Quantitative Nonblocking Supervisory Control of Discrete-Event Systems

Renyuan Zhang¹, Zenghui Wang¹ and Kai Cai²

Abstract

In this paper, we propose two new nonblocking properties of automata as quantitative measures of maximal distances to marker states. The first property, called quantitative nonblockingness, captures the practical requirement that at least one of the marker states (representing e.g. task completion) be reached within a prescribed number of steps. The second property, called heterogeneous quantitative nonblockingness, distinguishes individual marker states and requires that each marker state be reached within a given bounded number of steps. Accordingly, we formulate two new problems of quantitative nonblocking supervisory control and heterogeneous quantitative nonblocking supervisory control, and characterize their solvabilities in terms of new concepts of quantitative language completability and heterogeneous quantitative language completability respectively. It is proved that there exists the unique supremal (heterogeneous) quantitative completable sublanguage of a given language, and we develop generator-based algorithms to compute the supremal sublanguages. Finally, combining with the algorithm of computing the supremal controllable sublanguage, we design algorithms to compute the maximally permissive solutions to the formulated (heterogeneous) quantitative nonblocking supervisory control problems.

I. INTRODUCTION

In standard supervisory control of discrete-event systems (DES) [1-7], and other extensions and applications on nonblocking supervisory control, e.g. [8-14], 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 [4, 15]. Besides enforcing all imposed control specifications, a nonblocking supervisor ensures that every system trajectory can reach a marker

*This work was supported in part by the Natural Science Foundation of Shaanxi Province, China, Grant no. 2021JM-072, the Postdoctoral Science Foundation of China, Grant no. 2019M663819.

¹R. Zhang and Z. Wang are with the School of Automation, Northwestern Polytechnical University, Xi’an, China (ryzhang@nwpu.edu.cn).

²K. Cai is with the Department of Electrical and Information Engineering, Osaka City University, Osaka, Japan (kai.cai@eng.osaka-cu.ac.jp).
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.

While the nonblocking property is important, it only qualitatively guarantees finite reachability of marker states. There is no given bound on the number of steps for reaching marker states, so it can take an arbitrarily large (though finite) number of steps before a marker state is reached. Consequently, this qualitative nonblocking property might not be sufficient for many practical purposes, especially when there are prescribed bounds for reaching desired states. For example, a production cell [16] may be required not only to complete a task (e.g. transporting/processing a batch of workpieces) but also to do so within a prescribed number of operations; a warehouse AGV [17] is typically expected not only to return to a self-charging area but to do so periodically with a predetermined period (described by a number of events, each representing a movement from one area to the next); a communication protocol [18] is required not only to complete sending of a message and receiving of an acknowledgement, but also to do so in a bounded number of sending/recieving operations. In Section II below, we will present a detailed motivation example.

In this paper, we newly propose two quantitative nonblocking properties of an automaton to capture two different types of bound requirements in practice. The first type of bound requirement is that the automaton must reach (at least) one of the marker states of the automaton within a prescribed number of steps from an arbitrary (reachable) non-marker state. Roughly speaking, we measure the ‘maximal distance’ between non-marker states and marker states, and this is done by counting the number of events in every string leading a (reachable) non-marker state to a marker state. More specifically, let $N$ be a finite positive integer which denotes the required number of steps to reach a marker state; we define a quantitative nonblocking property (with respect to $N$) of an automaton that from every reachable state, all the strings that lead the state to a marker state have lengths smaller than or equal to $N$. That is, in the worst case, every reachable state can reach a marker state of the plant in no more than $N$ steps. If a marker state represents the completion of a task, this quantitative nonblocking property requires the automaton to be able to complete (any) ‘one of’ the tasks of the plant in no more than $N$ steps. In this sense, all tasks are treated the same.

The second type of bound requirement we consider is one that distinguishes individual tasks. In particular, every task is treated different, and the automaton is required to complete ‘every’ individual task in a prescribed number of steps. This means that the automaton must reach every marker state of the plant (representing a task) within a prescribed number of steps from any other state (either marked or not). Correspondingly, we define another heterogeneous quantitative nonblocking property of an automaton in which every marker state $q_{m,i}$ can be reached in no more than $N_i$ steps from any other (reachable) state.
(no matter it is marked or not). Here ‘heterogeneous’ refers to the treatment that each individual task is different.

With the above proposed two nonblocking properties, we first formulate a new quantitative nonblocking supervisory control problem (QNSCP) by requiring a supervisory control solution to be implementable by a quantitative nonblocking automaton. To solve this problem, we present a necessary and sufficient condition by identifying a new language property called quantitative completability. The latter roughly means that in the worst case, every string in the closure of a language can be extended to a string in the language in no more than $N$ steps. Further we show that this quantitative 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 QNSCP. Finally we design effective algorithms for the computation of such an optimal solution.

We then formulate another heterogeneous quantitative nonblocking supervisory control problem (HQNSCP) by requiring a supervisory control solution to be implementable by a heterogeneous quantitative nonblocking automaton. To solve this problem, we present another necessary and sufficient condition by identifying a language property called heterogeneous quantitative completability. Further we show that this heterogeneous quantitative language completability is also closed under arbitrary set unions, and together with language controllability which is also closed under unions, a maximally permissive solution exists for the formulated HQNSCP. Finally we design effective algorithms for the computation of such
an optimal solution.

The relationships among standard nonblockingness (corresponding to the standard nonblocking supervisory control problem (NSCP) [1-4]), quantitative nonblockingness (corresponding to QNSCP) and heterogeneous quantitative nonblockingness (corresponding to HQNSCP) are displayed in Fig. 1 and remarked in the sequel. First, the latter two quantitative nonblocking properties are both stronger than the standard (qualitative) nonblocking property, because they require that marker states must be visited in bounded number of steps. Second, the property of quantitative nonblockingness is suitable for situations where there is no distinction among the marker states, and it is not required which specified marker states the automaton must reach. For example, when there are multiple charging areas, a warehouse AGV can charge its battery by reaching any one of the charging areas in a prescribed (bounded) number of steps. Third, and in contrast, the property of heterogeneous quantitative nonblockingness is suitable for situations where every task is different, and each is required to be completed in a possibly different number of steps. Again taking the warehouse AGV as an example, the AGV may be required to transport two different types of items with different priorities to two corresponding destinations in different numbers of steps.

In the special case where the automaton has only a single marker state and the required bounds to reach this state are the same, quantitative nonblockingness and heterogeneous quantitative nonblockingness are identical. When there are more than one marker state, these two properties are generally different and one need not imply the other. On one hand, a quantitative nonblocking automaton can reach at least one of marker state \( q_i \in Q_m \) in bounded \( N \) steps, but is not guaranteed to reach every marker state \( q_i \) in bounded \( N_i \) steps; namely, the solution for QNSCP need not solve HQNSCP. On the other hand, a heterogeneous quantitative nonblocking automaton can reach every marker state \( q_i \) in bounded \( N_i \) steps. It can be seen that such an automaton is also quantitative nonblocking with respect to the minimum \( N_i \). However, as will be shown in Example 7 (Section IV.C), for the minimum bound \( N_i \), the automaton (modeling a plant) controlled by the optimal supervisor (a solution to HQNSCP) has strictly less permissive behavior than that controlled by the optimal supervisor constructed for solving QNSCP. Namely, the solution for HQNSCP also need not solve QNSCP. For these reasons, we will address these two properties and their corresponding supervisory control problems separately.

To our best knowledge, the concepts of quantitative nonblocking automaton, quantitative completable language, and quantitative nonblocking supervisory control problem and their heterogeneous counterparts are new in the literature. Previous work on weighted automata (e.g. [19, 20]), extended automata (e.g. [21, 22]), timed automata/DES (e.g. [23, 24]), and \( \omega \)-automata (e.g. [25, 26]) also considers quantification of various properties of automata; the quantification approaches are by means of introducing new features into the basic finite-state automata. In contrast, we work with the basic automata, and focus on the...
basic nonblocking property with a different purpose from \([19, 26]\). In addition, \(N\)-step opacity \([27, 28]\), diagnosability \([29]\), and detectability \([30, 31]\) are also studied in the literature; again, these work and ours are concerned with distinct properties and problems. Finally, \([11]\) introduced a concept of generalized nonblocking, which defines the coreachability between reachable states to subset of marker states representing particular properties; this generalized nonblockingness also does not consider the bound requirement on the number of steps reaching marker states.

The contributions of this paper are as follows.

- **Quantitative nonblocking supervisory control problem formulation, characterization of its solvability and algorithms for computing optimal solution (when it exists):** First, for a given positive integer \(N\), we propose a new property of quantitative nonblocking automaton to quantify the standard nonblocking property by capturing the practical requirement that (at least) one of the tasks can be completed within the bounded \(N\) steps. Second, we formulate a quantitative nonblocking supervisory control problem, and characterize its solvability by a new quantitative language completability in addition to language controllability. This problem and its solvability condition are generalizations of the standard supervisory control problem and solvability condition. Third, we prove that the language property of quantitative completability is closed under arbitrary set unions, and develop a generator-based algorithm to compute the supremal quantitative completable sublanguage. Forth, we develop a fixpoint algorithm to compute the supremal controllable and quantitative completable sublanguage of a given (specification) language, which synthesizes an optimal (maximally permissive) supervisory control solution for the quantitative nonblocking supervisory control problem.

- **Heterogeneous quantitative nonblocking supervisory control problem formulation, characterization of its solvability and algorithms for computing optimal solution (when it exists):** First, we propose another new property of heterogeneous quantitative nonblocking automaton to quantify the standard nonblocking property by capturing the requirement that every task can be completed within possibly different bounded steps. Second, we formulate a heterogeneous quantitative nonblocking supervisory control problem, and characterize its solvability by a new heterogeneous quantitative language completability. Third, we prove that the language property of heterogeneous quantitative completability is closed under arbitrary set unions, and develop a new algorithm to compute the supremal heterogeneous quantitative completable sublanguage. Finally, we design a fixpoint algorithm to compute the supremal controllable and heterogeneous quantitative completable sublanguage of a given (specification) language, which synthesizes an optimal (maximally permissive) supervisory control solution for the heterogeneous quantitative nonblocking supervisory control problem.
This paper distinguishes from its conference precursor [32] in the following aspects: (1) for the first part on quantitative nonblockingness, this paper provides all the proofs of formal results for QNSCP; (2) the second part on heterogeneous quantitative nonblockingness is completely new: this paper newly formulates HQNSCP, characterizes its solvability by the heterogeneous quantitative language completability and develops algorithms for computing the supremal solution.

The rest of this paper is organized as follows. Section II reviews the nonblocking supervisory control theory of DES and presents a motivating example for this work. Section III formulates the QNSCP, presents a necessary and sufficient condition for the solvability of QNSCP in terms of a new concept of quantitative language completability, and presents a solution to QNSCP. Section IV formulates the SQNSCP, presents a necessary and sufficient condition for the solvability of SQNSCP in terms of a new concept of strong quantitative language completability, and presents a solution to SQNSCP. Finally Section V states our conclusion and future work.

II. PRELIMINARIES AND MOTIVATING EXAMPLE

In this section, we review the standard nonblocking supervisory control theory of DES [1, 2, 4] and present a motivating example for our work.

A. Nonblocking Supervisory Control of DES

A DES plant is modeled by a generator (or automaton) [4]

\[ G = (Q, \Sigma, \delta, q_0, Q_m) \]  

where \( Q \) is the finite state set; \( q_0 \in Q \) is the initial state; \( Q_m \subseteq Q \) is the subset of marker states; \( \Sigma \) is the finite event set; \( \delta : Q \times \Sigma \rightarrow Q \) is the (partial) state transition function. Let \( \Sigma^* \) be the set of all finite-length strings of events in \( \Sigma \), including the empty string \( \epsilon \). In the usual way, \( \delta \) is extended to \( \delta : Q \times \Sigma^* \rightarrow Q \), and we write \( \delta(q, s)! \) to mean that \( \delta(q, s) \) is defined. The closed behavior of \( G \) is the language \( L(G) = \{ s \in \Sigma^* | \delta(q_0, s)! \} \subseteq \Sigma^* \) and the marked behavior is \( L_m(G) = \{ s \in L(G) | \delta(q_0, s) \in Q_m \} \subseteq L(G) \).

A string \( s_1 \) is a prefix of a string \( s \), written \( s_1 \leq s \), if there exists \( s_2 \) such that \( s_1s_2 = s \). For a string \( s \in \Sigma^* \), write \( \bar{s} := \{ s_1 \in \Sigma^* | s_1 \leq s \} \) for the set of all prefixes of \( s \). Note that \( \epsilon \) and \( s \) are members of \( \bar{s} \). For a language \( K \subseteq L_m(G) \), the (prefix) closure of \( K \) is \( \overline{K} := \{ s_1 \in \Sigma^* | (\exists s \in K) s_1 \leq s \} \). We say that \( K \) is closed if \( K = \overline{K} \).

