Supervising Passenger Land-Transport Systems

Kiam Tian Seow, Member, IEEE, and Michel Pasquier

Abstract—In this paper, we propose a supervisory control approach, based on controlled automata concepts, for the planning for online service-operations control of a new class of passenger land-transport systems (PLanTSs). A PLanTS belongs to a class of dynamic demand-responsive transportation systems. Rapid advances in information and communication technologies are providing a new infrastructural and communications basis upon which higher levels of automation, flexibility, and integration in the development of such transportation systems can be achieved, but to achieve these necessitates the use of more formal approaches for system planning and design. The supervisory control theory of Ramadge and Wonham offers one such methodology, which presents the design for service-operations control as a formal synthesis of a modular supervisory controller. Importantly, the design solution is guaranteed to satisfy the given behavioral specifications in some optimal fashion, without blocking the completion of certain defined “mandatory” tasks. The supervisory design methodology is presented and illustrated in detail via what, in our opinion, is a simplified but realistic PLanTS model. A structural property of the PLanTS model is used to analytically establish the nonblocking property of the modular supervisory controller designed. All automaton models for the PLanTS and the behavioral specifications considered are provided, together with the automata design of the corresponding modular supervisor.

Index Terms—Automata, discrete-event systems, passenger land transport, service operations, supervisory control.

I. INTRODUCTION

PASSENGER land-transportation systems (PLanTSs) are concerned with transporting travelers from their source locations to their destination locations in a fleet of carrier vehicles, subject to various qualitative and quantitative constraints. These constraints characterize the environmental traffic conditions in which the services of transportation are carried out, as well as the operating conditions, limitations, and preferences of the vehicle fleet operators and travelers. Taxi-service management is an example of such a system. These systems are, however, open loop in that the logical feedback-control to react to and interleave the occurrences of incidents (e.g., vehicle breakdown and admission of a travel request) in some desired manner is apparently absent, implicit or at best done by way of ad hoc human intervention. Traditionally, the techniques available for these systems, such as those surveyed in [1] and [2], do not close the loop, for they only determine the assignment of travelers to the fleet vehicles and construct the corresponding vehicles’ service schedules or route plans. The automatic feedback of dynamically changing logical conditions needed to update the online information, such as the availability of fleet vehicles and the status of travel requests, has never been formally characterized and explored. In other words, a basic research problem in passenger service-operations lies in the open-loop nature of the information process flow in these transportation systems.

With rapid advances in information and communication technologies, such as Internet technology [3], [4], geographic information systems (GIS) [5], [6], global positioning systems (GPS) [7], [8], and intelligent transportation systems (ITS) [9], a new infrastructural and communications basis has emerged upon which the information loop can be closed to potentially achieve higher levels of automation, flexibility, and integration toward the development of new transportation systems. But to achieve this necessitates the use of appropriate formal approaches for system planning and design. In particular, an alternative but complementary framework is needed, which views a fleet of service vehicles and travelers uniformly as behavior-based components, subject to various logical constraints to be met under closed-loop supervision or what we call service-operations control.

In this paper, we consider online service-operations control of a class of PLanTS, which is a system that receives and services geographically distributed travel requests, not known a priori, that demand immediate (i.e., “as-soon-as-possible”) service. It belongs to a class of dynamic demand-responsive transportation systems [10]. In an attempt to model and understand the dynamics of discrete information flow in the service-operations control of a PLanTS, we address the service-operations control problem using the controlled automata concepts and techniques of supervisory control for a class of logical discrete-event systems [11]–[15].

We model the service-operations in a PLanTS as a discrete-event system (DES) of interacting processes to be supervised or controlled. DESs represent dynamic systems that evolve in accordance to some abrupt and asynchronous occurrence of events. Such systems are encountered in a variety of many other fields, for example, in computer and communication networks [16], [17], manufacturing [18], [19] and task-level robotics [20].

To the best of our knowledge, our work represents a first effort to apply control-theoretic ideas of supervisory control to this class of transport service-operations problems. This approach is based on information feedback on the occurrence of events (see Fig. 1). Accordingly, the approach centers around three related elements, namely:

1) models of the system (as DESs) to be controlled;
2) models of the control objectives (also called behavioral specifications) to be satisfied;
3) supervisory controller to be synthesized.

These are graphically depicted in Fig. 2. The proposed methodology admits the design for service-operations control as an automata-based synthesis of a modular supervisory controller for
the PLanTS. Other techniques, such as Petri nets [21], [22] and communicating sequential processes [23] may be exploited, but the proposed methodology offers the following important advantages over these approaches:

1) supervisory controller is correct by automatic construction, such that the resulting controlled system does not contradict the behavioral specifications and is non-blocking

2) controlled system is optimal (or minimally restrictive) within the behavioral specifications, such that all events whose occurrences do not eventually contradict the specifications are allowed to occur.

In other words, the design solution is guaranteed to satisfy given behavioral specifications (e.g., vehicle seat capacity must not be exceeded) in some optimal fashion without blocking the completion of certain defined “mandatory” tasks such as emptying the service queue. A formal and conceptually rich control synthesis software CTCT [11] is now freely available\(^1\) to support the automatic synthesis of supervisory controllers. Using CTCT, a DES model, behavioral specification and supervisory controller are represented by finite state automata that allow qualitative information such as the admission of a travel request and the assignment of a request to an available vehicle to be treated in a uniform way as events that are the state transitions in the automata.

