Symbolic Computation of Nonblocking Control Function for Timed Discrete Event Systems

Symbolic Computation of Nonblocking Control Function for Timed Discrete Event Systems

S. Miremadi, Z. Fei, K. Åkesson and B. Lennartson
Automation Research Group, Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
{miremads, zhennan, knut, bengt.lennartson}@chalmers.se

***Abstract*—In this paper, we symbolically compute a minimally restrictive nonblocking supervisor for timed discrete event systems, in the supervisory control theory context. The method is based on Timed Extended Finite Automata, which is an augmentation of extended finite automata (EFAs) by incorporating discrete time into the model. EFAs are ordinary automaton extended with discrete variables, guard expressions and action functions. To tackle large problems all computations are based on binary decision diagrams (BDDs). The main feature of this approach is that the BDD-based fixed-point computations is not based on "tick" models that have been commonly used in this area, leading to better performance in many cases. As a case study, we effectively computed the minimally restrictive nonblocking supervisor for a well-known production cell.**

I. Introduction

Discrete Event Systems (DES) are discrete-state, event-driven systems where their state evolution depends entirely on the occurrence of asynchronous events over time. DES have many applications in modeling technological systems such as automated manufacturing and embedded systems. When designing control functions for DES, model-based approaches may be used to conveniently understand the system's behavior. A well known framework of such a model-based approach is supervisory control theory (SCT) [1]. Having a plant (the system to be controlled) and a specification, SCT automatically synthesizes a control function, called *supervisor*, that restricts the conduct of the plant to ensure that the system never violates the given specification. Most of the research in this field has focused on analyzing qualitative properties, such as safety or liveness specifications, by investigating the logical sequencing of events. However, the correct behavior of many real-time systems such as air traffic control systems and networked multimedia systems depends on the delays between events. In addition, on pure DES one cannot perform quantitative analysis such as time optimization or scheduling. Timed DES (TDES) is a generalization of DES in which the times that the events occur are also taken into consideration. In this work we do not consider stochastic properties of the models. The modeling formalism used in this work is an augmentation of a previously proposed modeling formalism, called extended finite automaton (EFA) [2], where time has been incorporated into the model. EFAs are ordinary automaton extended with discrete variables, guard expressions and action functions. The guards and action functions are attached to the transitions, which admits local design techniques of systems consisting of different parts. The main features of EFAs are that they are suitable for the SCT framework and that they usually yield compact models because of the existence of discrete variables. EFAs have been used in several research works and successfully applied to a range of examples such as [3], [4], [5]. The EFA framework has been implemented in Supremica [6], a verification and supervisory control tool, where powerful algorithms exist for analysis of DES [7], [8], [9].

There have been many attempts to model TDES and generalize SCT considering the real-time aspects. These works can be divided into two categories; they are either based on continuous time or discrete time. On the continuous side, timed automata [10] is the most popular modeling formalism used for modeling TDES and employing them in SCT [11], [12]. With respect to control function generation, there exist another approach that differs from the ones using the SCT theory [13], where the controller is defined as a winning strategy for a certain game defined for the timed automata, called timed game automata (TGA). There exist different works based on this paradigm such as [14].

There exists a lot of work that have analyzed discrete time models with respect to SCT such as [15], [16], [17], [18]. In these works, it is assumed that there exists a global digital clock. In [17], the timing information is incorporated into the system states in the form of timer variables, which are updated according to some rules relating event occurrences and the passage of time. The more common way to model TDES, described in [15], [18], is that lower and upper time bounds are associated with events to restrict their occurrence times. In addition, they use a special event "tick", which represents the passage of time, and is generated by the global clock. In [19], Brandin and Wonham applied SCT to Timed Transition Models (TTMs) proposed in [15]. The main problem with their approach is that by introducing the "tick" event more iterations maybe needed in the fixed point computations. In addition, it is more likely to get early state space explosion. In [20] some methods have been proposed to reduce the state space.

Consequently, there are many models and implementations that are suitable for quantitative analysis (such as time optimization), most of them based on continuous time; and there are many that are suitable for the SCT framework (qualitative analysis), most of them based on discrete time. Yet no work exists considering both aspects. In this paper, we attempt to combine the best of both paradigms by using *Timed EFAs* (TEFAs), EFAs equipped with a finite set of discrete clocks, where the value of each clock is increased implicitly as time progresses. Based on TEFAs, inspired by the "zone" concept from the timed automata community, we symbolically compute a minimally restrictive nonblocking supervisor by using BDDs. The main feature of our approach, in the context of SCT, compared to most of the other approaches with discrete time, is the elimination of the "tick" event in the BDD-based fixed-point computations. In most of the cases, this leads to less number of fixed-point iterations and more compact BDD representation yielding a more efficient implementation. Furthermore, from a modeling perspective, the advantage of using TEFAs compared to TTMs is that the time constraints are added as guards on the transitions (as in timed automata [10]), rather than lower and upper bounds on the events. This could potentially facilitate the modeling for the users. For instance, if the constraints are associated to the events, it will be complicated to model the situation, where the user wants to put different time constraints on an event that appears on different places on the same model. Usually the consequence is a larger state space. The mentioned advantages are demonstrated in Section V.

II. Timed Extended Finite Automata

A *Timed Extended Finite Automaton (TEFA)* is an EFA augmented with a finite set of digital clocks. Intuitively, a *clock* in a TEFA is a discrete variable in the sense of EFAs, restricted by some rules, mentioned later. The time automatically elapses only at locations, whereas the transitions occur instantaneously with zero delay.

A. Syntax and Semantics

In the following, we describe the syntax and semantics of TEFAs.

Definition 1 (Timed Extended Finite Automaton)**.**
A timed extended finite automaton is a 9-tuple

$$\mathsf{TE} = (L, D^{\mathcal{V}}, \mathcal{C}, \Sigma, \rightarrow, L^0, D^{\mathcal{V}_0}, L^m, D^m),$$

where

- L is a finite set of locations,
- $D^{\mathcal{V}} = D^{\mathcal{V}}_1 \times \ldots \times D^{\mathcal{V}}_n$ is the domain of n variables $\mathcal{V} = \{v_1, \ldots, v_n\}$, where $D^{\mathcal{V}}_i$ is a finite set of integers,
- $\mathcal{C}$ is a finite set of p discrete valued clocks $\{c_1, \ldots, c_p\}$,
- Σ is a nonempty finite set of events,
- $\rightarrow \subseteq L \times \Sigma \times \mathcal{G} \times \mathcal{A} \times L$ is the transition relation,
- $L^0 \subseteq L$ is the set of initial locations,
- $D^{\mathcal{V}_0} = D^{\mathcal{V}_0}_1 \times \ldots \times D^{\mathcal{V}_0}_n$ is the set of initial values of the variables,
- $L^m \subseteq L$ is the set of marked locations that are desired to be reached, and
- $D^m = D^{\mathcal{V}_m} \times D^{\mathcal{C}_m}$ is the set of pairs of marked valuations of the variables and clocks.