For a generator \( G \) as in [1], a state \( q \in Q \) is reachable if there is a string \( s \in L(G) \) such that \( q = \delta(q_0, s) \); state \( q \in Q \) is coreachable [4] [15] if there is a string \( s \in \Sigma^* \) such that \( \delta(q, s)! \) and

\[ \text{In the following we will use “generator” and “automaton” interchangeably.} \]
We say that $G$ is nonblocking if every reachable state in $G$ is coreachable. In fact $G$ is nonblocking if and only if $\overline{\delta(q,s)} \subseteq Q_m$. For two generators $G_i = (Q_i, \Sigma_i, \delta_i, q_{i0}, Q_{i0})$, $i = 1, 2$, their product generator is defined as $G_1 \times G_2 = (Q, \Sigma, \delta, q_0, Q_m)$, where $Q = Q_1 \times Q_2$, $\delta = \delta_1 \times \delta_2$, $q_0 = (q_{01}, q_{02})$, and $Q_m = Q_{m1} \times Q_{m2}$, with $(\delta_2 \times \delta_2)((q_1, q_2), \gamma) := (\delta_1(q_1, \gamma), \delta(q_2, \gamma))$. For the control purpose, the event set $\Sigma$ is partitioned into $\Sigma_c$ (the subset of controllable events) and $\Sigma_{uc}$ (the subset of uncontrollable events), i.e. $\Sigma = \Sigma_c \cup \Sigma_{uc}$. A supervisory control for $G$ is any map $V : L(G) \rightarrow \Gamma$, where $\Gamma := \{\gamma \subseteq \Sigma \mid \gamma \supseteq \Sigma_{uc}\}$. Then the closed-loop system is denoted by $V/G$, with closed behavior $L(V/G)$ defined as follows:

\begin{enumerate}
\item $\epsilon \in L(V/G)$;
\item $s \in L(V/G) \& \sigma \in V(s) \& s\sigma \in L(G) \Rightarrow s\sigma \in L(V/G)$;
\item no other strings belong to $L(V/G)$.
\end{enumerate}

On the other hand, for any sublanguage $K \subseteq L_m(G)$, the closed-loop system’s marked behavior $L_m(V/G)$ is given by

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

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

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

A language $K \subseteq L_m(G)$ is controllable (w.r.t. $G$ and $\Sigma_{uc}$) if $K \Sigma_{uc} \cap L(G) \subseteq \overline{K}$. The following is a central result of nonblocking supervisory control theory.

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

Further, the property of language controllability is closed under set union. Hence for any language $K \subseteq L_m(G)$ (whether or not controllable), the set $C(K) = \{K' \subseteq K \mid \overline{K' \Sigma_{uc}} \cap L(G) \subseteq \overline{K}\}$. With this definition of $L_m(V/G)$, the supervisory control $V$ is also known as a marking supervisory control for $(K, G)$.
unique supremal element denoted by $\text{sup} \mathcal{C}(K)$ \[\text{[2, 4]}\]. Whenever $\text{sup} \mathcal{C}(K)$ is nonempty, by Theorem \[\text{[1]}\] there exists a nonblocking supervisory control $V_{\text{sup}}$ that satisfies $L_m(V_{\text{sup}}/G) = \text{sup} \mathcal{C}(K)$ and may be implemented by a nonblocking generator $\text{SUP}$ with $L_m(\text{SUP}) = L_m(V_{\text{sup}}/G)$.

B. Motivating Example

Nonblockingness of supervisory control $V$ describes a general requirement that every string generated by the closed-loop system $V/G$ can be completed to a marked string in finite but indefinite steps. However, in many real-world applications, it is often required that a task be completed in a prescribed, bounded number of steps. As an illustration, we present the following example.

Example 1: Consider an autonomous vehicle for package collecting and delivery in a region. The vehicle can move in six zones numbered 0–5, following the routes displayed on the top of Fig. 2. 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 0, 3, 4 and 5 are the storage areas for incoming and outgoing packages. Namely, the task of the vehicle is to send packages in the storage areas (zones 0, 3, 4 and 5) to the service areas (zones 0 and 1), and collect packages from the services areas and store them into the storage areas.

We model the movement of the autonomous vehicle by a generator $G$ with transition graph displayed on the top of Fig. 2. States 1 and 2 are chosen to be marker states, representing the completion of a received task. For simplicity (and no affect on the point we are illustrating), all events (i.e. movement of the vehicle) are assumed to be controllable.
First we consider an instance of standard nonblocking supervisory control. Suppose that due to road maintenance, the (directed) route

\[ \text{zone 0} \rightarrow \text{zone 5} \rightarrow \text{zone 4} \]

is not usable. This constraint is imposed as a specification. To satisfy this specification, a nonblocking supervisory control can be synthesized \([2, 4]\), and implemented by a nonblocking generator \(\text{SUP}\) as displayed on the bottom of Fig. 2. This \(\text{SUP}\) disables event 25 at state 0, event 15 at state 5, and event 17 at state 4. Moreover, since \(\text{SUP}\) is nonblocking, every non-marker state can reach marker states 1 and 2 in a finite number of steps.

Now consider two additional requirements that the customers need timely services:

(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 three steps (one step means the movement of vehicle from one zone to the next); and whenever a customer calls for package collection, the vehicle must reach either zone 1 or zone 2 within three steps.

(ii) Every package sent to a customer in zone 1 (resp. zone 2) must be delivered by the vehicle to zone 1 (resp. zone 2) within three (resp. four) steps; and whenever a customer in zone 1 (resp. zone 2) calls for package collection, the vehicle must reach zone 1 (resp. zone 2) within three (resp. four) steps.

The above two requirements, though appearing to be similar, are different. The first one requires that whichever zone the vehicle is in, it must be able to reach either one of the two service areas (marker states 1 or 2) in no longer than three steps (or transitions). On the other hand, the second one requires that whichever zone the vehicle is in, it must be able to reach both service areas (marker states 1 and 2) in no longer than three and four steps respectively. In the first requirement, zones 1 and 2 are not distinguished, and the customers in these two areas will be provided with the same timely service. In the second requirement, zones 1 and 2 are treated independently, and the customers in the two areas will receive different timely services.

The nonblocking supervisor \(\text{SUP}\) in Fig. 2 fails to fulfill neither of the above requirements, because if the vehicle is at zone 4, it is not guaranteed to return to zone 1 or zone 2 in three steps as it may move between zones 3 and 4 repeatedly. Thus we need new concepts and methods that can quantify the number of steps from a reachable state to any one of the marker states or every marker state, and design new supervisors to satisfy the quantitative requirement with respect to reaching marker states. This is the motivation for our study in this work.

\(\diamond\)

For a general DES plant modeled by \(G\), the requirements (i) and (ii) on the service vehicle in the
above example can be stated as follows:

(R1) Whichever state the plant is at, it must be able to reach any one of marker states in no longer than \( N \) steps (\( N \) is a prescribed positive integer).

(R2) Whichever state the plant is at, it must be able to reach each marker state \( q_{m,i} \) in no longer than \( N_i \) steps (\( N_i \) is a positive integer heterogeneously prescribed for \( q_{m,i} \)).

If \( G \) has a single marker state, i.e. \( |Q_m| = 1 \), and \( N = N_i \), then the above two requirements are identical. For the more general case where \( |Q_m| > 1 \), (R1) and (R2) are different, and one does not subsume the other. In the subsequent sections, we will accordingly formulate two different problems of synthesizing nonblocking supervisors satisfying the above two requirements respectively, and provide two corresponding solutions.

### III. Quantitative Nonblocking Supervisory Control Problem and Its Solution

#### A. Quantitative Nonblocking Supervisory Control Problem Formulation

We start by introducing a new concept that quantifies the nonblocking property of a generator.

Let \( G = (Q, \Sigma, \delta, q_0, Q_m) \) be a generator (modelling the plant to be controlled) as in (1) and assume that \( G \) is nonblocking (i.e. every reachable state of \( G \) is also coreachable). Consider a specification \( E \subseteq \Sigma^* \), and let \( K = (X, \Sigma, \xi, x_0, X_m) \) be a nonblocking generator representing the language \( E \cap L_m(G) \), i.e. \( L_m(K) = E \cap L_m(G) \) and \( \overline{L_m(K)} = L(K) \).

Let \( x \in X \setminus X_m \) be an arbitrary reachable state in \( X \). We define the set of strings that lead \( x \) to a marker state in \( X_m \) for the first time, namely \[
C_K(x) := \{ s \in \Sigma^* | \xi(x, s) ! \& \xi(x, s) \in X_m \& (\forall s' \in \overline{\{ s \}}) \xi(x, s') \notin X_m \}\]

If \( x \in X_m \), we define \( C_K(x) := \{ \epsilon \} \). Since \( K \) is nonblocking, by definition \( C_K(q) \) is nonempty for every \( x \in X \).

Let \( x \in X \) be an arbitrary state and \( N \) a finite positive integer. We say that state \( x \) is \( N \)-step coreachable if

\[
\begin{align*}
(i) & \quad x \text{ is coreachable;} \text{ and} \\
(ii) & \quad (\forall s \in C_K(x)) \ |s| \leq N.
\end{align*}
\]

Condition (i) holds under the assumption that \( K \) is nonblocking and ensures that \( C_K(q) \) is nonempty. Condition (ii) means that all strings that lead \( x \) to a marker state for the first time have length at most \( N \). Intuitively, one may think of the ‘maximum distance’ between \( x \) and \( X_m \) is bounded by \( N \). Since
Fig. 3. Transition graphs of $K_1$, $K_2$ and $K_3$

$L_m(K) \subseteq L_m(G)$, each marker state $x_m \in X_m$ corresponds to a marker state $q_m \in Q_m$, namely a string leading $K$ to $x_m$ will lead the plant to a marker state $q_m$. Moreover, since in requirement (R1) all the marker states are treated equal, if $Q_m$ represents task completion, $X_m$ will also represent task completion. Hence condition (ii) means that in the worst case, it takes $N$ steps from state $x$ to complete tasks.

Now we introduce the new concept of quantitative nonblockingness of a generator.

**Definition 1:** Let $K$ be a nonblocking generator and $N$ a finite positive integer. We say that $K$ is **quantitative nonblocking** wrt. $N$ if every state $x \in X$ is $N$-step coreachable.

Next we quantify the nonblocking property of a supervisory control $V$. For this, we first introduce a new concept called **quantitative language completability**.

Consider the marked behavior $K = L_m(K)$ of the nonblocking generator $K$. For an arbitrary string $s \in \overline{K} \setminus K$, define the set of strings that lead $s$ to a member in $K$ for the first time:

$$M_K(s) := \{ t \in \Sigma^* | st \in K (\forall t' \in \overline{t} \setminus \{t\}) st' \notin K \}.$$  \hspace{1cm} (3)

If $s \in K$, we define $M_K(s) := \{\epsilon\}$. Thus this definition of $M_K(s)$ is the language counterpart of $C_K(x)$ for generator.

**Definition 2:** Let $G$ be a nonblocking generator, $K \subseteq L_m(G)$ a sublanguage, and $N$ a finite positive
integer. We say that $K$ is \textit{quantitative completable} wrt. $N$ if

$$(\forall s \in K)(\forall t \in \Sigma^*) \ t \in M_K(s) \Rightarrow |t| \leq N.$$  

If $K$ is quantitative completable wrt. $N$, then every string $s \in K$ may be extended to a member of $K$ by strings of lengths at most $N$. We illustrate this definition by the following example.

\textit{Example 2 (Continuing Example 1):} Consider the generator $G$ in Example 1, and let $K_1, K_2, K_3 \subseteq L_m(G)$ be sublanguages as represented by generators $K_1, K_2$ and $K_3$ respectively (displayed in Fig. 3). That is, $L_m(K_i) = K_i, i = 1, 2, 3$.

It is easily verified that $K_1$ is quantitative completable wrt. $N = 1$: non-marker state 0 (resp. state 3) may reach marker state 1 (resp. state 2) by string 11 (resp. string 24) of length 1. However, $K_2$ is not quantitative completable wrt. any finite positive integer $N$, because for every $N$, at state 3 there exists string $t = (19.17)^N.24 \in M_{K_2}(11.13.23)$ that does not satisfy $|t| \leq N$. Finally, $K_3$ is quantitative completable wrt. $N = 4$, because starting from any non-marker state (0, 3, 4, or 5), the marker state 1 may be reached in at most four steps. \hfill \diamond

With the above quantitative language completability, we introduce the quantitative nonblocking property of a supervisory control.

\textit{Definition 3:} Let $G$ be a nonblocking generator, $K \subseteq L_m(G)$ a sublanguage, $V : L(G) \rightarrow \Gamma$ a (marking) supervisory control (for $(K, G)$), and $N$ a finite positive integer. We say that $V$ is \textit{quantitative nonblocking} wrt. $N$ if

(i) $V$ is nonblocking; and

(ii) $(\forall s \in L(V/G))(\forall t \in \Sigma^*) \ t \in M_{L_m(V/G)}(s) \Rightarrow |t| \leq N.$

Condition (i) means that $L_m(V/G) = L(V/G)$, while condition (ii) means that $L_m(V/G)$ is quantitative completable wrt. $N$. According to Definitions 1 and 2 a language $K$ is quantitative completable wrt. $N$ iff the generator $K$ representing $K$ is quantitative nonblocking wrt. $N$. Thus when the closed-loop system $V/G$ is represented by a (nonblocking) generator $CG$ (say), i.e. $L_m(CG) = L_m(V/G)$ and $L(CG) = L(V/G)$, if the supervisor $V$ is quantitative nonblocking wrt. $N$, then the generator $CG$ is quantitative nonblocking wrt. $N$. Namely, the closed-loop system $V/G$ has the property of quantitative nonblockingness, and thus satisfies the requirement (R1).

