Efficient Symbolic Supervisor Synthesis for Extended Finite Automata

Zhennan Fei, Sajed Miremadi, Knut Åkesson and Bengt Lennartson

***Abstract*—The state-space explosion problem, resulting from the reachability computations in controller synthesis, is one of the main obstacles preventing supervisory control theory from having an industrial breakthrough. To alleviate this problem a strategy is to symbolically perform the synthesis procedure by using binary decision diagrams. Based on this principle, the work presented in this paper develops an efficient symbolic reachability approach for discrete event systems that are modeled as finite automata with variables, referred to as extended finite automata. By utilizing a disjunctive event partitioning technique, the proposed approach first partitions the transition relation of the considered system into a set of partial transition relations. These partial transition relations are then selected systematically to perform the reachability analysis, which is the most fundamental challenge for synthesizing supervisors. It has been shown through solving a set of benchmark supervisory control problems for extended finite automata that the proposed approach significantly improves scalability in comparison to previously published results.**

***Index Terms*—Supervisory control theory, extended finite automata, binary decision diagrams, partitioning techniques, reachability analysis.**

I. Introduction

SUPERVISORY control theory (SCT) [1] is a formal framework for modeling and control of discrete event systems (DESs). Given a system to be controlled, the *plant*, and a *specification*, a control function, referred to as the *supervisor*, can be automatically synthesized. The supervisor restricts the behavior of the plant in the least restrictive way, such that the specification is always fulfilled.

SCT has gained a lot of focus within academic research communities, though the industrial acceptance is scarce due to the following two drawbacks: the discrepancy between the signal-based reality and the event-based automata analysis framework makes the rigorous modeling of systems difficult; the computation of supervisors might result in *state-space explosion*, owing to the high computational complexity and limited amount of memory.

There are a certain number of modeling formalisms that can be used to model DESs and are suitable for supervisor synthesis. Specific examples include Petri nets [2], process algebra [3], hierarchical finite state machines (HFSM) [4] and state tree structure (STS) [5]. While Petri nets are able to model infinite state systems, structural constraints are in general necessary for the considered synthesis problem to be decidable. The ability for process algebra-based approaches to handle efficient synthesis of large scale systems without structural constraints has not been reported to the authors best knowledge. On the other hand, STS and HFSM are two variants of StartChart [6], which is an extension of finite automata (FAs) with hierarchy, concurrency and communication. The attractiveness of STS and HFSM is that complex systems can be modeled at different levels of detail, and hence, structured and comprehensible models can be obtained.

In [7], an alternative modeling framework was presented whereby users can model a discrete event system in the form of *extended finite automata* (EFAs). An EFA is an augmentation of ordinary automata, extended with variables, guard expressions and action functions. As seen in many industrial applications, parts of a system such as sensors, actuators and buffers can be conveniently modeled using variables. The richer structure and semantics that are provided by these variables enable the representation of the modeled behavior in a conciser manner than the ordinary automata.

Although the EFA modeling framework allows for compact representations of systems, the supervisory analysis of it remains a challenging task. In [7], the authors suggested an algorithm to transform EFAs into equivalent FAs. Thereafter, synthesis can be carried out by using existing approaches. However, the disadvantage of this algorithm is that in the worst case, the number of transitions created for FAs will be exponential to the number of EFA transitions, since the algorithm transforms guards into disjunctive normal form (DNF) and creates a transition for each term in the DNF. In [8], the authors presented an approach where a supervisor can be synthesized directly from EFA models by using *binary decision diagram* (BDD) [9] [10]. For BDDs, the computational complexity of synthesis is no longer dependent on the number of states but on the number of nodes in the BDDs representing the state-spaces. However, the approach in [8] essentially performs the breadth first search on a single monolithic BDD representing the monolithic transition relation of the composed EFA model. This might lead to huge number of nodes in the intermediate BDDs during the synthesis, although the final BDD is usually manageable.

Contributions: The contributions of this paper are the following (i) By using a partitioning technique [11], it is shown how a set of disjunctive partial transition relations, one for each event, is constructed to represent the transition relation of a set of EFAs. (ii) A new algorithm that exploits the disjunctive partial transition relations to compute a BDD representing all reachable states is proposed. (iii) Also, the correctness of the proposed algorithm is formally proved. (iv)

The authors are with the Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden (e-mail: zhennan@chalmers.se).

This work was carried out within the Wingquist Laboratory VINN Excellence Centre within the Area of Advance - Production at Chalmers, and supported by VINNOVA and the Swedish Science Foundation. The support is gratefully acknowledged.

It is shown through solving a set of benchmark supervisory control problems that the proposed algorithm has improved scalability in comparison to the symbolic approach presented in [8] due to its ability to explore the state-space in a structured way, which can significantly alleviate the problem with large intermediate BDDs.

Related work: A number of related works with respect to the efficient synthesis using BDD are presented respectively in [12] [13] and [14]. However these approaches are only applicable for finite automata without variables. A symbolic algorithm for supervisory control synthesis of state-tree structures (STS) is presented in [5], but like the aforementioned approaches, variables are not allowed in STS.

Outline: The content of the paper is organized as follows: Section II provides necessary preliminaries used throughout the paper. By means of a simple example, Section III informally illustrates the proposed approach. The main contributions pursued by this paper are then detailed in Section IV and Section V. Section VI discusses the experimental results and finally, Section VII adds some concluding remarks.

II. Preliminaries

A. Extended Finite Automata

EFA is an augmentation of an ordinary finite automaton (FA) with variables in the form of *guards* and *actions*.

Let $\mathcal{V} = \{v_1, \dots, v_n\}$ be a set of n variables and D_i be the domain of v_i. A variable evaluation, denoted by η, assigns a value in D_i to the variable $v_i \in \mathcal{V}$. For brevity, we simply write $\eta[i]$ to denote the value of v_i in η.

Definition II.1. Extended Finite Automaton. An extended finite automaton E defined over variable set $\mathcal{V} = \{v_1, \dots, v_n\}$ is an 8-tuple

$$E = (L, D, \Sigma, \rightarrow, \ell^0, d^0, L^m, D^m)$$

where L is the finite set of locations; $D = D_1 \times \ldots \times D_n$ is the finite domain of the variables; Σ is a finite set of events (i.e., the alphabet); $\ell^0 \in L$ is the initial location; $d^0 \in D$ is the vector of the initial values for the variables; $L^m \subseteq L$ is the set of marked locations; $D^m \subseteq D$ is the set of vectors of marked values for the variables; $\rightarrow \subseteq L \times \Sigma \times \mathcal{G} \times \mathcal{A} \times L$ is the transition relation where $\mathcal{G}$ is a set of guard predicates and $\mathcal{A}$ is a set of actions. Each action $a \in \mathcal{A}$, as an n-tuple of functions $(a_1, \dots, a_n)$, updates η to a new variable evaluation $\acute{\eta} = a(\eta)$. Any function a_i that does not update variable v_i is denoted by ξ. The symbol Ξ is used to denote an n-tuple $(\xi, \xi, \dots, \xi)$, indicating that no variable is updated.

The initial variable evaluation, denoted by η_0, assigns d^0 to $\mathcal{V}$. In the following, we simply write $\ell \overset{\sigma}{\rightarrow}_{g/a} \acute{\ell}$ to denote $(\ell, \sigma, g, a, \acute{\ell}) \in \rightarrow$. If g is absent (i.e., g is a tautology), then the transition is denoted as $\ell \overset{\sigma}{\rightarrow}_{a} \acute{\ell}$. If a is absent (i.e., $a = \Xi$), then it is denoted as $\ell \overset{\sigma}{\rightarrow}_{g} \acute{\ell}$.

