3-18-2008

Discrete Event System Modeling Of Demand Responsive Transportation Systems Operating In Real Time

Daniel Y. Yankov
University of South Florida

Discrete Event System Modeling Of Demand

Responsive Transportation Systems Operating In Real Time

By

Daniel Y. Yankov

A Dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Industrial and Management Systems Engineering
College of Engineering
University of South Florida

Date of Approval:
March 18, 2008

Major Professor: Ali Yalcin, Ph.D.

Natasha Jonoska, Ph.D.
Aliaksei Savachkin, Ph.D.
Kimon Valavanis, Ph.D.
Susana Lai Yuen, Ph.D.

Keywords: intelligent transportation systems, supervisory control, air charter service, aero medical evacuation, local supervisors, concurrent subsystems

Acknowledgements

I would like to thank my advisor Dr. Ali Yalcin for his support and mentoring throughout the course of this research.

I would like to thank my committee members Dr. Natasha Jonoska, Dr. Kimon Valavanis, Dr. Alex Savachkin and Dr. Susana Lai-Yuen for their kind reviews and support, and for the wisdom they possess which they allowed me to benefit.

Last but not the least, I would like to thank my family and friends, without them the journey would neither start nor complete.

Table of Contents

List of Tables

List of Figures

Discrete Event System Modeling of Demand Responsive Transportation Systems Operating in Real Time

Daniel Y. Yankov

ABSTRACT

Demand responsive transportation is a variable route service of passengers or freight from specific origin(s) to destination(s) in response to the request of users. Operational planning of DRT system encompasses the methods to provide efficient service to the passengers and to the system operators. These methods cover the assignments of vehicles to transportation requests and vehicle routings under various constraints such as environmental conditions, traffic and service limitations. Advances in the information and communication technologies, such as the Internet, mobile communication devices, GIS, GPS, Intelligent Transportation Systems have led to a significantly complex and highly dynamical decision making environment.

Recent approaches to DRT operational planning are based on "closed information loop" to achieve a higher level of automation, increased flexibility and efficiency. Intelligent and effective use of the available information in such a complex decision making environment requires the application of formal modeling and control approaches, which are robust, modular and computationally efficient.

In this study, DRT systems are modeled as Discrete Event Systems using Finite Automata formalism and DRT real time control is addressed using Supervisory Control Theory. Two application scenarios are considered; the first is based on air-charter service and illustrates uncontrolled system model and operational specification synthesis. The automatic synthesis of centralized and modular supervisors is demonstrated. The second scenario is a mission critical application based on emergency evacuation problem. Decentralized supervisory control architecture suitable for accommodating the real-time contingencies is presented. Conditions for parallel computation of local supervisors are specified and the computational advantages of alternative supervisory control architectures are discussed.

Discrete event system modeling and supervisory control theory are well established and powerful mathematical tools. In this dissertation, they are shown to be suitable for expressing the modeling and control requirements of complex and dynamic applications in DRT. The modeling and control approaches described herein, coupled with the mature body of research literature in Discrete Event Systems and Supervisory Control Theory, facilitate logical analysis of these complex systems and provide the necessary framework for development of intelligent decision making tools for real time operational planning and control in a broad range of DRT applications.

Chapter One

Introduction

DRT passenger services are public transportation services characterized by flexible routing and scheduling of relatively small capacity vehicles (occupancy of up to 20 persons) to provide shared-occupancy and personalized transportation on demand. The role of DRT services has changed dramatically in recent years. For example, rural transit, which is a wide-spread DRT service, was limited to a type of social service transportation for a specific set of clients who primarily traveled in groups to common meal sites, work centers for the disabled, or clinics in larger communities. Service schedules and passenger assignments were developed and augmented manually in a preset calendar. Due to the lack of advanced communication and information technologies, the early DRT systems tended to operate as advanced reservation systems with some service providers requiring users to make a reservation at least 24 hours in advance of their travel. Since it took hours to build the schedule, any last-minute changes could wreak havoc with the operational planning of the dispatch office. Nevertheless, given these parameters, a manual scheduling system worked for small DRT systems.

Lave at al. (1996) report that the advanced reservation DRT operation has been associated with significantly low system productivity. Despite the problems, such DRT systems allocated capacity easily and are less complex to implement than real time

reservation systems. However, for a number of the passenger groups, such as job commuters and clinic patients, the 24-hour preplanned schedule is not viable. They need a system that can take their request when they are ready. Workers and commuters especially need a system that is reliable and robust.

Although DRT service is user-friendly because of its door-to-door capability and semiprivate, comfortable vehicles, its adoption has not been widespread due to the relatively high cost of operation. DRT is a labor intensive mode of transportation with costs comparable to the taxicab, due to inherently low passenger productivity (passengers per vehicle-hour), (Lave at al. 1996). As a result, DRT service is most commonly offered by social service agencies to transport their clients, by transit districts experiencing high enough passenger transportation demand, or by counties and cities for persons with special needs or qualifying conditions.

The 1990 Americans with Disabilities Act (ADA) requires every U.S. transit agency operating in fixed-route transit to provide complementary DRT for persons with disabilities within their service areas without advanced reservation. Thus, the ADA mandate is causing an expansion of the number of DRT services and growth in the size of the existing services. This growth motivated the search for more cost-effective means of operating DRT systems. One promising means of improving the cost effective performance of demand-responsive transit is the use of latest developments of information and communication technologies.

Contemporary DRT systems accept telephone or internet requests for both immediate and advance reservation service; develop a continually changing set of vehicle

schedules that accommodate these trip requests, and route vehicles to the appropriate passenger origin and destination locations in accordance with the schedule. Because both the trip requests and the vehicle scheduling and routing decisions occur in real time, DRT control problem becomes complex even in systems where small number of vehicles and trip requests are involved.

One of the most critical, complex and dynamic application domain of DRT service is the military aeromedical regulation and evacuation (ARE) of patients to medical treatment facilities (MTFs). Doctrinally during both wartime and piece, patients requiring extended treatment must be evacuated by air to a suitable MTF. The process of routing and scheduling the required aeromedical evacuation flights (missions) and assigning patients to suitable missions is a critical part of the *evacuation* planning and execution, Sadeh and Kott (1996).

The major challenge in the design of any DRT operation is the choice between the level of efficiency and level of quality of the service. Service quality ranges from the most costly exclusive-ride taxi service, in which only one person rides at a time, to trips in which vehicles are shared, and each passenger may have to ride longer than is needed for his/her trip while the vehicle drops and picks up other riders. Assigning many passengers to a vehicle results in increased efficiency due to minimizing the total distance traveled by the vehicle and smaller vehicle fleet required. However, high passenger loads lower the quality of the service by increasing the average ride time and the variability of promised pickup and arrival times. These trade-offs are usually determined by specifying

minimum service levels in terms of the longest ride times allowable and the maximum lateness for a promised pickup or arrival.

With every service request the system operator obtains the parameters of the desired trip from the passenger - pickup point, drop-off point, desired pickup or delivery time, number of passengers, and any special requirements (e.g. wheelchair accessibility), and then communicates to the passenger whether the system is able to accommodate the trip request with these specific parameters and, if so, when a vehicle will arrive. The process of scheduling individual service requests while the customer is on the phone or using the Internet is called *real time* or *online* scheduling. This term refers to a scheduling system in which some means of accepting or denying a trip request is based on available system capacity and, if a request is accepted, an estimated time of arrival of the vehicle is given to the requester, usually within a specified time window. With the online service the requests are accepted during the travel of the vehicles and are to be inserted into their current schedules. Hence, vehicle fleet's routing and rerouting are to be done in real time, as well. Therefore, with the online communication DRT service experiences real time operational dynamics that necessitates higher level of automation, flexibility and integration of the system development. To achieve such a development, more formal approaches of system design must be applied.

We represent DRT systems as discrete event systems (DESs) where system models capture both the low level dynamics (such as infrastructure conditions, current status of vehicles) and high level dynamics (such as service demand requests) of system

evolution. Supervisory Control Theory based on Finite Automata formalism is applied to provide real time control of DRT service as supervisory control of DES.

The remainder of this dissertation is organized as follows: Chapter Two presents a literature review of the operational planning methodologies for DRT service. Chapter Three introduces the research problem domain, motivation, research goal and objectives. Chapter Four presents an introduction to Discrete Event Systems, Finite Automata formalism and Supervisory Control Theory. The possible architectures of decentralized supervisors and the conditions for their nonblocking behavior are discussed. In Chapter Five first a taxonomy of DRT systems is introduced and a framework of DRT operation modeling as DES is presented. A simple air charter system is used to illustrate the system modeling and the synthesis of centralized and modular supervisors. Chapter Six discusses the decentralized control of concurrent DESs. The computation of the local supervisors and the synthesis of the global one are illustrated with the control of a small aeromedical evacuation system. In Chapter Seven the completed work is summarized and the future research issues are discussed.

Chapter Two

Related Literature Review

The presented literature survey first reviews the fundaments of Operations Research problems related to DRT operation, covers the developed heuristic approaches for solving these problems, and reviews the recent intelligent transportation systems' methods in DRT operational planning and real time control. We limit our review to deterministic DRT problems and solution approaches, and do not cover stochastic methods. The emphasis of the review is on highlighting the advantages of the decentralized methods over the centralized ones in the operational planning of dynamical and complex DRT systems.

2.1. DRT related OR problems

The OR literature contains numerous studies addressing DRT related problems. In most of the works the Vehicle Routing Problem with Pickup and Delivery (VRPPD) represents the mathematical fundaments of DRT and henceforth is of great interest to our study. Since the most practical applications of the VRPPD include restrictions on the time at which each location may be visited by a vehicle, it is convenient to present a slightly more general variant of the problem, called the VRPPD with time windows (VRPPDTW). Cordeau at al. (2004) discuss that VRPPDTW is NP-hard, because it

generalizes the Traveling Salesman Problem (TSP), which is known to be NP-hard. In the presence of time windows, even finding a feasible solution to the problem is NP-hard since the feasibility problem for the TSP with time windows is itself NP-complete.

The Dial-a-Ride Problem (DARP) is a particular case of the VRPPD arising in contexts where passengers are transported, either in groups or individually, between specified origin and destination locations. The most common DARP application arises in door-to-door transportation services for elderly or handicapped people.

In their recent survey Cordeau and Laporte (2007) review the developed OR models and algorithms on the DARP. The goal of the DARP solutions is to plan a set of minimum cost vehicle routes capable of accommodating as many service requests as possible, under a set of constraints. The main emphasis is on the human satisfaction, and the reduction of passenger inconvenience should be balanced against minimizing the system operating costs.

Dial-a-ride services may operate in *static* or *dynamic* mode. In the static case, all transportation requests are known a priori, while in the dynamic case requests are accepted throughout the entire period of service (e.g. a shift) and vehicle routes are adjusted in real time to meet demand. In practice pure dynamic DARPs rarely exist since a subset of requests is often known when planning starts. Cordeau and Laporte (2007) present two formulations of the DARP – a three-index integer formulation in case of heterogeneous vehicle fleet, and a two-index formulation for the case of homogeneous fleet. The objectives in the static algorithms of multi-vehicle DARP vary in minimizing the fleet size, total route duration, total service cost, total distance traveled by vehicles

and by passengers, total service time, time window violations and/or minimizing linear combinations of some of these factors. Cordeau and Laporte (2007) discuss that the distinction between static and dynamic DARPs is often blurred in practice since the service requests are often cancelled and, as a result, transporters may allow the introduction of new requests in a solution designed for a static problem. The difficulty then is to design seed vehicle routes for these requests with sufficient slack time and capacity to accommodate future dynamic demand. The objectives in the dynamic algorithms of multi-vehicle DARP vary in maximizing the number of served passengers, minimizing the route lengths, ride times and time violations.

A special case of the DARP is the Dial-a-Flight Problem (DAFP), introduced by Cordeau at al. (2004). The operation is planned as "per-seat on-demand" service. Passengers select the destinations, time of arrival and the time window for travel. The *static* DAFP (SDAFP) is concerned with the scheduling of the single passenger requests for air transportation during a given time period (usually a single day). Each request specifies an origin airport, the earliest acceptable departure time, a destination airport, and the latest acceptable arrival time at the destination. A homogeneous fleet of airplanes operable by a single pilot is available to provide the requested air transportation. Each airplane and pilot has a home base, where they have to return at the end of each planning period. In the *dynamic* DAFP (DDAFP) passengers book seats online as they do with airline service, except there are no fixed schedules. The set of requests for air transportation arrives during the time of operation and with each request the service provider must immediately decide whether it is feasible to accept the request given the

available resources and the commitments already made. If it is feasible to accept the request, the provider will want to decide whether it is desirable to accept it, i.e. whether it will increase the profit of operation. The latter decision is especially complex as it depends on the requests that will arrive in the future. A more complex variant of DDAFP incorporates "same day travel" service, where requests can arrive during the execution of a flight schedule and have to be incorporated into the current schedule.

Cordeau at al. (2004) present an IP formulation of the SDAFP. It is a time-discretized multicommodity network flow model, which becomes large quickly and even solving medium size instances (e.g., involving 15 to 30 airports and 5 to 10 airplanes) require specialized solution approaches.

In DDAFP the operator has to decide in real time, given a set of already accepted requests, whether an incoming request can be served or not. Cordeau at al. (2004) suggest that fast heuristics will have to be part of that decision technology. In case the heuristics fail to accept a request quickly, a customer may be given the option of receiving final notification of acceptance or rejection in short time (e.g. 30 minutes) to allow time for optimization based techniques to try and accommodate the request.

Sadeh and Kott (1996) study the application of dynamic transportation planning technologies to the class of complex transportation planning problems, called Dynamic Dial-A-Ride Problem with Multiple Acceptable Destinations and/or Origins (D-DARPMADO). Their work was motivated by the military Aeromedical Regulation and Evacuation (ARE) of patients to Medical Treatment Facilities (MTFs). The problem domain is highly dynamic, complex and critical. There has been very limited experience

with this approach to handling patients other than in peace time. The first Persian Gulf war was the first significant armed conflict in which this concept has been put to a serious test. The results were far from satisfactory - about 60% of the patients ended up at the wrong destinations and half in the wrong country, Sadeh and Kott (1996).

The integrated medical regulation/evacuation problem requires the dynamic identification of appropriate MTFs for new patients and the planning/scheduling of aeromedical evacuation operations to transport these patients from their current locations to the selected MTFs. This is a large-scale, highly dynamic planning and scheduling problem that can involve hundreds or even thousands of simultaneous patient movement requests. Despite the similarities with DDAFP, D-DARPMADO is more complex and hard to control. Each patient has one or several medical requirements that constrain the type of MTF to which he or she can be evacuated and a ready-time prior to which evacuation cannot start. Additional constraints can include a maximum altitude above which the evacuation aircraft cannot take the patient, a maximum number of hours that a patient can spend in a flight before requiring an overnight rest, a maximum number of stops the patient can tolerate during evacuation, etc., (Sadeh and Kott 1996). The most challenging aspect in planning and scheduling medical evacuation operations has to do with the dynamics of a domain in which requirements and constraints continuously change over time. The authors clearly point out that the dynamic transportation problem domain is in many ways more complex than VRP/DARPs traditionally discussed in the literature. The D-DARP-MADO model expands DARP along two main directions:

- There may be multiple acceptable destination and/or origin locations for a given demand;
- Both the demands and the resources can change dynamically, while the initial schedule is being executed.

Dial (1995) introduced the concept of the Autonomous Dial-A-Ride Transit (ADART) service based on fully automated command and control, order-entry and routing and scheduling systems implemented on computers on-board vehicles. The approach outwits possible large size of DRT system with applying distributed communication between the passengers and the vehicles and negligible central management intervention. The system is fully automated, the only human intervention in the process is the customer requesting service. Furthermore, the routing and scheduling are not done at the central dispatching centre, but are distributed among vehicles through an auction mechanism.

In this section the OR problems related to DRT were introduced. The next two sections review the methods for solutions of DARP and DRT service optimization.

2.2. Heuristic approaches in DARP

Abundant research work has been done for both static and dynamic modes of DARP. Heuristics is the most widely used approach to provide fast and quality solutions for both subproblems of DARP, namely scheduling the passengers and routing of the vehicles. Scheduling subproblem concerns the assignments of passengers to the vehicles, and routing subproblem consists of search for the shortest sequence of visits the origin and destination locations of all the passengers scheduled to each vehicle.

2.2.1. Heuristics for SDARP

Based on the applied techniques the following four types of heuristics approaches for SDARP can be distinguished. For each of them one or two representative works are cited.

2.2.1.1. Insertion heuristics for SDARP

One of the first insertion heuristics for the multiple-vehicle version of the SDARP is presented by Jaw et al. (1986). In the problem formulation, customers booking in advance can specify the origin and destination locations and either a desired pick-up time or desired delivery time. The actual pick up or delivery time of a customer is allowed to deviate from the desired one, but constraints of a fixed maximum wait time window and a maximum ride time that a passenger may spend in the vehicle are imposed. The objective function of the model is the weighted sum of disutility to the customers and to the operator. The heuristic selects users in order of earliest feasible pickup time and gradually inserts them into vehicle routes so as to yield the least possible increase of the objective function. However, Wong and Bell (2006) note that the sequence or order of the requests to be inserted into the schedules has a significant impact on the performance of insertion heuristics.

A commonly used technique to reduce the computation time in the insertion heuristics for the SDARP is the clustering of the users to be served by the same vehicle prior to the routing. Such clustering leads to two-phase approaches. In the first phase, groups (clusters) of customers to be served within the same area at approximately the

same time are formed, and the algorithms search for optimal combination of the clusters to form feasible vehicle routes. In the second phase, each vehicle route is reoptimized with a single vehicle algorithm.

2.2.1.2. Parallel insertion heuristics for SDARP.

Another way to speed up the computation time is through the use of parallel computing. Toth and Vigo (1996) developed a parallel insertion procedure on the assigning of the requests to routes. Then the method performs intra-route and inter-route exchanges of passengers in search for better solutions.

Diana and Dessouky (2004) adopted the operating scenario of Jaw et al. (1986) and presented a new parallel insertion heuristic for SDARP with time windows. They developed a route initialization procedure by inserting an initial request to each of the vehicles, taking the spatial and temporal effects into account. A parallel regret insertion heuristic is then used for the rest of the requests not inserted into the initialization. Instead of ranking the requests by certain criteria (e.g., earliest pick-up time or latest delivery time) as in classic insertion heuristics, the regret insertion builds up an incremental cost matrix for each of the unassigned requests when assigned to each of the existing vehicle routes. A regret cost, which is a measure of the potential difficulty if a request is not immediately assigned, is calculated for each request, and the algorithm seeks the request with the largest regret cost, and inserts it into the existing schedules. The regret insertion algorithm requires at each step a feasibility check for the insertion of each unscheduled request in all the routes. The whole procedure is repeated until all requests are inserted. The algorithm is successfully implemented for a real case of up to 1000 service requests.

2.2.1.3. Metaheuristic approaches for SDARP

Because of their ability to find close to optimal solutions, metaheuristic algorithms have been sought to solve the SDARP. Tabu search stands out as a very powerful tool for the DARP since it is highly flexible and efficient. Flexibility stems from the capacity of handling a large number of variants within the same search framework. Efficiency is associated with solution quality. It is now clear that tabu search is capable of consistently generating high quality solutions on a large variety of routing problems. The negative side of tabu search algorithms is that their running time can be rather high. Cordeau and Laporte (2003) formulated and solved the static case applying sequential tabu search. Instead of measuring disutility by the deviation between the actual pick-up/drop-off times and the user-desired ones, their model allows users to specify a time window of a fixed width on their inbound or outbound trips, with an upper limit on the travel time for any user.

In general, the insertion heuristics are computationally fast, but may not provide as good solution as metaheuristics. On the other hand, metaheuristics may not be computationally feasible when a large number of requests need to be scheduled in a dynamic environment, and they usually require extensive computational tests to set up a number of parameters that are highly case-sensitive. Thus, in many of the approaches both methods are combined – the insertion part provides fast and rough solution, which is being improved with a metaheuristic local search. Such a combination leads to two or three phase heuristic approaches.

2.2.1.4. Two or three phase approaches for SDARP

Toth and Vigo (1997) are among the first who improved their solution method obtained after parallel insertion phase through the execution of a local search based on tabu thresholding optimization procedure. In their recent study Wong and Bell (2006) modified the parallel insertion heuristic into a three phase method. In the first phase trip characteristics are calculated and trips are ranked with a particular order for insertion. Next, a parallel insertion is performed to iteratively insert the requests into the existing routes. An optional local search procedure based on tabu search is used to further optimize the objective function.

Cordeau and Laporte (2007) conclude that excellent heuristics have already been developed for the SDARP, which allow solving instances with several hundreds of users within reasonable times and it should be possible to apply decomposition techniques for larger instances involving, two or three thousand users. Therefore, it is expected that more emphasis be put on the DDARP. This involves the construction of an initial solution for a limited set of requests known in advance and the design of features capable of determining whether a new request should be served or not and if so, how existing routes should be modified to accommodate it. In addition, it should be possible to update a partially built solution to deal with cancellations and other unforeseen events such as traffic delays and vehicle breakdowns.

A brief summary of the reviewed heuristic algorithms for SDARP is presented in Table 1.

Table 1. Summary of the discussed heuristics for SDAP.

Reference	*Objective*	*Time Windows*	*Constraints*	*Algorithm*
Jaw at al. (1986)	Minimize nonlinear combination of total disutility function	On pick up or on delivery	Vehicle capacity; Maximum ride time	Insertions
Toth and Vigo (1996)	Minimize total service cost	On pick up and on delivery	Vehicle capacity; Maximum ride time	Parallel insertion and route exchange
Diana and Desouky (2004)	Minimize weighted sum of distance, excess ride time, vehicle idle time	Lower bound on pick up time, upper bound on delivery time	Vehicle capacity; Maximum ride time; Maximum waiting time	Parallel regret insertion
Cordeau and Laporte (2003)	Minimize total route length	On pick up or on delivery	Vehicle capacity; Maximum route duration;	Tabu search
Toth and Vigo (1996)	Minimize total service cost	On pick up and on delivery	Vehicle capacity; Maximum ride time	Parallel insertion with tabu threshold search
Wong and Bell (2006)	Minimize weighted sum total operation time, passenger delay, penalty for unsatisfied demand	On pick up or on delivery	Heterogeneous fleet capacity, max wait time; max ride time	Three phase: ranking of trips; parallel insertion; local optimization.

2.2.2. Heuristics for DDARP

In the DDARP, operational constraints are the same as in the SDARP and the primary goal is to satisfy as many requests as possible with the available fleet of vehicles. As it was discussed in Section II.1, in some DRT systems if enough time is available, the

operators may apply static approaches in DDARP optimization. Requests are dealt with one at a time in a first come, first served fashion. Whenever a request can be served without violating any of the constraints, it is accepted and becomes a part of the problem. As the planning horizon goes on, the degree of flexibility decreases and the last requests to be released are likely to be rejected.

Transportation systems that provide dynamic dial-a-ride service are more flexible and can react to unpredicted events, but usually have tight real time constraints on the reoptimization algorithm. Moreover they require a monitoring system able to track the position of vehicles, their current load, and the state of the transportation network with a certain frequency. Dynamic dial-a-ride systems are more competitive than traditional transportation systems, but they need very good scheduling policy and route optimization.

Based on the applied search techniques two general types of heuristics approaches for DDARP can be distinguished – executing *global* and *local* search. For each of them we describe the most common methods and cite one or two representative works.

2.2.2.1. Heuristics performing global search

In this approach the heuristic algorithms perform search for near optimal scheduling and routing over the whole domain. The two main types covered are *constructive and iterative* heuristics and *dynamic insertion* heuristics.

2.2.2.1.1. Dynamic constructive techniques

In dynamic constructive methods the process begins with an incomplete or empty solution and constructs the missing elements of the solution. Typical examples are *rebuilding* new solutions from scratch, *insertion techniques*, *partial revision*, the *matchup scheduling* approach, the *conflict propagation* approach and *truth/reason maintenance* approach. Sadeh and Kott (1996) review two general dynamic replanning/rescheduling methods applicable for VRPTW and DARP. They discuss the possibilities for dynamic rerouting and rescheduling using *constructive* and *iterative repair* techniques. The authors envisioned two main concerns applying constructive approaches in large-scale domains with highly dynamic demand, such as the ARE domain. First, the computational requirements of such an approach could be prohibitive - by the time a new solution has been constructed, additional contingencies may have occurred, rendering the new solution obsolete. Second, in situations where it is possible to build a brand new solution each time a contingency occurs, this approach may still be undesirable because it introduces too many disruptions. The authors suggest that it is preferable to restrict solution revisions to small parts of the domain, because of two reasons - to avoid difficulties in communicating new solutions in real-time and adapting the system to new solutions.

Madsen et al. (1995) present a dynamic heuristics algorithm for passenger DARP with multiple capacities and multiple objectives as well as updating capability. The model is based on the procedure introduced by Jaw et al. (1986). Routes are pre-planned for the requests known at the beginning of the day, and the new requests can be

dynamically inserted throughout the day. Travel time updates and vehicle breakdowns can be considered. The developed insertion algorithm can be efficient enough to be implemented in a dynamic environment for online scheduling. The model was tested with 300 customers and 24 vehicle instance over a day operation, and the authors report that good quality solutions were generated in short time.

One of the challenges when optimizing dynamic transportation is to make good short term decisions without adverse long term effect. Mitrovic-Minic et al. (2004) considered the dynamic problem with a *double-horizon-based heuristic*, considering a short-term and long-term horizon. The short-term goal is to find the shortest route length, similar to the objective function of the static optimization problem. The routing decisions are taken with a constructive heuristic searching for the cheapest insertion procedure. The long-term goal is to minimize the linear combination of routes and travel time so that future requests are easily accommodated. Actually this is a mixed approach, because the solution can be improved through a longer term consideration, performed with a local tabu search heuristic. To obtain a better schedule, the *advanced dynamic waiting* strategy is applied. The available waiting time in a route is split into a few large waiting intervals which are arranged along the whole route. The route is partitioned into segments, each containing consecutive locations that are reasonably close to each other in the plane. The segments may change dynamically as new locations are inserted in a route or removed from it. The simulated test results with 100 and 500 requests show the superior performance of double horizon heuristic over the classical rolling horizon heuristics.

2.2.2.1.2. Dynamic iterative techniques

Dynamic *iterative repair* techniques traverse in the domain of complete, possible infeasible solutions, eliminate constraint violations and try to improve the quality of the solutions. Sadeh and Kott (1996) review two main iterative approaches – *interchange* approaches and *constraint-directed repair*. An interchange procedure iteratively considers possible interchanges in the neighborhood of the current solution. If a given interchange improves the quality of the solution, it is performed and a new solution is obtained. The procedure can be applied until a solution is found that can no longer be improved. In their simplest form, interchange procedures are only allowed to move from one feasible solution to another. By allowing the procedure to wander into infeasible regions of the search space, it is possible to eventually reach better solutions. If applied in their simplest form, interchange procedures usually get stuck in local optima. A number of techniques have been developed to allow the procedure to transition to neighboring solutions that are not as good as the current one in the hope of eventually reaching better solutions. Examples of such techniques include genetic algorithm procedures, simulated annealing or constraint-directed repair procedures reviewed by Sadeh and Kott (1996). Iterative improvement methods that exclude infeasible solutions can still be used to reoptimize solutions when favorable contingencies occur that make the problem easier and offer opportunities for improving the quality of the existing solution (e.g. cancellation of a request, addition of a new vehicle, duration of a trip is shorter than expected, etc.). In the face of contingencies that invalidate an existing solution (e.g. a transportation asset becoming unavailable for some period of time), iterative techniques

require heuristics to decide which part of the solution to restore, similar to constructive techniques. Thus, in large-scale systems with highly dynamic demand, both constructive and iterative techniques result in low efficiency if they search the entire domain for better solution.

2.2.2.2. Heuristics performing local search

In addition to the NP-hardness of the problem, the solution of a dynamic dial-a-ride system is time critical, because it must be performed in real time and repeated every time when significant variations of data occur. Therefore, some researchers seek for approximation, not for optimization. Two representative examples of approaches based on local search are reviewed in this section - *parallel metaheuristics* and *clustering and locating*.

2.2.2.2.1. Parallel metaheuristics

To improve the computation efficiency of metaheuristics, Attanasio et al. (2004) implemented a family of parallel tabu search heuristics. Their work is an extension of the method by Cordeau and Laporte (2003) to the dynamic case. First a static solution is constructed on the basis of the requests known at the beginning of the planning horizon. When a new request arrives, the algorithm performs a feasibility check for solution that can include the new service request. If the new request can be accepted, the algorithm performs a post-optimization, i.e., it tries to improve the current solution. The computational experiments indicate that parallel computing can be beneficial in solving real-time vehicle routing problems. Moreover, the penalty mechanism of the objective

function turns out to provide the best results while the choice of the initial static solution seems to be irrelevant.