We are ready to formulate the \textit{Quantitative Nonblocking Supervisory Control Problem} of DES (QN-SCP).
Consider a DES plant modeled by a generator $G = (Q, \Sigma_c \cup \Sigma_{uc}, \delta, q_0, Q_m)$, a specification language $E \subseteq \Sigma^*$, and a finite positive integer $N$; let $K \subseteq E \cap L_m(G)$. Construct a (marking) supervisory control $V : L(G) \rightarrow \Gamma$ (for $(K, G)$) satisfying the following properties:

- **Safety.** Marked behavior of the closed-loop system $V/G$ satisfies the imposed specification $E$ in the sense that $L_m(V/G) = K \subseteq E \cap L_m(G)$.
- **Quantitative nonblockingness.** Supervisory control $V$ is quantitative nonblocking wrt. $N$.
- **Maximal permissiveness.** Supervisory control $V$ does not restrict more behavior than necessary to satisfy safety and quantitative nonblockingness, i.e. for all other supervisory controls $V'$ it holds that $L_m(V'/G) \subseteq L_m(V/G)$.

The QNSCP is a generalization of the traditional nonblocking supervisory control problem [1, 2, 4], in that the second requirement of quantitative nonblockingness is stronger than the traditional nonblockingness. This generalized problem cannot be solved in general by supervisors synthesized using the standard method; an example is SUP in Example 1. In subsequent subsections, we will develop new algorithms to design supervisors satisfying the new requirement of quantitative nonblockingness and resolving the QNSCP.

B. Supremal Quantitative Completable Sublanguage and Its Computation

Towards solving the QNSCP formulated in the preceding section, we first present a basic result which is a counterpart to Theorem 1.

**Theorem 2:** Let $K \subseteq L_m(G)$ ($K \neq \emptyset$) and $N$ be a finite positive integer. There exists a quantitative nonblocking (marking) supervisory control $V$ (for $(K, G)$) such that $L_m(V/G) = K$ if and only if $K$ is controllable and quantitatively completable wrt. $N$. Moreover, if such a quantitative nonblocking supervisory control $V$ exists, then it may be implemented by a quantitative nonblocking generator $QSUP$, i.e. $L_m(QSUP) = L_m(V/G)$.

Theorem 2 asserts that when the $K$-synthesizing supervisory control $V$ is required to be quantitative nonblocking, it is necessary and sufficient to require that $K$ be not only controllable but also quantitative completable. Such a $V$ may be realized by a generator that is not only nonblocking but quantitative nonblocking. This result extends the standard one of supervisory control theory (i.e. Theorem 1 [1, 2, 4]).

If $K$ is indeed controllable and quantitative completable, then the supervisory control $V$ in Theorem 2 is the solution to the QNSCP. If, however, $K$ is either not controllable or not quantitative completable, then to achieve the third requirement of maximal permissiveness of QNSCP, one would hope that the supremal controllable and quantitative completable sublanguage of $K$ exists. For controllability alone,
it is well-known that the supremal element exists. Hence the key is to investigate if for quantitative
completablility the supremal element also exists. We provide a positive answer in Section III.B.1 below.
Before we proceed, the following is a proof of Theorem 2.

**Proof of Theorem 2** The first statement (i.e. the necessary and sufficient condition) is a direct result
of Definitions 2, 3, and Theorem 1. For the second statement, let \( V \) be an quantitative nonblocking supervisory control that synthesizes a controllable and quantitative completable \( K \), i.e. \( L_m(V/G) = K \). From Theorem 1 we know that \( V \) may be implemented by a nonblocking generator \( QSUP = (X, \Sigma, \xi, x_0, X_m) \); that is \( L_m(QSUP) = L_m(V/G) = K \). Without loss of generality, consider that \( QSUP \) does not contain non-reachable states; if there were non-reachable states, simply remove them and this does not change \( L_m(QSUP) \). Let \( x \in X \) be an arbitrary (reachable) state; since \( QSUP \) is nonblocking, \( x \) is also coreachable. The reachability of \( x \) means that there is a string \( s \in L(QSUP) \) such that \( \xi(x_0, s)! \) and \( \xi(x_0, s) = x \). It follows from \( L(QSUP) = L_m(QSUP) = K \) that \( s \in K \). Since \( K \) is quantitative completable, for every \( t \in M_K(s) \) the length \( |t| \) satisfies \( |t| \leq N \). By the definition of \( M_K(s) \), this implies that every string that leads state \( x \) to a marker state in \( X_m \) has length no more than \( N \). Hence \( x \) is \( N \)-step coreachable. Finally by Definition 1 \( QSUP \) is a quantitative nonblocking generator. This completes the proof.

**B.1. Supremal Quantitative Completable Sublanguage**

Let \( G \) be a nonblocking generator. First, we present the following proposition that quantitative language
completablility is closed under arbitrary set unions.

**Proposition 1:** Let \( K_1, K_2 \subseteq L_m(G) \) and \( N \) be a positive integer. If both \( K_1 \) and \( K_2 \) are quantitative
completable wrt. \( N \), then \( K := K_1 \cup K_2 \) is also quantitative completable wrt. \( N \).

**Proof:** Let \( s \in K_1, t \in \Sigma^* \), and suppose \( t \in M_K(s) \). According to Definition 2 to show that \( K \) is quantitative completable wrt. \( N \), we will show that \( |t| \leq N \).

Since \( K = K_1 \cup K_2 = K_1 \cup K_2 \), either \( s \in K_1 \) or \( s \in K_2 \). We consider the case \( s \in K_1 \); the other
case is similar. It follows from \( t \in M_K(s) \) that \( st \in K \) and for every \( t' \in \hat{t} \setminus \{t\} \) it holds that \( st' \notin K \).
This implies that \( st' \notin K_1 \) and \( st' \notin K_2 \) for every \( t' \in \hat{t} \setminus \{t\} \); and either \( st \in K_1 \) or \( st \in K_2 \). Hence either \( t \in M_{K_1}(s) \) or \( t \in M_{K_2}(s) \). No matter which is the case, since both \( K_1 \) and \( K_2 \) are quantitative
completable wrt. \( N \), we derive \( |t| \leq N \).

Consider a language \( K \subseteq L_m(G) \) and a positive integer \( N \). Whether or not \( K \) is quantitative
completable, let

\[ QC(K, N) = \{ K' \subseteq K \mid K' \text{ is quantitative completable wrt. } N \} \]

represent the set of sublanguages of \( K \) that are quantitative completable wrt. \( N \). Note from Definition
that the empty language \( \emptyset \) is trivially quantitative completable, so \( \emptyset \in QC(K, N) \) always holds. This means that \( QC(K, N) \) is nonempty. Moreover by Proposition 1, there exists the suprimal quantitative completable sublanguage of \( K \), given by

\[
\sup QC(K, N) := \bigcup \{ K' \mid K' \in QC(K, N) \}.
\]

In the next subsection, we present a generator-based algorithm to compute \( \sup QC(K, N) \) for any given language \( K \subseteq L_m(G) \) and any finite positive integer \( N \).

**B.2. Generator-Based Computation of \( \sup QC(K, N) \)**

Consider a language \( K \subseteq L_m(G) \) and a positive integer \( N \). We present an algorithm to compute the suprimal quantitative completable sublanguage \( \sup QC(K, N) \). Let \( K \) be represented by a (nonblocking) generator \( K = (X, \Sigma, \xi, x_0, X_m) \), i.e. \( L_m(K) = K \) and \( L(K) = \overline{K} \). The intuition is that we find for each state of \( K \) the quantitative completable strings, and remove other non-quantitative completable strings from the transition graph. The detailed steps are described in Algorithm 1. In the algorithm, we employ a first-in-last-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 \( F: Z \to \{ \text{true}, \text{false} \} \) to indicate whether or not an element of \( Z \) has been visited: \( F(z) = \text{true} \) represents that \( z \in Z \) has been visited.

In Step 4.2 above, note that the condition \( d' = N \) means that the \( N \)-step downstream transitions that have never reached a marker state will be removed, therefore guaranteeing that from an arbitrary state, at most \( N \)-step transitions are needed to reach a marker state.

**Example 3 (Continuing Example 1):** Inputting \( L_m(SUP) \) and \( N = 3 \), Algorithm 1 outputs a new generator \( SUP' \), as displayed in Fig. 4. From the figure, we see that non-marker states 0, 3, 4, 5, 6, 7, 8, 9 are guaranteed to reach the marker states 1 or 2 in at most 3 steps. By contrast in \( SUP \) (displayed in Fig. 3), the strings \((19.17)^N.24\) with \( N \geq 3 \) will also lead unmarked state 3 to the marker state 2; however, their lengths are greater than 3. Thus, in fact those strings are removed from \( SUP \) by Algorithm 1.

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

**Theorem 3:** Given a language \( K \subseteq L_m(G) \) and a positive integer \( N \), and let \( K' \) be the generator returned by Algorithm 1. Then \( L_m(K') = \sup QC(K, N) \).

**Proof:** First, we prove that \( L_m(K') \in QC(K, N) \). We start by showing that \( L_m(K') \subseteq K \). The reasons are as follows. (i) In Steps 3 and 4, all the transitions added to \( K' \) correspond to those existing in \( K \), so \( L(K') \subseteq L(K) = \overline{K} \). (ii) Since marker states in \( K' \) are copied from \( K \) by \( X'_m := \{(x, 0)\mid x \in X_m\} \), \( L_m(K') \subseteq L(K') \cap L_m(K) = K \).

Next we show that \( L_m(K') \) is quantitative completable wrt. \( N \). It is guaranteed by \( d' = N \) in Step
Algorithm 1: Algorithm of Computing $\sup QC(K, N)$

\textbf{Input}: Generator $K = (X, \Sigma, \xi, x_0, X_m)$ representing $K$, and positive integer $N$.
\textbf{Output}: Generator $K' = (X', \Sigma, \xi', x_0', X_m')$.

\textbf{Step 1}. Let $X' := \{(x, d)| x \in X, d \in \{0, ..., N - 1\}\}$, $\xi' = \emptyset$, $x_0' = (x_0, 0)$, and $X_m' := \{(x, 0)| x \in X_m\}$. Initially set $F((x, d)) = false$ for each state $x \in X$ and each $d \in \{0, ..., N - 1\}$. Then push the initial state $x_0' = (x_0, 0)$ into stack $ST$, and set $F((x_0, 0)) = true$.

\textbf{Step 2}. If stack $ST$ is empty, output the trimmed generator $K' = (X', \Sigma, \xi', x_0', X_m')$. Otherwise, pop out the top element $(x_i, d)$ of stack $ST$. If $x_i \in X_m$, go to Step 3; otherwise, go to Step 4.

\textbf{Step 3}. For each event $\sigma \in \Sigma$ defined at state $x_i$ (i.e. $\xi(x_i, \sigma)$), let $x_j := \xi(x_i, \sigma)$ and do the following two steps 3.1 and 3.2.

\textbf{Step 3.1} Add transition $((x_i, 0), \sigma, (x_j, 0))$ to $\xi'$, i.e.

$\xi' := \xi' \cup \{(x_i, 0), \sigma, (x_j, 0)\}$.

\textbf{Step 3.2} If $F((x_j, 0)) = false$, push $(x_j, 0)$ into stack $ST$ and set $F((x_j, 0)) = true$.

\textbf{Step 3.3} Go to Step 2 with updated stack $ST$.

\textbf{Step 4}. For each event $\sigma \in \Sigma$ defined at state $x_i$ (i.e. $\xi(x_i, \sigma)$), let $x_j := \xi(x_i, \sigma)$ and do the following three steps 4.1–4.3.

\textbf{Step 4.1} If $x_j \in X_m$, set $d' = 0$; if $x_j \notin X_m$, set $d' = d + 1$.

\textbf{Step 4.2} If $d = N$, go to Step 4.1 with the next event $\sigma$ defined at $x_i$; if no more events are defined at $x_i$, go to Step 4.4. Otherwise, add a new transition $((x_i, d), \sigma, (x_j, d'))$ to $\xi'$, i.e.

$\xi' := \xi' \cup \{(x_i, d), \sigma, (x_j, d')\}$

\textbf{Step 4.3} If $F((x_j, d')) = false$, push $(x_j, d')$ into stack $ST$ and set $F((x_j, d')) = true$.

\textbf{Step 4.4} Go to Step 2 with updated stack $ST$.

4.2 that if a string is longer than $N$ and has never reached a marker state, then it will not be added to $L(K')$. Thus, those strings $s$ added to $L(K')$ must satisfy that for every $t \in M_{L_m(K')}(s)$, there holds $|t| \leq N$. Hence, $L_m(K')$ is quantitative completable wrt. $N$. This establishes $L_m(K') \in QC(K, N)$.

It remains to show that $L_m(K')$ is the largest element in $QC(K, N)$. Let $M \in QC(K, N)$ be another element in $QC(K, N)$. It will be shown that $M \subseteq L_m(K')$. For this, we first prove that $M \subseteq L(K')$ by induction on the length of a string $s \in M$.

\textbf{Base case}: Let $s = \epsilon \in M$. Then $\epsilon \in K$ and the initial state $x_0$ exists in $K$. It follows from Step 1 that $x_0' = (x_0, 0)$ is designated to be the initial state of $K'$, and hence $\epsilon \in L(K')$.

