Symbolic Nonblocking Computation of 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

June 11, 2012

Abstract

In this paper, we symbolically compute a minimally restrictive nonblocking supervisor for timed discrete event systems (TDESs), 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 in the model. EFAs are ordinary automaton extended with discrete variables, guard expressions and updating functions. The main feature of this approach is that it is not based on "tick" models that have been commonly used in this area, leading to better performance in many cases. In addition, to tackle large problems all computations are based on binary decision diagrams. As a case study, we effectively computed the minimally restrictive nonblocking supervisor for a well-known production cell.

1 Introduction

Discrete Event Systems (DESs) are discrete-state, event-driven systems where their state evolution depends entirely on the occurrence of asynchronous events over time. DESs have many applications in modeling technological systems such as automated manufacturing and embedded systems, see [1, 2]. When designing control functions for DESs, 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) [3]. 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 DESs 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) [4], where time has been incorporated in 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 [5, 6, 7, 8, 9]. The EFA framework has been implemented in Supremica [10, 11], a verification and supervisory control tool, where powerful algorithms exist for analysis of DESs [12, 13, 14, 15]. A Timed EFA (TEFA) is equipped with a finite set of discrete clocks, where the value of each clock is increased implicitly as time progresses.

There have been many attempts to model TDESs 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, several models such as timed automata [16], hybrid automata [17], timed Petri nets [18, 19], and (max,+) automata [20] have been proposed. Among these models, timed automata are more popular and have been used in many research works for modeling TDESs and employing them in the SCT [21, 22, 23]

There exists a lot of work that have analyzed discrete time models with respect to SCT such as [24, 25, 26, 27, 28, 29, 30, 31]. Here it is assumed that there exists a global digital clock. In [27], a max-algebra representation is used to find the optimal behavior of a controlled timed event graph as an extremal solution to a set of inequalities defined on event occurrence times. In [30], the timing information is incorporated in 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 TDESs, described in [24, 25, 26, 28, 29, 31], 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 [32], Brandin and Wonham applied SCT to Timed Transition Models (TTMs) proposed in [26]. 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 [33, 34, 35] some methods have been proposed to reduce the state space and in [36] the state space is symbolically represented by Binary Decision Diagrams (BDDs) [37].

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. Based on TEFAs, inspired by the "zone" concept from the timed automata community [38], 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 representing time. Instead, we represent time symbolically as timing constraints, i.e., zones, using BDDs. 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 [16]), 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 7.

This paper is organized as follows. Section 3 describes Timed Extended Finite Automata that is the modeling formalism used to model our problems. In Section 4, we introduce Timed Transition Systems, which is the corresponding state transition models for the TEFAs. Section 5 explains the basics of Supervisor Control Theory. In Section 6, we explain how TEFAs and the synthesis procedure are symbolically computed by Binary Decision Diagram. As a case study, a well-known production cell has been modeled and analyzed in Section 7. Finally, Section 8 provides some conclusions and suggestions for future work.

2 Preliminaries

This section provides some preliminaries that are used throughout this paper.

2.1 Extended Finite Automata

An *Extended Finite Automaton (EFA)*, introduced in [4], is an augmentation of an ordinary finite automaton with discrete variables.

Definition 1 (Extended Finite Automaton).
An extended finite-state automaton E is a 6-tuple

$$\mathrm{E} = (L, D^{\mathcal{V}}, \Sigma, \rightarrow, L^0, D^{\mathcal{V}_0}, L^m, D^m),$$

where

- L is a finite set of locations,

- $D^{\mathcal{V}} = D_1^{\mathcal{V}} \times \ldots \times D_n^{\mathcal{V}}$ is the domain of n variables $\mathcal{V} = \{v_1, \ldots, v_n\}$, where $D_i^{\mathcal{V}} \subseteq \mathbb{Z}$,
- 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,
- $D^{\mathcal{V}_0} = D_1^{\mathcal{V}_0} \times \ldots \times D_n^{\mathcal{V}_0}$ 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 is the marked valuations of the variables,

where $\mathcal{G}$ and $\mathcal{A}$ is sets of constraining expressions, called *guards*, and updating functions, called *actions*, respectively.

The guards and actions are associated to the transitions of the automaton. A transition in an EFA is executed if and only if its corresponding event occurs and its corresponding guard becomes satisfied, which may follow by updates of a set of variables.

2.2 Binary Decision Diagrams

Binary Decision Diagrams (BDDs) are 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 [39, 13, 40, 12].

Given a set of m Boolean variables $\mathcal{B}$, a Boolean function $f\colon \mathbb{B}^m \rightarrow \mathbb{B}$ ($\mathbb{B}$ is the set of Boolean values, i.e., 0 and 1) can be expressed using Shannon's decomposition [41]. 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 $O(|\mathbf{B}_1| \cdot |\mathbf{B}_2|)$. However, the quantification operators have exponential time complexity. For a more elaborate and verbose exposition of BDDs and the implementation of different operators, refer to [42, 43].

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

Definition 2 (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:

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

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

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

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

$$\mathbf{b_1} \leftrightarrow \mathbf{b_2} \triangleq \bigwedge_{0 \le i < m} (b_{1i} \leftrightarrow b_{2i}), \tag{3}$$

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. In addition to functions, the characteristic function can also be extended to relations.

In a BDD graph, a variable b_1 has a lower (higher) *order* than variable b_2 if b_1 is closer (further) to the root and is denoted by $b_1 \prec b_2$ ($b_2 \prec b_1$). The variable ordering will impact the size of the BDD, however, finding an optimal variable ordering of a BDD is an NP-complete problem [44].

3 Timed Extended Finite Automata

A *Timed Extended Finite Automaton (TEFA)* is an EFA augmented with a finite set of digital clocks. A *clock* in a TEFA is nothing more than a discrete variable in the sense of EFAs. The time automatically elapses only at locations, whereas the transitions occur instantaneously with zero delay.

3.1 Syntax and Semantics

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

Definition 3 (Timed Extended Finite Automaton)**.**
A timed extended finite automaton is a 9-tuple

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

where

- L is a finite set of locations,
- $D^{\mathcal{V}} = D_1^{\mathcal{V}} \times \ldots \times D_n^{\mathcal{V}}$ is the domain of n variables $\mathcal{V} = \{v_1, \ldots, v_n\}$, where $D_i^{\mathcal{V}} \subseteq \mathbb{Z}$,
- $\mathcal{C}$ is a finite set of p discrete valued clocks $\{c_1, \ldots, c_p\}$,
- Σ is a nonempty finite set of events,

- $\rightarrow \subseteq L \times \Sigma \times \mathcal{G} \times \mathcal{A} \times 2^{\mathcal{C}} \times L$ is the transition relation,
- $L^0 \subseteq L$ is the set of initial locations,
- $D^{\mathcal{V}_0} = D_1^{\mathcal{V}_0} \times \ldots \times D_n^{\mathcal{V}_0}$ 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.

We assume that all clocks evolve synchronically with the same rate. Potentially, the clocks in $\mathcal{C}$ can have infinite domain because the time will elapse forever, which causes infinite state space. Based on the following reasoning we can make the state space to be finite. The relevant values of a clock are those that can impact the guards evaluations. Hence, for a clock, say c_i, values that are larger than the largest constant in a TEFA, which c_i is compared to are irrelevant, which makes it possible to make the clock domains finite. Assuming $\mu\text{max}_i^{\mathcal{C}}$ 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 $D_i^{\mathcal{C}} = \{0, 1, \ldots, \mu\text{max}_i^{\mathcal{C}}\}$. Note that for $c_i > x$, we have $\mu\text{max}^{\mathcal{C}_i} = x + 1$. Consequently, the domain of the clocks $D^{\mathcal{C}} = D_1^{\mathcal{C}} \times \ldots \times D_p^{\mathcal{C}}$ will be finite. The *global variable domain* denoted by $D_{\cup}^{\mathcal{V}}$ is the set that contains the values of all variables, defined formally as:

$$D_{\cup}^{\mathcal{V}} = \bigcup_{i=1}^{|\mathcal{V}|} D_i^{\mathcal{V}}.$$

The *global clock domain* denoted by $D_{\cup}^{\mathcal{C}}$ is defined similarly. The largest value in $D_{\cup}^{\mathcal{V}}$ and $D_{\cup}^{\mathcal{C}}$ 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 it is upon the developer to decide how to implement such cases. In contrast to variables, it is assumed that if a clock c_i exceeds its maximum value, it will not evolve anymore, keeping its maximum value until it is reset. For a clock c_i, this behavior is modeled by a saturation function $\varrho_i : D_i^{\mathcal{C}} \rightarrow D_i^{\mathcal{C}}$:

$$\varrho_i(\mu_i^{\mathcal{C}}) = \begin{cases} \mu_i^{\mathcal{C}} & \text{if } \mu_i^{\mathcal{C}} \leq \mu\text{max}_i^{\mathcal{C}} \\ \mu\text{max}_i^{\mathcal{C}} & \text{if } \mu_i^{\mathcal{C}} > \mu\text{max}_i^{\mathcal{C}} \end{cases},$$

where $\mu_i^{\mathcal{C}}$ is an evaluation of clock c_i. The function $\varrho : D^{\mathcal{C}} \rightarrow D^{\mathcal{C}}$ 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

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

where $v \in \mathcal{V}$, $c \in \mathcal{C}$, and $\omega \in D_{\cup}^{\mathcal{V}} \cup D_{\cup}^{\mathcal{C}}$. A *variable evaluation* for a variable $v_i \in \mathcal{V}$ is a function $\mu_i^{\mathcal{V}} : v_i \rightarrow D_i^{\mathcal{V}}$, assigning a value to the variable. A *clock* *evaluation* $\mu_i^{\mathcal{C}} : c_i \to D_i^{\mathcal{C}}$ 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 instead.

