Contents lists available at ScienceDirect

European Journal of Control

journal homepage: www.sciencedirect.com/journal/european-journal-of-control

Bounded-time nonblocking supervisory control of timed discrete-event systems[☆]

Renyuan Zhang[a],[*], Junhua Gou[a], Yabo Zhu[a], Bei Yang[a], Kai Cai[b]

[a] School of Automation, Northwestern Polytechnical University, Xi'an, China
[b] Control and Artificial Intelligence Group, Department of Core Informatics, Osaka Metropolitan University, Osaka, Japan

ARTICLE INFO

Recommended by T. Parisini

Keywords:
Timed discrete-event systems
Nonblocking supervisory control
Bounded-time nonblockingness
Bounded-time language completability

ABSTRACT

Recently an automaton property of quantitative nonblockingness was proposed in supervisory control of untimed discrete-event systems (DES), which *quantifies* the standard nonblocking property by capturing the practical requirement that all tasks be completed within a bounded number of steps. However, in practice tasks may be further required to be completed in specific time. To meet this new requirement, in this paper we introduce the concept of *bounded-time nonblockingness*, which extends the concept of quantitative nonblockingness from untimed DES to timed DES. This property requires that each task must be completed within a bounded time counted by the number of *ticks*, rather than bounded number of transition steps in quantitative nonblockingness. Accordingly, we formulate a new bounded-time nonblocking supervisory control problem (BTNSCP) of timed DES, and characterize its solvability in terms of a new concept of *bounded-time language completability*. Then we present an approach to compute the maximally permissive solution to the new BTNSCP.

1. Introduction

In standard supervisory control of (timed) discrete-event systems (DES) (Brandin & Wonham, 1994; Cai & Wonham, 2020; Cassandras & Lafortune, 2008; Ramadge & Wonham, 1987, 1989; Wonham & Cai, 2019; Wonham, Cai, & Rudie, 2018; Wonham & Ramadge, 1987), and other extensions and applications on nonblocking supervisory control (e.g. Balemi, Hoffmann, Gyugyi, Wong-Toi, and Franklin (1993), Brandin and Charbonnier (1994), Fabian and Kumar (1997), Kumar and Shayman (1994), Ma and Wonham (2006), Malik (2003), Malik and Leduc (2008)), the plant to be controlled is modeled by finite-state automata and *marker states* are used to represent 'desired states'. A desired state can be a goal location, a start/home configuration, or a task completion (Eilenberg, 1974; Wonham & Cai, 2019). Besides enforcing all imposed control specifications, a *nonblocking supervisor* ensures that every system trajectory can reach a marker state (in a finite number of steps). As a result, the system under supervision may always be able to reach a goal, return home, or complete a task.

The nonblocking property only *qualitatively* guarantees finite reachability of marker states. There is no given bound on the number of steps or time for reaching marker states, so it can take an arbitrarily large (though finite) number of steps or time before a marker state is reached. Consequently, this qualitatively nonblocking property might not be sufficient for many practical purposes, especially when there are prescribed bounds (of transition steps or time) for reaching desired states. To address this issue, recently Zhang, Wang, and Cai (2021) and Zhang , Wang , Wang, and Cai (2024) proposed a *quantitatively* nonblocking property to capture the practical requirement that marker states be reached within a prescribed number of (transition) steps. By this property, in the worst case, each reachable state can reach a marker state in no more than a prescribed number of steps. However, in many practical cases, tasks (represented by marker states) may often be required to be completed within a specific bounded *time*. For example, a rescue vehicle is required not only to reach a goal location but to do so within a given time; a warehouse AGV is expected not only to return to a self-charging area but to do so periodically with a predetermined period; and a production machine is required not only to complete a task (e.g. processing a workpiece) but also to do so within a prescribed time. In Section 2 below, we will present a more detailed motivation example.

☆ This work was supported in part by the Key Research and Development Projects of Shaanxi Province under grant no. 2024GX-ZDCYL-02-06, the Industrial Key Research and Development Projects of Shaanxi Province under grant no. 2024CY2-GJHX-91, the National Key Research and Development Program of China under Grant no. 2022YFB4702200, and JSPS KAKENHI under grant no. 22KK0155.

* Corresponding author.
E-mail addresses: ryzhang@nwpu.edu.cn (R. Zhang), kai.cai@eng.osaka-cu.ac.jp (K. Cai).

https://doi.org/10.1016/j.ejcon.2025.101266
Received 8 February 2025; Received in revised form 5 May 2025; Accepted 10 June 2025

With the above motivation, we extend the concept of quantitative nonblockingness from untimed DES (the elapse of time is not explicitly modeled) to the framework of timed DES (TDES) (Brandin & Wonham, 1994; Wonham & Cai, 2019), where the occurrence of a time unit is described by the occurrence of a *tick* event. Thus the bound of time for reaching a subset of marker states can be represented by the number of *ticks*. In this framework, we measure the 'maximal time' between the reachable states and the specific subset of marker states, and this is done by counting the number of *ticks* (rather than the number of all events) in every string leading a reachable state to one of the marker states in the specified subset. More specifically, consider a TDES plant modeled by a *tick*-automaton (which is a finite-state automaton with a special *tick* event), a cover $\{Q_{m,i} | i \in \mathcal{I}\}$ ($\mathcal{I}$ is an index set) on the set of marker states of the *tick*-automaton, and let N_i be a finite positive integer which denotes the required number of *ticks* to reach marker states in $Q_{m,i}$. We define a *bounded-time nonblocking property* (with respect to $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$) of the *tick*-automaton if from every reachable state of this *tick*-automaton, the number of *ticks* included in each string that leads the reachable state to marker states in $Q_{m,i}$ is smaller or equal to N_i. That is, in the worst case, for every marker state subset $Q_{m,i}$, every reachable state can reach $Q_{m,i}$ in no more than N_i *ticks* following any string.

Moreover, we formulate a new *bounded-time nonblocking supervisory control problem of TDES* by requiring a supervisory control solution to be implementable by a bounded-time nonblocking *tick*-automaton. To solve this problem, we present a necessary and sufficient condition by identifying a language property called *bounded-time completability*. The latter roughly means that in the worst case, for every sublanguage K_i ($i \in \mathcal{I}$) (defined according to a particular type of task corresponding to $Q_{m,i}$) of a given language K, every string in the closure of K can be extended to a string in the sublanguage K_i in no more than N_i *ticks*. Further we show that this bounded-time language completability is closed under arbitrary set unions, and together with language controllability which is also closed under unions, a maximally permissive solution exists for the newly formulated bounded-time nonblocking supervisory control problem of TDES. Finally we design effective algorithms for the computation of such an optimal solution.

Detailed literature review on work related to quantitative nonblockingness of untimed DES is referred in Zhang et al. (2021), Zhang, Wang, et al. (2024). For TDES, time-bounded liveness is similar to our concept of bounded-time nonblockingness; time-bounded liveness is introduced in Berard et al. (2001) to express and verify constraints on the delays described by exact time, and there are algorithms (e.g. Bonakdarpour and Kulkarni (2006)) and tools (e.g. KRONORS (Daws, Olivero, Tripakis, & SergioYovine, 2002)) that can verify the time-bounded liveness. However, unlike our supervisory control problems of TDES, uncontrollable events and maximal permissiveness of control strategies are not considered. Other related works including multi-step opacity (Zhang, Xia, & Fu, 2024), initial-state observability (Zhang, Xia, Fu, & Chen, 2022) and reachability analysis (Zhang, Xia, Chen, Yang, & Chen, 2020) are also reported in the literature.

The contributions of this paper are as follows.

- First, for a given set of tasks each represented by a subset $Q_{m,i}$ of marker states and associated with a positive integer N_i, we propose a new property of *tick*-automaton, called bounded-time nonblockingness. This property quantifies the standard nonblocking property by capturing the practical requirement that each task be completed within the bounded N_i *ticks*, which extends the concept of quantitative nonblockingness in untimed DES by counting the occurrences of the event *tick*.
- Second, we formulate a new bounded-time nonblocking supervisory control problem of TDES, and characterize its solvability by a bounded-time language completability in addition to language controllability. This problem and its solvability condition are again generalizations of the standard supervisory control problem and solvability condition, and extensions of the quantitatively nonblocking supervisory control problem.
- Third, we prove that the language property of bounded-time completability is closed under arbitrary set unions, and develop an automaton-based algorithm to compute the supremal bounded-time completable sublanguage. Comparing with the algorithm in Zhang, Wang, et al. (2024) for computing the supremal quantitatively completable sublanguage, the range of the counter d and the rules for updating d are different; the details are explained in Section 4.
- Fourth, we present a fixpoint algorithm to compute the supremal controllable and bounded-time completable sublanguage of a given (specification) language, which synthesizes an optimal (maximally permissive) supervisory control solution for the bounded-time nonblocking supervisory control problem of TDES. Moreover, this solution algorithm is of polynomial complexity.

The rest of this paper is organized as follows. Section 2 reviews the nonblocking supervisory control theory of TDES and presents a motivating example for this work. Section 3 introduces the new concept of bounded-time nonblocking *tick*-automaton, and formulates the bounded-time nonblocking supervisory control problem (BTNSCP) of TDES. Section 4 presents a necessary and sufficient condition for the solvability of BTNSCP in terms of a new concept of bounded-time language completability, and develops an algorithm to compute the supremal bounded-time completable sublanguage. Section 5 presents a solution to the bounded-time nonblocking supervisory control problem. Finally Section 6 states our conclusion.

2. Preliminaries and motivating example

In this section, we review the standard nonblocking supervisory control theory of TDES in the Brandin–Wonham framework (Brandin & Wonham, 1994)(Wonham & Cai, 2019, Chapter 9), and present a motivating example for our work.

2.1. Nonblocking supervisory control of TDES

First consider the untimed DES model $\mathbf{G}_{act} = (A, \Sigma_{act}, \delta_{act}, a_0, A_m)$; here A is the finite set of *activities*, Σ_{act} the finite set of *events*, $\delta_{act} : A \times \Sigma_{act} \to A$ the (partial) *transition function*, $a_0 \in A$ the *initial activity*, and $A_m \subseteq A$ the set of *marker activities*. Let $\mathbb{N}$ denote the set of natural numbers $\{0, 1, 2, \dots\}$, and introduce *time* into $\mathbf{G}_{act}$ by assigning to each event $\sigma \in \Sigma_{act}$ a *lower bound* $l_{\mathbf{G},\sigma} \in \mathbb{N}$ and an *upper bound* $u_{\mathbf{G},\sigma} \in \mathbb{N} \cup \{\infty\}$, such that $l_{\mathbf{G},\sigma} \le u_{\mathbf{G},\sigma}$. Also introduce a distinguished event, written *tick*, to represent "tick of the global clock". Then the TDES model is described by a *tick*-automaton

$$\mathbf{G} := (Q, \Sigma, \delta, q_0, Q_m), \tag{1}$$

which is constructed from $\mathbf{G}_{act}$ (the detailed construction rules are referred to Brandin and Wonham (1994) and Wonham and Cai (2019, Chapter 9)) such that Q is the finite set of *states*, $\Sigma := \Sigma_{act} \dot\cup \{tick\}$ the finite set of events, $\delta : Q \times \Sigma \to Q$ the (partial) *state transition function*, q_0 the *initial state*, and Q_m the set of *marker states*.

Let Σ^* be the set of all finite strings of elements in $\Sigma = \Sigma_{act} \dot\cup \{tick\}$, including the empty string ϵ. The transition function δ is extended to $\delta : Q \times \Sigma^* \to Q$ in the usual way. The *closed behavior* of $\mathbf{G}$ is the language $L(\mathbf{G}) := \{s \in \Sigma^* | \delta(q_0, s)!\}$ and the *marked behavior* is $L_m(\mathbf{G}) := \{s \in L(\mathbf{G}) | \delta(q_0, s) \in Q_m\} \subseteq L(\mathbf{G})$. Let $K \subseteq \Sigma^*$ be a language; its *prefix closure* is $\overline{K} := \{s \in \Sigma^* | (\exists t \in \Sigma^*)\ st \in K\}$. K is said to be $L_m(\mathbf{G})$*-closed* if $\overline{K} \cap L_m(\mathbf{G}) = K$.

For *tick*-automaton $\mathbf{G}$ as in (1), a state $q \in Q$ is *reachable* if there is a string $s \in L(\mathbf{G})$ such that $q = \delta(q_0, s)$; state $q \in Q$ is *coreachable* if there is a string $s \in \Sigma^*$ such that $\delta(q, s)!$ and $\delta(q, s) \in Q_m$. We say that $\mathbf{G}$ is *nonblocking* if every reachable state in $\mathbf{G}$ is coreachable. In fact $\mathbf{G}$ is *nonblocking* if and only if $\overline{L_m(\mathbf{G})} = L(\mathbf{G})$.

To use $\mathbf{G}$ in (1) for supervisory control, first designate a subset of events, denoted by $\Sigma_{hib} \subseteq \Sigma_{act}$, to be the *prohibitable* events which can be disabled by an external supervisor. Next, and specific to TDES, specify a subset of *forcible* events, denoted by $\Sigma_{for} \subseteq \Sigma_{act}$, which can *preempt* the occurrence of event *tick*.

