Synchronous composition of finite interval automata

Lin, Liyong; Su, Rong; Brandin, Bertil A.; Ware, Simon; Zhu, Yuting; Sun, Yajuan

2019

Lin, L., Su, R., Brandin, B. A., Ware, S., Zhu, Y., & Sun, Y. (2019). Synchronous composition of finite interval automata. Proceedings of 2019 IEEE 15th International Conference on Control and Automation (ICCA), 578-583. doi:10.1109/ICCA.2019.8899529

<https://hdl.handle.net/10356/137885>

© 2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at:
<https://doi.org/10.1109/ICCA.2019.8899529>

Downloaded on 26 Apr 2023 23:05:36 SGT

Synchronous Composition of Finite Interval Automata

Liyong Lin, Rong Su, Bertil A. Brandin, Simon Ware, Yuting Zhu

Abstract—In this work, we study the class of real-time discrete-event systems modeled by finite interval automata, i.e., a finite state automaton where each transition is labeled by an event and an interval. The class of finite interval automata is interpreted over discrete time models and corresponds exactly to the class of finite tick automata. An interval domain construction of the synchronous product of finite interval automata is provided that exactly encodes the synchronous product in the tick domain. This avoids explicit enumeration of clock ticks that is required by the timed discrete-event systems and finite tick automata formalism. To increase the expressive power of the finite interval automata, we also distinguish preempting transitions from non-preempting transitions. As a result, each timed discrete-event system can be encoded using a finite interval automaton.

Index Terms – timed discrete-event systems, interval automata, tick automata, synchronous product

I. INTRODUCTION

Models for real-time discrete-event systems and their supervisory control problems have been proposed and investigated under various different formalisms such as clock automata [1], timed transition models [2], timed discrete-event systems [3] and timed petri nets [4], with discrete time semantics; timed automata [5], with dense time semantics. Compared with dense time models, discrete time models do offer the advantage of decidability for basic model checking and synthesis problems. In this paper, we study the class of real-time discrete-event systems that can be modeled by finite interval automata, i.e., finite state automata where each transition is labeled by an event and an interval, interpreted over discrete time models. Finite interval automata are used to encode finite tick automata in a compact manner. Instead of explicit enumeration of ticks, we use intervals I of natural numbers to specify behavior like “after reaching a state, an event can be fired after the elapse of x amount of ticks, for any $x \in I$ ” using a single transition. The interval labeling each transition can be viewed as a guard on the local clock which keeps on ticking and gets reset after each transition. This often results in a much more compact automaton representation than finite tick automata. To support model composition, i.e., building a global interval automaton out of component interval automata, we provide an arithmetic-based construction of the synchronous product of finite interval automata.

The clock automata formalism in its original formulation is too expressive, i.e., the partial transition function can almost be arbitrarily specified, and thus clock automata are in general not

amenable to finite state representation and computation. In this work, we shall consider a relatively expressive subclass, that is, the subclass of finite tick automata¹. Each finite tick automaton can be compactly represented by a finite interval automaton. Finite interval automata formalism has previously been studied in several works (see for example [6] and [7], e.t.c.), however, interpreted over dense time models. Over dense time models, it is known that the class of finite interval automata is not even closed under subsystem composition [8]. This is not the case for finite interval automata interpreted over discrete time models. In fact, to compute the synchronous product in the interval domain, presently one could first convert each finite interval automaton into an equivalent finite tick automaton and then compute the synchronous product in the tick domain, and finally convert the resulting finite tick automaton back to the interval domain. This approach, although straightforward, requires an explicit enumeration of ticks of the clock, which may require a huge amount of memory and time, depending on the partial transition function. It is thus beneficial to develop a synchronous product procedure directly in the interval domain which efficiently operates upon the data structure of intervals and avoids explicit enumeration of ticks in many cases. Since each interval automaton uses an implicit local timer to count the number of ticks that has elapsed at current state, subsystem composition entails collapsing two local timers into a single timer for the global description, which is not straightforward. Intervals are also used in the description of timed discrete-event systems. The activity automaton of each timed discrete-event system has each event assigned an interval based timer for tracking its delay and deadline. However, it is not straightforward to understand how each timed discrete-event system, described by the activity automaton and timers, would behave in the tick domain before it is explicitly constructed; moreover, since there is only a unique delay and deadline for each event, the timed discrete-event systems formalism is less expressive than the finite interval automata formalism. For a synchronous product based construction of a timed discrete-event system from its activity automaton and the timers, reader is referred to [10]. In the clock automata formalism [1], the tick event is defined at each state. Thus, subsystem composition, although not finitely computable in general, is well defined and will never produce temporal deadlock, i.e., stop the clock. This is an advantage over the timed discrete-event systems formalism², in which not every two timed discrete-event systems can be composed [3].

All the above mentioned difficulties would be addressed in the finite interval automata formalism developed in this work.

L. Lin, R. Su, S. Ware and Y. Zhu are with School of EEE, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798. B. Brandin is with Advanced Remanufacturing and Technology Centre (ARTC), 3 Clean-Tech Loop, Singapore 637143. Email: liyong.lin@ntu.edu.sg. This work is supported by the National Research Foundation of Singapore Delta- NTU Corporate Lab Program with the project reference of DELTA-NTU CORP-SMA-RP2

¹Strictly speaking, a finite tick automaton is not a clock automaton, since transitions are additionally classified into preempting and non-preempting.

²In reality, systems do compose!

The main contributions of this paper are the following. We introduce our model of finite interval automata and formalize its semantic interpretation in terms of finite tick automata, also to be defined in this work. The definition of finite tick automata is suitably restricted so that each finite tick automaton also has an equivalent finite interval automaton representation. Then we provide an interval domain construction of the synchronous product of finite interval automata, which encodes the synchronous product in the tick domain. To our best knowledge, the problem of computing the synchronous product of finite interval automata directly in the interval domain has not been solved previously.

II. PRELIMINARIES

We assume reader is familiar with basic theory of formal languages and finite automata [9]. Some additional notations and terminologies are introduced in the following.

Let A and B be two sets. The set-theoretic difference of A and B is denoted by $A - B$. For any finite set A , $|A|$ is used to denote its cardinality. For any partial function $f : X \mapsto Y$, we write $f(x)!$ to mean that f is defined on x , i.e., x is in the domain $\text{dom}(f)$ of f . We shall identify partial function f with the relation $f \subseteq X \times Y$. Let $\mathbb{N}$ denote the set $\{0, 1, 2, 3, \dots\}$ of natural numbers and let $\mathbb{N}^+$ denote the set $\{1, 2, 3, \dots\}$ of positive integers. We write $[k_1, k_2]$, where $k_1, k_2 \in \mathbb{N}$ and $k_1 \leq k_2$, to denote the set $\{k \in \mathbb{N} \mid k_1 \leq k \leq k_2\}$, and write $[k_1, \infty)$ to denote the set $\{k \in \mathbb{N} \mid k \geq k_1\}$. Let $\mathcal{I}$ denote the collection of intervals of the above two types and let $\mathcal{I}_f$ denote the collection of intervals of the first type. For any interval I , we define $I.l = k_1, I.r := k_2$ if $I = [k_1, k_2]$; $I.l := k_1, I.r := \infty$ if $I = [k_1, \infty)$. For any non-negative integer $c \in \mathbb{N}$, we define $I \dot{-} c := [\max\{I.l - c, 0\}, I.r - c]$ if $c \leq I.r < \infty$; $I \dot{-} c := [\max\{I.l - c, 0\}, \infty)$ if $I.r = \infty$; $I \dot{-} c := \emptyset$ if $I.r < c$. Given any equivalence relation $\equiv$ on set A , a subset $B \subseteq A$ is said to be $\equiv$ -closed if B is the union of some equivalence classes.