\textbf{Inductive case}: Let $s \in \overline{M}$, $s \in L(K')$, $\sigma \in \Sigma$, and suppose that $s\sigma \in \overline{M}$; we will show that $s\sigma \in L(K')$ as well. Since $M \in QC(K, N)$, we have (i) $M \subseteq K$ and (ii) $M$ is quantitative completable.

It follows from (i) that $s \in L(K)$, i.e. $\xi(x_0, s)$! By the same reason, $\xi(x_0, s\sigma)$!. Letting $x = \xi(x_0, s)$, we derive $\xi(x, \sigma)$!. Since $s \in L(K')$, $\xi'(x_0, 0, s)$!. According to the definition of $\xi'$, there must exist $d \in \{0, ..., N - 1\}$ such that $(x, d) = \xi'((x_0, 0), s)$. We already know that $\xi(x, \sigma)$!. If $x \in X_m$, according
to Step 3, $\xi'(x, d, \sigma)$ is defined where $d = 0$. If $x \notin X_m$, according to Step 4.1, if $x, \sigma \in X_m$, $\xi'(x, d, \sigma)$ is defined with $d' = 0$; otherwise $d' = d + 1$. For the latter case, it must hold that $d' \leq N - 1$ because $x, \sigma \notin X_m$ and $M$ is quantitative completable (otherwise there exists a string $s' \in M$ and $t \in M(s)$ such that $|t| = N + 1$). According to Step 4.2, in this case $\xi'(x, d, \sigma)$ is also defined. Hence, we conclude that $s, \sigma \in L(K')$.

Therefore, by the above induction, $\overline{M} \subseteq L(K')$ is established. Moreover we prove that $L(K') \cap L_m(K) \subseteq L_m(K')$. Let $s \in L(K')$ and $s \in L_m(K)$; then we have $\xi'((x_0, 0), s)!$ and $\xi(x, s) \in X_m$. According to Steps 3.1 and 4.1, when $\xi(x, s) \in X_m$, $d' = 0$. Since $X_m' = \{(x, 0) | x \in X_m\}$ and $\xi'((x_0, 0), s) \in X_m'$, we derive $s \in L_m(K')$. Finally,

$$M \subseteq \overline{M} \cap L_m(K) \subseteq L(K') \cap L_m(K) \subseteq L_m(K').$$

The proof is now complete.

The above theorem confirms that Algorithm 1 computes the supremal quantitative completable sublanguage $\sup QC(K, N)$. According to Steps 3 and 4, the time complexity of Algorithm 1 is $O(|X| \cdot |\Sigma| \cdot N)$.
where \(|X|\) is the state number of automaton \(K\), because \(K\) has at most \(|X| - |\Sigma|\) transitions and each transitions are visited at most \(N - 1\) times.

C. Maximally Permissive Quantitative Nonblocking Supervisory Control

In this subsection, we present our solution to the QNSCP. Consider a DES plant modeled by a generator \(G\) as in (1) with \(\Sigma = \Sigma_c \cup \Sigma_{uc}\), a specification language \(E \subseteq \Sigma^*\), and a finite positive integer \(N\). Let \(K := E \cap L_m(G)\).

Whether or not \(K\) is controllable and quantitative completable, let \(CQC(K, N)\) be the set of sublanguages of \(K\) that are both controllable and quantitative completable, i.e.

\[ CQC(K, N) := \{K' \subseteq K \mid K' \text{ is both controllable and quantitative completable wrt. } N\}. \]

Since the empty language \(\emptyset\) is trivially controllable and quantitative completable, the set \(CQC(K, N)\) is nonempty. Moreover, since both controllability and quantitative completability are closed under arbitrary set unions, \(CQC(K, N)\) contains a unique supremal element given by

\[ \sup CQC(K, N) := \bigcup \{K' \subseteq K \mid K' \in CQC(K, N)\}. \]

**Theorem 4:** Suppose that \(\sup CQC(K, N) \neq \emptyset\). Then the supervisory control \(V_{sup}\) such that \(L_m(V_{sup}/G) = \sup CQC(K, N) \subseteq K\) is the solution to the QNSCP.

**Proof:** Let \(\sup CQC(K, N)\) be nonempty. Since \(\sup CQC(K, N)\) is controllable and quantitative completable, according to Theorem 2 there exists a quantitative nonblocking supervisory control \(V_{sup}\) such that \(L_m(V_{sup}/G) = \sup CQC(K, N) \subseteq K\). Hence the safety and quantitative nonblockingness requirements of the QNSCP are satisfied. Further, since \(\sup CQC(K, N)\) is the supremal element in \(CQC(K, N)\), the maximal permissiveness requirement of the QNSCP is also satisfied. Therefore, \(V_{sup}\) that synthesizes \(\sup CQC(K, N)\) is the solution to the QNSCP. 

We proceed to design an algorithm to compute this solution \(\sup CQC(K, N)\). Since there exists a well-known algorithm to compute the supremal controllable sublanguage \([2, 4]\), and in the preceding section we designed an algorithm to compute the supremal quantitative completable sublanguage, a natural idea is to iterate these two algorithms until the fixed point is reached. Then the fixed point is the supremal controllable and quantitative completable sublanguage. This idea is implemented in Algorithm 3 below based on generator computation.

For reference, we call the known algorithm that computes the supremal controllable sublanguage Algorithm 2, whose inputs are generator \(G = (Q, \Sigma, \delta, q_0, Q_m)\) and specification language \(E\), whereas output is a nonblocking generator \(SUP\) with \(L_m(SUP) = \sup \mathcal{C}(E \cap L_m(G))\). The time complexity
of Algorithm 2 is known to be \( O(|Q| \cdot |E| \cdot |L| \cdot |\Sigma|) \), where \(|E| \cdot |L|\) denotes the state size of the canonical generator \(^4\) representing language \( E \).

**Algorithm 3**: Algorithm of Computing \( \sup \mathcal{QC}(K, N) \)

**Input**: Plant generator \( G = (Q, \Sigma, \delta, q_0, Q_m) \), specification language \( E \subseteq \Sigma^* \), and positive integer \( N \).

**Output**: Generator \( QSUP = (X, \Sigma, \delta, x_0, X_m) \).

**Step 1.** Let \( K := E \cap L_m(G) \). Construct a nonblocking generator \( K_0 \) such that \( L_m(K_0) = K \).

**Step 2.** For \( i \geq 0 \), apply Algorithm 1 (with inputs \( K_i \) and \( N \)) to compute \( H_i \) such that \( L_m(H_i) = \sup \mathcal{QC}(L_m(K_i), N) \).

**Step 3.** Apply Algorithm 2 (with inputs \( G \) and \( L_m(H_i) \)) to compute \( K_{i+1} \) such that \( L_m(K_{i+1}) = \sup \mathcal{C}(L_m(H_i)) \).

**Step 4.** If \( K_{i+1} = K_i \), then output \( QSUP = K_{i+1} \). Otherwise, advance \( i \) to \( i + 1 \) and go to **Step 2**.

The correctness of Algorithm \(^3\) is stated in the following theorem.

**Theorem 5**: Given a plant generator \( G \), a specification language \( E \), a positive integer \( N \), let \( K := E \cap L_m(G) \), and \( QSUP \) be the generator returned by Algorithm \(^3\) Then \( L_m(QSUP) = \sup \mathcal{QC}(K, N) \).

**Proof**: In Algorithm 3, Step 2 performs the computation of supremal quantitative completable sublanguage, and Step 3 supremal controllable sublanguage. The iteration of Steps 2 and 3 generates the following sequence of languages:

\[
K = L_m(K_0) \supseteq L_m(H_0) \supseteq L_m(K_1) \supseteq L_m(H_1) \supseteq \cdots
\]

Since both Algorithms 1 and 2 are finitely convergent, so is the above sequence. When the sequence converges, i.e. in Step 4 \( K_{i+1} = K_i \) holds for some \( i \), \( L_m(K_{i+1}) \) is the supremal controllable and quantitative completable sublanguage of \( K \). This proves that \( L_m(QSUP) = \sup \mathcal{QC}(K, N) \). \( \blacksquare \)

The above theorem confirms that Algorithm 3 computes the supremal controllable and quantitative completable sublanguage \( \sup \mathcal{QC}(K, N) \). For the complexity of Algorithm 3, then the state number of \( K_0 \) is estimated as \( O(|Q| \cdot |E| \cdot |L|) \). Thus Algorithm 1 used in Step 2 has complexity \( O(|Q| \cdot |E| \cdot |L| \cdot |\Sigma| \cdot N) \) and the states number of \( H_i \) is estimated as \( O(|Q| \cdot |E| \cdot |L| \cdot N) \). Furthermore, since \( H_i \) is the subautomaton of \( K_i \), the state size of the product of \( G \) and \( H_i \) is also \( O(|Q| \cdot |E| \cdot |L| \cdot N) \), thus Algorithm 2 used in Step 3 has complexity \( O(|Q| \cdot |E| \cdot |L| \cdot |\Sigma| \cdot N) \). Finally, since there can be at most \( O(|Q| \cdot |E| \cdot |L| \cdot |\Sigma| \cdot N) \) iterations of Steps 2 and 3, the time complexity of Algorithm 3 is \( O(|Q|^2 \cdot |E|^2 \cdot |\Sigma|^2 \cdot N^2) \).

**Example 4 (Continuing Example 1)**: For the plant generators \( G \) and \( SUP \) displayed in Fig. 2 we now assume that the odd numbers represent controllable events and even numbers represent uncontrollable events. Input \( G \), \( L_m(SUP) \), and \( N = 3 \) to Algorithm 3. In Step 1, \( K_0 := SUP \) (as displayed in Fig. 2). Then in Step 2, Algorithm 1 generates a generator \( H_0 = SUP' \) (displayed in Fig. 4). We have
verified that \( L_m(H_0) \) is quantitative completable wrt. \( N = 3 \). However, it is not controllable, because uncontrollable event 16 is disabled at state 4. Next in Step 3, Algorithm 2 generates a generator \( K_1 \) as displayed Fig. 5. It is easily verified that \( L_m(K_1) \) is both controllable and quantitative completable wrt. \( N = 3 \). This means that in Step 4, although \( K_1 \neq K_0 \) and Steps 2 and 3 need to be repeated, no further change will be made. It follows that \( K_2 = K_1 \), and Algorithm 3 outputs \( QSUP = K_1 \).

This \( QSUP \) may be used as a supervisor for the autonomous vehicle to provide timely services \((N = 3)\) to the customers in zones 1 and 2. Indeed, the only way to ensure this is to never move to zone 5 when in zone 0 (this is to satisfy the originally imposed specification which has already been enforced by \( SUP \)), and never move to zone 4 when in zone 3.

IV. HETEROGENEOUS QUANTITATIVE NONBLOCKING SUPERVISORY CONTROL PROBLEM AND ITS SOLUTION

The supervisor \( QSUP \) in Example 4 guarantees that the autonomous vehicle provides timely services as described by the requirement (i) in Example 1 (in Section II.B). However, it does not fulfil the requirement (ii) that both zone 1 and zone 2 need to be reached in finite number of steps. This is because if the vehicle is in zone 3, it may move between zones 2 and 3 repeatedly, and is not guaranteed to reach zone 1 in three steps.

In this section, to meet the general requirement (R2) in Section II.B (generalized version of requirement (ii) in Example 1), we formulate another heterogeneous quantitative nonblocking supervisory control problem \((HQNSCP)\). To solve this problem, we introduce another concept of heterogeneous quantitative nonblockingness, and present a new method to synthesize supervisors satisfying the requirement (R2).
A. Heterogeneous Quantitative Nonblocking Supervisory Control Problem Formulation

We first introduce another new concept, called heterogeneous quantitative nonblockingness, which quantifies the nonblocking property of a generator by differentiating its marker states.

Let $G = (Q, \Sigma, \delta, q_0, Q_m)$ be a generator (modeling the plant to be controlled) as in (1) and assume that $G$ is nonblocking. Consider a specification $E \subseteq \Sigma^*$, and let $K = (X, \Sigma, \xi, x_0, X_m)$ be a nonblocking generator representing the language $E \cap L_m(G)$, i.e. $L_m(K) = E \cap L_m(G)$ and $\overline{L_m(K)} = L(K)$. Define

$$Q_m(K) := \{q_m \in Q_m | (\exists s \in \Sigma^*) \xi(x_0, s) \in X_m \& \delta(q_0, s) = q_m\}. \quad (5)$$

Hence $Q_m(K) \subseteq Q_m$ is the subset of marker states of the plant $G$ which corresponds to the imposed specification. Note in general that not all marker states of the plant may be relevant to an imposed specification, and for the control purpose of fulfilling the imposed specification $K$, it suffices to consider those relevant marker states of $G$ in $Q_m(K)$. To rule out the trivial case, we assume that $Q_m(K) \neq \emptyset$.

Write $Q_m(K) = \{q_{m,i} | i \in I\}$ ($I$ is an index set). For each $q_{m,i} \in Q_m(K)$, define

$$Rch_K(q_{m,i}) := \{x_m \in X_m | (\exists s \in \Sigma^*) \xi(x_0, s) = x_m \& \delta(q_0, s) = q_{m,i}\};$$

thus $Rch_K(q_{m,i})$ includes all the marker states in $K$ representing the same marked behavior as $q_{m,i}$ in $G$. Note that $Rch_K(q_{m,i})$ is nonempty because $q_{m,i} \in Q_m(K)$. Let $x \in X \setminus Rch_K(q_{m,i})$ be an arbitrary state in $X$. We define the set of strings that lead $x$ to $Rch_K(q_{m,i})$ for the first time, namely

$$C_K(x, q_{m,i}) := \{s \in \Sigma^* | \xi(x, s)! \& \xi(x, s) \in Rch_K(q_{m,i}) \& \forall s' \in \pi \setminus \{s\} \xi(x, s') \notin Rch_K(q_{m,i})\}$$

If $x \in Rch_K(q_{m,i})$, we define $C_K(x, q_{m,i}) := \{\epsilon\}$.