Now it is convenient to define the *controllable* event set $\Sigma_c := \Sigma_{hib} \dot\cup \{tick\}$. The *uncontrollable* event set is $\Sigma_{uc} := \Sigma \setminus \Sigma_c$. A *supervisory control* for $\mathbf{G}$ is any map $V : L(\mathbf{G}) \to \Gamma$, where $\Gamma := \{\gamma \subseteq \Sigma \mid \gamma \supseteq \Sigma_{uc}\}$. Then the *closed-loop system* is denoted by $V/\mathbf{G}$, with closed behavior $L(V/\mathbf{G})$ defined as follows:

(i) $\epsilon \in L(V/\mathbf{G})$;

(ii) $s \in L(V/\mathbf{G})$ & $\sigma \in V(s)$ & $s\sigma \in L(\mathbf{G}) \Rightarrow s\sigma \in L(V/\mathbf{G})$;

(iii) no other strings belong to $L(V/\mathbf{G})$.

On the other hand, for any sublanguage $K \subseteq L_m(\mathbf{G})$, the closed-loop system's marked behavior $L_m(V/\mathbf{G})$ is given by[1]

$$L_m(V/\mathbf{G}) := K \cap L(V/\mathbf{G}).$$

The closed behavior $L(V/\mathbf{G})$ represents the strings generated by the plant $\mathbf{G}$ under the control of V, while the marked behavior $L_m(V/\mathbf{G})$ represents the strings that have some special significance, for instance representing 'task completion'. We say that V is *nonblocking* if

$$\overline{L_m(V/\mathbf{G})} = L(V/\mathbf{G}). \quad (2)$$

A sublanguage $K \subseteq L_m(\mathbf{G})$ is *controllable* if, for all $s \in \overline{K}$,

$$Elig_K(s) \supseteq \begin{cases} Elig_{\mathbf{G}}(s) \cap (\Sigma_{uc} \dot\cup \{tick\}) \\ \qquad \text{if } Elig_K(s) \cap \Sigma_{for} = \emptyset, \\ Elig_{\mathbf{G}}(s) \cap \Sigma_{uc} \\ \qquad \text{if } Elig_K(s) \cap \Sigma_{for} \neq \emptyset \end{cases}$$

where $Elig_{\mathbf{G}}(s) := \{\sigma \in \Sigma \mid s\sigma \in L(\mathbf{G})\}$ and $Elig_K(s) := \{\sigma \in \Sigma \mid s\sigma \in \overline{K}\}$ are the subset of eligible events after string s in $L(\mathbf{G})$ and K respectively.

The following is a central result of nonblocking supervisory control theory (Brandin & Wonham, 1994; Wonham & Cai, 2019).

Theorem 1. *Let $K \subseteq L_m(\mathbf{G})$, $K \neq \emptyset$. There exists a nonblocking (marking) TDES supervisory control V (for $(K, \mathbf{G})$) such that $L_m(V/\mathbf{G}) = K$ if and only if K is controllable. Moreover, if such a nonblocking TDES supervisory control V exists, then it may be implemented by a nonblocking supervisor* **SUP**, *i.e.* $L_m(\mathbf{SUP}) = L_m(V/\mathbf{G})$. ⋄

Further, the property of language controllability is closed under set union. Hence for any language $K \subseteq L_m(\mathbf{G})$ (whether or not controllable), the set $\mathcal{C}(K) = \{K' \subseteq K \mid K' \text{ is controllable wrt. } \mathbf{G}\}$ contains a unique supremal element denoted by $\sup \mathcal{C}(K)$. Whenever $\sup \mathcal{C}(K)$ is nonempty, by Theorem 1 there exists a nonblocking supervisory control $V_{\sup}$ that satisfies $L_m(V_{\sup}/\mathbf{G}) = \sup \mathcal{C}(K)$ and may be implemented by a nonblocking supervisor **SUP** with

$$L_m(\mathbf{SUP}) = L_m(V_{\sup}/\mathbf{G}).$$

2.2. Motivating example

Nonblockingness of supervisory control V describes a general requirement that every string generated by the closed-loop system $V/\mathbf{G}$ can be completed to a marked string in indefinite (finite but unbounded) number of transitions, which may include indefinite number of *ticks*. Namely, the time needed for completing a task is unbounded. However, in many real-world applications, it is often required that a task be completed in a prescribed, bounded time. As an illustration, we present the following example (adapted from Example 1 in Zhang, Wang, et al. (2024)).

[1] With this definition of $L_m(V/\mathbf{G})$, the supervisory control V is also known as a *marking supervisory control* for $(K, \mathbf{G})$ (Wonham & Cai, 2019).

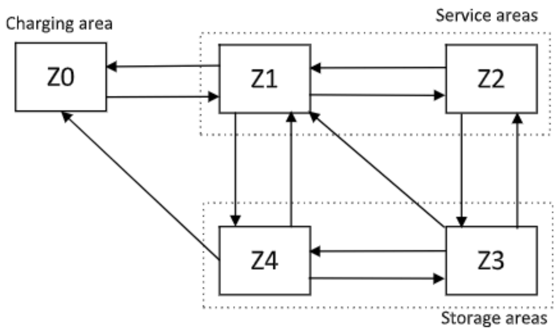

Fig. 1. Routes of the vehicle.

Example 1. Consider an autonomous vehicle for package collecting and delivery in a local region. The vehicle can move in five zones numbered 0–4, following the routes displayed in Fig. 1. Zone 0 is the charging area for the vehicle to charging its battery. Zones 1 and 2 are two service areas for customers where the customers can both receive packages from the vehicle and call the vehicle to come to collect packages to be sent. Zones 3 and 4 are the storage areas for incoming and outgoing packages. Namely, the task of the vehicle is to send packages in the storage areas (zones 3 and 4) to the service areas (zones 1 and 2), and collect packages from the service areas and store them into the storage areas. Also, the vehicle must be able to make a self-charging when it is running out of battery.

Each movement of the vehicle from one zone to the next is represented by two timed events: one represents the leaving from one zone and the other represents the arriving to the next. Both events have lower and upper bounds. As one example, assuming that the lower and upper bounds for the vehicle leaving from zone 1 for zone 0 are 1 (*tick*) and ∞ respectively and one time unit (a *tick*) represents 2 min, the vehicle may start to move from zone 1 at any time after 1 *tick* elapsed. For another example, assuming that the lower and upper bounds for the vehicle arriving zone 0 from zone 1 are 1 (*tick*) and 2 (*ticks*) respectively, the vehicle may arrive zone 0 in 1 or 2 *ticks* if it has left zone 1. Similarly, we assign lower and upper times bounds to other timed events, as displayed in Table 1.

We model the movement of the autonomous vehicle by an untimed DES model $\mathbf{G}_{act}$ with transition graph displayed in Fig. 2. States 0 (charging area Z0), 2 and 5 (service areas Z1 and Z2) are chosen to be the marker states, and the reaching of marker states represents that the vehicle arrives the corresponding areas. The timed DES model (*tick*-automaton) $\mathbf{G}$ of the vehicle can be generated according to the construction rules in Brandin and Wonham (1994), Wonham and Cai (2019); its transition graph is displayed as in Fig. 3. For better understanding the constructing rules, we consider the transitions $(0, 1, 1)$ and $(1, 2, 2)$ in ATG of $\mathbf{G}$ as displayed in Fig. 2 for example. Before that, as defined in Wonham and Cai (2019), the lower bound of an event represent the delay of its occurrence and the upper bound represents its hard deadline, namely, the event will occur after the lower bound, but before the upper bound. First, the lower bound and upper bound of event 1 are 1 and ∞ respectively, thus event 1 will occur after 1 *tick* delay; according to the rules in Section 9.2 in Wonham and Cai (2019), at state 0, only event 1 is defined and event 1 will occur after 1 *tick*, thus the transition $(0, tick, 1)$ is added to the TTG before event 1 occurs; then at state 1, event 1 and *tick* may occur, and since the upper bound of event 1 is ∞, event 1 may occur at any time and we add the transitions $(1, tick, 1)$ and $(1, 1, 2)$ to the TTG. Second, at state 2 of the TTG, corresponding to state 1 in ATG, only event 2 is defined and the lower and upper bounds of event 2 are 1, namely, event 2 will and must occur after 1 *tick* elapsed, thus we add the transition $(2, tick, 3)$ and $(3, 2, 4)$ to the TTG. Other transitions can be obtained similarly, which can be computed by the software TTCT (Wonham, 2016).

Table 1
Event representations of each vehicle route, and the corresponding lower and upper bounds. Notation: Zi ($i = 0, 1, 2, 3, 4, 5$) represents zone i.

route	event label	(lower bound, upper bound)
Leave $Z0$ for $Z1$	1	$(1, \infty)$
Arrive $Z1$ from $Z0$	2	(1,1)
Leave $Z1$ for $Z0$	11	$(0, \infty)$
Arrive $Z0$ from $Z1$	12	(1,1)
Leave $Z1$ for $Z2$	13	$(0, \infty)$
Arrive $Z2$ from $Z1$	14	(1,2)
Leave $Z1$ for $Z4$	15	$(0, \infty)$
Arrive $Z4$ from $Z1$	16	(1,1)
Leave $Z2$ for $Z1$	21	$(0, \infty)$
Arrive $Z1$ from $Z2$	22	(1,2)
Leave $Z2$ for $Z3$	23	$(0, \infty)$
Arrive $Z3$ from $Z2$	24	(1,1)
Leave $Z3$ for $Z1$	31	$(0, \infty)$
Arrive $Z1$ from $Z3$	32	(1,2)
Leave $Z3$ for $Z2$	33	$(0, \infty)$
Arrive $Z2$ from $Z3$	34	(1,1)
Leave $Z3$ for $Z4$	35	$(0, \infty)$
Arrive $Z4$ from $Z3$	36	(1,2)
Leave $Z4$ for $Z0$	41	$(0, \infty)$
Arrive $Z0$ from $Z4$	42	(2,2)
Leave $Z4$ for $Z1$	43	$(0, \infty)$
Arrive $Z1$ from $Z4$	44	(1,1)
Leave $Z4$ for $Z3$	45	$(0, \infty)$
Arrive $Z3$ from $Z4$	46	(1,1)

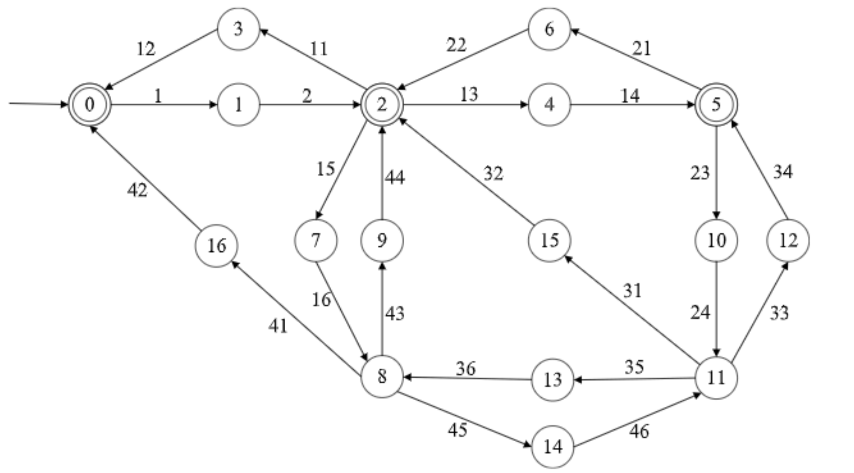

Fig. 2. Transition graph of $\mathbf{G}_{act}$. In the transition graph, states 0, 2, 5, 8, and 11 represent that the vehicle stays at the area Z0, Z1, Z2, Z4 and Z3 respectively; other states represent that the vehicle is in the process of moving from one zone to the next, e.g. state 1 represents that the vehicle is in the process of moving from zone 0 to zone 1.

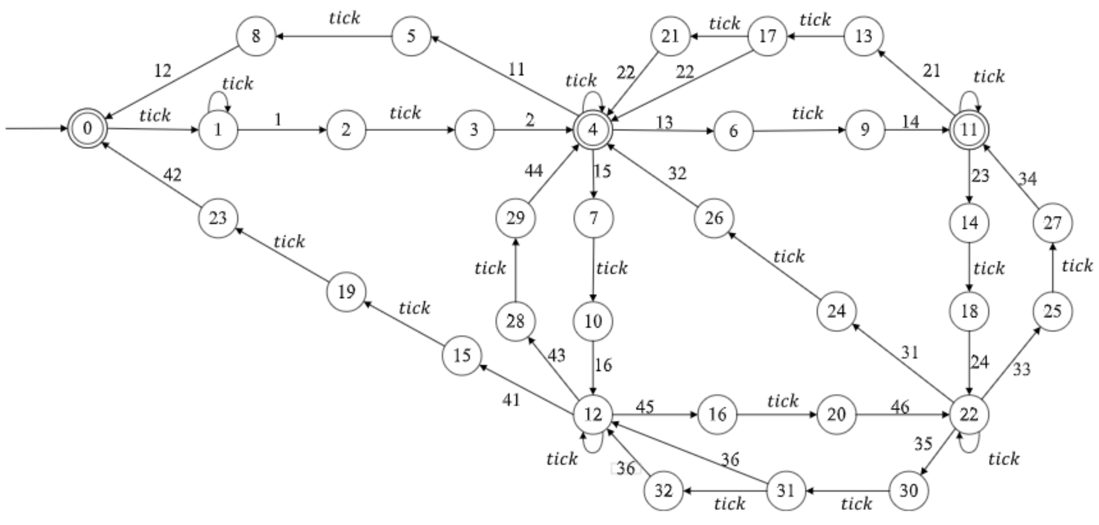

Fig. 3. Transition graph of **G**.