In automating service operations for passenger land transportation, it is not exactly clear what constitutes a DES model for a PLanTS. Strictly speaking, no known prior and related work on service-operations control of a demand-responsive transportation system has been formally done from which a suitable DES model can be abstracted. However, a service-oriented model for a general transportation system exists [24], which conceptually specifies three submodels of demand, supply, and demand–supply interactions. Our starting point is based on this conceptual model, but seen from a supervisory control perspective. Together with a general understanding of the conventional but related problems of vehicle assignment and route planning (and the variations thereof) [1], [2], [10], in Section IV we abstract and fix what, in our opinion, is a simplified but realistic DES model for a PLanTS. The DES model incorporates the behavioral components of service demand by travelers and service supply by a fleet of vehicles. Importantly, the PLanTS model provides a basis on which various interesting behavioral specifications of interest representing the desired demand–supply interactions can be formulated, lending a unique opportunity to demonstrate the applicability of modular supervisory control theory to this problem. In Section IV-C, in our analysis of a property that the PLanTS model has, we have also been able to infer some structural insights on a general DES model design that guarantees that the nonblocking property in a supervisory controller exists. These constitute the main contributions of this paper.

There has been some prior work on applying the supervisory control theory in different areas of intelligent transportation. For instance, Spathopoulos and de Ridder [25] consider the DES modeling and distributed supervisory control of a subway system and Yoo et al. [26] design and verify a supervisory controller for a high-speed train. However, this past research has restricted itself to the supervision of a physical system such as a train or subway system modeled as a DES; as opposed to a physical oriented model, the research herein attempts to characterize, understand, and supervise a service-oriented model for a demand-responsive transportation system.

There has also been a lot of prior work previously that is applicable to intelligent transportation. For instance, in the survey papers [1], [2], [10], [27], including those cited earlier, algorithms based on heuristics, tabu search, constraint model, and mathematical programming have reportedly been developed for the related problems of vehicle assignment and route planning. However, these algorithms are aimed at generating vehicle assignments and route plans that optimize (i.e., minimize or maximize) some quantitative performance specifications, such as some cost or benefit functions. In contrast, the service-operations control problem addressed herein is aimed at regulating the flow of service-related events in passenger land transportation in accordance with qualitative specifications and, thus, is related but incomparable with these existing efforts.

The rest of this paper is organized as follows. Section II reviews the formulation and concepts of the supervisory control theory that are relevant to our research. Section III presents the supervisory design methodology to address the supervisory control problem in transport service operations. Section IV illustrates the design methodology via a simplified but realistic PLanTS. All automaton models for the PLanTS are provided, together with the automata design of a modular supervisor, which ensures proper service operations according to a given set of behavioral specifications. Finally, Section V presents the conclusion and points to some future work. Preliminary versions of this research work appeared in [28], [29].

\(^1\)CTCT design software can be downloaded from the website http://www.control.toronto.edu/people/profs/wonham/wonham.html
II. REVIEW OF SUPERVISORY CONTROL THEORY

The control theory for DES considered in our work is based on controlled automata concepts. The essential concepts and results reviewed are taken from [11] and can also be found in [13]–[15].

A. Discrete-Event Behavior

1) Automaton Model: The behavior of DES, such as the service operations of a PLanTS, and behavioral specifications can be modeled by finite state automata [30] at some appropriate level of abstraction. An automaton is a five-tuple

\[ \mathbf{A} \cong (\Sigma, Q, \delta, Q_m, q_0) \]

in which

- \( \Sigma \): finite set of transitions or event labels;
- \( Q \): finite set of states;
- \( \delta : \Sigma \times Q \rightarrow Q \): state transition function;
- \( Q_m \): finite set of marked states (states indicating the completion of the tasks or sequences of tasks from a control perspective);
- \( q_0 \in Q \): initial state.

Finite state automata are naturally described by directed-transition graphs. In order to represent the automaton \( \mathbf{A} \), a state \( q \in Q \) is identified by a node (represented by \( \bullet \)) of the graph whose edges are labeled by transition labels \( \sigma \in \Sigma \) (represented by \( \cdot \). The initial state \( q_0 \in Q \) is labeled with an entering arrow \( \rightarrow \bullet \), while a marked state \( q_m \in Q_m \) is labeled with an exiting arrow \( \bullet \rightarrow \). When \( q_0 \in Q \) is also a marked state, it is labeled with a double arrow \( \leftrightarrow \).

2) DES as Composition of Automata: Consider an automaton \( \mathbf{G} \) modeling the behavior of a DES. A DES model \( \mathbf{G} \) is usually modeled as a system of several interacting processes, each modeled by an automaton \( \mathbf{G}_i \). To compose several automata \( \mathbf{G}_i \) to obtain the global automaton \( \mathbf{G} \), the idea of synchronous product of automata taken from [11] and [23] is utilized.

\[ \mathbf{G} = \mathbf{G}_1 || \mathbf{G}_2 || \cdots || \mathbf{G}_i \]

where || is the composition operator. To elaborate, consider the case of two automata, i.e., \( i = 2 \). Then, the synchronous product \( \mathbf{G}_1 || \mathbf{G}_2 \) models the behavior \( \mathbf{G}_1 \) and \( \mathbf{G}_2 \) operating concurrently, by interleaving sequences generated by \( \mathbf{G}_1 \) and \( \mathbf{G}_2 \) such that

- events common to both the automata can occur only if each automata is in a state where such an event is defined;
- events that are not common to both the automata may occur as long as they occur in the order defined by the transition functions of \( \mathbf{G}_1 \) and \( \mathbf{G}_2 \), respectively.

If the event sets of \( \mathbf{G}_1 \) and \( \mathbf{G}_2 \) are disjoint (i.e., no common event between the two), the synchronous product reduces to the shuffle product of \( \mathbf{G}_1 \) and \( \mathbf{G}_2 \). For a more formal definition, see Hoare [23].