In addition to $D^{\mathcal{V}}$, we also define $D^{\mathcal{C}}$ representing the domain of p clocks. Later we will explain how the domain of a clock is defined and show that it is finite. The *global variable domain* denoted by $D^{\mathcal{V}}_{\cup}$ is the set that contains the values of all variables, defined formally as:

$$D^{\mathcal{V}}_{\cup} = \bigcup_{i=1}^{|\mathcal{V}|} D^{\mathcal{V}}_i.$$

The *global clock domain* denoted by $D^{\mathcal{C}}_{\cup}$ is defined similarly. The largest value in $D^{\mathcal{V}}_{\cup}$ and $D^{\mathcal{C}}_{\cup}$ is denoted by $\mu\text{max}^{\mathcal{V}}$ and $\mu\text{max}^{\mathcal{C}}$, respectively. If a variable exceeds its domain, the result is not defined, and from an implementation point of view, it is upon the developer to decide how to implement such cases. For instance, the program can give the user a warning. In our implementation, values outside the domain are not reachable. In contrast to variables, it is assumed that if a clock c_i reaches its maximum value, it will keep its value until it is reset. For a clock c_i, this behavior is modeled by a saturation function $\varrho_i : \mathbb{N} \rightarrow D^{\mathcal{C}}_i$:

$$\varrho_i(x) = \begin{cases} x & \text{if } x < \mu\text{max}^{\mathcal{C}}_i \\ \mu\text{max}^{\mathcal{C}}_i & \text{if } x \geq \mu\text{max}^{\mathcal{C}}_i \end{cases},$$

where $\mathbb{N}$ is the set of natural numbers. The function $\varrho : \mathbb{N}^p \rightarrow D^{\mathcal{C}}$ is used to saturate the current value of all clocks.

The elements $\mathcal{G}$ and $\mathcal{A}$ are the sets of guards (conditional expressions) and action functions, respectively. In the TEFA framework, an arithmetic expression φ is formed according to the grammar

$$\varphi ::= \nu \mid v \mid c \mid (\varphi) \mid \varphi + \varphi \mid \varphi - \varphi \mid \varphi * \varphi \mid \varphi / \varphi \mid \varphi \% \varphi,$$

where $v \in \mathcal{V}$, $c \in \mathcal{C}$, $\nu \in D^{\mathcal{V}}_{\cup} \cup D^{\mathcal{C}}_{\cup}$, and $\%$ is the modulo operator. We use $\varphi^{\mathcal{V}}$ to denote an expression that does not contain any clocks and $\nu \in D^{\mathcal{V}}_{\cup}$. A *variable evaluation* for a variable $v_i \in \mathcal{V}$ is a function $\mu^{\mathcal{V}}_i : v_i \rightarrow D^{\mathcal{V}}_i$, assigning a value to the variable. A *clock evaluation* $\mu^{\mathcal{C}}_i : c_i \rightarrow D^{\mathcal{C}}_i$ is defined similarly. The set of evaluations for all variables and clocks is represented by $\mu^{\mathcal{V}}$ and $\mu^{\mathcal{C}}$, respectively. To denote the "current" evaluation of a variable or clock we use the notation η.

A guard $g \in \mathcal{G}$ is a propositional expression formed according to the grammar

$$g \ ::= (g) \mid g^{\mathcal{V}} \wedge g^{\mathcal{C}} \mid g^{\mathcal{V}} \vee g^{\mathcal{C}},$$

where $g^{\mathcal{V}}$ and $g^{\mathcal{C}}$ are guards that are based on regular variables and clocks, respectively,

$$\begin{aligned} g^{\mathcal{V}} \ &::= \varphi^{\mathcal{V}} < \varphi^{\mathcal{V}} \mid \varphi^{\mathcal{V}} \leq \varphi^{\mathcal{V}} \mid \varphi^{\mathcal{V}} > \varphi^{\mathcal{V}} \mid \varphi^{\mathcal{V}} \geq \varphi^{\mathcal{V}} \mid \\ &\quad \varphi^{\mathcal{V}} == \varphi^{\mathcal{V}} \mid (g^{\mathcal{V}}) \mid g^{\mathcal{V}} \wedge g^{\mathcal{V}} \mid g^{\mathcal{V}} \vee g^{\mathcal{V}} \mid \top \mid \bot, \\ g^{\mathcal{C}} \ &::= c < \omega \mid c \leq \omega \mid c > \omega \mid c \geq \omega \mid c == \omega \mid \\ &\quad (g^{\mathcal{C}}) \mid g^{\mathcal{C}} \wedge g^{\mathcal{C}} \mid \top \mid \bot, \end{aligned}$$

where $\top$ and $\bot$ represent Boolean logic `true` and `false`, respectively, and $\omega \in D^{\mathcal{C}}_{\cup}$. This implies that clocks can only be compared to constants. All nonzero values are considered as $\top$. The semantics of a guard g is specified by a *satisfaction relation* $\models$ indicating the pair of variable and clock evaluations $(\mu^{\mathcal{V}}, \mu^{\mathcal{C}})$ for which guard g is $\top$. It is written $(\mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \models g$.

An action $\mathbf{a} \in \mathcal{A}$ is an n-tuple of functions $(a_1, \ldots, a_{n+p})$, updating variables. An action $a_i : D^{\mathcal{V}} \times D^{\mathcal{C}} \rightarrow D^{\mathcal{V}}_i$ is a function that updates a variable or clock; in the case of a clock the range of the function is zero. Hence, for a variable, the action is formed as $v_i := \varphi$ and for a clock it is formed as $c_i := 0$. For brevity, we use the following notation:

$$\mathbf{a}(\mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \triangleq (a_1(\mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \ldots, a_{n+p}(\mu^{\mathcal{V}}, \mu^{\mathcal{C}})).$$

An action function a_i that does not update a variable or clock is denoted by ξ. The semantics of an action function can also be represented by a relation,

$$\mathsf{SAT}\mathcal{A}(\mathbf{a}) \triangleq \{((\mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \acute{\mu}^{\mathcal{V},\mathcal{C}}) \mid \acute{\mu}^{\mathcal{V},\mathcal{C}} = \mathbf{a}(\mu^{\mathcal{V}}, \mu^{\mathcal{C}})\},$$

where $\acute{\mu}^{\mathcal{V},\mathcal{C}}$ is the updated value for a variable or clock. Consequently, in contrast to variables, clocks may only be inspected, and reset to zero.

A partial transition relation is written as $l \xrightarrow{\sigma}_{g/\mathbf{a}} \acute{l}$, where $l, \acute{l} \in L$, $\sigma \in \Sigma$, $g \in \mathcal{G}$, and $\mathbf{a} \in \mathcal{A}$. A transition without guard indicates that there are no restrictions, i.e., $g = \top$.

We assume that all clocks evolve synchronically at rate one. The value of a clock denotes the amount of time that has been elapsed since its last reset. Potentially, the clocks in $\mathcal{C}$ can have an infinite domain because the time will elapse forever. Nevertheless, based on the following argument a finite domain can be considered for each clock. Among the possible values of a clock, only a subset is relevant: those that can impact the guards' evaluations. For instance, for a guard $c_1 \leq 4$, the values above 4 will all have the same impact on the guard; thus the relevant values of c_1 is $\{0, \ldots, 5\}$. Considering $\mu\text{largest}_i^{\mathcal{C}}$ to be the largest constant in the model (including all guards), which the clock c_i is compared to, the domain of the clock c_i is $D_i^{\mathcal{C}} = \{0, 1, \ldots, \mu\text{largest}_i^{\mathcal{C}} + 1\}$. Thus, $\mu\text{max}_i^{\mathcal{C}} = \mu\text{largest}_i^{\mathcal{C}} + 1$. Consequently, the domain of the clocks $D^{\mathcal{C}} = D_1^{\mathcal{C}} \times \ldots \times D_p^{\mathcal{C}}$ will be finite.