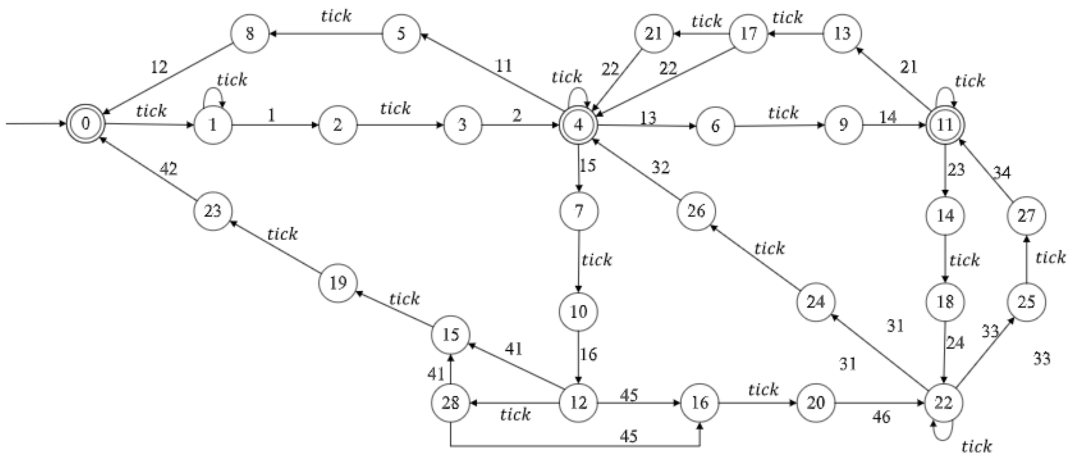

Fig. 4. Transition graph of **SUP**.

Assume that events 1, 11, 13, 15, 21, 23, 31, 33, 35, 41, 43 and 45 are both prohibitable and forcible.

Suppose that due to road maintenance, the (directed) route

zone 3 → zone 4 → zone 1

is not usable. This constraint is imposed as a safety specification. We further consider a temporal specification that after arriving zone 4, the vehicle should collect or store the package, and leave this area in 1 *tick*. To satisfy these two specifications, a nonblocking supervisory control can be synthesized (Brandin & Wonham, 1994; Wonham & Cai, 2019), and implemented by a nonblocking supervisor **SUP** as displayed in Fig. 4. This **SUP** needs to (1) disable event 35 at state 22 (zone 3), and event 43 at states 12 (zone 4) (to satisfy the safety specification); and (2) preempt event *tick* at state 12 by events 41 or 45 (to satisfy the temporal specification). Moreover, since **SUP** is nonblocking, every reachable state can reach the marker states 0, 4, 11 (representing zones 0, 1 and 2 respectively) in finite number of *ticks*.

Now consider two additional requirements:

(i) Every package sent to customers must be delivered by the vehicle to either one of the two service areas (zone 1 or 2) within 10 min (5 *ticks*); and whenever a customer calls for package collection, the vehicle must reach zone 1 or 2 within 10 min;
(ii) The vehicle must be able to return to zone 0 for charging its battery within 18 min (9 *ticks*).

Note that the above requirements are different from the temporal specification: the latter imposes temporal constraints on event occurrences; but the above requirements impose bounded time constraints on arriving marker states.

The nonblocking supervisor **SUP** in Fig. 4 fails to fulfill the above requirements, because if the vehicle is at zone 3 (state 11), it is not guaranteed that the vehicle can move to zone 1 (state 2) in 5 *ticks* (it may stay at zone 3 in any number of *ticks*).

Hence, we need a new method that can count the exact time (number of *ticks*) needed for completing each task, and design a supervisor to satisfy the bounded-time requirement. To address this issue, we adopt an idea similar to that in Zhang, Wang, et al. (2024) which synthesizes quantitatively nonblocking supervisors where the transitions caused by all events are counted, but with a novel change that the new algorithm must distinguish *tick* and non-*tick* events and count only *tick*.

Remark 1. The property of bounded-time nonblockingness requires that each task must be completed within a bounded time from *any* state counted by the number of *ticks*. One may consider representing the requirement by a specification language which prohibits the occurrence of prohibitable events after bounded number of ticks, or restricting the number of ticks among the events leaving a marker state and those entering the marker states. Both of these methods cannot be easily extended to general cases, the reasons are as follows. First, in the concept of bounded-time nonblockingness, the time bounds on each marker state representing different tasks may be different, so we need to construct different TTGs for each marker state. Second, the leaving off and arriving at different marker states are represented by different events, and thus we need an extra algorithm to find and distinguish such events. Following the idea of representing the requirement of bounded-time nonblockingness by specification languages, we propose a language formula to represent the supremal bounded-time completable sublanguage for a given language K as in Theorem 3. By this theorem, for any given specification K, we may obtain its sublanguage satisfying the bounded-time completability.

3. Bounded-time nonblocking supervisory control problem of TDES

We start by introducing a new concept that *quantifies* the nonblocking property of a *tick*-automaton.

Let $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$ be a *tick*-automaton (modeling the TDES plant to be controlled) as in (1) and assume that $\mathbf{G}$ is nonblocking (i.e. every reachable state of $\mathbf{G}$ is also coreachable). Bring in a cover $Q_{\mathbf{G}}$ on the marker state set Q_m as follows:

$$Q_{\mathbf{G}} := \{Q_{m,i} \subseteq Q_m | i \in \mathcal{I}\}. \tag{3}$$

Here $\mathcal{I}$ is an index set, $Q_{m,i} \neq \emptyset$ for each $i \in \mathcal{I}$, and $\bigcup\{Q_{m,i}|i \in \mathcal{I}\} = Q_m$. This cover $Q_{\mathbf{G}}$ represents a classification of different types of marker states. For example, the three marker states 0, 4, 11 in Example 2.2 can be classified into two types: $Q_{m,1} = \{4, 11\}$ meaning completion of a package collecting/delivery task, whereas $Q_{m,2} = \{0\}$ meaning battery charging.

Fix $i \in \mathcal{I}$ and let $q \in Q \setminus Q_{m,i}$ be an arbitrary state in Q but not in $Q_{m,i}$. We define the set of all strings that lead q to $Q_{m,i}$ for the first time, namely

$$C(q, Q_{m,i}) :=\{s \in \Sigma^* | \delta(q, s)! \ \& \ \delta(q, s) \in Q_{m,i} \ \& \\ (\forall s' \in \overline{s} \setminus \{s\})\ \delta(x, s') \notin Q_{m,i}\}.$$

If $q \in Q_{m,i}$, we define $C(q, Q_{m,i}) := \{\epsilon\}$.

Now associate $Q_{m,i}$ with a finite positive integer N_i, and consider an arbitrary state in $q \in Q$. Denote by $\#s(tick)$ the number of *tick* occurred in string s. We say that state q is N_i*-tick coreachable* (wrt. $Q_{m,i}$) if

(i) $C(q, Q_{m,i}) \neq \emptyset$; and
(ii) $(\forall s \in C(q, Q_{m,i}))\ \#s(tick) \leq N_i$.

Condition (i) requires that there exists at least one string $s \in \Sigma^*$ leading q to a marker state in $Q_{m,i}$. Condition (ii) means that all strings that lead q to $Q_{m,i}$ for the first time include at most N_i *ticks*. Intuitively, condition (ii) means that in the worst case, it takes N_i *ticks* from state q to arrive a marker state in $Q_{m,i}$. Hence if $Q_{m,i}$ represents a task completion, then condition (ii) means that in the worst case, it takes N *ticks* from state q to complete the task.

Remark 2. We remark here that a *tick*-automaton $\mathbf{G}$ is *activity-loop-free* (Wonham & Cai, 2019) (i.e $(\forall q \in Q)(\forall s \in \Sigma_{act}^* \setminus \{\epsilon\})\delta(q,s) \neq q$). Thus all loops (the strings visiting a state repeatedly) in $\mathbf{G}$ include at least one *tick*. It is easily verified that all the *tick*-automata representing the sublanguages of $L_m(\mathbf{G})$ are also activity-loop-free, and thus the *tick*-automata mentioned in this paper are activity-loop free. ⋄

Now we introduce the new concept of bounded-time nonblockingness of a *tick*-automaton.

Definition 1. Let $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$ be a *tick*-automaton, $\mathcal{Q}_{\mathbf{G}} = \{Q_{m,i} | i \in \mathcal{I}\}$ a cover on Q_m as defined in (3), and N_i a positive integer associated with each $Q_{m,i} \in \mathcal{Q}_{\mathbf{G}}$. We say that $\mathbf{G}$ is *bounded-time nonblocking* wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$ if for every $i \in \mathcal{I}$ and every reachable state $q \in Q$, q is N_i-*tick* coreachable (wrt. $Q_{m,i}$).

In words, a bounded-time nonblocking *tick*-automaton requires that every state q can reach every subset $Q_{m,i}$ of marker states within N_i *ticks*. Compared with quantitatively nonblockingness of untimed automaton in Zhang, Wang, et al. (2024), bounded-time nonblockingness of *tick*-automaton is focused on the time for reaching marker states being bounded, rather than in Zhang, Wang, et al. (2024) that the total number of transitions (caused by any events) for reaching marker state are bounded.

Next we define the bounded-time nonblocking property of a supervisory control V for TDES. For this, we first introduce a new concept called *bounded-time completability*.

Let $K \subseteq L_m(\mathbf{G})$ be a sublanguage of $L_m(\mathbf{G})$. For each marker state subset $Q_{m,i} \in \mathcal{Q}_{\mathbf{G}}$ define

$$L_{m,i}(\mathbf{G}) := \{s \in L_m(\mathbf{G}) | \delta(q_0, s) \in Q_{m,i}\}$$

i.e. $L_{m,i}(\mathbf{G})$ represents the marked behavior of $\mathbf{G}$ wrt. $Q_{m,i}$. Then for each $i \in \mathcal{I}$, define

$$K_i := K \cap L_{m,i}(\mathbf{G}). \tag{4}$$

For an arbitrary string $s \in \overline{K} \setminus K_i$, define the set of strings that lead s to K_i for the first time:

$$M_{K,i}(s) := \{t \in \Sigma^* \mid st \in K_i (\forall t' \in \bar{t} \setminus \{t\}) st' \notin K_i\}. \tag{5}$$

If already $s \in K_i$, we define $M_{K,i}(s) := \{\epsilon\}$.

Definition 2. Let $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$ be a *tick*-automaton, $K \subseteq L_m(\mathbf{G})$ a sublanguage, $\mathcal{Q}_{\mathbf{G}} = \{Q_{m,i} | i \in \mathcal{I}\}$ a cover on Q_m as defined in (3), and N_i a positive integer associated with each $Q_{m,i} \in \mathcal{Q}_{\mathbf{G}}$. For a fixed $i \in \mathcal{I}$, we say that K is *bounded-time completable* wrt. $(Q_{m,i}, N_i)$ if for all $s \in \overline{K}$,

(i) $M_{K,i}(s) \neq \emptyset$;

(ii) $(\forall t \in M_{K,i}(s))\ \#t(tick) \leq N_i$.

Moreover if K is bounded-time completable wrt. $(Q_{m,i}, N_i)$ for all $i \in \mathcal{I}$, we say that K is *bounded-time completable* wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$.

If K is bounded-time completable wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$, then for every $i \in \mathcal{I}$, every string $s \in \overline{K}$ may be extended to a string in $K_i (= K \cap L_{m,i}(\mathbf{G}))$ by strings including at most N_i *ticks*. Compared with quantitative completability in Zhang, Wang, et al. (2024), the second condition is different: here all the strings in $M_{K,i}(s)$ are required to include at most N_i *ticks*, rather than have length no more than N_i in defining quantitative completability.

The following result characterizes the relation between bounded-time completability of a language and bounded-time nonblockingness of a *tick*-automaton.

Proposition 1. *Let $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$ be a tick-automaton, $K \subseteq L_m(\mathbf{G})$ a sublanguage, $\mathcal{Q}_{\mathbf{G}} = \{Q_{m,i} | i \in \mathcal{I}\}$ a cover on Q_m, and N_i a positive integer associated with each $Q_{m,i} \in \mathcal{Q}_{\mathbf{G}}$.*

(i) If $K = L_m(\mathbf{G})$ and $\mathbf{G}$ is bounded-time nonblocking wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$, then K is bounded-time completable wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$.