III. LINGUISTICS SETUP

Let Σ be a finite set of events. A finite string s over $(\Sigma \times \mathcal{I}) \cup (\Sigma^\# \times \mathcal{I}_f)$ is said to be an *interval word* over Σ , where $\Sigma^\# := \{\sigma^\# \mid \sigma \in \Sigma\}$ is a copy of Σ with superscript $\#$ and the number sign $\#$ is used to indicate preemption, which shall be discussed in detail later. Elements of $(\Sigma \times \mathcal{I}) \cup (\Sigma^\# \times \mathcal{I}_f)$ are called interval events over Σ . We also use the shorthand notation σI to denote the interval event (σ, I) . s is said to be an *atomic* interval word if it is a string over $(\Sigma \cup \Sigma^\#) \times \{[k, k] \mid k \in \mathbb{N}\}$. A set L of interval words over Σ is said to be an *interval language* over Σ . L is said to be *atomic* if every interval word in L is atomic. Two syntactically different interval languages may describe exactly the same object, i.e., the same atomic interval language. We need to bring in the notion of *atomic expansion* for comparing interval languages.

Definition 1 (Atomic Expansion). *The atomic expansion of an interval word over Σ is inductively defined as follows:*

- 1) $\mathcal{E}(\varepsilon) := \varepsilon$
- 2) for any event $\sigma \in \Sigma$, any interval $I \in \mathcal{I}$, $\mathcal{E}((\sigma, I)) := \{(\sigma, [k, k]) \mid k \in I\}$

- 3) for any event $\sigma \in \Sigma$, any interval $I \in \mathcal{I}_f$, $\mathcal{E}((\sigma^\#, I)) := \{(\sigma^\#, [I.r, I.r])\} \cup \{(\sigma, [k, k]) \mid I.l \leq k \leq I.r - 1\}$
- 4) for any interval word s over Σ , any event $\sigma \in \Sigma$, any interval $I \in \mathcal{I}$, $\mathcal{E}(s(\sigma, I)) := \mathcal{E}(s)\mathcal{E}((\sigma, I))$
- 5) for any interval word s over Σ , any event $\sigma \in \Sigma$, any interval $I \in \mathcal{I}_f$, $\mathcal{E}(s(\sigma^\#, I)) := \mathcal{E}(s)\mathcal{E}((\sigma^\#, I))$

The atomic expansion of an interval language L is defined to be $\bigcup_{s \in L} \mathcal{E}(s)$ and denoted by $\mathcal{E}(L)$, which is an atomic interval language. Two interval languages L_1, L_2 are said to be equal if $\mathcal{E}(L_1) = \mathcal{E}(L_2)$. Similarly, L_1 is said to be a subset of L_2 if $\mathcal{E}(L_1) \subseteq \mathcal{E}(L_2)$. In the rest, we shall write $L_1 = L_2$, $L_1 \subseteq L_2$ instead for simplicity. Intuitively (and roughly speaking), an interval word $s = (\sigma_1, I_1)(\sigma_2, I_2) \dots (\sigma_n, I_n)$ stands for executions of events $\sigma_1, \sigma_2, \dots, \sigma_n$, where the execution of event σ_{i+1} is within interval I_{i+1} after the execution of event σ_i and the execution of event σ_1 is within interval I_1 since initialization.

Let $\equiv_t$ be an equivalence relation on $(\Sigma \cup \Sigma^\# \cup \{t\})^*$, where t denotes the special tick event, defined such that $s \equiv_t s'$ if and only if $\exists k \in \mathbb{N}, s' = st^k \vee s = s't^k$. Let $\pi : (\Sigma \cup \Sigma^\# \cup \{t\})^* \mapsto (\Sigma \cup \Sigma^\# \cup \{t\})^* / \equiv_t$ be the canonical projection that maps any string $s \in (\Sigma \cup \Sigma^\# \cup \{t\})^*$ to its equivalence class $\pi(s)$. In the rest, we shall also abuse the notation and use $\pi(s)$ to denote the unique shortest string $s' \in (\Sigma \cup \Sigma^\# \cup \{t\})^*$ that is $\equiv_t$ -equivalent to s . For any language $L \subseteq (\Sigma \cup \Sigma^\# \cup \{t\})^*$, we define $\pi(L) := \{\pi(s) \mid s \in L\}$. We now bring in the notion of *tick interpretation* of interval words.

Definition 2 (Tick Interpretation). *The tick interpretation of an interval word over Σ is inductively defined as follows:*

- 1) $\mathcal{T}(\varepsilon) := t^*$
- 2) for any $\sigma \in \Sigma$, any $I \in \mathcal{I}$, $\mathcal{T}((\sigma, I)) := \bigcup_{k \in I} t^k \sigma t^k$
- 3) for any $\sigma \in \Sigma$, any $I \in \mathcal{I}_f$,

$$\mathcal{T}((\sigma^\#, I)) := \bigcup_{I.l \leq k \leq I.r-1} t^k \sigma t^* \cup t^{I.r} \sigma^\# t^*$$

- 4) for any interval word s over Σ , any $\sigma \in \Sigma$, any $I \in \mathcal{I}$, $\mathcal{T}(s(\sigma, I)) := \pi(\mathcal{T}(s))\mathcal{T}((\sigma, I))$
- 5) for any interval word s over Σ , any $\sigma \in \Sigma$, any $I \in \mathcal{I}_f$, $\mathcal{T}(s(\sigma^\#, I)) := \pi(\mathcal{T}(s))\mathcal{T}((\sigma^\#, I))$

Then, the tick interpretation $\mathcal{T}(L)$ of an interval language L over Σ is defined to be $\bigcup_{s \in L} \mathcal{T}(s)$, which is clearly also equal to $\bigcup_{s \in \mathcal{E}(L)} \mathcal{T}(s)$. The tick interpretation of an interval language over Σ is said to be a *tick language* (over Σ), which is essentially an $\equiv_t$ -closed language over $\Sigma \cup \Sigma^\# \cup \{t\}$. The inverse of the tick interpretation operation is denoted by $\mathcal{I}$, called the *interval interpretation*, which is defined on the set of tick languages over Σ in the following way:

- 1) for any $k_1, \dots, k_n \in \mathbb{N}$ and any $\alpha_1, \dots, \alpha_n \in \Sigma \cup \Sigma^\#$, $I(t^{k_1} \alpha_1 t^{k_2} \alpha_2 \dots t^{k_n} \alpha_n t^*) := \alpha_1[k_1, k_1] \alpha_2[k_2, k_2] \dots \alpha_n[k_n, k_n]$
- 2) the definition is linearly extended to the domain of all tick languages over Σ .