Definition II.2. Explicit state transition relation. Let $E = (L, D, \Sigma, \rightarrow, \ell^0, d^0, L^m, D^m)$ be an EFA over $\mathcal{V} = \{v_1, \dots, v_n\}$. The explicit state transition relation of E is defined as

$$\begin{aligned}\mapsto_E = \{(\ell, \eta, \sigma, \acute{\ell}, \acute{\eta}) \in L \times D \times \Sigma \times L \times D \mid \\ \ell \overset{\sigma}{\rightarrow}_{g/a} \acute{\ell} \in \rightarrow, \eta \models g, \acute{\eta} = a(\eta)\},\end{aligned}$$

where η is the variable evaluation before executing the transition. If g holds for η, i.e., $\eta \models g$, then the transition can be taken and η is updated by a, i.e, $\acute{\eta} = a(\eta)$. Note that if variable $v_i \in \mathcal{V}$ is not updated, i.e., $a_i = \xi$, then $\acute{\eta}[i] = \eta[i]$. For notational convenience, we shall let $\mapsto_{\ell \overset{\sigma}{\rightarrow}_{g/a} \acute{\ell}}$ denote the explicit state transitions for $\ell \overset{\sigma}{\rightarrow}_{g/a} \acute{\ell}$. Furthermore, we let $\overset{\sigma}{\mapsto}_E$ be the subset of transitions in E that are labeled by $\sigma \in \Sigma$.

An EFA E is deterministic if there exists $(\ell, \eta, \sigma, \ell', \eta') \in \mapsto_E$ and $(\ell, \eta, \sigma, \ell'', \eta'') \in \mapsto_E$, then we always have $(\ell', \eta') = (\ell'', \eta'')$. In this work, we focus on deterministic EFAs.

The full composition of two EFAs is defined as the *extended full synchronous composition*.

Definition II.3. Extended Full Synchronous Composition (EFSC). Let $E_k = (L_k, D, \Sigma_k, \rightarrow_k, \ell_k^0, d^0, L_k^m, D^m)$, $k = 1, 2$, be two EFAs over variable set $\mathcal{V} = \{v_1, \dots, v_n\}$, and with the same initial and marked variable evaluations. The EFSC of E_1 and E_2 is defined as

$$E_1 || E_2 = (L_1 \times L_2, D, \Sigma_1 \cup \Sigma_2, \rightarrow, \langle \ell_1^0, \ell_2^0 \rangle, d^0, L_1^m \times L_2^m, D^m)$$

where a transition $\langle \ell_1, \ell_2 \rangle \overset{\sigma}{\rightarrow}_{g/a} \langle \acute{\ell}_1, \acute{\ell}_2 \rangle$ is defined by the following three rules:

1) if $\sigma \in \Sigma_1 \cap \Sigma_2$ and $\ell_1 \overset{\sigma}{\rightarrow}_{g^1/a^1} \acute{\ell}_1 \in \rightarrow_{E_1}$ and $\ell_2 \overset{\sigma}{\rightarrow}_{g^2/a^2} \acute{\ell}_2 \in \rightarrow_{E_2}$ are action consistent, then
   - $g = g^1 \wedge g^2$,
   - each action function a_i of a is defined as

$$a_i = \begin{cases} a_i^1 & \text{if } a_i^1 \neq \xi \text{ and } a_i^2 \neq \xi, \\ a_i^1 & \text{if } a_i^1 \neq \xi \text{ and } a_i^2 = \xi, \\ a_i^2 & \text{if } a_i^1 = \xi \text{ and } a_i^2 \neq \xi, \\ \xi & \text{if } a_i^1 = \xi \text{ and } a_i^2 = \xi. \end{cases}$$

2) if $\sigma \in \Sigma_1 \backslash \Sigma_2$ and $\ell_1 \overset{\sigma}{\rightarrow}_{g^1/a^1} \acute{\ell}_1 \in \rightarrow_{E_1}$ then $g = g^1$ and $a = a^1$ and $\acute{\ell}_2 = \ell_2$.
3) if $\sigma \in \Sigma_2 \backslash \Sigma_1$ and $\ell_2 \overset{\sigma}{\rightarrow}_{g^2/a^2} \acute{\ell}_2 \in \rightarrow_{E_2}$ then $g = g^2$ and $a = a^2$ and $\acute{\ell}_1 = \ell_1$.

The transitions $\ell_1 \overset{\sigma}{\rightarrow}_{g^1/a^1} \acute{\ell}_1 \in \rightarrow_{E_1}$ and $\ell_2 \overset{\sigma}{\rightarrow}_{g^2/a^2} \acute{\ell}_2 \in \rightarrow_{E_2}$ are action consistent if $a^1(\eta)[i] = a^2(\eta)[i]$ for all $\eta \in D$ and $i \in \{1, \dots, n\}$ where $a_i^1 \neq \xi$ and $a_i^2 \neq \xi$. Note that if the two transitions update variable v_i to different values, then the composed transition is not defined. A good modeling practice is that for each variable and for each event, only one EFA is allowed to update the variable with the event, while the same variable can be updated in different EFAs with any other event. In this case, the actions are structurally consistent. The EFSC operator is both commutative and associative, and thus it can be extended to handle an arbitrary number of EFAs.

B. Binary Decision Diagrams

Binary decision diagrams (BDDs) [10] are powerful data structures for representing Boolean functions. Given a set of binary/Boolean variables B, any Boolean function $h\colon 2^B \rightarrow$

TABLE I
SET OPERATIONS ON CHARACTERISTIC FUNCTIONS

Set/Operation	Characteristic function
$\emptyset$	0
U	1
$U \backslash \bar{U}$	$\neg\chi_{\bar{U}}$
$U_1 \cup U_2$	$\chi_{U_1} \vee \chi_{U_2}$
$U_1 \cap U_2$	$\chi_{U_1} \wedge \chi_{U_2}$
$U_1 = U_2$	$\chi_{U_1} \leftrightarrow \chi_{U_2}$

$\{0,1\}$ representing false and true respectively, can be expressed as

$$h = (\neg b \wedge h|_{b=0}) \vee (b \wedge h|_{b=1}) \quad b \in B \tag{1}$$

where $h|_{b=0}$ and $h|_{b=1}$ are the assignments of 0 and 1 to all occurrences of the Boolean variable b. If we apply (1) recursively to all the variables in B, a binary decision diagram (BDD) can be built to represent h. A BDD is a directed acyclic graph consisting of two types of nodes: *decision nodes* and *terminal nodes*. A terminal node can either be 0 or 1. Each decision node is labeled with a Boolean variable and has two edges to its *low-child* and *high-child*, corresponding to the cases in (1) where b is 0 and 1.

The *size* of a BDD refers to the number of decision nodes. With a fixed *variable ordering*, the size of a BDD can be reduced while it is still possible to apply logic operations on it efficiently. This is commonly referred to as reduced ordered BDDs (ROBDDs) [10]. In this paper the term BDD refers to ROBDD. The variable ordering is a major factor affecting BDD sizes. Unfortunately, finding the optimal variable ordering for a BDD is NP-complete [15]. In our framework, a number of heuristics are used to find suitable variable orderings. However, this is beyond the scope of this work and we refer the reader to [16] for the detail.

The efficiency of BDDs is mainly due to that the complexity of performing logical operations is $\mathcal{O}(|h| \cdot |g|)$ in the worst case, where $|h|$ and $|g|$ are the sizes of BDDs of h and g.