(ii) If $K \subseteq L_m(\mathbf{G})$ is bounded-time completable wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$, then there exists a tick automaton $\mathbf{K} = (X, \Sigma, \xi, x_0, X_m)$ such that $L_m(\mathbf{K}) = K$ and $\mathbf{K}$ is bounded-time nonblocking wrt. $\{(X_{m,i}, N_i) | i \in \mathcal{I}\}$, where $X_{m,i} = \{x_m \in X_m | (\exists s \in \Sigma^*) \xi(x_0, s) = x_m \ \&\ \delta(q_0, s) \in Q_{m,i}\}$.

The proof of Proposition 1 is similar to that of Proposition 4 in Zhang, Wang, et al. (2024). According to Proposition 1, for an arbitrary sublanguage $K \subseteq L_m(\mathbf{G})$ that is bounded-time completable wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$, we may construct a bounded-time nonblocking (wrt. $\{(X_{m,i}, N_i) | i \in \mathcal{I}\}$) *tick*-automaton $\mathbf{K}$ representing K, i.e. $L_m(\mathbf{K}) = K$.

With the above bounded-time completability of a language, we introduce the bounded-time nonblocking property of a supervisory control for TDES.

Definition 3. Let $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$ be a *tick*-automaton, $K \subseteq L_m(\mathbf{G})$ a sublanguage, $\mathcal{Q}_{\mathbf{G}} = \{Q_{m,i} | i \in \mathcal{I}\}$ a cover on Q_m as defined in (3), N_i a positive integer associated with each $Q_{m,i} \in \mathcal{Q}_{\mathbf{G}}$, and $V : L(\mathbf{G}) \to \Gamma$ a (marking) TDES supervisory control (wrt. $(K, \mathbf{G})$). We say that V is *bounded-time nonblocking* wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$ if

(i) V is nonblocking; and

(ii) $L_m(V/\mathbf{G})$ is bounded-time completable wrt.
$$\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}.$$

In words, quantitative nonblockingness of a TDES supervisory control V requires not only V being nonblocking (in the standard sense), but also the marked behavior $L_m(V/\mathbf{G})$ of the closed-loop system $V/\mathbf{G}$ being bounded-time completable. According to Proposition 1, $L_m(V/\mathbf{G})$ can be represented by a bounded-time nonblocking *tick*-automaton.

We are ready to formulate the *Bounded-Time Nonblocking Supervisory Control Problem* of TDES (*BTNSCP*):

Consider a TDES plant modeled by a tick-automaton $\mathbf{G} = (Q, \Sigma_c \dot{\cup} \Sigma_{uc}, \delta, q_0, Q_m)$, a specification language $E \subseteq \Sigma^*$, and let $K := E \cap L_m(\mathbf{G})$, $\mathcal{Q}_{\mathbf{G}} = \{Q_{m,i} \subseteq Q_m | i \in \mathcal{I}\}$ a cover on Q_m, and N_i a positive integer associated with each $Q_{m,i} \in \mathcal{Q}_{\mathbf{G}}$. Construct a (marking) TDES supervisory control $V : L(\mathbf{G}) \to \Gamma$ (for $(K, \mathbf{G})$) satisfying the following properties:

- **Safety**. *Marked behavior of the closed-loop system $V/\mathbf{G}$ satisfies the imposed specification E in the sense that $L_m(V/\mathbf{G}) \subseteq E \cap L_m(\mathbf{G})$.*
- **Bounded-time nonblockingness**.*TDES supervisory control V is bounded-time nonblocking wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$.*
- **Maximal permissiveness**.*TDES supervisory control V does not restrict more behavior than necessary to satisfy safety and bounded-time nonblockingness, i.e. for all other safe and bounded-time nonblocking TDES supervisory controls V' it holds that $L_m(V'/\mathbf{G}) \subseteq L_m(V/\mathbf{G})$.*

The BTNSCP is a variation of the traditional nonblocking supervisory control problem (Brandin & Wonham, 1994; Wonham & Cai, 2019) of TDES, in that the second requirement of bounded-time nonblockingness is stronger than the traditional nonblockingness. Namely, this problem cannot be solved in general by supervisors synthesized using the standard method.

4. Supremal bounded-time completable sublanguage and its computation

To solve the BTNSCP formulated in Section 3, we first present a basic result which is a counterpart to Theorem 1 in Brandin and Wonham (1994), Wonham and Cai (2019) and Theorem 6 in Zhang, Wang, et al. (2024).

Theorem 2. *Consider a TDES plant modeled by* tick*-automaton* $\mathbf{G} = (Q, \Sigma_c \dot{\cup} \Sigma_{uc}, \delta, q_0, Q_m)$*, a cover* $\mathcal{Q}_{\mathbf{G}} = \{Q_{m,i} \subseteq Q_m | i \in \mathcal{I}\}$ *on* Q_m*, and a positive integer* N_i *associated with each* $Q_{m,i} \in \mathcal{Q}_{\mathbf{G}}$*. Let* $K \subseteq L_m(\mathbf{G})$*,* $K \neq \emptyset$*. There exists a bounded-time nonblocking (marking) TDES supervisory control* V *(for* $(K, \mathbf{G})$*) such that* $L_m(V/\mathbf{G}) = K$ *if and only if* K *is controllable and bounded-time completable wrt.* $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$*. Moreover, if such a bounded-time nonblocking TDES supervisory control* V *exists, then it may be implemented by a bounded-time nonblocking* tick*-automaton* **QSUP**, *i.e.* $L_m(\mathbf{QSUP}) = L_m(V/\mathbf{G})$. ⋄

Theorem 2 asserts that when the K-synthesizing supervisory control V is required to be bounded-time nonblocking, it is necessary and sufficient to require that K be not only controllable but also bounded-time completable. If K is indeed controllable and bounded-time completable, then the TDES supervisory control V in Theorem 2 is the solution to the BTNSCP. If K is either not controllable or not bounded-time completable, then to achieve the third requirement of maximal permissiveness of BTNSCP, one hopes that the supremal controllable and bounded-time completable sublanguage of K exists. The key is to investigate if for bounded-time completability the supremal element also exists. We provide a positive answer below. Before we proceed, the following is a proof of Theorem 2.

Proof of Theorem 2.

We first prove the first statement. The direction of (only if) is a direct result from Theorem 1 and Definition 3. For the direction of (if), according to Theorem 1, since K is controllable, there exists a TDES supervisory control V such that V is nonblocking and $L_m(V/\mathbf{G}) = K$. Furthermore, according to Definition 3, it is derived from $L_m(V/\mathbf{G}) = K$ being bounded-time completable wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$ that V is bounded-time nonblocking wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$.

For the second statement, let V be a bounded-time nonblocking supervisory control that synthesizes a controllable and bounded-time completable K, i.e. $L_m(V/\mathbf{G}) = K$. Since K is bounded-time completable, it follows from Proposition 1 that there exists a bounded-time nonblocking *tick*-automaton **QSUP** such that $L_m(\mathbf{QSUP}) = K = L_m(V/\mathbf{G})$. This completes the proof. ■

4.1. Supremal bounded-time completable sublanguage

Let $\mathbf{G}$ be a nonblocking *tick*-automaton. First, we present the following proposition that bounded-time language completability is closed under arbitrary set unions.

Proposition 2. *Consider a* tick*-automaton* $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$*, a cover* $\mathcal{Q}_{\mathbf{G}} = \{Q_{m,i} \subseteq Q_m | i \in \mathcal{I}\}$ *on* Q_m*, and a positive integer* N_i *associated with each* $Q_{m,i} \in \mathcal{Q}_{\mathbf{G}}$*. Let* $K_1, K_2 \subseteq L_m(\mathbf{G})$*. If both* K_1 *and* K_2 *are bounded-time completable wrt.* $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$*, then* $K := K_1 \cup K_2$ *is also bounded-time completable wrt.* $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$*.*

Proof.

Let $s \in \overline{K}$ and $i \in \mathcal{I}$. According to Definition 2, to show that K is bounded-time completable, we need to show that (i) $M_{K,i}(s) \neq \emptyset$, i.e. there exists $t \in \Sigma^*$ such that $st \in K_i = K \cap L_{m,i}(\mathbf{G})$, and (ii) for all $t \in M_{K,i}(s)$, $\#t(tick) \leq N_i$. Since $\overline{K} = \overline{K_1 \cup K_2} = \overline{K_1} \cup \overline{K_2}$, either $s \in \overline{K_1}$ or $s \in \overline{K_2}$. We consider the case $s \in \overline{K_1}$; the other case is similar.

We first show that (i) holds. Since K_1 is bounded-time completable, $M_{K_1,i}(s) \neq \emptyset$, i.e. there exists string t such that $st \in K_1 \cap L_{m,i}(\mathbf{G}) \subseteq K \cap L_{m,i}(\mathbf{G})$. Thus (i) is established.

For (ii), let $t \in M_{K,i}(s)$; then $st \in K \cap L_{m,i}(\mathbf{G})$ and for all $t' \in \bar{t} \setminus \{t\}$, $st' \notin K \cap L_{m,i}(\mathbf{G})$. Since $K = K_1 \cup K_2$, there exist the following two cases: (a) $st \in K_1 \cap L_{m,i}(\mathbf{G})$ and for all $t' \in \bar{t} \setminus \{t\}$, $st' \notin K \cap L_{m,i}(\mathbf{G})$; (b) $st \in K_2 \cap L_{m,i}(\mathbf{G})$ and for all $t' \in \bar{t} \setminus \{t\}$, $st' \notin K \cap L_{m,i}(\mathbf{G})$. For case (a), it follows from $K \supseteq K_1$ that $st' \notin K_1 \cap L_{m,i}(\mathbf{G})$, so $t \in M_{K_1,i}(s)$. Since K_1 is bounded-time completable, it holds that $\#t(tick) \leq N_i$. The same conclusion holds for case (b) by a similar argument on K_2. Hence (ii) is established.

With (i) and (ii) as shown above, we conclude that K is bounded-time completable. ■

Now for a given sublanguage $K \subseteq L_m(\mathbf{G})$, whether or not K is bounded-time completable wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$, let

$$\mathcal{BTC}(K, \{(Q_{m,i}, N_i) | i \in \mathcal{I}\}) := \{K' \subseteq K \mid K' \text{is bounded-time completable wrt.} \{(Q_{m,i}, N_i) | i \in \mathcal{I}\}\}$$

represent the set of sublanguages of K that are bounded-time completable wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$. Note from Definition 2 that the empty language $\emptyset$ is trivially bounded-time completable, so $\emptyset \in \mathcal{BTC}(K, \{(Q_{m,i}, N_i) | i \in \mathcal{I}\})$ always holds. Moreover, it follows from Proposition 2 that there exists the supremal bounded-time completable sublanguage of K wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$, given by

$$\sup \mathcal{BTC}(K, \{(Q_{m,i}, N_i) | i \in \mathcal{I}\}) := \bigcup \{K' \mid K' \in \mathcal{BTC}(K, \{(Q_{m,i}, N_i) | i \in \mathcal{I}\})\}.$$

To compute this $\sup \mathcal{BTC}(K, \{(Q_{m,i}, N_i) | i \in \mathcal{I}\})$, we proceed as follows. Fix $i \in \mathcal{I}$ and let

$$\mathcal{BTC}(K, (Q_{m,i}, N_i)) := \{K' \subseteq K \mid K' \text{ is bounded-time completable wrt.} (Q_{m,i}, N_i)\}$$

be the set of all bounded-time completable sublanguage of K wrt. $(Q_{m,i}, N_i)$ (Definition 2). By the same reasoning as above, we have that $\sup \mathcal{BTC}(K, (Q_{m,i}, N_i))$ exists. The idea of our algorithm design is to first compute $\sup \mathcal{BTC}(K, (Q_{m,i}, N_i))$ for a fixed $i \in \mathcal{I}$, and then iterate over all $i \in \mathcal{I}$ until fixpoint in order to compute $\sup \mathcal{BTC}(K, \{(Q_{m,i}, N_i) | i \in \mathcal{I}\})$. In the next subsection, we present an automaton-based algorithm to compute $\sup \mathcal{BTC}(K, (Q_{m,i}, N_i))$ for any given language $K \subseteq L_m(\mathbf{G})$.

4.2. Automaton-based computation of $\sup \mathcal{BTC}(K, (Q_{m,i}, N_i))$

Consider a language $K \subseteq L_m(\mathbf{G})$ and $(Q_{m,i}, N_i)$ for a fixed $i \in \mathcal{I}$. In this subsection, we present a language formula for $\sup \mathcal{QC}(K, (Q_{m,i}, N_i))$.