3) Language Characterizations: The set \( \Sigma^* \) contains all possible finite sequences, or strings, over \( \Sigma \), plus the null string \( \varepsilon \). The definition of \( \delta \) can be extended to \( \Sigma^* \) as

\[ \delta(\varepsilon, q) = q \]
\[ (\forall \sigma \in \Sigma)(\forall s \in \Sigma^*), \quad \delta(s\sigma, q) = \delta(\sigma, \delta(s, q)) \]

The behavior may then be described by two languages: \( I(\mathbf{A}) \), the prefix-closed language generated by automaton \( \mathbf{A} \), and \( L_m(\mathbf{A}) \), the language marked by automaton \( \mathbf{A} \). More formally

\[ I(\mathbf{A}) = \{ s \in \Sigma^* : \delta(s, q_0) \text{ is defined} \} \]
\[ L_m(\mathbf{A}) = \{ s \in I(\mathbf{A}) : \delta(s, q_0) \in Q_m \} \]

By definition, \( L_m(\mathbf{A}) \subseteq I(\mathbf{A}) \) is the subset of strings in \( I(\mathbf{A}) \), which end in any of the states in \( Q_m \) and is a distinguished subset: if automaton \( \mathbf{A} \) represents a DES, then \( Q_m \) is meant to represent completed “tasks” (or sequences of tasks) carried out by the physical process that the model \( \mathbf{A} \) is intended to represent [13]. If automaton \( \mathbf{A} \) represents (or models) a behavioral specification \( K \), then \( K = L_m(\mathbf{A}) \), the behavior of interest.

An automaton \( \mathbf{A} \) is said to be trim if it is accessible (i.e., every state \( q \in Q \) is reachable in \( \mathbf{A} \)) and coaccessible (i.e., every state \( q \in Q \) is coreachable in \( \mathbf{A} \)). A state \( q \in Q \) is reachable in \( \mathbf{A} \) if there exists a string \( w \in \Sigma^* \) such that \( \delta(w, q) = q \) and coreachable in \( \mathbf{A} \) if there exists a string \( w \in \Sigma^* \) such that \( \delta(w, q) \in Q_m \). Note that if automaton \( \mathbf{A} \) is trim, then \( I(\mathbf{A}) = L_m(\mathbf{A}) \), i.e., every string in \( I(\mathbf{A}) \), can be completed to a string in \( L_m(\mathbf{A}) \).

B. Control Formulation and Concepts

As formally described in Section II-A, a DES (or plant) \( \mathbf{G} \) can be modeled by an automaton

\[ \mathbf{G} \cong (\Sigma, \delta, q_0, Q_m) \]

To establish the control framework, the event set \( \Sigma \) is partitioned into disjoint sets of controllable events \( \Sigma_c \) and uncontrollable events \( \Sigma_u \). Controllable events can be prevented (i.e., disabled) or allowed (i.e., enabled) by control, while uncontrollable events cannot be disabled by control and are deemed permanently enabled. The basic problem [13], [15] in supervisory control is to design a supervisory controller whose task is to enable or disable each of the controllable events during its observation of the event sequence generated by DES \( \mathbf{G} \) such that the resultant closed-loop system generates only a subset of \( I(\mathbf{G}) \). Conceptually, a supervisory controller \( \mathcal{S} \) consists of two components

\[ \mathcal{S} = (\mathbf{S}, \mathbf{V}) \]

where supervisor \( \mathbf{S} = (X, \Sigma, \zeta, x_0, X_m) \) is an automaton called recognizer and control law \( \mathbf{V} : X \mapsto 2^\Sigma \) is the state feedback map. In a typical closed-loop configuration, as shown in Fig. 1, the supervisor and DES interact with each other via what is called event-feedback through \( \mathbf{V}(x) \). The automaton \( \mathbf{S} \), as a language acceptor, is driven by the string of events generated by DES \( \mathbf{G} \) and fed back to \( \mathcal{S} \) that, in turn, with \( \mathbf{S} \) in state \( x \in X \), the next set of events \( \sigma \in \Sigma \) of DES \( \mathbf{G} \) are subjected to the control law \( \mathbf{V}(x) \) such that only events in \( \mathbf{V}(x) \) are enabled. Note then that such a supervisor may be dynamic in the sense that
not all strings of events of the DES $\mathbf{G}$ that lead to the same state
will necessarily result in the same control action at that state. In
a supervisor, the strings generated by the controlled system will
always bring its automaton $\mathbf{S}$ to a defined state in $\mathcal{X}$ through the
transition function $\zeta$ (see [13]). Consequently, the closed-loop system
$\mathcal{S}/\mathbf{G}$, called the supervised or controlled DES, is
another automaton that is defined as

\[ \mathcal{S}/\mathbf{G} = \text{Accessible}(\mathcal{X} \times Q, \Sigma_{\zeta}, (\zeta \times \delta)^{V}, (x_{0}, q_{0}), \mathcal{X} \times Q_{m}) \]

where $(\zeta \times \delta)^{V} : \Sigma \times \mathcal{X} \times Q \rightarrow \mathcal{X} \times Q$ is defined as shown in
the equation at the bottom of the page. For a proper supervisory controller
$\mathcal{S}$, at $(x, q) \in \mathcal{X} \times Q$,

\[ \sigma \in V(x) \text{ if } \zeta(\sigma, x) \text{ and } \delta(\sigma, q) \text{ are both defined} \]

such that

\[ \Sigma_{\zeta}(q) \subseteq V(x) \]

where $\Sigma_{\zeta}(q) = \{ \sigma \mid \delta(\sigma, q) \text{ is defined and } \sigma \in \Sigma_{\zeta} \}$. Henceforth,
unless otherwise stated, a supervisory controller is assumed to be proper.

