A Symbolic Approach to Large-Scale Discrete Event Systems Modeled as Finite Automata with Variables

Z. Fei, S. Miremadi, K. Åkesson and B. Lennartson

***Abstract*— The state-space explosion problem, resulting from the reachability computation of the synthesis task, is one of the main obstacles preventing the supervisory control theory (SCT) from having an industrial breakthrough. To alleviate this problem, a well-known strategy is to utilize binary decision diagrams (BDDs) to compute supervisors symbolically. Based on this principle, we present in this paper an efficient reachability approach to large-scale discrete event systems modeled as finite automata with variables. By making use of the disjunctive partitioning technique, the proposed approach partitions the transition relation of a considered system into a set of partial transition relations according to included events. Then those partial transition relations are selected systematically to perform the reachability computation. Experimental results show that more iterations might be required to compute the fixed point, but the intermediate BDDs are smaller. The approach has been implemented in the supervisory control tool Supremica and the efficiency is demonstrated on a set of industrially relevant benchmark problems.**

I. Introduction

The analysis of discrete event systems (DESs) has been paid extensive attention by researchers and scientists in the computer science community. One typical analysis approach is to utilize formal verification techniques, such as model checking, to verify whether considered systems fulfill given specifications or not. However, from the control engineering point of view, instead of verifying the correctness of a DES model, a controller, which automatically conducts the system behavior without violating specifications, is a necessity. The supervisory control theory (SCT) [1], [2] provides a control-theoretic framework for control engineers to design such a safety device, referred to as the *supervisor* of a system. Given a DES model to be controlled, the *plant*, and the intended behavior, the *specification*, the supervisor can be automatically synthesized, guaranteeing that the closed-loop system always achieves the given specification.

In [3], a framework was presented where users can both model a system and obtain the supervisor in the form of *extended finite automata* (EFAs) [4], which is an augmentation of an ordinary automata extended with variables. By taking the advantage of EFAs, more compact and comprehensible system models can be obtained. In addition, instead of representing the supervisor as a single automaton, the guard generation procedure provided in the framework can extract a set of logic formulas. Those extracted formulas, referred to as guards, are attached to the corresponding transitions of original models, which results in a modular representation of the supervisor.

Whereas the aforementioned framework allows compact representation of large state-spaces, when it comes to analysis, the number of states are not affected and could potentially cause the *state-space explosion* problem that typically occurs when the behavior of interacting sub-systems is studied. To alleviate the state-space explosion problem, a well-known strategy is to symbolically represent system models and compute supervisors by using *binary decision diagrams* (BDDs) [5], [6], [7], [8], [9], [10]. In [3], a BDD-based synthesis approach was presented. Particularly, based on a plant and a specification modeled by EFAs, initially, the EFAs are encoded as BDDs. Afterwards, the corresponding BDD for the states of the monolithic supervisor is computed iteratively. However, the main problem of such a monolithic approach is that during the reachability computations, the number of nodes in the intermediate BDDs might be significantly large.

To reach significant BDD reduction, it is crucial to explore the search space in an intelligent way. The key is to impose structure on the state-space exploration. Moreover, to realize such an intelligent state-space exploration, an important ingredient is the use of *partitioning techniques*, which was rigorously defined in [11]. In [12] and [13], a straightforward but non-trivial symbolic reachability approach was presented in the context of SCT. The approach, based on the disjunctive partitioning technique, represents the monolithic transition relation of a fully synchronized DES by a collection of partial transition relations. However, these approaches are based on finite automata without the introduction of variables. At the time of writing this paper, to the knowledge of the authors, yet no work has been presented to adapt these partitioning techniques to DESs with variables.

In the context of the aforementioned research developments, motivated by the above remarks, in this paper, we present an alternative symbolic reachability approach. By making use of the disjunctive partitioning technique, the proposed approach partitions the transition relation of a considered system into a set of partial transition relations according to included events. Then those partial transition relations are selected systematically to perform the reachability computation.

The paper has three main contributions:

- Suggesting a symbolic way to partition DESs modeled as EFAs by using the disjunctive partitioning technique.
- Proposing a straightforward algorithm to realize the structural state-space exploration.
- Integrating this approach with our modeling framework and demonstrating the efficiency on a set of industrially relevant benchmark examples.

II. PRELIMINARIES

In this section, some preliminaries used throughout the rest of the paper are provided and briefly explained.

A. Extended Finite Automata

Definition II.1 (Extended Finite Automata)**.** An extended finite automaton E is a 4-tuple

$$E = \langle L^E \times V, \Sigma^E, \rightarrow, (\ell_0^E, v_0)\rangle,$$

where:

- $L^E \times V$ is the extended finite set of states, denoted by Q, where L^E is a set of *locations* and V is the domain of definition of the *variables*;
- Σ^E is a non-empty finite set of events i.e. the alphabet;
- $\rightarrow \subseteq L^E \times \Sigma^E \times \mathcal{G} \times \mathcal{A} \times L^E$ is the transition relation where $\mathcal{G}$ is the set of guard predicates over V and $\mathcal{A} = \{a \mid a\colon$ a function from V to $V\}$ is a collection of action functions;
- $(\ell_0^E, v_0) \in L^E \times V$ is the initial state.