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

$$\begin{aligned} g \ ::= &\ \varphi < \varphi \mid \varphi \leq \varphi \mid \varphi > \varphi \mid \varphi \geq \varphi \mid \varphi == \varphi \mid \\ &(g) \mid g \wedge g \mid g \vee g \mid \top \mid \bot, \end{aligned}$$

where $\top$ and $\bot$ represent boolean logic `true` and `false`, respectively. All nonzero values are considered as $\top$. The semantics of a guard g is specified by a *satisfaction relation* $\models$ 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 $(a_1, \ldots, a_n)$, updating variables. An action function $a_i : D^{\mathcal{V}} \times D^{\mathcal{C}} \to D_i^{\mathcal{V}}$ is formed as $v_i := \varphi$. For brevity, we use the following notation:

$$\mathbf{a}(\mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \triangleq (a_1(\mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \ldots, a_n(\mu^{\mathcal{V}}, \mu^{\mathcal{C}})).$$

An action function a_i that does not update variable v_i is denoted by ξ. The symbol Ξ is used to denote a tuple $(\xi, \xi, \ldots, \xi)$, indicating that no variable is updated. The semantics of an action function can also be represented by a relation,

$$\mathsf{SAT}\mathcal{A}(\mathbf{a}) \triangleq \{((\mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \acute{\mu}^{\mathcal{V}}) \mid \acute{\mu}^{\mathcal{V}} = \mathbf{a}(\mu^{\mathcal{V}}, \mu^{\mathcal{C}})\}. \tag{4}$$

The fifth element on the transition relations $2^{\mathcal{C}}$ is a set of clocks that will be reset after executing the transition. For a set of clocks R to be reset, the clocks' valuations after the execution of the transition is defined by a function $\text{reset} : \{R\} \to D^{\mathcal{C}}$:

$$\begin{aligned} \text{reset}(R) &= (\mu_1^{\mathcal{C}}, \ldots, \mu_p^{\mathcal{C}}), \text{such that} \\ \mu_i^{\mathcal{C}} &= \begin{cases} 0 & \text{if } c_i \in R \\ \eta_i^{\mathcal{C}} & \text{if } c_i \notin R \end{cases}. \end{aligned}$$

For $d \in D_{\cup}^{\mathcal{C}}$, $\text{reset}(R) + \mathbf{d}$ is defined as follows:

$$\text{reset}(R) + \mathbf{d} \triangleq (\mu_1^{\mathcal{C}} + d, \ldots, \mu_p^{\mathcal{C}} + d),$$

where $\mathbf{d}$ is a p-tuple of values d. A partial transition relation is written as $l \xrightarrow{\sigma}_{g/\mathbf{a}/\text{reset}(R)} \acute{l}$, where $l, \acute{l} \in L$, $\sigma \in \Sigma$, $g \in \mathcal{G}$, $\mathbf{a} \in \mathcal{A}$, and $R \in 2^{\mathcal{C}}$.

For a variable v_i, $D_i^{\mathcal{V}_0}$ consists of the initial values of v_i. We assume that 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. Hence, at each state $(l, \mu^{\mathcal{V}}, \mu^{\mathcal{C}})$, the set of states $\forall d \in D_{\cup}^{\mathcal{C}} : \{(l, \mu^{\mathcal{V}}, \varrho(\mu^{\mathcal{C}} + \mathbf{d}))\}$ will eventually be reached (because time cannot be stopped). Based on this reasoning, we define a new set $D^{\mathcal{C}_0}$, which is the initial values of the clocks.

$$D^{\mathcal{C}_0} = \forall d \in D_{\cup}^{\mathcal{C}} : \{\varrho(\mathbf{d})\}.$$

Definition 4 (Deterministic TEFA)**.**
A TEFA is deterministic if

1. it only consists of a single initial location, i.e. $L^0 = \{l^0\}$,
2. each variable only has a single initial value, i.e., $D^{\mathcal{V}_0} = \{(\mu_1^{\mathcal{V}_0}, \ldots, \mu_n^{\mathcal{V}_0})\}$, and
3. at each state, when executing an event, the systems evolves to a single state.

In this work we assume that all TEFAs are deterministic.

3.2 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 [4].

A notation that will be used frequently in this paper, is the *SOS-notation* (Structured Operational Semantics) used for 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 5 (Full Synchronous Composition)**.**
Consider the following two TEFAs

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

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

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

where

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

$$\rightarrow \subseteq L_1 \times L_2 \times \Sigma \times \mathcal{G} \times \mathcal{A} \times 2^{\mathcal{C}} \times L_1 \times L_2$$

  is defined based on the following rules:

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

$$\frac{\begin{array}{c}(l_1, g_1, \mathbf{a_1}, R_1, \acute{l}_1) \in \rightarrow_1 \;\; \wedge \\ (l_2, g_2, \mathbf{a_2}, R_2, \acute{l}_2) \in \rightarrow_2\end{array}}{((l_1, l_2), \sigma, g, a, R, (\acute{l}_1, \acute{l}_2)) \in \rightarrow} \tag{5}$$

  such that,

  (i) $g = g_1 \wedge g_2$,
  (ii) $R = R_1 \cup R_2$,
  (iii) For $i = 1, \ldots, |\mathcal{V}|$,

$$\mathbf{a_i} = \begin{cases} a_{1i} & \text{if } a_{1i} = a_{2i} \\ a_{1i} & \text{if } a_{2i} = \xi \\ a_{2i} & \text{if } a_{1i} = \xi \\ \eta_i^{\mathcal{V}} & \text{otherwise} \end{cases}, \tag{6}$$

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

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

$$\frac{(l_1, \sigma, g_1, \mathbf{a_1}, R_1, \acute{l}_1) \in \rightarrow_1}{((l_1, l_2), \sigma, g_1, \mathbf{a_1}, R_1, (\acute{l}_1, \acute{l}_2)) \in \rightarrow \;\; \wedge \; l_2 = \acute{l}_2}; \tag{7}$$

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

$$\frac{(l_2, \sigma, g_2, \mathbf{a_2}, R_2, \acute{l}_2) \in \rightarrow_2}{((l_1, l_2), \sigma, g_2, \mathbf{a_2}, R_2, (\acute{l}_1, \acute{l}_2)) \in \rightarrow \;\; \wedge \; l_1 = \acute{l}_1}. \tag{8}$$

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

Similar to the proof in [45], 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.

Example 1. Consider a simple manufacturing process involving two machines, M_1 and M_2, and a buffer B with the capacity size 2 in between. When a part has been fetched and loaded on M_1, it takes at least 3 seconds and at most 6 seconds to process the part before unloading to the buffer. Similarly, once a part is loaded by M_2 from B, at least 2 seconds but no later than 4 seconds are needed to process the part.

The manufacturing cell can be modeled by two TEFAs as plants, shown in Fig. 1. Observing that instead of modeling the buffer B as a specification to prevent overflow or underflow situations from occurring, a variable, referred to as b_v, denoting the number of parts on the buffer, is declared and used in the guards to disable the occurence of certain transitions. For instance, when the buffer is full, e.g. $v_b = 2$, the transition labeled by the event `unload₁` in M_1 is not allowed to be taken unload until M_2 loads one, which cases the value of v_b decreased by 1. One more point which needs to be elaborated is these two clocks $clock_1$ and $clock_2$, which are used to express the local time passage when parts are processed by M_1 and M_2 respectively. As mentioned earlier, domains of both clocks are restricted to be finite with the upper bounds equal to the largest constants. In this case, $\mu\text{max}_1^{\mathcal{C}} = 6$ and $\mu\text{max}_1^{\mathcal{C}} = 4$. Moreover, it can be observed that once M_1 or M_2 loads a part, the value of the corresponding clock, e.g. $clock_1$ or $clock_2$ needs to be reset to 0. From the modeling perspective, by resetting clocks, not only is it convenient to count the processing time at the working stage, but also it has the possibility to make the state-space smaller.

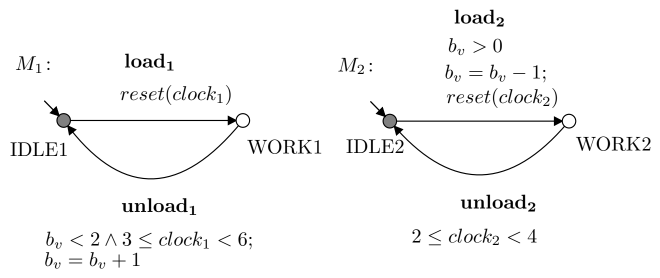

Figure 1: TEFAs modeling the manufacturing cell of Example 1, where all events are controllable and the initial locations are also marked.

4 Timed Transition Systems

The semantics of a TEFA can be represented by its corresponding *Timed Transition System (TTS)* that is based on the states of the model.

4.1 Syntax and Semantics

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

Definition 6 (Timed Transition System 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 timed transition system, denoted by $TTS(\text{TE}) = (Q, \Sigma, \rightarrowtail, Q^0, Q^m)$, is a 5-tuple where

- $Q = L \times D^{\mathcal{V}} \times D^{\mathcal{C}}$, is the finite set of states,
- Σ, is the set of events,
- $\rightarrowtail \subseteq Q \times \Sigma \times 2^Q$, is the explicit transition relation defined by the following rule:
$$\frac{(l, \sigma, g, \mathbf{a}, R, \acute{l}) \in \rightarrow \ \wedge \ (\mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \models g}{((l, \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{Q}) \in \rightarrowtail}, \tag{9}$$
where
$$\acute{Q} = \forall d \in D^{\mathcal{C}}_{\cup} : \{(\acute{l}, \mathbf{a}(\mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \varrho(\text{reset}(R) + \mathbf{d}))\},$$
- $Q^0 = L^0 \times D^{\mathcal{V}_0} \times D^{\mathcal{C}_0}$, is the set of initial states,
- $Q^m = L^m \times D^m$, is set of marked states.

Hence, in contrast to "tick" models [24], after executing each transition we reach a set of states. This feature is beneficial from different perspectives, specially in the reachability analysis. In addition, manipulating set of states is more suitable for symbolic representation, which will be described in more details in Section 6.

The explicit transition relation of a TTS can be recursively extended to strings (or sequences) of events:

$$(l, \mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \overset{\varepsilon}{\rightarrowtail} \{(l, \mu^{\mathcal{V}}, \varrho(\mu^{\mathcal{C}} + \mathbf{d})) \mid d \in D^{\mathcal{C}}_{\cup}\},$$
$$q \overset{s\sigma}{\rightarrowtail} \{q'' \in Q'' \mid q \overset{s}{\rightarrowtail} \acute{Q} \ : \ \forall \acute{q} \in \acute{Q} : \acute{q} \overset{s}{\rightarrowtail} Q''\},$$

where $s \in \Sigma^*$, $\sigma \in \Sigma$, and ε is a silent event, meaning that we will remain in the current location and the current valuations of the variables.