2.2.2.2.2. Clustering and locating

Colorni and Righini (2001) develop a two-phase model, based on clustering and local search rather than a constructive mechanism. The algorithm computes the ordered sequence of pick-up and destination points, and leaves the drivers to follow their own routes through the area. Local search algorithm is performed to find a better sequence of points in its neighborhood. The neighborhood of a solution is the set of all solutions that can be obtained from the current one by removing a customer, which is scheduled but not picked and insert them into another vehicle's sequence. The authors do not provide results from the simulation experiments, instead discuss that the level of service of the system is dependent on the following parameters: number of overlapping time windows of the requests, tightness of time windows, computational time, planning horizon, and number of vehicles with their capacities.

The quality of solutions produced by modern heuristics is strongly related to running time. Thus, if sufficient time is given, the algorithms attain near optimal or even optimal solutions, as borne out by empirical studies, Diana and Dessouky (2004). However, the time available for decision making in a real time service in highly dynamic environment is often short and a different approach is needed in such contexts.

A brief summary of the reviewed heuristic algorithms for DDARP is presented in Table 2.

Table 2. Summary of the discussed heuristics for DDAP

Reference	*Objective*	*Time Windows*	*Constraints*	*Algorithm*
Madsen at al. (1995)	Multi criteria	On pick up or on delivery	Vehicle capacity; Maximum route duration; Maximum deviation of ride time	Insertion heuristic performing global search
Mitrovic-Minic at al. (2004)	Minimize total route length	Time window from start to end service of request	All request to be served; pairing and preceding constraints	Double-horizon insertion
Attanasio et al. (2004)	Minimize time windows constraints, route duration and riding times	On pick up and on delivery	Upper bound of the ride times	Three phase insertion with tabu search for optimality
Colorni and Righini (2001)	Maximize number of served customers; Minimize total traveled distance	Time window from start to end service of request	Vehicle capacity; preceding constraints	Iterative clustering algorithm based on local search

After the OR transportation problems and the heuristic approaches of DARP were introduced, in the next two sections some of the applied approaches in DRT service are presented.

2.3. Simulation approaches in DRT

In this section we briefly review some practical applications of the heuristic methods discussed in the previous sections into DRT real time operations.

As it was discussed in Chapter One, in DRT operation passengers and service provider usually have opposite interests – passengers need quick and reliable service,

while the provider would like to have more passengers served by the same vehicle, driving in the shortest possible route between the pickup and drop-off locations. To cope with these conflicting requirements in real time some researchers developed dynamic multi-objective heuristic methods. Dessouky and Adam (1996) propose a *real time scheduling algorithm* for DRT service that considers vehicle location, vehicle capacity and passenger demand. The algorithm tries to optimize three conflicting objectives – minimum total travel distance of vehicles, minimum total travel time of passengers and minimum total lateness of passenger pickup or drop-off. The limiting assumptions are that the number of vehicles is given in any shift and the vehicles operate under a fixed schedule. At first step the algorithm determines the schedule based on the calculated total cost of service and at second step the solution is improved either within the schedule of the same vehicle, or with reassigning the passengers to different vehicles. The performance of the heuristic is simulated with data generated from real para-transit service. A service request is considered for scheduling 10 min before the desired pick up time, and a is considered to be *on-time* if it arrived no later than 15 minutes of the schedule for the advance reservation requests and 1 hour for the immediate requests. The authors conclude that when the DRT system's workload is low, it will operate similarly to a taxi service (depending on the selection of the penalties in the objectives). As soon as the workload increases over a given limit, ridesharing is the preferred alternative of the heuristic.

Horn (2002) introduces a software scheduling and dispatching system called *L2sched* for passenger DRT. Demand is realized as a stream of service requests, which

are scheduled as they arrive. Each service request applies to a group of one or more passengers and includes the locations and time windows for pick up and drop offs. Travel requirements are temporally elaborated to allow a long-sighted view of fleet management and exploit system optimization. Scheduling objectives are designed to obtain efficient fleet utilization while satisfying the service requirements of each request. Thus, the software applies the centralized approach in routing and scheduling. Each vehicle provides real time information about arrivals, departures, trip cancellations and breakdowns. The software provides dynamic scheduling and routing as an extension of the current system plan. Typically the difference between the current and the next plan is induced with a small change in scheduling and/or routing, e.g. assignment of additional request and inclusion a new trip. Thus, the optimal system plan does not change radically, but evolves over time. This evolution is implemented in a three-tier optimization strategy: least-cost insertions of new requests; search for local improvements in the neighborhood of the passenger; periodic reoptimization of the planned routes. A so-called "rank-homing" heuristic is also proposed for governing the relocation of idle vehicles. A set of locations, known as "cab-ranks", are specified in advance and the heuristic chooses the cab-rank where the idle vehicle should be dispatched. To make a decision, the heuristic exploits information about future patterns of demand at each cab-rank. The performance of the software is tested in simulated environments. Two major conditions with two levels are considered – single and shared riding; immediate service or reservations in advance. Initial experiments show that in single-ride mode the system accommodates approximately 95% of the demand with an upper limit of 15 min on waiting time. In a

case of shared riding and advanced reservations the number of possible implementations is significantly greater. The CPU execution time varies from 2:12 to 6:06 min in single hiring and immediate service, 2:31 – 26:01 min in single riding and advanced reservation, 2:24 – 6:18 in shared riding and immediate service, and 2:31 – 46:40 min in shared riding and advanced reservation. The test results show that the proposed software produces fast and quality solutions in both single riding cases, but in shared riding and in case of high rate of contingencies, the centralized optimization does not perform well.

To reduce the limitations of the centralized approach, Uchimura, Takahashi and Saitoh (2002) introduce a *hierarchical model* of three level transit operation system, called *local initiative for neighborhood circulation* (LINC). The first two levels provide regular transportation between the cities in the metropolitan area and between the communities within the cities, respectively. The third level provides a dial-a-ride service on passengers in a given area within the communities and the neighborhoods using small vans. Thus, the third level is a feeder service to both Level 1 and 2. To achieve better reliability drivers are given freedom to follow any route between the stations in Level 2. The system has the following operational characteristics: 10-15 min reservation; coverage area 1.5 – 2 sq mi with approximately 10,000 people; unlimited origins and destinations within the area; ADA accessible vehicles with maximum capacity of 20 passengers. To meet these service characteristics, the LINC system should select in real time the routes with the shortest overall trip time and minimum on-board time for most of the passengers. To track the origins and destinations of the requests in real time and to inform the passengers about the time of pickups, GIS with GPS will be used. Since the

combinatorial optimization would determine the economical route and the optimum scheduling in very long time, the authors have developed a heuristic based on genetic algorithms (GA) to obtain near optimal solution in real time. The heuristic follows a search procedure based on Dijkstra's algorithm to determine the minimum cost of vehicle's routes. The heuristics is tested with simulated instances of 10 passengers, which are solved in short processing time (approximately 40 s). However, the model does not incorporate any constraints such as traffic congestions, unmet service demand and multiple vehicle service.

It was observed that in many DRT systems in order to circumvent the undesirable feature of taxicab systems and to avoid traffic congestions, drivers are allowed to deviate from their direct routes between the destination points. This strategy increases the average riding times, but also increases the flexibility to serve other passengers, increases the average occupancy and productivity of the vehicles, and hence decreases average waiting times. Since DRT is a service operation, it is expected that the main stress is on customer's needs. Therefore, a reasonable objective can be of maximizing the sum of passenger and operator surplus. Such an objective function recognizes the separate roles of customers and providers and the trade-off of increasing operational costs and increasing service quality. Gillen and Raffaillac (2002) present an algorithm to measure the contribution of *automatic vehicle location* (AVL) to both passenger satisfaction and system efficiency. The model accurately predicts the average waiting and total time in the system and the average total distance traveled. A similar problem is faced by the recently developed "webvan" food delivery service, which takes orders for groceries over the

internet and commits to delivery to the order's address within a given time frame, and *telemarket logistics*, which is discussed with the next study. Both systems are of single origin with multiple destinations.

Sheu (2006) presents a dynamic customer *group-based resource allocation methodology* for the use in demand-responsive city logistics distribution operations. The motivating example comes from the resource allocation problem resulting from tele-shopping service to manage the corresponding inventories and to provide quick-responsive door-to-door logistics services to the corresponding end-customers. Thus, dynamic allocation of logistics resources defines the feasibility of an efficient demand-responsive city logistics distribution system by enhancing the resource utility as well as by shortening the pre-route work process time in quick response to changes in customer demands. In his review Sheu (2006) notes that some multi-resource allocation problems are formulated with globally optimized procedures under strong assumptions in the problem definition, demand and/or supply side, and thus lead to too simplified models. In addition, global optimization programming approaches may have difficulties in searching optimal solutions in large-scale distribution networks and high customer demand. Furthermore, these globally optimized models may not have the capabilities of updating and grouping customer orders dynamically in quick response of customer orders. For all these reasons the author formulates the *dynamic logistics resource allocation model with sequential mechanism*. The proposed methodology is composed of five sequential operational phases: order processing, customer grouping, customer group ranking, container assignment, and vehicle assignment. The whole procedure is executed each

time when the database of customer entries is input to trigger a new logistics distribution mission. The methodology is tested in a simulated environment of 136 orders served in one day by 14 vehicles with different capacities. Two generalizations can be made from the obtained results: first, the algorithm assigns the large-sized and medium-sized vehicles to grouped customer orders and small-sized vehicles for short-distance and miscellaneous goods delivery. Second, different customer groups can be consolidated, and then served by the same vehicle avoiding extra loading and dispatching. Sheu (2006) discusses that appropriate customer order grouping and resource assignment prior to vehicle dispatching do improve the performance of city logistics systems in reducing the operational costs and average lead time. The implementation of a novel route guidance technology with the proposed dynamic resource allocation method reduces the expected delivery time associated with each customer group, which is critical in stimulating the customer satisfaction with the improved average lead time. There is still a great potential for integrating more elaborate vehicle routing algorithms for quick-responsive logistics distribution operations. Such an integrated customer group-based logistics distribution operation appears even more important to provide efficient goods delivery service in a large-scale logistics network under time-varying traffic network conditions.

In the last two sections some of the simulation approaches in DRT operations were introduced. All of them adopted centralized approaches, where the control and decision-making is done through the objective(s) that maximize the global utility of the whole system (i.e. benefit for the service provider and convenience for the clients). These approaches are usually implemented as heuristic procedures that extend basic graph

search algorithms, acting over large data collections that describe the entities of the domain problem (service requests, vehicles and schedules). A key aspect when applying these approaches is the identification of a good estimation of the client's utility function, in order to allow the generation of adequate solutions from the client's point of view. However, this is not always feasible because not all the clients share the same desires, nor appreciate them with the same importance.

From the review of heuristic and simulation approaches the following general deficiencies of centralized DRT planning methods are observed:

- Computational complexity, i.e. the models suffer to adjust the schedules and routes in real time;
- Difficulties in planning of large scale and highly dynamic problems;
- Inability to respond in case of missing information about a service request or current status of a vehicle;
- Low utilization of the vehicle fleet due to special requirements such as handicapped people transportation;
- Possible high cost of operation, in some instants close to taxi service.

To address some of these deficiencies, some researchers perform metaheuristic local search instead of global one Cordeau and Laporte (2007), or search for approximation rather than optimization of the solutions, Attanasio et al. (2004).

2.4 Intelligent Transportation Systems (ITS) approaches in DRT

The advance in the information and communication technologies, such as Internet, Geographic Information Systems (GIS), Global Positioning Systems (GPS),

Artificial Intelligence (AI) and the availability of low-cost mobile communication devices have led to a significant changes in DRT operational planning. The real time reservations become easier to manage and simultaneously the systems can operate in more complex and highly dynamic decision making environment. The increase in automation has caused the shift to online reservation system, hence, requiring service providers to have real time scheduling and dynamic dispatching capabilities. In a dynamic dispatching mode, the schedules and routes of vehicles are modified in real-time to account for any trip cancellations or any new orders. To be effective, real-time scheduling and dynamic dispatching systems require immediate information and data on the location and status of each vehicle. By taking into account real time information concerning passenger demand, vehicle location, and road conditions, real time scheduling can give the best assignment of vehicles to riders and route selection. Hence, real time scheduling and routing have the potential to improve service efficiency, to reduce the cost of transit providers and to improve customer satisfaction.

ITS offer a number of newly developed approaches for DRT operational planning and control. To increase the service through increased system efficiency, two types of advanced technological responses have been implemented: AVL and *dynamic scheduling*, Kihl at al. (1996). AVL can track and report in real time the location of all vehicles in the fleet as frequently as every other second. With the aid of a real-time display map generated by an AVL system, trips can be inserted by the dispatcher and directly posted to the closest vehicle. The most utilized method of AVL is GPS. The main disadvantage of AVL is the high cost. Dynamic scheduling is time-specific, rather than

location-specific like AVL. Unlike AVL, it does not report the actual location of the vehicle, but rather it approximates the vehicle's location based on estimated travel time between points.

Based on the decision making process concerning service requests, the ITS approaches applied in DRT operation can be split in two main groups – centralized and decentralized, which are reviewed in the next two sections.

2.4.1. ITS approaches in centralized DRT systems

In DRT systems that adopt centralized approaches, the control and decision-making is done through the objective(s) that maximize the global utility of the whole system (i.e. benefit for the operator and convenience for the clients).

To adapt DRT operations in advance or to meet the current demand in real time, Finn and Breen (1996) introduce the *telematics* approach. Telematics can be broadly defined as the integration of telecommunications and informatics systems. It consist of a communication platform (either by wire or by air) and ITS. Telematics DRT systems are based on the integration of information and telecommunication (ITC) technologies – vehicle location systems, dispatch centers, communications, booking, and reservation systems. In addition, optimization systems are included to determine the routing, vehicle size, assigned passenger based on cost, passenger requirements, and fleet ability. The most utilized telematics technologies include the following components:

- Communications between the vehicles and dispatch centers (or depots) across the area of coverage.

- Vehicle location systems for effective system management and passenger information systems. The most practical form is GPS.
- Network Management and Control Systems - dispatch centers which have substantial data collection and processing capabilities, combined with the decision and communication mechanisms to implement needed interventions.
- Booking and reservation systems - by combining integrated databases of services with real time knowledge of network state, it is possible to operate a more dynamic booking service, and to use the network control communication system to advise the vehicle driver of seat availability.
- Ticket and fare collection systems can be linked to the booking and reservation systems to automatically generate travel documents. Currently, the greatest potential for the fare collection is smart cards.
- Passenger information services - allow potential users to determine the available service offer. All data is normally held in a centralized database with links to the systems of the individual operators. The construction of the database is to be designed to allow rapid retrieval of information.

The presented trial DRT system by Wipke (1996) utilizes most of the above discussed components – GPS to locate the vehicles, two-way communications between the vehicles and a central computer-server, and advanced dispatching and routing software to control the movement of vehicles within the fleet. To provide passenger information service, the developed advanced web site allows visitors to see all the updates of vehicle position on a map every 20 seconds. The project demonstrates how a

fixed-route, fixed schedule shuttle service can be converted to be demand-responsive with increased efficiency. The proposed concept is based on three essential telematic elements:

- Precise location of the vehicles through GPS and two-way electronic communicator;
- Advanced mapping software to take current vehicle locations and directions of travel, and the incoming passenger requests for rides;
- Optimization routines in real time to determine which vehicle should make the pickup and the optimal route to take.

Thus, DRT service overcomes many of the disadvantages of public transport by using state-of-the-art ITC technologies, GPS and system optimization to arrange pick-ups and drop-offs from the desired locations.

Casey at al. (2000) report on an Advanced Public Transportation System (APTS) project. The purpose of the project is to apply ITS technologies that will improve the intermodal transportation services in a rural area with seasonal variability of demand. While the paratransit/dial-a-ride system serves residents only, because of the summer tourist pattern of the area, the fixed-route services experience significant seasonal changes in demand. The system utilizes GPS to provide real-time information on vehicle locations and/or expected arrival times available to customers in the three ways - by phone calls, via the internet and at video monitors positioned at transit or public centers. Mobile data terminals are used to send messages between dispatchers and drivers, and to store data collected on board the vehicles. A GIS-based decision-support system

integrated with an Internet-based travel planner performs the scheduling of the passengers. This tool assists the client agencies and individual customers in planning their trips by displaying vehicle routes and schedules that can serve a desired trip origin/destination and time. In addition to making real-time information available, the APTS is able to increase the number of handled customer calls (including information requests) as a result of reducing the time required for other tasks. Without APTS callers sometimes give up service because of the long waiting time to communicate to the system dispatcher.

2.4.2. ITS approaches in decentralized DRT systems

In DRT systems applying decentralized decision making approach, vehicle fleet is represented as a community of agents that perform low-level planning, scheduling, execution, and control tasks**.** As opposite to centralized evaluations, optimization can be done with less information and, as consequence, the planning solutions could be far from the optimal for the whole system. This might be the main reason why very few researchers apply decentralized approach in their studies of DRT operations.

Cubillos at al. (2004) present a mixed multi-agent system (MAS) approach to perform distributed operational planning of DRT service. The method combines the best features of both centralized and decentralized decision making approaches. The model is structured as a two-layer architecture: the *Internet layer*, which provides the interface with the vehicles, clients and other systems, and the *Planning layer*, which encapsulates the assignment and scheduling services. The model involves a negotiation process to

solve the tradeoffs between the passengers and the service provider, incorporating the client only in the final decision making. The underlying MAS framework allows the implementation of different scheduling policies, and evaluates the insertions of the trips. The adopted policy finds all the feasible ways in which a new customer can be inserted into the actual vehicle's schedule, choosing the one that offers the maximum additional utility according to an objective function. The advantage of this approach is in avoiding the estimation of the utility function, because the client is involved only in the final decision process. This is the most utilized approach in the online search engines of transportation service.

In his decentralized ADART technology, Dial (1995) introduces a fully automated dispatching (FAD) system, which can field a customer requests, schedule and optimally route a vehicle without human intervention. Every vehicle is autonomous and when vehicle's on-board computer receives a customer request, it inserts this request into the vehicle's schedule and plans the optimal route to accomplish the schedule. Furthermore, the computer may pass the request off to another vehicle. Each vehicle's computer collectively assigns the new trip to a "cluster" belonging to the responsible vehicle, thus leaving each computer to solve only a small optimization problem. All vehicles' computers work on their particular routing and scheduling problems in parallel. Thus, the huge system problem is decomposed into several easier small problems, and all of them are solved simultaneously. This enables an ADART operation to keep up with even largest demand surges. In addition, each vehicle computer can operate in virtual

ignorance of the states of the other vehicles, while at the same time cooperating with the other computers towards minimizing the total cost of service.

After reviewing the simulation and ITS approaches in real time DRT control, we can note the following disadvantages of the centralized systems:

- The developed heuristics are not invariant to the sequences of the service requests to be inserted into the vehicle schedules;
- With approaching the end of the planning horizon, the degree of freedom of flexibility of inserting the last requests decreases;
- In a highly dynamic environment by the time a new solution is constructed, additional contingencies occur, causing too frequent disruptions of the determined assignments and schedules;
- In large scale systems with highly dynamic demand the developed heuristics work with low efficiency if search over the entire domain for better solution;
- The proposed simulation products do not produce quality real time solutions in case of high rate of contingencies and multi shared vehicles.

To reduce the low efficiency of the centralized systems in areas with heavy traffic contingencies, some of the DRT operators give their drivers freedom to select the actual routes between the pickup and drop off locations, Colorni and Righini (2001), or between the stationery bus stops Uchimura, Takahashi and Saitoh (2002). Thus, the actual routings of the vehicles are determined individually, not by a central processor. This partial decentralization of the routings saves computational time and reduces the information exchange between the vehicles and the operating center.

Chapter Three

Research Motivation, Problem Domain, Research Goal and Objectives

3.1. Research Motivation

DRT operational planning where transport requests are accepted and scheduled for service, and vehicles are routed/rerouted in real time has changed significantly due to the recent advances in Intelligent Transportation Systems. However, the high level of dynamics associated with real time communication between the system operator and passengers, and system operator and vehicles require fast processing of a number of parameters. Some of these parameters consider the passenger requests; others characterize the vehicle routings and the environmental conditions. Some of these groups of data might be unrelated to each other. In addition, some system related information may or may not be available continuously based on the reliability of the technological infrastructure. Thus, the intelligent and effective processing of the available information in such a complex decision making environment requires the use of formal modeling, analysis and control approaches which are robust, modular, and/or decentralized. Robustness will provide that the system behaves in the desired manner in the unpredictable and quickly changing environment. Modularity will provide independent modeling of the service requests' assignments to the vehicles, vehicle routings and reroutings and environmental conditions. In case of a conflict or other unpredicted

situation, only the modules that cover the particular request will be affected. The decentralization will reduce the computational efforts, improve the tractability of the solution and allow parallel computations.

3.2. Research Problem Domain

In this research, we aim to provide real time control of DRT operations in a complex transportation problem referred to as Dynamic Dial-A-Ride Problem with Multiple Acceptable Destinations and Origins (D-DARP-MADO).

A highly dynamical and critical application domain of D-DARP-MADO is the military Aeromedical Regulation and Evacuation (ARE) of patients to Medical Treatment Facilities (MTFs). In this problem, the origin of the service requests can be any location within the affected region, and the destination of the demand can be assumed to be one or more locations known a priori (such as MTFs). Routing and scheduling operations in such a domain require the dynamic coordination and (re)allocation of a large number of resources subject to a wide variety of constraints. Key assets/resources and associated constraints include vehicles (airplanes or helicopters) and their characteristics (e.g. capacity, length of travel, fueling requirements, etc.), pilot and medical crews and restrictions on the number of hours they can work in any given day, airports and their different characteristics (e.g. capacity, types of aircraft they can accommodate, etc.), number of hospital beds at MTFs and the types of patients each MTF can accommodate, etc. For example, in case of a natural or man - made disaster in Tampa bay area the community authorities may appoint several (let’s consider four) hospitals to serve as

temporary MTFs – Tampa General Hospital (TGH), St Joseph Hospital (SJH), Town & Country Hospital (TCH), and University Community Hospital (UCH), Fig3.1. Helicopters, light jets or heavy duty land transporters can be used to transport patients (passengers) to the MTFs, which provide shelter and first aid.

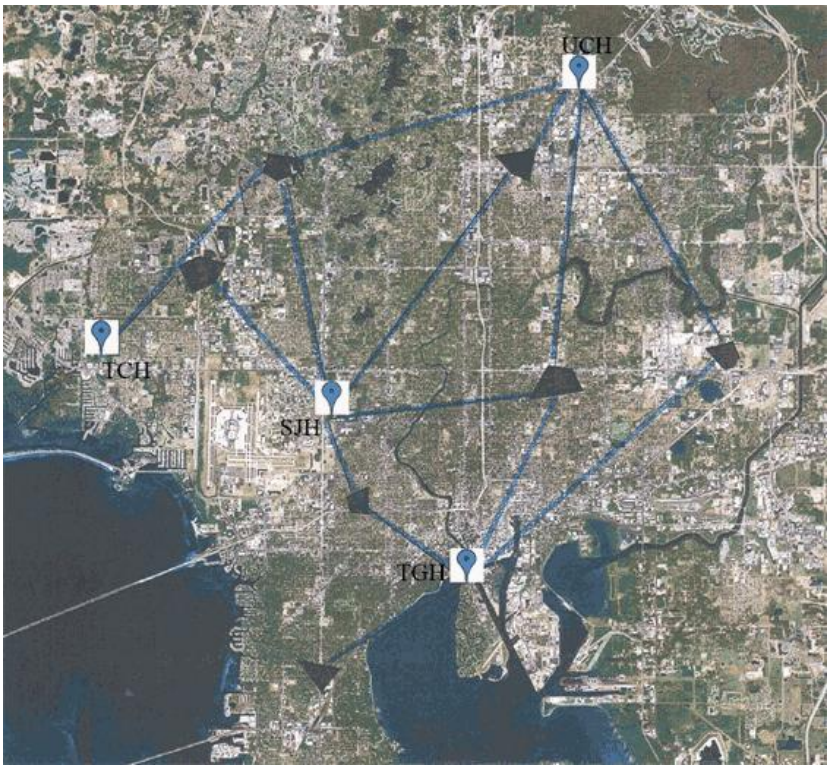


Fig. 3.1 A map of MTFs and patient pick up locations in Tampa bay area.

If patients can be accommodated at more than one possible MTF, the problem is with multiple acceptable destinations. In case the patients can get to different designated areas to be picked (the dark spots on Fig. 3.1), we talk about multiple acceptable origins. A special case of D-DARP-MADO is when patients can be picked from any possible location.

The most challenging aspect in planning and scheduling of medical evacuation operations is the high dynamics of the domain in which requirements and constraints continuously change over time. As it was discussed in Section II.1, ARE imposes two general extensions in DRT operations:

- Multiple acceptable destination and/or origin locations for a given demand; the solution to this problem must include assignments of each demand to a destination and/or origin locations;
- Both the demands and the resources can change dynamically while the initial route and schedule are being executed. The proposed solution method must be capable of real time revision of the assignments of patients to resources and routes and schedules of vehicles.

3.3. Research Goal and Objectives

In this study we propose the representation of DRT systems as a Discrete Event Systems (DESs) where the model captures both the low level dynamics (such as infrastructure conditions, current status of vehicles and limitations) and high level dynamics (such as service demand requests) of system evolution in a modular manner. The mathematical foundation of DES theory facilitates logical analysis of these complex systems and provides the necessary framework for the development of real time scheduling and intelligent decision making tools.

The real time control of DRT is developed as SC of DES, which synthesizes the supervisor(s) – i.e. the acceptable behaviors of all the elements of the system. Fig. 3.2 outlines the framework of the online DRT control structure.

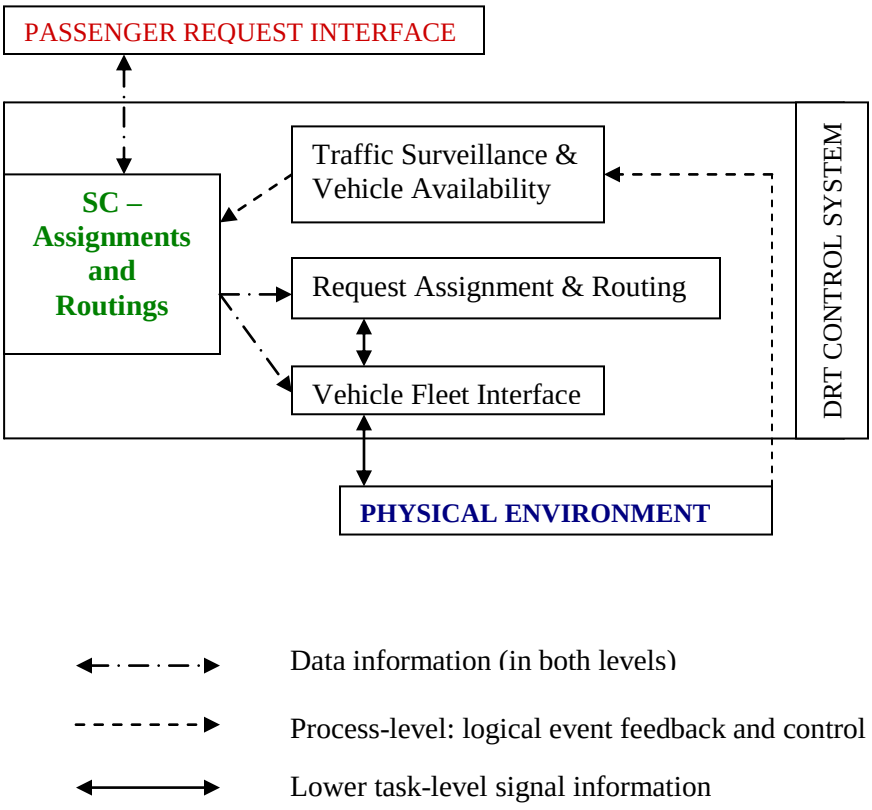


Fig. 3.2 Framework for real time DRT control.

In Fig.3.2, DRT control system takes input data from the passenger request interface and the physical environment (vehicle fleet and service area with its conditions). When a new service request is received the assignment controller checks if it is feasible to accept this passenger. If the request cannot be accepted because of operational limits, the system sends a signal of rejected request. If the request is feasible, the routing

supervisor generates the possible routings of the vehicles to serve the request. In case of more than one possible assignments and/or routings, the system may use an optimizer or rule based logic (e.g. Route Planner and Task-assignment) to select the preferred vehicle. In any case, the information to the selected vehicle is sent through the Vehicle Fleet Interface, and the passenger is informed for the service. During operation the system (e.g. Traffic Surveillance and Vehicle Availability) receives feedback information for the current conditions of the physical environment (vehicles breakdowns, traffic congestions, etc.).