For a variable v_i, $D_i^{\mathcal{V}_0}$ consists of the initial values of v_i. If the set of marked locations, valuations of a variable v_i, or a clock c_i is empty, then the entire domain is considered as marked:

$$\begin{aligned} L^m = \emptyset \ &\Rightarrow \ L^m = L, \\ D_i^{\mathcal{V}_m} = \emptyset \ &\Rightarrow \ D_i^{\mathcal{V}_m} = D_i^{\mathcal{V}}, \\ D_i^{\mathcal{C}_m} = \emptyset \ &\Rightarrow \ D_i^{\mathcal{C}_m} = D_i^{\mathcal{C}}. \end{aligned}$$

A transition will be executed if an event occurs, and if the guard on the corresponding transition (the transition that involves that event) is satisfied, which may follow by a number of updates on the variables and clocks. It is assumed that the time will elapse at locations and that the transitions are executed instantaneously. Furthermore, we assume that the initial values of all clocks are zero and that all TEFAs are deterministic in the sense of deterministic EFAs defined in [2].

B. Full Synchronous Composition

For modeling purposes, it is often easier to have a modular representation, specially for complex systems. Then, to have a monolithic model of the system we need to synchronize the components. For a model with a number of TEFAs, we assume that the variables $\mathcal{V}$ and clocks $\mathcal{C}$ are all *global*, i.e., they are shared between the TEFAs. Hence, the clocks evolve synchronically with the same rate. The *full synchronous composition* on TEFAs, can be defined similar to [2].

A notation that will be used frequently in this paper, is the *SOS-notation* (Structured Operational Semantics) used to define the transition relations. The notation

$$\frac{\text{premise}}{\text{conclusion}}$$

should be read as follows. If the proposition above the "solid line" (premise) holds, then the proposition under the fraction bar (conclusion) holds as well.

Definition 2 (Full Synchronous Composition). Consider the following two TEFAs

$$\text{TE}_k = (L_k, D^{\mathcal{V}}, \mathcal{C}, \Sigma_k, \rightarrow_k, L_k^0, D^{\mathcal{V}_0}, L_k^m, D^m),$$

where $k = 1, 2$. The Full Synchronous Composition (FSC) of TE_1 and TE_2, denoted by $\text{TE}_1 \| \text{TE}_2$, is defined as

$$\text{TE}_1 \| \text{TE}_2 = (L, D^{\mathcal{V}}, \mathcal{C}, \Sigma, \rightarrow, L^0, D^{\mathcal{V}_0}, L^m, D^m),$$

where

- $L = L_1 \times L_2$,
- $\Sigma = \Sigma_1 \cup \Sigma_2$,
- the transition relation

$$\rightarrow \subseteq L_1 \times L_2 \times \Sigma \times \mathcal{G} \times \mathcal{A} \times L_1 \times L_2$$

  is defined based on the following rules:

  a) $\sigma \in \Sigma_1 \cap \Sigma_2$,

$$\frac{\begin{array}{c}(l_1, \sigma, g_1, \mathbf{a_1}, \acute{l_1}) \in \rightarrow_1 \ \wedge \\ (l_2, \sigma, g_2, \mathbf{a_2}, \acute{l_2}) \in \rightarrow_2\end{array}}{((l_1, l_2), \sigma, g, \hat{\mathbf{a}}, (\acute{l_1}, \acute{l_2})) \in \rightarrow}$$

  such that,

  (i) $g = g_1 \wedge g_2$,

  (iii) For $i = 1, \ldots, |\mathcal{V}|$,

$$\hat{\mathbf{a}}_\mathbf{i} = \begin{cases} a_{1i} & \text{if } a_{1i} = a_{2i} \\ a_{1i} & \text{if } a_{2i} = \xi \\ a_{2i} & \text{if } a_{1i} = \xi \\ \eta_i^{\mathcal{V}} & \text{otherwise} \end{cases}, \tag{1}$$

  where a_{ki} is the action function belonging to $\rightarrow_k$, updating the i-th variable or clock;

  b) $\sigma \in \Sigma_1 \backslash \Sigma_2$,

$$\frac{(l_1, \sigma, g_1, \mathbf{a_1}, \acute{l_1}) \in \rightarrow_1}{((l_1, l_2), \sigma, g_1, \mathbf{a_1}, (\acute{l_1}, \acute{l_2})) \in \rightarrow \ \wedge \ l_2 = \acute{l_2}};$$

  c) $\sigma \in \Sigma_2 \backslash \Sigma_1$,

$$\frac{(l_2, \sigma, g_2, \mathbf{a_2}, \acute{l_2}) \in \rightarrow_2}{((l_1, l_2), \sigma, g_2, \mathbf{a_2}, (\acute{l_1}, \acute{l_2})) \in \rightarrow \ \wedge \ l_1 = \acute{l_1}}.$$

- $L^0 = L_1^0 \times L_2^0$, and
- $L^m = L_1^m \times L_2^m$.

Similar to the proof in [21], it can be proved that the FSC operator is both commutative and associative and can be extended to N TEFAs. Note that, in the case where the action functions of TE_1 and TE_2 explicitly try to update a shared variable to different values, we assume that the variable is not updated. It can indeed be discussed whether such a transition should be executed, nevertheless, such a situation is usually a consequence of bad modeling.

C. EFA Semantics

As mentioned earlier, the clocks in TEFAs are discrete-values, indicating that we imagine measuring time only with a global digital clock with output $tickcount : \mathbb{R}^+ \rightarrow \mathbb{N}$, where

$$tickcount(t) = n, \quad n \leq t < n + 1,$$

and $\mathbb{R}^+ = \{t \in \mathbb{R} | t \geq 0\}$ is the set of positive real values. Consequently, the temporal resolution available for modeling purposes is thus just one unit of clock time. For a TEFA, this behavior, can be represented by an EFA by introducing an additional event "tick" as in [15]. The event "tick" occurs exactly at the real time moments, which can be imagined to be generated by the global digital clock. Consider a TEFA with a single clock c_1. Lets treat c_1 as a regular variable with domain $\{0, \ldots, \mu\text{max}_1^{\mathcal{C}}\}$. The described time semantics of the TEFA can be achieved by adding two transitions to each location ℓ:

$$(\ell, tick, c_1 < \mu\text{max}_1^{\mathcal{C}}, c_1 := c_1 + 1, \ell) \text{ and}$$
$$(\ell, tick, c_1 \geq \mu\text{max}_1^{\mathcal{C}}, c_1 := c_1, \ell). \tag{2}$$

For a clock c_1, we call an EFA that has a single location ℓ and consists of the transitions in (2) a *clock-EFA* and denote it by CE_1. If there exists multiple clocks, all combinations of the transitions in (2) for different clocks should be considered. This can be carried out by synchronizing the clock-EFAs based on the full synchronous composition on EFAs [2].