The finite set $V = V^1 \times \ldots \times V^n$ is the domain of definition of an n-tuple of variables $v = (v^1, \ldots, v^n)$ with the initial values $v_0 = (v_0^1, \ldots, v_0^n) \in V$. A *guard* $g(v)$ is a predicate over the variables that relate each element of V to either 1 (`true`) or 0 (`false`). Actions are written as

$$\acute{v}\colon = a(v) = (a^1(v), \ldots, a^n(v)), \text{where } \acute{v} \in V.$$

The symbol ξ is used to denote implicit actions that do not update the values of variables. For instance, if $a^i(v) = \xi$, it means that action a^i does not update variable v^i, i.e. $\acute{v}^i = v^i$.

For convenience, the states (locations and variable values) can be explicitly written out in system transitions according to the following definition.

Definition II.2 (Explicit State Transition Relation)**.** Let $E = \langle L^E \times V, \Sigma^E, \mapsto, (\ell_0^E, v_0)\rangle$ be an EFA. The explicit state transition relation of E is defined as

$$\begin{aligned}\mapsto_E \quad \triangleq \quad & \{(\ell^E, v, \sigma, \acute{\ell}^E, \acute{v}) \in L^E \times V \times \Sigma \times L^E \times V \mid \\ & \exists \ell^E \xrightarrow{\sigma}_{g/a} \acute{\ell}^E\colon v \in \mathsf{SAT}\mathcal{G}(g) \wedge (v, \acute{v}) \in \mathsf{SAT}\mathcal{A}(a)\},\end{aligned}$$

where v and $\acute{v}$ are the values of the variables before and after executing the transition, respectively; $\mathsf{SAT}\mathcal{G}$ denotes the set of variable assignments that satisfies the guard $g(v)$,

$$\mathsf{SAT}\mathcal{G}(g) \triangleq \{v \in V \mid v \vDash g\};$$

and $\mathsf{SAT}\mathcal{A}$ denotes the following set:

$$\mathsf{SAT}\mathcal{A}(a) \triangleq \{(v, \acute{v}) \in V \times V \mid \acute{v} = a(v)\}.$$

For brevity, we denote the explicit representation of a transition $\ell \xrightarrow{\sigma}_{g/a} \acute{\ell}$ by $\mapsto_{\ell \xrightarrow{\sigma}_{g/a} \acute{\ell}}$. Additionally, since we are interested in deterministic systems, we merely focus on deterministic EFAs. In the sequel, for the sake of brevity, we simply write EFAs for deterministic EFAs.

The composition of two EFAs is defined by the *extended full synchronous composition (EFSC)*.

Definition II.3 (Extended Full Synchronous Composition)**.** Let $E_k = \langle L^{E_k} \times V, \Sigma^{E_k}, \rightarrow_{E_k}, (\ell_0^{E_k}, v_0)\rangle, k = 1, 2$, be two EFAs with the shared variables $v = (v^1, \ldots, v^n)$. The Extended Full Synchronous Composition (EFSC) of E_1 and E_2 is

$$E_1 \parallel E_2 = \langle L^{E_1} \times L^{E_2} \times V, \Sigma^{E_1} \cup \Sigma^{E_2}, \rightarrow, (\ell_0^{E_1}, \ell_0^{E_2}, v_0)\rangle$$

where the state transition relation $\rightarrow$ is defined as

1) $(\ell^{E_1}, \ell^{E_2}) \xrightarrow{\sigma}_{g/a} (\acute{\ell}^{E_1}, \acute{\ell}^{E_2}), \sigma \in \Sigma_1 \cap \Sigma_2$ if
   $\exists \ell^{E_1} \xrightarrow{\sigma}_{g_1/a_1} \acute{\ell}^{E_1} \in \rightarrow_{E_1}$ and
   $\exists \ell^{E_2} \xrightarrow{\sigma}_{g_2/a_2} \acute{\ell}^{E_2} \in \rightarrow_{E_2}$ such that:
   - $g = g_1 \wedge g_2$,
   - For $i = 1, \ldots, n$ and $\forall v \in V$:
   $$a^i(v) = \begin{cases} a_1^i(v) & \text{if } a_1^i(v) = a_2^i(v) \\ a_1^i(v) & \text{if } a_2^i(v) = \xi \\ a_2^i(v) & \text{if } a_1^i(v) = \xi \\ v^i & \text{otherwise} \end{cases}$$
2) $(\ell^{E_1}, \ell^{E_2}) \xrightarrow{\sigma}_{g/a} (\acute{\ell}^{E_1}, \acute{\ell}^{E_2}), \sigma \in \Sigma_1 \backslash \Sigma_2$ if
   $(\ell^{E_1}, \sigma, g, a, \acute{\ell}^{E_1}) \in \rightarrow_{E_1}$ and $\ell^{E_2} = \acute{\ell}^{E_2}$;
3) $(\ell^{E_1}, \ell^{E_2}) \xrightarrow{\sigma}_{g/a} (\acute{\ell}^{E_1}, \acute{\ell}^{E_2}), \sigma \in \Sigma_2 \backslash \Sigma_1$ if
   $(\ell^{E_2}, \sigma, g, a, \acute{\ell}^{E_2}) \in \rightarrow_{E_2}$ and $\ell^{E_1} = \acute{\ell}^{E_1}$.