To this end, we introduce several notation. For integer N_i, let $tick^{N_i}$ be the set of strings that have *tick* occurred no more than N_i times, i.e.

$$tick^{N_i} := \{t \in \Sigma^* | \ \#tick(t) \leq N_i\}$$

where $\#tick(t)$ represents the number of *tick* occurred in t. Next, for language $K \subseteq L_m(\mathbf{G})$, subset of marker states $Q_{m,i}$ and integer N_i, let $K_i := K \cap L_{m,i}(\mathbf{G})$ as defined in (4), and define

$$\widetilde{K_i} := \overline{K_i} \cap (tick^{N_i} \cup K_i tick^{N_i}) \tag{6}$$

where $K_i tick^{N_i} := \{st | s \in K_i \ \& \ t \in tick^{N_i}\}$. In simple words, $\widetilde{K_i}$ contains two subsets of $\overline{K_i}$: the first subset includes the strings that have number of *tick* no more than N_i. The second subset includes the strings each of which is a catenation of a string in K_i and a string having number of *tick* no more than N_i.

Now let

$$pre(\widetilde{K_i}) := \{s \in \Sigma^* | \bar{s} \subseteq \widetilde{K_i}\}. \tag{7}$$

It is not difficult to check that $pre(\widetilde{K_i})$ is prefix-closed, i.e. $pre(\widetilde{K_i}) = \overline{pre(\widetilde{K_i})}$. Based on (7), we can find all the prefixes of strings in $\overline{K_i}$ that lead a string from $\overline{K_i} \setminus K_i$ to K_i in no more than N_i steps. As will be confirmed by the following theorem, by the computation of $pre(\widetilde{K_i})$, we can find the supremal bounded-time completable sublanguage of K with respect to $(Q_{m,i}, N_i)$, i.e. $\sup \mathcal{BTC}(K, (Q_{m,i}, N_i))$.

Theorem 3. *Given a language* $K \subseteq L_m(\mathbf{G})$*, a subset* $Q_{m,i} \subseteq Q_m$ *of marker states and a positive integer* N_i*, let* $\widetilde{K_i}$ *and* $pre(\widetilde{K_i})$ *be the languages defined in (6) and (7) respectively. Then,*

$$\sup \mathcal{BTC}(K, (Q_{m,i}, N_i)) = pre(\widetilde{K_i}) \cap K. \tag{8}$$

Proof. For simplicity in notation, let $K' = pre(\widetilde{K_i}) \cap K$ in this proof. First, we prove that $K' \in \mathcal{BTC}(K, (Q_{m,i}, N_i))$. Since $K' = pre(\widetilde{K_i}) \cap K \subseteq K$ and the empty language is trivially bounded-time completable wrt. $(Q_{m,i}, N_i)$, we only need to show that when K' is nonempty, it is bounded-time completable wrt. $(Q_{m,i}, N_i)$.

Let $s \in \overline{K'}$; then there must exist a string $u \in \Sigma^*$ such that $su \in K' \subseteq K_i$, thus according to the definition of $M_{K',i}(s)$, $M_{K',i}(s) \neq \emptyset$. Let $t \in \Sigma^*$, and suppose $t \in M_{K',i}(s)$. According to Definition 2, to show that K' is bounded-time completable wrt. $(Q_{m,i}, N_i)$, we will show that $\#t(tick) \leq N_i$.

Since $s \in \overline{K'} \subseteq \overline{pre(\widetilde{K_i})} \cap \overline{K}$, we have $s \in \overline{pre(\widetilde{K_i})}$ and $s \in \overline{K}$. Since $t \in M_{K',i}(s)$, we have $st \in K' = pre(\widetilde{K_i}) \cap K$, i.e. $\overline{st} \subseteq \widetilde{K_i}$ and $st \in K$. By $\overline{st} \subseteq \widetilde{K_i}$, we have $st \in \widetilde{K_i}$, and for all prefix $t' \in \bar{t}$, $st' \in \widetilde{K_i}$. Also according to the definition of $M_{K',i}$, for all $t'' \in \bar{t} \setminus \{t\}$, $st'' \notin K'$. According to (6), $st \in \overline{K_i} \cap tick^{N_i}$ or $st \in \overline{K_i} \cap K_i tick^{N_i}$. In the former case, it holds that $\#t(tick) \leq N_i$ directly. In the latter case, if $s \in K_i$, then $t \in tick^{N_i}$; thus it also holds that $\#t(tick) \leq N_i$. It is left to consider the case that $s \notin K_i$ and $st \in \overline{K_i} \cap K_i tick^{N_i}$. In this case, there must exists $1 \leq k \leq N_i$ such that $s \in K_i tick^k$; namely, $t \in tick^{N_i - k}$, which derives that $\#t(tick) \leq N_i$. Thus, we conclude that $\#t(tick) \leq N_i$.

It remains to show that $K' = pre(\widetilde{K_i}) \cap K$ is the largest element in $\mathcal{BTC}(K, (Q_{m,i}, N_i))$. Let $M \in \mathcal{BTC}(K, (Q_{m,i}, N_i))$ be another element in $\mathcal{BTC}(K, (Q_{m,i}, N_i))$. It will be shown that $M \subseteq K'$. Namely, for any $s \in M$, we show that $s \in K'$.

Since $M \subseteq K$, we have $s \in K$ and thus $\bar{s} \subseteq \overline{K}$; we then need to prove that $\bar{s} \subseteq \overline{K_i} \cap (tick^{N_i} \cup K_i tick^{N_i})$. Let $\#s(tick) = k$ where $k \geq 0$, and write $s = u_0 ticku_1 ticku_2 ... ticku_k$. Since $s \in M \subseteq K$ and M is bounded-time completable wrt. $(Q_{m,i}, N_i)$, $\bar{s} \subseteq \overline{M \cap L_{m_i}(\mathbf{G})} \subseteq \overline{K \cap L_{m_i}(\mathbf{G})} = \overline{K_i}$. Thus, we show that $\bar{s} \subseteq tick^{N_i}$ or $\bar{s} \subseteq \overline{K_i tick^{N_i}}$. If $k \leq N_i$, it follows that $\bar{s} \subseteq tick^{N_i}$; if $k > N_i$, we prove that $\bar{s} \subseteq tick^{N_i} \cup K_i tick^{N_i}$.

First, we claim that there must exist a prefix s_0 of s such that $s_0 = u_0 ticku_1 ticku_2 ... ticku_{k_0}$ with $k_0 \leq N_i < k$ such that $s_0 \in M \cap L_{m,i}(\mathbf{G})$; otherwise string $s \in M_{M,i}(\epsilon)$, but $\#s(tick) > N_i$, which implies that M is not bounded-time completable wrt. $(Q_{m,i}, N_i)$ (hence a contradiction). By $s_0 \in M \cap L_{m,i}(\mathbf{G}) \subseteq K \cap L_{m,i}(\mathbf{G}) = K_i$, we have $s_0 \in K_i$ and thus $\overline{s_0} \subseteq \overline{K_i}$. Then, string s can be written as $s = s_0 u'_{k_0} ticku_{k_0+1} ... ticku_k$ where $u'_{k_0} \in \Sigma^*_{act}$ and satisfies that $s_0 u'_{k_0} tick$ is the prefix of s. Then (i) $k \leq k_0 + N_i$; or (ii) $k > k_0 + N_i$. In case (i), we have $s \in K_i tick^{N_i}$. Then since $\overline{s_0} \subseteq tick^{N_i}$, we have $\bar{s} = \overline{s_0} \cup s_0 u'_{k_0} ticku_{k_0+1} ... ticku_{k_0+k_1} \subseteq tick^{N_i} \cup K_i tick^{N_i}$.

In the later case for the same reason that M is bounded-time completable wrt. $(Q_{m,i}, N_i)$, there must exist a string s_1 such that $s_1 = s_0 u'_{k_0} ticku_{k_0+1} tick\ u_{k_0+2} ... ticku_{k_0+k_1}$ with $k_1 \leq N_i$, $s_1 \in M \cap L_{m,i}(\mathbf{G})$, and $s = s_1 u'_{k_0+k_1} ticku_{k_0+k_1+1}\ ... ticku_k$. Repeating the above process, since string s is finite, s can be written as $s = s_i u'_{k_0+k_1+\dots+k_i} ticku_{k_0+k_1+\dots+k_i+1} ... ticku_k$ with $i > 0$, and $k \leq k_0 + k_1 + \dots + k_i + N_i$, and by the same reason above, we have $s_i \in \overline{K_i}$ and $\bar{s} \subseteq tick^{N_i} \cup K_i tick^{N_i} \cup K_i tick^{N_i} = \bar{s} \subseteq tick^{N_i} \cup K_i tick^{N_i}$. Finally, we conclude that for all $s \in M$, we have $s \in K \cap pre(\widetilde{K_i})$. The proof is now complete. ■

By the above theorem, $\sup \mathcal{BTC}(K, (Q_{m,i}, N_i))$ can be expressed by the formula (8), and thus can be computed by the operations on languages (union, intersection, catenation) as expressed by formulas (4)–(8). In particular, (4), (6), and (8) can be implemented by the product of generators representing languages K, $L_{m,i}(\mathbf{G})$, $\overline{K_i}$, $tick^{N_i}$ and $K_i tick^{N_i}$, and (7) can be implemented by removing the non-marker states of the automaton representing $\widetilde{K_i}$ which in turn need generators representing languages $\overline{K_i}$, $tick^{N_i}$, and $K_i tick^{N_i}$. Thus the key is to construct two generators representing $tick^{N_i}$ and $K_i tick^{N_i}$, respectively (generators representing K, $L_{m,i}(\mathbf{G})$, and $\overline{K_i}$ are readily constructible).

First, for $tick^{N_i}$, we construct $\mathbf{A}_1 = (Y_1, \Sigma, \eta_1, y_{1,0}, Y_1)$ with $Y_1 = \{y_{1,0}, y_{1,1}, ..., y_{1,N_i}\}$, and $\eta_1(y_{1,i}, tick) = y_{1,i+1}$ for all $0 \leq i \leq N_i - 1$, and $\eta_1(y_{1,i}, \sigma) = y_{1,i}$ for all $\sigma \in \Sigma \setminus \{tick\}$ and $0 \leq i \leq N_i$. It is easily verified that $L_m(\mathbf{A}_1) = tick^{N_i}$.

Second, since $K_i \Sigma^{N_i}$ is the catenation of two languages K_i and Σ^{N_i}, a standard method in Hopcroft, Motwani, and Ullman (2014) is to first construct two generators $\mathbf{B}_1$ and $\mathbf{B}_2$ representing K_i and Σ^{N_i} respectively and then add ϵ-transitions between the marker states of $\mathbf{B}_1$ and the initial state of $\mathbf{B}_2$. However, this combined generator is non-deterministic, and transforming it into a deterministic generator is exponential in the state size of the combined generator in the worst case. More precisely, it is shown in Jirásek, Jirásková, and Szabari (2005), Yu, Zhuang, and Salomaa (1994) that the complexity of computing the catenation $K_i \Sigma^{N_i}$ is $O((2m - k)2^{n-1})$, where m and k are respectively the numbers of the states and marker states of $\mathbf{B}_1$, and n is the number of states of $\mathbf{B}_2$. Since $n = N_i + 1$ according to the construction of $\mathbf{A}_1$ above, the complexity of computing $K_i \Sigma^{N_i}$ is exponential in N_i. Hence, based purely on language operations, the complexity of computing $pre(\widetilde{K_i})$ and $\sup \mathcal{BTC}(K, (Q_{m,i}, N_i))$ is exponential in N_i.

Instead, we present in the following a polynomial algorithm (extension of Algorithm 2 (containing Algorithm 1) in Zhang, Wang, et al. (2024) for computing the supremal quantitatively completable sublanguage) to compute the supremal bounded-time completable sublanguage $\sup \mathcal{BTC}(K, (Q_{m,i}, N_i))$. The intuition is that we find for each prefix of $\overline{K}$ the bounded-time completable strings, and remove other non-bounded-time completable strings from the transition graph. The detailed steps are described in Algorithm 1. In the algorithm, we employ a last-in-first-out stack ST to store the states to be processed (a first-in-first-out queue can also be used instead to perform a different order of search), and for a set Z a flag $F : Z \to \{true, false\}$ to indicate whether or not an element of Z has been visited: i.e. $F(z) = true$ iff $z \in Z$ has been visited.

In Step 5.2 of Algorithm 1 above, note that the condition $d' > N_i$ means that the downstream transitions including more than N_i $ticks$ that have never reached a marker state in $Q_{m,i}$ will be removed, therefore guaranteeing that from an arbitrary state, at most N_i $ticks$ are needed to reach a marker state in $Q_{m,i}$.