Definition 3 (isomorphic EFA of a TEFA)**.** Let TE $= (L, D^{\mathcal{V}}, \mathcal{C}, \Sigma, \rightarrow, L^0, D^{\mathcal{V}_0}, L^m, D^m)$ be an TEFA. Its corresponding isomorphic EFA, denoted by **iEFA**(TE), is an 8-tuple $(L, D^{\mathcal{V}^E}, \Sigma, \rightarrow, L^0, D^{\mathcal{V}_0^E}, L^m, D^m)$:

- $D^{\mathcal{V}^E} = D^{\mathcal{V}} \times D^{\mathcal{C}}$, where $\mathcal{V}^E = \mathcal{V} \cup \mathcal{C}$;
- $D^{\mathcal{V}_0^E} = D^{\mathcal{V}_0} \times \{\mathbf{0}^{|\mathcal{C}|}\}$, where $\mathbf{0}^{|\mathcal{C}|}$ is a $|\mathcal{C}|$-tuple of zeros.

Proposition 1. *For N TEFAs, the following statement holds:*

$$\textbf{\textit{iEFA}}(TE_1 \parallel \ldots \parallel TE_N) = \textbf{\textit{iEFA}}(TE_1) \parallel \ldots \parallel \textbf{\textit{iEFA}}(TE_N).$$

Proof: The proof follows directly from Definition 2 and the full synchronous composition of EFAs, defined in [2]. ■

Let TE $= (L, D^{\mathcal{V}}, \mathcal{C}, \Sigma, \rightarrow, L^0, D^{\mathcal{V}_0}, L^m, D^m)$ be an TEFA. Based on the full synchronous composition of EFAs, TE's corresponding EFA, denoted by **EFA**(TE) is computed as follows:

$$\textbf{EFA}(\text{TE}) = \textbf{iEFA}(\text{TE}) \parallel CE_1 \parallel \ldots \parallel CE_{|\mathcal{C}|}. \tag{3}$$

We call **EFA**(TE) the *tick-EFA* of TE.

Lemma 2. *In the corresponding EFA of a TEFA the "tick" event never becomes restricted due to synchronization between the clock-EFAs.*

Proof: Based on the definition of full synchronous composition on EFAs and the following facts, the statement can be directly deduced:

- *For a clock c_i, since $(c_i < \mu max_i^{\mathcal{C}}) \vee (c_i \geq \mu max_i^{\mathcal{C}}) = \top$, CE_i will always allow either of the transitions;*
- *The clock-EFAs do not share any variables and thus cannot restrict each other in synchronization.*

■

Theorem 3. *For N TEFAs and p clocks, the following statement holds:*

$$\textbf{\textit{EFA}}(TE_1 \parallel \ldots \parallel TE_N) = \textbf{\textit{EFA}}(TE_1) \parallel \ldots \parallel \textbf{\textit{EFA}}(TE_N). \tag{4}$$

Proof: Let $CE = CE_1 \parallel \ldots \parallel CE_p$. We construct the left-hand side by starting from the right-hand side and using (3) and propositions 1 and 2:

$$\begin{aligned}
&\textbf{\textit{EFA}}(TE_1) \parallel \ldots \parallel \textbf{\textit{EFA}}(TE_N) = \\
&\big(\textbf{\textit{iEFA}}(TE_1) \parallel CE\big) \parallel \ldots \parallel \big(\textbf{\textit{iEFA}}(TE_N) \parallel CE\big) = \\
&\textbf{\textit{iEFA}}(TE_1) \parallel \ldots \parallel \textbf{\textit{iEFA}}(TE_N) \parallel CE = \\
&\textbf{\textit{EFA}}(TE_1 \parallel \ldots \parallel TE_N).
\end{aligned}$$

■

The above theorem will be the basis for applying supervisory control theory to TEFAs. However, as we will see later in Section IV, the symbolic computations will be performed on an abstraction of the tick-EFAs by eliminating the "tick" event. This will be the main advantage of this approach compared to the "tick"-based approach in [22], [23].

III. Supervisory Control Theory

Supervisory Control Theory (SCT) [1], [24] is the first control theory for a general class of DES, where a control function is automatically synthesized, referred to as *supervisor*, based on a given *plant* and a *specification.* A specification describes the allowed and inhibited behaviors. The supervisor restricts the conduct of plant to guarantee that the system never violates the given specification. However, it is often desired, and also in our work, that the supervisor restricts the plant as least as possible, referred to as *optimal* or *minimally restrictive* supervisor. This gives the developers several alternatives to implement the controller and performing further analysis such as time or energy optimization.

A plant P can be described by the synchronization of a number of sub-plants $P = P_1 \parallel P_2 \parallel \ldots$ and similarly for a specification $Sp = Sp_1 \parallel Sp_2 \parallel \ldots$. There are different ways of computing a supervisor such as monolithic [1], modular [25], and compositional [26] synthesis. In our approach we apply monolithic synthesis, which is performing fixed-point computations on the single composed automaton $S_0 = P \| Sp$. After the synthesis procedure, some *blocking* states may be identified, which are the states from where no marked state can be reached. By removing the blocking states from S_0, a *nonblocking* supervisor is obtained. In SCT, the events can be divided into controllable and uncontrollable events (events that cannot be influenced by the supervisor) , causing controllability issues, which is not the focus in this paper.

In the context of SCT, the behavior of a system is usually represented by its language, i.e. the sets of strings that the system may generate. Conventionally, automata has been used as the modeling formalism to generate the language. In this work, the problems are modeled by TEFAs, while the SCT analysis is performed on their corresponding EFA models. In [3], [27] it is described how a nonblocking and minimally restrictive supervisor is symbolically computed. The fixed-point computations are performed on the corresponding state transition system of the EFAs. The state transition system of an EFA is based on its *explicit transition relation*, which can be thought as the transition relation of an ordinary automaton. The explicit transition relation $(\ell, \sigma, g, \mathbf{a}, \acute{\ell})$ belonging to an EFA is defined as $(\ell, \mu^{\mathcal{V}}, \sigma, \acute{\ell}, \acute{\mu}^{\mathcal{V}})$, where $\mu^{\mathcal{V}} \in \mathsf{SAT}\mathcal{G}(g)$ and $(\mu^{\mathcal{V}}, \acute{\mu}^{\mathcal{V}}) \in \mathsf{SAT}\mathcal{A}(\mathbf{a})$, represented as $(\ell, \mu^{\mathcal{V}}) \overset{\sigma}{\mapsto} (\acute{\ell}, \acute{\mu}^{\mathcal{V}})$. The symbolic computations is based on the explicit transition relation of S_0.

IV. Symbolic Representations and Computations

When performing fixed point computations for systems of industrially interesting sizes, exploring all states in the composed model *explicitly* can be computationally expensive, in terms of both time and memory, due to the state space explosion problem. We tackle this problem by representing the models and performing the computations *symbolically* using Binary Decision Diagrams (BDDs), powerful data structures for representing Boolean functions. For large systems where the number of states grows exponentially, BDDs can improve the efficiency of set and Boolean operations performed on the state sets [8], [28], [7].

Given a set of x Boolean variables $\mathcal{B}$, a Boolean function $f\colon \mathbb{B}^x \rightarrow \mathbb{B}$ ($\mathbb{B}$ is the set of Boolean values, i.e., 0 and 1) can be expressed using Shannon's decomposition. This decomposition can be expressed by a directed acyclic graph, called a BDD, 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*, corresponding to assigning 0 and 1 to the variable, respectively. The *size* of a BDD, denoted as $|\mathbf{B}|$, refers to the number of decision nodes.

The power of BDDs lies in their simplicity and efficiency to perform binary operations. The time complexity of a binary operator between two BDDs $\mathbf{B}_1$ and $\mathbf{B}_2$ is $O(|\mathbf{B}_1| \cdot |\mathbf{B}_2|)$.