A particular operator that is used extensively in the following is the *existential quantification* of a function h over its Boolean variables. For a variable $b \in B$, the existential quantification of h is defined by $\exists b.h = h|_{b=0} \vee h|_{b=1}$. Also, if $\bar{B} = (b_1, \ldots, b_k) \subseteq B$, then $\exists \bar{B}.h$ is a shorthand notation for $\exists b_1.\exists b_2.\ldots\exists b_k.h$. In plain terms, $\exists \bar{B}.h$ denotes all those truth assignments of the variable set $B \backslash \bar{B}$ that can be extended over the set $\bar{B}$ in a way that function h is eventually satisfied.

BDDs are computationally useful for representing subsets embedded in a large set. Let U be a finite set whose n elements are represented by $\lceil \log_2 n \rceil$ Boolean variables. Given a subset $\bar{U}$ of U, its *characteristic function* $\chi_{\bar{U}} : U \rightarrow \{0,1\}$ assigns 1 to all elements $u \in \bar{U}$ and 0 to all elements $u \notin \bar{U}$. Furthermore, set operations can be carried out on characteristic functions using basic Boolean operators (see Table I).

To represent EFA E over $\mathcal{V} = \{v_1 \ldots, v_n\}$ by a characteristic function, sets of Boolean variables are needed to represent the current and updated locations and values of variables. We denote these sets of Boolean variables by B^L, $\acute{B}^L$, B^{D_i} and $\acute{B}^{D_i}$ where $i = 1, \ldots, n$. In addition, we employ another set of Boolean variables, denoted by B^{Σ}, to represent the alphabet.

For encoding a transition $\ell \xrightarrow{\sigma}_{g/a} \acute{\ell} \in \rightarrow_E$, we let $B^L(\ell)$, $\acute{B}^L(\acute{\ell})$ and $B^{\Sigma}(\sigma)$ to respectively denote the Boolean representations of ℓ, $\acute{\ell}$ and σ using the corresponding Boolean variables. Recall that $\mapsto_{\ell \xrightarrow{\sigma}_{g/a} \acute{\ell}}$ is a set of explicit state transitions where we assume that $\{\eta_1, \ldots, \eta_m\}$ is the set of variable evaluations satisfying g. Hence, the characteristic function of $\mapsto_{\ell \xrightarrow{\sigma}_{g/a} \acute{\ell}}$ is constructed as

$$\begin{aligned}\chi_{\mapsto_{\ell \xrightarrow{\sigma}_{g/a} \acute{\ell}}} &:= B^L(\ell) \wedge \acute{B}^L(\acute{\ell}) \wedge B^{\Sigma}(\sigma) \wedge \\ &\bigvee_{j=1}^{m} \Big(\bigwedge_{i=1}^{n} \big(B^{D_i}(\eta_j[i]) \wedge \acute{B}^{D_i}(a(\eta_j)[i])\big)\Big)\end{aligned} \tag{2}$$

where $B^{D_i}(\eta_j[i])$ and $\acute{B}^{D_i}(a(\eta_j)[i])$ denote the Boolean representations of current and updated values of v_i. Consequently, the characteristic function of EFA E is constructed as

$$\chi_{\mapsto_E} := \bigvee_{\ell \xrightarrow{\sigma}_{g/a} \acute{\ell} \in \rightarrow_E} \chi_{\mapsto_{\ell \xrightarrow{\sigma}_{g/a} \acute{\ell}}}. \tag{3}$$

Furthermore, the characteristic function of $\overset{\sigma}{\mapsto}_E$, denoted by $\chi_{\overset{\sigma}{\mapsto}_E}$ can be constructed as

$$\chi_{\overset{\sigma}{\mapsto}_E} := \chi_{\mapsto_E} \wedge B^{\Sigma}(\sigma). \tag{4}$$

C. Supervisory Control

SCT is a general framework for the design of supervisors for DESs. Given the model of a system to be controlled, the plant P, and the desired system behavior, the specification Sp, the minimally restrictive supervisor S can be automatically synthesized. The supervisor restricts the behavior of P such that Sp is fulfilled. Two properties are expected from such a supervisor: *controllable* and *nonblocking* [1]. Controllability typically captures safety requirements, while nonblocking is a special kind of liveness property.

In this work, the considered DESs are modeled by EFAs while the SCT synthesis is performed on the underlying FAs. Basically, supervisor synthesis starts with the generation of the closed-loop system $S_0 = P||Sp$ by using the EFSC. If the plant is given as $P_1, \ldots, P_n$, then $P = P_1|| \ldots ||P_n$. Similarly, $Sp = Sp_1|| \ldots ||Sp_m$. Note that according to Definition II.3, the composed model may have transitions where states (locations and variable evaluations) are not reachable from the initial state. Next, the analysis is performed on the underlying FA model of S_0. Specifically, an iterative removal of states and transitions is carried out through a series of reachability computations, until the remaining states are both controllable and nonblocking. The resulting system is the supervisor S, which can be either represented by FA(s) or by a set guards attached to the original plant [8]. For an elaborate exposition of the supervisory control methodology, refer to [1].

III. A MOTIVATION EXAMPLE

Consider the simplified manufacturing cell in Fig. 1 where two robots Robot 1 (left) and Robot 2 (right) book the resources in two zones in opposite order to carry out the tasks. To avoid collisions, the robots are not allowed to occupy one zone simultaneously. It is also required that Robot 1 should start before Robot 2. The goal is now to generate the minimally restrictive nonblocking supervisor.

The cell can be modeled as EFAs depicted in Fig. 2 where R_1 and R_2 model the robots while Sp models the specification. Boolean variables z_1 and z_2 indicate the resource availability in the upper and lower zones.

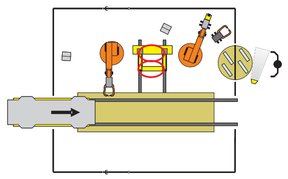

Fig. 1. A simplified manufacturing cell

Taking R_1 as an example, initially $z_1 = 0$, meaning that the resource in the upper zone is available, then event α_1 occurs and the value of z_1 is updated accordingly. Next, if the resource in the lower zone is available, i.e., $z_2 = 0$, event α_3 occurs. Meanwhile, the previously used resource is deallocated, i.e., $z_1 := 0$, and the needed one is allocated.

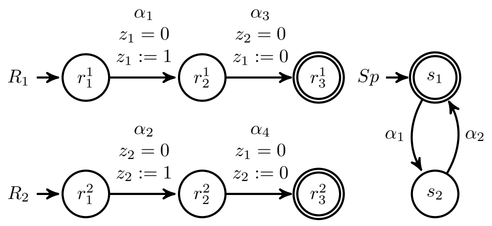

Fig. 2. The EFA model of the manufacturing cell where s_1 is both initial and marked (double-circle). For simplicity, all values of z_1 and z_2 are marked.

To get a nonblocking supervisor symbolically, the approach in [8] first constructs a monolithic BDD representation of $\mapsto_{S_0}$, where $S_0 = R_1||R_2||Sp$. Then two sets of states, namely the states reachable from the initial state and the ones coreachable from the marked states, can be computed by performing the reachability computations on this BDD. After removing the non-coreachable states from the reachable ones, the nonblocking states are obtained. As mentioned earlier, due to the large number of nodes in the intermediate BDDs, this approach might not survive from the reachability computations for relatively large systems. To alleviate this problem, the idea is to construct a set of smaller BDDs, each one corresponding to an event. In this way, the monolithic BDD is partitioned as a set of BDDs with disjunction in between.