Example 2. Based on Definition 6, we explain the corresponding TTS of TEFA M_1 in Example 1. First of all, at the initial location IDLE1, the value of the local clock $clock_1$ could be any value in $\{0, 1, \ldots, 6\}$. Hence the initial state set consists of 7 states. Next, from *any* initial state, denoted by (IDLE1, $clock_1$, 0) where 0 is the initial value of b_v, the transition labeled by the event $\texttt{load}_1$ can be taken. The current location is then moved to WORK1 and the value of $clock_1$ is reset to 0. As the time evolves, other values of $clock_1$ can be reached successively and thus $clock_1$ could be any value in $\{0, 1, \ldots, 6\}$ again. Finally, we observe that there is a guard associating with the second transition labeled by the event $\texttt{unload}_1$ saying that the transition is enabled if the buffer is available and the time elapses at least 3 seconds but no later than 6 seconds. Therefore, as long as the buffer is available, all states with $clock_1$ from 3 to 5 can take the transition while other states are considered as blocking states, since they cannot reach the marked states.

4.2 Timed Full Synchronous Composition

Full synchronous composition can also be defined on TTSs. In this case, it is important that that the clocks evolve synchronized. In addition, it must be possible to track the partial transitions, where the variables has not been updated.

The following notations are used in the definition of timed full synchronous composition:

$$\begin{aligned}\xi\text{-ET}^i &= \{((l, \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma) \mid \forall (l, g, \mathbf{a}, R, \acute{l}) \in \rightarrow : (\mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \models g \ \wedge \ a_i = \xi\},\\ R\text{-ET}^i &= \{((l, \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma) \mid \forall (l, g, \mathbf{a}, R, \acute{l}) \in \rightarrow : (\mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \models g \ \wedge \ c_i \in R\}.\end{aligned}$$

We use the notation ξ-ET for a $|\mathcal{V}|$-tuple of ξ-ET^{i}s, referred to as ξ-expTrans. Similarly, R-ET denotes a $|\mathcal{C}|$-tuple of R-ET^{i}s.

Definition 7 (Timed Full Synchronous Composition)**.**
For two TEFAs TE$_1$ and TE$_2$, consider their corresponding TTSs

$$\begin{aligned}TTS(\text{TE}_k) &= (Q_k, \Sigma_k, \rightarrowtail_k, Q^0, Q^m),\\ Q_k &= L_k \times D^{\mathcal{V}} \times D^{\mathcal{C}},\\ Q^0 &= L_k^0 \times D^{\mathcal{V}_0} \times D^{\mathcal{C}_0},\\ Q^m &= L_k^m \times D^m,\end{aligned}$$

where $k = 1, 2$. Also let ξ-ET$_k$ be their corresponding ξ-expTrans. The Timed Full Synchronous Composition (TFSC) of $TTS(\text{TE}_1)$ and $TTS(\text{TE}_2)$, denoted by $TTS(\text{TE}_1) \| TTS(\text{TE}_2)$, is defined as

$$TTS(\text{TE}_1) \parallel TTS(\text{TE}_2) = (Q, \Sigma, \rightarrowtail, Q^0, Q^m),$$

where

- $Q = L_1 \times L_2 \times D^{\mathcal{V}} \times D^{\mathcal{C}}$,
- $\Sigma = \Sigma_1 \cup \Sigma_2$,
- the explicit transition relation

  $$\rightarrowtail \subseteq Q_1 \times Q_2 \times \Sigma \times 2^{Q_1 \times Q_2}$$

  is defined based on the following rules:

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

  $$\frac{\begin{array}{c}((l_1, \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{Q}_1) \in \rightarrowtail_1 \ \wedge\\ ((l_2, \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{Q}_2) \in \rightarrowtail_2\end{array}}{((l_1, l_2), \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{Q}) \in \rightarrowtail} \tag{10}$$

where $\acute{Q}_k = \forall d \in D^{\mathcal{C}}_{\cup} : \{(\acute{l}_k, \acute{\mu}^{\mathcal{V}}_k, \varrho(\acute{\mu}^{\mathcal{C}}_k + \mathbf{d}))\}$, $\acute{\mu}^{\mathcal{V}}_k$ and $\acute{\mu}^{\mathcal{C}}_k$ are the values of the variables and clocks immediately after executing transition $\rightarrowtail_k$, and

$$\acute{Q} = \forall d \in D^{\mathcal{C}}_{\cup} : \{((\acute{l}_1, \acute{l}_2), \acute{\mu}^{\mathcal{V}}, \varrho(\acute{\mu}^{\mathcal{C}} + \mathbf{d}))\},$$

$$\acute{\mu}^{\mathcal{V}}_i = \begin{cases} \acute{\mu}^{\mathcal{V}}_{1i} & \text{if } \acute{\mu}^{\mathcal{V}}_{1i} = \acute{\mu}^{\mathcal{V}}_{2i} \\ \acute{\mu}^{\mathcal{V}}_{2i} & \text{if } (l_1, \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma) \in \xi\text{-ET}^i_1 \\ \acute{\mu}^{\mathcal{V}}_{1i} & \text{if } (l_2, \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma) \in \xi\text{-ET}^i_2 \\ \mu^{\mathcal{V}}_i & \text{otherwise} \end{cases}, \tag{11}$$

$$\acute{\mu}^{\mathcal{C}}_i = \begin{cases} 0 & \text{if } (l_1, \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma) \in R\text{-ET}^i_1 \vee \\ & \quad (l_2, \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma) \in R\text{-ET}^i_2 \\ \mu^{\mathcal{C}}_i & \text{otherwise} \end{cases}; \tag{12}$$

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

$$\frac{((l_1, \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{Q}_1) \in \rightarrowtail_1 \ \wedge l_2 = \acute{l}_2}{((l_1, l_2), \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{Q}) \in \rightarrowtail}, \tag{13}$$

where $\acute{Q} = \forall d \in D^{\mathcal{C}}_{\cup} : \{((\acute{l}_1, \acute{l}_2), \acute{\mu}^{\mathcal{V}}_1, \varrho(\acute{\mu}^{\mathcal{C}}_1 + \mathbf{d}))\}$;

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

$$\frac{((l_2, \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{Q}_2) \in \rightarrowtail_2 \ \wedge l_1 = \acute{l}_1}{((l_1, l_2), \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{Q}) \in \rightarrowtail},$$

where $\acute{Q} = \forall d \in D^{\mathcal{C}}_{\cup} : \{((\acute{l}_1, \acute{l}_2), \acute{\mu}^{\mathcal{V}}_2, \varrho(\acute{\mu}^{\mathcal{C}}_2 + \mathbf{d}))\}$,

- $Q^0 = L^0_1 \times L^0_2 \times D^{\mathcal{V}_0} \times D^{\mathcal{C}_0}$, and
- $Q^m = L^m_1 \times L^m_2 \times D^m$.

Proposition 1. *The TFSC operator is commutative.*

Proposition 2. *The TFSC operator is associative.*

The above propositions can be proved similar to the proof in [45]. Following these two properties, it is deduced that TFSC can be extended to any number of TTSs. The relation between FSC and TFSC is described by the following lemma.

Lemma 1. *For two TEFAs*

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

where $k = 1, 2$, the following statement holds:

$$TTS(TE_1 \| TE_2) = TTS(TE_1) \| TTS(TE_2).$$

Proof. We prove the equality from left to right. Let

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

Based on definitions 6 and 5 we have

$$TTS(\mathrm{TE}_1 \| \mathrm{TE}_2) = (Q, \Sigma, \rightarrowtail, Q^0, Q^m),$$

where

- $Q = L_1 \times L_2 \times D^{\mathcal{V}} \times D^{\mathcal{C}}$,
- $\Sigma = \Sigma_1 \cup \Sigma_2$,
- $Q^0 = L_1^0 \times L_2^0 \times D^{\mathcal{V}_0} \times D^{\mathcal{C}_0}$,
- $Q^m = L_1^m \times L_2^m \times D^m$.

Based on definition 7, it can be observed that these four elements are equivalent to corresponding elements in $TTS(\mathrm{TE}_1) \| TTS(\mathrm{TE}_2)$. Now we have to prove the equivalence of their corresponding explicit transition relations. We can construct $\rightarrowtail$ by the rule in (9):

$$\frac{((l_1, l_2), \sigma, g, \mathbf{a}, R, (\acute{l}_1, \acute{l}_2)) \in \rightarrow \ \wedge \ (\mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \models g}{(((l_1, l_2), \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{Q}) \in \rightarrowtail}. \tag{14}$$

We divide the construction of $\rightarrowtail$ to three different cases:

a) $\sigma \in \Sigma_1 \cap \Sigma_2$. Based on definition 5, in (14) we have

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

   (*ii*) $R = R_1 \cup R_2$,

   (*iii*) $\acute{Q} = \forall d \in D^{\mathcal{C}}_{\cup} : \{(\acute{l}, \mathbf{a}(\mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \varrho(\mathrm{reset}(R) + \mathbf{d}))\}$,

   where $\mathbf{a}(\mu^{\mathcal{V}}, \mu^{\mathcal{C}})$ is defined according to (6). $\mathbf{a}(\mu^{\mathcal{V}}, \mu^{\mathcal{C}})$ and reset(R) can be replaced by their corresponding set-based expressions in (11) and (12). Hence, the "conclusion" parts of (10) and (14) are equivalent. Now we prove that the "premise" parts are equivalent. Based on (5) and the fact that

$$(\mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \models g_1 \wedge g_2 \equiv (\mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \models g_1 \wedge (\mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \models g_2,$$

   we can rewrite (14) as

$$\frac{\begin{array}{c}(l_1, g_1, \mathbf{a_1}, R_1, \acute{l}_1) \in \rightarrow_1 \ \wedge \ (\mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \models g_1 \ \wedge \\ (l_2, g_2, \mathbf{a_2}, R_2, \acute{l}_2) \in \rightarrow_2 \ \wedge \ (\mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \models g_2\end{array}}{((l_1, l_2), \sigma, g, a, R, (\acute{l}_1, \acute{l}_2)) \in \rightarrow},$$

   and based on (9), the above equation can be written as

$$\frac{\begin{array}{c}((l_1, \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{Q}_1) \in \rightarrowtail_1 \ \wedge \\ ((l_2, \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{Q}_2) \in \rightarrowtail_2\end{array}}{((l_1, l_2), \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{Q}) \in \rightarrowtail},$$

where
$$\acute{Q}_k = \forall d \in D_{\cup}^{\mathcal{C}} : \{(\acute{l}_k, a_k(\mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \varrho(\text{reset}(R_k) + \mathbf{d}))\}.$$