To the best of our knowledge, Seow, Pasquier and Hong (1999) are the first researchers who proposed the application of Supervisory Control Theory (SCT) to the modeling and real time operational control of the class of land DRT systems. The main advantages of SCT for online service control of DRT systems over the heuristic and simulation methods for operational planning are:

- Possibilities to consider service of a new request without affecting the already scheduled requests;
- Possible modularity and decentralization of the supervised control, which allows autonomous service operational control of the vehicles and parallel computation of their supervisors;
- Dealing with unobserved events that may occur in complex systems.

In this research we provide several supervisory controller synthesis methodologies applicable in real time control of large scale DRT systems operating in ARE environment. Within this goal are the following objectives:

- Model the uncontrolled system behavior and specifications of ARE problem using Finite Automata (FA);
- Synthesize centralized supervisory controller to demonstrate the decision making of accepting or rejecting service requests;
- Synthesize general supervisor from the independent modular supervisors of the different specifications;
- Apply decentralized supervisory control to compute in parallel the local supervisors of concurrent groups of vehicles and passengers; synthesize the global supervisor of the entire system.

To accomplish these objectives, we apply and extend the DES modeling framework in the study of Seow and Pasquier (2004) of DRT supervisory control of the land transportation model in the following four main directions:

- Extend modular SC with additional specifications which are characteristics of ARE problem domain: maximum length of the routes (e.g. flights); finite set of origin-destinations of the requests.
- In modular SC the action of the central supervisor *S* is represented as a combination of the control actions of two or more supervisors. The advantage of this method is in the simplified procedure to check the feasibility of any

service request. If a given request cannot be accepted by one of the supervisors, there is no need to check for the rest of the supervisors.

- Develop decentralized SC: A decentralized SC consists of "processing nodes" that jointly control a distributed system, Cassandras and Lafortune (1999). In a decentralized DRT system each vehicle and its assigned passengers form a subsystem. Thus, vehicles' routings and assignments of each subsystem do not interfere with the routings and assignments from any other subsystem. Hence, local supervisors of each subsystem may not observe and do not control the behavior of the rest of the subsystems.
- Since the formed subsystems operate simultaneously they form concurrent DESs, which are independent to each other. Thus, all the local supervisors can be synthesized in parallel.

In the centralized planning approach (see Section II.4), the scheduling and routing of the entire system is updated with any new request or change in the domain. Despite efficient heuristics and communication technologies, the permanent update of all the passenger assignments and vehicle routings take computational time, which cannot be neglected in real time planning of a complex problem like the emergency aeromedical evacuation. In addition, the heuristics need all the relevant information of the requests to compute the passenger assignments and calculate the vehicle routings.

The results of this research are expected to overcome the disadvantages centralized control and achieve a methodology for synthesis of robust, modular and decentralized real time control of concurrent systems.

Chapter Four

Discrete Event Systems and Supervisory Control

In this chapter we introduce the basic concepts of DES, supervisory control theory (SCT) and their representation with finite automata (FA).

4.1. Discrete Event Systems

DESs are dynamic systems driven by event occurrences usually at irregular intervals. These events take the systems from one state to another. Such systems arise in a variety of contexts such as information and communication networks, complex and multimode production processes and robotics, logistics and vehicular traffic. These applications require control and coordination to ensure the orderly flow of events. As controlled (or controllable) dynamic systems, DESs qualify for a proper subject for control theory (CT). CT for DES considered in this study is based on FA concepts. The essential concepts and modeling of DES can be found in Cassandras and Lafortune (1999) and the fundamentals of the FA theory and supervisory control theory (SCT) in Wonham (2006). In the following review of DES modeling and SCT background till Section 4.2.3 we adopt the formalism of Cassandras and Lafortune (1999), and Sections 4.2.4 and 4.2.5 are based on the study of Yoo and Lafortune (2002).

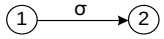
4.1.1. FA modeling of DES

An *automaton* is a device that is capable of representing a sequence of events according to well defined rules. Automata are used as a modeling formalism since they are easy to use, intuitive, amenable to all the unary and composition operations, and easy to analyze.

A DES can be modeled as a five-tuple automaton *A*, i.e. $A=(Q,\Sigma,\delta,q_0,Q_m)$, where *Q* is a set of states, Σ is a non-empty set of events (alphabet), $\delta : Q x \Sigma \to Q$ is a transition function, $q_0 \in Q$ is the initial state and $Q_m \in Q$ is the set of marked states (i.e. states indicating the completion of the tasks or sequences of tasks from a control perspective). A transition in the automaton *A* is any element of δ, and may be denoted simply by the triple (q,σ,q'), where $\delta(q,\sigma)=q'$.

If the transition function δ is partial, only a proper subset of Σ can occur, and a more flexible and economical representation of DES is provided by a *generator G*, i.e. $G=(Q,\Sigma,\delta,q_0,Q_m)$. If $\delta(q,\sigma)$ is defined, then we say that σ is eligible at *q* in *G* and denote it as $\delta(q,\sigma)!$. The set of all feasible events that can be executed at state *q* is denoted by $\Gamma(q)$, i.e. $\Gamma(q)=\{\sigma \in \Sigma : \delta(q,\sigma)!\}$.

Finite state automata are graphically described by directed-transition graphs. In order to represent an automaton, a state is identified by a node (represented by a circle with the state's number inside, e.g. ①) of the graph whose edges are labeled by

transition labels (represented by an arrow, e.g. 

). The initial state is labeled with an entering arrow 

, while a marked state is labeled with an emitting arrow 

When $q_0 \in Q$ is also a marked state, it is labeled with a double arrow 

.

4.1.2. Language and language characteristics

A *language L* defined over an event set Σ is a set of finite-length strings formed from events in Σ. The set Σ^* contains all possible finite sequences, or strings, over Σ, plus the null string ε. The definition of δ can be extended to Σ^* as follows:

- $\delta(q,\varepsilon) = q$
- $\delta(q,s\sigma) = \delta(\delta(q,s),\sigma)$ *for* $s \in \Sigma^*$ *and* $\sigma \in \Sigma$.

System's behavior may then be described by two languages: *L(A)*, the prefix-closed language generated by automaton *A*, and $L_m(A)$, the language marked by automaton *A*. Formally, $L(A) = \{s \in \Sigma^* : \delta(q_0, s)!\}$ and

$$L_m(A) = \{s \in L(A) : \delta(q_0, s) \in Q_m\}.$$

The language generated by automaton *A* can be interpreted as the set of all the sequences of events that take the system from its initial state to some reachable state in *A*. The language marked by *A* can be interpreted as the set of all the strings that take the system from its initial state to some marked state i.e. final state or a state of satisfactory completion. By definition, $L_m(A) \subseteq L(A)$ is the subset of strings in *L(A)*, which ends in any of the final states Q_m. Thus, if an automaton *A* represents a DES, then Q_m represents

completed tasks executed with the physical process of the DES. If automaton *A* models a behavioral specification *K*, then $K = L_m(A)$ is the behavior of interest.

4.1.3. Operations on languages

The following three operations on languages are essential in language composition:

- *Concatenation*: If $L_a, L_b \in \Sigma^*$ then a string *s* is in $L_a L_b$, if it can be written as the concatenation of a string in L_a with a string in L_b is:
  $L_a L_b = \{s \in \Sigma^* : (s = s_a s_b) and (s_a \in L_a) and (s_b \in L_b)\}$.
- *Prefix-closure*: The prefix closure of L is the language denoted by $\bar{L}$, consisting of all the prefixes of all the strings in L. If $L \in \Sigma^*$, then $\bar{L} = \{s \in \Sigma^* : \exists t \in \Sigma^* (st \in L)\}$. L is said to be *prefix-closed* if any prefix of any string in L is also an element of L, i.e. $L = \bar{L}$.
- *Language-closure*: a language $L \subseteq L_m(A)$ is said to be $L_m(A)$-*closed* if $\bar{L} \cap L_m(A) = L$.

4.1.4. Unary operations on automata

The following three operations on automata are essential in FA theory:

- *Accessible states*: The set of all the states that can be reached from the initial state is called the accessible states subset. Let Q_a denotes the accessible states subset, and is described as: $Q_a = \{q \in Q : (\exists s \in \Sigma^*), \delta(q_0, s) = q\}$.

- *Co-accessible states*: The set of all the states q from which some marked state can be reached is called the co-accessible states subset. The co-accessible states subset denoted by Q_{ca}, $Q_{ca} = \{q \in Q : (\exists\, s \in \Sigma^*), \delta(q_0, s) \in Q_m\}$.
- *Trim automaton*: an automaton that is both accessible and co-accessible is said to be trimmed.

4.1.5. Composition operations on automata

The following two composition operations on automata are of great importance in SCT:

- *Product* of two automata A_1 and A_2 is the accessible automaton A,

$A = A_1 \times A_2 = (Q_1 \times Q_2, \Sigma_1 \cap \Sigma_2, \delta, (q_{01}, q_{02}), Q_{m1} \times Q_{m2})$, where

$$\delta((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) & \text{if } \delta_1(q_1, \sigma)! \text{ and } \delta_2(q_2, \sigma)! \\ \text{undefied} & \text{otherwise} \end{cases}$$

In the product, the transitions of the two automata are synchronized on a common event, i.e. $\Sigma_1 \cap \Sigma_2$. It is verified that $L(A_1 \times A_2) = L(A_1) \cap L(A_2)$ and $L_m(A_1 \times A_2) = L_m(A_1) \cap L_m(A_2)$

- *Parallel composition* of two automata A_1 and A_2 is the automaton A,

$A = A_1 \| A_2 = (Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, (q_{01}, q_{02}), Q_{m1} \times Q_{m2})$, where

$$\delta((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) & \text{if } \delta_1(q_1, \sigma)! \text{ and } \delta_2(q_2, \sigma)! \\ (\delta_1(q_1, \sigma), q_2) & \text{if only } \delta_1(q_1, \sigma)! \\ (q_1, \delta_2(q_2, \sigma)) & \text{if only } \delta_2(q_2, \sigma)! \\ \text{undefied} & \text{otherwise} \end{cases}$$

In the parallel composition a common event $\sigma \in \Sigma_1 \cap \Sigma_2$ can only be executed if both automata execute it simultaneously. The rest of the events $\in (\Sigma_2 \setminus \Sigma_1) \cup (\Sigma_1 \setminus \Sigma_2)$ can be executed whenever possible. If $\Sigma_1 \cap \Sigma_2 = \varnothing$, then there are no synchronized transitions and $A_1 \| A_2$ is the concurrent behavior of the two automata. This is also called the *shuffle* of A_1 and A_2.

4.1.6. Analysis of DES

One of the key reasons for applying finite state automata (FSA) to model DES is their flexibility and amenability to analysis for answering various questions about the behavior of the system. The computational complexity of navigating the state transition diagram of a deterministic automaton if there is no need of iterations is linear of the state space, i.e. $O(n)$, where n is the state space, $n = |Q|$. If iterations are necessary, the complexity typically is $O(n^2)$.

In the next subsections the most-often encountered analysis problems for DES are reviewed.

- *Safety properties* are concerned with the reachability of certain undesired states, i.e. the presence of certain undesirable strings or substrings in the language generated by the automaton. A DES model of a system is usually built in two steps: first automaton models of the components of the system are defined; next the complete system model is obtained by either product and/or parallel composition of the constituent automata. The safety questions are

posed on this complete automaton. The algorithms that answer all these safety questions are quite straightforward and described in Cassandras and Lafortune (1999):

- To determine if a given state q_2 is reachable from another state q_1, one has to check if q_2 is accessible from q_1 being initial state.
- To determine if a given substring s_1 is possible in the automaton, one has to try to execute s_1 from all the accessible states.
- To test the inclusion $A \subseteq B$ is equivalent to testing $A \cap B^c = 0$. The intersection is implemented by taking the product of *A* and *B*.
- *Blocking properties* are concerned with the coaccessibility of states to the set of marked states. An automaton *A* is said to be *blocking* if $\overline{L_m(A)} \subset L(A)$ and nonblocking if $\overline{L_m(A)} = L(A)$.

This implies that for every string $s \in L(A)$, there is at least one string ω such that $s\omega \in L_m(A)$. In other words, an automaton is nonblocking if every string starting from the initial state can be completed to some string that leads to a marked state. To determine if a given accessible automaton *A* is blocking, one has to check if all the states of *A* are coaccessible. If there are states that are not coaccessible, *A* is blocking, otherwise it is nonblocking. If *A* can reach a state *q*, where $\delta(q,\sigma) = 0, \forall \sigma \in \Sigma, and\ q \notin Q_m$, then *q* is said to be a *deadlock* state. Deadlock states can be found by examining the active event sets of the states. *A* can also reach an unmarked state *p*, which is *strongly connected* to a set of unmarked states *P*, i.e. these states are reachable from one another but there is no

transition going out of *P*. In such a case there is always at least one transition that can be executed but *A* can never reach any of the marked states. This situation is called a *livelock*.

- *Unobservable events* are events that occur in the system but are not seen or observed by an outside observer of the system behavior. For example, *fault events* that do not cause any immediate change in the sensor readings are unobservable events.

If the transitions caused by all the unobservable events are labeled by ε, then a nondeterministic automaton model of the system will be obtained. In order to keep the determinism, the event set Σ is partitioned into two disjoint sets: Σ_o – the set of observable events, and Σ_{uo} – the set of unobservable events.

Recall from section IV.1.1 an automaton with a partial transition function is called a *generator* (G). With the structure (G, Σ_o) the *natural projection* $P : \Sigma^* \to \Sigma_o^*$ is defined as follows:

- $P(\varepsilon) = \varepsilon$
- $P(e) = \begin{cases} e, if\ e \in \Sigma_o \\ \varepsilon, if\ e \notin \Sigma_o \end{cases}$
- $P(se) = P(s)P(e)$ $for\ s \in \Sigma^*, e \in \Sigma$.

In other words P erases only the unobservable events. If G_{obs} denotes the minimum deterministic automaton equivalent to the generator of interest G, we have that:

- $L(G_{obs}) = P[L(G)]$
- $L_m(G_{obs}) = P[L_m(G)]$

- The state of G_{obs} reached after string $s \in P[L(G)]$ will contain all the states of G that can be reached after any of the strings in $P^{-1}(s) \cap L(G)$. In words, the state of G_{obs} is the union of all the states of G consistent with the observable events occurred so far (i.e. string s).

4.2. Supervisory control

In supervisory control of a given DES the behavior of the system must be modified by feedback control to achieve a given set of *specifications*. If a generator G models a DES, then it is said that G represents the *uncontrolled behavior* of the system. The premise is that this behavior is not satisfactory and must be modified by control; modifying the behavior is restricting to a subset of $L(G)$. To alter the behavior of G we need a supervisor S. S observes some (possibly all) of the events that G generates and tells G which of the defined events are allowed. Thus, the two key considerations are that S is limited in terms of observing the events executed by G and S is also limited in disabling feasible events of G. Therefore, we consider the observable events in Σ - those that S can observe and controllable events in Σ - those that S can disable.

4.2.1. Controlled DES

Let a DES be modeled by a pair of languages L and L_m, where L is the set of all strings that can be generated by the system and $L_m \subseteq L$ is the set of marked strings that represent the completion of some tasks by the DES. Assume that both L and L_m are the languages generated by $G = (Q, \Sigma, \delta, q_0, Q_m)$.

The event set Σ is partitioned in two disjoint subsets: $\Sigma = \Sigma_c \cup \Sigma_{uc}$, where Σ_c is the set of controllable events that can be prevented from occurring by a supervisor *S* and Σ_{uc} is the set of uncontrollable events that cannot be prevented from happening.

The adjoined supervisor *S* interacts with generator *G* in a feedback manner, as depicted in Fig. 4.1.

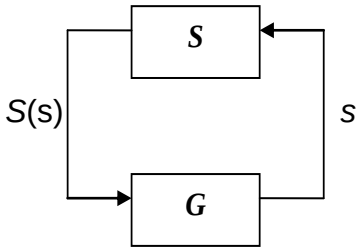


Fig. 4.1 The feedback loop of supervisory control.

Let all the events in Σ be observed by *S*. Thus, in Fig. 4.1 *s* represents all the strings of the events executed by *G* so far and observed by *S*. The control pattern means that the transition function δ can be controlled by *S* in the sense that Σ_c can be dynamically enabled or disabled, so that the modeled system exhibits a desired language. Formally, *S* is any map $S : L(G) \to 2^{\Sigma}$. Thus, for each $s \in L(G)$ generated by *G*, the set of enabled events that *G* can execute at its current state $\delta(q_o, s)$ if $S(s) \cap \delta(q_o, s)!$. *S* is said to be *admissible* if for all $s \in L(G)$, $\Sigma_{uc} \cap \delta(q_o, s)! \subseteq S(s)$, i.e., *S* is not allowed to disable a feasible uncontrollable event. Given *G* and an admissible *S*, the resulting closed-loop system is denoted by *S*/*G* (i.e. *S controlling G*). The controlled system *S*/*G* is a DES, characterized with its generated and marked languages $L(S/G)$ *and* $L_m(S/G)$. The generated language $L(S/G)$ is defined recursively as follows:

- $\varepsilon \in L(S/G)$
- $[(s \in L(S/G)) \text{ and } (s\sigma \in L(G)) \text{ and } (\sigma \in S(s))] \Leftrightarrow [s\sigma \in L(S/G)]$

Since always $\{\varepsilon\} \subseteq L(S/G) \subseteq L(G)$, $L(S/G)$ is nonempty and closed.

The marked language $L_m(S/G)$ is defined as follows: $L_m(S/G) = L(S/G) \cap L_m(G)$.

The DES *S*/*G* is said to be *blocking* if $L(S/G) \neq \overline{L_m(S/G)}$ and *nonblocking* when $L(S/G) = \overline{L_m(S/G)}$. Since $\overline{L_m(S/G)} \subseteq L(S/G)$ always holds, the nonblocking condition is also equivalent to $L(S/G) \subseteq \overline{L_m(S/G)}$.

4.2.2. Controllability theorem and realization of supervisors

The key existence result for supervisors in the presence of uncontrolled events is specified by the *Controllability Theorem* (CTh): Let a DES is modeled by the generator $G = (Q, \Sigma, \delta, q_0, Q_m)$, where $\Sigma_{uc} \subseteq \Sigma$ is the set of uncontrolled events, and $K \subseteq L(G)$, $K \neq \varnothing$. There exist a supervisor *S* such that $L(S/G) = \overline{K}$ if and only if $\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}$. This condition is called the *controllability condition*. The proof of the theorem is presented in Cassandras and Lafortune (1999).

CTh is utilized to define when a language is controllable with respect to another given language. Thus, if *K* and $M = \overline{M}$ are languages over event set Σ and $\Sigma_{uc} \subseteq \Sigma$, *K* is said to be *controllable* with respect to M and Σ_{uc} if $\overline{K}\Sigma_{uc} \cap M \subseteq \overline{K}$. Since

controllability is a property of prefix-closure, K is controllable if and only if $\overline{K}$ is controllable.

Suppose a language $K \subseteq L(G)$ is controllable with respect to G and $L(S/G) = \overline{K}$. From the proof of CT it follows that the supervisor S of the controlled system S/G is defined by $S(s) = [\Sigma_{uc} \cap \Gamma(\delta(q_0, s))] \cup \{\sigma \in \Sigma_c : s\sigma \in \overline{K}\}$, for $s \in L(G)$, and results in $L(S/G) = \overline{K}$, Cassandras and Lafortune (1999).

To build an automaton realization of S, it suffices to build an automaton that marks $\overline{K}$. Let R be such an automaton, i.e. $R = (P, \Sigma, \gamma, p_0, P_m)$, where R is trim, and $L_m(R) = L(R) = \overline{K}$. R can be connected to G by product operation and the result $R \times G$ is the desired behavior of the system S/G;

$$\begin{aligned} L(R \times G) &= L(R) \cap L(G) \\ &= \overline{K} \cap L(G) \\ &= \overline{K} = L(S/G) \end{aligned}$$

Similarly,

$$L_m(R \times G) = L(S/G) \cap L_m(G) = L_m(S/G).$$

Note that R is defined over the same event set Σ, thus $R \,||\, G = R \times G$. Hence, the control action S(s) is encoded into the transition structure of R i.e.

$$\begin{aligned} S(s) &= [\Sigma_{uc} \cap \Gamma(\delta(q_0, s))] \cup \{\sigma \in \Sigma_c : s\sigma \in \overline{K}\} \\ &= \Gamma_R(\gamma(p_0, s)) \\ &= \Gamma_{R \times G}(\gamma \times \delta((p_0, q_0), s)) \end{aligned}$$

In the latter, $\Gamma_{R \times G}$ and $\gamma \times \delta$ denote the active event set and transition function of $R \times G$, respectively.

The interpretation with the control paradigm is as follows: Let *G* is in state *q* and *R* is in state *p* following the execution of a string $s \in L(S/G)$, and *G* generates an event σ that is enabled. The same event is also present in the active event set of *R* at *p*. Thus, *R* also executes *σ*. If q' and p' are the new states of *G* and *R* after execution of *σ*, the set of enabled events of G after string $s\sigma$ is given by the active event set of *R* at p'. With this procedure *R* is called the *standard realization* of *S*.

Consider the reverse question – if there is a given automaton *C* and we form the product $C \times G$, can that be interpreted as controlling *G* by *C*? The supervisor *S* for *G* induced by *C* can be defined as;

$$S(s) = \begin{cases} [\Sigma_{uc} \cap \Gamma(\delta(q_0, s))] \cup \{\sigma \in \Sigma_c : s\sigma \in L(C)\} & \text{if } s \in L(G) \cap L(C) \\ \Sigma_{uc} & \text{otherwise} \end{cases}$$

Therefore, $L(S/G) = L(C \times G)$ if and only if $L(C)$ is controllable with respect to $L(G)$ and Σ_{uc}, i.e. $\overline{L(C)}\Sigma_{uc} \cap L(G) \subseteq \overline{L(C)}$. The resulting closed loop behavior is defined with the languages:

- $L(S/G) = L(C \times G) = L(C) \cap L(G)$
- $L_m(S/G) = L_m(C \times G) = L_m(C) \cap L_m(G)$.

If a given language *L* is not controllable, it is useful to find the "largest" sublanguage of *L* that is controllable, denoted by $L^{\uparrow C}$. Cassandras and Lafortune (1999) present two effective algorithms to calculate $L^{\uparrow C}$ in prefix-close case and in general case.

4.2.3. Modular supervisory control

In modular control, the control action of a supervisor S is given by combination of the control action of two or more supervisors. Consider the case of two supervisors S_1 and S_2 each defined for G, the modular supervisor is determined as $S_{mod12}(s) = S_1(s) \cap S_2(s)$. Thus, an event σ is enabled if and only if it is enabled by both S_1 and S_2. Fig. 4.2 depicts the architecture of a modular supervisory control with two supervisors.

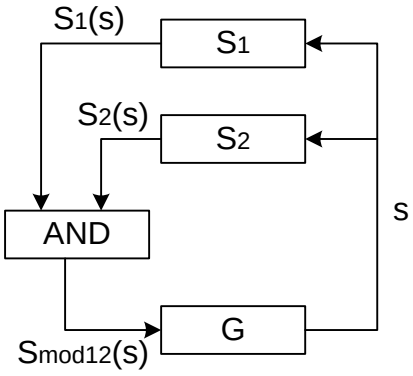


Fig. 4.2 Modular supervisory control with two supervisors.

The closed-loop behavior under modular control is formalized with the following languages:

- $L(S_{mod12} / G) = L(S_1 / G) \cap L(S_2 / G)$
- $L_m(S_{mod12} / G) = L_m(S_1 / G) \cap L_m(S_2 / G)$.

Modular supervisory control is introduced as a solution to the problem of state space increase faced by the centralized supervisory control. The idea is in presenting $S_{mod12}(s)$ as the intersection of the active event sets of R_1 and R_2, i.e.

$S_{mod12}(s) = R_1 \times R_2 \times G$. Then, if the standard realizations R_1 and R_2 of S_1 and S_2 have n_1 and n_2 states respectively, the model needs to store a total of $n_1 + n_2$ states instead of $n_1 n_2$.

The modular supervisory control problem (MSCP) with a given a DES *G* with event set Σ, uncontrollable event set $\Sigma_{uc} \subseteq \Sigma$, and admissible language $L_a = L_{a1} \cap L_{a2} \cap \ldots \cap L_{an}$, where $L_{ai} = \overline{L_{ai}} \subseteq L(G)$ *for* $i = 1, 2, ..., n$, is to find a modular supervisor S_{mod} (according to the architecture in Figure 4.2) such that $L(S_{mod} / G) = L_a^{\uparrow C}$.

To solve MSCP, first we build the standard realizations R_{i} of S_{i} such that $L(S_i / G) = L_{ai}^{\uparrow C}$. Next, take S_{mod} to be the modular supervisor, such that $S_{mod}(s) = S_{1n}(s) = S_1(s) \cap S_2(s) \cap \ldots \cap S_n(s)$. With this choice of modular supervisor S_{mod} the desired solution is $L(S_{mod1n} / G) = L_{a1}^{\uparrow C} \cap L_{a2}^{\uparrow C} \cap \ldots \cap L_{an}^{\uparrow C} = L_a^{\uparrow C}$.

Wonham and Ramadge (1988) defined two languages $L_1, L_2 \in \Sigma^*$ to be *nonconflicting* if $\overline{L_1 \cap L_2} = \overline{L_1} \cap \overline{L_2}$.

If S_i *for* $i = 1, 2, ..., n$ are the individual nonblocking supervisors for *G*, then S_{mod1n} is *nonblocking* if and only if every $L_m(S_i / C)$ is a nonconflicting language, i.e.

$\overline{L_m(S_1 / C) \cap \ldots \cap L_m(S_n / C)} = \overline{L_m(S_1 / C)} \cap \ldots \cap \overline{L_m(S_n / C)}$. This statement is proved in section IV.2.5.

4.2.4. Decentralized supervisory control

Decentralized control represents the situation where there are several local supervisors that are jointly controlling a given system that is inherently distributed. Such decentralized control architectures arise in a variety of network systems such as mobile communications, automated vehicular systems, and integrated sensor networks.

There are two main advantages of the decentralization – improved computational tractability of the control and possibility of partial observation of the event set. Consider a DES controlled by n local supervisors and the i^{th} having m_i states, $i = 1,2,\ldots,n$. A global supervisor with the same control action will require $m_1 m_2 \ldots m_n$ states. Let the complexity of designing a supervisor with m states is $f(m)$. Then the complexity of designing a global supervisor is $a = f(m_1 \ldots m_n)$, while the complexity of designing n local supervisors is $b = f(m_1) + \ldots + f(m_n)$. Lin and Wonham (1988) report that in a typical case $f(m) = C_1 m^3, n = 3, m_i = 20$ and the ratio a/b explodes to 2.13×10^7. Let the memory requirement for implementation of a supervisor is $g(m)$. Then the memory required to implement a centralized supervisor is $c = g(m_1 \ldots m_n)$, while the memory required to implement n local supervisors is $d = g(m_1) + \ldots + g(m_n)$. Typically $g(m)$ is a linear function of m, i.e. $g(m) = C_2 m$. Lin and Wonham also report that with the same values of n and m_i, the ratio $c/d \cong 1.33 \times 10^2$.

Another distinguishing feature of the decentralized from the modular control architecture is the possibility that the individual supervisors can be partial-observation supervisors and moreover their respective sets of observable and controllable events need

not be the same. To formulate the decentralized supervisory control problem consider a set of n partial-observation supervisors, each associated with a different projection P_i, $i = 1,\ldots,n$ jointly controlling the given DES G with event set Σ. Four sets of events are associated with G: Σ_c, Σ_{uc}, Σ_o, and Σ_{uo}. With each supervisor S_i we have: the set of controllable events $\Sigma_{i,c} \subseteq \Sigma_c$, where $\bigcup_{i=1}^{n} \Sigma_{i,c} = \Sigma_c$, the set of observable events $\Sigma_{i,o} \subseteq \Sigma_o$, where $\bigcup_{i=1}^{n} \Sigma_{i,o} = \Sigma_o$, and the natural projection $P_i : \Sigma^* \to \Sigma_{i,o}^*$ corresponding to $\Sigma_{i,o}$. The domain of partial-observation supervisor can be extended from $P_i[L(G)]$ to $L(G)$ and $S_i(s) = S_{Pi}\ [P_i(s)]$.

Here we briefly review the three architectures of decentralized supervision: *conjunctive*, *disjunctive* and *general* described by Yoo and Lafortune (2002).

4.2.4.1. Conjunctive decentralized architecture

Similarly to modular control, the net control action of conjunctive architecture is the intersection of the sets of the events enabled by each supervisor, i.e. $S_{conj}(s) = \bigcap_{i=1}^{n} S_i(s)$. For the conjunctive architecture, a local decision rule of S_i enables by default the set $\Sigma_c \setminus \Sigma_{c,i}$. Fig. 4.3 depicts the architecture of conjunctive decentralized supervisory control with two supervisors.