Let $\Sigma(q) = \{ \sigma \mid \delta(\sigma, q) \text{ is defined and } \sigma \in \Sigma \}$. Then, with $\mathbf{S}$ in state $x \in \mathbf{X}$ and $\mathbf{G}$ in state $q \in Q$

\[ V(x) = \Sigma(q) - \overline{V(x)} \]

where $\overline{V(x)}$ defines the subset of events $\sigma \in \Sigma_{\zeta}$ disabled at
$x \in \mathcal{X}$.

In general, a supervisory controller $\mathcal{S}$ can be decomposed into
two or more subsupervisors, giving rise to modular supervision.
In the case of a modular supervisor $\mathcal{S}$ consisting of two supervisory controllers $\mathcal{S}_{1}$ and $\mathcal{S}_{2}$, given by

\[ \mathcal{S} = \mathcal{S}_{1} \land \mathcal{S}_{2} \]

we denote $\mathcal{S}$ by

\[ c_{1}(x_{1}, x_{2}) \text{ and } V(x) = V_{1}(x_{1}) \cap V_{2}(x_{2}). \]

Then, with $\mathbf{S}$ in state $x = (x_{1}, x_{2}) \in \mathcal{X}$ and $\mathbf{G}$ in state $q \in Q$

\[ V(x) = \Sigma(q) - \bigcup_{i=1}^{2} V_{i}(x_{i}) \]

and, in this sense, we say that these two subsupervisors jointly enable (or disable) events. The above notion of modular supervision can be readily extended to more than two subsupervisors.

The behavior of the supervised DES is described by the languages
$L(\mathcal{S}/\mathbf{G}) = L(\mathbf{S}) \cap L(\mathbf{G})$ and $L_{m}(\mathcal{S}/\mathbf{G}) = L_{m}(\mathbf{S}) \cap
L_{m}(\mathbf{G})$. Clearly, $L_{m}(\mathcal{S}/\mathbf{G}) \subseteq L(\mathcal{S}/\mathbf{G})$. Supervisor $\mathcal{S}$ is said to be nonblocking if

\[ L_{m}(\mathcal{S}/\mathbf{G}) = L(\mathcal{S}/\mathbf{G}) \]

i.e., every string in $L(\mathcal{S}/\mathbf{G})$ can be completed to a string in
$L_{m}(\mathcal{S}/\mathbf{G})$.

C. Nonblocking Supervisory Control Problem

The basic supervisory control problem considered [13], [15] is as follows. Given a DES automaton $\mathbf{G}$ over an event set $\Sigma$, with the associated languages $L(\mathbf{G})$ and $L_{m}(\mathbf{G})$ and a behavioral specification (or control objective) $K \subseteq \Sigma^{*}$, the supervisory control problem is to find a (proper) nonblocking supervisory controller $\mathcal{S}$ such that $L_{m}(\mathcal{S}/\mathbf{G}) \subseteq K$ (or we say the DES under nonblocking control $\mathcal{S}/\mathbf{G}$ satisfies specification $K$).

What this framework captures is a DES (finite-state machine)
$\mathbf{G}$ and a behavioral specification describing a desired or legal behavior $K$, with a supervisory controller being sought so that only desirable sequences of $L_{m}(\mathbf{G}) \cap K$ are generated.

1) Centralized Solution: To provide a solution to the above problem, the notion of language controllability is introduced. A language $M \subseteq \Sigma^{*}$ is said to be controllable with respect to $\mathbf{G}$ if $M \Sigma_{\zeta} \cap L(\mathbf{G}) \subseteq M$, where $M \Sigma_{\zeta} = \{ s \sigma \mid s \in M \text{ and } \sigma \in \Sigma_{\zeta} \}$. This controllability condition requires that if any $w \sigma \in M$, i.e., any prefix of a string in $M$, followed by an uncontrollable event $\sigma \in \Sigma_{\zeta}$ is in $L(\mathbf{G})$, then $w \sigma \in M$, i.e., $w \sigma \in L(\mathbf{G})$ must also be a prefix of a string in $M$.

Now, suppose there is a $s \in K$ such that $s \sigma \notin L(\mathbf{G})$ is not in $K$. Then, $s \sigma \notin K$. So, if $\sigma \in \Sigma_{\zeta}$, no $\mathbf{S}$ exists that can exercise control to guarantee $L_{m}(\mathcal{S}/\mathbf{G}) = K$. In this case, we say that $K$ is not controllable with respect to DES $\mathbf{G}$, but a largest or supremal controllable sublanguage (possibly $\emptyset$) of the marked language $K \subseteq L_{m}(\mathbf{G})$ with respect to $\mathbf{G}$ can always be found [14]. This is denoted by $\operatorname{sup}(C(K, \mathbf{G}) \subseteq K$ and is a solution language $L(\mathbf{S})$ for the nonblocking supervisory controller $\mathcal{S}$ such that $L_{m}(\mathcal{S}/\mathbf{G}) \subseteq K$. To emphasize, the supervisory controller $\mathcal{S}$ is maximally permissive, i.e., it disables events in DES $\mathbf{G}$ only when absolutely necessary, as evident from the fact that the sublanguage $\operatorname{sup}(C(K, \mathbf{G}) \subseteq K$ generated as a result is the largest.