Regarding the example, for each $\alpha_i, i = 1, \ldots, 4$, a BDD that symbolically represents the explicit state transitions labeled by α_i in S_0, denoted by $\overset{\alpha_i}{\mapsto}_{S_0}$, is constructed accordingly. Section IV shows this event-based partitioning approach formally. Subsequently, to perform reachability computations on these constructed BDDs while suppressing the sizes of intermediate BDDs, each time only a single partitioned transition relation BDD is selected and explored locally. Note that a BDD may have to be selected for multiple times in order to realize the exhaustive state-space exploration. In Section V, a traversal algorithm is presented and used to select these partial transition relations systematically to realize the structural exploration of symbolically represented state-space.

IV. Partitioning of the Full Synchronous Composition

Partitioning of the transition relation [11] has become the standard guideline for alleviating the state-space problem with large intermediate BDDs. In general this is done by splitting the transition relation into a set of *partial transition relations*, connected by disjunction or conjunction. In [13], an adaption of the disjunctive partitioning technique to FAs was introduced. However, this approach does not work for EFAs, since no mechanism is provided to keep track of the variables that are not updated. By definition, the values of these variables should remain the same values after the corresponding transitions are taken.

In this section, we present a symbolic approach to partitioning the transition relation of a modular DES modeled as EFAs. The approach constructs the set of partial transition relations on the basis of events in the alphabet. In the sequel, we do not differentiate between sub-plants and sub-specifications but focus on a set of EFAs.

Let $\mathbf{E}$ be $N \geq 2$ EFAs $E_1, \ldots, E_N$ defined over variable set $\mathcal{V} = \{v_1, \ldots, v_n\}$ and $\chi_{\mapsto_{E_1}}, \ldots, \chi_{\mapsto_{E_N}}$ be the corresponding characteristic functions of $\mapsto_{E_1}, \ldots, \mapsto_{E_n}$. For each $\sigma \in \Sigma_{\mathbf{E}}$ where $\Sigma_{\mathbf{E}} = \bigcup_{j=1}^{N} \Sigma_j$, the characteristic function of $\overset{\sigma}{\mapsto}_{||\mathbf{E}}$, denoted by $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}}$, where $||\mathbf{E} = E_1|| \ldots ||E_N$, can be constructed from $\chi_{\mapsto_{E_1}}, \ldots, \chi_{\mapsto_{E_N}}$ by the following three steps: (1) Let $\mathbf{E}^\sigma = \{E \mid E \in \mathbf{E}, \sigma \in \Sigma_{\mathbf{E}}\}$ and $||\mathbf{E}^\sigma$ denote the EFSC of EFAs in $\mathbf{E}^\sigma$. We compute the characteristic function of $\overset{\sigma}{\mapsto}_{||\mathbf{E}^\sigma}$, denoted by $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}^\sigma}}$. (2) Attach a self-loop transition labeled by σ to every location of every EFA in $\mathbf{E} \backslash \mathbf{E}^\sigma$. Let $\hat{\mathbf{E}}^\sigma$ denote the set of these modified EFAs and $||\hat{\mathbf{E}}^\sigma$ denote the EFSC of them. We then compute the characteristic function of $\overset{\sigma}{\mapsto}_{||\hat{\mathbf{E}}^\sigma}$, denoted by $\chi_{\overset{\sigma}{\mapsto}_{||\hat{\mathbf{E}}^\sigma}}$. (3) Finally $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}}$ can be computed by performing the conjunctive operation on $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}^\sigma}}$ and $\chi_{\overset{\sigma}{\mapsto}_{||\hat{\mathbf{E}}^\sigma}}$.

Computation of the characteristic function of $\overset{\sigma}{\mapsto}_{||\mathbf{E}^\sigma}$. Let us assume that $\mathbf{E}^\sigma = \{E_1, \ldots, E_M\}$, where $1 \leq M \leq N$. Hence, $\overset{\sigma}{\mapsto}_{||\mathbf{E}^\sigma}$ consists of a set of explicit state transitions that are in the form of $(\langle \ell_1, \ldots, \ell_M \rangle, \eta, \sigma, \langle \acute{\ell}_1, \ldots, \acute{\ell}_M \rangle, \acute{\eta})$ where $\eta, \acute{\eta}$ are the current and updated values of n variables. According to Definition II.3, the value of $v_i \in \mathcal{V}$ in the variable evaluation $\acute{\eta}$, i.e., $\acute{\eta}[i]$, is dependent on how it is updated during the EFSC. If v_i is not updated by any EFA, then $\acute{\eta}[i] = \eta[i]$. On the other hand, if it is updated to different values, then according to Definition II.3 the transition does not occur. By taking this into account, we further decompose the computation of $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}^\sigma}}$ into two stages. First, we define $\overset{\sigma}{\hookrightarrow}_{||\mathbf{E}^\sigma}$ as the explicit state transitions of $\overset{\sigma}{\mapsto}_{||\mathbf{E}^\sigma}$ where all the updated variable evaluations are excluded. The characteristic function of $\overset{\sigma}{\hookrightarrow}_{||\mathbf{E}^\sigma}$, denoted by $\chi_{\overset{\sigma}{\hookrightarrow}_{||\mathbf{E}^\sigma}}$, can be constructed as

$$\chi_{\overset{\sigma}{\hookrightarrow}_{||\mathbf{E}^\sigma}} := \bigwedge_{k=1}^{M} \exists \acute{B}^D . \chi_{\overset{\sigma}{\mapsto}_{E_k}} \tag{5}$$

where Boolean variable set $\acute{B}^D = \acute{B}^{D_1} \cup \ldots \cup \acute{B}^{D_n}$. In (5), the existential quantification, described in Section II-B, is performed on all $\chi_{\overset{\sigma}{\mapsto}_{E_k}}$ to remove all the Boolean variables representing the updated values of variables in $\mathcal{V}$.

Next, we deal with the updating of variables that are missing in the characteristic function of $\overset{\sigma}{\hookrightarrow}_{||\mathbf{E}^\sigma}$. In the following definitions we focus on the update of one single variable v_i between two EFAs E_1 and E_2, and then we extend the result to all variables in $\mathcal{V}$ for all EFAs in $\mathbf{E}^\sigma$.

Definition IV.1. Updated transition relation for variable v_i. For EFA E and variable v_i, the updated state transition relation for v_i through σ, denoted by $\overset{\sigma}{\mapsto}_{v_i,E}$, is defined as

$$\overset{\sigma}{\mapsto}_{v_i,E} = \{(\ell,\eta,\sigma,\acute{\ell},\acute{\eta}) \mid \ell \overset{\sigma}{\rightarrow}_{g/a} \acute{\ell} \in \rightarrow_E \text{ and } \eta \models g \text{ and } a(\eta)=\acute{\eta} \text{ and } a_i \neq \xi\}.$$

In this definition, $\overset{\sigma}{\mapsto}_{v_i,E}$ contains all the state transitions of σ where variable v_i is updated; other variables can either be updated or not. Note that $\overset{\sigma}{\mapsto}_{v_i,E}$ might be empty if for all transitions of σ, $a_i = \xi$. Moreover, we denote by the set $\overset{\sigma}{\mapsto}_E \setminus \overset{\sigma}{\mapsto}_{v_i,E}$ those transitions of σ with v_i not being updated.