The prefix closed language generated by the conjunctive supervisor is expressed as follows:

- $\varepsilon \in L(S_{conj} / G)$;

- $\left[s \in L\left(S_{conj} / G\right)\right] \wedge \left[s\sigma \in L(G)\right] \wedge \left[\forall i, \sigma \in S_i(s)\right] \Leftrightarrow s\sigma \in L\left(S_{conj} / G\right)$.

The marked language is defined as: $L_m\left(S_{conj} / G\right) = L\left(S_{conj} / G\right) \cap L_m(G)$.

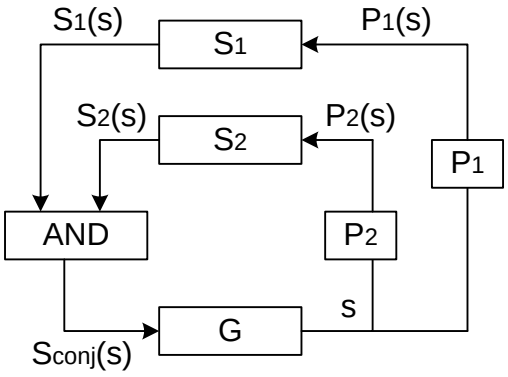


Fig. 4.3 Conjunctive decentralized supervisory control with two supervisors.

4.2.4.2. Disjunctive decentralized architecture

For the disjunctive architecture, a local decision rule of S_i disables by default the set $\Sigma_c \setminus \Sigma_{c,i}$, which is controllable by the other supervisors. The disjunctive supervisor S_{disj} is defined as follows: $S_{disj}(s) = \bigcup_{i=1}^{n} S_i(s)$. Fig. 4.4 depicts the architecture of disjunctive decentralized supervisory control with two supervisors.

The prefix closed language generated by the disjunctive supervisor is expressed as follows:

- $\varepsilon \in L\left(S_{disj} / G\right)$;
- $\left[s \in L\left(S_{disj} / G\right)\right] \wedge \left[s\sigma \in L(G)\right] \wedge \left[\forall i, \sigma \in S_i(s)\right] \Leftrightarrow s\sigma \in L\left(S_{disj} / G\right)$.

Analogously, the marked language of the disjunctive supervisor is $L_m\left(S_{disj} / G\right) = L\left(S_{disj} / G\right) \cap L_m(G)$.

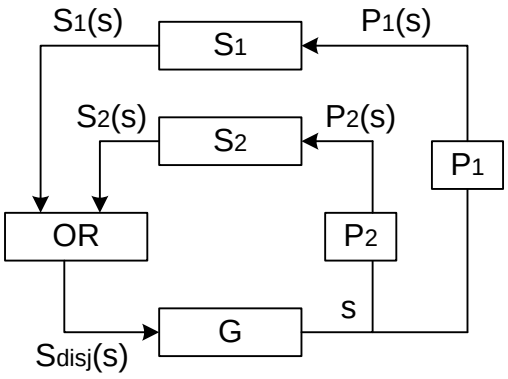


Fig. 4.4 Disjunctive decentralized supervisory control with two supervisors.

4.2.4.3. General decentralized architecture

In the general architecture the set of controllable events Σ_c is partitioned into two subsets $\Sigma_{c,e}$ and $\Sigma_{c,d}$: $\Sigma_c = \Sigma_{c,e} \cup \Sigma_{c,d}$. Here $\Sigma_{c,e}$ is the set of controllable events for which the default setting is enablement, while $\Sigma_{c,d}$ is the set of controllable events for which the default setting is disablement.

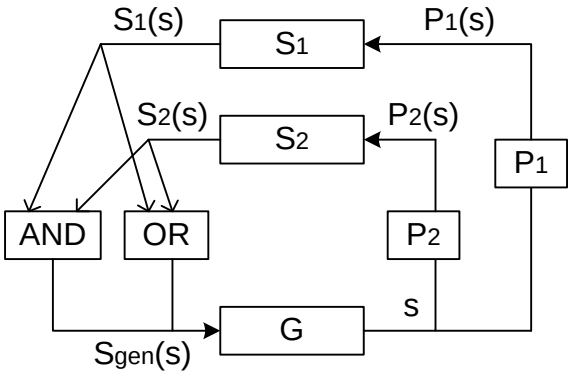


Fig. 4.5 General decentralized supervisory control with two supervisors

Fig. 4.5 depicts the architecture of disjunctive decentralized supervisory control with two supervisors. The generalized decentralized supervisor S_{gend} is defined as

follows: $S_{gend}(s) = P_{ce}\lfloor S_{conj}(s)\rfloor \cup P_{cd}\lfloor S_{disj}(s)\rfloor \cup \Sigma_{uc}$, where P_{ce} and P_{cd} are the following projection mappings: $P_{ce} : \Sigma \to \Sigma_{c,e}$ and $P_{cd} : \Sigma \to \Sigma_{c,d}$. The prefix closed language $L(S_{gend} / G)$ generated in the general architecture is:

- $\varepsilon \in L(S_{gend} / G)$;
- $\lfloor s \in L(S_{gend} / G)\rfloor \wedge [s\sigma \in L(G)] \wedge \lfloor \forall i, \sigma \in S_{gdec}(s)\rfloor \Leftrightarrow s\sigma \in L(S_{gdec} / G)$.

The marked language is $L_m(S_{gend} / G) = L(S_{gend} / G) \cap L_m(G)$, Yoo and Lafortune (2002).

It is important to note that when the sets $\Sigma_{c,i}$ are mutually disjoint, the three architectures (general, disjunctive and conjunctive) are the same. The reason is that each controllable event is controlled by only one supervisor, i.e. the event is enabled if and only if the corresponding supervisor enables it.

4.2.5. Nonblocking decentralized supervisory control

In this section we present the conditions under which the conjunctive and disjunctive decentralized supervisors are nonblocking. Recall from Section IV.1.6 that a language generated by G is nonblocking if $\overline{L_m(G)} = L(G)$, and from Section IV.2.1 $L_m(S / G) = L(S / G) \cap L_m(G)$. Thus, $L(S / G)$ is said to be nonblocking supervisor if $\overline{L_m(S / G)} = L(S / G)$, i.e. S is nonblocking for G if every state trajectory of the closed loop process can be extended to reach the set of marked states of G.

4.2.5.1. Nonblocking conjunctive decentralized supervisor

Wonham and Ramadge (1988) prove that the conjunctive supervisor $S_{conj} = S_1 \wedge S_2$ is nonblocking if and only if the marked languages $L_m(S_1/G)$ and $L_m(S_2/G)$ are nonconflicting (*Proposition* 4.2).

Here we restate the proof from Wonham and Ramadge:

$S_{conj} = S_1 \wedge S_2$ is nonblocking

$$\Leftrightarrow \overline{L_m[(S_1 \wedge S_2)/G]} = L[(S_1 \wedge S_2)/G],$$

$$\Leftrightarrow \overline{L_m(S_1/G) \cap L_m(S_2/G)} = L(S_1/G) \cap L(S_2/G),$$

$$\Leftrightarrow \overline{L_m(S_1/G) \cap L_m(S_2/G)} = \overline{L_m(S_1/G)} \cap \overline{L_m(S_2/G)},$$

$$\Leftrightarrow L_m(S_1/G) \text{ and } L_m(S_2/G) \text{ are nonconflicting}.$$

□

With the extension of the nonblocking property for finite number of languages, the above proof is valid for finite set of languages. Hence, the conjunction decentralized supervisor $S_{conj} = S_1 \wedge S_2 \wedge \ldots \wedge S_n$ is nonblocking with respect to *G* if all individual supervisors $S_1, S_2, \ldots, S_n$ are nonconflicting.

4.2.5.2. Nonblocking disjunctive decentralized supervisor

Theorem 3.4.1 by Wonham (2006) states that there exist a nonblocking supervisory controller $L_m(S/G)$ for *G* if and only if $L_m(S/G)$ is controllable with respect to *G* and $L_m(S/G)$ is $L_m(G)$-closed. Thus, to prove that $S_{disj} = S_1 \vee S_2$ is nonblocking, we have to

show that (i) S_{disj} is controllable with respect to G and (ii) S_{disj} is $L_m(G)$-closed. Let $S_1 = L_m(S_1/G),\ S_2 = L_m(S_2/G)$ then $S_{disj} = L_m(S_1/G) \cup L_m(S_2/G)$.

Proof:

(i) need to show that if $L_m(S_1/G)$ and $L_m(S_2/G)$ are controllable, then

$L_m(S_1/G) \cup L_m(S_2/G)$ is also controllable.

$$\begin{aligned}\overline{L_m(S_1/G) \cup L_m(S_2/G)}\Sigma_{uc} \cap L(G) &= \left(\overline{L_m(S_1/G)} \cup \overline{L_m(S_2/G)}\right)\Sigma_{uc} \cap L(G) \\ &= \left(\overline{L_m(S_1/G)}\Sigma_{uc} \cap L(G)\right) \cup \left(\overline{L_m(S_2/G)}\Sigma_{uc} \cap L(G)\right) \\ &\subseteq \overline{L_m(S_1/G)} \cup \overline{L_m(S_2/G)} \\ &= \overline{L_m(S_1/G) \cup L_m(S_2/G)}\end{aligned}$$

(ii) need to show that if $L_m(S_1/G)$ and $L_m(S_2/G)$ are $L_{\text{m}}(G)$-closed, then $L_m(S_1/G) \cup L_m(S_2/G)$ is also $L_{\text{m}}(G)$-closed.

$$\begin{aligned}\overline{L_m(S_1/G) \cup L_m(S_2/G)} \cap L_m(G) &= \left(\overline{L_m(S_1/G)} \cup \overline{L_m(S_2/G)}\right) \cap L_m(G) \\ &= L_m(S_1/G) \cup L_m(S_2/G).\end{aligned}$$

□

With the extension of the nonblocking property for finite number of languages, the above proof is valid for finite set of languages. Hence, the disjunction decentralized supervisor $S_{disj} = S_1 \vee S_2 \vee \ldots \vee S_n$ is nonblocking with respect to G if all individual supervisors $S_1, S_2, \ldots, S_n$ are controllable and $L_{\text{m}}(G)$-closed.

4.2.5.3. Nonblocking general decentralized supervisor

Based on the above two proofs, in case of a general decentralized supervisor S_{gend}, i.e. $S_{gend} = \left(S_{conj_1} \wedge \ldots \wedge S_{conj_p}\right) \cup \left(S_{disj_1} \vee \ldots \vee S_{disjq}\right)$, we may say that S_{gend} is nonblocking

with respect to G if all individual conjunctive supervisors $S_{conj_1} \wedge \ldots \wedge S_{conj_p}$ are nonconflicting and all individual disjunctive supervisors $S_{disj_1} \vee \ldots \vee S_{disjq}$ are controllable and $L_m(G)$-closed.

Chapter Five

Taxonomy of DRT Systems, DRT Modeling with FA and Illustrative Example

In this chapter we present an approach of modeling DRT systems as DESs and their real time control with centralized and modular supervisors. To facilitate the formalism of modeling and analysis of the systems, we first present taxonomy of the DRT systems according to their characteristics relevant to DES representation.

5.1. Taxonomy of DRT systems

Every DRT system is determined with the following three component characteristics: origin/destination characteristics, vehicle fleet characteristics, and transportation demand characteristics. Based on these components, DRT systems can be classified in the manner described below.

5.1.1. Origin and destination considerations

- *Many to one* – these systems transport passengers or freight from many origin locations to one destination location. Typical examples are systems with single commodity PDP, e.g. an armored vehicle that transports money from local branches to the head office of a bank; on-demand air charter (taxi) service

utilizing Dial-a-Flight-Problem (DAFP) picking passengers from small airports and transferring them to a larger airport (hub).

- *Many to few* - these systems serve more than one, but a fixed number of origin or destination locations; Example include *n*-commodity PDP, where *n* types of goods are considered and each commodity requires single pickup and delivery node, military and ARE service, the emergency services like police patrols, ambulance fleet management.
- *Many to many* - these systems serve large and usually random number of origin and destination locations. Typical examples are based on Urban Courier Service Problem (UCSP), taxi cab service.

5.1.2. Vehicle fleet characteristics

- Systems with fleet of vehicles where no capacity constraints are considered like postal and courier service, emergency fire fighting.
- Systems with a homogeneous fleet with the same load capacity and speed capabilities like taxi cabs, shuttle vans.
- Systems with a heterogeneous fleet with the different capacities and/or speed capabilities like air charters operating with different size airplanes.
- Systems with constraints on length or duration of vehicle routes – e.g. range of an aircraft, pilot shift restrictions in air taxi systems.
- Systems where the fleet is located at one central or multiple depots. After the end of service all the vehicles must return back to the depot(s) – like in taxi operators.

- Systems where vehicles are subject to unpredicted stoppages or re-routings like caught in a traffic jam, detours, or breakdown. Examples include all the land transportation systems operating in urban areas.

5.1.3. Transportation demand characteristics

- Systems with a priori known static demand that accept service reservations made in advance. Classical examples are the school bus service and fixed route dial-a-ride systems working with advance reservations.
- Systems with dynamic service demand where every customer request is eligible for immediate consideration and requires real time adjustments of the already established routes and schedules. Typical example is a courier service system.
- Systems where some groups of passengers are given priority over the rest or have special service requirements. Examples include service of people with disabilities, air charter transportation of special cargos.

5.2. Modeling of DRT systems with FA

The application of SCT in DRT control, where it is required to provide automated system update in real time is based on the following three groups of models, Seow and Pasquier (2005):

- Plant - models of the uncontrolled behavior of system's components with FA;
- Specifications - models of control objectives (behaviors) to be specified with FA;
- A supervisory controller to be synthesized.

The taxonomy presented in Section V.1 is used here as a guideline to present the plant and specifications automata modeling various feathers of a DRT system operation.

- To model a system with origins and destination locations from a fixed finite set, the origins/destinations can be presented as states, and the travels between every two destinations as events. For example a small air taxi system covering the demand over four airports A, B, C, and D is presented in Fig. 5.1. Any airport is reachable from the other airports by the used jets.

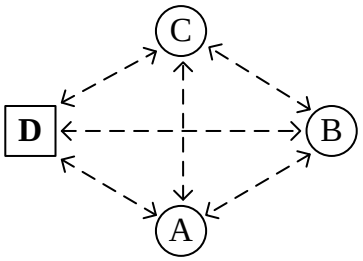


Fig. 5.1 Simple air taxi DRT system operating at four airports.

All possible flights of jet *j* are depicted in automaton $pjet_j$ of Fig.5.2. The set of states of $pjet_j$ $Q = \{0,1,2,3\}$ represents all the possible locations of the jet – i.e. the four airports D, A, B, and C respectively and the set of transitions $\Sigma = \{jDA,\ldots,jCD\}$ – i.e. the events represent the flights of jet *j* between the corresponding airports (e.g. *jDA* means that *j* is in flight from the depot D to airport A).

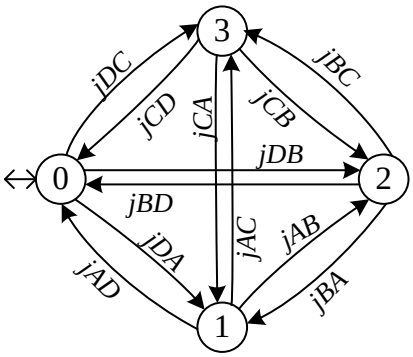


Fig.5.2 Automaton $pjet_j$ - the possible locations and flights of jet *j*.

- In modeling systems with depot(s), where fleet starts and ends its operation, specific automata must assure that all the allowed sequences of transitions of the fleet (events) start and end at the state representing the depot(s).
- To model a system with constraints on the length of vehicle routes, automata of vehicle behavior should limit the number of the possible consecutive transitions. For example in the air taxi system described in Fig.5.1, if airport D is the system's depot and each jet is allowed maximum three flights per trip, the automaton $trip_j$ in Fig. 5.3 guarantees that all the flights start and end at D, and jet *j* performs at most three flights per trip. State 0 represents *j* being located at depot D, state 1 – all the possible locations of *j* after the first flight from D, and state 2 all the possible locations of *j* after one more flight. Since each sequence of transitions ends at state 0 the max allowed flights are three.

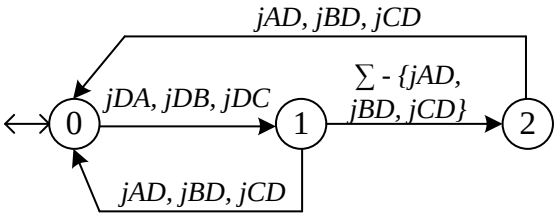


Fig.5.3 Automaton $trip_j$ – the maximum allowed flight within a trip.

- If vehicles are subject to unpredicted stoppages like in traffic jams or breakdowns, the events that lead to these states are to be introduced in the plant model. For example in modeling a land DRT system, if both traffic jams and breakdowns are considered, the automaton $vehust_j$ in Fig. 5.4 describes such a behavior of vehicle j. When j is in service (state 1) it may get in a jam (state 3) and after the jam is eliminated it is back in service; if it breakdowns (state 2), after repair it is in initial standby state (state 0).

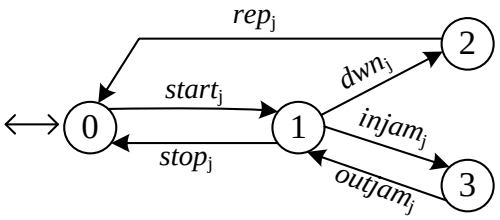


Fig.5.4 Automaton $vehust_j$ – vehicle j in unpredicted stoppages.

- In modeling systems where the vehicles have capacity constraints, the number of passengers on board or loaded cargo units represent different states of the vehicle and the picking up or dropping of a passenger or delivery of a cargo –events. In

the air taxi system example, if the seating capacity of jet *j* is two passengers, the automaton cap_j in Fig. 5.5 limits the possible pickups and drops off. State 0 represents the jet without passengers on board, states 1 and 2 represent the jet with 1 and 2 passengers on board, respectively.

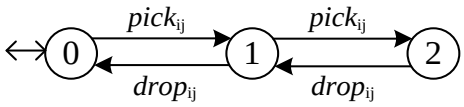


Fig.5.5 Automaton cap_j - jet *j* may pickup at most two passengers.

- If a prioritization in the service of a group of passengers is needed, a group of automata should impose that the service of the rest of the passengers starts after all the passengers with priority have been served. For example the automaton $prior_i$ in Fig.5.6 assures that all the reassigned passengers (event ras_{ij}) are helped before the remaining passengers that have to be assigned (event ac_{ij}) for first time.

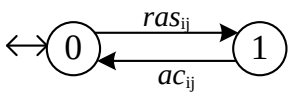


Fig.5.6 Automaton $prior_i$ gives priority of reassigned passengers.

5.3. Illustrative example of a small air-charter service operation

In this section we develop a DES model that provides nonblocking behavior of a DRT system, capable of making real time decisions regarding the acceptance of passenger requests. The model can also cover the case of service with minimum possible fleet size, i.e. a new vehicle is being activated only if none of the currently active vehicles can meet a particular service request. In addition, there is a constraint on the length of vehicle service operation during a working shift. An example of a destination-specific DRT system is used. It is based on DDAFP defined by Cordeau at al. (2004). The system is a small air taxi operator providing on-demand air charter service. Such a business encounters an increased interest because of its ability to quickly respond to the customer's needs and flexible service.

The modeling of this type of service is close to D-DARPMADO operations studied by Sadeh and Kott (1996), and to a large group of emergency and rescue air logistics problems, Shen, Dessouky and Ordonez (2005). The service of an air taxi operator is similar to the emergency ARE problem in the following characteristics: high dynamics of operations that requires immediate decision about the feasibility of a request and real time update of jets routings and schedules; limited jet capacities with small number of seats or beds; limited length of flights; possible closures of some airports causing unpredicted changes of the flights. The dissimilarities are that the origins of the requests belong to a set of airports in the air taxi service and could be anywhere in the covered region in ARE environment, and the available jets are not subject to change or breakdown during service.

An air taxi service operates over a given set of airports, which implies that flight and schedule optimizers can be successfully applied. The operation is planned as "per-seat on-demand" service. Customers book seats online as they do with airline service, except there are no fixed schedules.

5.3.1. Problem description of a small air-charter system's operation

Consider an air charter DRT system which covers the demand over a fixed set of four airports (P = 4) by means of a homogeneous fleet of jets $j = 1,\ldots,M$ (Fig.5.1). A jet may fly from any to any other airport. One of the airports (D) serves as a depot, where all the jets are kept and after the end of their services must return. The fleet consists of very light jets (VLJ) with seating capacity of two passengers. The system receives randomly initiated passenger requests $i = 1,\ldots,N$ (N is the current number of passengers to serve) with origin and destination locations, and provides real time answers – i.e. the dispatcher must decide in real time whether the system can serve a particular request, assign the passenger to a jet, and route or reroute that jet.

To formalize the length of service of a jet, we define a *flight of a jet* within the system to be the route from one airport to another; a *trip of a jet* to be a sequence of flights which starts and ends at D. To incorporate the limits of pilot duty, VLJ flight range, etc. the following constraints are included:

- At most two intermediate stops are allowed during a trip;
- A jet may complete up to one trip through a working shift.

Hence, a working shift (i.e. a trip) may include up to three flights.

Let at the beginning of a shift the system receives a request from $passenger_1$, who wants to fly from airport A to airport C. The control procedure needs to compute the possible behavior of jet_1, so that $passenger_1$ will be picked from its location and transported to the desired destination.

5.3.2. DES modeling of a small air charter DRT system

The set of all the events Σ of the considered system is summarized in Table 3. The pickup and drop off events have two indexes representing the number of the passenger and the number of the jet serving that passenger. The first event is controllable (can be controlled by the operator), while the second one is uncontrollable (whether a passenger will reach the final destination depends on airport condition, flight condition, etc. – all uncontrolled). Each flight is labeled as a combination of a digit followed by two letters. The digit represents the number of the jet and the letters – the origin and the destination correspondingly. All flights are considered as controllable events.

Table 3 The set Σ of all the events of the small air charter.

Process	Events – c: controllable; u: uncontrollable		
Passenger's demand service	$pick_{ij}$	Passenger *i* picked with jet *j*	c
	$drop_{ij}$	Passenger *i* transported with jet *j*	u
Flights	*jDA*	Jet *j* flies from D to A	c
	jDB	Jet *j* flies from D to B	c
	jDC	Jet *j* flies from D to C	c

Table 3 (Continued)

	jAB	Jet *j* flies from A to B	c
	jAC	Jet *j* flies from A to C	c
	jAD	Jet *j* flies from A to D	c
	jBA	Jet *j* flies from B to A	c
	jBC	Jet *j* flies from B to C	c
	jBD	Jet *j* flies from B to D	c
	jCA	Jet *j* flies from C to A	c
	jCB	Jet *j* flies from C to B	c
	jCD	Jet *j* flies from C to D	c

5.3.2.1. Computation of centralized supervisor

We apply the procedure of Section V.2. and develop the following three models:

Plant model: the plant consists of two automata - $pjet_1$ (Fig. 5.7) models the possible behavior of jet_1, and $pass_1$ (Fig. 5.8) describes the behavior of $passenger_1$.

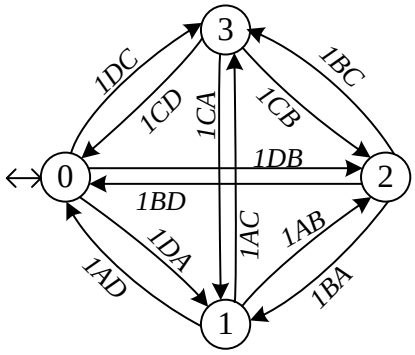


Fig.5.7 Automaton $pjet_1$.

The set of states Q and the set of events Σ of $pjet_1$ are the same Q and Σ for automaton $pjet_j$ of Fig.5.2.

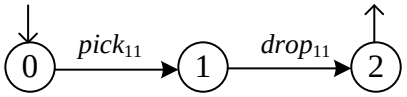
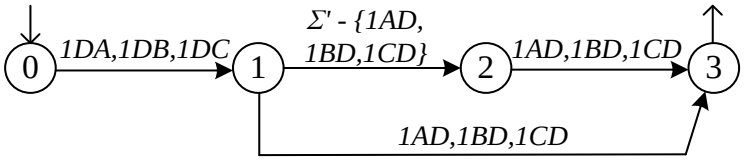


Fig.5.8 Automaton $pass_1$.

In automaton $pass_{1,}$ $passenger_1$ releases a service request at the initial state 0, next it is picked by jet_1 (state 1) and jet_1 drops off $passenger_1$ (state 2).

In this case the plant is obtained by parallel composition of the two automata, i.e. $Plant_1 = pjet_1 \parallel pass_1$.

Specification models: two specifications are considered - automaton $trip_1$ (Fig. 5.9) ensures that jet_1 will make up to three flights, and automaton $paspd_1$ (Fig. 5.10) specifies after which flights $passenger_1$ can be picked and dropped off.



$$Selfloop = \{pick_{11}, drop_{11}\}$$

Fig.5.9 Automaton $trip_1$.

The states and events of $trip_1$ are analogous to the state and event sets of automaton $trip_j$ of Fig.5.3. In the $trip_1$ automaton the selfloops (not shown) are adjoined to each state and account for the events that are irrelevant to the specification, but may be

executed in the model. In the graphs of the automata of this section Σ' denotes the set of all flights of jet_1, i.e. $\Sigma' = \{1DA, \ldots, 1CD\}$.

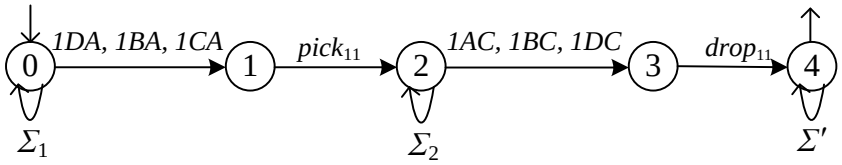


Fig.5.10 Automaton $paspd_1$.

In state 0 of $paspd_1$ jet_1 can fly from the depot D to any location and $passenger_1$ is at airport A. Only the flights that end at airport A allow the jet to get to the passenger (state 1), and after picking them up (state 2) jet_1 can fly to any location. The flights that end at airport C take the system to state 3, and after dropping off $passenger_1$ at its destination, the system reaches in the marked state 4. The event sets of $paspd_1$ Σ_1 and Σ_2 denote the flight sets $\Sigma \setminus \{1DA, 1BA, 1CA\}$ and $\Sigma' \setminus \{1AC, 1BC, 1DC\}$ respectively.

The automaton $Spec_1 = trip_1 \times paspd_1$ represents the synchronization of both specification automata. It has 12 states and 33 transitions.

Synthesis of the centralized supervisor: the intersection of languages marked by $Plant_1$ and $Spec_1$ automata provides the centralized supervisor (CS_1), i.e. $CS_1 = Plant_1 \cap Spec_1$. The described three steps of the procedures are performed with XPTCT-software developed by Systems Control Group in the Dept. of Electrical & Computer Engineering at University of Toronto, (Design Software: XPTCT). The computed CS_1 is controllable with 5 states and 5 transitions: i.e. starting from the initial

position at depot D (state 0) jet_1 must fly to airport A (state 1), pick $passenger_1$ (state 2), flies to airport C (state 3), drops off the passenger (state 4) and flies back to depot D, Fig. 5.11.

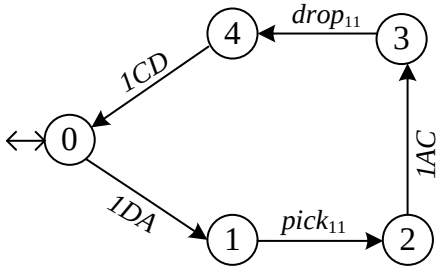


Fig. 5.11 Supervisor CS_1.

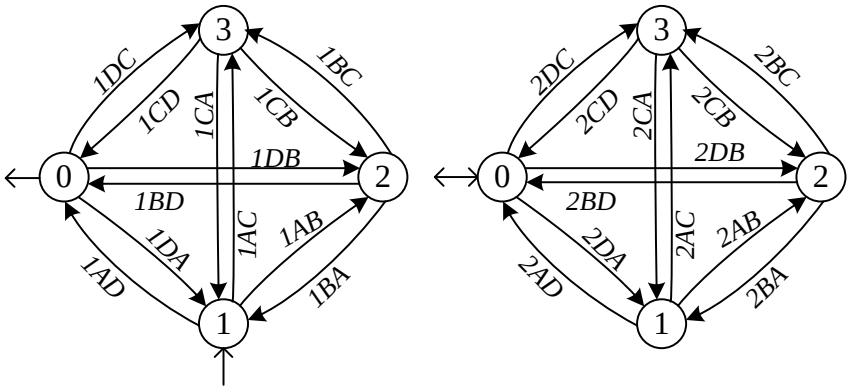
5.3.2.2. Computation of modular supervisor

Let at the current time instant jet_1 is at airport A picking $passenger_1$ and a new request is received: $passenger_2$ wants to fly from airport B to airport D. The problem consists of making an immediate decision if it is feasible to accept the new request given the available resources (active jet_1) and the existing schedule. In the case of the small air charter system, the control procedure would check if the active jet will be enough to meet the demand, and if not, the scheduling and routing for two jets should be developed.