2) Modular Solution: Let $A_{1}$ and $A_{2}$ be two automata. Then, the prefix-closed languages $L(A_{1})$ and $L(A_{2})$ are said to be nonconflicting provided that $L_{m}(A_{1}) \cap L_{m}(A_{2}) = L(A_{1}) \cap L(A_{2})$. If $A_{1} = (A_{1}, V_{1})$ is a supervisory controller (of the form (1)) for $A_{2}$, then $A_{1}$ is a nonblocking supervisor for $A_{2}$, provided that $L(A_{1})$ and $L(A_{2})$ are nonconflicting.

With the above definitions, the result on modular supervision, readily extendible to more than two subsupervisors, may be given as follows if

\[ (\zeta \times \delta)^{V}((\sigma, x), q) = \begin{cases} (\zeta(\sigma, x), \delta(\sigma, q)), & \text{if } \sigma \in V(x) \text{ and both transitions are defined} \\ \text{undefined}, & \text{otherwise} \end{cases} \]
1) $L(S_1)$ and $L(S_2)$ are each controllable with respect to DES $G$
2) $L(S_1) \cap L(S_2)$ and $L(G)$ are nonconflicting.

Then, $S_1 \land S_2$ is a nonblocking (modular) supervisory controller for DES $G$.

A generic software CTCT [11] is now available to support the automatic synthesis of supervisory controllers. The DES and behavioral specification are input as automata to the software CTCT; operations supported by CTCT include the composition of automata, supremal controllable sublanguage computation for a given behavioral specification with respect to the DES of interest, as well as a nonconflict test between two languages.

III. DESIGN METHODOLOGY

To encompass the components of the supervisory control framework (see Fig. 2), the methodology to facilitate the planning for online supervision of a PLanTS consists of the following steps.

Step 1) Model the PLanTS and Behavioral Specifications

Both the service-operations process behavior of a typical traveler and vehicle and the behavioral specifications are modeled as DESs translated into the form of automata. The modeling of the service-operations processes should also, through appropriate abstractions, take into account of the dynamic vehicle assignment (and routing) capabilities [27], [31] that an underlying planner is assumed to possess. More will be said about this planner later.

Step 2) Synthesizing the Supervisor and Control Law

Taking into account the supervisory control architecture adopted (for example, centralized or modular), the automata representing the system of service-operations processes to be controlled and the automata representing the corresponding behavioral specifications are fed to the control synthesis program CTCT [11], which will tell us whether it is possible for the system to behave within the specifications and return the supervisor(s) and the corresponding control law(s) that ensure that the controlled behavior of the system is maximally permissive within the latter specifications.

Step 3) Simulating the Supervisor and Control Law

The supervisory control system is simulated to evaluate its effectiveness in that it takes appropriate actions in accordance to the supervisor(s) and corresponding control law(s). By default, all controllable events are assumed to be disabled. Let automaton $S$ denote a supervisor and $G$ denote the PLanTS model. Then, the simulation allows an enabled event as input (to simulate its occurrence) and control evaluation updates the current state $x$ of $S$ to, say, $x'$ and subsequently produces the corresponding control $V(x')$—the updated (online) permission set—only from which the next enabled event can be input. Only transitions in $V(x)$ are events enabled or permitted to occur and their occurrences never result in any eventual contradiction of the behavioral specifications. In this manner, $V(x)$ keeps the system operations within the behavioral specifications.

Generally speaking, specifications should encompass the most desired2 dynamic but orderly conditions under which a subset of vehicles in a given fleet is chosen, from which the admitted travel requests can be assigned to based on quantitative specifications asserting the desired quality of service to be achieved. These quantitative specifications are, of course, to be met by the underlying planner; the extent to which they would be met depends on various factors that the planner considers, such as the vehicle-assignment and route-planning techniques used, the task-execution capabilities of the vehicle fleet, and the accurate update of traffic information by the surveillance system. How the many existing algorithms—basic and applied—as reported in the literature (see [2], [10], [27], [32], and the references contained therein) might be exploited to address the related optimization problems of vehicle assignment and route planning under close-loop supervision are beyond the scope of this paper. In the terminology of DES [11], the planner can be viewed as part of the underlying “decision-making engine” of PLanTS that is capable of some choices of spontaneous occurrences of events, including the assignment and reassignment events denoted, respectively, by $as_j^f$ and $ras_j^f$, as precisely defined in Table II of Appendix I. These events are decision “outputs” of the planner underlying our illustrative PLanTS model, as described in Section IV.

IV. SIMPLE AUTOMATED PLANTS

In this section, the design of a modular supervisory controller for a PLanTS using the methodology discussed in Section III is presented in detail.

A. Problem Description

In the scenario considered, travel requests are randomly initiated, geographically distributed, and require immediate or emergency attention. Each request is associated with only one person. The transport fleet is homogeneous; it consists of a small fleet of $N$ vehicles and has a small seat capacity of $C_s$ requests per vehicle and an assignment capacity of $C_a$ requests per vehicle. The assignment-capacity $C_a$ of a vehicle refers to the maximum number of requests to which it can be assigned, but are yet to be fetched by the vehicle.

1) PLanT System Components: The main behavioral components are described as follows.

1) Initiator Behavior: a simple process that initiates the start and end of the transport service operations.

2) Vehicle Behavior: from an initial shutdown or idling state, each vehicle can be service started or restarted. In the service-ready state, two possibilities are the vehicle ending its service operations or breaking down during operation. By the former occurrence, the vehicle returns to its idling state. By the latter, it falls into the breakdown state; repair and maintenance are then needed to get it to

2What is meant by “most desired” is decidedly a subjective opinion of the system analyst.
return to its initial state. In any state, it is possible that the vehicle gets trapped in or out of a traffic jam. The vehicle’s task is considered completed once it ends its operations and is out of the traffic jam.