Recall that, from Definition II.3, each action function a_i can be divided into four if-then constructs: (1) Both a_i^1 and a_i^2 update v_i; $a_i = a_i^1$. (2) Only a_i^1 updates v_i; $a_i = a_i^1$. (3) Only a_i^2 updates v_i; $a_i = a_i^2$. (4) None of a_i^1 and a_i^2 updates v_i; $a_i = \xi$. Subsequently we define the *interaction transition relation* for variable v_i through event σ, to represent the set of transitions in correspondence with item (1) – (4) above.

Definition IV.2. Interaction transition relation for variable v_i. Let E_1 and E_2 be two EFAs and v_i be a variable. The interaction transition relation for variable v_i through event σ, denoted by $\mathbf{I}_j^{(\sigma,v_i)}$ where $j = 1,\ldots,4$, can be defined as

$$\begin{aligned}
\mathbf{I}_1^{(\sigma,v_i)} &= \{(\langle\ell_1,\ell_2\rangle,\eta,\sigma,\langle\acute{\ell}_1,\acute{\ell}_2\rangle,\acute{\eta}_1) \mid \\
&\qquad (\ell_1,\eta,\sigma,\acute{\ell}_1,\acute{\eta}_1) \in \overset{\sigma}{\mapsto}_{v_i,E_1} \text{ and} \\
&\qquad (\ell_2,\eta,\sigma,\acute{\ell}_2,\acute{\eta}_2) \in \overset{\sigma}{\mapsto}_{v_i,E_2}\}. \\
\mathbf{I}_2^{(\sigma,v_i)} &= \{(\langle\ell_1,\ell_2\rangle,\eta,\sigma,\langle\acute{\ell}_1,\acute{\ell}_2\rangle,\acute{\eta}_1) \mid \\
&\qquad (\ell_1,\eta,\sigma,\acute{\ell}_1,\acute{\eta}_1) \in \overset{\sigma}{\mapsto}_{v_i,E_1} \text{ and} \\
&\qquad (\ell_2,\eta,\sigma,\acute{\ell}_2,\acute{\eta}_2) \in \overset{\sigma}{\mapsto}_{E_2} \setminus \overset{\sigma}{\mapsto}_{v_i,E_2}\}. \\
\mathbf{I}_3^{(\sigma,v_i)} &= \{(\langle\ell_1,\ell_2\rangle,\eta,\sigma,\langle\acute{\ell}_1,\acute{\ell}_2\rangle,\acute{\eta}_2) \mid \\
&\qquad (\ell_2,\eta,\sigma,\acute{\ell}_2,\acute{\eta}_2) \in \overset{\sigma}{\mapsto}_{v_i,E_2} \text{ and} \\
&\qquad (\ell_1,\eta,\sigma,\acute{\ell}_1,\acute{\eta}_1) \in \overset{\sigma}{\mapsto}_{E_1} \setminus \overset{\sigma}{\mapsto}_{v_i,E_1}\}. \\
\mathbf{I}_4^{(\sigma,v_i)} &= \{(\langle\ell_1,\ell_2\rangle,\eta,\sigma,\langle\acute{\ell}_1,\acute{\ell}_2\rangle,\eta) \mid \\
&\qquad (\ell_1,\eta,\sigma,\acute{\ell}_1,\acute{\eta}_1) \in \overset{\sigma}{\mapsto}_{E_1} \setminus \overset{\sigma}{\mapsto}_{v_i,E_1} \text{ and} \\
&\qquad (\ell_2,\eta,\sigma,\acute{\ell}_2,\acute{\eta}_2) \in \overset{\sigma}{\mapsto}_{E_2} \setminus \overset{\sigma}{\mapsto}_{v_i,E_2}\}.
\end{aligned}$$

By Definition IV.2, we can compute the characteristic function of the updated state transition relation of $E_1||E_2$ for v_i through σ, denoted by $\overset{\sigma}{\mapsto}_{v_i,E_1||E_2}$, as follows,

$$\chi_{\overset{\sigma}{\mapsto}_{v_i,E_1||E_2}} := \bigvee_{j=1}^{3} \chi_{\mathbf{I}_j^{(\sigma,v_i)}} \tag{6}$$

Note that the characteristic function of $\mathbf{I}_4^{(\sigma,v_i)}$ is not included in the computation, since variable v_i is not updated there.

Next, we recursively apply Definition IV.2 and the computation of (6) to all the EFAs in $\mathbf{E}^\sigma$ and denote the result by the characteristic function $\chi_{\overset{\sigma}{\mapsto}_{v_i,||\mathbf{E}^\sigma}}$. Furthermore, $\mathbf{I}_4^{(\sigma,v_i)}$ in Definition IV.2 can also be recursively applied to all EFAs in $\mathbf{E}^\sigma$. For the sake of simplicity, we stick with the same notation $\mathbf{I}_4^{(\sigma,v_i)}$ for the extension. Subsequently, we let $\overset{\sigma}{\mapsto}_{v,||\mathbf{E}^\sigma}$ be the updated state transition relation for all variables in $\mathcal{V}$ through σ and its characteristic function, denoted by $\chi_{\overset{\sigma}{\mapsto}_{v,||\mathbf{E}^\sigma}}$ can be constructed as

$$\chi_{\overset{\sigma}{\mapsto}_{v,||\mathbf{E}^\sigma}} := \bigwedge_{i=1}^{n} (\chi_{\overset{\sigma}{\mapsto}_{v_i,||\mathbf{E}^\sigma}} \vee \chi_{\mathbf{I}_4^{(\sigma,v_i)}}). \tag{7}$$

That is, $\chi_{\overset{\sigma}{\mapsto}_{v,||\mathbf{E}^\sigma}}$ contains a set of composed state transitions labeled by σ. Among the set of transitions, some variables are updated while others that are not updated will remain the same values, as expressed by the characteristic function $\chi_{\mathbf{I}_4^{(\sigma,v_i)}}$.

Finally, by the computations (5) and (7), we have

$$\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}^\sigma}} := \chi_{\overset{\sigma}{\hookrightarrow}_{||\mathbf{E}^\sigma}} \wedge \chi_{\overset{\sigma}{\mapsto}_{v,||\mathbf{E}^\sigma}}. \tag{8}$$

Computation of the characteristic function of $\overset{\sigma}{\mapsto}_{||\hat{\mathbf{E}}^\sigma}$. The second step is concerned with how locations and variables are updated for the EFAs in $\mathbf{E}\backslash\mathbf{E}^\sigma$ on occurrence of σ. It is known that when σ occurs, all the locations in $E \in \mathbf{E}\backslash\mathbf{E}^\sigma$ will remain the same while the updating of variables will follow that in the transitions labelled by σ in $\mathbf{E}^\sigma$. In this step, we start by attaching a self-loop transition labeled by σ to every location of $E \in \mathbf{E}\backslash\mathbf{E}^\sigma$. We use $\hat{\mathbf{E}}^\sigma$ and $\overset{\sigma}{\curvearrowright}_E$ to respectively denote the set of the modified EFAs and the set of self-loop transitions for every $E \in \hat{\mathbf{E}}^\sigma$. Then, by using the characteristic functions of $\overset{\sigma}{\curvearrowright}_E$, denoted by $\chi_{\overset{\sigma}{\curvearrowright}_E}$, $\chi_{\overset{\sigma}{\mapsto}_{||\hat{\mathbf{E}}^\sigma}}$ can be constructed as

$$\chi_{\overset{\sigma}{\mapsto}_{||\hat{\mathbf{E}}^\sigma}} := \bigwedge_{E\in\hat{\mathbf{E}}^\sigma} \chi_{\overset{\sigma}{\curvearrowright}_E}. \tag{9}$$