Thus, a new jet is introduced in the system – jet_2. To illustrate the modularity of the supervisory synthesis, the three steps of the procedure will be developed in such a way, that two supervisors will be synthesized – one controlling system operation with jet_1 only, and one – controlling service with both jet_1 and jet_2.

Plant model is the synchronization of the following two pairs of automata:

- $pjet_j\ (j=1,2)$ (Fig. 5.12a and 5.12b) models the possible flights of jet_1 and jet_2, respectively. Automaton $pjet_1$ is updated with the current location of jet_1 – airport A. Thus, the initial state of $pjet_1$ at this step is 1, not 0, i.e. $q_0 = \{1\}$.



a) Automaton $pjet_1$ b) Automaton $pjet_2$

Fig.5.12 Automata $pjet_j\ (j=1,2)$.

Fig. 5.13 depicts the parallel synchronization of automata $pjet_1$ and $pjet_2$. The flights of jet_1 are in continuous line and the flights of jet_2 - in dash. To avoid obscurity only the flights in the first row and column are labeled.

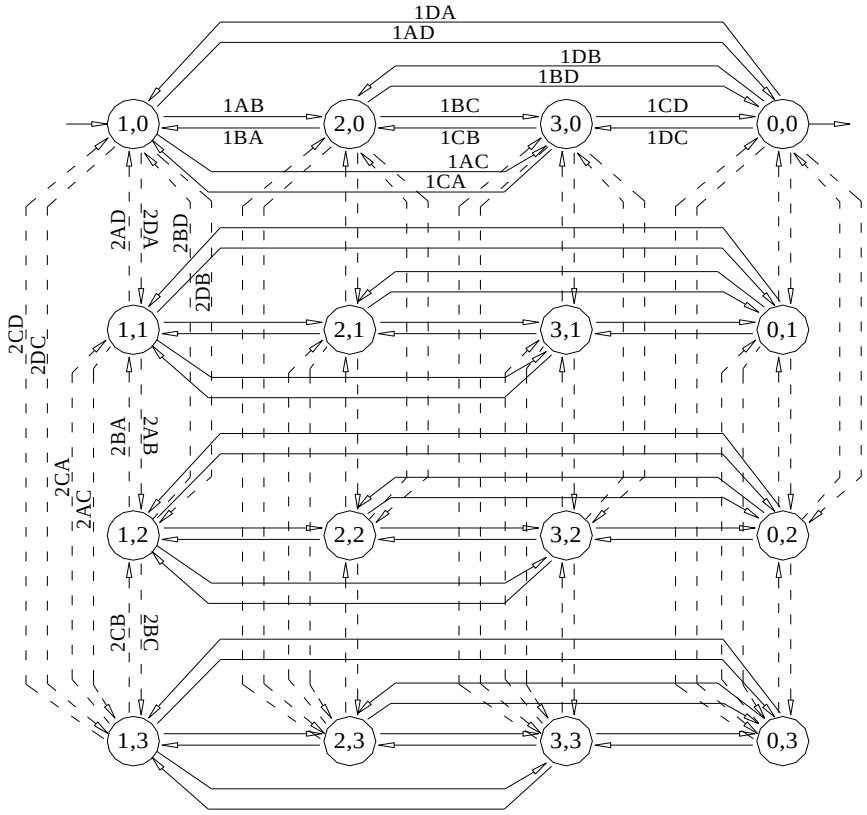


Fig. 5.13 Parallel synchronization of automata $pjet_1$ and $pjet_2$.

- $pass_i\ (i=1,2)$ (Fig. 5.14a and 5.14b), where $pass_1$ encounters that $passenger_1$ is to be picked by jet_1 and $passenger_2$ can be picked by any jet.

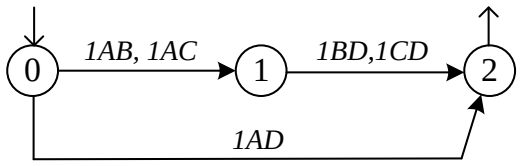
a) Automaton $pass_1$. b) Automaton $pass_2$.

Fig.5.14 Automata $pass_i\ (i=1,2)$.

After synchronization, the plan automaton of this case $Plant_2$ is obtained: $Plant_2 = pjet_1 \| pjet_2 \| pass_1 \| pass_2$ has 144 states and 1152 transitions.

The specifications of this case are modeled with the following two pairs of automata:

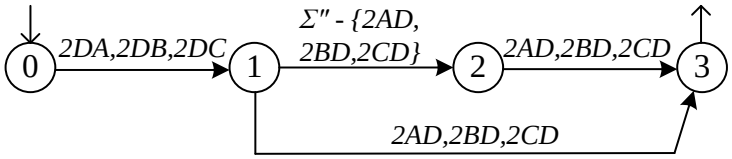
- $trip_j \ (j = 1,2)$ (Fig. 5.15a and 5.15b), where $trip_1$ is updated with the current position of jet_1 and encounters that jet_1 has two flights left, i.e. at state 0 jet_1 is at airport A, at state 1 it is either at B or C and at state 2 it is back at depot D. Automaton $trip_2$ is analogous to $trip_j$ from Fig. 5.9.



$Selfloop = \{pick_{11}, drop_{11}, \Sigma''\}$

a) Automaton $trip_1$.

Here $\Sigma'' = \{2DA, \ldots, 2CD\}$.

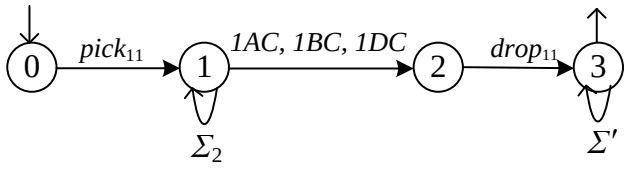


$Selfloop = \{pick_{11}, drop_{11}, pick_{21}, drop_{21}, pick_{22}, drop_{22}, \Sigma'\}$

b) Automaton $trip_2$.

Fig. 5.15 Automata $trip_j \ (j = 1,2)$.

- $paspd_i$ (i = 1, 2) (Fig. 5.16a and 5.16b), $paspd_1$ is updated with the current position of jet_1 and $paspd_2$ covers the possibilities that $passenger_2$ can be picked by either jet_1 or jet_2.



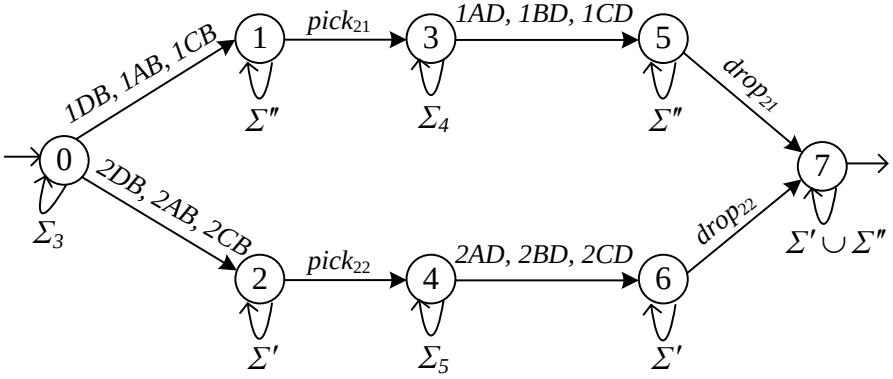
$$Selfloop = \Sigma'' \cup \{pick_{21}, pick_{22}, drop_{21}, drop_{22}\}$$

a) Automaton $paspd_1$.

Recall that $\Sigma' = \{1DA, \ldots, 1CD\}$ and $\Sigma_2 = \Sigma' \setminus \{1AC, 1BC, 1DC\}$.

In addition, $\Sigma_3 = \{\Sigma' \setminus \{1AB, 1CB, 1DB\}\} \cup \{\Sigma'' \setminus \{2AB, 2CB, 2DB\}\}$,

$\Sigma_4 = \{\Sigma''\} \cup \{\Sigma' \setminus \{1AD, 1BD, 1CD\}\}$, $\Sigma_5 = \{\Sigma'\} \cup \{\Sigma'' \setminus \{2AD, 2BD, 2CD\}\}$.



b) Automaton $paspd_2$.

Fig. 5.16 Automata $paspd_i$ (i = 1, 2).

The state space and transitions of the upper rung of $paspd_2$ cover the case when $passenger_2$ is picked and dropped off by jet_1, and the bottom rung consider the possibility

that $passenger_2$ is transported by jet_2. Each of these rungs has three intermediate states (1-3-5 or 2-4-6, respectively) and 53 transitions.

Synthesis of the modular supervisor:

- First, the planning procedure may check if jet_1 can transport both passengers. The required specification for that case $Spec_2$ is computed with the product of automata $trip_1$, $paspd_1$, and $paspd_2$, i.e. $Spec_2 = trip_1 \times paspd_1 \times paspd_2$. The supervisor MS_1 is obtained as the intersection of $Plant_1$ and $Spec_2$, i.e. $MS_1 = Plant_1 \cap Spec_2$ (fig. 5.17). It has 19 states and 18 transitions.

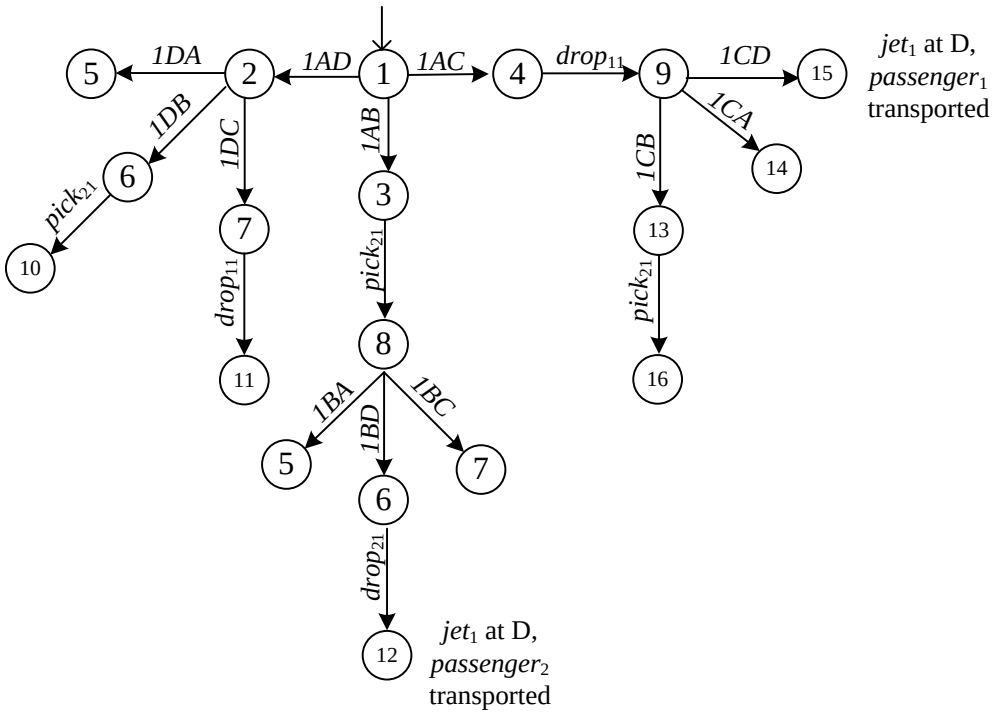


Fig.5.17 Supervisor MS_1.

However, there is no marked state on the graph. When the system gets to state 12 jet_1 arrives at depot D and has dropped off only $passenger_2$, and when at state 15 jet_1 arrives at D and has dropped off only $passenger_1$. Thus, jet_1 cannot transport both

passengers in one shift. To meet the demand the service provider has to use one more jet - jet_2.

To compute the specification when jet_2 is introduced, the control procedure may use that $passenger_1$, which is already picked by jet_1 is to be transported by the same jet hence, $passenger_2$ should be picked by jet_2. Thus, one module of specifications is $Specm_1 = trip_1 \times paspd_1$ for $passenger_1$ - jet_1 coordination depicted in Fig.5.18, and another specification $Specm_2 = trip_2 \times paspd_2$ for $passenger_2 - jet_2$, Fig.5.19.

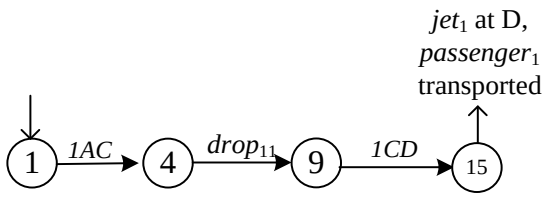


Fig.5.18 $Specm_1$ - synchronization of jet_1 and $passenger_1$.

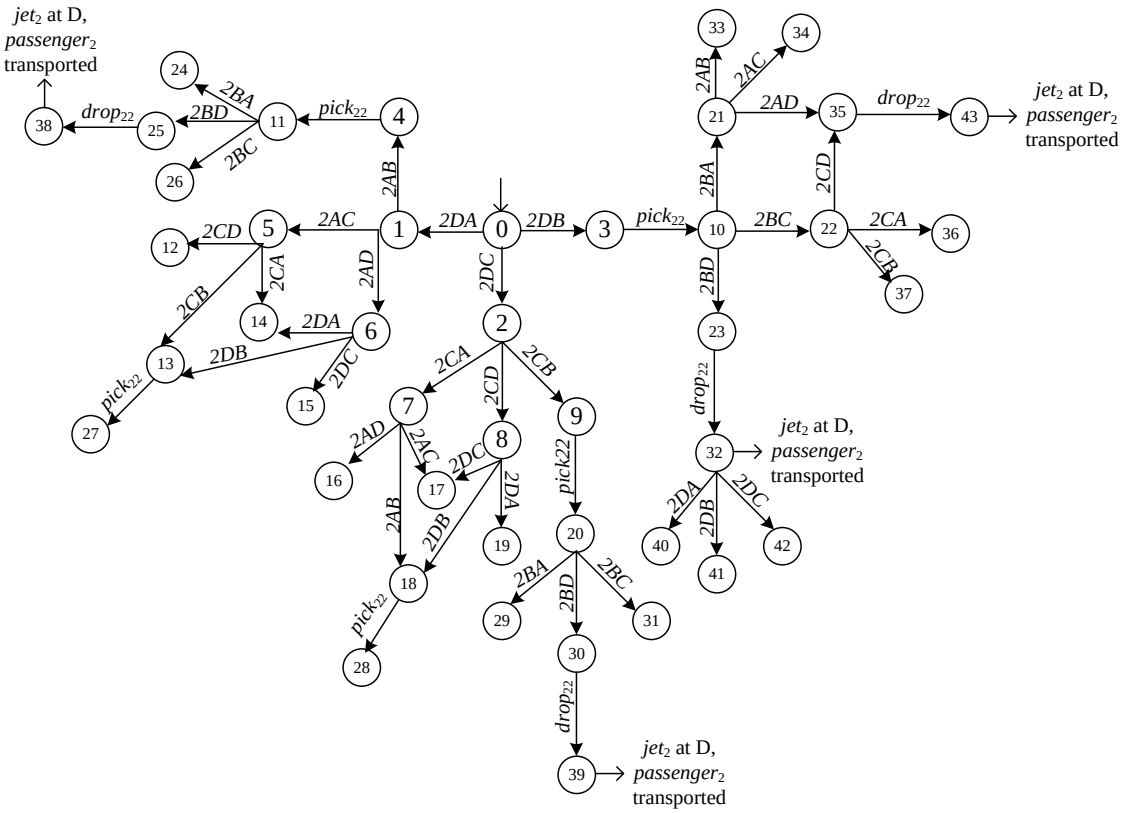


Fig. 5.19 $Specm_2$ - synchronization of jet_2 and $passenger_2$.

- The modular supervisors for transportation of $passenger_1$ - SP_1 and for transportation of $passenger_2$ – SP_2 are computed with the intersections of $Plant_2$ with $Specm_1$ (Fig. 5.18) and $Specm_2$ (Fig. 5.20), respectively. $SP_1 = Plant_2 \cap Specm_1,\ SP_2 = Plant_2 \cap Specm_2$.

$Specm_1$ provides the only possible way to complete service of $passenger_1$ and its supervisor is the same as the specification, i.e. $SP_1 = Specm_1$.

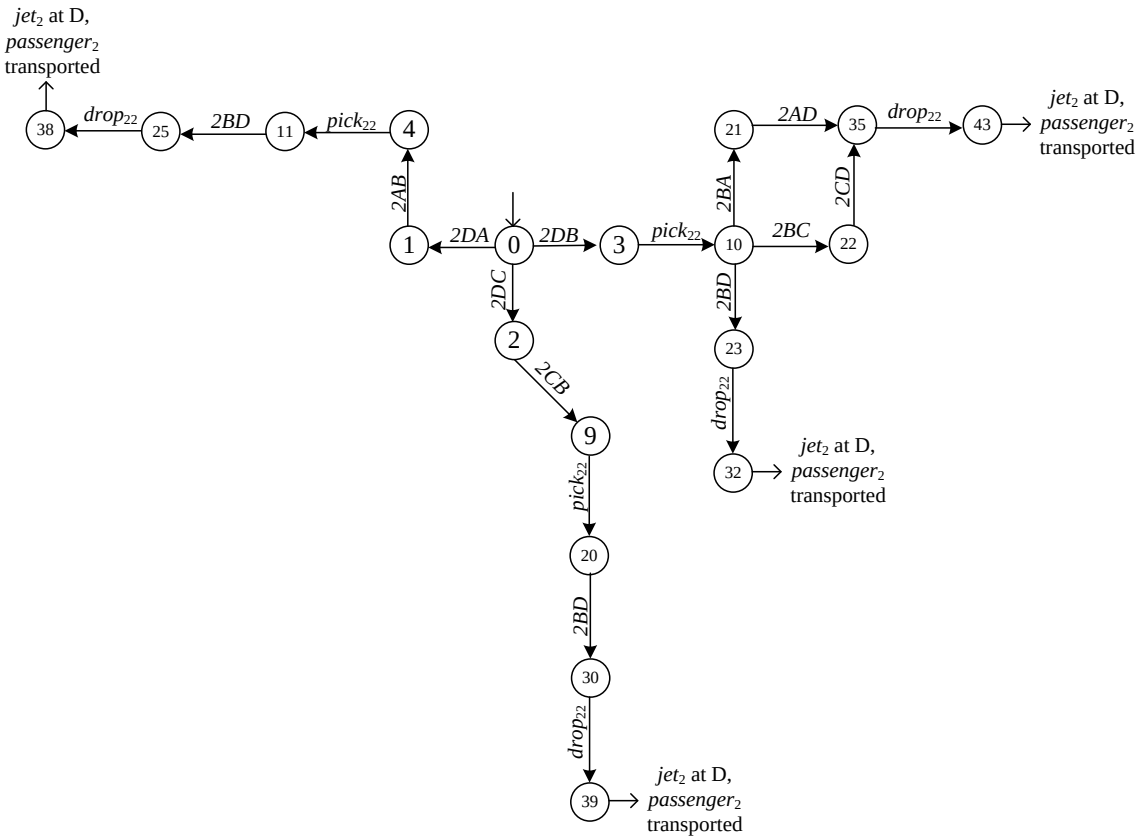


Fig.5.20 Language SP_2.

Since both supervisors SP_1 and SP_2 have no common transitions, the modular supervisor MS_2 is their union, $MS_2 = SP_1 \cup SP_2$. It has 23 states and 22 transitions. As a comparison, the centralized supervisor for transportation of two passengers by two jets calculated with XPTCT-software has 55 states and 114 transitions.

5.2.2.3. Computational complexity of supervisor synthesis

Let the plant generator G be modeled with r states and two supervisors S_1 and S_2 with p_1 and p_2 states respectively, jointly control the system. Cassandras and Lafortune (1999) discuss the significant computational and memory savings of modular control. The supervision of G can be interpreted as the product $S_1 \times S_2 \times G$. If the centralized supervisor $S = S_1 \times S_2$ is built, we need to store totally $p_1(p_2)$ states and in modular control – only $(p_1 + p_2)$ states. In the worst case, the computational complexity for centralized supervisor synthesis is $O(p_1 p_2 r)$ and $O(max(p_1, p_2)r)$ for the modular supervisor.

Consider the general case when at a given state of the DRT system the available m number of jets are supposed to transport n passengers. Let the seating capacity of the jets is two passengers and $m \geq \frac{n}{2}$. There will be m_0 jets that have not assigned passengers, m_1 jets that have one passenger assigned, and m_2 jets that have two passengers assigned, i.e. $m = m_0 + m_1 + m_2$. Passengers can be split into two groups: n_0 that are not assigned yet and n_1 that are already assigned to any jet, i.e. $n = n_0 + n_1$.

Obviously, $n_1 = 2m_2 + m_1$. Thus, for the number of the unassigned passengers n_0, we have:

$$\begin{aligned} n_0 &= n - n_1 \\ &= n - 2m_2 - m_1. \end{aligned}$$

For all n_0 requests the control procedure is to check for feasibility of service first with the m_1 jets. As we saw in V.3.2.2, this is done as a product of automata $trip_j$ and $paspd_i$.

However, the numbers of states and transitions of automata $trip_j$ and $paspd_i$ depend on the current location and the number of remaining flights of jet_j. If jet_j is at the depot D and can make a 3 flight trip (e.g. $trip_2$ of Fig. 5.15b), automaton $trip_j$ will have four states and 18 transitions plus a selfloop of $2n_1 + 2n_0 m_1 + 12(m_1 - 1)$ transitions at each state. If jet_j is at an airport and has two flights remaining in its trip (e.g. $trip_1$ of Fig. 5.15a) then $trip_j$ will have 3 states and 5 transitions plus a selfloop of $2n_1 + 2n_0 m_1 + 12(m_1 - 1)$ transitions at each state. For each unassigned $passenger_i$ (from 1 to n_0) the corresponding automaton $paspd_i$ will have $(m_0 + m_1)$ rungs of three states and each of them will generate additional $36(m_0 + m_1) - 25$ transitions and a selfloop of $2n_1$ transitions for each state. In addition, $(m_0 + m_1)$ automata cap_j of Fig 5.5 with 3 states and $4n_0(m_0 + m_1)$ transitions should be used to secure that up to two passengers will be assigned to each jet.

In any real case combination of n_0, n_1, m_0 and m_1, the product of these automata will be large enough to cause computational complexity in the synthesis of the modular supervisor. Thus, we need a procedure that will limit the check for a feasible jet for every new service request. In the next chapter we present such a method, based on decentralized supervisory control.

Chapter Six

Decentralized Supervisory Control of Concurrent DES

In Section 4.2.4. we introduced the decentralized supervisory control (DSC) with three modeling architectures and in Section 4.2.5 the nonblocking conditions of the decentralized control architectures were presented. In this chapter we consider the decentralized control of DES with specialization to local supervisory control (SC) and concurrent systems. The advantages of the DSC are illustrated in an example of emergency ARE DRT service.

6.1. Decentralized control of concurrent DESs

Concurrent DESs are defined as collections of components (subsystems) that perform simultaneously and may interact with each other. Consider a DES G composed of n concurrent subsystems G_i, with event sets $\Sigma_i, i = 1,2,...,n$. Suppose that for each subsystem G_i there is a local supervisor S_i that observes and controls only the events of Σ_i. The global controlled DES can be obtained as the concurrent operation of the locally controlled subsystems $S_i / G_i, 1 \le i \le n$. Thus, the problem of decentralized control of concurrent DESs is to find the conditions under which local synthesis and control for any

specifications of G_i do not result in loss of optimality compared to the global supervisor's control S/G, and control of one subsystem G_i never incurs blocking in the other subsystem G_j.

Recall from the *Controllability theorem* introduced in Section 4.2.2 that controllability of the language of the desired behavior is a necessary and sufficient condition for the existence of a supervisor that achieves this behavior for a given DES under the complete observation of the events. In the case of decentralized control when there are *n* local supervisors observing and controlling their corresponding sets of events Σ_i, Cieslak at al.(1988) introduced the condition of *co-observability* if the controlled behavior is given as a prefix closed language. Lafortune at al. (2001) relaxed the conditions of co-observability for the existence of local supervisors in the conjunctive, disjunctive and general decentralized architectures.

Willner and Heymann (1991) introduce the notion of *separability* of a language - L is said to be separable with respect to (w.r.t.) $\{\Sigma_i\}_{i=1}^n$ if there exists a set of languages $L_i \subset \Sigma^*_i$, $1 \le i \le n$ called *a generating set* of L, such that $L = \|_{i=1}^n L_i$. For a finite set of languages $\{L_i \subset \Sigma^*_i\}_{i=1}^n$ the parallel composition of $\{L_i\}$, denoted $\|_{i=1}^n L_i$ is defined as $\|_{i=1}^n L_i = \cap_{i=1}^n P_i^{-1}(L_i)$. Recall, P_i is the natural projection $\mathrm{P_i} : \Sigma^* \to \Sigma_\mathrm{i}^*$.

Consider a set of concurrent DESs $G_i = (Q_i, \Sigma_i, \delta_i, q_{0i}, Q_{mi})$, $1 \le i \le n$ with event partitions $\Sigma_i = \Sigma_{ic} \cup \Sigma_{iuc}$ such that

$$(\forall i \neq j)\Sigma_{iuc} \cap \Sigma_j = \varnothing \qquad \text{(eq. 6.1)}$$

This assumption means that there is no synchronization between the uncontrolled events of the systems. Willner and Heymann (1991) prove that separability under assumption (eq. 6.1) is a necessary and sufficient condition that guaranties that the decentralized control can achieve the optimal behavior of the centralized supervisor. Since their work is closely related to our method, we briefly review it in the remaining part of this section.

The model of the global system G is defined by $G = (Q, \Sigma, \delta, q_0, Q_m)$, where $Q = Q_1 \times Q_2 \times ... \times Q_n$, $\Sigma = \bigcup_{i=1}^{n} \Sigma_i$ (with $\Sigma_c = \bigcup_{i=1}^{n} \Sigma_{ic}$, $\Sigma_{uc} = \bigcup_{i=1}^{n} \Sigma_{iuc}$), $q_0 = (q_{01}, q_{02}, ..., q_{0n})$ and $\delta : \Sigma \times Q \to Q$ is given by;

$$\delta((q_1,\ q_2, ..., q_n),\ \sigma) = \begin{cases} (q_1',\ q_2', ..., q_n'), \textit{ where } q_i' = \delta(q_i, \sigma) \textit{ for } \forall i | \sigma \in \Sigma_i, \textit{if } \delta(q_i, \sigma) \textit{is defined and} \\ \qquad\qquad\qquad\qquad q_i' = \delta(q_i, \sigma) \textit{ for } \forall i | \sigma \notin \Sigma_i; \\ \textit{undefined} \qquad\qquad \textit{otherwise.} \end{cases}$$

We denote $G = \|_{i=1}^{n} G_i$ for the entire (global) system. Thus, in G an event that belongs to exactly one subsystem can occur asynchronously and independently. If an event belongs to several subsystems, it must occur simultaneously in all of them, in order to occur in the composite system. It follows that if $G = \|_{i=1}^{n} G_i$, then $L(G) = \|_{i=1}^{n} L(G_i)$. In particular, if Σ_i are all disjoint, then G is the shuffle product of G_i (see section IV.2.4).

Recall from section 4.2.2 that a global supervisor S achieves optimal (i.e. less restrictive) behavior of G under the controlled specification C by synthesizing the language $K = L(S/G) = L(C \times G) = L(C) \cap L(G) \subset \Sigma^*$. In the case of concurrent systems, where each subsystem G_i is controlled by its local supervisor S_i, the concurrent operation

of all controlled subsystems S_i / G_i generates a new global system G_{gl}, namely $\tilde{K} = L(G_{gl}) = \|_{i=1}^{n} L(S_i / G_i) = \cap_{i=1}^{n} P_i^{-1}(L(S_i / G_i))$.

The following theorem (Theorem 4.1 in [27]) gives the conditions under which the concurrent control scheme achieves the optimal global behavior.

*Theorem*1: Let a global DES $G = \|_{i=1}^{n} G_i$, where $G_i = (Q_i, \Sigma_i, \delta_i, q_{0i}, Q_{mi})$, $1 \le i \le n$. There exist local supervisors S_i, which observe and control only the events of Σ_i of each G_i such that $\tilde{K} = K$ if and only if K is separable w.r.t. $\{\Sigma_i\}_{i=1}^{n}$.

The proof of Theorem1 is given in Willner and Heymann (1991). The authors introduce an algorithm of polynomial complexity for checking the separability of a language K when the subsets $\Sigma_i, i = 1,2,...,n$ are pairwise disjoint. Since our method is similar to this algorithm, here we introduce it in brief.

Let $A = (Q, \Sigma, \delta, q_0, Q)$ be a deterministic automaton with m states that accepts a language K and Σ_i are pairwise disjoint subset of event set Σ.