3) **Traveler Behavior**: from the service-operations viewpoint, once a traveler is admitted for service, his request can possibly be cancelled either by the system or himself, or can be assigned to a particular vehicle by the underlying planner, after which the request cannot be cancelled unless the timeout set occurs before he boards the vehicle. While in a vehicle, the traveler has the options of making an urgent call (for another vehicle’s service) or leaving the vehicle. The service task is considered completed once the traveler exits the system.

2) **Resource Limits** \( N, C_s, \) and \( C_a \): In our current work, we consider the following limits: number of vehicles \( N = 3 \), vehicle seat capacity \( C_s = 2 \), and vehicle assignment capacity \( C_a = 1 \). These limits determine the upper bound on what we term the system-processing limit, as discussed next.

3) **System-Processing Limit** \( M \leq N(C_s + C_a) \): \( M \) is assumed to be the maximum number of travelers that can be concurrently serviced without degrading the performance of the underlying planner. Then, in general, the upper bound of the limit \( M \), denoted \( M_{upp} \), is \( N(C_s + C_a) \), the total of the maximal service capacity \( (C_s + C_a) \) of each vehicle that can possibly be concurrently utilized. By the resource limits in our current work, \( M_{upp} = 9 \), but we assume that \( M = 4 \leq 9 \).

4) **Behavioral Specifications**: The specifications to be conformed to via supervisory control are described below. In the context of PLanTS, these specifications are to be satisfied without blocking or preventing the completion of any of the following mandatory tasks:

- *emptying* of the service queue and all service vehicles;
- *service termination* of all service vehicles in normal traffic conditions.

1) **Service Startup/Shutdown Operations**

a) Request Admission: Once the system operation start has been initiated, all vehicles must be ready for service first before any travel request can be admitted.

b) Service Continuity: During the system operation, no vehicle is allowed to end its individual service operations until the system operation stop has been initiated, in which case no more travel requests will be admitted and no vehicle will be service restarted.

2) **Service Incident-Response Operations**

a) Vehicle Traffic Jam: When a vehicle is caught in a traffic jam, no task (i.e., travel request) already assigned to another vehicle is allowed to be reassigned to the vehicle until it is out of the jam.

b) Limit on Service Capacities:

i) Vehicle Seat Capacity: number of travelers (tasks-in-execution) in a vehicle must not exceed its seat capacity of \( C_s \).

ii) Vehicle Assignment Capacity: The number of (pending) assignments for a vehicle must not exceed \( C_a \). Once assigned to a particular vehicle, a travel request must not be serviced by any other vehicle unless it is reassigned or timeout occurs.

A vehicle will not end its service operations when its service capacity (i.e., either seat or assignment capacity) is not empty.

c) Emergency Requests: A traveler can make an emergency call to request service by another vehicle only when the vehicle servicing him has broken down.

d) Vehicle Breakdown: When a vehicle breaks down, no further assignment or reassignment will be given to it, nor will any traveler be allowed to enter it unless it is service restarted.

e) Fleet Service-Diligence: Once system operation is ready, no vehicle is allowed to end its individual service operations when the pending travel request queue is not empty.

B. **Modeling for PLanTS and Behavioral Specifications**

Formalizing, the PLanTS’s component processes and the behavioral specifications introduced above are embodied in au-
Fig. 5. Mechanism underlying traveler $j$, $1 \leq j \leq M$.

$\text{Selfloop} = \Sigma - \{tstart, tstop, ad_j, st_i, end_i\}$

Fig. 6. Vi_SUDSP: service startup/shutdown.

$\text{Selfloop} = \Sigma - \{inj_i, out_j, ras_j\}$

Fig. 7. Vi_TJMSP: vehicle traffic jam.

$\text{Selfloop} = \Sigma - \{tstart, tstop, ad_j, st_i, end_i\}$

Fig. 8. Vi_SEATSP: vehicle seat capacity.

$L_p = \{enter^i_p, enter^k_p, tout_n, ras^k_n | n \neq p\}$

$O_p = \{as^i_p, ras^i_p\}$, $I_p = \{enter^i_p, tout_p, ras^k_p\}$

$\text{(Fixed } i, \text{ Variable } j)\quad \text{Selfloop} = \Sigma - \{end_i, enter^j_i, leave^j_i, ecall^j_i\}$

Fig. 9. Vi_AGNSP: vehicle assignment capacity.

$\text{Selfloop} = \Sigma - \{st_i, dwn_i, ecall^j_i\}$

Fig. 10. Vi_EMSP: emergency requests.

$\text{Selfloop} = \Sigma - \{st_i, dwn_i, enter^j_i, as^j_i, ras^j_i\}$

Fig. 11. Vi_DWNSP: vehicle breakdown.

$\text{Selfloop} = \Sigma - \{st_i, dwn_i, enter^j_i, as^j_i, ras^j_i\}$

tomata shown in Figs. 3–5 and Figs. 6–12, respectively, and listed in Table I. The event definitions are given in Table II.

The vehicle identification number (ID) is $i$, $1 \leq i \leq N$, and the admitted traveler ID is $j$, $1 \leq j \leq M$. In the PLanTS considered, $N = 3$ and $M = 4$. In the “Code No.” column of Table II, the CTCT code representations of the various events in PLanTS are located. For instance, event $a^2_3$—“Traveler 1 assignment to Vehicle 3”—is arbitrarily represented in CTCT as 115, an odd number denoting that it is controllable, while event $a^2_3$—“Traveler 3 leaves Vehicle 2”—is represented as 362, an even number denoting an uncontrollable event.

In each of the specification automata (Figs. 6–12), self-loops (not shown) must be adjoined to account for all events that are irrelevant to the specification, but may be executed in the PLanTS model.