Proposition 1. *The interval interpretation operation $\mathcal{I}$ and the tick interpretation operation $\mathcal{T}$ satisfy the property that for any interval language L over Σ , $\mathcal{I}(\mathcal{T}(L)) = L$ and for*

any tick language L over Σ , $\mathcal{T}(\mathcal{I}(L)) = L$. That is, they are inverse to each other.

IV. FINITE INTERVAL AUTOMATA

We first introduce the definition of *finite interval automata* and then discuss some relevant notions.

A. Main Definition and Notions

Definition 3. A *finite interval automaton* over Σ is a 7-tuple $\mathcal{G}_{\mathcal{I}} = (Q, \Sigma, \delta_{\mathcal{I}}, q_0, Q_m, \mathcal{I}, p_{\mathcal{I}})$, where Q is the finite set of states, $q_0 \in Q$ is the initial state, $Q_m \subseteq Q$ is the subset of marked states, $\delta_{\mathcal{I}} : Q \times \Sigma \times \mathcal{I} \mapsto Q$ is the partial transition function and $p_{\mathcal{I}} : Q \times \Sigma \times \mathcal{I} \mapsto \{0, 1\}$ is the partial map with $\text{dom}(p_{\mathcal{I}}) = \text{dom}(\delta_{\mathcal{I}})$, such that

- 1) for any state $q \in Q$, $\{(\sigma, I) \in \Sigma \times \mathcal{I} \mid \delta_{\mathcal{I}}(q, \sigma, I)!\}$ is a finite set
- 2) for any state $q \in Q$, any event $\sigma \in \Sigma$ and any two intervals $I, I' \in \mathcal{I}$, if $\delta_{\mathcal{I}}(q, \sigma, I)!$ and $\delta_{\mathcal{I}}(q, \sigma, I')!$, then either $I \cap I' = \emptyset$ or $I = I' \wedge \delta_{\mathcal{I}}(q, \sigma, I) = \delta_{\mathcal{I}}(q, \sigma, I')$
- 3) for any state $q \in Q$, any event $\sigma \in \Sigma$ and any interval $I \in \mathcal{I}$, if $p_{\mathcal{I}}(q, \sigma, I) = 1$, then $I.r \neq \infty$
- 4) for any marked state $q \in Q_m$, any event $\sigma \in \Sigma$ and any interval $I \in \mathcal{I}$, if $\delta_{\mathcal{I}}(q, \sigma, I)!$, then $p_{\mathcal{I}}(q, \sigma, I) = 0$

Finite interval automata will be used in this work to model real-time discrete-event systems. The physical interpretation is given as follows: each finite interval automaton $\mathcal{G}_{\mathcal{I}}$ is associated with a local digital timer, which is always synchronized with an external global digital clock that keeps on ticking. The timer records at each current state $q \in Q$ the number $c \in \mathbb{N}$ of ticks of the clock that have elapsed since q is last entered. When c falls within some interval $I \in \mathcal{I}$ where $\delta_{\mathcal{I}}(q, \sigma, I)!$, $\mathcal{G}_{\mathcal{I}}$ can fire event σ and enter state $q' := \delta_{\mathcal{I}}(q, \sigma, I)$. Once each new state is entered, the timer value is immediately reset to zero and then keeps on increasing again. The partial map $p_{\mathcal{I}}$ classifies transitions into preempting and non-preempting. The valuation $p_{\mathcal{I}}(q, \sigma, I) = 1$ (respectively, $p_{\mathcal{I}}(q, \sigma, I) = 0$) means that at state q event σ will preempt tick of the clock once the timer value reaches $I.r$ (respectively, does not preempt tick whenever c is within I); and, correspondingly, the transition $\delta_{\mathcal{I}}(q, \sigma, I) = q'$ is said to be preempting (respectively, non-preempting). If the valuation of $p_{\mathcal{I}}$ is zero for all the transitions defined at state q , then $\mathcal{G}_{\mathcal{I}}$ may keep staying at state q , while the timer value keeps on increasing. In above definition, $\Sigma \times \mathcal{I}$ is an infinite set. Thus, Condition 1) is imposed here to ensure the finiteness of the interval automaton. We impose Condition 2) to ensure that $\mathcal{G}_{\mathcal{I}}$ is deterministic and Condition 3) ensures that $I.r$ is finite when $p_{\mathcal{I}}(q, \sigma, I) = 1$, which is a rather natural condition in view of the above interpretation of the valuation $p_{\mathcal{I}}(q, \sigma, I) = 1$. Finally, Condition 4) is imposed to ensure that the marked behavior of $\mathcal{G}_{\mathcal{I}}$ on the “executorial level” is an interval language. Indeed, the valuation $p_{\mathcal{I}}(q, \sigma, I) = 1$ alone, if q is a marked state, means that staying at state q completes the task as long as $c \leq I.r$ and $\mathcal{G}_{\mathcal{I}}$ will leave state q before the timer reaches $I.r + 1$ due to the preemption, implying staying at state q does not complete the task whenever $c \geq I.r + 1$. This is undesirable, since a completed task that could be made

Fig. 1: Endangered pedestrian: interval automata

to become uncompleted by firing ticks shall be considered to be uncompleted. It is often more convenient to directly encode the map $p_{\mathcal{I}}$ into the partial transition function. Indeed, a finite interval automaton over Σ can also be represented using a 6-tuple $\mathcal{G}_{\mathcal{I}} = (Q, \Sigma, \delta_{\mathcal{I}}^{\#}, q_0, Q_m, \mathcal{I})$, where Q, q_0, Q_m are the same as in Definition 3, and the new partial transition function $\delta_{\mathcal{I}}^{\#} : Q \times (\Sigma \cup \Sigma^{\#}) \times \mathcal{I} \mapsto Q$ is used here to encode $\delta_{\mathcal{I}}$ and $p_{\mathcal{I}}$: we encode $\delta_{\mathcal{I}}(q, \sigma, I) = q' \wedge p_{\mathcal{I}}(q, \sigma, I) = 0$ with transition $\delta_{\mathcal{I}}^{\#}(q, \sigma, I) = q'$ and encode $\delta_{\mathcal{I}}(q, \sigma, I) = q' \wedge p_{\mathcal{I}}(q, \sigma, I) = 1$ with transition $\delta_{\mathcal{I}}^{\#}(q, \sigma^{\#}, I) = q'$. Intuitively, the number sign $\#$ is used to indicate the preemption. In the rest, we shall freely switch between either representation of $\mathcal{G}_{\mathcal{I}}$ for the purpose of convenience, whenever there is no potential risk of confusion. We often use $\delta_{\mathcal{I}}$ to emphasize the transition, when preemption is not important, and use $\delta_{\mathcal{I}}^{\#}$ to emphasize both the transition and the preemption.