*Algorithm*1 (Algorithm 4.1 in [36]):

(1) For each $i = 1,2,...,n$ construct the automaton $A_i = (Q, \Sigma_i, \delta_i, q_{0i}, Q)$ as defined in step 2.

(2) For each pair (i, q), where $i = 1,2,...,n$ and $q \in Q$, define $A_{iq} = (Q, \Sigma_i, \delta_i, q_{0i}, \{q\})$ and define $d_i(q) = \{\sigma \in \Sigma_i | \delta_i(q, \sigma) = \varnothing\}$.

(2a) Construct the product automaton $A_{iq} \times A_i = (Q \times Q, \Sigma_i, \delta_i, q_{0i} \times q_{0i}, \{q\} \times Q)$. Define $Q_{iq} \subset Q$ to be the set of all states $q' \in Q$ such that (q, q') is an accessible state in $A_{iq} \times A_i$, i.e. $Q_{iq} = \{q' \in Q \mid (\exists t \in \Sigma_i^*, \exists q_{10}, q_{20} \in Q_{i0})(q, q') \in \delta((q_{10}, q_{20}), t)\}$.

(2b) If there exists $q' \in Q_{iq}$ such that $d_i(q) \not\subset d_i(q')$, then stop.

(3) If all the pairs (i, q) were checked, then stop. Else repeat step (2) for another pair (i, q).

The concept of *Algorithm*1 is as follows. For each pair (i, q), $L(A_{iq})$ is the set of all strings $P_i(t)$, such that $t \in K$ and $\delta(q_0, t) = q$, and $d_i(q)$ is the set of all $\sigma \in \Sigma_i$ such that $t\sigma \notin K$. By constructing $A_{iq} \times A_i$ the algorithm identifies the set Q_{iq}, which is the set of all states $q' \in Q$ such that there exists $s, t \in K$, which satisfies and $\delta(q_0, t) = q$, and $\delta(q_0, s) = q'$. If $d_i(q) \not\subset d_i(q')$ then there exists $\sigma \in \Sigma$ such that $t\sigma \notin K$ and $s\sigma \in K$, which contradicts separability.

The complexity of *Algorithm*1 is $O(m \cdot n^3)$.

6.2. Decentralized supervisor of separate groups of vehicles – passengers

To avoid the discussed increases of the state space and number of transitions in section 5.2.2.3, a decentralized approach of supervisory control can be applied for a DRT system split in separate subsystems (groups) of vehicles and passengers. Let all the *n* passengers and *m* vehicles are split in disjoint groups, such that each group of passengers is to be served only by their designated group of vehicles. Fig.6.1 depicts the case when groups of two passengers are to be served by two vehicles.

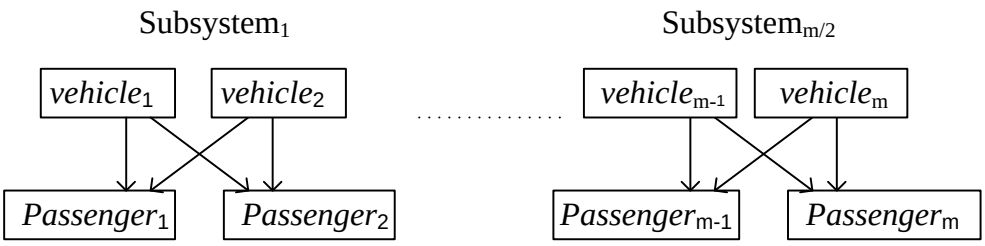


Fig. 6.1 DES split in subsystems of vehicles and passengers.

In this case the local supervisors of each subsystem are easily computed similarly to computation of MS_2 in section 5.2.2.2. Since the event sets of the groups are disjoint, the decentralized supervisor S_{dec} of the global DES will be the union of all the local supervisors of each group:

$$S_{dec} = S_1 \wedge S_2 \wedge \ldots \wedge S_{m/2}.$$

There are two main advantages of such a decentralization: very limited state space and number of transitions for each local supervisor, and all the local supervisors can be computed in parallel. However, with the decentralized architecture of separate groups if the vehicles are designated only to one group of passengers, some of them will not be utilized with full capacity, e.g. can have assigned one passenger (or generally less than their seating capacity) and thus, the global supervisor S_{dec} is not optimal. As Leduc at al. (2005) discusses, this is the price we have to pay for the advantages that the approach offers.

To avoid the possibility of underutilization of the vehicles and thus using the smallest possible fleet, we develop a DSC of dynamic subsystems of vehicles and passengers. Every vehicle with its assigned passenger(s) is a subsystem of the global DES

and is controlled with its local supervisor. With every new request, all the local supervisors check if their vehicles can serve the new passenger. A new vehicle is to be involved only if none of the active vehicles can serve received request. In the next section we demonstrate the method with an example for control of DRT system for emergency evacuation.

6.3. Illustrative Example of ARE Service in D-DARP MADO Environment

In the present section we develop a DSC model capable for real time nonblocking control of a DRT system offering emergency service in ARE environment. The system operates under D-DARPMADO conditions over a region of natural or man-made disaster providing emergency evacuation of passengers from their origins to specific destinations (MTFs) as defined by Sadeh and Kott (1996). The DRT system is modeled as a global DES system consisting of a set of concurrent subsystems. Each subsystem is a DES modeling the behavior of a vehicle (e.g. a helicopter or a VLJ) and its assigned passenger(s). The local supervisors (LSs) of the particular subsystems are capable in real time decision making of accepting passenger requests and routing or rerouting the vehicles. If there is no interaction between the vehicles and passengers, the event sets of all the subsystems are disjoint and the global supervisor (GS) of the entire system is constructed as a conjunction of all LS_j, i.e. $GS = LS_1 \wedge LS_2 \wedge ... \wedge LS_M$, where M is the number of the vehicles, Fig. 6.2.

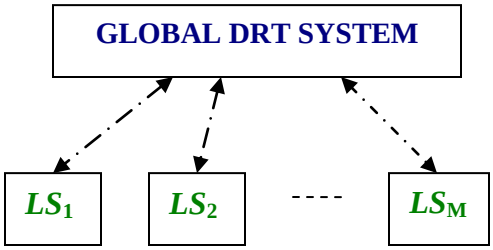


Fig.6.2 Structure of the global system and local control.

Similar to the model of Section 5.3.2 this model controls the service with minimum possible fleet size, and satisfies the same constraint on the length of vehicle service during a working shift. The emergency evacuation DRT service in ARE environment differs from the air charter service in the following characteristics:

- Vehicles do not have specific depots to be kept, i.e. they may stand by at any MTF and do not have to conclude their service at a given depot;
- Some of the MTFs can be closed during service and the vehicles with passengers whose destinations are closed should be redirected to other ones;
- Some of the passengers may have more than one possible destination, i.e. they may be transported to either one of two different MTFs.

The common features of both problems are high dynamics of operations that requires immediate decision about the feasibility of a request and real time update of jets' routings and schedules; limited carriage capacities with small number of seats or beds; limited length of flights, the available vehicles are not subject to change or breakdown during service.

The most challenging question of D-DARPMADO problem is the set of the possible origins of the service requests - they may belong to a large but finite set of locations or could be any point in the covered region. In the next sections we solve the problem with a finite set of origins. In Chapter Seven we discuss the challenges and possible ways to solve the problem with infinite many origins of requests.

6.3.1. Problem description of ARE Service in D-DARP MADO environment

Consider a DRT system which covers the demand for emergency evacuation of people over a region R with a fleet of four jets $j = \{1, 2, 3, 4\}$, (Fig.6.3). There are five origin locations in R $(O_1,...,O_5)$ where passengers can release service requests and can be picked up, and three MTF destinations (F_1, F_2, F_3).

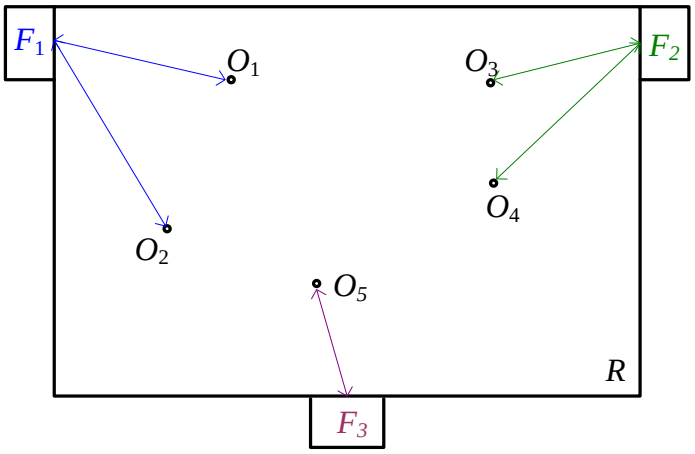


Fig. 6.3 Region R with 5 origins and 3 destinations.

Let the fleet consists of vehicles (VLJs and/or helicopters) with limited seating capacities – vehicles1, 2 and 4 can accommodate two passengers, and $vehicle_3$ – three passengers.

The system receives randomly initiated passenger requests for transportation from one of the origins to some of the destinations. Because of the limits of the software used for verification of the model (XPTCT-software) we will review the modeling of only the first nine requests, i.e. $i = \{1, \ldots, 9\}$.

In this model we keep the same definitions of a *flight of a jet* and a *trip of a jet* as in the model in section 5.3.2. In addition, the same constraints of at most two intermediate stops during a trip and up to one trip through a working shift per vehicle are to be satisfied. Thus, a working shift (i.e. a trip) includes up to three flights. However, if the destination facility of some of the passengers on board a given vehicle is closed, then the vehicle is assumed to have a traveling resource to make an emergency flight to another MTF i.e. destination.

The main difference of the two models is in the way their SCs are implemented. In the model of section 5.3.2 the centralized and modular SC were computed. With this model we demonstrate the synthesis of distributed SC of concurrent systems. Each of the vehicles and its assigned passengers form a subsystem which performs concurrently with the other subsystems of the rest of the fleet and their passengers. Since there is no interaction between the vehicles and the passengers assigned to different vehicles, the DSC of the subsystems is conjunctive.

To utilize a minimum number of vehicles, with every new released request the procedure checks if any of the activated vehicles can be assigned to that passenger. If

adding the origin and destination locations in the route of a vehicle with enough seating capacity does not violate the constraints, the *LS* of that subsystem provides the updated language of desired behavior of these vehicle and passenger. If the control procedure finds the updated language of the LS_j of the subsystem of some $vehicle_j$ to be feasible, i.e. $LS_j \neq \varnothing$, that vehicle can accommodate the request and the passenger is assigned to $vehicle_j$. There is no need for the procedure to check for the rest of LS_j.

Let at the beginning of the working shift $vehicle_1$ and $vehicle_2$ are positioned at F_1, $vehicle_3$ is at F_2 and $vehicle_4$ is at F_3. Let the DRT system follows some simple rules for the initial activating of the vehicles: if a request is released from either O_1 or O_2 and there is no active vehicle, $vehicle_1$ is activated; if the request comes from either O_3 or O_4, $vehicle_3$ is activated, and if the request is released from O_5, $vehicle_4$ is activated.

At the beginning of the working shift the system receives a request from $passenger_1$, who needs to be transported from O_1 to F_1. Since there are no active vehicles, $vehicle_1$ is activated. The control procedure needs to compute the possible behavior of $vehicle_1$, so that $passenger_1$ will be picked from its location (O_1) and transported to the desired destination (F_1).

6.3.2. DES modeling of a small emergency DRT system in D-DARP MADO environment

The set of all the events Σ of the considered system is summarized in Table 4. Any vehicle can be in active state or waiting at a MTF. The assignment, pickup drop off and emergency drop off events have two indexes – *i* represents the number of the

passenger and *j* - the number of the vehicle serving that passenger. Facilities can be open or closed. Each flight is labeled as a combination of a digit followed by two letters. The digit represents the number of the vehicle and the letters – the origin and the destination correspondingly. There are two uncontrollable events – when a vehicle lands at a facility and (atf_j) and when a facility is closed ($closed_k$).

Table 4 The set Σ of all the events of a small emergency DRT system.

Process	Events – c: controllable; u: uncontrollable		
Vehicle status	act_j	Vehicle *j* is in service	c
	atf_j	Vehicle *j* is landed at a MTF	u
Passenger's demand service	$pasgn_{ij}$	Passenger *i* assigned to vehicle *j*	c
	$pick_{ij}$	Passenger *i* picked by vehicle *j*	c
	$drop_{ij}$	Passenger *i* dropped by vehicle *j*	c
	$edrop_{ij}$	Passenger *i* dropped in emergency by vehicle *j*	c
Facility status	$open_k$	MTF *k* is open for passenger acceptance	c
	$closed_k$	MTF *k* is closed for passengers	u
Flights	jF_1O_1	Vehicle *j* flies from F_1 to O_1	c
	jF_1O_2	------ // ---- from F_1 to O_2	c
	jF_2O_3	------ // ---- from F_2 to O_3	c
	jF_2O_4	------ // ---- from F_2 to O_4	c
	jF_3O_5	------ // ---- from F_3 to O_5	c

Table 4 (Continued)

	jO_1O_2	------ // ---- from O_1 to O_2	c
	jO_1O_3	------ // ---- from O_1 to O_3	c
	jO_1O_4	------ // ---- from O_1 to O_4	c
	jO_1O_5	------ // ---- from O_1 to O_5	c
	jO_1F_1	------ // ---- from O_1 to F_1	c
	jO_1F_2	------ // ---- from O_1 to F_2	c
	jO_1F_3	------ // ---- from O_1 to F_3	c
	jO_2O_1	------ // ---- from O_2 to O_1	c
	jO_2O_3	------ // ---- from O_2 to O_3	c
	jO_2O_4	------ // ---- from O_2 to O_4	c
	jO_2O_5	------ // ---- from O_2 to O_5	c
	jO_2F_1	------ // ---- from O_2 to F_1	c
	jO_2F_2	------ // ---- from O_2 to F_2	c
	jO_2F_3	------ // ---- from O_2 to F_3	c
	jO_3O_1	------ // ---- from O_3 to O_1	c
	jO_3O_2	------ // ---- from O_3 to O_2	c
	jO_3O_4	------ // ---- from O_3 to O_4	c
	jO_3O_5	------ // ---- from O_3 to O_5	c
	jO_3F_1	------ // ---- from O_3 to F_1	c
	jO_3F_2	------ // ---- from O_3 to F_2	c
	jO_3F_3	------ // ---- from O_3 to F_3	c

Table 4 (Continued)

	jO_4O_1	------ // ---- from O_4 to O_1	c
	jO_4O_2	------ // ---- from O_4 to O_2	c
	jO_4O_3	------ // ---- from O_4 to O_3	c
	jO_4O_5	------ // ---- from O_4 to O_5	c
	jO_4F_1	------ // ---- from O_4 to F_1	c
	jO_4F_2	------ // ---- from O_4 to F_2	c
	jO_4F_3	------ // ---- from O_4 to F_3	c
	jO_5O_1	------ // ---- from O_5 to O_1	c
	jO_5O_2	------ // ---- from O_5 to O_2	c
	jO_5O_3	------ // ---- from O_5 to O_3	c
	jO_5O_4	------ // ---- from O_5 to O_4	c
	jO_5F_1	------ // ---- from O_5 to F_1	c
	jO_5F_2	------ // ---- from O_5 to F_2	c
	jO_5F_3	------ // ---- from O_5 to F_3	c
Emergency flights	jeF_1F_2	Emergency flight of vehicle j from F_1 to F_2	c
	jeF_1F_3	Emergency flight of vehicle j from F_1 to F_3	c
	jeF_2F_1	Emergency flight of vehicle j from F_2 to F_1	c
	jeF_2F_3	Emergency flight of vehicle j from F_2 to F_3	c
	jeF_3F_1	Emergency flight of vehicle j from F_3 to F_1	c
	jeF_3F_2	Emergency flight of vehicle j from F_3 to F_2	c

6.3.2.1. Computation of the supervisor of one vehicle - one passenger (LS_{11})

Formalization of the plant model: having only one passenger at O_1, we use $vehicle_1$ by default. The plant consists of the following three automata:

- $pasn_1$ (Fig. 6.4) represents the possible behavior of $passenger_1$ - it releases its service request (state 0), $passenger_1$ is assigned to $vehicle_1$ (state 1), $passenger_1$ is picked by $vehicle_1$ (state 2) and dropped off (state 3).

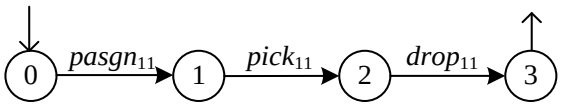


Fig.6.4 Automaton $pasn_1$.

- $pveh_1$ (Fig. 6.5) presents the possible behavior of $vehicle_1$.

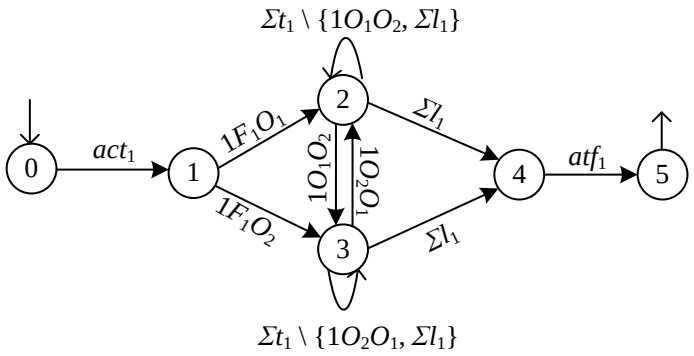


Fig. 6.5 Automaton $pveh_1$.

In this chapter, Σt_1 denotes all the flight events of $vehicle_1$ – i.e. $\Sigma t_1 = \{1F_1O_1, 1F_1O_2, \ldots, 1O_5F_3\}$, and Σl_1 denotes all the flights of $vehicle_1$ ending at any

MTF, i.e. $\Sigma l_1 = \{1O_1F_1, 1O_1F_2, \ldots, 1O_5F_3\}$. After *vehicle*$_1$ is activated from its stand by position at F_1 (state 1), it can fly either to O_1 or O_2 (states 2 and 3, respectively). Any new flight except those ending at the MTFs keeps the vehicle in these states. When a flight from Σl_1 is executed, the system is in state 4, and if *vehicle*$_1$ lands at a MTF, the system is in state 5.

- *fd*$_1$ (Fig. 6.6) coordinates the flights with which *vehicle*$_1$ ends its trips. For example, the vehicle may lend at any MTF through O_1, e.g. $\{1O_1F_1, 1O_1F_2, \ldots, 1O_5F_3\}$ if it has visited O_1 with the previous flight, e.g. one of these flights has been executed: $\{1F_1O_1, 1O_2O_1, 1O_3O_1, 1O_4O_1, 1O_5O_1\}$.

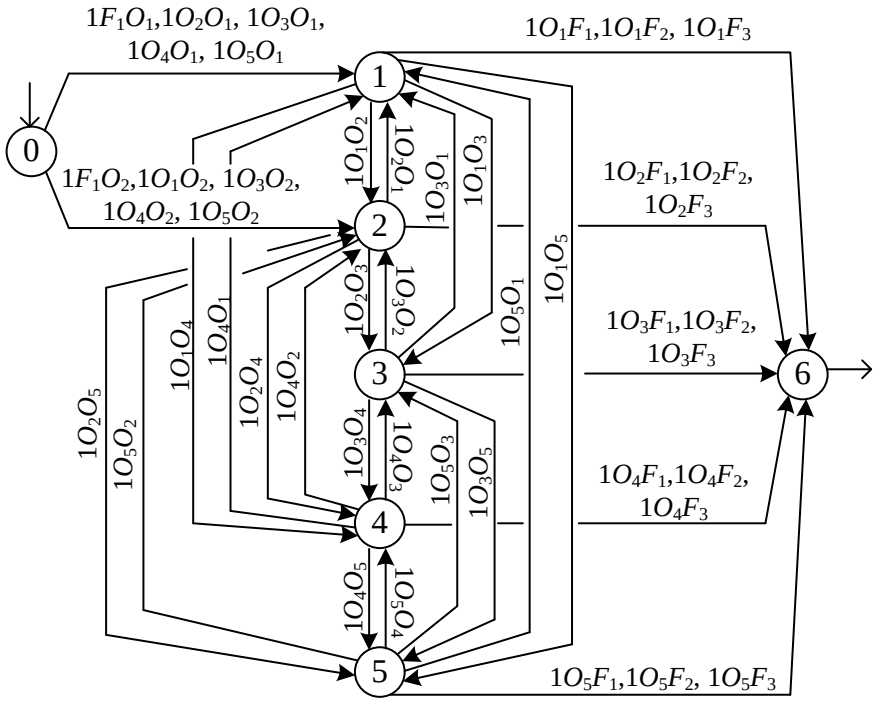


Fig.6.6 Automaton *fd*$_1$.

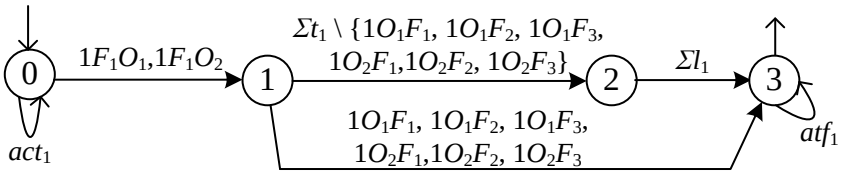
From the initial state (0), with every possible flight of $vehicle_1$ that goes to O_1 the system gets to state 1, and every flight of $vehicle_1$ that goes to O_2 takes the system to state 2. Similarly, every flight between all the five origin locations takes the system to the corresponding state – e.g. state 5 represents that $vehicle_1$ is in O_5. From any state 1 to 5 $vehicle_1$ can fly to any MTF, thus bringing the system to the marked state 6.

Automaton fd_1 becomes necessary in modeling the emergency DRT system, because in automaton $pveh_1$, which describes the possible behavior of $vehicle_1$, all the flights among the origin locations are modeled with two states – 2 and 3 and all the flights to the MTFs take the system to one state - 4. This simplicity in representation of the possible flights does not consider where exactly the vehicle is, like in the small air charter model (section V.2.2), and reduces the state space. However, the price for it is the necessary additional automaton to secure that after all the flights the vehicle gets to the final MTF with the correct sequence of flights.

Thus, the plant automaton of the model is the parallel composition of three automata, i.e. $Plant_{11} = pasn_1 \| pveh_1 \| fd_1$. It is comprised of 56 states and 338 transitions.

Formalization of the specifications: the following three automata specify the desired behavior of $vehicle_1$ and $passenger_1$:

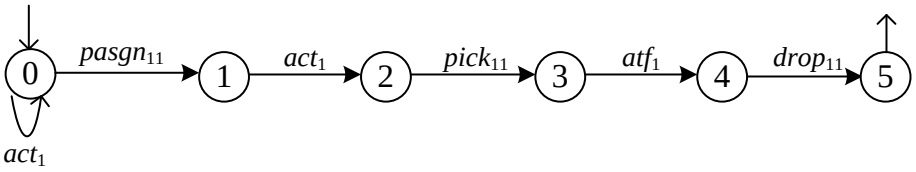
- Similarly to the small air charter model, we use an automaton $trips_1$ (Fig. 6.7) to ensure that $vehicle_1$ makes up to three flights per trip.



$$Selfloop = \{pasgn_{11}, pick_{11}, drop_{11}\}$$

Fig. 6.7 Automaton $trips_1$.

- $vehdil_{11}$ (Fig. 6.8) ensures the diligent service of $vehicle_1$ – i.e. $passenger_1$ can be assigned to $vehicle_1$ if the vehicle is activated or not (state 0), and if it is not, it must be activated, (state 2), next $passenger_1$ must be picked up (state 3), after $vehicle_1$ gets at the facility (state 4), $passenger_1$ can be dropped off (state 5).



$$Selfloop = \Sigma t_1$$

Fig. 6.8 Automaton $vehdil_{11}$.

- $paspd_1$ (Fig. 6.9) specifies that $vehicle_1$ can pick up $passenger_1$ right after a flight to O_1 (state 1), and $passenger_1$ can be dropped off when the vehicle gets to F_1.

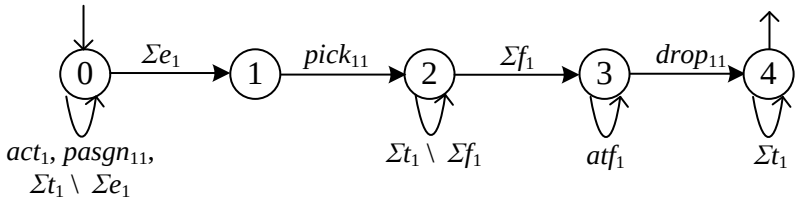


Fig. 6.9 Automaton $paspd_{11}$.

With Σe_1 we denote all the flights of $vehicle_1$ which go to O_1, i.e. $\Sigma e_1 = \{1F_1O_1, 1O_2O_1, 1O_3O_1, 1O_4O_1, 1O_5O_1\}$, and Σf_1 denotes all the flights of $vehicle_1$ which end up in F_1, i.e. $\Sigma f_1 = \{1O_1F_1, 1O_2F_1, 1O_3F_1, 1O_4F_1, 1O_5F_1\}$.

The cross product of $trips_1$, $vehdil_1$ and $paspd_1$ generates the specification automaton, i.e. $Spec_{11} = trips_1 \times vehdil_{11} \times paspd_{11}$, which consists of 24 states and 132 transitions.

Synthesis of the supervisor of $vehicle_1$ - $passenger_1$ (LS_{11}): the intersection of the languages marked by $Plant_{11}$ and $Spec_{11}$ automata produces LS_{11} (Fig. 6.10) i.e. $LS_{11} = Plant_{11} \cap Spec_{11}$. Again, we use the XPTCT-software to compute all the languages of the automata in the three steps.

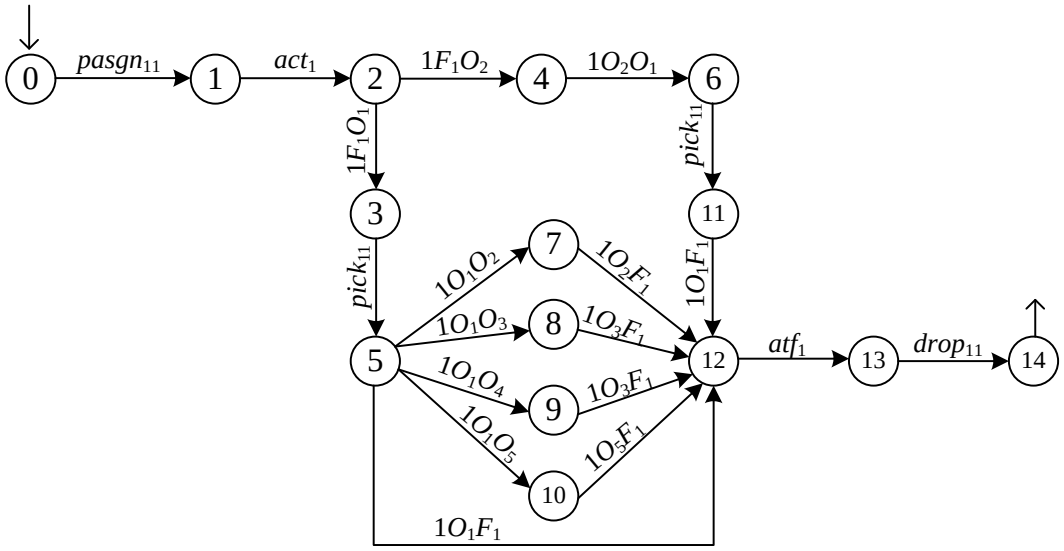


Fig. 6.10 Automaton of LS_{11}.

The synthesized LS_{11} is controllable with 15 states and 19 transitions. Starting from the initial state (0), $passenger_1$ has to be assigned to $vehicle_1$ (state 1), after $vehicle_1$ is activated (state 2), there are two possible routes – through O_1 (states 3-5-7, 8, 9, 10-12) and through O_2 (states 4-6-11-12). In either way, $passenger_1$ is picked up when the vehicle is at O_1 (states 3 or 11) and when $vehicle_1$ gets to F_1 (state 12) it is at facility (state 13). At the facility $passenger_1$ can be dropped off - (state 14), marked state.

6.3.2.2. Computation of the supervisor of one vehicle - two passengers (LS_{12})

Let at a given time instant $passenger_1$ is assigned to $vehicle_1$, $vehicle_1$ is activated and is taking off from F_1 when another service request arrives - $passenger_2$ has to be transferred from O_3 to F_2. The operational planning procedure is to check if the active

vehicle$_1$ is capable to meet the second request. If it is, then *passenger*$_2$ has to be assigned to the same vehicle, and if not, a new vehicle is to be activated.

Formalization of the plant model: with two passengers the plant consists of the following three automata:

- *pasn*$_{12}$ (Fig.6.11) describes the updated behavior of *passenger*$_1$ – being already assigned it has to be picked and dropped off; and *pasn*$_2$ (Fig.6.12) describes the possible behavior of *passenger*$_2$.