Hence, the "premise" parts of (10) and (14) are also equivalent.

b) $\sigma \in \Sigma_1 \backslash \Sigma_2$. Based on definition 5, we can rewrite (14) as follows,

$$\frac{((l_1, l_2), \sigma, g_1, \mathbf{a_1}, R_1, (\acute{l}_1, \acute{l}_2)) \in \rightarrow \ \wedge \ (\mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \models g_1}{(((l_1, l_2), \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{Q}) \in \rightarrowtail}, \tag{15}$$

where $\acute{Q} = \forall d \in D_{\cup}^{\mathcal{C}} : \{(\acute{l}, \acute{\mu}_1^{\mathcal{V}}, \varrho(\acute{\mu}_1^{\mathcal{C}} + \mathbf{d}))\}$, such that $\acute{\mu}_k^{\mathcal{V}} = \mathbf{a_1}(\mu^{\mathcal{V}}, \mu^{\mathcal{C}})$ and $\acute{\mu}_1^{\mathcal{C}} = \text{reset}(R)$. Hence, the "conclusion" parts of (13) and (14) are equivalent. Now we prove that the "premise" parts are equivalent. Based on (7), we can rewrite (15) as

$$\frac{(l_1, \sigma, g_1, \mathbf{a_1}, R_1, \acute{l}_1) \in \rightarrow_1 \ \wedge \ l_2 = \acute{l}_2}{(((l_1, l_2), \mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \sigma, \acute{Q}) \in \rightarrowtail},$$

which proves that the "premise" parts of (13) and (14) are also equivalent.

c) $\sigma \in \Sigma_2 \backslash \Sigma_1$. This case can be proved similar to part b).

Hence, based on definition 7, the tuple of $TTS(\text{TE}_1 \| \text{TE}_2)$ is equivalent to $TTS(\text{TE}_1) \| TTS(\text{TE}_2)$. $\square$

Since both FSC and TFSC are both commutative associative, the lemma can be extended to any number of TEFAs.

5 Supervisory Control Theory

Supervisory Control Theory (SCT) [3, 46] is a general theory to automatically synthesize a control function, referred to as *supervisor*, based on a given plant and 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.

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 TTS models. So from now on, wherever a model is mentioned, we mean its corresponding TTS model.

A plant P can be described by the synchronization of a number of sub-plants $P = P_1 \| P_2 \| \ldots \| P_l$, and similarly for a specification $Sp = Sp_1 \| Sp_2 \| \ldots \| Sp_m$.

There are different ways of computing a supervisor such as monolithic [3], modular [47], and compositional [48] 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. In SCT, the events can be divided into two disjoint subsets: *controllable events*, that can be prevented from executing by the supervisor; and *uncontrollable events*, which cannot be influenced by the supervisor [3, 46]. In addition to the blocking states, the synthesis procedure may also identify some *uncontrollable* states, which are states from where the plant executes an uncontrollable event that violates the specification. By removing the blocking or controllable states from S_0, a *nonblocking* or *controllable* supervisor is obtained, respectively. In this paper, we aim to compute a nonblocking supervisor and leave the controllability issues for future work.

As mentioned earlier, we assume that the systems are deterministic, in the sense that at each moment, the supervisor knows the current and next state of the system. Therefore, we only consider deterministic TEFAs. This might seem to be in conflict with the definition of TTS, where give a state and an event, the system will be in a set of states. However, based on the following reasoning, the supervisor will always know the current state. Assume that there exists a global clock that always evolves and never gets reset. Thus, each state in the system will also consist of the current value of the global clock. Consequently, after the execution of any event, the supervisor will always know the next state of the system.

Following, we provide formal definitions of the nonblocking property.

Definition 8 (Reachable states)**.** The set of *reachable* states of a TTS are the states that can be reached from the initial state:

$$Q^{reach} \triangleq \{\acute{q} \in \acute{Q} \mid \exists s \in \Sigma^* : \; q_0 \overset{s}{\rightarrowtail} \acute{Q}\}$$

Definition 9 (Coreachable states)**.** The set of *coreachable* states of a TTS are the states that can reach at least one marked state:

$$Q^{coreach} \triangleq \{q \mid \exists s \in \Sigma^* : \; q \overset{s}{\rightarrowtail} \acute{Q} \wedge (\acute{Q} \cap Q^m) \neq \emptyset\}$$

In other words, the coreachable states are the states that can be reached if starting by a marked stated and executing the transitions in reverse order. In our work, the computation of coreachable states is based on this paradigm. We define the *explicit backward transition relation*, denoted as $\leftarrowtail$, by the following rule,

$$\frac{\begin{array}{c}(l, \sigma, g, \mathbf{a}, R, \acute{l}) \in \rightarrow \; \wedge \; ((\mu^{\mathcal{V}}, \mu^{\mathcal{C}}), \acute{\mu}^{\mathcal{V}}) \in \mathsf{SAT}\mathcal{A}(\mathbf{a}) \; \wedge \\ (\mu^{\mathcal{V}}, \mu^{\mathcal{C}}) \models g \; \wedge \; \acute{\mu}^{\mathcal{C}} = \mu^{\mathcal{C}}\end{array}}{((\acute{l}, \acute{\mu}^{\mathcal{V}}, \acute{\mu}^{\mathcal{C}}), \sigma, Q) \in \leftarrowtail}, \tag{16}$$

where

$$\acute{Q} = \forall d \in D^{\mathcal{C}}_{\cup} : \{(l, \mu^{\mathcal{V}}, \rho(\acute{\mu}^{\mathcal{C}} - \mathbf{d})) \mid \forall c_i \in R : \acute{\mu}^{\mathcal{C}}_i = 0\},$$

and ρ is a $|\mathcal{C}|$-tuple function defined as

$$\rho_i(\mu_i^{\mathcal{C}}) = \left\{ \begin{array}{ll} \mu_i^{\mathcal{C}} & \text{if } \mu_i^{\mathcal{C}} > 0 \\ 0 & \text{if } \mu_i^{\mathcal{C}} \leq 0 \end{array} \right. .$$

Based on the backward transition relation, the set of coreachable states can also be defined as:

$$Q^{coreach} \triangleq \{q \in Q \mid \forall \acute{q} \in Q^m : \exists s \in \Sigma^* : \ Q \overset{s}{\leftarrowtail} \acute{q}\}.$$

In the sequel, to avoid any ambiguity, we will refer to $\rightarrowtail$ as *explicit forward transition relation.*

Definition 10 (Nonblocking states)**.** The *nonblocking* states of a TTS are the states that are both reachable and coreachable:

$$Q^{nbl} \triangleq Q^{reach} \cap Q^{coreach}.$$

Definition 11 (Nonblocking TTS)**.** A TTS is *nonblocking* if all reachable states are nonblocking, i.e., the following property holds:

$$Q^{reach} = Q^{nbl}.$$

A supervisor S is nonblocking with respect to a plant G if $G\|S$ is nonblocking.

The nonblocking states can be synthesized by fixed point computations [12] on the synchronized model S_0. There core operator in performing the fixed point computations is the `IMAGE` operator. Given a set of states $W \subseteq Q$; $\texttt{IMAGE}(W, \rightarrowtail)$ computes the set of states that can be reached in one transition,

$$\texttt{IMAGE}(W, \rightarrowtail) \triangleq \{\acute{q} \in \acute{Q} | \ \exists q \in W, \sigma \in \Sigma : q \overset{\sigma}{\rightarrowtail} \acute{Q}\}, \tag{17}$$

where $\rightarrowtail$ is the transition relation of S_0. By this operator the set of reachable states can be computed. By applying the `IMAGE` operator on the backward transition relation, the set of states that can reach states in W can be computed (used in computing the coreachable states).

6 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 BDDs.

In [14, 49], it is shown how EFAs are represented by BDDs and how the synthesis procedure is performed efficiently using BDDs. In this work, we extend the framework to TEFAs. As mentioned earlier, all computations are based on the corresponding transition relations of the TTSs. The main feature of

TTSs is that from any state and a given event, a set of states can be reached directly. This is in contrast to "tick" models [26], where the clocks evolve on the occurrence of the "tick" event. Hence, when performing the fixed point computations on TTSs, less number of iterations is needed compared to the "tick" models. Another advantages of TTSs is the fact that a *set* of states (the target states) usually has a more compact BDD representation compared to a single state. *The combination of using TTSs and implementing them by BDDs is the main reason behind the effectiveness of our approach.*

As mentioned in Section 5, the fixed point computations are performed on $\rightarrowtail_{S_0}$ and $\leftarrowtail_{S_0}$, i.e., the forward and backward explicit transition relations of S_0. However, there are some challenges in constructing the BDD representing the explicit transition relations of S_0. In [49], we shown how EFAs and their synchronous operator are transformed to BDDs. Nevertheless, 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. In the sequel, as mentioned earlier in Section 2.2, we will represent the BDDs by their corresponding characteristic functions. For brevity, we only focus on the forward transition relation; the backward transition relation can be computed in an analogous manner.