Two BDD operations that have been used extensively in our implementation is the *existential quantification* and the *replacement* operators. Let $\mathbf{B}$ be a BDD and $\mathcal{B}_1$ and $\mathcal{B}_2$ two sets of Boolean variables. The operation $\exists\mathcal{B}_1 : \mathbf{B}$ removes all variables belonging to $\mathcal{B}_1$ that have appeared in $\mathbf{B}$. The operation $replace(\mathbf{B}, (\mathcal{B}_1, \mathcal{B}_2))$ will replace all occurrences of variables belonging to $\mathcal{B}_1$ by variables belonging to $\mathcal{B}_2$. For a more elaborate and verbose exposition of BDDs and the implementation of different operators, refer to [29].

The corresponding BDD for a finite set $W \subseteq U$ can be represented using its corresponding *characteristic function*.

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

$$\chi_W(a) = \left\{ \begin{array}{ll} 1 & \text{iff } a \in W \\ 0 & \text{iff } a \notin W \end{array} \right. .$$

Since the set U is finite, in practice its elements are represented with numbers in $\mathbb{Z}_{|U|}$ or their corresponding binary x-tuples belonging to $\mathbb{B}^x$ ($x = \lceil \log_2^{|U|} \rceil$). For a binary characteristic function, an injective function $\theta : U \rightarrow \mathbb{B}^x$ is used to map the elements in U to elements in $\mathbb{B}^x$. In general, $\chi_W(a)$ is constructed as

$$\chi_W(a) = \bigvee_{w \in W} a \leftrightarrow \theta(w), \tag{5}$$

where $\leftrightarrow$ on two binary x-tuples $\mathbf{b_1}$ and $\mathbf{b_2}$ is defined as

$$\mathbf{b_1} \leftrightarrow \mathbf{b_2} \triangleq \bigwedge_{0 \leq i < x} (b_{1i} \leftrightarrow b_{2i}), \tag{6}$$

where b_{ji} denotes the i-th element of b_j.

Hence, different set-operations can be carried out on χ using basic Boolean operators.

A. Abstraction of Tick-EFAs

As stated earlier, supervisory control on timed DES based on "tick" models has been investigated in several works such as [22], [23]. The "tick" models suffer from a major problem. The state size is very sensitive to the clock frequency: a "tick" event must be associated with the passage of each unit of time. As the clock frequency increases, so must the number of tick events. As a consequence, performing reachability analysis based on "tick" models using BDDs follows with two main issues:

1) usually many iterations are needed in the fixed point computations;
2) the intermediate BDDs representing the reachable states can be very big that may need more memory than available, i.e., state space explosion.

Following, we explain how the iterations caused by the "tick" event can be eliminated in the BDD implementation to tackle the above-mentioned issues.

The idea lies on the fact that time cannot be stopped, which indicates that all the "tick" transitions will eventually occur. For instance, consider two clocks with domains $\{0, \dots, 3\}$ and $\{0, \dots, 5\}$ and assume $(\ell, 1, 2)$ is the current state of the system. Following shows the sequence of the states that can be reached by the "tick" event:

$$(\ell, 1, 2) \overset{tick}{\mapsto} (\ell, 2, 3) \overset{tick}{\mapsto} (\ell, 3, 4) \overset{tick}{\mapsto} (\ell, 3, 5).$$

Since all "tick" transitions will eventually occur, it can be directly computed that when the state $(\ell, 1, 2)$ is reached, the states $\{(\ell, 2, 3), (\ell, 3, 4), (\ell, 3, 5)\}$ are also reachable. Hence, having a transition $(\ell, \sigma, g, \mathbf{a}, \acute{\ell})$ belonging to a clock-EFA, instead of performing the fixed-point computations on $(\ell, (\mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{\ell}, (\acute{\mu}^{\mathcal{V}}, \acute{\mu}^{\mathcal{C}}))$, we can use a *reachability transition* $(\ell, (\mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{\ell}, \acute{Q})$, where

$$\acute{Q} = \forall d \in D^{\mathcal{C}}_{\cup} : \{(\acute{l}, \acute{\mu}^{\mathcal{V}}, \varrho(\acute{\mu}^{\mathcal{C}} + \mathbf{d}))\}, \tag{7}$$

denoted as $(\ell, (\mu^{\mathcal{V}}, \mu^{\mathcal{C}})) \overset{\sigma}{\rightarrowtail} (\acute{\ell}, \acute{Q})$. The backward reachability transition relation can be defined similarly. Using the reachability transition relation in a BDD-based fixed-point computation has two advantages:

1) a number of states can be reached with a single iteration, compared to the "tick" transitions, where multiple iterations are required;
2) usually the corresponding BDD of a set of states is smaller than the intermediate BDDs resulted after executing a "tick" transition.

In [3], we shown how EFAs and their synchronous operator are transformed to BDDs. However, this transformation becomes more complicated when clocks are included in the model, specially when it comes to synchronizing the clocks with the same rate. Following we will discuss these challenges and motivate the solution we used to construct the corresponding BDD for the explicit transition relations of S_0.

Assume we have a model with a single TEFA and a single clock c_1. Lets construct the corresponding characteristic function of the explicit transition representing a partial transition $l \xrightarrow{\sigma}_{g/\mathbf{a}} \acute{l}$; for brevity, we write $\chi_{l \xrightarrow{\sigma}_{g/\mathbf{a}} \acute{l}}$. Let $\mathbf{b}^{\Sigma}$ be an $\lceil \log_2^{|\Sigma|} \rceil$-tuple of Boolean variables used to represent the events; $\mathbf{b}^L$ be an $\lceil \log_2^{|L|} \rceil$-tuple of Boolean variables used to represent the locations; $\mathbf{b}_i^{\mathcal{V}}$ be an $\lceil \log_2^{|D_i^{\mathcal{V}}|} \rceil$-tuple of Boolean variables used to represent the valuations of variable v_i. Similarly, let $\acute{\mathbf{b}}^L$ and $\acute{\mathbf{b}}_i^{\mathcal{V}}$ denote Boolean tuples used to represent the target (updated) locations and valuations of v_i after executing a transition, respectively. In [3], for the case of EFAs, we shown how a partial transition represented by its corresponding characteristic function. Based on the characteristic function in [3] and (7), the characteristic function of a partial transition with a single clock is constructed as follows,

$$\begin{aligned}
&\chi_{l \xrightarrow{\sigma}_{g/\mathbf{a}} \acute{l}}(\mathbf{b}_1^{\mathcal{V}}, \dots, \mathbf{b}_n^{\mathcal{V}}, \acute{\mathbf{b}}_1^{\mathcal{V}}, \dots, \acute{\mathbf{b}}_n^{\mathcal{V}}, \\
&\qquad\qquad \mathbf{b}_1^{\mathcal{C}}, \acute{\mathbf{b}}_1^{\mathcal{C}}, \mathbf{b}^{L}, \acute{\mathbf{b}}^{L}, \mathbf{b}^{\Sigma}) = \\
&\Big(\bigvee_{(\mu^{\mathcal{V}},\mu^{\mathcal{C}}) \models g \ \wedge \ ((\mu^{\mathcal{V}},\mu^{\mathcal{C}}),\acute{\mu}^{\mathcal{V}}) \in \mathsf{SAT}\mathcal{A}(a)} \\
&\bigwedge_{i=1}^{n} \mathbf{b}_i^{V} \leftrightarrow \theta(\mu_i^{\mathcal{V}}) \ \wedge \ \acute{\mathbf{b}}_i^{V} \leftrightarrow \theta(\acute{\mu}_i^{\mathcal{V}}) \Big) \ \wedge \\
&\mathbf{b}_1^{\mathcal{C}} \leftrightarrow \theta(\mu_1^{\mathcal{C}}) \ \wedge \ \chi_{\acute{M}_1}(\acute{\mathbf{b}}_1^{\mathcal{C}}) \ \wedge \\
&\mathbf{b}^{L} \leftrightarrow \theta(l) \ \wedge \ \acute{\mathbf{b}}^{L} \leftrightarrow \theta(\acute{l}) \ \wedge \ \mathbf{b}^{\Sigma} \leftrightarrow \theta(\sigma),
\end{aligned} \tag{8}$$