B. Supervisory Control Theory

As described in Section I, the goal of SCT [1], [2] is to automatically synthesize a minimally restrictive supervisor S which guarantees that the behavior of the plant P always fulfills the given specification Sp. Notice that if the plant is given as a number of sub-plants $P_1, \ldots, P_n$, the plant $P = P_1 \parallel \ldots \parallel P_n$. Similarly, $Sp = Sp_1 \parallel \ldots \parallel Sp_m$. For each sub-specification Sp_i, $\Sigma^{Sp_i} \subseteq \Sigma^P$, meaning the specification can not specify more than what the plant can achieve. Within the theory, some states of an automaton E, typically a specification, are identified as *marked states* Q_m^E. The marked states are the states that are desired to be reached from the initial state. The set of marked states of a composed automaton $E_1 \parallel E_2$ is the cartesian product of the corresponding sets of marked states. In addition, the alphabet is divided into two disjoint subsets, the controllable event set Σ_c, and the uncontrollable event set Σ_u.

In SCT, the supervisor of a DES to be synthesized is assumed to be *minimally restrictive*, meaning that the plant is given the greatest amount of freedom to generate events. Moreover, there are two properties that the supervisor ought to have:

- *Controllability*: The supervisor S is never allowed to disable any uncontrollable event that might be generated by the plant P.
- *Non-blocking*: The supervisor S guarantees that at least one marked state can be reached from every state.

The supervisory synthesis starts by generating the system $S_0 = P \parallel Sp$ and detecting a set of initiallly uncontrollable states. Through a series of reachability computations, forbidden states are iteratively excluded from Q^{S_0} until the remaining states are both controllable and nonblocking. The resulting system is the supervisor S and all of the included

states are hereby called *safe states*, denoted by Q^S. refer to [13] for more details.

C. Binary Decision Diagrams

Binary Decision Diagrams (BDDs) [5], [6], compact and operation-efficient data structures for representing Boolean functions, have proven to be a powerful technique to combat the state-space explosion problem. Given a set of Boolean variables B, a BDD is a Boolean function $h\colon 2^B \to \{0,1\}$, which can be expressed using Shannon's decomposition [14]:

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

where $h\,|_{b_j=0}$ and $h\,|_{b_j=1}$ refer to assignment 0 and 1 to all occurrences of the Boolean variable b_j, respectively. A BDD is represented as a directed acyclic graph, which consists of two types of nodes: *decision nodes* and *terminal nodes*. A terminal node can either be *0-terminal* or *1-terminal*. Each decision node is labeled by a Boolean variable and has two edges to its *low-child* and *high-child*. The low- and high-child corresponds to the cases in the above equation where b_j is 0 and 1 respectively. The *size* of a BDD refers to the number of decision nodes. More details can be found in [15].

Given an EFA E, BDDs can be used to represent the explicit transition relation II.2. The key point is to make used of the *characteristic function*:

Definition II.4 (Characteristic Function)**.** Let W be a finite set so that $W \subseteq U$, where U is the finite universal set. A *characteristic function* $\chi_W\colon U \to \mathbb{B}$ is defined by

$$\chi_W(a) = \begin{cases} 1 & \text{iff } a \in W \\ 0 & \text{iff } a \notin W \end{cases} \tag{1}$$

Let n be the number of elements in U. In practice its elements can be represented by binary m-tuples in $\mathbb{B}^m (m = \lceil \log_2^n \rceil)$. Hence, an injective function $\theta\colon U \to \mathbb{B}^m$ is used to map the elements in U to elements in $\mathbb{B}^m$:

$$\chi_W(a) = \bigvee_{w \in W} a \leftrightarrow \theta(w), \tag{2}$$

where $\leftrightarrow$ on two m-tuples v_1 and v_2 is defined as

$$v_1 \leftrightarrow v_2 \triangleq \bigwedge_{0 \le i < m} (v_1^i \leftrightarrow v_2^i), \tag{3}$$

where v^i denotes the i:th element in the binary m-tuple v.

Based on Definition II.4, the characteristic function for one element of the explicit state transition relation, $\mapsto_{\ell \xrightarrow{\sigma}_{g/a} \acute{\ell}}$ can be constructed as:

$$\begin{aligned} \chi_{\mapsto_{\ell \xrightarrow{\sigma}_{g/a} \acute{\ell}}}(b^{V^1}, \ldots, b^{V^n}, \acute{b}^{V^1}, \ldots, \acute{b}^{V^n}, b^L, \acute{b}^L, b^\Sigma) = \\ \Bigg(\bigvee_{(v,\acute{v}) \in \mathsf{SAT}\mathcal{A}(a) | v \in \mathsf{SAT}\mathcal{G}(g)} \\ \bigwedge_{i=1}^{n} \left(b^{V^i} \leftrightarrow \theta(v^i) \ \wedge \ \acute{b}^{V^i} \leftrightarrow \theta(\acute{v}^i) \right) \Bigg) \ \wedge \\ b^L \leftrightarrow \theta(\ell) \ \wedge \ \acute{b}^L \leftrightarrow \theta(\acute{\ell}) \ \wedge \ b^\Sigma \leftrightarrow \theta(\sigma), \end{aligned} \tag{4}$$

where b^Σ denotes the Boolean variables representing the alphabet while b^L and $\acute{b}^L$ are two different sets of Boolean variables representing the current and updated locations. For an EFA where n variables are defined, b^{V^i} and $\acute{b}^{V^i}$ denote the current and updated integer values of the i:th variable. In our framework, integers are represented as the two's complement system as array of BDDs [16]. Consequently, the characteristic function of the transition relation of an EFA E, $\chi_{\mapsto_E}$ will be

$$\chi_{\mapsto_E} = \bigvee_{\ell \xrightarrow{\sigma}_{g/a} \acute{\ell} \in \to_E} \chi_{\mapsto_{\ell \xrightarrow{\sigma}_{g/a} \acute{\ell}}}$$