Let $x \in X$ be an arbitrary state in $K$, $q_{m,i} \in Q_m(K)$ an arbitrary marker state in $G$, and $N_i$ a finite positive integer (which is $q_{m,i}$ specific). We say that state $x$ is $N_i$-step coreachable (wrt. $q_{m,i}$) if

(i) $C_K(x, q_{m,i}) \neq \emptyset$; and

(ii) $\forall s \in C_K(x, q_{m,i}) |s| \leq N_i$.

Condition (i) ensures that for each marker state $q_{m,i} \in Q_m(K)$, there exists at least a string $s \in \Sigma^*$ leading $x$ to a marker state in $Rch_K(q_{m,i})$. Note that by string $s$, the plant $G$ will visit marker state $q_{m,i}$. Condition (ii) means that all strings that lead $x$ to $Rch_K(q_{m,i})$ for the first time have length at most $N_i$.

Intuitively, condition (ii) means that in the worst case, it takes $N_i$ steps from state $x$ to complete the task.
represented by \( q_{m,i} \). Now we introduce the new concept of heterogeneous quantitative nonblockingness of a generator.

**Definition 4:** Let \( G \) be a nonblocking generator, \( K \) be a nonblocking generator with \( L_m(K) \subseteq L_m(G) \), \( Q_m(K) \subseteq Q_m \) a subset of marker states as defined in (5), and \( N_i \) a positive integer for each \( q_{m,i} \in Q_m(K) \). We say that \( K \) is **heterogeneous quantitative nonblocking** wrt. \( \{(q_{m,i}, N_i) | i \in I \} \) if for every marker state \( q_{m,i} \in Q_m(K) \) and every state \( x \in X \), \( x \) is \( N_i \)-step coreachable (wrt. \( q_{m,i} \)).

This definition distinguishes each individual marker state (in \( Q_m(K) \)), which is in contrast with Definition 1 of quantitative nonblocking generator where all marker states are treated the same.

Next we define the heterogeneous quantitative nonblocking property of a supervisory control \( V \). For this, we first introduce a new concept called **heterogeneous quantitative completability**.

Let \( K \subseteq L_m(G) \) be a sublanguage of \( L_m(G) \), and define

\[
Q_m(K) := \{ q_m \in Q_m | \exists s \in K \, \delta(q_0, s) = q_m \}. \tag{6}
\]

If \( K \) is represented by a nonblocking generator \( K \), i.e. \( L_m(K) = K \), then we have \( Q_m(K) = Q_m(K) \).

We also write \( Q_m(K) = \{ q_{m,i} | i \in I \} \) (\( I \) is an index set). For each \( q_{m,i} \in Q_m(K) \), define

\[
Str_{K}(q_{m,i}) := \{ s \in K | \delta(q_0, s) = q_{m,i} \}; \tag{7}
\]

then \( Str_{K}(q_{m,i}) \) includes all the strings in \( K \) whose occurrences mean the completion of a marked behavior as represented by \( q_{m,i} \). Note that \( Str_{K}(q_{m,i}) \) is nonempty because \( q_{m,i} \in Q_m(K) \). For an arbitrary string \( s \in \overline{K} \) and an arbitrary marker state \( q_{m,i} \in Q_m(K) \), define the set of strings that lead \( s \) to \( Str_{K}(q_{m,i}) \) for the first time:

\[
M_{K}(s, q_{m,i}) := \{ t \in \Sigma^* | s \in Str_{K}(q_{m,i}) \land (\forall t' \in \overline{I} \setminus \{t\})st' \notin Str_{K}(q_{m,i}) \}; \tag{8}
\]

If \( s \in Str_{K}(q_{m,i}) \), we define \( M_{K}(s, q_{m,i}) := \{ \epsilon \} \).

**Definition 5:** Let \( G \) be a nonblocking generator, \( K \subseteq L_m(G) \) a sublanguage, \( Q_m(K) \subseteq Q_m \) a subset of marker states as defined in (6), and \( N_i \) a positive integer for each \( q_{m,i} \in Q_m(K) \). We say that \( K \) is **heterogeneous quantitative completable** wrt. \( \{(q_{m,i}, N_i) | i \in I \} \) if for all \( s \in \overline{K} \) and all \( q_{m,i} \in Q_m(K) \),

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

(ii) \( (\forall t \in M_{K}(s, q_{m,i})) |t| \leq N_i \).

If \( K \) is heterogeneous quantitative completable, then for every marker state \( q_{m,i} \in Q_m(K) \), every string \( s \in \overline{K} \) may be extended to a string in \( Str_{K}(q_{m,i}) \) (generating the marked behavior represented by \( q_{m,i} \))
by strings of lengths at most $N_i$. Comparing with Definition 2 of quantitative completablity, the above definition distinguishes individual marker states (in $Q_m(K)$).

We illustrate Definition 5 by the following example, with emphasis on the difference from quantitative completablity.

**Example 5 (Continuing Example 1):** Reconsider the generator $G$ in Example 1, and let $K_1, K_2, K_3 \subseteq L_m(G)$ be sublanguages as represented by generators $K_1, K_2$ and $K_3$ respectively (displayed in Fig. 3).

For all these three cases $Q_m(K_i) = \{1, 2\}$.

First, although $K_1$ is quantitative completable wrt. $N = 1$, it is not heterogeneous quantitative completable wrt. $\{(1, N_1 = 1), (2, N_2 = 1)\}$. This is because if the vehicle is at state 3, it is not guaranteed to arrive marker state 1 (corresponding to $Str_{K_1}(1)$) in one step.

Second, $K_2$ is not quantitative completable wrt. any finite positive integer $N$, and by the same reason as described in Example 1 (the finite but unboundedly occurrences of string 19.17) it is also not heterogeneous quantitative completable wrt. any $\{(1, N_1), (2, N_2)\}$.

Third, $K_3$ is quantitative completable wrt. $N = 4$, but it is not heterogeneous quantitative completable wrt. $\{(1, N_1 = 4), (2, N_2 = 4)\}$, because marker state 1 is not guaranteed to be arrived from state 2 in 4 steps. Also, $K_3$ is not heterogeneous quantitative completable wrt. $\{(1, N_1 = 5), (2, N_2)\}$ for any positive integer $N_2$, because although marker state 1 can be arrived from any other state in 5 steps, but marker state 2 is not guaranteed to be arrived from any other state in $N_2$ steps (due to the existence of string $(11.29)^{N_2}$).

Finally, remove transitions $(1, 21, 4)$ and $(1, 12, 0)$ from $K_3$ and denote the new generator by $K'_3$ (see Fig. 6). Let $K'_3 = L_m(K'_3)$, and it is verified that $K'_3$ is both quantitative completable wrt. $N = 5$ and heterogeneous quantitative completable wrt. $\{(1, N_1 = 5), (2, N_2 = 5)\}$.  

Note in general that Definitions 2 and 5 do not imply each other when there is no particular relationship between $N$ and $\{N_i|i \in \mathcal{I}\}$. But there are two special cases: (i) when $|Q_m(K)| = 1$ and $N_i = N$, the two concepts are identical; (ii) when $N$ is greater than or equal to the minimum $N_i$, if a language is
heterogeneous quantitative completable wrt. \( \{(q_{m,i}, N_i) | i \in \mathcal{I}\} \), then it is quantitative completable wrt. \( N \). The first result is obvious, and the reason for (ii) is as follows. Let \( q_{m,i} \in Q_m(K) \) be the marker state corresponding to the minimum \( N_i \). Required by heterogeneous quantitative completability, every string \( s \in K \) may be extended to a string in \( Str_K(q_{m,i}) \) by strings of lengths at most \( N_i \), and thus \( s \) can be extended to a string in \( K \) (due to \( Str_K(q_{m,i}) \subseteq K \)). Thus \( K \) is quantitative completable wrt. the minimum \( N_i \), and is in turn quantitative completable wrt. any other integer \( N \) greater than or equal to the minimum \( N_i \).

With the above newly introduced heterogeneous quantitative language completability, we introduce a heterogeneous quantitative nonblocking property of a supervisory control.

**Definition 6:** Let \( G \) be a nonblocking generator, \( K \subseteq L_m(G) \) a sublanguage, \( Q_m(K) \subseteq Q_m \) a subset of marker states as defined in (6), \( N_i \) a positive integer for each \( q_{m,i} \in Q_m(K) \), and \( V : L(G) \to \Gamma \) a (marking) supervisory control (for \( (K, G) \)). We say that \( V \) is heterogeneous quantitative nonblocking wrt. \( \{(q_{m,i}, N_i) | i \in \mathcal{I}\} \) if

(i) \( V \) is nonblocking; and

(ii) \( L_m(V/G) \) is heterogeneous quantitative completable wrt. \( \{(q_{m,i}, N_i) | i \in \mathcal{I}\} \).

According to Definitions 4 and 5 a language \( K \) is heterogeneous quantitative completable wrt. \( \{(q_{m,i}, N_i) | i \in \mathcal{I}\} \) iff the generator \( K \) representing \( K \) is heterogeneous quantitative nonblocking wrt. \( \{(q_{m,i}, N_i) | i \in \mathcal{I}\} \). Thus when the closed-loop system \( V/G \) is represented by a (nonblocking) generator \( CG \) (say), i.e. \( L_m(CG) = L_m(V/G) \) and \( L(CG) = L(V/G) \), if \( V \) is heterogeneous quantitative nonblocking wrt. \( \{(q_{m,i}, N_i) | i \in \mathcal{I}\} \), then the generator \( CG \) is heterogeneous quantitative nonblocking wrt. \( \{(q_{m,i}, N_i) | i \in \mathcal{I}\} \). Namely, the closed-loop system \( V/G \) has the property of heterogeneous quantitative nonblockingness, and thus satisfies the requirement (R2).

We are ready to formulate the Heterogeneous Quantitative Nonblocking Supervisory Control Problem of DES (HQNSCP).

Consider a DES plant modeled by a generator \( G = (Q, \Sigma_c \cup \Sigma_{uc}, \delta, q_0, Q_m) \), a specification language \( E \subseteq \Sigma^* \), and let \( K \subseteq E \cap L_m(G) \), \( Q_m(K) \) as defined in (6), and \( N_i \) a positive integer for each \( q_{m,i} \in Q_m(K) \). Construct a (marking) supervisory control \( V : L(G) \to \Gamma \) (for \( (K, G) \)) satisfying the following properties:

- **Safety.** Marked behavior of the closed-loop system \( V/G \) satisfies the imposed specification \( E \) in the sense that \( L_m(V/G) = K \subseteq E \cap L_m(G) = K \).
- **Heterogeneous quantitative nonblockingness.** Supervisory control \( V \) is heterogeneous quantitative
nonblocking wrt. \( \{(q_m,i, N_i) | i \in I \} \).

- **Maximal permissiveness.** Supervisory control \( V \) does not restrict more behavior than necessary to satisfy safety and heterogeneous quantitative nonblockingness, i.e. for all other supervisory controls \( V' \) it holds that \( L_m(V'/G) \subseteq L_m(V/G) \).

The above HQNSCP is also a generalization of the traditional nonblocking supervisory control problem [1, 2, 4], in that the second requirement of heterogeneous quantitative nonblockingness is stronger than the traditional nonblockingness. This generalized problem cannot be solved in general by supervisors synthesized using the standard method. Also note that HQNSCP cannot be solved by the method introduced above for solving QNSCP. For example, with the same \( G, E \) and \( N = 3 \), the supervisor \( QSUP \) in Fig. 5 cannot solve HQNSCP wrt. \( \{(1, N_1 = N), (2, N_2 = N)\} \), because if the vehicle is in zone 3 (state 3), it is not guaranteed to arrive zone 1 (state 1) in three steps. In the rest of this section we will present a solution for HQNSCP.

### B. Supremal Heterogeneous Quantitative Completable Sublanguage and Its Computation

To solve the HQNSCP formulated above, we first present the following result.

**Theorem 6:** Let \( K \subseteq E \cap L_m(G) (K \neq \emptyset) \), \( Q_m(K) \) as defined in (6), and \( N_i \) a positive integer for each \( q_m,i \in Q_m(K) \). There exists a heterogeneous quantitative nonblocking supervisory control \( V \) (for \( (K, G) \)) such that \( L_m(V/G) = K \) if and only if \( K \) is controllable and heterogeneous quantitative completable wrt. \( \{(q_m,i, N_i) | i \in I \} \). Moreover, if such a heterogeneous quantitative nonblocking supervisory control \( V \) exists, then it may be implemented by a heterogeneous quantitative nonblocking generator \( HQSUP \), i.e. \( L_m(HQSUP) = L_m(V/G) \).

\[ \diamond \]

Theorem 6 asserts that when the \( K \)-synthesizing supervisory control \( V \) is required to be heterogeneous quantitative nonblocking, it is necessary and sufficient to require that \( K \) be not only controllable but also heterogeneous quantitative completable. Such a \( V \) may be realized by a generator that is not only nonblocking but heterogeneous quantitative nonblocking. This result extends the standard one of supervisory control theory (i.e. Theorem [1, 1, 2, 4]).