6.1 Discussion

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 $l \xrightarrow{\sigma}_{g/\mathbf{a}/\text{reset}(R)} \acute{l}$; for brevity, we write $\chi_{l \xrightarrow{\sigma}_{g/\mathbf{a}/\text{reset}(R)} \acute{l}}$. 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 [49], for the case of EFAs, we shown that a partial transition without clocks is represented by the following characteristic function:

$$\begin{aligned} &\chi_{l \xrightarrow{\sigma}_{g/\mathbf{a}} \acute{l}}(\mathbf{b}_1^{\mathcal{V}}, \ldots, \mathbf{b}_n^{\mathcal{V}}, \acute{\mathbf{b}}_1^{\mathcal{V}}, \ldots, \acute{\mathbf{b}}_n^{\mathcal{V}}, \mathbf{b}^{L}, \acute{\mathbf{b}}^{L}, \mathbf{b}^{\Sigma}) = \\ &\quad \Big(\bigvee_{\mu^{\mathcal{V}} \models g \;\wedge\; (\mu^{\mathcal{V}}, \acute{\mu}^{\mathcal{V}}) \in \mathsf{SAT}\mathcal{A}(a)} \\ &\quad \bigwedge_{i=1}^{n} \mathbf{b}_i^{\mathcal{V}} \leftrightarrow \theta(\mu_i^{\mathcal{V}}) \;\wedge\; \acute{\mathbf{b}}_i^{\mathcal{V}} \leftrightarrow \theta(\acute{\mu}_i^{\mathcal{V}}) \Big) \;\wedge \\ &\quad \mathbf{b}^{L} \leftrightarrow \theta(l) \;\wedge\; \acute{\mathbf{b}}^{L} \leftrightarrow \theta(\acute{l}) \;\wedge\; \mathbf{b}^{\Sigma} \leftrightarrow \theta(\sigma), \end{aligned} \tag{18}$$

which can be deduced from (2) and (3). The characteristic function of the transition relation of the TEFA can be computed by disjuncting the corresponding characteristic functions of all partial transition relations. Now, lets include a single clock in the model, include it in the guard, but treat it as an ordinary variable:

$$\begin{aligned}&\chi_{l \xrightarrow{\sigma}_{g/\mathbf{a}/\mathrm{reset}(\emptyset)} \acute{l}}(\mathbf{b}_1^{\mathcal{V}}, \dots, \mathbf{b}_n^{\mathcal{V}}, \acute{\mathbf{b}}_1^{\mathcal{V}}, \dots, \acute{\mathbf{b}}_n^{\mathcal{V}}, \\ &\qquad\qquad \mathbf{b}_1^{\mathcal{C}}, \acute{\mathbf{b}}_1^{\mathcal{C}}, \mathbf{b}^{L}, \acute{\mathbf{b}}^{L}, \mathbf{b}^{\Sigma}) = \\ &\Big(\bigvee_{(\mu^{\mathcal{V}}, \mu_1^{\mathcal{C}}) \models g \ \wedge \ ((\mu^{\mathcal{V}}, \mu_1^{\mathcal{C}}), \acute{\mu}^{\mathcal{V}}) \in \mathsf{SAT}\mathcal{A}(a)} \\ &\bigwedge_{i=1}^{n} \mathbf{b}_i^{\mathcal{V}} \leftrightarrow \theta(\mu_i^{\mathcal{V}}) \ \wedge \ \acute{\mathbf{b}}_i^{\mathcal{V}} \leftrightarrow \theta(\acute{\mu}_i^{\mathcal{V}}) \ \wedge \\ &\mathbf{b}_1^{\mathcal{C}} \leftrightarrow \theta(\mu_1^{\mathcal{C}}) \ \wedge \ \acute{\mathbf{b}}_1^{\mathcal{C}} \leftrightarrow \theta(\acute{\mu}_1^{\mathcal{C}}) \Big) \ \wedge \\ &\mathbf{b}^{L} \leftrightarrow \theta(l) \ \wedge \ \acute{\mathbf{b}}^{L} \leftrightarrow \theta(\acute{l}) \ \wedge \ \mathbf{b}^{\Sigma} \leftrightarrow \theta(\sigma). \end{aligned} \tag{19}$$

To implement the time evolution and give the clock its real semantics, based on (9), the term $\acute{\mathbf{b}}_1^{\mathcal{C}} \leftrightarrow \theta(\acute{\mu}_1^{\mathcal{C}})$ should be replaced by $\chi_{\acute{M}_1}(\acute{\mathbf{b}}_1^{\mathcal{C}})$, 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 $\chi_{\acute{M}_1}(\acute{\mathbf{b}}_1^{\mathcal{C}})$ should be removed, indicating that the target valuations of the clock are all values in $D_1^{\mathcal{C}}$. However, if we follow the above formula 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 above result will be logically conjuncted $\mathbf{b}_2^{\mathcal{C}} \leftrightarrow \theta(\mu_2^{\mathcal{C}}) \wedge \chi_{\acute{M}_2}(\acute{\mathbf{b}}_2^{\mathcal{C}})$. Thus, the term $\chi_{\acute{M}_1}(\acute{\mathbf{b}}_1^{\mathcal{C}}) \wedge \chi_{\acute{M}_2}(\acute{\mathbf{b}}_2^{\mathcal{C}})$ 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.

Hence, when there exists $p > 1$ clocks, to get the correct result, the statement in (19) should be:

$$\bigwedge_{i=1}^{p} \mathbf{b}_i^{\mathcal{C}} \leftrightarrow \theta(\mu_i^{\mathcal{C}}) \ \wedge \ \chi_{\acute{M}}(\acute{\mathbf{b}}^{\mathcal{C}}),$$

where $\acute{M} = \forall d \in D_{\cup}^{\mathcal{C}} : \{\varrho(\mu^{\mathcal{C}} + \mathbf{d})\}$ and thus

$$\chi_{\acute{M}}(\acute{\mathbf{b}}^{\mathcal{C}}) = \bigvee_{d=0}^{|D_{\cup}^{\mathcal{C}}|} \bigwedge_{j=1}^{p} \acute{\mathbf{b}}_j^{\mathcal{C}} \leftrightarrow \theta(\varrho(\mu_j^{\mathcal{C}} + d)). \tag{20}$$

Having this in mind, in the next section we show how the corresponding BDD of the transition relation representing the synchronization of a number of timed EFAs can be computed.

6.2 BDD Construction of $\rightarrowtail_{S_0}$

The construction of the BDD representing the synchronization of a number of EFAs has already been elaborated in [14]. By extending the method in [14], 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. consider the time evolution BDD from step 3 in the BDD from step 2.

The first two steps have been described in [14]. We denote the characteristic function of the BDD from step 2 by $\chi_{\mapsto_{S_0}}$. Note that based on (6), if a clock is not reset on a transition it will keep its old valuation.

In step 3, we implement the *time evolution* by constructing a BDD that represents (20) for all clock valuations $\mu^{\mathcal{C}}$. As mentioned earlier, the main idea is to for each transition *expand* the target value of each clock,say $\acute{\mu}^{\mathcal{C}}_i$, to all larger values in its domain, i.e., $\forall d \in D^{\mathcal{C}}_i : \{\acute{\mu}^{\mathcal{C}}_i + d\}$. We notate the expand operator by $\rightsquigarrow$. For instance, if a clock has domain $\{0, 1, \dots, 10\}$, then a target value 4 is expanded to $\{4, 5, \dots, 10\}$ and denoted as $4 \rightsquigarrow \{4, 5, \dots, 10\}$. The main issue is to make all the clocks expand their values synchronously. Moreover, we use the expand operator to replace the target value by a set of values. The replacement occurs in step 4 and to do this on the BDD level, in addition to $\acute{\mathbf{b}}^{\mathcal{C}}$, we introduce a set of temporary Boolean variables $\hat{\mathbf{b}}^{\mathcal{C}}$.

Before continuing, as an example, we apply steps 3 and 4 to the BDD representing (19). We first compute the BDD representing time evolution for all values in $D^{\mathcal{C}_1}$:

$$\bigvee_{\mu^{\mathcal{C}_1} \in D^{\mathcal{C}}_1} \Big(\acute{\mathbf{b}}^{\mathcal{C}}_1 \leftrightarrow \theta(\acute{\mu}^{\mathcal{C}}_1) \ \wedge \bigvee_{d=0}^{|D^{\mathcal{C}}_1| - \mu^{\mathcal{C}}_1} \hat{\mathbf{b}}^{\mathcal{C}}_1 \leftrightarrow \theta(\acute{\mu}^{\mathcal{C}}_1 + d) \Big). \tag{21}$$

Next, we compute (21) $\wedge$ (19) yielding:

$$\begin{aligned} &\Big(\bigvee_{(\mu^{\mathcal{V}}, \mu^{\mathcal{C}}_1) \models g \ \wedge \ ((\mu^{\mathcal{V}}, \mu^{\mathcal{C}}_1), \acute{\mu}^{\mathcal{V}}) \in \mathsf{SAT}\mathcal{A}(a)} \\ &\bigwedge_{i=1}^{n} \mathbf{b}^{\mathcal{V}}_i \leftrightarrow \theta(\mu^{\mathcal{V}}_i) \ \wedge \ \acute{\mathbf{b}}^{\mathcal{V}}_i \leftrightarrow \theta(\acute{\mu}^{\mathcal{V}}_i) \ \wedge \\ &\mathbf{b}^{\mathcal{C}}_1 \leftrightarrow \theta(\mu^{\mathcal{C}}_1) \ \wedge \ \acute{\mathbf{b}}^{\mathcal{C}}_1 \leftrightarrow \theta(\mu^{\mathcal{C}}_1) \ \wedge \bigvee_{d=0}^{|D^{\mathcal{C}}_1| - \mu^{\mathcal{C}}_1} \hat{\mathbf{b}}^{\mathcal{C}}_1 \leftrightarrow \theta(\mu^{\mathcal{C}}_1 + d) \Big) \ \wedge \\ &\mathbf{b}^{L} \leftrightarrow \theta(l) \ \wedge \ \acute{\mathbf{b}}^{L} \leftrightarrow \theta(\acute{l}) \ \wedge \ \mathbf{b}^{\Sigma} \leftrightarrow \theta(\sigma). \end{aligned} \tag{22}$$

Let $\mathbf{B}$ be the BDD representing (22). The final step is to quantify away the Boolean variables in $\acute{\mathbf{b}}^{\mathcal{C}}_1$ and then replace the variables in $\hat{\mathbf{b}}^{\mathcal{C}}_1$ by $\acute{\mathbf{b}}^{\mathcal{C}}_1$,

$$\mathit{replace}\Big(\exists \acute{\mathbf{b}}^{\mathcal{C}} : \mathbf{B} \ , \ (\hat{\mathbf{b}}^{\mathcal{C}}, \acute{\mathbf{b}}^{\mathcal{C}}) \Big),$$

which represents the following characteristic function

$$\Big(\bigvee_{(\mu^{\mathcal{V}},\mu_1^{\mathcal{C}})\models g \ \wedge \ ((\mu^{\mathcal{V}},\mu_1^{\mathcal{C}}),\acute{\mu}^{\mathcal{V}})\in \mathsf{SAT}\mathcal{A}(a)}$$
$$\bigwedge_{i=1}^{n} \mathbf{b}_i^{\mathcal{V}} \leftrightarrow \theta(\mu_i^{\mathcal{V}}) \ \wedge \ \acute{\mathbf{b}}_i^{\mathcal{V}} \leftrightarrow \theta(\acute{\mu}_i^{\mathcal{V}}) \ \wedge$$
$$\mathbf{b}_1^{\mathcal{C}} \leftrightarrow \theta(\mu_1^{\mathcal{C}}) \ \wedge \bigvee_{d=0}^{|D_1^{\mathcal{C}}|-\mu_1^{\mathcal{C}}} \acute{\mathbf{b}}_1^{\mathcal{C}} \leftrightarrow \theta(\mu_1^{\mathcal{C}}+d)\Big) \ \wedge$$
$$\mathbf{b}^{L} \leftrightarrow \theta(l) \ \wedge \ \acute{\mathbf{b}}^{L} \leftrightarrow \theta(\acute{l}) \ \wedge \ \mathbf{b}^{\Sigma} \leftrightarrow \theta(\sigma).$$

Hence, each value $\mu_1^{\mathcal{C}}$ represented by $\acute{\mathbf{b}}_1^{\mathcal{C}}$ has been replaced by $\{\mu_1^{\mathcal{C}}, \mu_1^{\mathcal{C}}+1, \ldots, |D_1^{\mathcal{C}}|\}$.