III. Efficient Symbolic Reachability Computation

Not surprisingly, reachability (co-reachability) computations turn out to be the bottle-neck of the SCT synthesis algorithm. Adopting the symbolic representation using binary decision diagrams, we can partially solve this problem. However, with more complicated DESs, the BDD representation of the monolithic transition relation, $\chi_{\mapsto_{S_0}}$, might be extremely large to be constructed. More importantly, even though such BDD representing the monolithic transition relation is managed to be constructed, the reachability computation may still suffer from the state-space explosion due to the large intermediate BDDs. In this section, we present a way to partition DESs modeled by EFAs by using the disjunctive partitioning technique and then a straightforward but nontrivial algorithm, based on the partitioned BDDs, is presented to guide the state-space exploration.

Since S_0 is the synchronization of a number of sub-plants and sub-specifications in the form of EFAs, in all of the following computations we focus on $N \ge 2$ EFAs and let $\mathbf{E} = E_1 \parallel \ldots \parallel E_N$.

A. Partitioning of the full synchronous composition

Partitioning of the transition relation as introduced in [11] has become the standard guideline to alleviate the state-space problem. This is done by splitting the transition relation into a set of *partial transition relations*, connected by either disjunction or conjunction. In this paper, in correspondence with each event $\sigma \in \Sigma^{\mathbf{E}}$, the partial transition relation $\chi_{\overset{\sigma}{\mapsto}_{\mathbf{E}}}$ under full synchronous composition can be constructed in the following steps:

1) Compute $\chi_{\overset{\sigma}{\mapsto}_{\mathbf{E}^\dagger}}$ where $\mathbf{E}^\dagger = E_1^\dagger \parallel \ldots \parallel E_m^\dagger$, $\{E_1^\dagger, \ldots, E_m^\dagger\} \subseteq \{E_1, \ldots, E_N\}$ and $\sigma \in E_1^\dagger \cap \ldots \cap E_m^\dagger$.
2) Compute $\chi_{\overset{\sigma}{\mapsto}_{\mathbf{E}^\ddagger}}$ where $\mathbf{E}^\ddagger = E_1^\ddagger \parallel \ldots \parallel E_{m'}^\ddagger$ and $\{E_1^\ddagger, \ldots, E_{m'}^\ddagger\} = \{E_1, \ldots, E_N\} \backslash \{E_1^\dagger, \ldots, E_m^\dagger\}$.

Regarding step 1, computing $\chi_{\overset{\sigma}{\mapsto}_{\mathbf{E}^\dagger}}$, two further steps need to be performed in advance:

- Compute $\chi'_{\overset{\sigma}{\mapsto}_{\mathbf{E}^\dagger}}$, which denotes the characteristic function of $\overset{\sigma}{\mapsto}_{\mathbf{E}^\dagger}$ excluding the action functions of EFA variables,
- Compute $\chi_{\overset{\sigma}{\mapsto}^v_{\mathbf{E}^\dagger}}$ denoting the update of EFA variables.

To compute $\chi'_{\overset{\sigma}{\mapsto}_{\mathbf{E}^\dagger}}$, we make use of the following two propositions.

Proposition 1. *For an EFA E and an event $\sigma \in \Sigma^E$, the characteristic function representing the explicit transition relation through σ of E, denoted by $\chi_{\overset{\sigma}{\mapsto}_E}$, is computed as follows:*

$$\chi_{\overset{\sigma}{\mapsto}_E} = \chi_{\mapsto_E} \wedge \chi_\sigma,$$

where χ_σ is the characteristic function of the event σ and $\chi_{\mapsto_E}$ is the explicit transition relation of the EFA E.

Proposition 2. *Let $E_1^\dagger, \ldots, E_m^\dagger$ be $m \geq 2$ EFAs and $\sigma \in \Sigma^{E_1^\dagger} \cap \ldots \cap \Sigma^{E_m^\dagger}$. Then*

$$\chi'_{\overset{\sigma}{\mapsto}_{\mathbf{E}^\dagger}} = \bigwedge_{k=1}^{m} (\exists\ (\acute{b}^{V^1}, \ldots, \acute{b}^{V^n}).\chi_{\overset{\sigma}{\mapsto}_{E_k^\dagger}}), \tag{5}$$

where $\mathbf{E}^\dagger = E_1^\dagger \parallel \ldots \parallel E_m^\dagger$.

Subsequently, we compute $\chi_{\overset{\sigma\ v}{\mapsto}_{\mathbf{E}^\dagger}}$, which represents the update of EFA variables after the occurrence of σ. In the following computations, we focus on the update of a single variable between two EFAs and extend it to all variables for all EFAs in the model.