The connections and differences between the above new Algorithm 1 and the Algorithm 2 (containing Algorithm 1) in Zhang, Wang, et al. (2024) for computing supremal quantitatively completable language are summarized in Table 2. In particular, there are two main differences. The first difference is the definition of X_i'. In the new algorithm, we have $X_i' := \{(x_i, d) | x_i \in X_i, d \in \{0, \dots, N_i\}\}$, namely, d is in the range of $[0, N_i]$, rather than $[0, N_i - 1]$ in the Algorithm 1 in Zhang, Wang, et al. (2024). The reason is illustrated in Fig. 5. Assume that the top subfigure represents partial transition functions of a $tick$-automaton $\mathbf{K}_i$ and let $N_i = 2$. In the process of constructing $\mathbf{K}_i'$ (as represented by the bottom subfigure) by Algorithm 1, from state (2,1) to state (5,2), $d = 1$ is advanced to $d' = 2$ since $\sigma = tick$. In the algorithm in Zhang, Wang, et al. (2024) of computing the supremal quantitatively completable sublanguage, since state 5 is not a marker state, this transition will be removed. However, since the downstream string $\alpha.\beta$ will lead the automaton to marker state 8 (where d' will be set to 0) and the total number of occurred $ticks$ is no more than 2 $ticks$, such states and transitions should be reserved. Thus in Algorithm 1, by Step 5.2 the states with $d' = N_i$ are included in $\mathbf{K}_i'$. Also by this step, the states with d' larger than 2 are removed from $\mathbf{K}_i$; e.g., transition $(6, tick, 7)$ is removed by Step 5.2 because now $d' = 3 > N_i$ (as represented by the dashed lines in the transition graph of $\mathbf{K}_i'$).

The second difference is the rules for updating d at Step 5.1: in Algorithm 1 above, only when $\sigma = tick$ will d' be set to $d+1$. The reason is that only the $tick$ transitions are counted for satisfying bounded-time completability; while in Algorithm 1 in Zhang, Wang, et al. (2024), the transitions caused by all events are counted. Namely, in the above new algorithm, the $tick$ and non-$tick$ events on the transitions need be distinguished for selecting different rules for updating d. However, in the algorithm in Zhang, Wang, et al. (2024), no such event distinguishing is needed.

Now we present an example to illustrate Algorithm 1.

[2] 'Trimmed' means that all non-reachable and non-coreachable states (if they exist) are removed Eilenberg (1974), Wonham and Cai (2019).

Table 2
Differences and connections between Algorithm 1 in this paper and Algorithm 2 (containing Algorithm 1) in Zhang, Wang, et al. (2024).

Differences & connections	Algorithm 1 in this paper	Algorithm 2 in Zhang, Wang, et al. (2024)
Input	*tick*-automaton **G** language K marker states subset $Q_{m,i}$ and positive integer N_i	Generator **G** language K marker states subset $Q_{m,i}$ and positive integer N_i
Output	Language K_i'	Language K_i'
Complexity	$O(\lvert Q\rvert \cdot \lvert X\rvert \cdot \lvert\Sigma\rvert \cdot N_i)$	$O(\lvert Q\rvert \cdot \lvert X\rvert \cdot \lvert\Sigma\rvert \cdot N_i)$
Range of d	$[0, N_i]$	$[0, N_i - 1]$
Condition of updating d to $d+1$	only when *tick* event occurs	when any event occurs

Algorithm 1 : Algorithm of Computing $\sup \mathcal{BTC}(K, (Q_{m,i}, N_i))$

```
Input: tick-automaton $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$, language $K \subseteq L_m(\mathbf{G})$, subset $Q_{m,i} \subseteq Q_m$ of marker states, and positive integer $N_i$.
Output: Language $K_i'$.

Step 1. Construct a tick-automaton $\mathbf{K_i} = (X_i, \Sigma, \xi_i, x_{i,0}, X_{i,m})$ to represent $K_i = K \cap L_{m,i}(\mathbf{G})$. If $K_i = \emptyset$, output language $K_i' = \emptyset$; otherwise go to Step 2.

Step 2. Let
    $X_i' := \{(x_i, d) | x_i \in X_i, d \in \{0, ..., N_i\}\}$,
    $\xi_i' = \emptyset$, $x_{i,0}' = (x_{i,0}, 0)$, and $X_{i,m}' := \{(x_i, 0) | x_i \in X_{i,m}\}$. Initially set $F((x_i, d)) = false$ for each state $x_i \in X_i$ and each $d \in \{0, ..., N_i - 1\}$. Then push the initial state $x_{i,0}' = (x_{i,0}, 0)$ into stack $ST$, and set $F((x_{i,0}, 0)) = true$.

Step 3. If stack $ST$ is empty, trim[2] the tick-automaton $\mathbf{K}_i' = (X_i', \Sigma, \xi_i', x_{i,0}', X_{i,m}')$, and go to Step 6. Otherwise, pop out the top element $(x_{i,j}, d)$ of stack $ST$. If $x_{i,j} \in X_{i,m}$, go to Step 4; otherwise, go to Step 5.

Step 4. For each event $\sigma \in \Sigma$ defined at state $x_{i,j}$ (i.e. $\xi_i(x_{i,j}, \sigma)!$), let $x_{i,k} := \xi_i(x_{i,j}, \sigma)$ and do the following two steps 4.1 and 4.2; then go to Step 3 with updated stack $ST$.
    Step 4.1 Add transition $((x_{i,j}, 0), \sigma, (x_{i,k}, 0))$ to $\xi_i'$, i.e.
        $\xi_i' := \xi_i' \cup \{((x_{i,j}, 0), \sigma, (x_{i,k}, 0))\}$.
    Step 4.2 If $F((x_{i,k}, 0)) = false$, push $(x_{i,k}, 0)$ into stack $ST$ and set $F((x_{i,k}, 0)) = true$.

Step 5. For each event $\sigma \in \Sigma$ defined at state $x_{i,j}$ (i.e. $\xi_i(x_{i,j}, \sigma)!$), do the following three steps 5.1–5.3; then go to Step 3 with updated stack $ST$.
    Step 5.1 Let $x_{i,k} := \xi_i(x_{i,j}, \sigma)$. If $\sigma = tick$, set $d' = d + 1$; If $\sigma \neq tick$ and $x_{i,k} \in X_{i,m}$, set $d' = 0$; if $\sigma \neq tick$ and $x_{i,k} \notin X_{i,m}$, set $d' = d$.
    Step 5.2 If $d' > N_i$, go to Step 5.1 with the next event $\sigma$ defined at $x_{i,j}$. Otherwise, add a new transition $((x_{i,j}, d), \sigma, (x_{i,k}, d'))$ to $\xi_i'$, i.e.
        $\xi_i' := \xi_i' \cup \{((x_{i,j}, d), \sigma, (x_{i,k}, d'))\}$
    Step 5.3 If $F((x_{i,k}, d')) = false$, push $(x_{i,k}, d')$ into stack $ST$ and set $F((x_{i,k}, d')) = true$.

Step 6. Output the language $K_i' = K \cap L(\mathbf{K}_i')$.
```

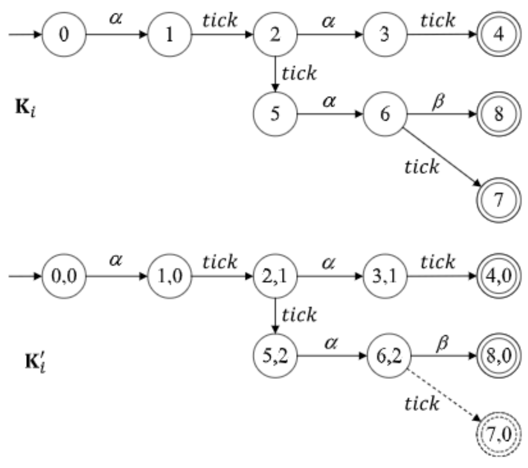

Fig. 5. Illustration on the reason for defining $d \in [0, N_i]$ in X_i'.

Example 2 (*Continuing Example 2.2*). Inputting *tick*-automaton **G**, language $K = L_m(\mathbf{SUP})$, marker state subset $Q_{m,1} = \{4, 11\} \subseteq Q_m$ and positive integer $N_1 = 5$, Algorithm 1 generates a trimmed automaton $\mathbf{TSUP}_1$, as displayed in Fig. 6.

From Fig. 6, we observe that the algorithm will terminate the search before the sixth *tick* occurs. Every reachable states are guaranteed to reach the marker states (4,0) and (11,0) in at most 5 *ticks*. By contrast in **SUP** (displayed in Fig. 4), the strings $(tick)^5.1.tick.2$ will lead state 0 to the marker state (4,0); however, they include more than 5 *ticks*. Thus, in fact those strings are removed from **SUP** by Algorithm 1. Also, the states and the transitions that cannot reach marker state are removed by the trim operation at Step 3. ⋄

The correctness of Algorithm 1 is confirmed by the following theorem.

Theorem 4. *Given a tick-automaton* **G**, *a sublanguage* $K \subseteq L_m(\mathbf{G})$, *subset* $Q_{m,i} \subseteq Q_m$ *of marker states, and positive integer* N_i, *let* K_i' *be the language returned by Algorithm 1. Then* $K_i' = \sup \mathcal{BTC}(K, (Q_{m,i}, N_i))$.

Proof. First, we prove that $K_i' \in \mathcal{BTC}(K, (Q_{m,i}, N_i))$. We start by showing that $K_i' \subseteq K$, which can be directly obtained by Step 6.

Next we show that K_i' is bounded-time completable wrt. $(Q_{m,i}, N_i)$. For this, let $\mathbf{K}_i' = (X_i', \Sigma, \xi_i', x_{i,0}', X_{i,m}')$ be the *tick*-automaton generated by Step 3 of Algorithm 1. According to Step 6, it is easily verified that $\overline{K_i'} = L(\mathbf{K}_i')$ and $K_i' \cap L_{m,i}(\mathbf{G}) = L_m(\mathbf{K}_i') \cap L_{m,i}(\mathbf{G})$ (because $K_i' \cap L_{m,i}(\mathbf{G}) = K \cap L(\mathbf{K}_i') \cap L_{m,i}(\mathbf{G}) = L_m(\mathbf{K}_i) \cap L(\mathbf{K}_i') \cap L_{m,i}(\mathbf{G}) = L_m(\mathbf{K}_i') \cap L_{m,i}(\mathbf{G})$). Let $s \in \overline{K_i'}$, then $s \in \overline{K}$ and $s \in L(\mathbf{K}_i')$; then there exists $t \in \Sigma^*$ such that $st \in L_m(\mathbf{K}_i')$, i.e. $st \in K_i' \cap L_{m,i}(\mathbf{G})$. According to the definition of $M_{K_i',i}(s)$,

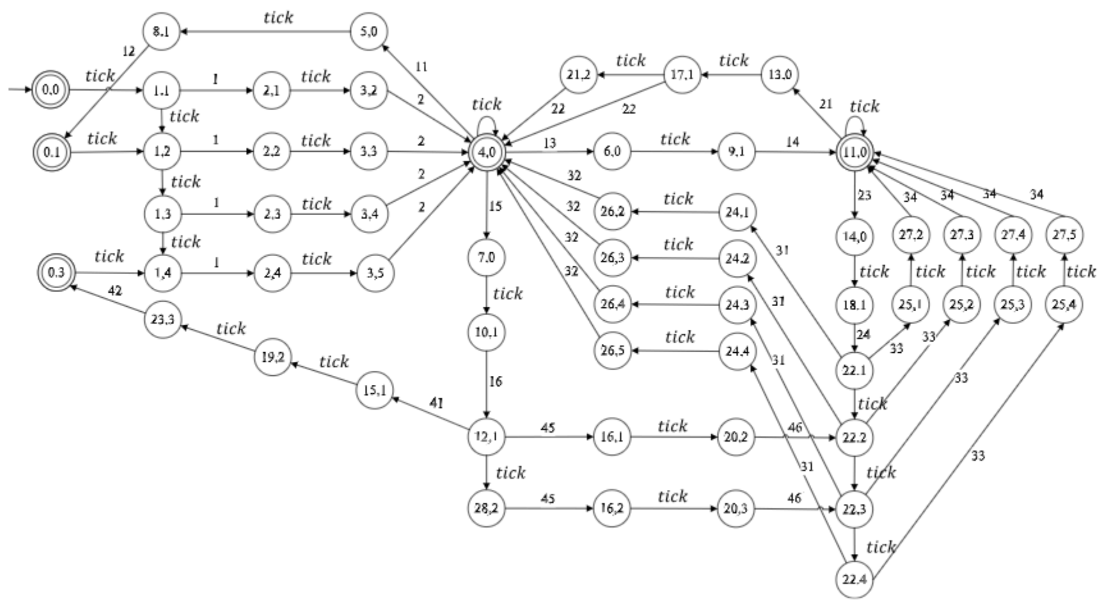

Fig. 6. Transition graph of $\mathbf{TSUP}_1$.

$M_{K'_i,i}(s) \neq \emptyset$; namely, condition (i) of bounded-time completability is satisfied. Furthermore, according to the definition of M in (5), it is derived from $K'_i \cap L_{m,i}(\mathbf{G}) = L_m(\mathbf{K}'_i) \cap L_{m,i}(\mathbf{G})$ that for each $t \in M_{K'_i,i}(s)$, $t \in M_{L_m(\mathbf{K}'_i),i}(s)$. It is guaranteed by $d' > N_i$ in Step 5.2 that if a string includes more than N_i *ticks* and has not reached a marker state in $Q_{m,i}$, it will not be added to $L(\mathbf{K}'_i)$. In other words, those strings st added to $L(\mathbf{K}'_i)$ must satisfy that for every $t \in M_{L_m(\mathbf{K}'_i),i}(s)$, there holds $\#t(tick) \leq N_i$. Namely, condition (ii) of bounded-time completability is also satisfied. Hence, K'_i is bounded-time completable wrt. $(Q_{m,i}, N_i)$. This establishes that $K'_i \in \mathcal{BTC}(K,(Q_{m,i},N_i))$.