Example 1. We now consider a modified endangered pedestrian example [3] shown in Fig. 1. The finite interval automaton on the right hand side models the pedestrian who makes a single jump across the road at any moment. The finite interval automaton on the left hand side models the bus. It either makes a single transition pass when the timer value becomes 1, or it can be stopped by a police officer when the timer value is 0 or 1 and wait for one tick, and then the process is repeated. We note that event s (“stop”) and p (“pass”) compete for the execution at the initial state when the timer value is 1, since preempting transitions are assumed to be weakly preemptive.

The marked (respectively, closed) behavior of $\mathcal{G}_{\mathcal{I}}$ is defined as usual and denoted by $L_m(\mathcal{G}_{\mathcal{I}})$ (respectively, $L(\mathcal{G}_{\mathcal{I}})$). We shall note that $\mathcal{G}_{\mathcal{I}}$ may be “non-executorial” and there could be some non-executable strings in the marked behavior due to the preemption. In order to study the marked behavior on the “executorial” level, we introduce the following notations. Let $K_{\mathcal{G}_{\mathcal{I}}} : Q \mapsto 2^{\mathcal{I}}$ be a map that assigns to each state $q \in Q$ a set $K_{\mathcal{G}_{\mathcal{I}}}(q) := \{I \in \mathcal{I} \mid \exists \sigma \in \Sigma, \delta_{\mathcal{I}}^{\#}(q, \sigma^{\#}, I)!\}$ of intervals. Let $M_{\mathcal{G}_{\mathcal{I}}} : Q \mapsto \mathbb{N} \cup \{\infty\}$ be a map that assigns to each state $q \in Q$ a number $M_{\mathcal{G}_{\mathcal{I}}}(q) := \min_{I \in K_{\mathcal{G}_{\mathcal{I}}}(q)} I.r$, if $K_{\mathcal{G}_{\mathcal{I}}}(q) \neq \emptyset$; $M_{\mathcal{G}_{\mathcal{I}}}(q) := \infty$, otherwise. Intuitively, $K_{\mathcal{G}_{\mathcal{I}}}(q)$ denotes the set of intervals labeling the preempting transitions defined at state q . $M_{\mathcal{G}_{\mathcal{I}}}(q)$ then denotes the smallest value of the timer when tick of the clock will be preempted at state q .

Given any interval automaton $\mathcal{G}_{\mathcal{I}} = (Q, \Sigma, \delta_{\mathcal{I}}^{\#}, q_0, Q_m, \mathcal{I})$, another interval automaton $\mathcal{G}_{\mathcal{I}}^{\mathcal{P}} := (Q, \Sigma, \delta_{\mathcal{I}}^{\# \cdot \mathcal{P}}, q_0, Q_m, \mathcal{I})$ is induced, where the induced partial transition function $\delta_{\mathcal{I}}^{\# \cdot \mathcal{P}}$ is defined as follows: for any state $q, q' \in Q$, any event $\sigma \in \Sigma$ and any interval $I \in \mathcal{I}$,

- 1) $\delta_{\mathcal{I}}^{\# \cdot \mathcal{P}}(q, \sigma, I \cap [0, M_{\mathcal{G}_{\mathcal{I}}}(q)]) := q'$ if $\delta_{\mathcal{I}}^{\#}(q, \sigma, I) = q'$
- 2) $\delta_{\mathcal{I}}^{\# \cdot \mathcal{P}}(q, \sigma, I \cap [0, M_{\mathcal{G}_{\mathcal{I}}}(q)]) := q'$ if $\delta_{\mathcal{I}}^{\#}(q, \sigma^{\#}, I) = q' \wedge$

$$\begin{aligned}
& I.r > M_{\mathcal{G}_{\mathcal{I}}}(q) \\
3) \quad & \delta_{\mathcal{I}}^{\#,P}(q, \sigma^{\#}, I) := q' \text{ if} \\
& \delta_{\mathcal{I}}^{\#}(q, \sigma^{\#}, I) = q' \wedge I.r = M_{\mathcal{G}_{\mathcal{I}}}(q)
\end{aligned}$$

Intuitively, $\mathcal{G}_{\mathcal{I}}^P$ is obtained from $\mathcal{G}_{\mathcal{I}}$ by applying the effect of preemption at each state: each interval I labeling a transition defined at state q is replaced by $I \cap [0, M_{\mathcal{G}_{\mathcal{I}}}(q)]$; in addition, if the least upper bound, i.e., $I.r$, of the interval I labeling a preempting transition defined at state q is larger than $M_{\mathcal{G}_{\mathcal{I}}}(q)$, then that preempting transition becomes non-preempting, i.e., the event label $\sigma^{\#}$ is replaced by σ . The marked behavior of $\mathcal{G}_{\mathcal{I}}$ on the executional level is captured by the marked behavior $L_m(\mathcal{G}_{\mathcal{I}}^P)$ of $\mathcal{G}_{\mathcal{I}}^P$. We also call $L_m(\mathcal{G}_{\mathcal{I}}^P)$ the marked behavior of $\mathcal{G}_{\mathcal{I}}$ under preemption. A finite interval automaton $\mathcal{G}_{\mathcal{I}}$ is said to be *in the preempted form* if $\mathcal{G}_{\mathcal{I}} = \mathcal{G}_{\mathcal{I}}^P$. We shall remark here that transitions of the form $\delta_{\mathcal{I}}(q, \sigma, \emptyset) = q'$ are considered to be undefined and thus are removed from the graphical display. According to the definition of $\mathcal{G}_{\mathcal{I}}^P$, a state that is reachable in $\mathcal{G}_{\mathcal{I}}$ may become unreachable in $\mathcal{G}_{\mathcal{I}}^P$, since $I \cap [0, M_{\mathcal{G}_{\mathcal{I}}}(q)]$ may become an empty set and the corresponding transition will be removed.

Given any interval automaton $\mathcal{G}_{\mathcal{I}}$, a state $q \in Q$ is said to be *non-critical* if $\exists \sigma \in \Sigma, \exists I \in \mathcal{I}, \delta_{\mathcal{I}}^{\#}(q, \sigma^{\#}, I)!$ or $\exists \sigma \in \Sigma, \exists I \in \mathcal{I}, (\delta_{\mathcal{I}}^{\#}(q, \sigma, I)!) \wedge I.r = \infty$; otherwise, q is said to be *critical*. By definition, a state $q \in Q$ is critical in $\mathcal{G}_{\mathcal{I}}$ if and only if it is critical in $\mathcal{G}_{\mathcal{I}}^P$. $\mathcal{G}_{\mathcal{I}}$ is said to be *non-blocking under preemption* if $\mathcal{G}_{\mathcal{I}}^P$ satisfies the following two conditions: 1) each reachable state in $\mathcal{G}_{\mathcal{I}}^P$ is coreachable; 2) each reachable critical state in $\mathcal{G}_{\mathcal{I}}^P$ is marked. It is worth noting that each critical, non-marked state is considered to be a blocking state since, by definition, the interval automaton may stay at a critical state forever, while the timer value keeps on increasing.