Definition III.1 (Updated Transition Relation Through σ)**.** For an EFA E and a single variable v^i, the updated transition relation for v^i through σ, denoted by $\overset{\sigma}{\mapsto}_{v^i,E}$, can be defined as

$$\overset{\sigma}{\mapsto}_{v^i,E} = \{(\ell, v, \sigma, \acute{\ell}, \acute{v}) \mid \forall (\ell, v, \sigma, \acute{\ell}, \acute{v}) \in \overset{\sigma}{\mapsto}_E \wedge\ \acute{v}^i \neq v^i\}.$$

Recall that,from Definition II.3, the result of $a^i(v)$ can be divided into four if-then constructs, which we denote by C_j. Each C_j consists of an `if` part, denoted by I_j, and a `then` part, denoted by T_j:

- I_1: $a_1^i = a_2^i$; both actions update the variables to the same value.
- T_1: $a^i(v) = a_1^i$ or $a^i(v) = a_2^i$.
- I_2: $a_2^i = \xi$; the first action updates the variable but not the second action.
- T_2: $a^i(v) = a_1^i$.
- I_3: $a_1^i = \xi$; the second action updates the variable but not the first action.
- T_3: $a^i(v) = a_2^i$.
- I_4: $otherwise$; none of the actions updates the variable, or the actions update the variable to different values.
- T_4: $a^i(v) = v^i$.

Definition III.2 (Interaction Transition Relation Through σ)**.** For two EFAs E_1 and E_2, and a variable v^i, the interaction transition relation through the event σ, denoted by $C_j(\overset{\sigma}{\mapsto}_{v^i,E_1\parallel E_2})$, can be defined as

$$\begin{aligned} C_1(\overset{\sigma}{\mapsto}_{v^i,E_1\parallel E_2}) \triangleq \{((\ell^{E_1}, \ell^{E_2}), v, \sigma, (\acute{\ell}^{E_1}, \acute{\ell}^{E_2}), \acute{v}) \mid \\ (\ell^{E_1}, v, \sigma, \acute{\ell}^{E_1}, \acute{v}) \in \overset{\sigma}{\mapsto}_{v^i,E_1} \wedge \\ (\ell^{E_2}, v, \sigma, \acute{\ell}^{E_2}, \acute{v}) \in \overset{\sigma}{\mapsto}_{v^i,E_2}\}, \end{aligned}$$

$$\begin{aligned} C_2(\overset{\sigma}{\mapsto}_{v^i,E_1\parallel E_2}) \triangleq \{((\ell^{E_1}, \ell^{E_2}), v, \sigma, (\acute{\ell}^{E_1}, \acute{\ell}^{E_2}), \acute{v}) \mid \\ (\ell^{E_1}, v, \sigma, \acute{\ell}^{E_1}, \acute{\nu}) \in \overset{\sigma}{\mapsto}_{v^i,E_1} \wedge \\ (\ell^{E_2}, v, \sigma, \acute{\ell}^{E_2}, \acute{\nu}) \in \overset{\sigma}{\mapsto}_{E_2} \setminus \overset{\sigma}{\mapsto}_{v^i,E_2}\}, \end{aligned}$$

$$\begin{aligned} C_3(\overset{\sigma}{\mapsto}_{v^i,E_1\parallel E_2}) \triangleq \{((\ell^{E_1}, \ell^{E_2}), v, \sigma, (\acute{\ell}^{E_1}, \acute{\ell}^{E_2}), \acute{v}) \mid \\ (\ell^{E_1}, v, \sigma, \acute{\ell}^{E_1}, \acute{\nu}) \in \overset{\sigma}{\mapsto}_{E_1} \setminus \overset{\sigma}{\mapsto}_{v^i,E_1} \wedge \\ (\ell^{E_2}, v, \sigma, \acute{\ell}^{E_2}, \acute{v}) \in \overset{\sigma}{\mapsto}_{v^i,E_2}\}, \end{aligned}$$

$$\begin{aligned} C_4(\overset{\sigma}{\mapsto}_{v^i,E_1\parallel E_2}) \triangleq \{((\ell^{E_1}, \ell^{E_2}), v, \sigma, (\acute{\ell}^{E_1}, \acute{\ell}^{E_2}), \acute{v}) \mid \\ ((\ell^{E_1}, \ell^{E_2}), v, \sigma, (\acute{\ell}^{E_1}, \acute{\ell}^{E_2}), \acute{v}) \notin \\ \bigcup_{j=1}^{3} (C_j \overset{\sigma}{\mapsto}_{v^i,E_1\parallel E_2}), \end{aligned}$$

where $\acute{\nu} = (\acute{v}^1, \ldots, \acute{v}^{i-1}, \xi, \acute{v}^{i+1}, \ldots, \acute{v}^n)$.