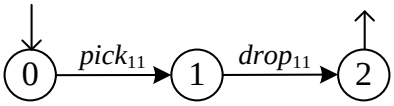


Fig.6.11 Automaton *pasn*$_{12}$.

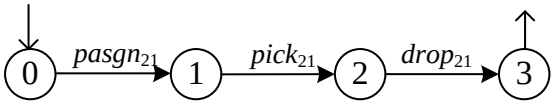


Fig.6.12 Automaton *pasn*$_2$.

- *pveh*$_{12}$ (Fig.6.13) describes the updated possible behavior of *vehicle*$_1$ – the new initial state is at F_1.

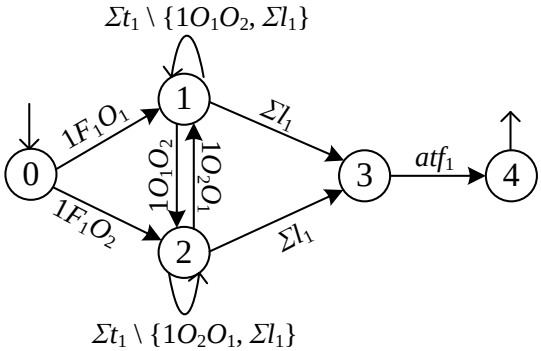


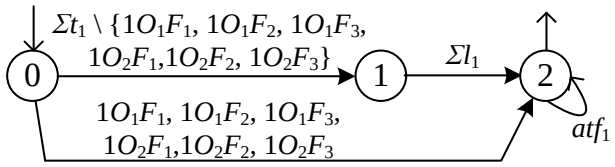
Fig. 6.13 Automaton *pveh*$_{12}$.

- there is no change in automaton fd_1 (Fig. 6.6) that coordinates the flights with which $vehicle_1$ ends its trips.

Thus, the plant is computed with the parallel composition of the four automata, i.e. $Plant_{12} = pasn_{12} \| pasn_2 \| pveh_{12} \| fd_1$.

Formalization of the specifications: the following five automata specify the desired behavior of $vehicle_1$ and both passengers:

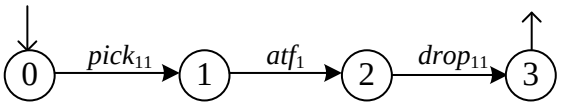
- $trips_{12}$ (Fig. 6.14) is the updated automaton of $trips_1$ and ensures that $vehicle_1$ has no more than two flights remaining.



$$Selfloop = \{pick_{11}, drop_{11}, pasgn_{21}, pick_{21}, drop_{21}\}$$

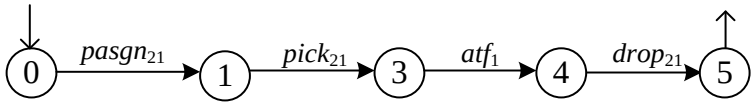
Fig. 6.14 Automaton $trips_{12}$.

- two automata $vehdil_{i1}, i = (1,2)$ (Fig. 6.15) secure diligent service for both passengers by $vehicle_1$ - $vehdil_{11}$ (Fig. 6.15a) is the updated automaton of $vehdil_1$ and encounters the remaining events that must be executed for service of $passenger_1$, and $vehdil_{21}$ (Fig. 6.15b) is the corresponding automaton to ensure diligent service for $passenger_2$ by the same vehicle.



$$Selfloop = \{pasgn_{21}, pick_{21}, drop_{21}\} \cup \Sigma t_1$$

a) Automaton $vehdil_{11}$.

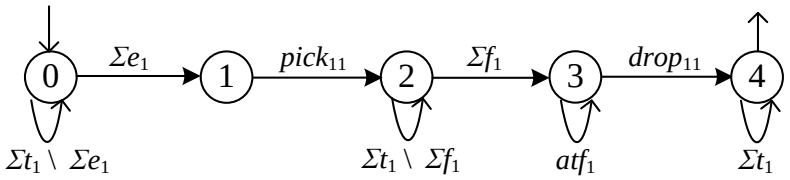


$$Selfloop = \{pick_{11}, drop_{11}\} \cup \Sigma t_1$$

b) Automaton $vehdil_{21}$.

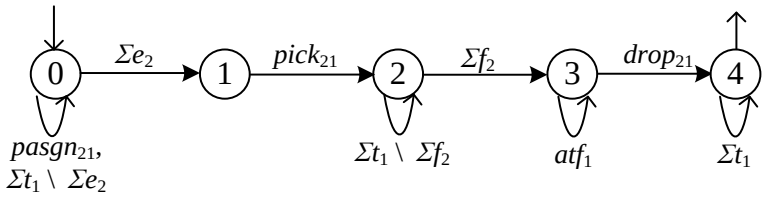
Fig. 6.15 Automata $vehdil_{i1}, i = (1,2)$.

- two automata $paspd_{i1}, i = (1,2)$ (Fig. 6.16) are needed to specify when each passenger can be picked and dropped off by $vehicle_1$: $paspd_{11}$ (Fig. 6.16a) covers the picking and dropping of $passenger_1$ – since it is assigned but not picked yet and $vehicle_1$ is activated, the change compared with $paspd_1$ from (Fig. 6.9) is the elimination of events act_1 and $pasgn_{21}$, and a selfloop that considers the necessary events for the service of the other passenger; $paspd_{21}$ (Fig. 6.16b) is the corresponding automaton for $passenger_2$.



$$Selfloop = \{pasgn_{21}, pick_{21}, drop_{21}\}$$

a) Automaton $paspd_{11}$.



$$Selfloop = \{pick_{11}, drop_{11}\}$$

b) Automaton $paspd_{21}$.

Fig. 6.16 Automata $paspd_{i1}, i = (1,2)$.

With Σe_2 we denote all the flights of $vehicle_1$ that end at O_3, i.e. $\Sigma e_2 = \{1O_1O_3, \ldots, 1O_5O_3\}$, and Σf_2 denotes all the flights of $vehicle_1$ that end at F_2, i.e. $\Sigma f_2 = \{1O_1F_2, \ldots, 1O_5F_2\}$.

Thus, the specification of the service of the two passengers with one vehicle is computed with the cross product of $trips_{12}$, $vehdil_{11}$, $vehdil_{21}$, $paspd_{11}$, and $paspd_{21}$ - i.e. $Spec_{12} = trips_{12} \times vehdil_{11} \times vehdil_{21} \times paspd_{11} \times paspd_{21}$.

Synthesis of the supervisor of $vehicle_1$ - $passenger_1$ and $passenger_2$ (LS_{12}): the intersection of the languages marked by $Plant_{12}$ and $Spec_{12}$ automata produces LS_{12} i.e. $LS_{12} = Plant_{12} \cap Spec_{12}$.

However, LS_{12} is empty (i.e. it has zero states and zero events) because it is infeasible for a vehicle to visit two different origin locations (O_1 and O_3) and two different destinations (F_1 and F_2) in one trip. Therefore, we need another vehicle to be involved – by default it will be $vehicle_3$, currently located at MTF_2.

6.3.2.3. Computation of the local supervisor of one vehicle - one passenger (LS_{32})

As $vehicle_3$ is getting involved, a need arises for another supervisor – LS_{32}. Since at this moment $passenger_2$ will be the only passengers of $vehicle_3$, LS_{32} will be analogous to LS_{11} – one vehicle – one passenger. Here we briefly describe the synthesis of LS_{23} to demonstrate the difference in the notations and indexes.

The synthesis of the plant is the parallel composition of the following three automata:

- $pasn_{23}$ (Fig. 6.17) - represents the possible behavior of $passenger_2$

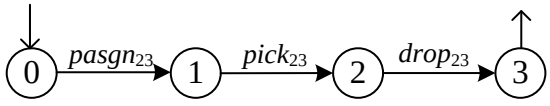


Fig. 6.17 Automaton $pasn_{23}$.

- $pveh_3$ (Fig. 6.18) – describes the possible behavior of $vehicle_3$

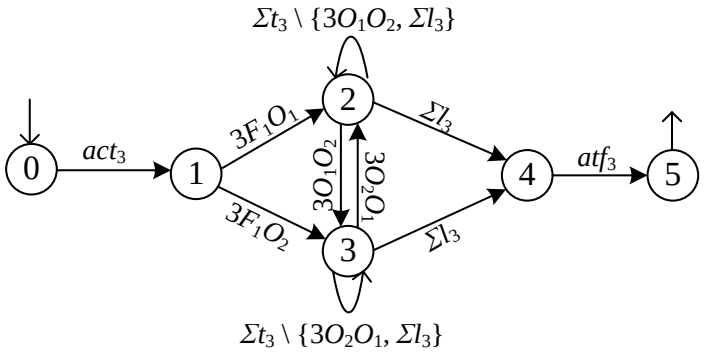


Fig. 6.18 Automaton $pveh_3$.

Similarly to LS_{11}, here Σt_3 denotes all the flight events of $vehicle_3$, i.e. $\Sigma t_3 = \{3F_3O_5, 3O_1O_2, ..., 3O_5F_3\}$, and Σl_3 denotes all the flights of $vehicle_3$ ending at any MTF, i.e. $\Sigma l_3 = \{3O_1F_1, 3O_1F_2, \ldots, 3O_5F_3\}$.

- fd_3 (Fig. 6.19) – coordinates the flights with which $vehicle_3$ has to end its trips;

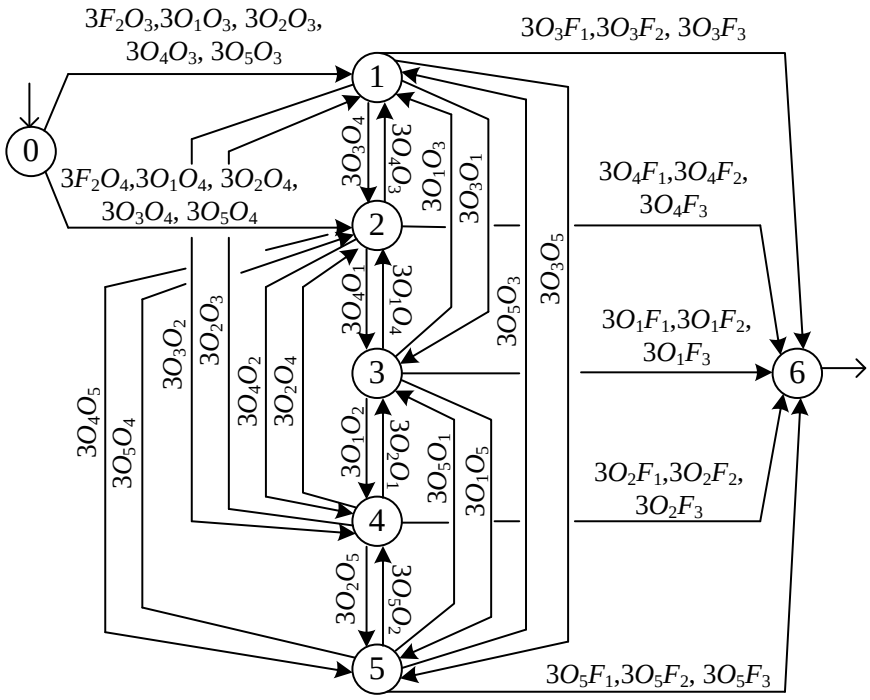
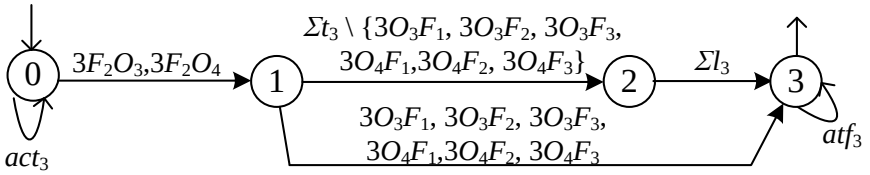


Fig. 6.19 Automaton fd_3.

Thus, $Plant_{32} = pasn_{23} \| pveh_3 \| fd_3$.

The synthesis of the specifications is the cross product of the following three automata:

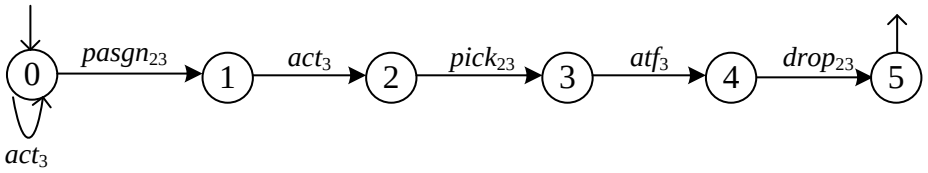
- $trips_3$ (Fig. 6.20) limits the number of the flights in the trip of $vehicle_3$



$$Selfloop = \{pasgn_{23}, pick_{23}, drop_{23}\}$$

Fig. 6.20 Automaton $trips_3$.

- *vehdil*$_{23}$ (Fig.6.21) ensures diligent service for *passenger*$_2$ by *vehicle*$_3$



$$Selfloop = \Sigma t_3$$

Fig. 6.21 Automaton *vehdil*$_{23}$.

- *paspd*$_{23}$ (Fig.6.22) specifies after which flights *passenger*$_2$ can be picked and dropped off by *vehicle*$_3$

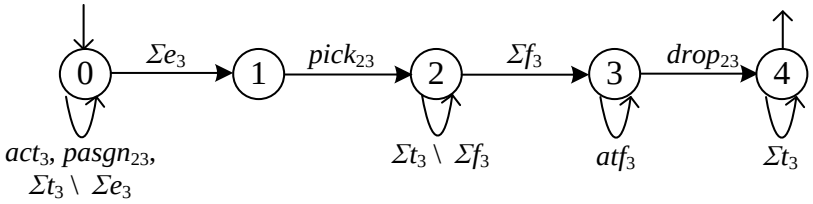


Fig. 6.22 Automaton *paspd*$_{23}$.

Here Σe_3 denotes all the flight events of *vehicle*$_3$ going to O_3, i.e. $\Sigma e_3 = \{3F_2O_3, \ldots, 3O_5O_3\}$, and Σf_3 denotes all the flight events of *vehicle*$_3$ going to F_2, i.e. $\Sigma f_3 = \{3O_1F_2, \ldots, 3O_5F_2\}$.

Hence, the automaton of specifications of the service of *passenger*$_2$ by *vehicle*$_3$, $Spec_{32}$ is computed: $Spec_{32} = trips_3 \times vehdil_{23} \times paspd_{23}$ The local supervisor LS_{32} (Fig. 6.23) of *vehicle*$_3$ serving *passenger*$_2$ can be synthesized: $LS_{32} = Plant_{32} \cap Spec_{32}$.

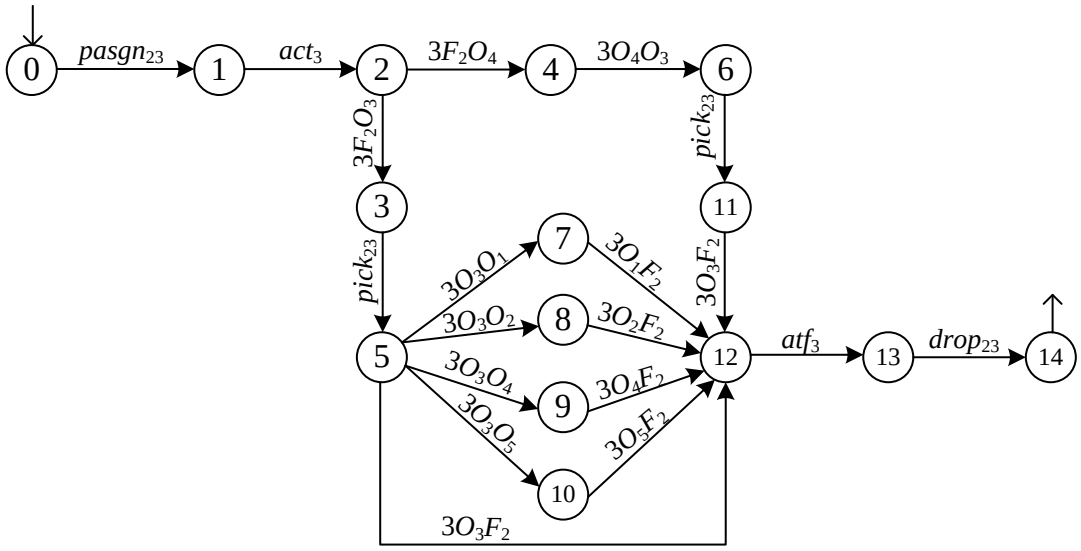


Fig. 6.23 Supervisor LS_{32}.

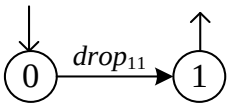
6.3.2.4. Computation of the local supervisor of one vehicle - two passengers (LS_{113})

Let at the current time instant $vehicle_1$ has picked $passenger_1$, $passenger_2$ is assigned to $vehicle_3$, which is activated, took off from its initial location F_2 and another service request is received: $passenger_3$ has to be transferred from O_2 to F_1 or F_2. Now the control procedure checks from all the active vehicles if any of them is capable to meet this request. In the remaining part of this section we will demonstrate that $vehicle_1$ can transfer $passenger_3$ without violation of its current routing and scheduling.

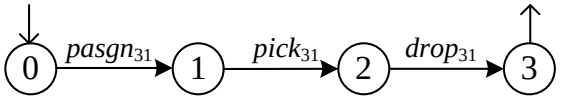
Considering the current state of $vehicle_1$ and both passengers, the plant of the subsystem, $Plant_{113}$ is composed with the following four automata:

- a pair of automata $pasn_i, (i = 1,3)$ (Fig. 6.24) model the updated behavior of both passengers - $pasn_1$ (Fig. 6.24a) is updated, generating the only remaining event of

service of $passenger_1$ and $pasn_3$ (Fig. 6.24b) models the necessary behavior of $passenger_3$.



a) Automaton $pasn_1$.



b) Automaton $pasn_3$.

Fig. 6.24 Automata $pasn_i$.

- $pveh_{13}$ (Fig. 6.25) describes the updated possible behavior of $vehicle_1$

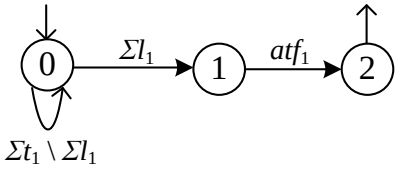


Fig. 6.25 Automaton $pveh_{13}$.

- fd_{13} (Fig. 6.26) synchronizes the all flight events with the necessary end flights to the possible destinations

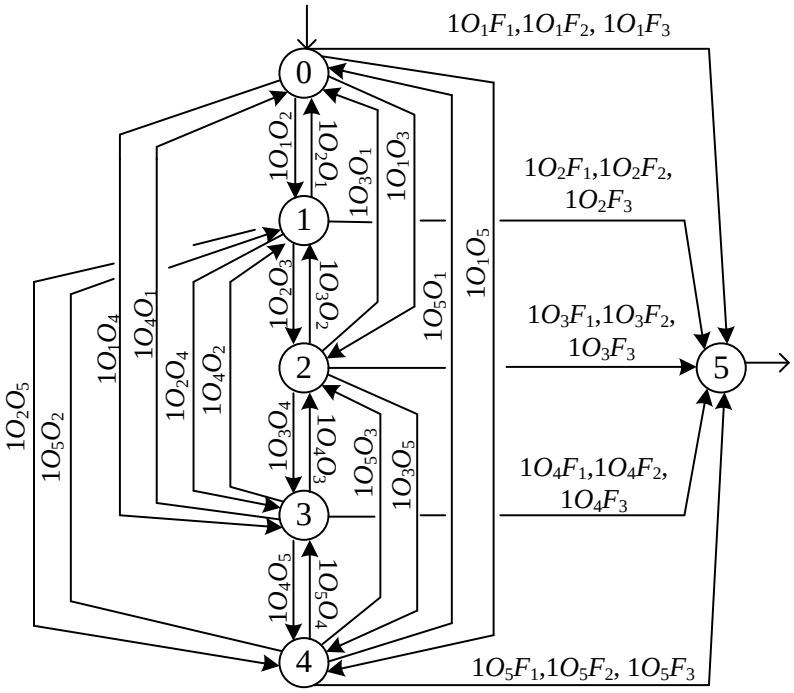
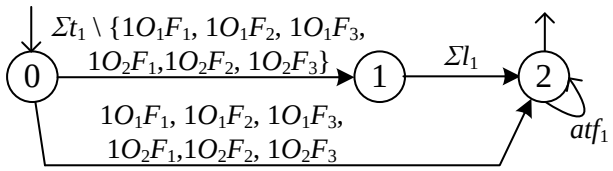


Fig. 6.26 Automaton fd_{13}.

Hence, $Plant_{113} = pasn_1 \| pasn_3 \| pveh_3 \| fd_{13}$. It has 56 states and 438 transitions.

The specification automaton of the subsystem $vehicle_1$ and $passenger_1$ and $passenger_3$, $Spec_{113}$ is computed with the product of the following five automata:

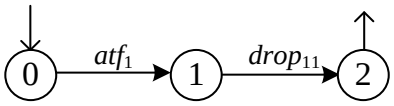
- $trips_{13}$ (Fig. 6.27) specifies that up to two flights remain in the trip of $vehicle_1$.



$$Selfloop = \{drop_{11}, pasgn_{31}, pick_{31}, drop_{31}\}$$

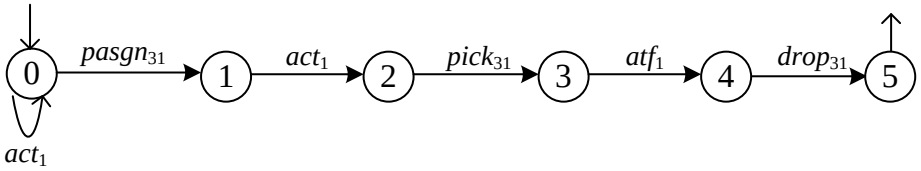
Fig. 6.27 Automaton $trips_{13}$.

- two automata $vehdil_{1i}, (i = 1,3)$ (Fig. 6.28) ensure diligent service of the vehicle for both passengers - $vehdil_{11}$ (Fig.6.28a) is the updated automaton $vehdil_1$ that covers service for $passenger_1$ and $vehdil_{13}$ (Fig.6.28b) is the corresponding automaton for $passenger_3$.



$$Selfloop = \{pasgn_{31}, pick_{31}, drop_{31}\} \cup \Sigma t_1$$

a) Automaton $vehdil_{11}$

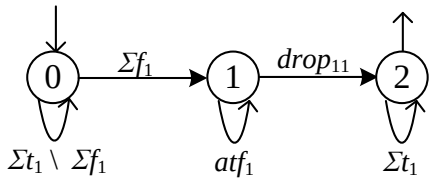


$$Selfloop = \{drop_{11}\} \cup \Sigma t_1$$

b) Automaton $vehdil_{13}$

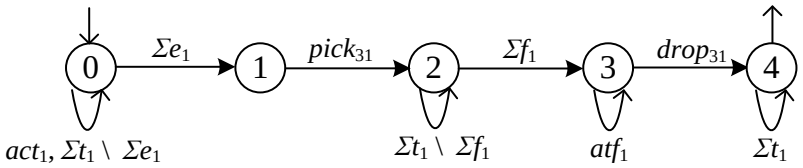
Fig.6.28 Automata $vehdil_{1i}, (i = 1,3)$

- two analogous automata $paspd_{1i}, (i = 1,3)$ (Fig. 6.29) specify when both passengers can be picked and dropped off - since $passenger_1$ is already picked, $paspd_{11}$ (Fig. 6.29a) covers only its dropping off , while $paspd_{13}$ (Fig. 6. 29b) ensures both picking and dropping of $passenger_3$.



$$Selfloop = \{pasgn_{31}, pick_{31}, drop_{31}\}$$

a) Automaton $paspd_{11}$.



$$Selfloop = \{drop_{11}\}$$

b) Automaton $paspd_{13}$.

Fig. 6.29 Automata $paspd_{1i}, (i = 1,3)$.

Thus, $Spec_{113} = trips_{13} \times vehdil_{11} \times vehdil_{13} \times paspd_{11} \times paspd_{13}$. It contains 20 states and 123 transitions. The local supervisor of the subsystem, LS_{113} (Fig. 6.30) is then computed, i.e. $LS_{113} = Plant_{113} \cap Spec_{113}$.

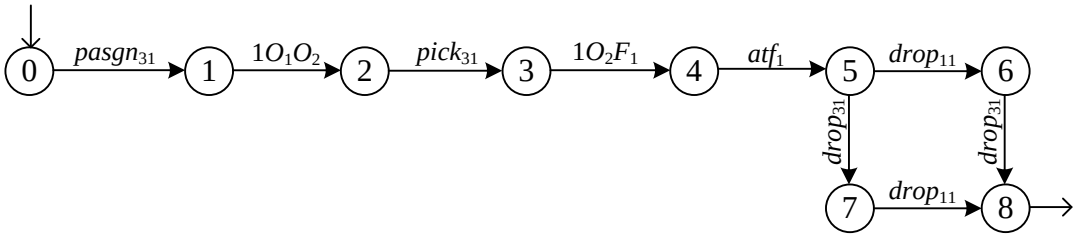


Fig. 6.30 Automaton LS_{113}.

One can note that LS_{113} is a part of LS_{11} (Fig.6.10) with added new states and events for picking and drop off *passenger*3. Starting from state 5 of LS_{11}, which corresponds to state 0 in Fig. 6.30, LS_{113} assigns $passenger_3$ to $vehicle_1$ (state 1 of Fig. 6.30), then travels to O_2 (states 7 and 2 of LS_{11} and LS_{113}, correspondingly), picks $passenger_3$ (state 3 of LS_{113}), travels to F_1 (states 11 and 4 of LS_{11} and LS_{113}), arrives at a facility (states 13 and 5 of LS_{11} and LS_{113}), where the passengers are dropped off (states14 of LS_{11}, and 6-8 of LS_{113}).

With the so far developed cases of distributed SC of operation of emergency DRT system in sections 6.3.2.1 through 6.3.2.4 we modeled subsystems and obtained the local supervisors of one vehicle – one passenger (LS_{11} and LS_{32}), one vehicle – two passengers with infeasible operation (LS_{12}), and one vehicle – two passengers with feasible service (LS_{113}). In these four cases all the resources (i.e. vehicles and MTFs) were available during the entire operation. However, as it was discussed in Sections 3.1 and 6.1, one of the most critical features of the emergency DRT service in ARE environment is that some of the resources may become suddenly unavailable during service. In the next section we

demonstrate the system control in case when one of the MTFs is closed and cannot accept any vehicles to land.

6.3.2.5. *Computation of the local supervisor of one vehicle - two passengers in case of a closed MTF* (LS_{448})

Consider the following possible development of our system – $passenger_4$ and $passenger_8$ have released service requests for transportation from O_5 to F_1 or F_3, and from O_1 to F_3 respectively. Both have been assigned to $vehicle_4$, which have been routed from its initial location F_3 to visit O_5, picked $passenger_4$ from O_5 traveled to O_1 and right after picking up $passenger_8$ the system receives a signal that the desired destination of $vehicle_4$ - F_3 is closed.

In modeling such a scenario we utilize the emergency flight events, which have not been used so far.

The plant of the subsystem, $Plant_{448}$ is composed as the parallel composition of the following five automata:

- A pair of automata $pasn_i, (i = 4,8)$ (Fig. 6.31) model the possible behaviors of both passengers - $pasn_4$ (Fig. 6.31a) describes the behavior of $passenger_4$ and $pasn_8$ (Fig. 6.31b) – of $passenger_8$, respectively.

a) Automaton $pasn_4$. b) Automaton $pasn_8$.

Fig. 6.31 Automata $pasn_i, (i = 4,8)$.

- *pveh*$_4$ (Fig. 6.32) describes the uncontrolled behavior of *vehicle*$_4$

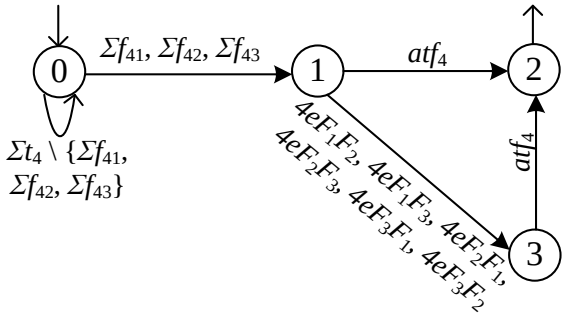


Fig. 6.32 Automaton *pveh*$_4$.

Similarly to the notations in automata *pveh*$_1$ and *pveh*$_3$, Σt_4 denotes all the flight events of *vehicle*$_4$, i.e. $\Sigma t_4 = \{4F_3O_5, 4O_1O_2, \ldots, 4O_5F_3\}$. In addition, Σf_{41} denotes all the flights of *vehicle*$_4$ that end at F_1 without the emergency flights, i.e. $\Sigma f_{41} = \{4O_1F_1, \ldots, 4O_5F_1\}$, Σf_{42} denotes all the flights of *vehicle*$_4$ that end at F_2 without the emergency flights, i.e. $\Sigma f_{42} = \{4O_1F_2, \ldots, 4O_5F_2\}$, and Σf_{43} denotes all the flights of *vehicle*$_4$ that end at F_3 without the emergency flights, i.e. $\Sigma f_{43} = \{4O_1F_3, \ldots, 4O_5F_3\}$.