The trim automaton model $\mathbf{G}$ for PLanTS is a composition of its component processes via synchronous product [11], [23] as discussed in Section II-A.2. The completion of the mandatory tasks of the PLanTS is represented by a marked state, which is formed by collecting together the marked state in each of the PLanTS’s component processes. The “Number” column in Table I indicates the number of automata in each category for the PLanTS with $N = 3$ and $M = 4$. 
C. Supervisor and Control Law Synthesis

In this section, we first discuss, in relation to the notion of language nonconflict, the special structures of the automata representing the PLanTS model and all the behavioral specifications considered. These structures help us to analytically establish the nonblocking property of our modular supervisor design.

1) Special Structures and Nonconflicting Languages: The trim structure of the automaton model $G$ for PLanTS is such that there exists a string of uncontrollable events that leads any unmarked state in the structure to a marked state (which is the initial state) of the system. This property is formalized as follows.

Property 1: At any unmarked state $q \in Q - Q_m$ of the DES model $G = (\Sigma, Q, \delta, Q_m, q_0)$, there exists a string $t \in \Sigma^*$ such that $\delta(t,q) \in Q_m$.

In the following, Property 1 and the notion of language nonconflict [11] (reviewed in Section II-C.2) are used to establish Theorem 1. As the subsequent section will show, this theorem allows us to analytically establish the second condition (as in Section II-C.2) of nonblocking modular control synthesis [11], [15] for PLanTS, without directly testing the property of language nonconflict that, for the whole set of specification automata considered (as shown in Appendix I-B), is found to be infeasible for the CTCT software to verify, due to the large state space complexity of intersecting these behavioral specifications.

Theorem 1: A controllable prefix-closed language $L(A)$, with automaton $A$ having the property of $G$-closure [13], i.e.,

$$\text{if } s \in L(A) \cap L_m(G), \text{then } s \in L_m(A)$$

is nonconflicting with (the prefix-closed language of) a DES model $G$ that satisfies Property 1.

Proof: Given that the prefix-closed language $L(A)$ is controllable with respect to $G$, then suppose that $L(A)$ conflicts with $L(G)$; this means that there exists a prefix $s \in L(A) \cap L(G) - L_m(A) \cap L_m(G)$; $\delta(s, q_0) \in Q - Q_m$ i.e., the prefix $s$ is not marked by automaton $G$ since automaton $A$ is $G$-closed. By Property 1, there exists a $t \in \Sigma^*$ such that $s t \in L_m(G)$. But $s t \not\in L_m(A)$ (because prefix $s$ cannot be completed to a marked string in $L_m(A) \cap L_m(G)$). Clearly, $s t \not\in L(A)$ since automaton $A$ is $G$-closed, thus contradicting the fact that $L(A)$ is controllable.

Remark 1: It is easy to redesignate some unmarked states as marked states to transform a given automaton into an automaton $A$ that is $G$ closed. Therefore, we can readily infer from Theorem 1 that an arbitrary DES model design that satisfies Property 1 will guarantee nonblocking in a supervisory controller $(A, V)$ that exists.

2) Nonblocking Modular Synthesis: The trim automaton $G'$ obtained for PLanTS has 559 872 states and 11 197 440 transitions! Fortunately, we could find a simpler trim model $G''$ for PLanTS, having 110 592 states and 2 875 392 transitions, which renders the control computation using CTCT feasible. The control synthesis with respect to model $G''$ is the same as that with respect to model $G$, the elaboration of which is given in Appendix I. The automata representing the behavioral specifications, as shown in Figs. 6–12, are arbitrarily referenced herein as $S_k$. Using CTCT on model $G''$, each prefix-closed language $L(S_k)$ is found to be controllable with respect to PLanTS model $G$. Hence, the overall prefix-closed language $(\bigcap_{k \in \mathbb{N}} L(S_k))$ is also controllable [11]. Let $L(A) = \bigcap_{k \in \mathbb{N}} L_m(S_k)$, with automaton $A$ being the (reachable) Cartesian product of all $S_k$; hence, $L_m(A) = \bigcap_{k \in \mathbb{N}} L_m(S_k)$. By inspection, each simple automaton $S_k$ is $G$-closed; it then follows that automaton $A$
Fig. 15. Vi_SEATCON: vehicle seat-capacity supervisors.

Fig. 16. Vi_AGNCON: vehicle assignment-capacity supervisors.

Fig. 17. Vi_EMCON: emergency-requests supervisors.

Fig. 18. Vi_DWNCON: vehicle-breakdown supervisors.

Fig. 19. FLT_DILCON: fleet service-diligence supervisor.

is also G-closed and, hence, by Theorem 1, \( \bigcap_{k=1}^{\infty} L(S_k) \) is nonconflicting with G. Thus, according to the modular control result [11], [15] (reviewed in Section II-C.2), \( \bigwedge_{k=1}^{\infty} S_k \), where \( S_k \), in coded form, are as shown in Figs. 13–19, can serve as a modular supervisory controller that, when acting in synchrony with PLanTS, is nonblocking with respect to those mandatory tasks defined at the beginning of Section IV-A.4 and, therefore, generates the largest (marked) sublanguage of the PLanTS model that lies within the overall specification \( L_m(A) \).

D. Supervisor and Control Law Simulation

To illustrate that the supervisory subcontrollers thus obtained jointly enable or disable events correctly, the following case is simulated.

Proper Startup/Shutdown: The following two sequences test the specification \( \bigcap_{k=1}^{3} L_m(\text{Vi}_\text{SUDSP}) \), where automaton \( \text{Vi}_\text{SUDSP} \) is shown in Fig. 6. Note that these test (prefix) sequences are feasible sequences of PLanTS and violate only the specification under test.