where $\acute{M}_1 = \forall d \in D_{\cup}^{\mathcal{C}} : \{\mu_1^{\mathcal{C}} + d\}$. In this case, we assumed that the clock is not reset. Otherwise, the term $\chi_{\acute{C}}(\acute{\mathbf{b}}_1^{\mathcal{C}})$ should be removed, indicating that the target valuations of the clock are all values in $D_1^{\mathcal{C}}$. However, if we follow (8) to construct a partial transition relation with multiple clocks, the clocks will not be synchronized. If we add another clock to the model, then the result will be (8) $\wedge \ \chi_{\acute{M}_2}(\acute{\mathbf{b}}_2^{\mathcal{C}})$. Thus, the term $\chi_{\acute{C}_1}(\acute{\mathbf{b}}_1^{\mathcal{C}}) \wedge \chi_{\acute{C}_2}(\acute{\mathbf{b}}_2^{\mathcal{C}})$ will yield states, where the target valuations of the clocks will be $\acute{M}_1 \times \acute{M}_2$, which clearly means that clocks do not evolve synchronously with the same rate. Instead, as mentioned earlier, the result should yield states, where the target clock valuations are:

$$\acute{M} = \forall d \in D_{\cup}^{\mathcal{C}} : \{\varrho(\mu^{\mathcal{C}} + \mathbf{d})\}. \tag{9}$$

This issue is a special case of synchronizing two TEFAs, which will be explained in the sequel.

B. BDD Construction of the Reachability Transition Relation

The construction of the BDD representing the synchronization of a number of EFAs has already been elaborated in [3]. By extending the method in [3], we construct the BDD representing $\rightarrowtail_{S_0}$ by performing the following steps:

1) consider the clocks as ordinary variables and construct the BDD of the explicit transition relation of each TEFA (which is now considered as an EFA),
2) construct the BDD representing the synchronization of all TEFAs,
3) construct a BDD representing the time evolution,
4) construct the BDD of $\rightarrowtail_{S_0}$ based on the BDDs in steps 2 and 3.

The first two steps have been described in [3]. We denote the characteristic function of the BDD from step 2 by $\chi_{\mapsto_{S_0}}$. Note that based on (1), if a clock is not reset on a transition it will keep its old valuation.

In step 3, the time evolution BDD is constructed, which will synchronously extend the target valuations of the clocks. The characteristic function of the time evolution BDD is,

$$\chi_{\text{timeEvolution}}(\acute{\mathbf{b}}_1^{\mathcal{C}}, \dots, \acute{\mathbf{b}}_n^{\mathcal{C}}, \hat{\mathbf{b}}_1^{\mathcal{C}}, \dots, \hat{\mathbf{b}}_n^{\mathcal{C}}) =$$

$$\bigvee_{\mu^{\mathcal{C}} \in D^{\mathcal{C}}} \Big(\bigwedge_{i=1}^{|\mathcal{C}|} \acute{\mathbf{b}}_i^{\mathcal{C}} \leftrightarrow \theta(\mu_i^{\mathcal{C}}) \ \wedge \ \bigvee_{d=1}^{|D_{\cup}^{\mathcal{C}}|} \bigwedge_{i=1}^{|\mathcal{C}|} \hat{\mathbf{b}}_i^{\mathcal{C}} \leftrightarrow \theta(\varrho(\mu_i^{\mathcal{C}} + d)) \Big),$$

Algorithm 1: The algorithm for constructing the BDD representing the time evolution.

```
   Input: $\mathcal{C}$ and $D^{\mathcal{C}}$
   Output: $\mathsf{B}_{\text{timeEvolution}}$
1  $\mathsf{B}_{\text{timeEvolution}} \leftarrow \mathbf{B}(1)$;
2  for $k \leftarrow 2$ to $|\mathcal{C}|$ do
3      $\mathsf{B}_{\text{forward}} \leftarrow \mathbf{B}(0)$;
4      for $i \leftarrow 0$ to $\mu max_1^{\mathcal{C}}$ do
5          for $j \leftarrow 0$ to $\mu max_k^{\mathcal{C}}$ do
6              if $i \geq j$ then
7                  $\mathsf{B}_1 \leftarrow \Big((\overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_1^{\mathcal{C}}) - \overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_k^{\mathcal{C}})) \leftrightarrow (\overrightarrow{\mathbf{B}}(i, \hat{\mathbf{b}}_1^{\mathcal{C}}) - \overrightarrow{\mathbf{B}}(j, \hat{\mathbf{b}}_k^{\mathcal{C}}))\Big)$;
               else
8                  $\mathsf{B}_1 \leftarrow \Big((\overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_k^{\mathcal{C}}) - \overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_1^{\mathcal{C}})) \leftrightarrow (\overrightarrow{\mathbf{B}}(j, \hat{\mathbf{b}}_k^{\mathcal{C}}) - \overrightarrow{\mathbf{B}}(i, \hat{\mathbf{b}}_1^{\mathcal{C}}))\Big)$;
               end
9              $\mathsf{diffG} \leftarrow (\mu max_1^{\mathcal{C}} - i)$;
10             $\mathsf{diffC} \leftarrow (\mu max_k^{\mathcal{C}} - j)$;
11             if $\mathsf{diffG} \leq \mathsf{diffC}$ then
12                 $\mathsf{B}_2 \leftarrow \Big(\mathbf{B}(\mu max_k^{\mathcal{C}}) \wedge (\overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_k^{\mathcal{C}}) \geq \overrightarrow{\mathbf{B}}(j + \mathsf{diffG}, \hat{\mathbf{b}}_k^{\mathcal{C}}))\Big)$;
               else
13                 $\mathsf{B}_2 \leftarrow \Big(\mathbf{B}(\mu max_1^{\mathcal{C}}) \wedge (\overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_1^{\mathcal{C}}) \geq \overrightarrow{\mathbf{B}}(i + \mathsf{diffC}, \hat{\mathbf{b}}_1^{\mathcal{C}}))\Big)$;
               end
14             $\mathsf{B}_{\text{forward}} \leftarrow \Big(\mathsf{B}_{\text{forward}} \vee (\mathbf{B}(i, \hat{\mathbf{b}}_1^{\mathcal{C}}) \wedge \mathbf{B}(j, \hat{\mathbf{b}}_k^{\mathcal{C}}) \wedge (\mathsf{B}_1 \vee \mathsf{B}_2))\Big)$;
           end
       end
15     $\mathsf{B}_{\text{timeEvolution}} \leftarrow (\mathsf{B}_{\text{timeEvolution}} \wedge \mathsf{B}_{\text{forward}})$;
   end
```