Computation of the characteristic function of $\overset{\sigma}{\mapsto}_{||\mathbf{E}}$. Consequently, by (8) and (9), $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}}$ can be constructed as

$$\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}} := \chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}^\sigma}} \wedge \chi_{\overset{\sigma}{\mapsto}_{||\hat{\mathbf{E}}^\sigma}}. \tag{10}$$

By performing the above symbolic computation for each $\sigma \in \Sigma_{\mathbf{E}}$, the symbolic representation of $\mapsto_{\mathbf{E}}$ can be equally represented as the disjunction of all the symbolically represented partial transition relations. That is,

$$\chi_{\mapsto_{||\mathbf{E}}} = \bigvee_{\sigma\in\Sigma_{\mathbf{E}}} \chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}}. \tag{11}$$

V. Efficient Reachability Computation

The reachability computation is the most fundamental challenge for synthesizing supervisors. The basic symbolic algorithm for computing reachable states requires a single BDD of the monolithic transition relation $\mapsto_{||\mathbf{E}}$, which can be constructed by the approach in [8]. Unfortunately, for many practical applications, the size of this BDD is very large. The partitioned partial transition relations, constructed in Section IV can provide a much more concise representation, but they cannot be used with the basic reachability computation. In this section, we present a novel algorithm that performs the reachability search for partial transition relations and meanwhile, reduces the sizes of intermediate BDDs.

A. Event-based Reachability Algorithm

Taking as input the characteristic function of the initial state and the set of partial transition relations, Algorithm 1 computes the characteristic function of all reachable states of the composed system $||\mathbf{E}$. More specifically, the algorithm maintains a set of active partial transition relations, W_k, during the execution. For each iteration, depending on some heuristics $\mathbb{H}$, similar to the heuristics described [14], the partial transition relation where new states are most likely to be reached by the existing ones, is selected and a local exhaustive reachability search (Algorithm 2) is performed on it. If more states are reached on $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}}$, the dependent transition relation sets $\mathbf{D}_e^\sigma$ and $\mathbf{D}_v^\sigma$, defined in Definition V.1 and V.2, are appended to W_k for further exploration. In particular, $\mathbf{D}_e^\sigma$ contains all the partial transition relations with all events σ' being topologically next to σ in $\mathbf{E}$; $\mathbf{D}_v^\sigma$ contains all the transition relations with all events σ' being identified with respect to the variables in the associated guards that are possibly updated by the transitions of σ. The algorithm terminates when there are no active transition relations left in W_k.

Definition V.1. Event dependent transition relation set. Let $\mathbf{E} = \{E_1, \ldots, E_N\}$ where $N \geq 2$, $\Sigma_{\mathbf{E}} = \bigcup_{j=1}^N \Sigma_j$ and $||\mathbf{E} = E_1|| \ldots ||E_N$. The event dependent transition relation set of σ, denoted by $\mathbf{D}_e^\sigma$ is defined as

$$\mathbf{D}_e^\sigma = \{\overset{\sigma'}{\mapsto}_{||\mathbf{E}} |\ \sigma' \in \Sigma_e^\sigma \text{ and } \sigma' \neq \sigma\},$$

where $\Sigma_e^\sigma \subseteq \Sigma_{\mathbf{E}}$ is the successor event set of σ in accordance with locations, defined as

$$\Sigma_e^\sigma = \{\sigma' \in \Sigma_{\mathbf{E}} \mid \exists E \in \mathbf{E} \text{ s.t. } (\ell, \sigma, g, a, \ell') \in \rightarrow_E \\ \text{and } (\ell', \sigma', g', a', \ell'') \in \rightarrow_E\}.$$

Definition V.2. Variable dependent transition relation set. Let $\mathbf{E} = \{E_1, \ldots, E_N\}$ be the set of $N \geq 2$ EFAs over $\mathcal{V} = \{v_1, \ldots, v_n\}$, $\Sigma_{\mathbf{E}} = \bigcup_{j=1}^N \Sigma_j$ and $||\mathbf{E} = E_1|| \ldots ||E_N$. The variable dependent transition relation set of σ, denoted by $\mathbf{D}_v^\sigma$, is defined as

$$\mathbf{D}_v^\sigma = \{\overset{\sigma'}{\mapsto}_{||\mathbf{E}} |\ \sigma' \in \Sigma_v^\sigma \text{ and } \sigma' \neq \sigma\}$$

where $\Sigma_v^\sigma \subseteq \Sigma_{\mathbf{E}}$ is a set of events related to the alteration of guard evaluations. As a transition of σ is taken, variables may be updated, which might alter the guard evaluations for other transitions. The set of events for these transitions is defined as

$$\Sigma_v^\sigma = \{\sigma' \in \Sigma_{\mathbf{E}} \mid \exists v_i \in \mathcal{V} \text{ and } \exists E_j \in \mathbf{E} \text{ and } \exists E_k \in \mathbf{E} \text{ s.t.} \\ (\ell_j, \sigma, g^j, a^j, \ell_j') \in \rightarrow_{E_j} \text{ and } (\ell_k, \sigma', g^k, a^k, \ell_k') \in \rightarrow_{E_k} \\ \text{and } a_i^j \neq \xi \text{ and } v_i \in var(g^k)\}$$

where $var(g^k)$ denotes the variables that appear in guard g^k.

B. Proof for the correctness of Algorithm 1 in Section V

Next, we prove the correctness of Algorithm 1 by proving that it computes all the reachable states.

Lemma V.1. *At iteration k of Algorithm 1, W_k contains all active transition relations that are sufficient for the current reachable state set χ_Q^k to reach more states in one step.*

Algorithm 1 Event-based Reachability

```
1:  input $W_0 := \{\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}} \mid \sigma \in \bigcup_{j=1}^N \Sigma_j\}$, $q_0 := (\langle \ell_1^0, \ldots, \ell_N^0 \rangle, \eta_0)$
2:  let $k := 0, \chi_Q^k := \chi_{\{q_0\}}$
3:  repeat
4:      $k := k + 1$
5:      $\mathbb{H}$: Pick and remove $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}} \in W_{k-1}$
6:      $\chi_Q^k :=$ Local Reachability $(\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}}, \chi_Q^{k-1})$
7:      if $\neg(\chi_Q^k \leftrightarrow \chi_Q^{k-1})$ then
8:          $W_k := W_{k-1} \cup \{\chi_{\overset{\sigma'}{\mapsto}_{||\mathbf{E}}} \mid \sigma' \in \mathbf{D}_e^\sigma \cup \mathbf{D}_v^\sigma\}$
9:      end if
10: until $W_k = \emptyset$
11: return $\chi_Q^k$
```

Algorithm 2 Local Reachability

```
1: input $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}}, \chi_Q$
2: let $m := 0, \chi_Q^m := \chi_Q$
3: repeat
4:     $m := m + 1$
       /*$B_{\mathbf{E}}^L$ is the set of Boolean variables representing locations of
       EFAs in $\mathbf{E}$; $B^D = B^{D_1} \cup \ldots \cup B^{D_n}$ is the set of Boolean
       variables representing the current values of $n$ variables; $B^\Sigma$
       is the set of Boolean variables representing $\Sigma_{\mathbf{E}}$. */
5:     $\chi_Q^m := \chi_Q^{m-1} \vee \exists(B_{\mathbf{E}}^L \cup B^D \cup B^\Sigma).(\chi_Q^{m-1} \wedge \chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}})$
6: until $\chi_Q^m \leftrightarrow \chi_Q^{m-1}$
7: return $\chi_Q^m$
```

Proof: The lemma can be proved by induction.