Algorithm 1 shows the construction of the BDD representing the time evolution (expand operator) having the following characteristic function:

$$\chi_{\text{timeEvolution}}(\acute{\mathbf{b}}_1^{\mathcal{C}}, \ldots, \acute{\mathbf{b}}_n^{\mathcal{C}}, \hat{\mathbf{b}}_1^{\mathcal{C}}, \ldots, \hat{\mathbf{b}}_n^{\mathcal{C}}) =$$
$$\bigvee_{\mu^{\mathcal{C}}\in D^{\mathcal{C}}} \Big(\bigwedge_{i=1}^{|\mathcal{C}|} \acute{\mathbf{b}}_i^{\mathcal{C}} \leftrightarrow \theta(\mu_i^{\mathcal{C}}) \ \wedge \bigvee_{d=0}^{|D_{\cup}^{\mathcal{C}}|} \bigwedge_{j=1}^{|\mathcal{C}|} \hat{\mathbf{b}}_j^{\mathcal{C}} \leftrightarrow \theta(\varrho(\mu_j^{\mathcal{C}}+d))\Big). \tag{23}$$

Before digging into the algorithm, it is worth to describe the principle behind implementing the saturation function ϱ, which is especially an issue when there exist multiple clocks. The basic idea is to let the values of the clocks grow even when they exceed the domains of the clocks. In other words, we enlarge the domains of the clocks. Then, in the next step, all values outside of the domain will be replace by the largest value. For instance, assume there exist two clocks each with domain $\{0, 1, 2\}$ and let $(0, 1)$ be a tuple which we want to expand. Without enlarging the domain the result is $(0, 1) \rightsquigarrow \{(0, 1), (1, 2)\}$, however, changing the domain to $\{0, 1, 2, 3\}$ the result would be $(0, 1) \rightsquigarrow \{(0, 1), (1, 2), (2, 3)\}$. Finally, value 3 will be replaced by 2, i.e., the largest value in the old domain, yielding $(0, 1) \rightsquigarrow \{(0, 1), (1, 2), (2, 2)\}$, which will be the result of the ϱ function. This implementation could be done in other ways to but the main reason that we want to enlarge the domain is to always have unique values in the tuples, which is necessary for the correctness of the algorithm. Note that the increase of the domain will be applied to the temporary Boolean variables $\hat{\mathbf{b}}_j^{\mathcal{C}}$. To ensure that the values always increase when we want to expand a tuple of multiple clocks, we let $\hat{\mathbf{b}}_j^{\mathcal{C}}$ include $\lceil \log(2 \cdot |D_j^{\mathcal{C}}|)\rceil$ Boolean variables.

In the algorithm, $\mathbf{B}(0)$ and $\mathbf{B}(1)$ denote the 0 and 1 terminals, respectively; $\mathbf{b}_k^{\mathcal{C}}$ represents a number of Boolean variables used to represent $D_k^{\mathcal{C}}$; $\mathbf{b}^{\mathcal{C}}$ represents a number of Boolean variables used to represent $D^{\mathcal{C}}$; and $\mathbf{B}(i, \mathbf{b}_k^{\mathcal{C}})$ corresponds to the BDD representing value i by using $\mathbf{b}_k^{\mathcal{C}}$. In our implementation, we represent integers and the arithmetic operations by *BDD bit-vectors* [50]. The notation $\overrightarrow{\mathbf{B}}(i, \mathbf{b}_k^{\mathcal{C}})$ is the BDD bit-vector representing value i, where each bit is a BDD using $\mathbf{b}_k^{\mathcal{C}}$. $\overrightarrow{\mathbf{B}}(\cdot, \mathbf{b}_k^{\mathcal{C}})$ is the BDD bit-vector for all values that can be represented by

$\mathbf{b}_k^{\mathcal{C}}$. For a more detailed description on how arithmetic operations are performed on BDD bit-vectors refer to [50].

Lines 2-7 constructs the time evolution when there is only a single clock in the model. In this case,

$$\chi_{\text{timeEvolution}}(\acute{\mathbf{b}}_1^{\mathcal{C}}, \ldots, \acute{\mathbf{b}}_n^{\mathcal{C}}, \hat{\mathbf{b}}_1^{\mathcal{C}}, \ldots, \hat{\mathbf{b}}_n^{\mathcal{C}}) =$$
$$\bigvee_{\mu_1^{\mathcal{C}} \in D_1^{\mathcal{C}}} \Big(\acute{\mathbf{b}}_1^{\mathcal{C}} \leftrightarrow \theta(\mu_1^{\mathcal{C}}) \ \wedge \bigvee_{d=0}^{|D_1^{\mathcal{C}}|} \hat{\mathbf{b}}_1^{\mathcal{C}} \leftrightarrow \theta(\varrho(\mu_1^{\mathcal{C}} + d)) \Big). \tag{24}$$

This represents the set

$$\begin{aligned} \{\ & 0 \rightsquigarrow \{0, \ldots, 2 \cdot |D_1^{\mathcal{C}}|\}, \\ & 1 \rightsquigarrow \{1, \ldots, 2 \cdot |D_1^{\mathcal{C}}|\}, \\ & \ldots, \\ & |D_1^{\mathcal{C}}| \rightsquigarrow \{2 \cdot |D_1^{\mathcal{C}}|\}\ \}, \end{aligned} \tag{25}$$

Lines 8-19 synchronize the clocks without considering the saturation function ϱ. The basic idea is to synchronize each clock in the model with the first clock c_1 and conjunct it with the BDD that has been computed so far for the previous clocks. In line 16, B_{diffG} represents all valuation pairs larger than i and j for clocks c_1 and c_k, respectively, where the difference is $i - j$. $B_{ijTarg} \wedge B_{diffG}$ will then represent

$$(i, j) \rightsquigarrow \{(i, j),\ (i+1, j+1),\ \ldots\ ,\ (i+\omega, j+\omega)\}, \tag{26}$$

where $\omega = \min\big((2 \cdot |D_1^{\mathcal{C}}| - i), (2 \cdot |D_k^{\mathcal{C}}| - j)\big)$. Such a BDD will be constructed for all is and js in $D_1^{\mathcal{C}}$ and $D_k^{\mathcal{C}}$, respectively, and will be disjuncted together, stored in B_{frwd}. Then, B_{frwd} will be conjuncted with $B_{frwdEvol}$ that represents the time evolution for the clocks that have been computed so far. In lines 20-24 the saturation function ϱ is implemented. As mentioned earlier, for each clock c_k all values that are larger than $|D_k^{\mathcal{C}}|$ will be replaced by $|D_k^{\mathcal{C}}|$.

Lemma 2. *Algorithm 1 returns the corresponding BDD for* $\chi_{timeEvolution}$.

Proof. Without loss of generality, we perform the proof based on the $\rightsquigarrow$ symbol introduced earlier, rather than using the characteristic functions.
When $|\mathcal{C}| = 1$, $\chi_{\text{timeEvolution}}$ will be equal to (24). In this case, only lines 1-7 will be executed and it is straightforward to see that B_{frwd} represents (25).
For $|\mathcal{C}| \geq 1$, we divide the algorithm into two parts: lines 8-19 and lines 20-24 (saturation implementation), and prove the correctness of each part separately.
For lines 8-19, we prove that $B_{frwdEvol}$ represents

$$\{\mu^{\mathcal{C}} \rightsquigarrow (\mu^{\mathcal{C}}, \mu^{\mathcal{C}} + \mathbf{1}, \ldots, \mu^{\mathcal{C}} + \omega) \mid \mu^{\mathcal{C}} \in D^{\mathcal{C}}\ :\ \omega = \min_{p=1}^{|\mathcal{C}|}(2 \cdot |D_p^{\mathcal{C}}| - \mu_p^{\mathcal{C}})\}. \tag{27}$$