Hence, by definition we have:

$$\chi_{\overset{\sigma}{\mapsto}_{v^i,E_1\parallel E_2}} = \bigvee_{j=1}^{4} \chi_{C_j(\overset{\sigma}{\mapsto}_{v^i,E_1\parallel E_2})}.$$

Based on Definition III.2:

$$\chi_{\overset{\sigma\ v}{\mapsto}_{\mathbf{E}^\dagger}} = \bigwedge_{i=1}^{n} \chi_{\overset{\sigma}{\mapsto}_{v^i,\mathbf{E}^\dagger}}. \tag{6}$$

Moreover, $\chi_{\overset{\sigma}{\mapsto}_{\mathbf{E}^\dagger}}$ can be computed according to (5) and (6):

$$\chi_{\overset{\sigma}{\mapsto}_{\mathbf{E}^\dagger}} = \chi'_{\overset{\sigma}{\mapsto}_{\mathbf{E}^\dagger}} \wedge \chi_{\overset{\sigma\ v}{\mapsto}_{\mathbf{E}^\dagger}}. \tag{7}$$

At this stage, we are done with step 1.

Remark. Recall from Definition II.3, that if there exists an event σ, such that $\sigma \in \Sigma^{E_1} \setminus \Sigma^{E_2}$, on the occurrence of σ, E_2 would remain the previous location, i.e. $\forall \ell, \acute{\ell} \in L^{E_2}, \ell = \acute{\ell}$. On the other hand, the values of variables are updated according to the transitions labeled by σ in E_1.

Definition III.3 (Remained Transition Relation of σ)**.** For an EFA E and an event $\sigma \notin \Sigma^E$, the remained transition relation of σ for E, denoted by $\overset{\sigma}{\curvearrowright}_E$ can be defined by

$$\overset{\sigma}{\curvearrowright}_E = \{(\ell, \sigma, \acute{\ell}) \mid \forall \ell, \acute{\ell} \in L^E \wedge \ell = \acute{\ell}\}.$$

Therefore, the characteristic function representing $\overset{\sigma}{\mapsto}_{\mathbf{E}^\ddagger}$ can be computed according to the following proposition.

Proposition 3. *Let $E_1^\ddagger, \ldots, E_n^\ddagger$ be $n \geq 2$ EFAs and $\sigma \notin \Sigma^{E_1^\ddagger} \cup \ldots \cup \Sigma^{E_n^\ddagger}$, then*

$$\chi_{\overset{\sigma}{\mapsto}_{E_1^\ddagger \parallel \ldots \parallel E_n^\ddagger}} = \bigwedge_{k=1}^{n} \chi_{\overset{\sigma}{\curvearrowright}_{E_k^\ddagger}}. \tag{8}$$

Based on (7) and (8):

$$\chi_{\overset{\sigma}{\mapsto}_{\mathbf{E}}} = \chi_{\overset{\sigma}{\mapsto}_{\mathbf{E}^\dagger}} \wedge \chi_{\overset{\sigma}{\mapsto}_{\mathbf{E}^\ddagger}} \tag{9}$$

Theorem III.1. *For $N \geq 2$ EFAs $E_1, \ldots, E_N$ and $\mathbf{E} = E_1 \parallel \ldots \parallel E_N$ and an n-tuple of variables $v^i, \ldots, v^n$, the following statement holds:*

$$\chi_{\mapsto_{\mathbf{E}}} = \bigvee_{\sigma \in \bigcup_{k=1}^{N} \Sigma^{E_k}} \chi_{\overset{\sigma}{\mapsto}_{\mathbf{E}}} \tag{10}$$

For the proof, refer to [17].

B. Structural state-space exploration

Following the previous section, we conclude that in order to design successful BDD-based reachability algorithms for large-scaled systems, it is vital to traverse the state-space in a structural way. For this purpose, we present an alternative algorithm which is structurally similar to the workset algorithm in [13] with the difference that it works for the systems modeled as extended finite automata.

After the transition relation of a system has been partitioned into a set of event-based BDDs, the reachability computation, as is shown in Algorithm 1 starts to execute. Taking as input the initial state and the set of partial transition relations, the algorithm maintains a set of active partial transition relations, W_k. For each iteration, one partial transition relation is selected and a saturated reachability search (Algorithm 2) is performed on it. If more reachable states are found, based on Definition III.4 and III.5, more partial transition relations are appended to the workset. The algorithm terminates as long as there is no transition relation in W_k. The formal proof of the correctness of the algorithm can be found in [17].

Definition III.4 (Event Dependent Transition Relation Set of σ)**.** For $N \geq 2$ EFAs, $E_1, \ldots, E_N$, the event dependent transition relation sets of σ, denoted by $D^e(\overset{\sigma}{\mapsto}_{E_1\|\ldots\|E_N})$ is defined as:

$$D^e(\overset{\sigma}{\mapsto}_{E_1\|\ldots\|E_N}) = \{\overset{\sigma'}{\mapsto}_{E_1\|\ldots\|E_N} |\ \sigma' \in D^e(\sigma) \wedge \sigma' \neq \sigma\},$$

where

$$D^e(\sigma) = \{\sigma' \mid \exists E \in \{E_1, \ldots, E_N\}, \text{ such that } \sigma' \text{ is the successor of } \sigma \text{ in } E\}.$$