The basic step: When $k = 0$, the current reachable state set χ_Q^0 comprises merely the initial state, while W_0 initially contains all partial transition relations for the considered system. Therefore, the lemma holds for $k = 0$.

The inductive step: We assume that at iteration k where $k \geq 1$, the lemma holds. Next we prove by contradiction that the lemma holds at iteration $k + 1$. Assuming that at iteration $k + 1$, there exists $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}} \notin W_k$ but it contains new states that can be reached from χ_Q^k. Since the lemma holds at iteration k, W_{k-1} contains all transition relations leading to reachable states from χ_Q^{k-1}. Let us assume that there exists a subset of W_{k-1}, denoted by $\hat{W}_{k-1}$ that consists of all partial transition relations where more states can be reached from χ_Q^{k-1}. (i) Consider that $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}}$ is removed before iteration k. Then $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}} \notin W_{k-1}$ at iteration k. Suppose the algorithm selected another transition relation $\chi_{\overset{\sigma'}{\mapsto}_{||\mathbf{E}}}$ at iteration k. An exhaustive reachability search is then performed on it. If $\chi_{\overset{\sigma'}{\mapsto}_{||\mathbf{E}}} \in \hat{W}_{k-1}$, then new reachable states are found by Algorithm 2 and Algorithm 1 puts these partial transition relations of $\mathbf{D}_e^{\sigma'}$ and $\mathbf{D}_v^{\sigma'}$ back into the relation set. If $\chi_{\overset{\sigma'}{\mapsto}_{||\mathbf{E}}} \notin \hat{W}_{k-1}$, the reachable state set will be unchanged. By the assumption, if $\chi_{\overset{\sigma'}{\mapsto}_{||\mathbf{E}}} \in \hat{W}_{k-1}$, we have $\chi_{\overset{\sigma}{\mapsto}_{\mathbf{E}}} \notin \mathbf{D}_e^{\sigma'} \cup \mathbf{D}_v^{\sigma'}$. Then, there will be no more state that can be reached from χ_Q^k in $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}}$ at iteration $k + 1$, which contradicts the assumption. If $\chi_{\overset{\sigma'}{\mapsto}_{||\mathbf{E}}} \notin \hat{W}_{k-1}$, the partial transition relations that lead to more reachable states at iteration $k + 1$ are all in W_k. This also contradicts the assumption. (ii) Consider that $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}}$ is removed from W_{k-1} at iteration k. If $\chi_{\overset{\sigma}{\mapsto}_{\mathbf{E}}} \in \hat{W}_{k-1}$, there exists some states that are reached in one step from χ_Q^{k-1}. Since Algorithm 2 is an exhaustive search, at iteration $k+1$, there can not be more new states that are found in $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}}$. This contradicts the assumption. If $\chi_{\overset{\sigma}{\mapsto}_{||\mathbf{E}}} \notin \hat{W}_{k-1}$, the reachable state set is unchanged, which again contradicts the assumption.

To sum up, on the basis of the two cases above we can prove that the lemma holds at the iteration $k+1$. ■

Theorem V.1. *Algorithm 1 terminates in a finite time and the output is the complete set of reachable states.*

Proof: In each step of Algorithm 1, zero or more states are added to the set of reachable states. Since the total set of states is finite, the reachable state set can only grow in a finite number of times. When no new state is found, one transition relation is removed from W_k at each time. Since W_k is also finite, the algorithm terminates in finite time. If the algorithm terminates, by Lemma V.1, W_k will hold all partial transition relations that can be used to find more reachable states. Since the algorithm has terminated because of the emptiness of W_k, no more state can be reached from χ_Q^k. Therefore, Algorithm 1 returns the complete set of reachable states. ■

C. Algorithm Efficiency

It is known that the minimally restrictive supervisor synthesis is NP-hard [17], which implies in the worst case no solution can be faster than the brute-force one. However, by leveraging the efficiencies of symbolic computation, this issue can be alleviated for many practical systems. After representing sets by BDDs, the computational complexity of synthesis is no longer dependent on the number of states but the sizes of relevant BDDs. Therefore, a symbolic algorithm involving BDDs with smaller sizes, tends to be more efficient.

As mentioned in Section II that for two BDDs h and g, the computational complexity of performing logic operations on h and g is $\mathcal{O}(|h| \cdot |g|)$ in the worst case. To expand the reachable state set, the approach of [8] performs the operation $\chi_Q^{k-1} \wedge \chi_{\mapsto_{||\mathbf{E}}}$ where χ_Q^{k-1} denotes the BDD representation of the existing reachable states. However, the monolithic $\chi_{\mapsto_{||\mathbf{E}}}$ usually contains numerous nodes and therefore the operation will be time and memory consuming. Algorithm 1, on the other hand, has a lower space demand, since each time the algorithm only manipulates one selected partial transition relation that for most cases is much smaller than the monolithic one.

In contrast to the straightforward way in [11] that reachable states are expanded by performing one-step reachability search iteratively on each partitioned transition relation, the strategy employed by the proposed algorithm can effectively reduce the sizes of intermediate BDDs. In particular, we notice that Algorithm 1 is based on two nested loops. The inner loop, as depicted in Algorithm 2, performs exhaustive breadth-first search on the selected partial transition relation for the corresponding event. The local exhaustive search can make the BDD representing the intermediate reachable states enter its saturated shape earlier, thus redundant nodes are eliminated by the reduction rules, whereas the technique of [11] generates new nodes in a near random fashion. Moreover, when more reachable states are found on a partial transition relation in the inner loop, by Definition V.1 and V.2, the dependent partial transition relations are added into set W_k. The algorithm then switches to the most related region of the state-space for further exploration. In this way, not only the fix-point can be reached earlier, but also the intermediate BDDs with more redundant nodes could be attained.

TABLE II
COMPARISON BETWEEN THE EXPLICIT STATE ENUMERATION FA SYNTHESIS AND THE SYMBOLIC SYNTHESIS

Model	Classical Synthesis (s)	Symbolic Synthesis (s)
CMT (1, 5)	0.003	0.02
CMT (5, 1)	0.009	0.04
CMT (1, 7)	0.06	0.39
CMT (7, 1)	0.02	0.06
CMT (3, 3)	25	3.1
CMT (5, 5)	M.O.	59
EDP (5, 10)	4.2	0.4
EDP (5, 50)	M.O.	0.55
RAS	1.55	0.13
RAS-EH	41.33	0.87
BSP	T.O.	210
AGVs	M.O.	0.87
PME	3.42	0.14

M.O. indicates memory out and T.O. indicates time out (15 min).

VI. CASE STUDIES

The proposed approach has been integrated in the synthesis algorithm in *Supremica* [18] that uses *JavaBDD* [19] as the BDD package. In this section, it is applied to the following set of benchmark examples to demonstrate the efficiency[1]: the resource allocation system (RAS) together with its extension (RAS-EH) [20], the iron ball sorting process (BSP) [21], automated guided vehicles (AGVs) [22], the parallel manufacturing example (PME) [23], cat and mouse tower (CMT) and extended Dinning Philosophers (EDP) [24].

We first compare the performance between the classic explicit state enumeration approach and the presented approach. As Table II shows, for those systems with smaller state-spaces, the state enumeration method is slightly better than the symbolic approach where the BDD representation may contain more nodes than the states. As the systems become larger and more complex, e.g., AGVs, the symbolic approach obviously outperforms the standard approach that suffers from enumerating a large number of states explicitly when handling systems with the state-space 10^6. Regarding BSP, the conversion from EFAs to FAs that are the input for the classical approach, takes long time (more than 15 minutes).