If \( K \) is indeed controllable and heterogeneous quantitative completable, then the supervisory control \( V \) in Theorem 6 is the solution to the HQNSCP. If \( K \) is either not controllable or not heterogeneous quantitative completable, then to achieve the third requirement of maximal permissiveness of HQNSCP, one would hope that the supremal controllable and heterogeneous quantitative completable sublanguage of \( K \) exists. Again the key is to investigate if for heterogeneous quantitative completability the supremal element also exists. We provide a positive answer below. Before we proceed, the following is a proof of Theorem 6.
Proof of Theorem 6: We first prove the first statement. The direction of (only if) is a direct result from Theorem 1 and Definition 6. For the direction of (if), according to Theorem 1, since $K$ is controllable, there exists a supervisor $V$ such that $V$ is nonblocking and $L_m(V/G) = K$. Furthermore, according to Definition 6, it is derived from $L_m(V/G) = K$ is heterogeneous quantitative completable wrt. $\{(q_{m,i}, N_i)| i \in \mathcal{I}\}$ that $V$ is heterogeneous quantitative nonblocking wrt. $\{(q_{m,i}, N_i)| i \in \mathcal{I}\}$.

For the second statement, let $V$ be a heterogeneous quantitative nonblocking supervisory control that synthesizes a controllable and heterogeneous quantitative completable $K$, i.e. $L_m(V/G) = K$. From Theorem 1, we know that $V$ may be implemented by a nonblocking generator $\text{HQSUP} = (X, \Sigma, \xi, x_0, X_m)$; that is $L_m(\text{HQSUP}) = L_m(V/G) = K$. Without loss of generality, consider that $\text{HQSUP}$ does not contain non-reachable states; if there were non-reachable states, simply remove them and this does not change $L_m(\text{HQSUP})$. Let $x \in X$ be an arbitrary (reachable) state; since $\text{HQSUP}$ is nonblocking, $x$ is also coreachable. The reachability of $x$ means that there is a string $s \in L(\text{HQSUP})$ such that $\xi(x_0, s)!$ and $\xi(x_0, s) = x$. It follows from $L(\text{HQSUP}) = L_m(\text{HQSUP}) = K$ that $s \in K$. Since $K$ is heterogeneous quantitative completable wrt. $\{(q_{m,i}, N_i)| i \in \mathcal{I}\}$, for all $q_{m,i} \in Q_m(K)$, $M_K(s, q_{m,i}) \neq \emptyset$, and for every $t \in M_K(s, q_{m,i})$, the length $|t|$ satisfies $|t| \leq N$. By the definition of $M_K(s, q_{m,i})$, this implies that every string $t$ that leads state $x$ to a marker state in $\text{Str}_K(q_{m,i})$ has length no more than $N$, i.e. $|t| \leq N$. According to the definition of $\text{Rch}_K(q_{m,i})$ and $\text{C}_K(x, q_{m,i})$, it derives from $M_K(s, q_{m,i}) \neq \emptyset$ that $\text{C}_K(x, q_{m,i}) \neq \emptyset$. For all $t \in \text{C}_K(x, q_{m,i})$, it holds that $t \in M_K(s, q_{m,i})$; thus we have $|t| \leq N$. Hence $x$ is $N_i$-step coreachable (wrt. $q_{m,i}$). Finally by Definition 4, $\text{HQSUP}$ is a heterogeneous quantitative nonblocking generator. This completes the proof. 

B.1. Supremal Heterogeneous Quantitative Completabe Sublanguage

The following proposition asserts that, similar to quantitative completable, if two languages $K_1$ and $K_2$ are heterogeneous quantitative completable and are compatible in the sense that $Q_m(K_1) = Q_m(K_2)$, then their union is also heterogeneous quantitative completable.

Proposition 2: Let $K_1, K_2 \subseteq L_m(G)$, $Q_m(K_1) = Q_m(K_2)$, and $N_i$ a positive integer for each $q_{m,i} \in Q_m(K_1) = Q_m(K_2)$. If both $K_1$ and $K_2$ are heterogeneous quantitative completable wrt. $\{(q_{m,i}, N_i)| i \in \mathcal{I}\}$, then $K := K_1 \cup K_2$ is also heterogeneous quantitative completable wrt. $\{(q_{m,i}, N_i)| i \in \mathcal{I}\}$.

Proof: Since $Q_m(K_1) = Q_m(K_2)$, we have $Q_m(K) = Q_m(K_1) = Q_m(K_2)$. Let $s \in K$ and $q_{m,i} \in Q_m(K)$. According to Definition 5, to show that $K$ is heterogeneous quantitative completable wrt. $\{(q_{m,i}, N_i)| i \in \mathcal{I}\}$, we will show that (i) there exists $t \in \Sigma^*$ such that $st \in \text{Str}_K(q_{m,i})$, and (ii) for all $t \in M_K(s, q_{m,i})$, $|t| \leq N$. Since $K = K_1 \cup K_2 = K_1 \cup K_2$, either $s \in K_1$ or $s \in K_2$. We consider the case $s \in K_1$; the other case is similar.

We first show that (i) holds. Since $K_1$ is heterogeneous quantitative completable wrt. $\{(q_{m,i}, N_i)| i \in \mathcal{I}\}$,
there exists string \( t \) such that \( st \in Str_{K_1}(q_{m,i}) \subseteq Str_K(q_{m,i}) \).

For (ii), let \( t \in M_K(s,q_{m,i}) \); then \( st \in Str_K(q_{m,i}) \) and for all \( t' \notin \{t\} \), \( st' \notin Str_K(q_{m,i}) \). Then, there may exist the following two cases: (a) \( st \in Str_{K_1}(q_{m,i}) \) and for all \( t' \notin \{t\} \), \( st' \notin Str_{K_1 \cup K_2}(q_{m,i}) \); (b) \( st \in Str_{K_2}(q_{m,i}) \) and for all \( t' \notin \{t\} \), \( st' \notin Str_{K_1 \cup K_2}(q_{m,i}) \). When case (a) holds, it derives from \( Str_{K_1 \cup K_2}(q_{m,i}) \supseteq Str_{K_1}(q_{m,i}) \) that \( st' \notin Str_{K_1}(q_{m,i}) \); so \( t \in M_K(s,q_{m,i}) \); since \( K_1 \) is heterogeneous quantitative completable wrt. \( \{q_{m,i}, N_i\} \), it must hold that \( |t| \leq N \). When case (b) holds, it derives from \( st \in Str_{K_2}(q_{m,i}) \) that \( st \in K_2 \subseteq K_2 \); thus \( t \in M_{K_2}(s,Str_{K_2}(q_{m,i})) \). Since \( K_2 \) is heterogeneous quantitative completable wrt. \( \{q_{m,i}, N_i\} \), it also must hold that \( |t| \leq N \). Hence we conclude that \( K \) is also heterogeneous quantitative completable wrt. \( \{q_{m,i}, N_i\} \).

Consider a language \( K \subseteq L_m(G), Q_m(K) \subseteq Q_m \) a subset of marker states as defined in (6), and \( N_i \) a positive integer for each \( q_{m,i} \in Q_m(K) \). Whether or not \( K \) is heterogeneous quantitative completable wrt. \( \{q_{m,i}, N_i\} \), let

\[
\mathcal{HQC}(K,Q_m(K),\{N_i|i \in I\}) := \{K' \subseteq K | K' \text{ is heterogeneous quantitative completable wrt. } \{q_{m,i}, N_i\} \}
\]

represent the set of sublanguages of \( K \) that are heterogeneous quantitative completable wrt. \( \{q_{m,i}, N_i\} \). Note from Definition 5 that the empty language \( \emptyset \) is trivially heterogeneous quantitative completable, so \( \emptyset \in \mathcal{HQC}(K,Q_m(K),\{N_i|i \in I\}) \) always holds. This means that \( \mathcal{HQC}(K,Q_m(K),\{N_i|i \in I\}) \) is nonempty. Moreover by Proposition 2 there exists the supremal heterogeneous quantitative completable sublanguage of \( K \), given by

\[
\sup \mathcal{HQC}(K,Q_m(K),\{N_i|i \in I\}) := \bigcup \{K' | K' \in \mathcal{HQC}(K,Q_m(K),\{N_i|i \in I\}) \}.
\]

In the sequel, we present a generator-based algorithm to compute \( \sup \mathcal{HQC}(K,Q_m(K),\{N_i|i \in I\}) \) for any given language \( K \subseteq L_m(G), Q_m(K) \) and \( \{N_i|i \in I\} \).

\textbf{B.2. Generator-Based Computation of } \( \sup \mathcal{HQC}(K,Q_m(K),\{N_i|i \in I\}) \)

Consider a language \( K \subseteq L_m(G), Q_m(K) \subseteq Q_m \) a subset of marker states as defined in (6), and \( N_i \) a positive integer for each \( q_{m,i} \in Q_m(K) \). We present an algorithm to compute the supremal heterogeneous quantitative completable sublanguage \( \sup \mathcal{HQC}(K,Q_m(K),\{N_i|i \in I\}) \). Let \( K \) be represented by a (nonblocking) generator \( K = (X,\Sigma,\xi,x_0,X_m) \), i.e. \( L_m(K) = K \) and \( L(K) = \overline{K} \). As mentioned, heterogeneous quantitative completable sublanguage corresponds to quantitative completable in the special case when \( |Q_m| = 1 \) and \( N_i = N \). Also, the supremal quantitative completable sublanguage is effectively computable by Algorithm 1. Hence, the intuition of computing the supremal heterogeneous quantitative
completable sublanguage is to apply Algorithm 1 iteratively for each \(q_{m,i} \in Q_m(K)\).

However, several important modifications are necessary. In particular, we need to find the states that are \(N_i\)-step coreachable wrt. \(q_{m,i}\) in each iteration. For this purpose, unlike Algorithm 1 which treats all the marker states of \(K\) the same, we need to compute \(Rch_K(q_{m,i})\) for each \(q_{m,i}\), where \(Rch_K(q_{m,i})\) includes all the marker states in \(K\) representing the same marked behavior represented by \(q_{m,i}\). Thus we modify plant \(G\) to be \(G_i\) that has only one marker state \(q_{m,i}\) for each iteration, and by the synchronous product of \(G_i\) with \(K\), the subset \(Rch_K(q_{m,i})\) can be obtained. The detailed steps are described in Algorithm 3.

---

**Algorithm 4**: Algorithm of Computing \(\sup HQC(K, Q_m(K), \{N_i|i \in \mathcal{I}\})\)

**Input**: Generator \(G = (\Sigma, \delta, q_0, Q_m)\), generator \(K = (X, \Sigma, \xi, x_0, X_m)\) representing \(K, Q_m(K) = \{q_{m,i}|i \in \mathcal{I}\}\) and \(\{N_i|i \in \mathcal{I}\}\).

**Output**: Generator \(K' = (X', \Sigma', \zeta', x'_0, X'_m)\).

**Step 1.** Let \(j = 1\) and \(K_j = K\).

**Step 2.** Let \(i = 1\). Let \(K_{i,j} = K_j\).

**Step 2.1** For each \(q_{m,i} \in Q_m(K)\), let \(G_i = (\Sigma, \delta, q_0, \{q_{m,i}\})\) and compute the product generator \(GK_{i,j}\) of \(G_i\) and \(K_{i,j}\), i.e. \(L_m(GK_{i,j}) = L_m(K_{i,j}) \cap L_m(G_i)\) and \(L(GK_{i,j}) = L(K_{i,j}) \cap L(G_j)\). Apply Algorithm 1 with inputs \(GK_{i,j}\) and \(N_i\), obtain \(L_m(X'_{i,j}) = \sup QC(L_m(GK_{i,j}), N_i)\).

**Step 2.2** Let \(X'_{i,j}, X_{i,m,i,j}\) be the state set and marker state set of \(K'_{i,j}\) respectively, and \(X \times X'_{i,j}\) be the state set of the product generator of \(K\) and \(K'_{i,j}\). For each \((x, x') \in X \times X'_{i,j}\), if \(x \in X_m\), add \(x'\) to the marker state set \(X'_{i,m,i,j}\); we rename the modified \(K'_{i,j}\) as \(K''_{i,j}\).

**Step 2.3** If \(i < |\mathcal{I}|\), let \(K_{i+1,j} = K''_{i,j}\), advance \(i\) to \(i + 1\) and go to Step 2.1; otherwise, go to Step 3.

**Step 3.** Let \(K_{j+1} = K'_{|\mathcal{I}|,j}\). If \(K_{j+1} = K_j\), output \(K' = K_{j+1}\). Otherwise, advance \(j\) to \(j + 1\) and go to Step 2.

---

**Example 6 (Continuing Example 1):** Applying Algorithm 4 with inputs \(G, \sup. Q_m(L_m(\sup)) = \{q_{m,1} = 1, q_{m,2} = 2\}\) \((\mathcal{I} = \{1, 2\})\) and \(\{N_1 = 3, N_2 = 4\}\), we compute the supremal heterogeneous quantitative completable sublanguage of \(L_m(K_3)\) with respect to \(\{(1, N_1 = 3), (2, N_2 = 4)\}\). At Step 1, set \(K_1 = K\).