where $\hat{\mathbf{b}}_i^{\mathcal{C}}$ is a new set of temporary Boolean variables used to represent the valuations of clock c_i. The reason of introducing a new set of variables is related to step 4 of constructing the BDD of $\rightarrowtail_{S_0}$. Algorithm 1 shows the construction of the corresponding BDD of $\chi_{\text{timeEvolution}}$. In the algorithm, $\mathbf{B}(0)$ and $\mathbf{B}(1)$ denote the 0 and 1 terminals, respectively; $\mathbf{B}(i, \hat{\mathbf{b}}_k^{\mathcal{C}})$ is a BDD representing value i by using the Boolean variables in the tuple $\mathbf{b}$. In our implementation, we represent integers and the arithmetic operations by *BDD bit-vectors*, discussed in [30]. The notation $\overrightarrow{\mathbf{B}}(i, \hat{\mathbf{b}}_k^{\mathcal{C}})$ is the BDD bit-vector representing value i, where each bit is a BDD using the Boolean variables $\hat{\mathbf{b}}_k^{\mathcal{C}}$. $\overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_k^{\mathcal{C}})$ is the BDD bit-vector for all values that can be represented by $\hat{\mathbf{b}}_k^{\mathcal{C}}$. For a more detailed description on how arithmetic operations are performed on BDD bit-vectors refer to [30]. In line 7 clocks c_1 and c_k are synchronized, yielding a BDD, i.e. B_1, representing all pairs of valuations, where the difference is $i - j$. In lines 12 and 13 the saturation function ϱ is implemented.

The time complexity of the algorithm is $O(|C| \cdot |D_{\cup}^{\mathcal{C}}|^2 \cdot K)$, where K is the time complexity of performing the BDD operations in the loops, which is proportional to the sizes of the BDDs. The exact time complexity and the proof of correctness of the algorithm are included in [31].

Since the clocks are considered as ordinary variables, each target valuation $\acute{\mu}^{\mathcal{C}}$ in $\chi_{\mapsto S_0}$ should be synchronously extended as (9). Hence, each term $\bigwedge_{i=1}^{|\mathcal{C}|} \acute{\mathbf{b}}_i^{\mathcal{C}} \leftrightarrow \theta(\acute{\mu}_i^{\mathcal{C}})$ in $\chi_{\mapsto S_0}$ should be substituted by $\bigvee_{d=1}^{|D_{\cup}^{\mathcal{C}}|} \bigwedge_{i=1}^{|\mathcal{C}|} \acute{\mathbf{b}}_i^{\mathcal{C}} \leftrightarrow \theta(\varrho(\acute{\mu}_i^{\mathcal{C}} + d))$. The substitution is performed by utilizing some extra temporary Boolean variables mentioned in the algorithm.

Lemma 4. *The BDD representing $\rightarrowtail_{S_0}$ is constructed as follows:*

$$\mathbf{B}_{\rightarrowtail_{S_0}} = replace\Big(\exists \acute{\mathcal{B}} : (\mathbf{B}_{\rightarrowtail_{S_0}} \wedge \mathbf{B}_{timeEvolution}) \ , \ (\hat{\mathcal{B}}, \acute{\mathcal{B}})\Big).$$

The proof is included in [31].

The BDD for the backward reachability transition relation, used for coreachability, can be computed in an analogous manner.

V. Case Study: A Production Cell

In this section, the symbolic approach discussed from Section IV is applied to a benchmark example: the production cell example in [32]. The benchmark example is of interest to formal method researchers as it is complicated but still manageable. In the context of supervisory control, it has been investigated in [33] based on the *State Tree Structure* (STS) methodology and then extended to timed STS in [18].

The production cell, shown in Fig. 1, consists of six interconnected parts: feed belt, elevating rotary table, robot, press, deposit belt and traveling crane. One notable feature is that the robot has two arms to maximize the capacity of the press, namely to make it possible for the press to be forging while arm1 is picking up another metal blank. More exposition can be found in [33]. The main object is to prevent collisions among certain parts at the same time guarantee nonblocking.

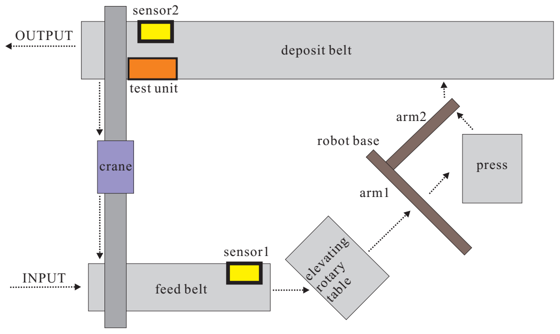

Figure 1: The Production Cell

Due to the complexity of the example and the page limitation, we only focus on the modeling of one component: the elevating rotary table. In addition, there are six specifications expressed as logic formulas to prevent the system from reaching collision states. For the sake of simplicity, those safety specifications are not taken into account. We forgo the discussion and only synthesize the nonblocking supervisor of the production cell example.

The table can move vertically and horizontally. Its task is to lift blanks to the top position and rotates by 50 degrees so that arm1 of the robot can pick them up. Subsequently, it needs to come back the bottom position with 0 degree to acquire another blank from the feed belt. In our work, we model the table as two modular TEFAs, **Ta_H**, shown in Fig. 2 and **Ta_V**, modeling the horizontal and vertical movement respectively. The complete behavior of the table can be obtained by the synchronous product **Ta_H** $\|$ **Ta_V**. As a comparison, Fig. 3 shows the corresponding "tick" model where a regular integer variable t_H, rather than a clock, is used to express the time information in an explicit way. In order to synchronize multiple clocks, two self-loops, labeled by a common event t but with different guards, are attached into every location of the model. From the modeling perspective, the TEFA model that embeds the evolution of clocks implicitly at locations, gives a more compact and comprehensible representation.

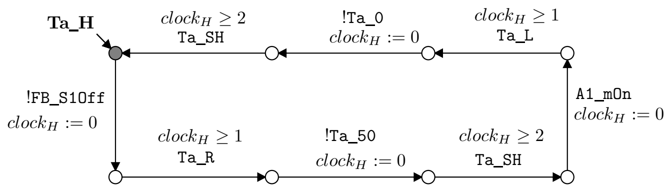

Figure 2: TEFA modeling the horizontal movement of the elevating rotary table. The description of the alphabet can be found in [33].

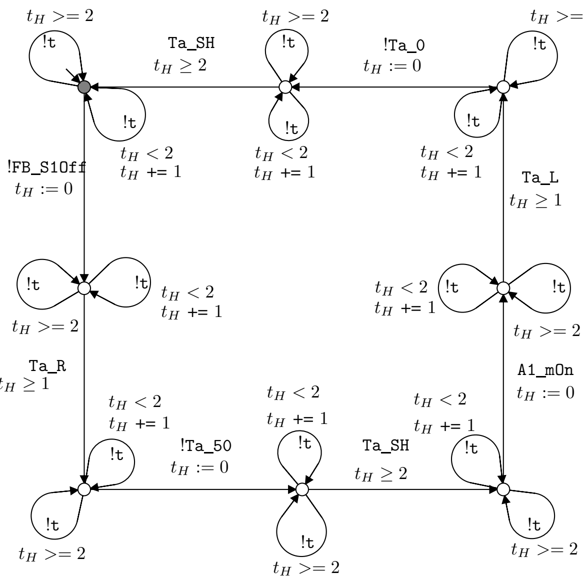

Figure 3: The corresponding tick-EFA of Ta_H.