It remains to show that K'_i is the largest element in $\mathcal{BTC}(K,(Q_{m,i},N_i))$. Let M be another element in $\mathcal{BTC}(K,(Q_{m,i},N_i))$, i.e. $M \in \mathcal{BTC}(K,(Q_{m,i},N_i))$. It will be shown that $M \subseteq K'_i$. For this, we first prove that $\overline{M} \subseteq \overline{K'_i} = L(\mathbf{K}'_i)$ by induction on the length of a string $s \in \overline{M}$.

Base case: Let $s = \epsilon \in \overline{M}$. Then $\epsilon \in \overline{K}$ and the initial state $x_{i,0}$ exists in $\mathbf{K}_i$. It follows from Step 2 that $x'_{i,0} = (x_{i,0}, 0)$ is designated to be the initial state of $\mathbf{K}'_i$, and hence $\epsilon \in L(\mathbf{K}'_i)$.

Inductive case: Let $s \in \overline{M}$, $s \in L(\mathbf{K}'_i)$, $\sigma \in \Sigma$, and suppose that $s\sigma \in \overline{M}$; we will show that $s\sigma \in L(\mathbf{K}'_i)$ as well. Since $M \in \mathcal{BTC}(K,(Q_{m,i},N_i))$, we have (i) $M \subseteq K$ and (ii) M is bounded-time completable wrt $(Q_{m,i},N_i)$. It follows from (i) that $s \in \overline{K} \cap L(\mathbf{G}) = L(\mathbf{K}_i)$, i.e. $\xi_i(x_{i,0}, s)!$. By the same reason, $\xi_i(x_{i,0}, s\sigma)!$. Letting $x_i = \xi_i(x_{i,0}, s)$, we derive $\xi_i(x_i, \sigma)!$. Since $s \in L(\mathbf{K}'_i)$, $\xi'_i((x_{i,0}, 0), s)!$. According to the definition of ξ'_i, there must exist $d \in \{0, \dots, N_i\}$ such that $(x_i, d) = \xi'_i((x_{i,0}, 0), s)$. We already know that $\xi_i(x_i, \sigma)!$. If $x_i \in X_{i,m}$, according to Step 4, $\xi'_i((x_i, d), \sigma)$ with $d = 0$ is defined. If $x_i \notin X_{i,m}$, according to Step 5.1, there may exist the following three cases: (a) $\sigma = tick$; (b) $\sigma \neq tick$ and $\xi_i(x_i, \sigma) \in X_{i,m}$; (c) $\sigma \neq tick$ and $\xi_i(x_i, \sigma) \notin X_{i,m}$. First, in case (a), $d' = d + 1$. According to Step 5.2, $d' > N_i$ or $d' \leq N_i$. The former is impossible because in that case $s \notin M$ (since $x_i \notin X_{i,m}$, $s \notin K$ or $s \notin L_{m,i}(\mathbf{G})$; in both cases, $s \notin M$) but $s.tick \in \overline{M}$ will imply that it needs at least $N_i + 1$ *ticks* to lead $\overline{M}$ to M, which contradicts to the assumption that M is bounded-time completable $(Q_{m,i}, N_i)$. When $d' \leq N_i$, $\xi'((x,d),\sigma)$ is defined with $d' \leq N_i$. For case (b), $d' = 0$, according to Step 5.2, $\xi'((x,d),\sigma)$ is defined with $d' = 0$. For case (c), $\xi'((x,d),\sigma)$ is defined with $d \leq N$ and thus $d' = d \leq N_i$. Hence, we conclude that $\xi'_i((x_i,d),\sigma)$ is defined, i.e. $s\sigma \in L(\mathbf{K}'_i)$.

Therefore, by the above induction, $\overline{M} \subseteq L(\mathbf{K}'_i)$ is established. So, we have

$$\begin{aligned} M &\subseteq \overline{M} \cap K \quad (\text{by } M \in \mathcal{BTC}(K,(Q_{m,i},N_i))) \\ &\subseteq L(\mathbf{K}'_i) \cap K \quad (\text{according to Step 6}) \\ &\subseteq K'_i. \end{aligned}$$

The proof is now complete. ■

The above theorem confirms that Algorithm 1 computes the supremal bounded-time completable sublanguage $\sup \mathcal{BTC}(K,(Q_{m,i},N_i))$. The time complexity of Algorithm 1 is $O(|Q| \cdot |X| \cdot |\Sigma| \cdot N_i)$ where $|Q|$ and $|X|$ are the states number of $\mathbf{G}$ and *tick*-automaton representing K respectively. This complexity is derived according to Steps 3 to 5, because $\mathbf{K}_i$ has at most $|Q| \cdot |X| \cdot |\Sigma|$ transitions and each transitions are visited at most N_i times.

In this section, we have presented a language formula and an automaton-based algorithm to compute the supremal bounded-time completable sublanguage of a given language (or specification). However, since the controllable sublanguage of a bounded-time completable language may not be bounded-time completable, to solve BTNSCP, we need an algorithm to compute the supremal controllable and bounded-time completable sublanguage.

5. Maximally permissive bounded-time nonblocking supervisory control

In this section, we present our solution to the BTNSCP. Consider a TDES plant modeled by *tick*-automaton $\mathbf{G} = (Q, \Sigma_c \dot\cup \Sigma_{uc}, \delta, q_0, Q_m)$, and a specification language $E \subseteq \Sigma^*$. Let $K := E \cap L_m(\mathbf{G})$, $\mathcal{Q}_{\mathbf{G}} = \{Q_{m,i} | i \in \mathcal{I}\}$ be a cover on Q_m, and a positive integer N_i associated with each $Q_{m,i} \in \mathcal{Q}_{\mathbf{G}}$.

Whether or not K is controllable and bounded-time completable, let $\mathcal{CBTC}(K, \{(Q_{m,i}, N_i) | i \in \mathcal{I}\})$ be the set of sublanguages of K that are both controllable and bounded-time completable wrt. $\{(Q_{m,i}, N_i) | i \in \mathcal{I}\}$, i.e.

$$\begin{aligned} \mathcal{CBTC}(K, \{(Q_{m,i}, N_i) | i \in \mathcal{I}\}) := \{K' \subseteq K \mid K' \text{ is} \\ \text{controllable and bounded-time completable wrt.} \\ \{(Q_{m,i}, N_i) | i \in \mathcal{I}\}\}. \end{aligned}$$

Since the empty language $\emptyset$ is trivially controllable and bounded-time completable, the set $\mathcal{CBTC}(K, \{(Q_{m,i}, N_i) | i \in \mathcal{I}\})$ is nonempty. Moreover, since both controllability and bounded-time completability are closed under arbitrary set unions, $\mathcal{CBTC}(K, \{(Q_{m,i}, N_i) | i \in \mathcal{I}\})$ contains a unique supremal element given by

$$\sup \mathcal{CBTC}(K, \{(Q_{m,i}, N_i) | i \in \mathcal{I}\}) := \bigcup \{K' \subseteq K \mid K' \in$$

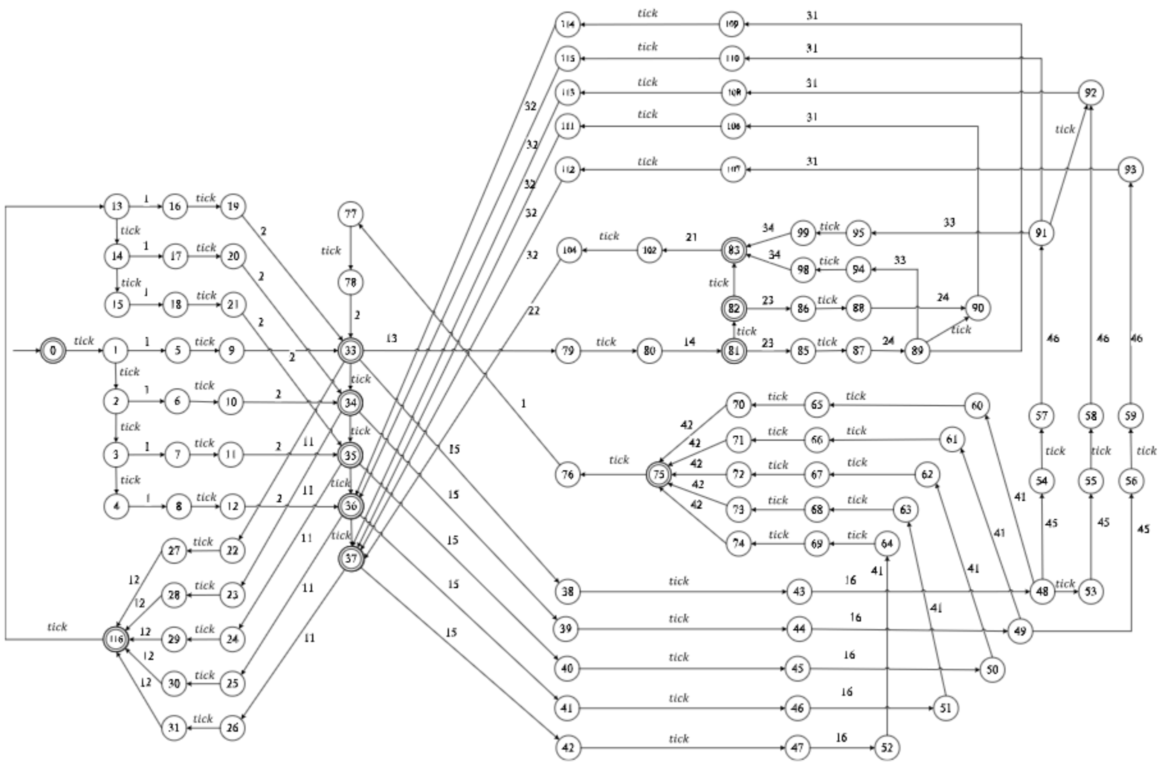

Fig. 7. Transition graph of **NK**.

$\mathcal{CBTC}(K, \{(Q_{m,i}, N_i)|i \in \mathcal{I}\})$.

Our main result in this section is the following.

Theorem 5. *Suppose that* $\sup \mathcal{CBTC}(K, \{(Q_{m,i}, N_i)|i \in \mathcal{I}\}) \neq \emptyset$. *Then the supervisory control* V_{sup} *such that* $L_m(V_{\text{sup}}/\mathbf{G}) = \sup \mathcal{CBTC}(K, \{(Q_{m,i}, N_i)| i \in \mathcal{I}\}) \subseteq K$ *is the solution to the BTNSCP.*

The proof of the above result is similar to that of Theorem 10 in Zhang, Wang, et al. (2024) with $\sup \mathcal{CQC}(K, \{(Q_{m,i}, N_i)|i \in \mathcal{I}\})$ replaced by $\sup \mathcal{CBTC}(K, \{(Q_{m,i}, N_i)|i \in \mathcal{I}\})$.

We proceed to design an algorithm (adapted from Algorithm 3 in Zhang, Wang, et al. (2024) for computing the supremal controllable and quantitatively completable sublanguage) to compute this solution $\sup \mathcal{CBTC}(K, \{(Q_{m,i}, N_i)|i \in \mathcal{I}\})$. The change is that the algorithm for computing the supremal quantitatively completable sublanguage is replaced by Algorithm 1 in Section 4.2 for computing the supremal bounded-time completable sublanguage. For self-containedness, we present the algorithm below.

The correctness of Algorithm 2 is confirmed similar to that of Algorithm 3 in Zhang, Wang, et al. (2024) (the detailed proof is referred to Zhang, Wang, et al. (2024)).

Theorem 6. *Given a* $tick$*-automaton* $\mathbf{G}$, *a specification language* E, *let* $K := E \cap L_m(\mathbf{G})$, *a cover* $\mathcal{Q}_{\mathbf{G}} = \{Q_{m,i} \subseteq Q_m | i \in \mathcal{I}\}$ *on* Q_m, *and a set of positive integer* N_i *each associated with a* $Q_{m,i} \in \mathcal{Q}_{\mathbf{G}}$. *Then Algorithm 2 terminates in a finite number of steps and outputs a language* CTK *such that* $CTK = \sup \mathcal{CBTC}(K, \{(Q_{m,i}, N_i)|i \in \mathcal{I}\})$.

The complexity of one complete iteration over all $i \in \mathcal{I}$ in Step 2 of Algorithm 2 is $O(|Q| \cdot |X| \cdot |\Sigma| \cdot \prod_{i=1}^{|\mathcal{I}|} N_i)$. Since Algorithm SC does not increase the state/transition number of NK_M^j ($M = |\mathcal{I}|$), the complexity of each iteration including Steps 2 and 3 is again $O(|Q| \cdot |X| \cdot |\Sigma| \cdot \prod_{i=1}^{M} N_i)$. Finally since there can be at most $|Q| \cdot |X| \cdot |\Sigma| \cdot \prod_{i=1}^{M} N_i$ iterations, the overall time complexity of Algorithm 2 is $O\big((|Q| \cdot |X| \cdot |\Sigma| \cdot \prod_{i=1}^{M} N_i)^2\big)$.