At Step 2, \(K_{1,1} = K_1\). At Step 2.1, let \(G_1\) be the generator that has the same transition function of \(G\), but state 1 as the unique marker state. Compute the product generator \(GK_{1,1}\) of \(G_1\) and \(K_{1,1}\). Applying Algorithm 1 with inputs \(GK_{1,1}\), and \(N_1 = 3\), we obtain a generator \(K'_{1,1}\) as displayed in Fig. 7. At Step 2.2, we add marker state 2 and 5 to \(K'_{1,1}\) and obtain \(K''_{1,1}\). Note that this step is necessary because without it, the sublanguages that are not included in \(L_m(G_1)\) will be considered in the next iteration. At Step 2.3, since \(i = 1 < |\mathcal{I}| = 2\), let \(K_{2,1} = K'_{1,1}\) and go to Step 2.1 with \(i = 2\). Repeat Steps 2.1 and 2.2 with \(G_2\) (considering marker state 2 of \(G\)), \(K_{2,1}\) and \(N_2 = 4\), we obtain generators \(K'_{2,1}\) as displayed in Fig. 8 and \(K''_{2,1}\) (as displayed in Fig. 9). Then at step 2.3, since \(i = 2 = |\mathcal{I}|\), go to Step 3 and let \(K_2 = K''_{2,1}\) (as shown in Fig. 9). Obviously, \(K_2 \neq K_1\), thus we repeat Steps 2, 2.1-2.3 with \(K_2\).
and then obtain $K_3$. It is verified that $K_3 = K_2$; thus the algorithm terminates and output $K_3$ which has the same transition structure as $K_2$.

From Fig. 9 we see that in $K_3$, states 1, 4 and 9 correspond to marker state 1 of $G$, and states 2 and 5 correspond to marker state 2 of $G$. Starting from any states of $K_3$, the states 1 or 4 or 9 will be reached in three steps; also starting from any states, states 2 or 5 will be reached in four steps. Hence, $K_3$ implements a heterogeneous quantitative nonblocking supervisor; and under the control of $K_3$, $G$ are guaranteed to reach the marker state 1 in at most three steps from any other states, and marker state 2 in at most four steps.

The correctness of Algorithm 4 is confirmed by the following theorem.

**Theorem 7:** Given a generator $G = (Q, \Sigma, \delta, q_0, Q_m)$, a sublanguage $K \subseteq L_m(G)$, a subset of marker states $Q_m(K) \subseteq Q_m$ as defined in 6, and a positive integer $N_i$ for each $q_{m,i} \in Q_m(K)$, let $K'$ be the generator returned by Algorithm 4. Then $L_m(K') = \sup \mathcal{HC}(K, Q_m(K), \{N_i\mid i \in I\})$.

Before proving Theorem 7, we demonstrate that Algorithm 4 will terminate in a finite number of iterations. Initially, the transition number of both $K_1$ and $K_{1,1}$ is $|X_1|$. Then by Step 2.1, the states

---

4The sentence ‘state $x$ in $K_3$ corresponds to state $q$ in $G$’ means that the strings visiting $x$ in $K_3$ also visit $q$ in $G$. 

---
number $|X'_{1,1}|$ of $K_{1,1}'$ is at most $|Q| \cdot |X_{1,1}| \cdot N_1$; for simplicity, we write $|X'_{1,1}| = O(|Q| \cdot |X_{1,1}| \cdot N_1)$.

By Step 2.2, from $K_{1,1}'$ to $K_{1,1}''$, we first compute the product of $K_{1,1}'$ and $K$, and then add marker state to the state set. Actually, $|X''_{1,1}| = |X'_{1,1}|$, because (i) since $K_{1,1}'$ is obtained from the product of $G_1$ and
Thus by steps 2 and 4, we have that \( N \) quantitative completable wrt. state sizes of the generators in Algorithm 4 is: hence, \( s \) number of the generators is \( K \) overall time complexity of Algorithm 4 is \( O \). Furthermore, the state size of \( K'' \) is \( |X_{2,1}'| = |X_{2,1}| \). Inductively, it is derived that the maximal state sizes of the generators in Algorithm 4 is: \( O(|Q| \cdot |X| \cdot \prod_{i=1}^{|I|} N_i) \); thus the maximal transition number of the generators is \( O(|Q| \cdot |X| \cdot \Sigma \cdot \prod_{i=1}^{|I|} N_i) \). Hence, in the worst case, Algorithm 4 will terminate in \( O(|Q| \cdot |X| \cdot \Sigma \cdot \prod_{i=1}^{|I|} N_i) \) number of iterations. At each iteration (including Steps 2 and 3), there are \(|I| \) times calling of Algorithm 4 which has time complexity \( |X_{i,j}| \cdot |\Sigma| \cdot |N_i| \); thus the time complexity of each iteration is \( O(|Q| \cdot |X| \cdot \Sigma \cdot \prod_{i=1}^{|I|} N_i) \). Hence the overall time complexity of Algorithm 4 is \( O(|Q| \cdot |X| \cdot \Sigma \cdot \prod_{i=1}^{|I|} N_i) \cdot O(\sum_{i=1}^{|I|} (|Q| \cdot |X| \cdot \Sigma \cdot \prod_{i=1}^{|I|} N_i)) = O(|I| \cdot (|Q| \cdot |X| \cdot |\Sigma| \cdot \prod_{i=1}^{|I|} N_i)^2) \).

**Proof of Theorem 4** First, we prove that \( L_m(K') \in HQC(K, Q_m(K), \{N_i|i \in I\}) \). If \( L_m(K') = \emptyset \), the result holds obviously; in the following, we assume that \( L_m(K') \neq \emptyset \). We start by showing that \( L_m(K') \subseteq K \). First, by Steps 2.1 and 2.2, we have

\[
L_m(K'_{i,j}) = \sup QC(L_m(K_{i,j}) \cap L_m(G_i), N),
L_m(K''_{i,j}) = L_m(K'_{i,j}) \cap K;
\]

(9)

thus by steps 2 and 4, we have

\[
L_m(K_{j+1}) = L_m(K''_{j,j}) \subseteq L_m(K_{j,j}) \subseteq \ldots \subseteq L_m(K''_{1,j}) \subseteq L_m(K_{1,j}) = L_m(K_{j,j}) \subseteq \ldots \subseteq L_m(K_1) = L_m(K) = K.
\]

(10)

Hence, \( L_m(K') \subseteq L_m(K_{j+1}) \subseteq K \).

Next we show that \( L_m(K') \) is heterogeneously quantitatively completable wrt. \( \{(q_{m,i}, N_i)|i \in I\} \). Let \( s \in L(K') \); then for each \( i \) and \( j \) in Algorithm 4, \( s \in L(K_{i,j}) \). According to Step 3, we know that if for some \( j \), \( K_j = K_{j-1} \), then \( K' = K_j \). Namely, at step 2.1-2.2, for each \( i \in I \), \( L_m(K_{i,j-1}) \cap L_m(G_i) \) is quantitative completable wrt. \( N_i \), which means that string \( s \) can be completed to strings visiting every

\footnote{For two generators A and B having the same event set, if C is the product generator of A and B, (i.e. \( C = A \times B \)), and D is the product generator of C and B (i.e. \( D = C \times B \)), then the state size of D is equal to that of C.}
marker state \( q_{m,i} \) in \( N_i \)-steps. Hence, we conclude that \( L_m(K') \) is heterogeneous quantitative completable wrt. \( \{ (q_{m,i}, N_i) | i \in I \} \).

It remains to show that \( L_m(K') \) is the largest element in \( \mathcal{HQC}(K, Q_m(K), \{ N_i | i \in I \}) \). Assuming that \( M \) be another nonempty element in \( \mathcal{HQC}(K, Q_m(K), \{ N_i | i \in I \}) \), it will be shown that \( M \subseteq L_m(K') \). We prove this by contradiction. Assume that there exists string \( s \in \Sigma^* \) such that \( s \in M \) but \( s \notin L_m(K') \). By \( s \in K \) (since \( s \in M \)) and \( s \notin L_m(K') \); according to (10), there must exists \( i \) and \( j \) such that \( s \in L_m(K_{i,j}) \cap L_m(G_i) \), but \( s \notin \sup \mathcal{QC}(L_m(K_{i,j}) \cap L_m(G_i), N) \cap K \). Namely, one of the following two cases must hold: (i) there does not exist string \( t \) such that \( st \in L_m(K_{i,j}) \cap L_m(G_i) \) or (ii), for all string \( t \in M L_m(K_{i,j}) \cap L_m(G_i)(s), |t| > N_i \). Namely, string \( t \) cannot be completed to state \( q_{m,i} \) in \( N_i \)-steps. This is contradiction to the assumption that \( M \in \mathcal{HQC}(K, Q_m(K), \{ N_i | i \in I \}) \) and \( s \in M \subseteq \overline{M} \). Hence, we conclude that \( L_m(K') = \sup \mathcal{HQC}(K, Q_m(K), \{ N_i | i \in I \}) \). The proof is now complete.

### C. Maximally Permissive Heterogeneous Quantitative Nonblocking Supervisory Control

In this subsection, we present our solution to the HQNSCP. Consider a DES plant modeled by a generator \( G \) as in (1) with \( \Sigma = \Sigma_e \cup \Sigma_{uc} \), and a specification language \( E \subseteq \Sigma^* \). Let \( K := E \cap L_m(G), Q_m(K) \) as defined in (6), and \( N_i \) a positive integer for each \( q_{m,i} \in Q_m(K) \).

Whether or not \( K \) is controllable and heterogeneous quantitative completable, let \( CHQC(K, Q_m(K), \{ N_i | i \in I \}) \) be the set of sublanguages of \( K \) that are both controllable and heterogeneous quantitative completable wrt. \( \{ (q_{m,i}, N_i) | i \in I \} \), i.e.

\[
CHQC(K, Q_m(K), \{ N_i | i \in I \}) := \{ K' \subseteq K | K' \text{ is both controllable and heterogeneous quantitative completable wrt. } \{ (q_{m,i}, N_i) | i \in I \} \}.
\]

Since the empty language \( \emptyset \) is trivially controllable and heterogeneous quantitative completable, the set \( CHQC(K, T, N) \) is nonempty. Moreover, since both controllability and heterogeneous quantitative completability are closed under arbitrary set unions, \( CHQC(K, T, N) \) contains a unique supremal element given by

\[
\sup CHQC(K, Q_m(K), \{ N_i | i \in I \}) := \bigcup \{ K' \subseteq K | K' \in CHQC(K, Q_m(K), \{ N_i | i \in I \}) \}.
\]

Our main result in this subsection is the following.

**Theorem 8:** Suppose that \( \sup CHQC(K, Q_m(K), \{ N_i | i \in I \}) \neq \emptyset \). Then the supervisory control \( V_{hsup} \) such that \( L_m(V_{hsup}/G) = \sup CHQC(K, Q_m(K), \{ N_i | i \in I \}) \subseteq K \) is the solution to the HQNSCP.
Proof: Let $\sup CHQC(K, Q_m(K), \{N_i|i \in I\})$ be nonempty. According to Theorem 6, since $\sup CHQC(K, Q_m(K), \{N_i|i \in I\})$ is controllable and heterogeneous quantitative completable, there exists a heterogeneous quantitative nonblocking supervisory control $V_{hsup}$ such that $L_m(V_{hsup}/G) = \sup CHQC(K, Q_m(K), \{N_i|i \in I\}) \subseteq K$. Hence the safety and heterogeneous quantitative nonblockingness requirements of the HQNSCP are satisfied. Further, since $\sup CHQC(K, Q_m(K), \{N_i|i \in I\})$ is the supremal element in $CHQC(K, Q_m(K), \{N_i|i \in I\})$, the maximal permissiveness requirement of the HQNSCP is also satisfied. Therefore, $V_{hsup}$ that synthesizes $\sup CHQC(K, Q_m(K), \{N_i|i \in I\})$ is the solution to the HQNSCP.

We proceed to design an algorithm to compute this solution $\sup CHQC(K, Q_m(K), \{N_i|i \in I\})$. Similar to Algorithm 3 of computing $\sup CQC(K, N)$, we suitably modify Algorithm 3 by replacing Algorithm 1 (for computing the supremal quantitative completable sublanguage) by Algorithm 5 for computing the supremal heterogeneous quantitative completable sublanguage. The new algorithm is as follows.

---

**Algorithm 5**: Algorithm of Computing $\sup CHQC(K, Q_m(K), \{N_i|i \in I\})$

**Input**: Plant generator $G = (Q, \Sigma, \delta, q_0, Q_m)$, specification language $E \subseteq \Sigma$, $Q_m(E \cap L_m(G)) = \{q_{m,i}|i \in I\}$ and $\{N_i|i \in I\}$.

**Output**: Generator $HQSUP = (X, \Sigma, \delta, x_0, X_m)$.

**Step 1.** Let $K := E \cap L_m(G)$ (thus $Q_m(K) = Q_m(E \cap L_m(G))$). Construct a nonblocking generator $K_0$ such that $L_m(K_0) = K$.

**Step 2.** For $i \geq 0$, apply Algorithm 4 (with inputs $K_i, Q_m(K) = \{q_{m,i}|i \in I\}$, and $\{N_i|i \in I\}$) to compute $H_i$ such that $L_m(H_i) = \sup SQC(L_m(K_i), Q_m(K), \{N_i|i \in I\})$.