We prove this by induction.
For the basic case, where $k = 2$, from (26) it can be directly deduced that at

```
    Input: $\mathcal{C}$ and $D^{\mathcal{C}}$
    Output: $\mathsf{B}_{\text{frwdEvol}}$
 1  $\mathsf{B}_{\text{frwdEvol}} \leftarrow \mathbf{B}(1)$;
 2  if $|\mathcal{C}| = 1$ then
 3      $\mathsf{B}_{\text{frwd}} \leftarrow \mathbf{B}(0)$;
 4      for $i \leftarrow 0$ to $|D_1^{\mathcal{C}}|$ do
 5          $\mathsf{B}_{geq} \leftarrow \Big(\overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_1^{\mathcal{C}}) \geq \overrightarrow{\mathbf{B}}(i, \hat{\mathbf{b}}_1^{\mathcal{C}})\Big)$;
 6          $\mathsf{B}_{\text{frwd}} \leftarrow \Big(\mathsf{B}_{\text{frwd}} \ \vee \big(\mathbf{B}(i, \acute{\mathbf{b}}_1^{\mathcal{C}}) \wedge \mathsf{B}_{geq}\big)\Big)$;
        end
 7      $\mathsf{B}_{\text{frwdEvol}} \leftarrow \mathsf{B}_{\text{frwd}}$;
    else
 8      for $k \leftarrow 2$ to $|\mathcal{C}|$ do
 9          $\mathsf{B}_{\text{frwd}} \leftarrow \mathbf{B}(0)$;
10          for $i \leftarrow 0$ to $|D_1^{\mathcal{C}}|$ do
11              for $j \leftarrow 0$ to $|D_k^{\mathcal{C}}|$ do
12                  if $i \geq j$ then
13                      $\mathsf{B}_{diff} \leftarrow \Big(\big(\overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_1^{\mathcal{C}}) - \overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_k^{\mathcal{C}})\big) \leftrightarrow \big(\overrightarrow{\mathbf{B}}(i, \hat{\mathbf{b}}_1^{\mathcal{C}}) - \overrightarrow{\mathbf{B}}(j, \hat{\mathbf{b}}_k^{\mathcal{C}})\big)\Big)$;
                  else
14                      $\mathsf{B}_{diff} \leftarrow \Big(\big(\overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_k^{\mathcal{C}}) - \overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_1^{\mathcal{C}})\big) \leftrightarrow \big(\overrightarrow{\mathbf{B}}(j, \hat{\mathbf{b}}_k^{\mathcal{C}}) - \overrightarrow{\mathbf{B}}(i, \hat{\mathbf{b}}_1^{\mathcal{C}})\big)\Big)$;
                  end
15                  $\mathsf{B}_{geq} \leftarrow \Big(\overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_1^{\mathcal{C}}) \geq \overrightarrow{\mathbf{B}}(i, \hat{\mathbf{b}}_1^{\mathcal{C}}) \ \wedge \overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_k^{\mathcal{C}}) \geq \overrightarrow{\mathbf{B}}(j, \hat{\mathbf{b}}_k^{\mathcal{C}})\Big)$;
16                  $\mathsf{B}_{diffG} \leftarrow \Big(\mathsf{B}_{diff} \ \wedge \ \mathsf{B}_{geq}\Big)$;
17                  $\mathsf{B}_{ijTarg} \leftarrow \Big(\mathbf{B}(i, \acute{\mathbf{b}}_1^{\mathcal{C}}) \wedge \mathbf{B}(j, \acute{\mathbf{b}}_k^{\mathcal{C}})\Big)$;
18                  $\mathsf{B}_{\text{frwd}} \leftarrow \Big(\mathsf{B}_{\text{frwd}} \ \vee (\mathsf{B}_{ijTarg} \wedge \mathsf{B}_{diffG})\Big)$;
              end
          end
19          $\mathsf{B}_{\text{frwdEvol}} \leftarrow (\mathsf{B}_{\text{frwdEvol}} \wedge \mathsf{B}_{\text{frwd}})$;
      end
20      for $k \leftarrow 1$ to $|\mathcal{C}|$ do
21          $\mathsf{B}_1 \leftarrow \Big(\mathsf{B}_{\text{frwdEvol}} \wedge (\overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_k^{\mathcal{C}}) \geq \overrightarrow{\mathbf{B}}(|D_k^{\mathcal{C}}|, \hat{\mathbf{b}}_k^{\mathcal{C}})) \wedge \mathbf{B}(|D_k^{\mathcal{C}}|, \check{\mathbf{b}}_k^{\mathcal{C}})\Big)$;
22          $\mathsf{B}_1 \leftarrow replace\big((\exists \hat{\mathbf{b}}^{\mathcal{C}} : \mathsf{B}_1)\ ,\ (\check{\mathbf{b}}^{\mathcal{C}}, \hat{\mathbf{b}}^{\mathcal{C}})\big)$
23          $\mathsf{B}_2 \leftarrow \Big(\mathsf{B}_{\text{frwdEvol}} \wedge (\overrightarrow{\mathbf{B}}(\cdot, \hat{\mathbf{b}}_k^{\mathcal{C}}) < \overrightarrow{\mathbf{B}}(|D_k^{\mathcal{C}}|, \hat{\mathbf{b}}_k^{\mathcal{C}}))\Big)$;
24          $\mathsf{B}_{\text{frwdEvol}} \leftarrow \mathsf{B}_1 \vee \mathsf{B}_2$;
        end
    end
```

Algorithm 1: BDD construction of time evolution.

the end of the iterations, B_{frwd} will represent (27). Since the loop in line 8 only iterates once, $B_{frwdEvol} = B(1)$ when line 19 is reached, which means that $B_{frwdEvol} = B_{frwd}$. Hence, $B_{frwdEvol}$ represents (27) and the basic step is proved.
Now for the inductive step, let assume that $B_{frwdEvol}$ represents (27) for the $k = \kappa$ first clocks, denoted by

$$\{\mu^{\mathcal{C}}_{1,\dots,\kappa} \rightsquigarrow (\mu^{\mathcal{C}}_{1,\dots,\kappa}, \mu^{\mathcal{C}}_{1,\dots,\kappa}+\mathbf{1}, \dots, \mu^{\mathcal{C}}_{1,\dots,\kappa}+\omega) \mid \mu^{\mathcal{C}}_{1,\dots,\kappa} \in D^{\mathcal{C}}_{1,\dots,\kappa} \ : \ \omega = \min_{p=1}^{\kappa}(2 \cdot |D^{\mathcal{C}}_p| - \mu^{\mathcal{C}}_p)\}, \quad (28)$$

where $\mu^{\mathcal{C}}_{1,\dots,\kappa}$ is a κ-tuple and $D^{\mathcal{C}}_{1,\dots,\kappa}$ is the domain for the κ first clocks. We prove that this also holds for $k = \kappa + 1$. In iteration $\kappa + 1$, B_{frwd} represents the time evolution between clock 1 and clock $\kappa + 1$. Based on (26), the BDD represents

$$\{\mu^{\mathcal{C}}_{1,\kappa+1} \rightsquigarrow (\mu^{\mathcal{C}}_{1,\kappa+1}, \mu^{\mathcal{C}}_{1,\kappa+1}+\mathbf{1}, \dots, \mu^{\mathcal{C}}_{1,\kappa+1}+\omega') \mid \mu^{\mathcal{C}}_{1,\kappa+1} \in D^{\mathcal{C}}_{1,\kappa+1} \ : \ \omega' = \min(2 \cdot |D^{\mathcal{C}}_1| - \mu^{\mathcal{C}}_1 \ , \ 2 \cdot |D^{\mathcal{C}}_{\kappa+1}| - \mu^{\mathcal{C}}_{\kappa+1})\}. \quad (29)$$

Now, lets compute $B_{frwdEvol}$ in line 19, which is obtained by conjuncting the corresponding BDDs for (28) and (29). We perform the conjunction on their corresponding characteristic functions. From (23), we have that the corresponding characteristic function for (28) is,

$$\bigvee_{\mu^{\mathcal{C}}_{1,\dots,\kappa} \in D^{\mathcal{C}}_{1,\dots,\kappa}} \Big(\acute{\mathbf{b}}^{\mathcal{C}}_1 \leftrightarrow \theta(\mu^{\mathcal{C}}_1) \wedge \bigwedge_{i=2}^{\kappa} \acute{\mathbf{b}}^{\mathcal{C}}_i \leftrightarrow \theta(\mu^{\mathcal{C}}_i) \ \wedge \bigvee_{d=0}^{\omega} \big(\hat{\mathbf{b}}^{\mathcal{C}}_1 \leftrightarrow \theta(\mu^{\mathcal{C}}_1 + d) \wedge \bigwedge_{j=2}^{\kappa} \hat{\mathbf{b}}^{\mathcal{C}}_j \leftrightarrow \theta(\mu^{\mathcal{C}}_j + d) \big) \Big). \quad (30)$$

and for (29), we have

$$\bigvee_{\mu^{\mathcal{C}}_{1,\kappa+1} \in D^{\mathcal{C}}_{1,\kappa+1}} \Big((\acute{\mathbf{b}}^{\mathcal{C}}_1 \leftrightarrow \theta(\mu^{\mathcal{C}}_1)) \ \wedge \ (\acute{\mathbf{b}}^{\mathcal{C}}_{\kappa+1} \leftrightarrow \theta(\mu^{\mathcal{C}}_{\kappa+1})) \ \wedge \bigvee_{d=0}^{\omega'} \big((\hat{\mathbf{b}}^{\mathcal{C}}_1 \leftrightarrow \theta(\mu^{\mathcal{C}}_1 + d)) \ \wedge \ (\hat{\mathbf{b}}^{\mathcal{C}}_{\kappa+1} \leftrightarrow \theta(\mu^{\mathcal{C}}_{\kappa+1} + d)) \big) \Big). \quad (31)$$

By conjuncting (30) and (31) we get

$$\bigvee_{\mu^{\mathcal{C}}_{1,\dots,\kappa+1} \in D^{\mathcal{C}}_{1,\dots,\kappa+1}} \Big(\acute{\mathbf{b}}^{\mathcal{C}}_1 \leftrightarrow \theta(\mu^{\mathcal{C}}_1) \wedge \bigwedge_{i=2}^{\kappa+1} \acute{\mathbf{b}}^{\mathcal{C}}_i \leftrightarrow \theta(\mu^{\mathcal{C}}_i) \ \wedge \bigvee_{d=0}^{\min(\omega,\omega')} \big(\hat{\mathbf{b}}^{\mathcal{C}}_1 \leftrightarrow \theta(\mu^{\mathcal{C}}_1 + d) \wedge \bigwedge_{j=2}^{\kappa+1} \hat{\mathbf{b}}^{\mathcal{C}}_j \leftrightarrow \theta(\mu^{\mathcal{C}}_j + d) \big) \Big),$$

which represents

$$\{\mu^{\mathcal{C}}_{1,\ldots,\kappa+1} \rightsquigarrow (\mu^{\mathcal{C}}_{1,\ldots,\kappa+1}, \mu^{\mathcal{C}}_{1,\ldots,\kappa+1} + \mathbf{1}, \ldots, \mu^{\mathcal{C}}_{1,\ldots,\kappa+1} + \omega) \mid \\ \mu^{\mathcal{C}}_{1,\ldots,\kappa+1} \in D^{\mathcal{C}}_{1,\ldots,\kappa+1} \ : \ \omega = \min_{p=1}^{\kappa+1}(2 \cdot |D^{\mathcal{C}}_p| - \mu^{\mathcal{C}}_p)\},$$

and thus the inductive step is proved.
Finally, in lines 20-24 of the algorithm (as stated earlier), for each clock c_k in $B_{frwdEvol}$ the values that are larger than $|D^{\mathcal{C}}_k$ will be replaced by $|D^{\mathcal{C}}_k$, which will yield a BDD representing (23), i.e., $\chi_{\text{timeEvolution}}$. The correctness of these lines is straightforward. Hence, the correctness of the entire algorithm is proved. □

Proposition 3. *The time complexity of Algorithm 1 is $O(|\mathcal{C}| \cdot |D^{\mathcal{C}}_{\cup}|^2 \cdot K)$, where K is the time complexity of performing the BDD operations in the loops, which is proportional to the sizes of the BDDs.*

Proof. The algorithm consists of three sequential parts, lines 2-7, lines 8-19 and lines 20-24. Since the time complexity of lines 8-19 is larger than the other two parts, it can be deduced that the time complexity of the entire algorithm is equal to the time complexity of lines 8-19, which is equal to $O(|\mathcal{C}| \cdot |D^{\mathcal{C}}_{\cup}|^2 \cdot K)$. □

The following theorem relates to step 4 of constructing $\mathbf{B}_{\rightarrowtail_{S_0}}$ that was described earlier.

