 2006.

[3] R. Leduc, P. Dai, and R. Song, “Synthesis method for hierarchical interface-based supervisory control,” *IEEE Transactions on Automatic Control*, vol. 54, no. 7, pp. 1548 –1560, July 2009.

[4] K.-T. Cheng and A. S. Krishnakumar, “Automatic generation of functional vectors using the extended finite state machine model,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 1, no. 1, pp. 57–79, 1996.

[5] Y.-L. Chen and F. Lin, “Modeling of discrete event systems using finite state machines with parameters,” in *Proc. of the IEEE International Conference on Control Applications*, Sept. 2000, pp. 941 – 946.

[6] M. Sköldstam, K. Åkesson, and M. Fabian, “Modeling of discrete event systems using finite automata with variables,” in *46th IEEE Conference on Decision and Control*, Dec. 2007, pp. 3387–3392.

[7] Y.-L. Chen and F. Lin, “Safety control of discrete event systems using finite state machines with parameters,” in *Proceedings of American Control Conference*, Arlington, VA, June 2001, pp. 975–980.

[8] S. Miremadi, K. Åkesson, and B. Lennartson, “Extraction and representation of a supervisor using guards in extended finite automata,” in *9th International Workshop on Discrete Event Systems (WODES 2008)*, May 2008, pp. 193–199.

[9] R. E. Bryant, “Symbolic boolean manipulation with ordered binary-decision diagrams,” *ACM Computing Surveys*, vol. 24, pp. 293–318, Sep. 1992.

[10] Y. Yang, A. Mannani, and P. Gohari, “Implementation of supervisory control using extended finite-state machines,” *International Journal of Systems Science*, vol. 39, no. 12, pp. 1115–1125, 2008.

[11] M. Sköldstam, K. Åkesson, and M. Fabian, “Supervisory control applied to automata extended with variables - revised,” Department of Signals and Systems, Chalmers University of Technology, Tech. Rep. R001/2008, 2008.

[12] A. Voronov and K. Åkesson, “Verification of process operations using model checking,” in *IEEE International Conference on Automation Science and Engineering (CASE 2009)*, August 2009, pp. 415 –420.

[13] T. Le Gall, B. Jeannet, and H. Marchand, “Supervisory control of infinite symbolic systems using abstract interpretation,” in *44th IEEE Conference on Decision and Control and 2005 European Control Conference (CDC-ECC ’05)*, Dec. 2005, pp. 30–35.

[14] B. Gaudin and P. H. Deussen, “Supervisory control on concurrent discrete event systems with variables,” in *Proceedings American Control Conference (ACC ’07)*, Jul. 2007, pp. 4274–4279.

[15] A. Mannani, Y. Yang, and P. Gohari, “Distributed extended finite-state machines: communication and control,” in *8th International Workshop on Discrete Event Systems*, July 2006, pp. 161–167.

[16] R. Kumar and V. K. Garg, “On computation of state avoidance control for infinite state systems in assignment program framework,” *IEEE Transactions on Automation Science and Engineering*, vol. 2, no. 1, pp. 87–91, January 2005.

[17] L. Ouedraogo, R. Kumar, R. Malik, and K. Åkesson, “Symbolic approach to nonblocking and safe control of extended finite automata,” in *6th IEEE Conference on Automation Science and Engineering (CASE 2010)*, Toronto, Canada, Aug. 2010, pp. 471 –476.

[18] R. Kumar, V. K. Garg, and S. I. Marcus, “Predicates and predicate transformers for supervisory control of discrete event dynamical systems,” *IEEE Transactions on Automatic Control*, vol. 38, no. 2, pp. 232–247, Feb. 1993.

Lucien Ouedraogo received the M.A.Sc and Ph.D degrees in Electrical Engineering from Université de Sherbrooke (Québec, Canada) in 2003 and 2008 respectively. He is currently a Postdoctoral Fellow in the Department of Electrical and Computer Engineering at Iowa State University (USA). His current research interests include the synthesis and performance analysis of communication protocols for real-time applications, and the modeling, supervisory control and failures diagnosis of discrete event dynamic systems.

Ratnesh Kumar (S87-M90-SM00-F07) received the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology at Kanpur, India, in 1987, and the M.S. and the Ph.D. degree in Electrical and Computer Engineering from the University of Texas at Austin, Texas, in 1989 and 1991, respectively. From 1991-2002 he was on the faculty of University of Kentucky, and since 2002 he is on the faculty of the Iowa State University. He has held visiting position at the Institute of Systems Research at the University of Maryland at College Park, the Applied Research Laboratory at the Pennsylvania State University, the NASA Ames Research Center, the Idaho National Laboratory, and the United Technology Research Center. He is a coauthor of the book Modeling and Control of Logical Discrete Event Systems, Kluwer Academic Publishers, 1995. He serves on the program committee for the IEEE Control Systems Society, the International Workshop on Discrete Event Systems, and the IEEE Workshop on Software Cybernetics. He is or has been an associate editor of SIAM Journal on Control and Optimization, IEEE Transactions on Robotics and Automation, Journal of Discrete Event Dynamical Systems, and IEEE Control Systems Society. He is a Fellow of the IEEE.

Robi Malik received the M.S. and Ph.D. degree in computer science from the University of Kaiserslautern, Germany, in 1993 and 1997, respectively. From 1998 to 2002, he worked in a research and development group at Siemens Corporate Research in Munich, Germany, where he was involved in the development and application of modelling and analysis software for discrete event systems. Since 2003, he is lecturing at the Department of Computer Science at the University of Waikato in Hamilton, New Zealand. He is participating in the development of the Supremica software for modelling and analysis of discrete event systems. His current research interests are in the area of model checking and synthesis of large discrete event systems and other finite-state machine models.

Knut Åkesson received the M.S. degree in 1997 in computer science and engineering from Lund Institute of Technology, Lund, Sweden, and the Ph.D. degree in 2002 in control engineering from Chalmers University of Technology, Gothenburg, Sweden, where he currently is an associate professor. His main research interest is to develop and applying formal methods for verification and synthesis of control logic. He also initiated the development of Supremica, a tool for verification and synthesis of discrete event supervisors.