**Step 3.** Apply Algorithm 2 (with inputs $G$ and $L_m(H_i)$) to compute $K_{i+1}$ such that $L_m(K_{i+1}) = \sup C(L_m(H_i))$.

**Step 4.** If $K_{i+1} = K_i$, then output $HQSUP = K_{i+1}$. Otherwise, advance $i$ to $i + 1$ and go to Step 2.

---

The correctness of this algorithm is confirmed by the following theorem.

**Theorem 9:** Given a plant generator $G$, a specification language $E$, let $K := E \cap L_m(G)$, $Q_m(K) \subseteq Q_m$ a subset of marker states as defined in (6), $N_i$ a positive integer for each $q_{m,i} \in Q_m(K)$, and $HQSUP$ be the generator returned by Algorithm 5. Then $L_m(HQSUP) = \sup CHQC(K, Q_m(K), \{N_i|i \in I\})$.

The proof of this theorem is similar to that of Theorem 5 with $L_m(H_i) = \sup QC(L_m(K_i), N)$ being replaced by $L_m(H_i) = \sup HCQC(L_m(K_i), Q_m(K), \{N_i|i \in I\})$.

Since Algorithm 4 used in Step 2 has complexity $O((|I| \cdot (|Q| \cdot ||E||_{L} \cdot |\Sigma| \cdot \prod_{i=1}^{|I|} N_i)^2)$, Algorithm 2 used in Step 3 has complexity $O(|Q| \cdot ||E||_{L} \cdot |\Sigma| \cdot \prod_{i=1}^{|I|} N_i)$, and there can be at most $O(|Q| \cdot ||E||_{L} \cdot |\Sigma| \cdot \prod_{i=1}^{|I|} N_i)$ iterations of Steps 2 and 3, the time complexity of Algorithm 5 is $O(|I| \cdot (|Q| \cdot ||E||_{L} \cdot |\Sigma| \cdot \prod_{i=1}^{|I|} N_i)^3)$.  

---
Example 7 (Continuing Example 1): Considering plant generator $G$ and nonblocking supervisor $SUP$ displayed in Fig. 2, we now assume that the odd numbers represent controllable events and the even numbers represent uncontrollable events. Input $G$, $L_m(SUP)$, $Q_m(L_m(SUP)) = \{q_{m,1} = 1, q_{m,2} = 2\}$ ($I = \{1, 2\}$) and $\{N_1 = 3, N_2 = 4\}$ to Algorithm 5. In Step 1, $K_0 := SUP$. Then in Step 2, Algorithm 4 generates a generator $H_0 = K_3$ (displayed in Fig. 9); it is verified that $L_m(H_0) = L_m(K_3)$ is heterogeneous quantitative completable wrt. $\{(q_{m,1} = 1, N_1 = 3), (q_{m,2} = 2, N_1 = 4)\}$. However, it is not controllable, because uncontrollable event 16 is disabled at states 6 and 11. Next in Step 3, Algorithm 2 generates a generator $K_1$ as displayed Fig. 10. It is verified that $L_m(K_1)$ is both controllable and heterogeneous quantitative completable wrt. $\{(q_{m,1} = 1, N_1 = 3), (q_{m,2} = 2, N_1 = 4)\}$. This means that in Step 4, although $K_1 \neq K_0$ and Steps 2 and 3 need to be repeated, no further change will be made. It follows that $K_2 = K_1$, and Algorithm 3 outputs $HQSUP = K_1$.

This $HQSUP$ may be used as a supervisor for the autonomous vehicle to provide timely services to the customers in zone 1 (three steps) and zone 2 (four steps). The control logics are as follows: (i) never move to zone 5 when in zone 0; (ii) never move to zone 4 when in zone 1 or zone 3; and (iii) if the vehicle is in zone 1, it cannot move to zone 0 unless it has just returned back from zone 2. These logics guarantee that the second requirement (in Section II.B) on the vehicle can be satisfied; namely, every package sent to a customer in zone 1 (resp. zone 2) can be delivered by the vehicle to zone 1 (resp. zone 2) within three steps (resp. four steps), and whenever a customer in zone 1 (resp. zone 2) calls for package collection, the vehicle can reach zone 1 (resp. zone 2) within three steps (resp. four steps).

Additionally, we demonstrate that the solution for HQNSCP need not solve QNSCP in the following two cases. First, consider the case that $N = 3$ which is equal to the minimum of $N_1 = 3$ and $N_2 = 4$. It is verified that $L_m(HQSUP)$ is controllable and quantitative completable wrt. $N = 3$. Namely, $HQSUP$ can act as a supervisor satisfying the safety and quantitative nonblockingness of QNSCP.
(with $G$, $SUP$ and $N = 3$). However, comparing $HQSUP$ with the optimal supervisor $QSUP$ (as displayed in Fig. 5) for QNSUP, we find that there exists strings $(11.29)^N$ in $L_m(QSUP)$, but not in $L_m(HQSUP)$. That is, the repeated visits of zones 0 and 1 is not permitted for $HQSUP$, but is permitted for $HQSUP$. Specifically, the vehicle cannot provide timely services to the customers in zone 2 (violating the requirement (i) in Example 1); however, if the visits of zones 1 and 2 are treated the same (as in the requirement (i) in Example 1), the repeated visits of zones 1 and 2 is legal for supplying timely services. Namely, the maximal permissiveness of QNSCP is not satisfied, which demonstrates that $HQSUP$ for HQNSCP is not optimal for solving QNSCP.

Second, consider the case that $N = 1$, and $\{N_1 = 3, N_2 = 1\}$ (i.e. $N$ is again equal to the minimum of $N_1$ and $N_2$). Applying Algorithm 3 with inputs $G$, $L_m(SUP)$ and $N = 1$, we obtain an optimal supervisor $QSUP'$ (as displayed in Fig. 11) for solving QNSCP. From the figure, we see that $L_m(QSUP') \neq \emptyset$. However, applying Algorithm 5 with inputs $G$, $L_m(SUP)$, $Q_m(SUP) = \{q_{m,1} = 1, q_{m,2} = 2\}$ ($\mathcal{I} = \{1, 2\}$) and $\{N_1 = 3, N_2 = 1\}$, we get an empty generator, because marker state 2 cannot be reached in one step from the initial state. This result demonstrates that even when Algorithm 5 returns no solution for HQNSCP, Algorithm 3 can find a nonempty optimal supervisor for solving QNSCP.

\[\Box\]

V. CONCLUSION AND FUTURE WORK

In this paper, we have introduced two new properties of quantitative nonblockingness of generators. First, we have introduced a concept of quantitative nonblockingness for a given positive integer $N$, which requires that at least one of the tasks be completed in $N$ steps. With this property, we have formulated a new quantitative nonblocking supervisory control problem, characterized its solution in terms of quantitative language completability, and developed algorithms to compute the optimal solution. Second, we have introduced another new concept of heterogeneous quantitative nonblockingness for a given subset of marker states and a subset of positive integers $N_i$ one for each marker state, which requires that every task represented by a marker state be completed in corresponding $N_i$ steps. We have also formulated another heterogeneous quantitative nonblocking supervisory control problem, characterized its solution in terms of heterogeneous quantitative language completability, and developed algorithms to compute the optimal solution.
In this paper the bounds on task completion are specified in terms of the number of transition steps. In practice, the bounds may also need to be described by the number of time units or even real times. Thus in future work, we are interested in extending the concepts of quantitative nonblockingness to the nonblocking supervisory control framework of timed discrete-event systems [23, 24].

REFERENCES

[1] P. Ramadge and W. Wonham, “Supervisory control of a class of discrete event processes,” *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[2] W. Wonham and P. Ramadge, “On the supremal controllable sublanguage of a given language,” *SIAM Journal on Control and Optimization*, vol. 25, no. 3, pp. 637–659, 1987.

[3] P. Ramadge and W. Wonham, “The control of discrete event systems,” *The Proceedings of IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[4] W. Wonham and K. Cai, *Supervisory Control of Discrete-Event Systems*. Springer, 2019.

[5] K. Cai and W. Wonham, *Supervisory control of discrete-event systems*. Encyclopedia of Systems and Control, 2nd ed., Springer, 2020.

[6] W. Wonham, K. Cai, and K. Rudie, “Supervisory control of discrete-event systems: a brief history,” *Annual Reviews in Control*, vol. 45, pp. 250–256, 2018.

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

[8] R. Kumar and M. Shayman, “Non-blocking supervisory control of deterministic discrete event systems,” in *Proc. 1994 American Control Conference*, 1994, pp. 1089–1093.

[9] M. Fabian and R. Kumar, “Mutually nonblocking supervisory control of discrete event systems,” in *Proc. 36th IEEE Conference on Decision and Control*, 1997, pp. 2970–2975.

[10] C. Ma and W. Wonham, “Nonblocking supervisory control of state tree structures,” *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 782–793, 2006.

[11] R. Malik and R. Leduc, “Generalised nonblocking,” in *Proc. 9th International Workshop on Discrete Event Systems*, 2008, pp. 340–345.

[12] S. Balemi, G. Hoffmann, P. Gyugyi, H. Wong-Toi, and G. Franklin, “Supervisory control of a rapid thermal multiprocessor,” *IEEE Transactions on Automatic Control*, vol. 38, no. 7, pp. 1040–1059, 1993.

[13] B. Brandin and F. Charbonnier, “The supervisory control of the automated manufacturing system of the aip,” in *Proc. Rensselaer’s 4th Int. Conf. Computer Integrated Manufacturing and Automation Technology*, 1994, pp. 319–324.

[14] P. Malik, “From supervisory control to nonblocking controllers for discrete event systems,” Ph.D. dissertation, University of Kaiserslautern, 2003.

[15] S. Eilenberg, *Automata, Languages and Machines Volume A*. Academic Press, 1974.

[16] L. Feng, K. Cai, and W. M. Wonham, “A structural approach to the nonblocking supervisory control of
discrete-event systems,” *International Journal of Advanced Manufacturing Technology*, vol. 41, no. 11, pp. 1152–1167, 2009.

[17] J. Gagliardi, J. Renaud, and A. Ruiz, “Models for automated storage and retrieval systems: a literature review,” *International Journal of Production Research*, vol. 50, no. 24, pp. 7110–7125, 2012.

[18] R. Milner, *Communication and Concurrency*. Prentice Hall, Englewood Cliffs, NJ, 1989.

[19] R. Su, “On the complexity of synthesizing a minimum-weighted supervisor under partial observation,” *Automatica*, vol. 50, no. 6, pp. 1725–1729, 2014.

[20] S. Pruekprasert, T. Ushio, and T. Kanazawa, “Quantitative supervisory control game for discrete event systems,” *IEEE Transactions on Automatic Control*, vol. 61, no. 10, pp. 2987–3000, 2016.

[21] L. Ouedraogo, R. Kumar, R. Malik, and K. Akesson, “Nonblocking and safe control of discrete-event systems modeled as extended finite automata,” *IEEE Transactions on Automaton Science and Engineering*, vol. 8, no. 3, pp. 560–569, 2011.

[22] M. Goorden, J. van de Mortel-Fronczak, M. Reniers, W. Fokkink, and J. Rooda, “Structuring multilevel discrete-event systems with dependence structure matrices,” *IEEE Transactions on Automatic Control*, vol. 65, no. 4, pp. 1625–1639, 2020.

[23] A. Gouin, L. Libeaut, and J. Ferrier, “Supervisory control of timed automata,” in *1999 European Control Conference (ECC)*, 1999, pp. 543–550.

[24] B. Brandin and W. Wonham, “Supervisory control of timed discrete-event systems,” *IEEE Transactions on Automatic Control*, vol. 39, no. 2, pp. 329–342, 1994.

[25] J. Thistle and W. Wonham, “Supervision of infinite behavior of discrete-event systems,” *SIAM Journal on Control and Optimization*, vol. 32, no. 4, pp. 1098–1113, 1994.

[26] A. K. Schmuck, T. Moor, and R. Majumdar, “On the relation between reactive synthesis and supervisory control of non-terminating processes,” *Discrete Event Dynamic Systems*, vol. 30, no. 1, pp. 81–124, 2020.

[27] A. Saboori and C. Hadjicostis, “Verification of K-step opacity and analysis of its complexity,” *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 3, pp. 549–559, 2011.

[28] X. Yin, Z. Li, W. Wang, and S. Li, “Infinite-step opacity and K-step opacity of stochastic discrete-event systems,” *Automatica*, vol. 99, pp. 266–274, 2019.

[29] F. Lin, L. Wang, W. Chen, L. Han, and B. Shen, “N-diagnosability for active on-line diagnosis in discrete event systems,” *Automatica*, vol. 83, pp. 220–225, 2017.

[30] N. Hadjicostis and C. Seatzu, “K-detectability in discrete event systems,” in *Proc. 55th IEEE Conference on Decision and Control*, 2016, pp. 420–425.

[31] K. Zhang and A. Giua, “K-delayed strong detectability of discrete-event systems,” in *Proc. 58th IEEE Conference on Decision and Control*, 2019, pp. 7647–7652.

[32] R. Zhang, Z. Wang, and K. Cai, “N-step nonblocking supervisory control of discrete-event systems,” accepted for presentation in 60th IEEE Conference on Decision and Control, 2021.

[33] J. Hopcroft, R. Motwani, and J. Ullman, *Introduction to Automata Theory Languages and Computation*. 37