Theorem 1. *The BDD representing $\rightarrowtail_{S_0}$ is constructed as follows:*

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

Proof. Based on the proof of correctness for $\mathbf{B}_{\mapsto_{S_0}}$ in [49] and Lemma 2, the correctness of the theorem can be directly deduced. □

The BDD for $\leftarrowtail_{S_0}$ can be computed in an analogous manner.

7 Case Study: A Production Cell

In this section, the symbolic approach discussed from Section 6 is applied to a benchmark example: the production cell example in [51]. 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 [52] based on the *State Tree Structure* (STS) methodology and then extended to the timed STS in [31].

The production cell, shown in Fig. 2, 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 [52]. The main

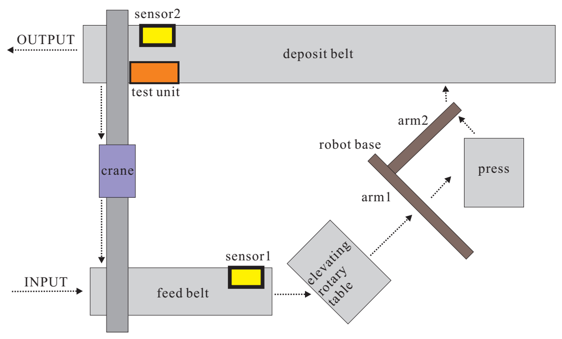

Figure 2: The Production Cell

object is to prevent collisions among certain parts at the same time guarantee nonblocking.

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 or overflow states. Since there is no time constraint involved in those specifications, we forgo the discussion about them and let readers refer to [52].

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. 3 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. 4 shows the corresponding "tick" model. From the modeling perspective, it can be observed that the TEFA model that embeds the time information as guards and actions to the transitions, gives a more compact and comprehensible representation.

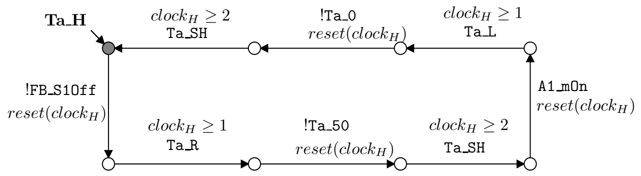

Figure 3: TEFA modeling the horizontal movement of the elevating rotary table. The description of the alphabet can be found in [52].

The symbolic approach proposed in this paper has been implemented and integrated in the supervisory control tool *Supremica* [10] which uses *JavaBDD* [53] as the BDD package. The experiment is carried out on a standard personal

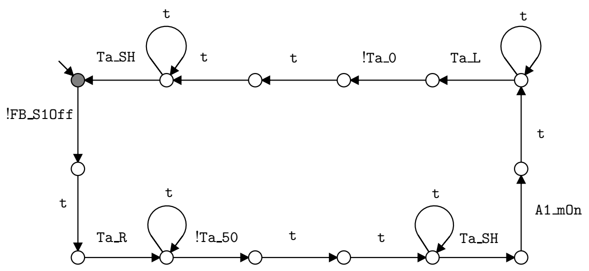

Figure 4: The corresponding "tick" model of Ta_H.

computer (Intel Dual Core CPU @ 1.83GHz and 1.5GB RAM) running Ubuntu 10.10. The result is shown in Table 1.

Table 1: Non-blocking Supervisory Synthesis

State-space	Reachable States	Supervisor	Iterations	BDD nodes	Computation Time
10^{12}	3.6×10^7	3.7×10^6	100	16102	2.7sec

8 Conclusions and Future Works

In this paper, we presented a method to efficiently compute a minimally restrictive nonblocking supervisor for a timed discrete event system. The systems are modeled by timed extended finite automata, ordinary automata extended with variables and clocks, and the supervisor synthesis is performed on their corresponding state transition models called timed transition systems. In order to be able to handle large systems, we perform all computations symbolically on the TTSs' corresponding binary decision diagrams. We also proved the correctness the entire procedure. As a case study, we applied our method to a classical production cell and computed its nonblocking supervisor

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] L. Feng, W. Wonham, and P. S. Thiagarajan, "Designing communicating transaction processes by supervisory control theory," *Form. Methods Syst. Des.*, vol. 30, no. 2, pp. 117–141, 2007.

[2] K. Andersson, J. Richardsson, B. Lennartson, and M. Fabian, "Coordination of Operations by Relation Extraction for Manufacturing Cell Controllers," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 414–429, Mar. 2010.

[3] P. Ramadge and W. Wonham, "The control of discrete event systems," *Proceedings of the IEEE, Special Issue on Discrete Event Dynamic Systems*, vol. 77, no. 1, pp. 81–98, 1989.

[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. Miremadi, K. Åkesson, and B. Lennartson, "Supervisor Computation and Representation: A Case Study," 2010.

[6] B. Lennartson, K. Bengtsson, C. Yuan, K. Andersson, M. Fabian, P. Falkman, and K. Åkesson, "Sequence planning for integrated product, process and automation design," *Automation Science and Engineering, IEEE Transactions on*, vol. 7, no. 4, pp. 791–802, Oct. 2010.

[7] 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.

[8] M. R. Shoaei, B. Lennartson, and S. Miremadi, "Automatic generation of controllers for collision-free flexible manufacturing systems," in *6th IEEE International Conference on Automation Science and Engineering.* IEEE, Aug. 2010, pp. 368–373.

[9] 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.

[10] 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.

[11] K. Åkesson, M. Fabian, H. Flordal, and A. Vahidi, "Supremica—a Tool for Verification and Synthesis of Discrete Event Supervisors," in *11th Mediterranean Conference on Control and Automation*, Rhodos, Greece, 2003.

[12] 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.

[13] 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.

[14] S. Miremadi, B. Lennartson, and K. Åkesson, "BDD-based Supervisory Control on Extended Finite Automata," in *Proceedings of the 7th Annual IEEE Conference on Automation Science and Engineering, CASE'11*, Trieste, 2011.

[15] 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.

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

[17] R. Alur, C. Courcoubetis, T. Henzinger, and P. Ho, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems," in *Lecture Notes in Computer Science.* Springer Berlin / Heidelberg, 1993, vol. 736, pp. 209–229.

[18] Y. Brave and B. H. Krogh, "Maximally permissive policies for controlled time marked graphs," in *Proceedings of the 12th IFAC World Congress*, 1993, pp. 263–266.

[19] A. Sava and H. Alla, "A control synthesis approach for time discrete event systems," *Mathematics and Computers in Simulation*, vol. 70, no. 5-6, pp. 250–265, Feb. 2006.

[20] J. Komenda, S. Lahaye, and J.-L. Boimond, "Supervisory Control of (max,+) Automata: A Behavioral Approach," *Discrete Event Dynamic Systems*, vol. 19, no. 4, pp. 525–549, Sep. 2009.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] C. Golaszewski and P. Ramadge, "On the Control of Real-Time Discrete Event Systems," in *23rd Annual Conference on Information Sciences and Systems*, 1989, pp. 98–102.

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

[27] D. Cofer and V. Garg, "Supervisory control of real-time discrete-event systems using lattice theory," *IEEE Transactions on Automatic Control*, vol. 41, no. 2, pp. 199–209, 1996.

[28] T.-J. Ho, "Control of timed discrete-event systems - A framework," in *Proceedings of the 34th Annual Allerton Conference on Communication, Control and Computing*, Monticello, Illinois, 1996, pp. 796–805.

[29] 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.

[30] 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.

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

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

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

[34] S.-J. Park, K.-H. Cho, and J.-T. Lim, "Supervisory control of real-time discrete event systems under bounded time constraints," in *IEE Proceedings of Control Theory and Applications*, vol. 151, no. 3, 2004, pp. 347–352.

[35] A. Saadatpoor and W. Wonham, "Supervisor State Size Reduction for Timed Discrete-Event Systems," in *2007 American Control Conference*. IEEE, Jul. 2007, pp. 4280–4284.

[36] ——, "State based control of timed discrete event systems using binary decision diagrams," *Systems & Control Letters*, vol. 56, no. 1, pp. 62–74, Jan. 2007.

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

[38] D. Dill and J. Sifakis, "Automatic Verification Methods for Finite State Systems," *Lecture Notes in Computer Science*, vol. 407, pp. 197–212, 1990.

[39] 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 *Proceedings of the Fifth Annual IEEE Symposium on e Logic in Computer Science, 1990. LICS '90,*, Jun. 1990, pp. 428–439.

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

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

[42] R. E. Bryant, "Symbolic Boolean manipulation with ordered binary-decision diagrams," *ACM Comput. Surv.*, vol. 24, no. 3, pp. 293–318, 1992.

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

[44] B. Bollig and I. Wegener, "Improving the Variable Ordering of OBDDs Is NP-Complete," *IEEE Trans. Comput.*, vol. 45, no. 9, pp. 993–1002, 1996.

[45] C. A. R. Hoare, *Communicating sequential processes*, ser. Series in Computer Science. ACM, Aug. 1978, vol. 21, no. 8.

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

[47] W. 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.

[48] 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.

[49] 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. PP, no. 99, pp. 1–15, 2011.

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

[51] 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.

[52] C. Ma, “Nonblocking Supervisory Control of State Tree Structures,” Ph.D. dissertation, University of Toronto, 2005.

[53] “JavaBDD.” [Online]. Available: http://javabdd.sourceforge.net