The following example demonstrates how to synthesize supervisors satisfying both controllability and bounded-time completability.

Algorithm 2 : Algorithm of Computing $\sup \mathcal{CBTC}(K, \{(Q_{m,i}, N_i)|i \in \mathcal{I}\})$ $(\mathcal{I} = \{1, \dots, M\})$

Input: $tick$-automaton $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$, language $K \subseteq L_m(\mathbf{G})$, cover $\mathcal{Q}_{\mathbf{G}} = \{Q_{m,i}|i \in \mathcal{I}\}$ on marker state set Q_m, and set of positive integers $\{N_i|i \in \mathcal{I}\}$.
Output: Language CTK.

Step 1. Let $j = 1$ and $K^j = K$ (i.e. $K^1 = K$).
Step 2. Let $i = 1$. Let $K_i^j = K^j$.
Step 2.1 Apply Algorithm 1 with inputs $\mathbf{G}$, K_i^j, $Q_{m,i}$ and N_i, and obtain $NK_i^j = \sup \mathcal{BTC}(K_i^j, (Q_{m,i}, N_i))$.
Step 2.2 If $i < M$, let $K_{i+1}^j = NK_i^j$, advance i to $i + 1$ and go to Step 2.1; otherwise ($i = M$), go to Step 3.
Step 3. Apply Algorithm SC with inputs $\mathbf{G}$ and NK_M^j to compute K^{j+1} such that $K^{j+1} = \sup \mathcal{C}(NK_M^j)$.
Step 4. If $K^{j+1} = K^j$, output $CTK = K^{j+1}$. Otherwise, advance j to $j + 1$ and go to **Step 2.**

Example 3 (*Continuing Example 2.2*). Consider TDES plant modeled by $tick$-automaton $\mathbf{G}$ and nonblocking supervisor **SUP** displayed in Fig. 4. First, applying Algorithm 1 with inputs $\mathbf{G}$, $K = L_m(\mathbf{SUP})$, $\mathcal{Q}_{\mathbf{G}} = \{Q_{m,1} = \{4, 11\}, Q_{m,2} = \{0\}\}$ and $\{N_1 = 5,\ N_2 = 9\}$, we get the supremal bounded-time completable language NK (wrt. $\{(Q_{m,1} = \{4, 11\}, N_1 = 5), (Q_{m,2} = \{0\}, N_2 = 9)\}$) represented by $tick$-automaton **NK** displayed in Fig. 7. However, it is not controllable, because event $tick$ is disabled at state 102, but event 22 is not forcible, which cannot preempt the occurrence of $tick$. Next applying Algorithm SC with inputs $\mathbf{G}$ and NK (although state 83 is a marker state, $tick$ is preempted on it but after event 21 is disabled, none of the forcible events is eligible at state 83; thus states 83, 94, 95, 98, 99 and corresponding transition arriving them are removed from **NK**), we get an automaton **TQCSUP** as displayed Fig. 8.

It is verified that $L_m(\mathbf{TQCSUP})$ is both controllable and bounded-time completable wrt. $\{(Q_{m,1} = \{4, 11\}, N_1 = 5), (Q_{m,2} = \{0\}, N_2 =$

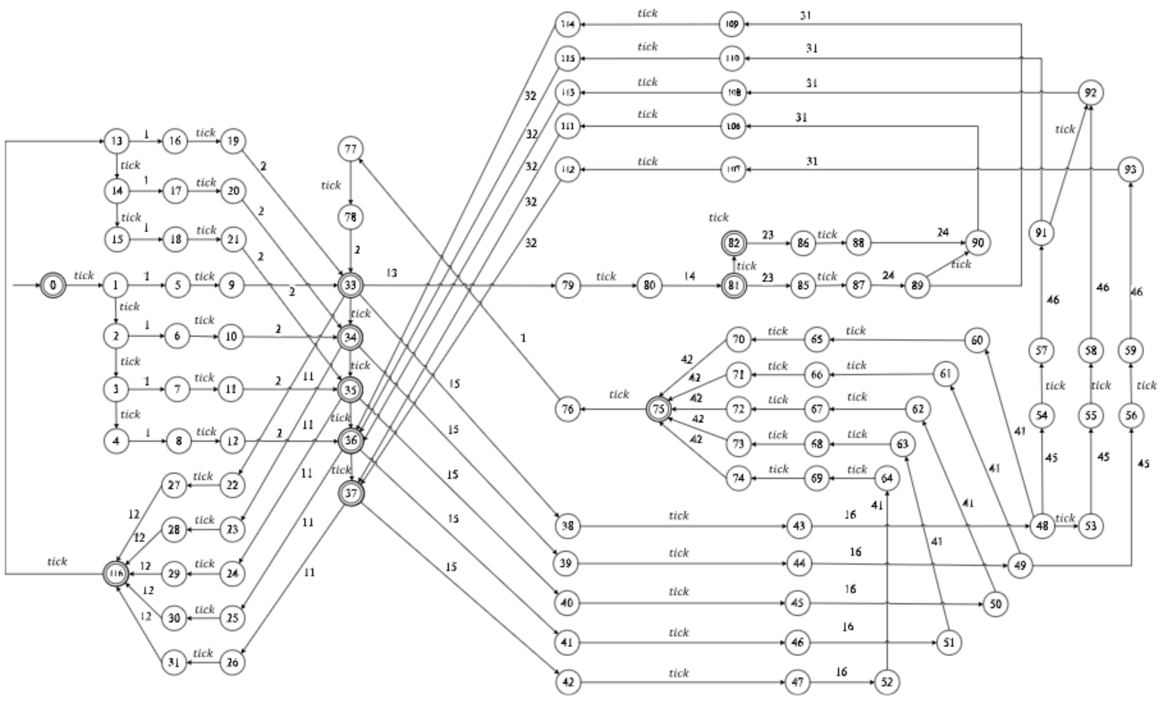

Fig. 8. Transition graph of **TQCSUP**.

9)}, and thus according to Theorem 2, **TQCSUP** may be used as a bounded-time nonblocking supervisor.

This supervisor **TQCSUP** is used to make the autonomous vehicle provide timely services in Example 1. The control logics of **TQCSUP** are as follows: (1) never move to zone 2 (corresponding to states 81 and 88) and zone 4 (corresponding to state 48, 49, 50, 51 and 52) when in zone 3 (corresponding to states 89, 90, 91 and 92); (ii) never move to zone 1 (corresponding to states 33, 34, 35, 36 and 37) when in zone 4; (iii) if the vehicle is in zone 1, it is safe to move to zone 2 and zone 4 if it has just returned from zone 0 (i.e. finished self-charging; corresponding to states 0, 75, 116); and (iv) if the vehicle has moved to zone 2 and 3, it must return (either by moving through zone 1, or moving though zone 4) for self-charging before the next round of service.

These logics guarantee that the two requirements ((i) and (ii) in Example 2.2 of Section 2.2) on the vehicle are satisfied. First, every package sent to customers can be delivered by the vehicle to one of the two service areas (zone 1 or 2) within 10 min; and whenever a customer calls for package collection, the vehicle can reach either zone 1 or 2 within 10 min no matter where the vehicle is and no matter which paths (permitted by the supervisor **TQCSUP**) the vehicle follows. Second, no matter where the vehicle is, it can return to zone 0 for self-charging within 18 min no matter which paths the vehicle follows.

6. Conclusion

In this paper, we have introduced a new concept of bounded-time nonblockingness of *tick*-automaton, which requires that every task (each represented by a subset of marker states) must be completed in prescribed time. Moreover, we have formulated a new bounded-time nonblocking supervisory control problem of TDES, characterized its solution in terms of bounded-time language completability, and developed algorithms to compute the optimal solution.

CRediT authorship contribution statement

Renyuan Zhang: Writing – review & editing, Writing – original draft, Supervision, Project administration, Formal analysis, Conceptualization. **Junhua Gou:** Writing – original draft. **Yabo Zhu:** Writing – original draft. **Bei Yang:** Writing – original draft. **Kai Cai:** Supervision, Formal analysis, Conceptualization.

Declaration of competing interest

We declare that we have no conflict of interest.

References

Balemi, S., Hoffmann, G., Gyugyi, P., Wong-Toi, H., & Franklin, G. (1993). Supervisory control of a rapid thermal multiprocessor. *IEEE Transactions on Automatic Control*, *38*(7), 1040–1059.

Berard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., et al. (2001). *Systems and software verification*. Springer-Verlag Berlin Heidelberg.

Bonakdarpour, B., & Kulkarni, S. (2006). *Complexity issues in automated addition of time-bounded liveness properties*: *Tech. rep.*, Department of Computer Science and Engineering, Michigan State University.

Brandin, B., & Charbonnier, F. (1994). The supervisory control of the automated manufacturing system of the AIP. In *Proc. rensselaer's 4th int. conf. computer integrated manufacturing and automation technology* (pp. 319–324).

Brandin, B., & Wonham, W. (1994). Supervisory control of timed discrete-event systems. *IEEE Transactions on Automatic Control*, *39*(2), 329–342.

Cai, K., & Wonham, W. (2020). Supervisory control of discrete-event systems. In *Encyclopedia of systems and control* (2nd ed.). Springer.

Cassandras, C., & Lafortune, S. (2008). *Introduction to discrete event systems* (2nd ed.). Springer.

Daws, C., Olivero, A., Tripakis, S., & SergioYovine (2002). Kronos: A verification tool for real-time systems. Available at https://www-verimag.imag.fr/DIST-TOOLS/TEMPO/kronos/.

Eilenberg, S. (1974). *Automata, languages and machines voluma A*. Academic Press.

Fabian, M., & Kumar, R. (1997). Mutually nonblocking supervisory control of discrete event systems. In *Proc. 36th IEEE conference on decision and control* (pp. 2970–2975).

Hopcroft, J., Motwani, R., & Ullman, J. (2014). *Introduction to automata theory languages and computation*. Pearson Education.

Jirásek, J., Jirásková, G., & Szabari, A. (2005). State complexity of concatenation and complementation of regular languages. *International Journal of Foundations of Computer Science*, *16*(3), 511–529.

Kumar, R., & Shayman, M. (1994). Non-blocking supervisory control of deterministic discrete event systems. In *Proc. 1994 American control conference* (pp. 1089–1093).

Ma, C., & Wonham, W. (2006). Nonblocking supervisory control of state tree structures. *IEEE Transactions on Automatic Control*, *51*(5), 782–793.

Malik, P. (2003). *From supervisory control to nonblocking controllers for discrete event systems* (Ph.D. thesis), University of Kaiserslautern.

Malik, R., & Leduc, R. (2008). Generalised nonblocking. In *Proc. 9th international workshop on discrete event systems* (pp. 340–345).

Ramadge, P., & Wonham, W. (1987). Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, *25*(1), 206–230.

Ramadge, P., & Wonham, W. (1989). The control of discrete event systems. *The Proceedings of IEEE*, *77*(1), 81–98.

Wonham, W. (2016). *Design software: TTCT*. Toronto, ON, Canada: Systems Control Group, ECE Dept., University of Toronto, Available at https://github.com/TCT-Wonham/TTCT.

Wonham, W., & Cai, K. (2019). *Supervisory control of discrete-event systems*. Springer.

Wonham, W., Cai, K., & Rudie, K. (2018). Supervisory control of discrete-event systems: a brief history. *Annual Reviews in Control*, *45*, 250–256.

Wonham, W., & Ramadge, P. (1987). On the supremal controllable sublanguage of a given language. *SIAM Journal on Control and Optimization*, *25*(3), 637–659.

Yu, S., Zhuang, Q., & Salomaa, K. (1994). The state complexities of some basic operations on regular languages. *Theoretical Computer Science*, *125*(2), 315–328.

Zhang, R., Wang, Z., & Cai, K. (2021). N-step nonblocking supervisory control of discrete-event systems. In *Proc. 2021 60th IEEE conference on decision and control* (pp. 339–344).

Zhang, R., Wang, J., Wang, Z., & Cai, K. (2024). Quantitatively nonblocking supervisory control of discrete-event systems. *Automatica*, *170*, Article 111879.

Zhang, Z., Xia, C., Chen, S., Yang, T., & Chen, Z. (2020). Reachability analysis of networked finite state machine with communication losses: a switched perspective. *IEEE Journal on Selected Areas in Communications*, *38*(5), 845–853.

Zhang, Z., Xia, C., Fu, J., & Chen, Z. (2022). Initial-state observability of mealy-based finite state machine with nondeterministic output functions. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *52*(10), 6396–6405.

Zhang, Z., Xia, C., G, Q., & Fu, J. (2024). Multi-step state-based opacity for unambiguous weighted machines. *Science China (Information Sciences)*, *67*(11), Article 212204.