Example 2. For the modified endangered pedestrian example shown in Fig. 1, the two finite interval automata are both in the preempted form. They are also non-blocking under preemption. Indeed, the unique critical state of each interval automaton is the only state that does not have any outgoing transition, which happens to be marked. If the interval event $s[0, 1]$ is replaced by $s[0, \infty)$, then the interval automaton on the left-hand side is not in the preempted form and there will be non-executable strings, e.g., $s[2, 2]w^{\#}[1, 1]p^{\#}[1, 1]$.

It is crucially important to note that although $\mathcal{G}_{\mathcal{I}}^P$ represents the behavior of $\mathcal{G}_{\mathcal{I}}$ on the executional level, only $\mathcal{G}_{\mathcal{I}}$ contains the complete details which is necessary for the synchronous composition with other automata. This is because preempting transitions in $\mathcal{G}_{\mathcal{I}}$ could be disabled by other components in the synchronous interaction and thus may not always take effect. In comparison, automaton $\mathcal{G}_{\mathcal{I}}^P$ is the result of applying preemption on $\mathcal{G}_{\mathcal{I}}$, with the implicit assumption that preempting transitions always take effect. This makes $\mathcal{G}_{\mathcal{I}}^P$ an incomplete model for synchronous composition. However, automata $\mathcal{G}_{\mathcal{I}}^P$ and $\mathcal{G}_{\mathcal{I}}$ become indistinguishable in the synchronous composition as far as behavior on the executional level is concerned, when events labeling preempting transitions in $\mathcal{G}_{\mathcal{I}}$ will never be disabled by other components. A formal definition of the synchronous product operation in the interval domain shall be presented later in Section V. We shall remark that when $\mathcal{G}_{\mathcal{I}}$ is a

Fig. 2: Endangered pedestrian: tick automata

closed-system, i.e., there is no other automaton in the system, our object of study will be $\mathcal{G}_{\mathcal{I}}^P$ instead of $\mathcal{G}_{\mathcal{I}}$.

B. Semantic Interpretation

In this subsection, we shall provide a semantic interpretation of finite interval automata in terms of finite tick automata, and vice versa. To that end, we first introduce our notion of *finite tick automata* and the relevant notions.

1) Finite Tick Automata:

Definition 4. A finite tick automaton over Σ is a 6-tuple $\mathcal{G} = (Q, \Sigma \cup \{t\}, \delta, q_0, Q_m, p)$, where Q is the finite set of states, $t \notin \Sigma$ is the special tick event, $q_0 \in Q$ is the initial state, $Q_m \subseteq Q$ is the set of marked states, $\delta : Q \times (\Sigma \cup \{t\}) \mapsto Q$ is the partial transition function and $p : Q \times \Sigma \mapsto \{0, 1\}$ is the partial map with $\text{dom}(p) = \text{dom}(\delta) \cap (Q \times \Sigma)$, such that

- 1) for any state $q \in Q$, $\delta(q, t)!$
- 2) for any string $s \in (\Sigma \cup \{t\})^*$, if $\delta(q_0, s) \in Q_m$, then $\delta(q_0, s') \in Q_m$ for any other string $s' \in (\Sigma \cup \{t\})^*$ that is $\equiv_t$ -equivalent to s
- 3) for any state $q \in Q$, if $\delta(q, t) = q$, then for any $\sigma \in \Sigma$, if $\delta(q, \sigma)!$, then $p(q, \sigma) = 0$
- 4) for any marked state $q \in Q_m$ and any event $\sigma \in \Sigma$, if $\delta(q, \sigma)!$, then $p(q, \sigma) = 0$
- 5) for any state $q \in Q$, if $\delta(q, t) \neq q$, then $q \notin \delta(q, t^+)$

In Definition 4 introduced above, Condition 1) is imposed to avoid potential temporal deadlock situation when two or more tick automata are synchronized. Conditions 2), 3), 4) and 5) are imposed to ensure that there is a one to one correspondence between finite interval automata and finite tick automata. In particular, Conditions 3), 4) above correspond to Conditions 3), 4) of Definition 3. It is often convenient to view a finite tick automaton over Σ as a 5-tuple $\mathcal{G} = (Q, \Sigma \cup \{t\}, \delta^{\#}, q_0, Q_m)$, where the new partial transition function $\delta^{\#} : Q \times (\Sigma \cup \Sigma^{\#} \cup \{t\}) \mapsto Q$ is used here to encode δ and p : we encode $\delta(q, \sigma) = q' \wedge p(q, \sigma) = 0$ with transition $\delta^{\#}(q, \sigma) = q'$ and encode $\delta(q, \sigma) = q' \wedge p(q, \sigma) = 1$ with transition $\delta^{\#}(q, \sigma^{\#}) = q'$; and $\delta^{\#}(q, t) := \delta(q, t)$.

Example 3. The corresponding tick automaton models of the modified endangered pedestrian example are shown in Fig. 2, which provides an intuitive explanation of the semantics of the interval automaton models shown in Fig. 1. The tick automata models could also be automatically computed from the interval automata models through the tick interpretation construction introduced later. We remark that the tick automaton model of

the bus is not yet minimal, which is not unusual for the tick interpretation construction.

Let $\mathcal{G} = (Q, \Sigma \cup \{t\}, \delta^\#, q_0, Q_m)$ be any finite tick automaton over Σ . The marked (respectively, closed) behavior $L_m(\mathcal{G})$ (respectively, $L(\mathcal{G})$) of $\mathcal{G}$ is defined as usual. In order to study the marked behavior of finite tick automata on the executional level, we introduce the following notation. For any two states $q, q' \in Q$ of the same tick automaton $\mathcal{G}$, we define $q <_\exists q'$ if $\exists k \in \mathbb{N}^+, \delta^\#(q, t^k) = q'$. Another tick automaton $\mathcal{G}^P$ can be obtained from $\mathcal{G}$ by applying the effect of preemption, i.e., removing the non-tick transitions defined at each state $q' \in Q$ that satisfies the condition $\exists q <_\exists q', \exists \sigma \in \Sigma, \delta^\#(q, \sigma^\#)!$. The marked behavior $L_m(\mathcal{G}^P)$ of $\mathcal{G}^P$ is called the marked behavior of $\mathcal{G}$ under preemption, which is the marked behavior of $\mathcal{G}$ on the executional level. $\mathcal{G}$ is said to be *non-blocking under preemption* if the finite state automaton $\mathcal{G}_f$ obtained from $\mathcal{G}$ by removing the tick transition defined at each state $q \in Q$ that satisfies the condition $\exists \sigma \in \Sigma, \delta^\#(q, \sigma^\#)!$ is a non-blocking finite state automaton. $\mathcal{G}$ is said to be *in the preempted form* if $\mathcal{G} = \mathcal{G}^P$ holds. By our definition of finite tick automata, we have $L_m(\mathcal{G}_f) = L_m(\mathcal{G}^P)$. Thus, in the rest we shall mainly work with finite tick automaton $\mathcal{G}^P$ explicitly, due to the fact that $\mathcal{G}_f$ is in general not a finite tick automaton. However, one can easily obtain $\mathcal{G}_f$ from $\mathcal{G}^P$ and vice versa.