The second comparison is made between the monolithic symbolic approach of [8] and the presented approach. It can be observed from Table III that both the monolithic and the partitioning approaches can handle AGVs and synthesize the supervisor in a short time. However, by comparing the maximum number of BDD nodes, i.e, BDD Peak, during the reachability computation, which can express the maximum memory usage, the monolithic approach needs nine times more memory than the partitioning approach. Regarding the iron ball sorting process, even though the final number of supervisor states is only 2026, the intermediate BDDs are large due to

[1]The experiments are carried out on a standard PC (Intel Core 2 Quad CPU, 2GB RAM) running Ubuntu 12.04.

TABLE III
COMPARISON BETWEEN TWO SYMBOLIC SYNTHESIS APPROACHES

			BDD Monolithic Approach		BDD Partitioning Approach	
Model	Reachable States	Supervisor states	BDD Peak (R)	Computation Time (s)	BDD Peak (R)	Computation Time (s)
RAS	1.19×10^4	8761	2826	0.49	215	0.13
RAS-EH	1.84×10^6	0.68×10^6	42314	18.67	2275	0.87
BSP	2026	2026	M.O.	–	16640	210
AGVs	2.29×10^7	1.15×10^7	9663	3.60	1001	0.87
PME	8.13×10^5	0.46×10^5	1022	0.24	225	0.14
CMT (1, 5)	605	579	447	0.01	255	0.02
CMT (5, 1)	1056	76	635	0.06	590	0.04
CMT (1, 7)	1198	1156	801	0.10	321	0.39
CMT (7, 1)	2710	155	1074	0.15	974	0.06
CMT (3, 3)	2.96×10^5	1.64×10^5	8761	5.0	1963	3.1
CMT (5, 5)	1.07×10^{10}	3.15×10^9	–	T.O.	17353	59
EDP (5, 10)	1.6×10^5	1596	1157	0.5	134	0.4
EDP (5, 50)	3.46×10^8	1.38×10^5	7743	1.25	178	0.55
EDP (5, 100)	1.05×10^{10}	1.05×10^6	16915	80	192	1.3
EDP (5, 200)	3.28×10^{11}	8.20×10^6	–	T.O.	206	3.5

M.O. indicates memory out during the reachability search (due to large intermediate BDDs) and T.O. indicates time out (15 min).

the high complexity. The monolithic approach fails to explore the state-space, while the partitioning approach obtains the supervisor within four minutes. For the last two benchmark examples, cat and mouse tower and extended dining philosophers, the partitioning approach can also handle relatively large problem instances within an acceptable time. As mentioned before, since the proposed partitioning algorithm is on a basis of the alphabet, more iterations than the monolithic algorithm might be needed to reach the final fixed point. However, the intermediate BDDs produced during the computation are smaller in terms of numbers of BDD nodes.

VII. CONCLUSIONS

This paper presents a symbolic supervisor synthesis approach for DESs modeled by EFAs. Experimental results have shown that the proposed algorithm has better performance and scalability than the previously presented work. We believe this can further promote the practical usage of EFAs for modeling and analyzing large scale DESs. In our future work, we will seek to improve the performance of the algorithm by further developing the heuristics for selecting partial transition relations. We also consider the possibility of combining our approach with hierarchical approaches such as STS, which is presented in [5].

REFERENCES

[1] P. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[2] L. E. Holloway, B. H. Krogh, and A. Giua, "A survey of Petri net methods for controlled discrete event systems," *Discrete Event Dynamic Systems*, vol. 7, no. 2, pp. 151–190, 1997.

[3] J. C. M. Baeten, D. A. Van Beek, B. Luttik, J. Markovski, and J. Rooda, "A process-theoretic approach to supervisory control theory," in *American Control Conference*, 2011, pp. 4496–4501.

[4] H. Marchand and B. Gaudin, "Supervisory control problems of hierarchical finite state machines," in *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 2, 2002, pp. 1199–1204.

[5] C. Ma and W. M. Wonham, "Nonblocking supervisory control of state tree structures," *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 782–793, 2006.

[6] D. Harel, "Statecharts: A visual formalism for complex systems," *Sci. Comput. Program.*, vol. 8, no. 3, pp. 231–274, 1987.

[7] M. Sköldstam, K. Åkesson, and M. Fabian, "Modeling of discrete event systems using finite automata with variables," in *Proceedings of the 46st IEEE Conference on Decision and Control*, 2007, pp. 3387–3392.

[8] S. Miremadi, B. Lennartson, and K. Åkesson, "A BDD-based approach for modeling plant and supervisor by extended finite automata," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 6, pp. 1421–1435, 2012.

[9] S. B. Akers, "Binary Decision Diagrams," *IEEE Transactions on Computers*, vol. 27, pp. 509–516, 1978.

[10] R. E. Bryant, "Symbolic Boolean manipulation with Ordered Binary Decision Diagrams," *ACM Comput. Surv.*, vol. 24, no. 3, pp. 293–318, 1992.

[11] J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan, and D. L. Dill, "Symbolic model checking for sequential circuit verification," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 13, no. 4, pp. 401–424, 1994.

[12] G. Hoffmann and H. Wong-Toi, "Symbolic synthesis of supervisory controllers," in *American Control Conference*, 1992, pp. 2789–2793.

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

[14] Z. Fei, S. Miremadi, K. Å kesson, and B. Lennartson, "Symbolic state-space exploration and guard generation in supervisory control theory," in *Agents and Artificial Intelligence – Communications in Computer and Information Science*. Springer, 2013, vol. 271, pp. 161–175.

[15] B. Bollig and I. Wegener, "Improving the variable ordering of OBDDs is NP-complete," *IEEE Trans. Comput.*, vol. 45, no. 9, pp. 993–1002, 1996.

[16] A. Aziz, S. Tasiran, and R. K. Brayton, "BDD variable ordering for interacting finite state machines," in *Proceedings of the 31st annual Design Automation Conference*, 1994, pp. 283–288.

[17] P. Gohari and W. Wonham, "On the complexity of supervisory control design in the RW framework," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 30, no. 5, pp. 643–652, 2000.

[18] K. Åkesson, M. Fabian, H. Flordal, and R. Malik, "Supremica - an integrated environment for verification, synthesis and simulation of discrete event systems," in *the 8th International Workshop on Discrete Event Systems*, 2006, pp. 384–385.

[19] "JavaBDD." [Online]. Available: javabdd.sourceforge.net

[20] Z. Fei, S. Miremadi, and K. Åkesson, "Modeling sequential resource allocation systems using extended finite automata," in *Proceedings of the 7th IEEE Conference on Automation Science and Engineering*, 2011, pp. 444–449.

[21] G. Cengic and K. Åkesson, "A control software development method using iec 61499 function blocks, simulation and formal verification," in *Proceedings of the 17th IFAC World Congress*, 2008, pp. 22–27.

[22] L. E. Holloway and B. H. Krogh, "Synthesis of feedback control logic for a class of controlled Petri nets," *IEEE Transactions on Automatic Control*, vol. 35, no. 5, pp. 514–523, 1990.

[23] R. J. Leduc, "Hierarchical interface-based supervisory control," Ph.D. dissertation, Electrical and Computer Engineering, Toronto, Canada, 2002.

[24] S. Miremadi, K. Åkesson, M. Fabian, A. Vahidi, and B. Lennartson, "Solving two supervisory control benchmark problems using Supremica," in *the 9th International Workshop on Discrete Event Systems*, 2008, pp. 131–136.