By sequence 1, \( \text{start}, \text{s1}_1, \text{s2}_1, \ldots \), the system operation start is successfully initiated, but only vehicles 1 and 2 are ready...
for service when travel request 2 is admitted, thus violating the specification that requires all vehicles to be ready first after startup before any request can be admitted. Indeed, the simulation succeeds because only the events $\text{start}$ can be input in that order; event $\text{end}$ cannot be input thereafter because it is disabled.

By sequence 2, $\text{start}, \text{st}_1, \text{st}_2, \text{st}_3, \text{end}_1, \ldots$, the system operation start is successfully initiated and completed with all vehicles ready for service when vehicle 1 ends its service operations, thus violating the specification that requires an order of shutdown (event $\text{stop}$) to be given first before any vehicle could end its individual operations. Again, the simulation succeeds this time because event $\text{end}_1$ cannot be input at that respective instance.

V. CONCLUSION AND FUTURE WORK

Our initial study reported herein suggests that the theory of supervisory control [11] provides a useful framework for capturing the high-level structure of a dynamic service-operations manager for a PLanTS. The structure is realized in the form of a permission-based supervisory policy. The supervisory control theory offers a simple methodology for describing the event-based characteristics of a PLanTS and for determining the existence of nonblocking supervision. If nonblocking supervision exists, it is guaranteed that the PLanTS under such control will satisfy the behavioral specifications in the least restrictive manner without preventing (or blocking) any marked service-operations task from completion. Besides, changing the desired behavior of the PLanTS is a matter of adding or removing a behavioral specification.

In the domain of planning for passenger transport service, the supervisory control framework serves to provide a new basis to support the systematic development for online supervision of PLanTS. In this research, a modular approach using at least two subsupervisors to jointly track the behavior of PLanTS is considered. Importantly, the modular supervisor designed is structurally very simple and issues permissions $V(\chi)$ that form the necessary logical condition for the online validity of the vehicle assignment and route plans generated by the underlying planner.
The state-space complexity that arises from the intersection of all $L(S_k)$, where $S_k$ refers to the automaton of each behavioral specification given in Figs. 6–12, prohibits the use of the CTCT software to verify the property of nonconflict between $\bigcap_{k=1}^{N} L(S_k)$ and $L(G)$. But, fortunately, by exploiting the special structures of the PLanTS model $G$ and all $S_k$, as formally discussed in Section IV-C, the property of nonconflict and, hence, nonblocking for modular supervision, could be verified. The structural property of the PLanTS model provides a practical guide to general DES model design, which guarantees the nonblocking property in a supervisory controller that exists.

It is hoped that, in the future, progress on supervisory control will render the complexity problem more manageable, so that a more capable transport-service planner in terms of a larger $N$, $C_s$, and $C_a$ may be deployed using the proposed supervisory control approach. Note that in our illustrative supervisory design for the PLanTS, the system limit $M$ is restricted to only four for a fleet of three vehicles ($N = 3$). Our conjecture about the design is that it is scalable in the sense that the controllability and nonconflicting properties of the corresponding modular supervisor can be preserved for the same set of behavioral specifications and PLanTS, but extended accordingly to account for a larger $N > 3$ and $M > 4$.

Finally, the hybrid architectural issues of developing the underlying planner to perform lower-level tasks, such as distributed vehicle assignment and route planning under close-loop supervision would need to be defined and investigated.

### Appendix I

#### On PLANTS Modeling

Composing (via synchronous product [11]) the trim automaton for Traveler $j$ [as shown in Fig. 4(b)] and all the $N$ automata (as shown in Fig. 5) yields another trim automaton (conventionally referred to as $\mathbf{T N}_{j}$) for Traveler $j$, which includes the natural constraint that Traveler $j$ can only leave the vehicle he has entered (not any other vehicle!). We call these automata entry–exit laws when we refer to the kind of constraint they impose, via composition, on the automaton for Traveler $j$ [as shown in Fig. 4(b)]. Referring to our illustrative PLanTS with $N = 3$ and $M = 4$, incorporating the set of $M$ such composed automata $\mathbf{T N}_j$ (each of six states and 18 transitions) instead via synchronous product results in an overall automaton model $G$ for PLanTS having 559 872 states and 11 197 440 transitions. For the behavioral specifications that we consider, the control synthesis with respect to this resultant PLanTS model $G$ is beyond the computational capacity of CTCT running on a personal computer with a 200-MHz CPU and 64-MB memory. Fortunately, the feasibility of control computation using CTCT can still be achieved by considering only the trim automaton that has a smaller state size [of four states and 18 transitions, as shown in Fig. 4(b)] as the behavioral process of Traveler $j$. The $N$ entry–exit laws consist only of events in this trim automaton for Traveler $j$ and composing all of them yields another trim automaton $\mathbf{T N}_j$ for Traveler $j$ that only excludes the obviously impossible situations of Traveler $j$ in one vehicle leaving another vehicle. Hence, these $N$ laws effectively form a component of the PLanTS’s underlying engine for the behavioral process of Traveler $j$. Thus, it is practically inconsequential to supervision (that exists) whether or not these automata-theoretic laws are directly incorporated into the overall model for PLanTS, but by not doing so, a PLanTS model $G'$ (of 110 592 states and 2 875 392 transitions) with a much smaller state size results; this renders our control synthesis of individual specification automata using CTCT computationally feasible.

### Acknowledgment

The authors would like to thank the anonymous reviewers for their critical but constructive comments on the review version of this manuscript. The responsibility for the integrity of the work, of course, remains solely with the authors.

### References