2) *Tick Interpretation*: In order to formally present tick interpretation operation, we need to introduce the following extra notations. Let $R : \mathcal{I} \mapsto \mathbb{N}$ be a map that assigns to each interval $I \in \mathcal{I}$ a number $R(I) := I.l$, if $I.r = \infty$; $R(I) := I.r + 1$, if $I.r < \infty$. Let $S : 2^\mathcal{I} \mapsto \mathbb{N}$ be a partial map that assigns to each finite set $F \subseteq \mathcal{I}$ of intervals a number $S(F) := 0$, if $F = \emptyset$; $S(F) := \max\{R(I) \mid I \in F\}$, otherwise. For any given finite interval automaton $\mathcal{G}_\mathcal{I} = (Q, \Sigma, \delta_\mathcal{I}, q_0, Q_m, \mathcal{I}, p_\mathcal{I})$, let $f_{\mathcal{G}_\mathcal{I}} : Q \mapsto 2^\mathcal{I}$ be a map that assigns to each state $q \in Q$ the set $f_{\mathcal{G}_\mathcal{I}}(q) := \{I \in \mathcal{I} \mid \exists \sigma \in \Sigma, \delta_\mathcal{I}(q, \sigma, I)!\}$ of intervals defined at state q . Consider the transition $\delta_\mathcal{I}(q, \sigma, I) = q'$ of $\mathcal{G}_\mathcal{I}$. If $I.r = \infty$, then at state q the event σ can be fired at any later moment when the timer value reaches at least $R(I) = I.l$. If $I.r < \infty$, then at state q the transition $\delta_\mathcal{I}(q, \sigma, I) = q'$ cannot be fired when the timer value reaches at least $R(I) = I.r + 1$. Intuitively, $R(I)$ provides a region of stabilization of the timer values for the transition $\delta_\mathcal{I}(q, \sigma, I) = q'$. Then, by definition, $S(f_{\mathcal{G}_\mathcal{I}}(q))$ provides a region of stabilization of the timer values for all the transitions defined at state q , and we only need to count the timer value up to $S(f_{\mathcal{G}_\mathcal{I}}(q))$ for each state q . Now we are ready to provide a semantic interpretation of finite interval automata in terms of finite tick automata.

Let $\mathcal{G}_\mathcal{I} = (Q, \Sigma, \delta_\mathcal{I}, q_0, Q_m, \mathcal{I})$ be any given finite interval automaton over Σ . Consider the following finite tick automaton $\mathcal{G}' := (Q', \Sigma \cup \{t\}, \delta'^\#, q'_0, Q'_m)$ over Σ , where $Q', \delta'^\#, q'_0$ and Q'_m are defined as follows:

- 1) for each state $q \in Q$, create $S(f_{\mathcal{G}_\mathcal{I}}(q)) + 1$ superscripted states $\{q^0, q^1, q^2, \dots, q^{S(f_{\mathcal{G}_\mathcal{I}}(q))}\}$. Let

$$Q' := \bigcup_{q \in Q} \{q^0, q^1, q^2, \dots, q^{S(f_{\mathcal{G}_\mathcal{I}}(q))}\}$$

- 2) $\delta'^\#$ is defined such that for each state $q, q_1 \in Q$,
 - a) $\delta'^\#(q^{S(f_{\mathcal{G}_\mathcal{I}}(q))}, t) := q^{S(f_{\mathcal{G}_\mathcal{I}}(q))}$

- b) for each $i \in [0, S(f_{\mathcal{G}_\mathcal{I}}(q)) - 1]$, $\delta'^\#(q^i, t) := q^{i+1}$
- c) if $\delta_\mathcal{I}^\#(q, \sigma, I) = q_1 \wedge I.r < \infty$, then for each $k \in \mathbb{N}$, $\delta'^\#(q^k, \sigma) := q_1^0$
- d) if $\delta_\mathcal{I}^\#(q, \sigma^\#, I) = q_1$, then for each $I.l \leq k < I.r$, $\delta'^\#(q^k, \sigma) := q_1^0$ and $\delta'^\#(q^{I.r}, \sigma^\#) := q_1^0$
- e) if $\delta_\mathcal{I}^\#(q, \sigma, I) = q_1 \wedge I.r = \infty$, then for each $I.l \leq k \leq S(f_{\mathcal{G}_\mathcal{I}}(q))$, $\delta'^\#(q^k, \sigma) := q_1^0$

$$3) q'_0 := q_0^0$$

$$4) Q'_m := \bigcup_{q \in Q_m} \{q^0, q^1, q^2, \dots, q^{S(f_{\mathcal{G}_\mathcal{I}}(q))}\}$$

The above constructed finite tick automaton $\mathcal{G}'$ is said to be the *tick interpretation* of interval automaton $\mathcal{G}_\mathcal{I}$, denoted $\mathcal{T}(\mathcal{G}_\mathcal{I})$. Intuitively, timer values are incorporated into each state as the superscripts. Indeed, state $q^i \in Q'$, where $i < S(f_{\mathcal{G}_\mathcal{I}}(q))$, of $\mathcal{G}'$ is used to represent that the interval automaton $\mathcal{G}_\mathcal{I}$ is at state $q \in Q$ and the timer value is $c = i$; state $q^i \in Q'$, when $i = S(f_{\mathcal{G}_\mathcal{I}}(q))$, is used to represent that the interval automaton $\mathcal{G}_\mathcal{I}$ is at state $q \in Q$ and the timer value is $c \geq S(f_{\mathcal{G}_\mathcal{I}}(q))$. We remark that the constructed $\mathcal{G}'$ is indeed a finite tick automaton over Σ , i.e., Conditions 1), 2), 3), 4) and 5) of Definition 4 are satisfied. We note that in the above construction, the state size of the constructed finite tick automaton is upper bounded by $|Q|N$, where $N := \max\{S(f_{\mathcal{G}_\mathcal{I}}(q)) \mid q \in Q\} + 1$. Thus, a finite interval automaton could be a much compact representation of a finite tick automaton, especially when N is large. In the rest, we shall not distinguish between the superscripted state q^c and the 2-tuple (q, c) representation.

3) *Interval Interpretation*: In the rest of this subsection, we address the problem of constructing an equivalent finite interval automaton corresponding to a given finite tick automaton, which will also facilitate the understanding of the construction of the interval synchronous product in the next subsection.