Definition III.5 (Variable Dependent Transition Relation Set of σ)**.** For $N \geq 2$ EFAs and a n-tuple of variables $v^1, \ldots, v^n$, the variable dependent transition relation sets of σ, denoted by $D^v(\overset{\sigma}{\mapsto}_{E_1\|\ldots\|E_N})$, is defined as:

$$D^v(\overset{\sigma}{\mapsto}_{E_1\|\ldots\|E_N}) = \{\overset{\sigma'}{\mapsto}_{E_1\|\ldots\|E_N} |\ \sigma' \in D^v(\sigma) \wedge \sigma' \neq \sigma\},$$

where

$$D^v(\sigma) = \{\sigma' \mid \exists(\ell, \sigma', g, a, \acute{\ell}) \in \rightarrow_{E_1\|\ldots\|E_N}, \forall \chi_{v^i} \in g \text{ such that } \exists(\ell, v, \sigma, \acute{\ell}, \acute{v}) \in \overset{\sigma}{\mapsto}_{E_1\|\ldots\|E_N} \wedge v^i \neq \acute{v}^i)\}.$$

IV. Case Studies

The proposed partitioning and traversal algorithm in this paper has been implemented in the supervisory control tool *Supremica* [18] which uses *JavaBDD* [19] as the BDD package. In this section, it is applied to a set of academic and industrial benchmark examples to demonstrate the efficiency.

The benchmark examples where experiments are carried out are: Resource Allocation System (RAS) [20], [21], Resource Allocation Systems with Error Handling (RAS-EH) [17], Ball Sorting Process (BSP) [22], Automated Guided Vehicles (AGV) [23], Parallel Manufacturing Example (PME) [24], Cat and Mouse Tower (CMT) and Extended Dinning Philosophers (EDP) [25], [17].

Algorithm 1 Event-based Forward Reachability

1: **input** $q_0 := (\ell_0^{E_1} \times \ldots \times \ell_0^{E_N} \times v_0)$,
   $W_0 := \{\overset{\sigma}{\mapsto}_{E_1\|\ldots\|E_N} |\ \forall\sigma \in \Sigma^{E_1} \cup \ldots \cup \Sigma^{E_N}\}$
2: **let** $Q_0 := \{q_0\}, k := 0$
3: **repeat**
4: Pick and remove $\overset{\sigma}{\mapsto}_{E_1\|\ldots\|E_N} \in W_k$
5: $k := k + 1$
6: $Q_k := Q_{k-1} \cup \texttt{Reachability}(Q_{k-1}, \overset{\sigma}{\mapsto}_{E_1\|\ldots\|E_N})$
7: **if** $Q_k \neq Q_{k-1}$ **then**
8: $W_k := W_{k-1} \cup D^e(\overset{\sigma}{\mapsto}_{E_1\|\ldots\|E_N}) \cup D^v(\overset{\sigma}{\mapsto}_{E_1\|\ldots\|E_N})$
9: **end if**
10: **until** $W_k = \emptyset$
11: **return** Q_k

Algorithm 2 Reachability

1: **input** $Q, \overset{\sigma}{\mapsto}_{E_1\|\ldots\|E_N}$
2: **let** $Q_0 := Q, k := 0$
3: **repeat**
4: $k := k + 1$
5: $Q_k := Q_{k-1} \cup \{(\acute{q}, \acute{v}) \mid (\acute{q}, \acute{v}) \in Q_k$ such that $\exists(q, v) \in Q_{k-1} \wedge (q, v, \sigma, \acute{q}, \acute{v}) \in \overset{\sigma}{\mapsto}_{E_1\|\ldots\|E_N}\}$
6: **until** $Q_k = Q_{k-1}$
7: **return** Q

Experiments are carried out on a standard PC (Intel Core 2 Quad CPU @ 2.4 GHz and 3GB RAM) running Windows 7 and the result is shown in Table I. For each benchmark example, the minimally restrictive supervisor generated by the algorithm is both non-blocking and controllable. It can be observed that both the monolithic and partitioning approaches can handle AGV, for which the number of reachable states is up to 10^7. However, by comparing the maximal number of BDD nodes during the reachability computation, which can express the maximal memory usage, the monolithic approach needs 9 times more memory than the partitioning approach. Regarding the example BSP, event though the final number of supervisor states is only 706, the intermediate BDDs during the state-space exploration, on the other hand, are large due to the high interactive complexity of the system. The monolithic approach fails to explore the state-space while the partitioning approach can survive and synthesize the supervisor within 11 seconds. As mentioned before, since the proposed partitioning algorithm is based on the alphabet which might contain a large number of events, more iterations than the standard algorithm are needed to reach the final fixed point. However, the intermediate BDDs produced during the computation are smaller, leading to improved memory and runtime efficiency. Finally, with respect to the last two benchmark examples, Cat and Mouse Tower and Extended Dining Philosophers, the partitioning approach can also handle some relatively large problem instances with the acceptable time. However, with the values of parameters growing, both the computation time and memory used increase rapidly.

TABLE I: Comparison Between Two Symbolic Synthesis Approaches

Model	Reachable States	Supervisor states	BDD Monolithic Approach		BDD Partitioning Approach	
			BDD Peak	Computation Time (s)	BDD Peak	Computation Time (s)
RAS	1.19×10^4	0.88×10^4	2826	0.49	215	0.13
RAS-EH	1.84×10^6	0.68×10^6	42314	18.67	2275	0.87
BSP	706	706	M.O.	–	16640	10.48
AGV	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	16770	24	5070	4.1
CMT (5,5)	1.07×10^{10}	3.15×10^9	M.O.	–	65102	79
EDP (5,10)	167761	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	–	T.O.	192	1.3
EDP (5,200)	3.28×10^{11}	8.20×10^6	–	T.O.	206	6.5

M.O. indicates memory out during reachability search (due to large intermediate BDDs) and T.O. indicates time out (10 min).