The symbolic approach proposed in this paper has been implemented and integrated in the supervisory control tool *Supremica* [6] which uses *JavaBDD* [34] as the BDD package. The experiment is carried out on a standard personal computer (Intel Dual Core CPU @ 1.83GHz and 1.5GB RAM) running Ubuntu 10.10. Table I shows the comparison between two symbolic approaches based on tick-EFAs and TEFAs. It can be observed that both can handle the production cell example, which has 5.58×10^{10} and 3.26×10^{10} reachable and nonblocking states, respectively. The latter shows a better performance, almost twice as fast as the EFA-tick approach, due to the less number of iterations during the fixed-point computation. The second column shows the average size of the intermediate BDDs during the fixed-point computation of reachable states. The average BDD size of the TEFA approach is significantly less than the tick-EFA approach.

It should be mentioned that the result, in terms of the number of states, computed from either of those two approaches is different from the result in [18] due to distinct modeling formalisms used to model the production cell.

Table I: Nonblocking Supervisory Synthesis.

Approach	Iterations	Average BDD size	Computation Time
tick-EFA	343	39876	19sec
TEFA	129	16173	10sec

VI. Conclusions and Future Works

In this paper, we presented a method to efficiently compute a minimally restrictive nonblocking supervisor for a timed DES. The system is modeled by TEFAs, ordinary automata extended with variables and clocks, and the supervisor is symbolically computed based on the TEFAs' corresponding tick-EFAs using BDDs. However, in the BDD-based fixed-point computations, the "tick" event is eliminated by abstracting the tick-EFAs. This leads to less iterations and smaller intermediate BDDs in the fixed-point computations. As a case study, we applied our method to a classical production cell.

There are some possible directions for future work that we are currently working on. From an SCT perspective, we still does not handle controllability. From a modeling point of view, we desire to be able to model *invariants*, i.e., deadlines that must be satisfied, by TEFAs. Finally, we also desire to develop efficient algorithms for quantitative analysis such as time optimization, beside the qualitative analysis (supervisor synthesis). The interesting point about optimization on TEFAs is the existence of uncontrollable events that may lead to several optimal solutions.

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] 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.

[3] S. Miremadi, B. Lennartson, and K. Å kesson, "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.

[4] K. Bengtsson, C. Thorstensson, B. Lennartson, K. ÅKesson, S. Miremadi, and P. Falkman, "Relations identification and visualization for sequence planning and automation design," in *2010 IEEE International Conference on Automation Science and Engineering*, Aug. 2010, pp. 841–848.

[5] P. Magnusson, N. Sundström, K. Bengtsson, B. Lennartson, P. Falkman, and M. Fabian, "Planning transport sequences for flexible manufacturing systems," in *Preprints of the 18th IFAC World Congress*, Milano, Italy, 2011, pp. 9494–9499.

[6] 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'06*, Ann Arbor, MI, USA, 2006, pp. 384–385.

[7] 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.

[8] 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.

[9] Z. Fei, S. Miremadi, and K. Åkesson, "Efficient symbolic supervisory synthesis and guard generation," in *3rd International Conference on Agents and Artificial Intelligence*, Rome, Italy, 2011, pp. 106–115.

[10] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, Apr. 1994.

[11] 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.

[12] A. Khoumsi and L. Ouedraogo, "A New Method for Transforming Timed Automata," *Electronic Notes in Theoretical Computer Science*, vol. 130, pp. 101–128, May 2005.

[13] E. Asarin, O. Maler, and A. Pnueli, "Symbolic controller synthesis for discrete and timed systems," *Hybrid Systems II - Lecture Notes in Computer Science*, vol. 999, pp. 1–20, 1995.

[14] F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime, "Efficient on-the-fly algorithms for the analysis of timed games," in *In proceedings of the 16th International Conference on Concurrency Theory, CONCUR'05.* Springer, 2005, pp. 66–80.

[15] J. S. Ostroff and W. M. Wonham, "A framework for real-time discrete event control," *IEEE Transactions on Automatic Control*, vol. 35, no. 4, pp. 386–397, Apr. 1990.

[16] H. Chen and H. Li, "Maximally permissive state feedback logic for controlled time Petri nets," in *Proceedings of the 1997 American Control Conference*, vol. 4. American Autom. Control Council, 1997, pp. 2359–2363.

[17] B. A. Brandin, "The Modeling and Supervisory Control of Timed DES," in *Proceedings of the 4th International Workshop of Discrete Event Systems, WODES'98*, 1998, pp. 8–14.

[18] A. Saadatpoor, "Timed State Tree Structures: Superviory Control and Fault Diagnosis," Ph.D. dissertation, University of Toronto, 2009.

[19] B. A. Brandin and W. M. Wonham, "Supervisory Control of Timed Discrete-Event Systems," *IEEE Transactions on Automatic Control*, vol. 39, no. 2, pp. 329–342, 1994.

[20] P. Gohari and W. M. Wonham, "Reduced supervisors for timed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 48, no. 7, pp. 1187–1198, Jul. 2003.

[21] C. A. R. Hoare, "Communicating sequential processes," *Communications of the ACM*, vol. 21, no. 8, pp. 666–667, 1978.

[22] Y. Brave and M. Heymann, "Formulation and control of real time discrete event processes," in *Proceedings of the 27th IEEE Conference on Decision and Control.* IEEE, 1988, pp. 1131–1132.

[23] B. A. Brandin and W. M. Wonham, "The supervisory control of timed DES," *IEEE Transactions on Automatic Control*, vol. 39, no. 2, pp. 329–342, 1994.

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

[25] W. M. Wonham and P. Ramadge, "Modular supervisory control of discrete-event systems," *Mathematics of Control Signals and Systems*, vol. 1, no. 1, pp. 13–30, 1988.

[26] H. Flordal, R. Malik, M. Fabian, and K. Åkesson, "Compositional Synthesis of Maximally Permissive Supervisors Using Supervision Equivalence," *Discrete Event Dynamic Systems*, vol. 17, no. 4, pp. 475–504, Aug. 2007.

[27] Z. Fei, S. Miremadi, K. Å kesson, and B. Lennartson, "Efficient supervisory synthesis to large-scale discrete event systems modeled as extended finite automata," Chalmers University of Technology, Göteborg, Sweden, Tech. Rep., 2012.

[28] C. Ma and W. M. Wonham, "STSLib and its application to two benchmarks," in *9th International Workshop on Discrete Event Systems, 2008, WODES'08.*, May 2008, pp. 119–124.

[29] H. Andersen, "An introduction to binary decision diagrams," Department of Information Technology, Technical University of Denmark, Tech. Rep., 1999.

[30] 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.

[31] S. Miremadi, Z. Fei, K. Åkesson, and B. Lennartson, "Symbolic nonblocking computation of timed discrete event systems, extended version," Chalmers University of Technology, Tech. Rep., 2012.

[32] C. Lewerentz and T. Lindner, Eds., *Formal Development of Reactive Systems—Case Study Production Cell*, ser. Lecture Notes in Computer Science. Springer, 1995, vol. 891, ch. II, pp. 7–19.

[33] C. Ma, "Nonblocking Supervisory Control of State Tree Structures," Ph.D. dissertation, University of Toronto, 2005.

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