Let $\mathcal{G} = (Q, \Sigma \cup \{t\}, \delta^\#, q_0, Q_m)$ be any given finite tick automaton over Σ . We define a finite interval automaton $\mathcal{G}'_\mathcal{I} := (Q', \Sigma, \delta'^\#, q'_0, Q'_m, \mathcal{I})$ over Σ , where

1)

$$Q' := \{q \in Q \mid \exists q' \in Q, \sigma \in \Sigma, \delta(q', \sigma) = q\} \cup \{q_0\}$$

$$2) q'_0 := q_0$$

$$3) Q'_m := \{q \in Q' \mid q \in Q_m\}$$

$$4) \text{ for any } \alpha \in \Sigma \cup \Sigma^\#, \text{ any } q, q' \in Q' \text{ and any } k \in \mathbb{N}, \delta'^\#_\mathcal{I}(q, \alpha, [k, k]) := q' \text{ if } \delta^\#(q, t^k \alpha) = q' \wedge \delta^\#(q, t^{k+1}) \neq \delta^\#(q, t^k)$$

$$5) \text{ for any } \sigma \in \Sigma, \text{ any } q, q' \in Q' \text{ and any } k \in \mathbb{N},$$

$$\delta'^\#_\mathcal{I}(q, \sigma, [k, \infty)) := q' \text{ if } \delta^\#(q, t^k \sigma) = q' \wedge \delta^\#(q, t^{k+1}) = \delta^\#(q, t^k)$$

The above constructed $\mathcal{G}'_\mathcal{I}$ satisfies Conditions 1), 2), 3) and 4) of Definition 3 and is said to be the *interval interpretation* of $\mathcal{G}$, denoted $\mathcal{I}(\mathcal{G})$. The definition of $\delta'^\#_\mathcal{I}$ could be made more compact by merging the set

$$\begin{aligned} \{\delta'^\#_\mathcal{I}(q, \sigma, [k, k]) = q', \delta'^\#_\mathcal{I}(q, \sigma, [k+1, k+1]) = \\ q' \dots \delta'^\#_\mathcal{I}(q, \sigma, [k+n-1, k+n-1]) = \\ q', \delta'^\#_\mathcal{I}(q, \alpha, [k+n, k+n]) = q'\} \end{aligned}$$

of transitions, where $\alpha = \sigma$ or $\alpha = \sigma^\#$, into the transition $\delta'^\#_\mathcal{I}(q, \alpha, [k, k+n]) = q'$ and the set

$$\begin{aligned} \{\delta'^\#_\mathcal{I}(q, \sigma, [k, k]) = q', \delta'^\#_\mathcal{I}(q, \sigma, [k+1, k+1]) = \\ q' \dots \delta'^\#_\mathcal{I}(q, \sigma, [k+n-1, k+n-1]) = \end{aligned}$$

$$q', \delta_{\mathcal{I}}^{\#}(q, \sigma, [k+n, \infty)) = q'\}$$

of transitions into the transition $\delta_{\mathcal{I}}^{\#}(q, \sigma, [k, \infty)) = q'$.

V. INTERVAL SYNCHRONOUS PRODUCT

The synchronous interaction of finite interval automata can be fully captured by their synchronous product. We shall occasionally refer to synchronous product in the interval domain as interval synchronous product and synchronous product in the tick domain as tick synchronous product. The definition of interval synchronous product is provided below. We shall use the symbol “ $\prod$ ” to denote the interval synchronous product operation. In order to motivate its construction and show the defined interval synchronous product operation is correct, we first remind about the synchronous product operation on finite tick automata, which is defined to be exactly the same as the synchronous product operation on finite state automata, except that σ can be synchronized with $\sigma^{\#}$ and $\sigma^{\#}$ is rendered. In the rest, the synchronous product of two finite tick automata $\mathcal{G}_1, \mathcal{G}_2$ is denoted by $\mathcal{G}_1 \parallel \mathcal{G}_2$.

Definition 5. Let $\mathcal{G}_{i,\mathcal{I}} = (Q_i, \Sigma_i, \delta_{i,\mathcal{I}}, q_{i,0}, Q_{i,m}, \mathcal{I}, p_{i,\mathcal{I}})$, $i = 1, 2$, be any two finite interval automata. Their interval synchronous product $\mathcal{G}_{1,\mathcal{I}} \prod \mathcal{G}_{2,\mathcal{I}}$ is defined to be the finite interval automaton $\mathcal{G}_{\mathcal{I}} := (Q, \Sigma, \delta_{\mathcal{I}}, q_0, Q_m, \mathcal{I}, p_{\mathcal{I}})$, where

1) $Q :=$

$$\{(q_1, q_2, c_1, c_2) \mid q_1 \in Q_1, q_2 \in Q_2, 0 \leq c_1 \leq S(f_{\mathcal{G}_{1,\mathcal{I}}}(q_1)), 0 \leq c_2 \leq S(f_{\mathcal{G}_{2,\mathcal{I}}}(q_2))\}$$

2) $\Sigma := \Sigma_1 \cup \Sigma_2$

3) $\delta_{\mathcal{I}} : Q \times \Sigma \times \mathcal{I} \mapsto Q$ is the partial transition function such that for any $q_1, q'_1 \in Q_1, q_2, q'_2 \in Q_2, \sigma \in \Sigma, c_1 \in [0, S(f_{\mathcal{G}_{1,\mathcal{I}}}(q_1))], c_2 \in [0, S(f_{\mathcal{G}_{2,\mathcal{I}}}(q_2))], k \in \mathbb{N}, I_1, I_2 \in \mathcal{I}$,

a) if

$$\begin{aligned} \sigma \in \Sigma_1 \cap \Sigma_2, \delta_{1,\mathcal{I}}(q_1, \sigma, I_1) = q'_1 \text{ and } \\ \delta_{2,\mathcal{I}}(q_2, \sigma, I_2) = q'_2, \text{ then} \\ \delta_{\mathcal{I}}((q_1, q_2, c_1, c_2), \sigma, (I_1 \dot{-} c_1) \cap (I_2 \dot{-} c_2)) = \\ (q'_1, q'_2, 0, 0) \end{aligned}$$

b) if

$$\begin{aligned} \sigma \in \Sigma_1 - \Sigma_2, \delta_{1,\mathcal{I}}(q_1, \sigma, I_1) = q'_1, k \in I_1 \dot{-} c_1 \text{ and } \\ k < S(f_{\mathcal{G}_{2,\mathcal{I}}}(q_2)) - c_2, \text{ then} \\ \delta_{\mathcal{I}}((q_1, q_2, c_1, c_2), \sigma, [k, k]) = (q'_1, q_2, 0, c_2 + k) \end{aligned}$$

c) if

$$\begin{aligned} \sigma \in \Sigma_1 - \Sigma_2, \delta_{1,\mathcal{I}}(q_1, \sigma, I_1) = q'_1, \text{ then} \\ \delta_{\mathcal{I}}((q_1, q_2, c_1, c_2), \sigma, (I_1 \dot{-} c_1) \cap [S(f_{\mathcal{G}_{2,\mathcal{I}}}(q_2)) - c_2, \infty)) = (q'_1, q_2, 0, S(f_{\mathcal{G}_{2,\mathcal{I}}}(q_2))) \end{aligned}$$

d) if

$$\begin{aligned} \sigma \in \Sigma_2 - \Sigma_1, \delta_{2,\mathcal{I}}(q_2, \sigma, I_2) = q'_2, k \in I_2 \dot{-} c_2 \text{ and } \\ k < S(f_{\mathcal{G}_{1,\mathcal{I}}}(q_1)) - c_1, \text{ then} \\ \delta_{\mathcal{I}}((q_1, q_2, c_1, c_2), \sigma, [k, k]) = (q_1, q'_2, c_1 + k, 0) \end{aligned}$$

e) if

$$\begin{aligned} \sigma \in \Sigma_2 - \Sigma_1, \delta_{2,\mathcal{I}}(q_2, \sigma, I_2) = q'_2, \text{ then} \\ \delta_{\mathcal{I}}((q_1, q_2, c_1, c_2), \sigma, (I_2 \dot{-} c_2) \cap [S(f_{\mathcal{G}_{1,\mathcal{I}}}(q_1)) - c_1, \infty)) = (q_1, q'_2, S(f_{\mathcal{G}_{1,\mathcal{I}}}(q_1)), 0) \end{aligned}$$