V. Conclusions

In this paper, we presented an alternative symbolic approach to large-scaled systems modeled as extended finite automata. The proposed approach first partitions the closed-loop system under full synchronous composition based on the disjunctive partitioning technique, and then depends on an efficient algorithm to explore the state-space in a structural way.

The proposed approach has been implemented and integrated into prior work. Besides, it is applied to a set of academic and industrial examples to demonstrate the efficiency. Overall, the whole framework provides the convenience for users to model systems and obtain control functions in the same model domain. All computations are performed symbolically by BDDs, which are transparent and the only interface users deal with is the EFA framework.

References

[1] P. J. G. Ramadge and W. M. Wonham, "The Control of Discrete Event Systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[2] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. Springer, 2008.

[3] S. Miremadi, B. Lennartson, and K. Åkesson, "A BDD-based Approach for Modeling Plant and Supervisor by Extended Finite Automata," *accepted for IEEE Transactions on Control Systems Technology*, 2011.

[4] M. Sköldstam, K. Åkesson, and M. Fabian, "Modeling of Discrete Event Systems using Finite Automata with Variables," *Decision and Control, 2007 46th IEEE Conference on*, pp. 3387–3392, 2007.

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

[6] R. Bryant, "Graph-based Algorithms for Boolean Function Manipulation," *IEEE Transactions on Computers*, vol. 35, no. 8, pp. 677–691, 1986.

[7] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, "Symbolic Model Checking: 10^{20} States and Beyond," in *5th IEEE Symposium on Logic in Computer Science, 1990. LICS '90,*, Jun. 1990, pp. 428–439.

[8] H. Wong-Toi and G. Hoffmann, "The control of dense real-time discrete event systems," in *Proceedings of the 30th IEEE Conference on Decision and Control*. IEEE, 1991, pp. 1527–1528.

[9] 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, May 2006.

[10] R. Song and R. J. Leduc, "Symbolic synthesis and verification of hierarchical interface-based supervisory control," in *8th international Workshop Discrete Event Systems, WODES'06*, Ann Arbor, MI, USA, Jul. 2006, pp. 419–426.

[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] M. Byröd, B. Lennartson, A. Vahidi, and K. Åkesson, "Efficient Reachability Analysis on Modular Discrete-Event Systems using Binary Decision Diagrams," in *8th international Workshop on Discrete Event Systems, WODES'06*, Ann Arbor, MI, USA, Jul. 2006, pp. 288–293.

[13] A. Vahidi, M. Fabian, and B. Lennartson, "Efficient Supervisory Synthesis of Large Systems," *Control Engineering Practice*, vol. 14, no. 10, pp. 1157–1167, Oct. 2006.

[14] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423,625–656, 1948.

[15] R. E. Bryant, "Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams," *ACM Computing Surveys*, vol. 24, no. 3, pp. 293–318, 1992.

[16] E. M. Clarke, K. L. Mcmillan, X. Zhao, M. Fujita, and J. Yang, "Spectral Transforms for Large Boolean Functions with Applications to Technology Mapping," *Form. Methods Syst. Des.*, vol. 10, no. 2-3, pp. 137–148, 1997.

[17] Z. Fei, S. Miremadi, K. Åkesson, and B. Lennartson, "Efficient Supervisory Synthesis to Large-Scale Discrete Event Systems Modeled as Extended Finite Automata," Automation Research Group, Department of Signals and Systems, Chalmers University of Technology, Tech. Rep., 2012.

[18] K. Åkesson, M. Fabian, H. Flordal, and R. Malik, "Supremica—An Integrated Environment for Verification, Synthesis and Simulation of Discrete Event Systems," in *Proceedings of the 8th international Workshop on Discrete Event Systems, WODES'08*, Ann Arbor, MI, USA, 2006, pp. 384–385.

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

[20] J. Ezpeleta, J. Colom, and J. Martinez, "A Petri Net Based Deadlock Prevention Policy for Flexible Manufacturing Systems," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 2, pp. 173 –184, 1995.

[21] Z. Fei, S. Miremadi, and K. Åkesson, "Modeling Sequential Resource Allocation Systems using Extended Finite Automata," in *7th IEEE International Conference on Automation Science and Engineering*, Trieste, 2011, pp. 444–449.

[22] G. Cengi, "A Control Software Development Method Using IEC 61499 Function Blocks , Simulation and Formal Verification," in *the 17th IFAC World Congress*, 2008.

[23] 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.

[24] R. J. Leduc, "Hierarchical interface-based supervisory control," Ph.D. dissertation, Department of Electrical and Computer Engineering, University of Toronto, 2002.

[25] S. Miremadi, K. Åkesson, M. Fabian, A. Vahidi, and B. Lennartson, "Solving Two Supervisory Control Benchmark Problems using Supremica," in *9th International Workshop on Discrete Event Systems, 2008, WODES 08.*, May 2008, pp. 131–136.