4) $q_0 := (q_{1,0}, q_{2,0}, 0, 0)$

5) $Q_m :=$

$$\{(q_1, q_2, c_1, c_2) \mid q_1 \in Q_{1,m}, q_2 \in Q_{2,m}, 0 \leq c_1 \leq S(f_{\mathcal{G}_{1,\mathcal{I}}}(q_1)), 0 \leq c_2 \leq S(f_{\mathcal{G}_{2,\mathcal{I}}}(q_2))\}$$

6) $p_{\mathcal{I}} : Q \times \Sigma \times \mathcal{I} \mapsto Q$ is the partial map defined on $\text{dom}(\delta_{\mathcal{I}})$ such that $p_{\mathcal{I}}((q_1, q_2, c_1, c_2), \sigma, I) :=$

- a) I , if $\sigma \in \Sigma_1 \cap \Sigma_2$, $(\exists k \in \mathbb{N}, p_{1,\mathcal{I}}(q_1, \sigma, [k, I.r + c_1]) = 1 \vee p_{2,\mathcal{I}}(q_2, \sigma, [k, I.r + c_2]) = 1)$ and $I.r < \infty$
- b) I , if $\sigma \in \Sigma_1 - \Sigma_2$, $(\exists k \in \mathbb{N}, p_{1,\mathcal{I}}(q_1, \sigma, [k, I.r + c_1]) = 1)$ and $I.r < \infty$
- c) I , if $\sigma \in \Sigma_2 - \Sigma_1$, $(\exists k \in \mathbb{N}, p_{2,\mathcal{I}}(q_2, \sigma, [k, I.r + c_2]) = 1)$ and $I.r < \infty$
- d) 0 , otherwise

The underlying idea of Definition 5 is now explained below: instead of explicitly converting the two finite interval automata to equivalent finite tick automata, constructing the synchronous product in the tick domain and then followed by converting the resulting finite tick automaton to an equivalent finite interval automaton, we keep these several steps implicit and avoid the enumeration of ticks of the clock, using interval operations. The states and timer values of both finite interval automata are used to construct the set of states. In order to construct the transitions defined at each state (q_1, q_2, c_1, c_2) , we need to examine those sequences of transitions labeled by $t^k \sigma$ that are defined at the corresponding state (q_1, c_1, q_2, c_2) of the synchronous product of the equivalent finite tick automata. In particular, different sequences of transitions labeled by $t^k \sigma$, for the same event $\sigma \in \Sigma$, could be combined into an interval transition labeled by σ defined at state (q_1, q_2, c_1, c_2) . There are several cases that shall be analyzed. Item 3.a) deals with the case when $\sigma \in \Sigma_1 \cap \Sigma_2$. Suppose $\delta_{1,\mathcal{I}}(q_1, \sigma, I_1) = q'_1$ and $\delta_{2,\mathcal{I}}(q_2, \sigma, I_2) = q'_2$. Then at state (q_1, c_1) of $\mathcal{T}(\mathcal{G}_{1,\mathcal{I}})$, event σ can be fired after the occurrence of c ticks, for any $c \in I_1 \dot{-} c_1$. Similarly, at state (q_2, c_2) of $\mathcal{T}(\mathcal{G}_{2,\mathcal{I}})$, event σ can be fired after the occurrence of c ticks, for any $c \in I_2 \dot{-} c_2$. Thus, at state (q_1, c_1, q_2, c_2) , event σ can be fired after the occurrence of c ticks, for any $c \in (I_1 \dot{-} c_1) \cap (I_2 \dot{-} c_2)$, and state $(q'_1, 0, q'_2, 0)$ is entered. This corresponds to the transition $\delta_{\mathcal{I}}((q_1, q_2, c_1, c_2), \sigma, (I_1 \dot{-} c_1) \cap (I_2 \dot{-} c_2)) = (q'_1, q'_2, 0, 0)$ that is defined at state (q_1, q_2, c_1, c_2) . Items 3.b) and 3.c) deal with the case when $\sigma \in \Sigma_1 - \Sigma_2$. Suppose $\delta_{1,\mathcal{I}}(q_1, \sigma, I_1) = q'_1$. If $k \in I_1 \dot{-} c_1$, then at state (q_1, c_1, q_2, c_2) , event σ can be fired after the occurrence of k ticks, and state $(q'_1, 0, q_2, \max\{c_2 + k, S(f_{\mathcal{G}_{2,\mathcal{I}}}(q_2)))\})$ is entered. There are two different cases, depending on the range of k . When $k < S(f_{\mathcal{G}_{2,\mathcal{I}}}(q_2)) - c_2$, we have $\max\{c_2 + k, S(f_{\mathcal{G}_{2,\mathcal{I}}}(q_2)))\} = c_2 + k$. In particular, $c_2 + k$ is strictly increasing when $k < S(f_{\mathcal{G}_{2,\mathcal{I}}}(q_2)) - c_2$, and thus different states are entered for different values of k . This corresponds to the transition $\delta_{\mathcal{I}}((q_1, q_2, c_1, c_2), \sigma, [k, k]) = (q'_1, q_2, 0, c_2 + k)$ defined at state (q_1, q_2, c_1, c_2) . Whenever $k \geq S(f_{\mathcal{G}_{2,\mathcal{I}}}(q_2)) - c_2$, the same state $(q'_1, 0, q_2, S(f_{\mathcal{G}_{2,\mathcal{I}}}(q_2)))$ is entered. This corresponds to the transition $\delta_{\mathcal{I}}((q_1, q_2, c_1, c_2), \sigma, (I_1 \dot{-} c_1) \cap [S(f_{\mathcal{G}_{2,\mathcal{I}}}(q_2)) - c_2, \infty)) = (q'_1, q_2, 0, S(f_{\mathcal{G}_{2,\mathcal{I}}}(q_2)))$. Items 3.d) and 3.e) are exactly symmetric to Items 3.b) and 3.c), and deal with the case when $\sigma \in \Sigma_2 - \Sigma_1$. Items 4 and 5) then define respectively the initial state and the set of marked states, which is quite straightforward. Item 6) deals with the preemption