- *fd*$_4$ (Fig.6.33) ensures that all flight events of *vehicle*$_4$ are bound with the necessary terminal flights to the three possible destinations. The emergency flight events keep the system at the marked state 5.

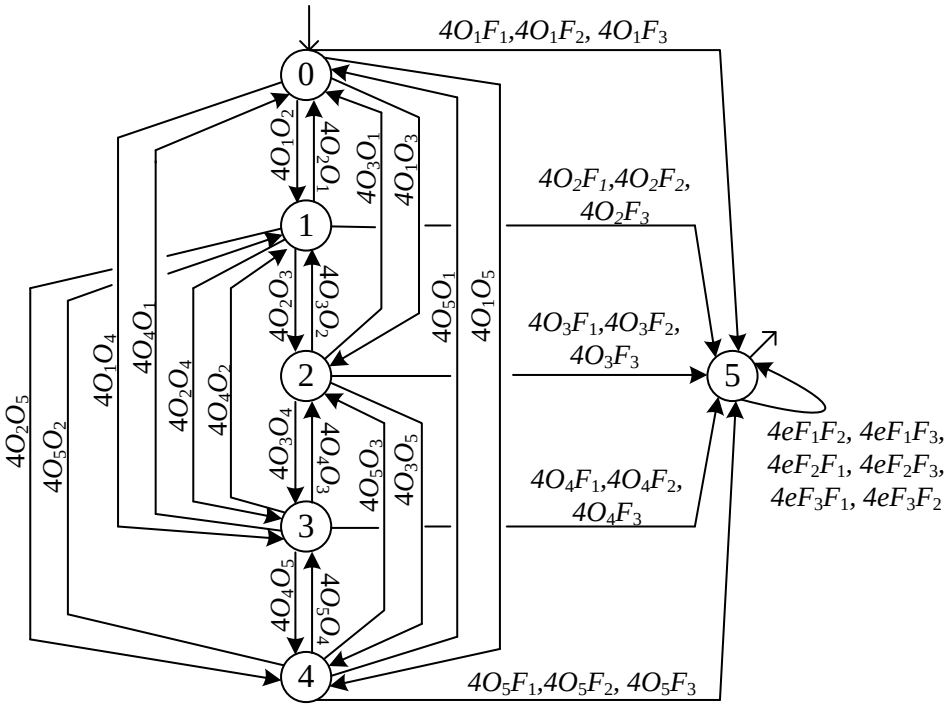


Fig.6.33 Automaton fd_4.

- $fstat_3$ (Fig. 6.34) specifies that F_3 is closed

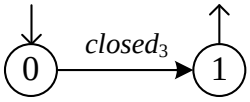
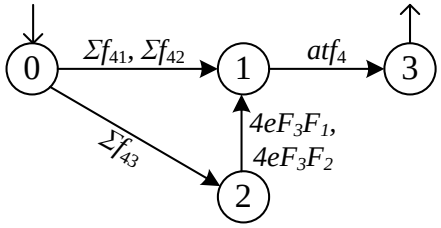


Fig.6.34 Automaton $fstat_3$.

Therefore, $Plant_{448} = pasn_4 \| pasn_8 \| pveh_4 \| fd_4 \| fstat_3$. It has 64 states and 544 transitions.

The specification automaton $Spec_{448}$ is synthesized with the cross product of the following six automata:

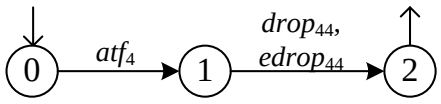
- *trips*$_{48}$ (Fig.6.35) limits the number of the allowed flights in the trip - being at O_1 (state 0) *vehicle*$_4$ has one more flight to end the trip - if it is from the sets Σf_{41} and Σf_{42}, the system gets to state 1, where it is considered that the vehicle is at facility (state 3), if the flight is from set Σf_{43}, the system gets to state 2, where some emergency flight to F_1 or F_2 must be executed.



$$Selfloop = \{closed_3, drop_{44}, edrop_{44}, edrop_{84}\}$$

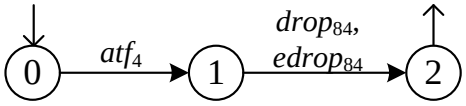
Fig. 6.35 Automaton *trips*$_{48}$

- two analogous automata $vehdil_{4i}, (i = 4,8)$ (Fig. 6.36) ensure the diligent service of *vehicle*$_4$ for both passengers. Automaton *vehdil*$_{44}$ (Fig. 6.36a) models the service for *passenger*$_4$ and *vehdil*$_{48}$ (Fig. 6.36b) - for and *passenger*$_8$, respectively – being already picked (state 0), the vehicle has to get to a MTF (state 1) in order to do drop off or emergency drop off the passengers (state 2).



$$Selfloop = \{closed_3, drop_{84}, edrop_{84}\} \cup \Sigma t_4$$

a) Automaton *vehdil*$_{44}$.

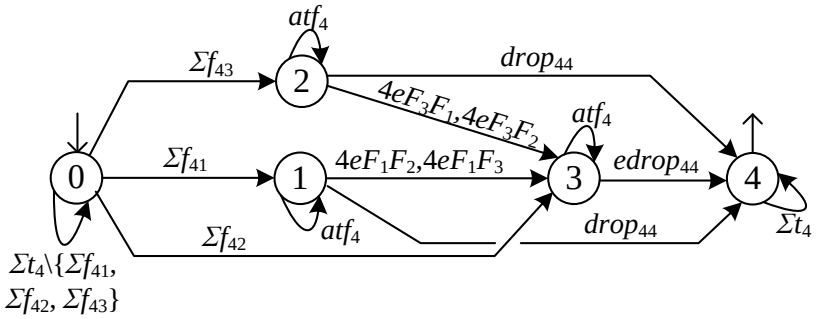


$$Selfloop = \{closed_3, drop_{44}, edrop_{44}\} \cup \Sigma t_4$$

b) Automaton $vehdil_{48}$.

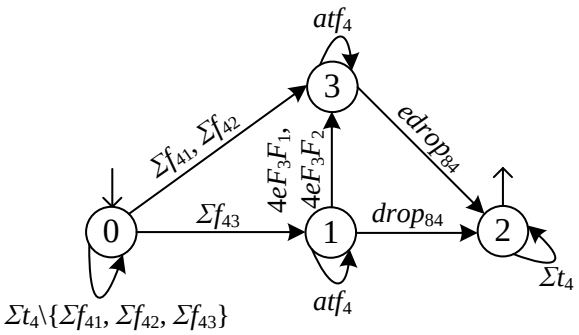
Fig. 6.36 Automata $vehdil_{4i}, (i = 4,8)$.

- a pair of automata $paspd_{4i}, (i = 4,8)$ (Fig. 6.37) specifies when the passengers can be dropped off. Automaton $paspd_{44}$ (Fig. 6.37a) covers the dropping of $passenger_4$ and $paspd_{48}$ (Fig. 6.37b) – of $passenger_8$, respectively. Both passengers can be dropped off either at their regular or emergency destinations.



$$Selfloop = \{closed_3, drop_{84}, edrop_{84}\}$$

a) Automaton $paspd_{44}$.

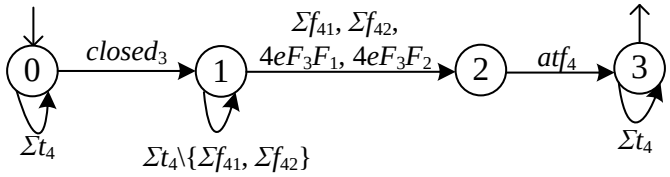


$$Selfloop = \{closed_3, drop_{44}, edrop_{44}\}$$

b) Automaton $paspd_{48}$.

Fig. 6.37 Automata $paspd_{4i}, (i = 4,8)$.

- $fstat_3$ (Fig. 6.38) specifies when $vehicle_4$ gets at an open MTF providing F_3 is closed – at the initial state 0 the system receives a signal (event $closed_3$) and gets in state 1, next only the flight events from sets Σf_{41} and Σf_{42}, and the emergency flights $4eF_3F_1$ and $4eF_3F_2$ can take the system to state 2, where the vehicle is considered at a MTF and gets to the marked state 3.



$$Selfloop = \{drop_{44}, edrop_{44}, drop_{84}, edrop_{84}\}$$

Fig. 6.38 Automaton $fstat_3$.

Therefore, $Spec_{448} = trips_{48} \times vehdil_{44} \times vehdil_{48} \times paspd_{44} \times paspd_{48} \times fstat_3$. It includes 16 states and 43 transitions. The local supervisor of the subsystem *vehicle*$_4$ - *passenger*$_4$ and *passenger*$_8$, LS_{448} (Fig. 6.39) is computed: $LS_{448} = Plant_{448} \cap Spec_{448}$. At the initial state 0 *vehicle*$_4$ is at O_1 and is routed to F_3. The two branches going out of state 0 are determined from the exact receiving of the event *closed*$_3$ - if it comes before the take off, the system gets to state 1, and if *vehicle*$_4$ takes off first, the system gets to state 2. Then the signal for closed F_3 comes during the flight to F_3 (state 5).

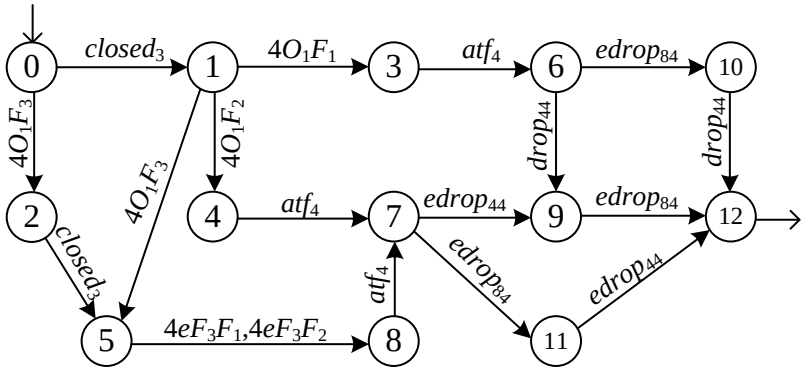


Fig. 6.39 Automaton of LS_{448}

In the first branch the vehicle may fly to all MTFs, but if it goes to F_1 or F_2 (states 3 and 4, respectively) it is considered at an open MTF, and can drop both passengers (*passenger*$_8$ is always dropped off in emergency). If *vehicle*$_4$ flies to F_3 (state 5), it joins the second branch and has to make one more emergency flight to F_1 or F_2 (state 8) before it gets to a MTF and consecutively drops off the passengers.

6.3.2.6. Generating the global SC of the emergency DRT

With the models for LSs of the received passenger requests considered in sections 6.3.2.1 through 6.3.2.5 we covered the basic cases of passenger's assignments to vehicles and vehicle routings of the emergency DRT problem described in 6.3.1. Because of the limitations of the applied XPTCT-software in terms of the number of states and events, we were able to verify the modeling of 9 passenger requests served with 5 vehicles. Table 5 shows all the requests, their assignments to the vehicles and the controlling *LS*s.

Table 5 – Considered requests, assigned vehicles and *LS*s.

Request	Passenger#	Origin - destination Facility	Assigned vehicle	*LS*
1	$passenger_1$	O_1 - F_1	$vehicle_1$	LS_{113}
2	$passenger_2$	$O_3 – F_2$	$vehicle_3$	LS_{3259}
3	$passenger_3$	O_2 - F_1 or F_2	$vehicle_1$	LS_{113}
4	$passenger_4$	O_5 - F_1 or F_3	$vehicle_4$	LS_{448}
5	$passenger_5$	$O_4 – F_2$	$vehicle_3$	LS_{3259}
6	$passenger_6$	O_5 - F_1 or F_2	$vehicle_2$	LS_{267}
7	$passenger_7$	$O_2 – F_2$	$vehicle_2$	LS_{267}
8	$passenger_8$	$O_1 – F_3$	$vehicle_4$	LS_{448}
9	$passenger_9$	$O_4 – F_2$ or F_3	$vehicle_3$	LS_{3259}

Although dynamically formed, every *LS* controls a group of a vehicle with its assigned passengers, which does not interact with the other groups. Thus, there are no

shared events (i.e. transitions) between the groups except of a closing or opening a MTF (e.g. $open_k$, $closed_k$), if F_k is a common destination. Since there are no limits in the number of vehicles to land at any F_k, $open_k$ and $closed_k$ do not cause any interaction or dependency between the corresponding *LS*s. Therefore, the general SC of the global DRT system, SC_{gen} can be computed as the union of all the *LS*s, i.e.

$$SG_{gen} = LS_{113} \wedge LS_{3259} \wedge LS_{448} \wedge LS_{267} .$$

6.3.2.7. Computational complexity of decentralized supervisor

Recall from Section 5.2.2.3 that in the worst case the computational complexity of modular supervisory control is $O(max(p_1, p_2)r)$. In decentralized synthesis if the service of a given passenger with the vehicle is developed as an independent module, the computational complexity of the corresponding *LS* has the same upper limit as in the modular control. The main advantage of decentralization is that if the subsystems are disjunctive, all the *LS*s can be computed in parallel and the nonblocking property of the SC_{gen} is still guaranteed.

Chapter Seven

Contribution of the Study and Future Research

7.1. Summary of the completed work and contribution of the study

In this study DRT systems are modeled as DES using Finite Automate formalism, and DRT operational planning and real time control are addressed using discrete event supervisory control theory. DES modeling and supervisory control theory are well established and powerful mathematical tools. In this dissertation, they are shown to be suitable for expressing the modeling and control requirements associated with the complex and dynamic applications in DRT. The modeling and control approaches described herein, coupled with the mature body of research literature in discrete event systems and supervisory control theory, facilitates logical analysis of these complex systems and provides the necessary framework for the development of real time scheduling and intelligent decision making tools for operational planning in a broad range of DRT applications. To this extent, this work includes several significant contributions to the field of DRT systems modeling and operational control.

To establish a systematic approach to the study of DRT systems, a taxonomy of the identifying features of DRT application domains is presented. This taxonomy is based on origin/destination characteristics, fleet characteristics, and demand characteristics.

Within this taxonomy, several characteristics associated with DRT systems such as capacity constraints, route lengths etc. are modeled using Finite Automata. The representation of systems specifications and characteristics associated with DRT are straight-forward to express in spoken languages, however correct mathematical representation of these features is not without challenge. Two application scenarios are considered; the first is based on air-taxi service operation and illustrates uncontrolled system model and operational specification synthesis. Based on the uncontrolled system model and the specifications models, the automatic synthesis of centralized and modular supervisors are demonstrated. The second scenario is a mission critical application based on the emergency aero-medical evacuation problem. In this scenario, decentralized supervisory control architecture suitable for accommodating the real-time contingencies associated with this application is presented. The conditions for parallel computation of local supervisors are specified and the computational advantages of alternative supervisory control architectures are discussed.

The alternative control architectures utilized in this work exhibit varying degrees of suitability to different application domains within DRT systems. Centralized control schemes suffer from exponentially increasing computational complexity and are only suitable for small sized static systems (as illustrated with the air-taxi service application). Decentralized control schemes provide a robust control solution to highly dynamic applications, such as the emergency evacuation. Furthermore, it is shown that, following the appropriate design procedures, the decentralized architectures still manage to maintain desirable supervisory control characteristics such as nonblocking and are computationally tractable for a subset of the DRT application domains.

7.2. Future Research

The research should continue with modeling and control of *many to many* type of DRT system where the origin and destination locations of the service requests can be anywhere over the covered region. The main challenge is to control the allowed length of travel of the vehicles. There are two possible approaches to cope with this problem:

7.2.1. Application of timed DES (TDES)

In this approach the length of travel will be controlled with the limits of time it can take. In TDES both logical behavior and timing information are considered in system's evolution. In modeling TDES first a FA called *activity transition graph* denoted with G_{atg} is introduced to describe the untimed behavior of the system. $G_{act} = (A, \Sigma_{act}, \delta_{act}, a_0, A_m)$, where A is the finite set of activities, Σ_{act} is the finite set of events, $\delta_{act} : A \times \Sigma_{act} \to A$ is the partial activity transition function, a_0 is the initial activity and $A_m \subseteq A$ is the set of marked activities.

Timing information is introduced into G_{act} with the following way: each event $\sigma \in \Sigma_{act}$ is given a *lower time bound* $l_\sigma \in N$ and *upper time bound* $u_\sigma \in N \cup \infty$, such that $l_\sigma \le u_\sigma$ and N denote the nonnegative integers. The set of events Σ_{act} is decomposed into two subsets: $\Sigma_{spe} = \{\sigma \in \Sigma_{act} | u_\sigma \in N\}$ and $\Sigma_{rem} = \{\sigma \in \Sigma_{act} | u_\sigma = \infty\}$, where Σ_{spe} is the set of *prospective* and Σ_{rem} is the set of *remote* events. The lower time

bound typically represents a delay in control, while the upper time bound is a hard deadline. For each $\sigma \in \Sigma_{act}$ the time interval T_σ is defined as follows:

$$T_\sigma = \begin{cases} [0, u_\sigma] \; if \; \sigma \in \Sigma_{spe}, \\ [0, l_\sigma] \; if \; \sigma \in \Sigma_{rem}. \end{cases}$$

TDES is defined as a FA $G = (Q, \Sigma, \delta, q_0, Q_m)$, where the state set Q is defined as $Q = A \times \prod \{T_\sigma | \sigma \in \Sigma_{act}\}$. A state $q \in Q$ is of the form $q = (a, \{t_\sigma | \sigma \in \Sigma_{act}\})$, where activity $a \in A$ and timer $t_\sigma \in T_\sigma$. Timer t_σ encounters the passage of global time for each σ. The set $Q_{\mathrm{m}} \subseteq Q$ is given as by a subset of $A_m \times \prod \{T_\sigma | \sigma \in \Sigma_{act}\}$. The event set Σ is defined as $\Sigma = \Sigma_{act} \cup \{tick\}$, where event *tick* represents the passage of one time unit. The state transition function δ is defined as follows - for each $\sigma \in \Sigma$ and $q = (a, \{t_\tau | \tau \in \Sigma_{act}\}) \in Q$, $\delta(q, \sigma)$ is defined, i.e. $\delta(q, \sigma)!$, if and only if one of the following conditions hold:

- $\sigma = tick$ and $(\forall \tau \in \Sigma_{spe}) \; \delta_{act}(a, \tau)! \Rightarrow \mathrm{t}_\tau > 0$,
- $\sigma = \Sigma_{\mathrm{spe}}$ and $\delta_{act}(a, \sigma)!$ and $0 \le t_\sigma \le u_\sigma - l_\sigma$,
- $\sigma = \Sigma_{\mathrm{rem}}$ and $\delta_{act}(a, \sigma)!$ and $t_\sigma = 0$.

In DRT system, every flight and travel of a vehicle will have its lower and upper bounds, i.e. the limits of beginning and end of service. However, the main disadvantage of TDES approach is the very large state space, caused by tracking all the states at any tick of time. To improve the efficiency of the model, Saadatpoor and Wonham (2007) propose instead of language control, state-based predicates in compressed form represented with binary decision diagrams (BDDs). In this approach, the structure in the

states in the form of event timers of the modeled TDES can help reduce the size of BDDs.

7.2.2. Application of hybrid DES (HDES)

In HDES modeling, some of the state variables are discrete and some are continuous. The dynamic behavior of discrete state systems is usually simpler to represent, but the mathematical tools to formally express and solve the state equations may be more complex. In contrast, continuous state models ultimately reduce to the analysis of differential equations, for which many mathematical techniques are available. The type of supervisory control problems that is of interest in HDES arises whenever a continuous system is to be controlled by a discrete process such as a digital computer program. The continuous process to be controlled, together with any continuous controllers, is identified as the *Plant* and is typically described by differential/difference equations. The *Controller* includes a discrete decision process that is typically a represented by FA. The *Interface* makes it possible for these different processes to communicate with each other. This control framework is quite flexible and can describe modern engineering systems where a computer process is used to control and coordinate several physical processes over a computer network. It can also describe a switching control system where a continuous plant is controlled by different continuous controllers over a number of operating regions. The discrete event controllers for hybrid systems are based on discrete abstractions of the continuous dynamics. Applications have been primarily in the continuous process industry and transportation service. The advantage of this approach is that it generalizes well-known concepts from digital control design. One

of the main characteristics of the SC approach has been the emphasis and explicit identification of the interface issues between the continuous and discrete dynamics. These interface issues are the cornerstone of any HDES study. Koutsoukos at al. (2000) present a detailed framework for hybrid systems modeling and synthesis of SC for continuous *Plant* and discrete *Controller* (supervisor). The developed *Interface* consists of a generator and an actuator. The generator converts the continuous time output (states) of the *Plant* to an asynchronous, symbolic input for the supervisor. The actuator sends the appropriate control signal into the *Plant*.

In HDES modeling of DRT service different continuous controllers can provide supervision of the vehicles' location and travel, and discrete event controllers can supervise passengers' requests. The main issue will be in the complexity of the interface to coordinate the behaviors of all the system elements.

References

1. Casey R., Porter C., Buffkin T. and Hussey L. (2000) Evaluation Plan for the Cape Cod Advanced Public Transportation System, US DoT, http://www.itsdocs.fhwa.dot.gov/jpodocs/repts_te/

2. Cassandras C. and Lafortune S. (1999) Introduction to Discrete Event Systems, *Kluwer Academic Publishers*.

3. Cieslak R., Desclaux C., Fawaz A. and Varaiya P. (1988) Supervisory control of disrete-event processes with partial observations. *IEEE Transactions on Automatic Control*, 33 (3), 249-260

4. Cordeau J. and Laporte G. (2003) A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research 37B*, 579–594.

5. Cordeau J. and Laporte G. (2007) The dial-a-ride problem: models and algorithms, an updated version of a paper in *4OR* 1:89-101, 2003 http://neumann.hec.ca/chairedistributique/common/DARP.pdf

6. Cordeau J., Laporte G., Potvin J. and Savelsbergh M. (2004) Transportation on Demand. *CRT-2004-25*, http://www2.isye.gatech.edu/~mwps/publications/

7. Cubillos C., Polanco F., and Demartini C. (2004) Multi Agent Infrastructure for Distributed Planning of Demand-Responsive Passenger Transportation Service, *Systems, Man and Cybernetics*, IEEE International Conference, vol. 2: 2013- 2017

8. Dessouky M. and Adam S. (1998) Real-time Scheduling Rules for Demand Responsive Transit Systems. *Systems, Man, and Cybernetics, IEEE International Conference*, vol. 3: 2956-2961

9. Dial R. (1995) Autonomous Dial-A-Ride Transit Introductory Overview, *Transportation Research* part C, 3:261–275

10. Dial R. and Ghani G. (2003) The ADART driver interface. *Proceedings of the 83th Annual Conference of the Transportation Research Board*

11. Diana M. and Dessouky M. (2004) A new regret insertion heuristic for solving large-scale dial-a-ride problems with tome windows, *Transportation Research* 38B, 539-557

12. Finn B. and Breen P. (1996) The use of transport telematics in inter-urban and rural bus services, *Public Transport Electronic Systems* Int'l Conference, 5-10

13. Gillen D. and Raffaillac J. (2002) Assessing the role of AVL in demand responsive transportation systems, *California Partners for Advanced Transit and Highways (PATH). Research Reports*: Paper UCB-ITS-PRR-2002-16. http://repositories.cdlib.org/its/path/reports/UCB-ITS-PRR-2002-16

14. Heymann M. and Lin F. (1994) Online control of partially observed discrete event systems. *Discrete Event Dynamic Systems*, 4, 221-236

15. Horn M. (2002) Fleet Scheduling and Dispatching for Demand Responsive Passenger Services. *Transportation Research*, Part C, vol. 10, No 1, 35-63 http://www.dist-systems.bbn.com/people/krohloff/krohloff_publications.shtml

16. Hunsaker B. and Savelsbergh M. (2002) Efficient testing for dial-a-ride problems. *Operations Research Letters*, 30:169–173

17. Jaw J., Odoni A., Psaraftis H. and Wilson N. (1986) A heuristic algorithm for the multi-vehicle many to many advance request dial-a-ride problem. *Transportation Research 20B,* 243–257

18. Jiang S. and Kumar R. (2000) Decentralized control of discrete event systems with specializations to local control and concurrent systems. *IEEE Transactions on Systems, Man and Cybernetics – part B,* 30:5

19. Kihl M., Crum M., Shinn D. (1996) Linking real time and location in scheduling demand-responsive transit, *Final Report*, Midwest Transportation Center http://www.ctre.iastate.edu/reports/kihlrpt.pdf

20. Koutsoukos X., Antsaklis P., Lemmon M. and Stiver J. (2000) Supervisory Control of Hybrid Systems, *Proceedings of the IEEE Conference*.

21. Lafortune S., Rohloff K. and Yoo T. (2001) Recent Advances on the Control of Partially-Observed Discrete-Event Systems, *Proceedings of, Symposium on the Supervisory Contol of Discrete Event Systems* (SCODES)

22. Lave E., Teal R. and Piras P. (1996) A Handbook for Acquiring Demand Responsive Transit Software. Transit Cooperative Research Program Report #18, *Transportation Research Board*, Washington D.C.

23. Leduc R., Brandin B., Lawford M. and Wonham W. (2005) Hierarchical interface-based supervisory control – part I: serial case. *IEEE Transaction on Automatic Control*, vol. 50, No 9, 1322-1335

24. Leduc R., Lawford M. and Wonham W. (2005) Hierarchical interface-based supervisory control – part II: parallel case. *IEEE Transaction on Automatic Control*, vol. 50, No 9, 1336-1348

25. Lin F. and Wonham W. (1988) Decentralized supervisory control of discrete event systems. *Elsevier Science Publishing Co., Inc*.

26. Queiroz M. and Cury J. (2000) Modular control of composed systems. *Proceedings of the American Control Conference*, Chicago, Il.

27. Rudie K. and Wonham W. (1992) Think globally, act locally: Decentralized supervisory control. *IEEE Transaction on Automatic Control*, vol. 37, 1692-1708

28. Saadatpoor A. and Wonham W. (2007) State based control of timed discrete event systems using binary decision diagrams. *System & Control Letters*, vol. 56, 62-74

29. Sadeh N. and Kott A. (1996**)** Models and Techniques for Dynamic Planning Demand- Responsive Transportation. *CMU-RI-TR-96-04*, Robotics Institute, Carnegie Mellon University

30. Seow, K. T. and Pasquier, M. (2004) Supervising Passenger Land-Transportation Systems. *IEEE Transaction on Intelligent Transportation Systems*, vol. 5, No. 3, 165 – 176

31. Seow, K. T., Pasquier, M. and Hong, M. (1999) A Formal Design Methodology for Land-Transport Operations. *Intelligent Transportation Systems*, Proceedings, IEEE/IEE/JSA International Conference, 110 – 115

32. Sheu J. (2006) A novel dynamic resource allocation model for demand-responsive city logistics distribution operations. *Transportation Research* Part E 42:445-472

33. Takai S. and Ushio T. (2005) Decentralized supervisory control of discrete event systems using dynamic default control. *IEICE Transactions Fundamentals, Vol. E88-A*

34. Toth P., Vigo D. (1997) Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science* 31, 60–71

35. Uchimura K., Takahashi H. and Saitoh T. (2002) Demand Responsive Services in Hierarchical Public Transportation System. *IEEE Transactions on Vehicular Technology*, vol.51, issue 4, 760-766

36. Willner Y. and Heymann M. (1991) Supervisory control of concurrent discrete event systems. *International Journal of Control*, vol. 54, issue 5, 1143-1169

37. Wipke K. (1996) Reducing VMTs through transit-on-demand with GPS and satellite communications, *IEEE Xplore Northcon*, 404-408

38. Wong K. and Bell M. (2006) Solution for the dial-a-ride problem with multi-dimensional capacity constraints, *International Transactions in Operation Research*, 195-208

39. Wonham, W. M. (2006) Supervisory Control of Discrete-Event Systems. *monograph ECE 1336F / 1637S*, http://www.control.utoronto.ca/DES/

40. Wonham, W. M. and Ramadge P.J. (1988) Modular supervisory control of discrete event systems, *Mathematics of Control, Signals, and Systems*, 1, 13-30

41. Yoo T. and Lafortune S. (2002) A general architecture for decentralized supervisory control of discrete-event systems, *Discrete Event Dynamic Systems: Theory and Applications*, 12, 335-377

Bibliography

1. Antsaklis P. and Koutsoukos X. (2005) Hybrid systems: review and recent progress. Chapter in Software-Enabled Control: *Information Technologies for Dynamical Systems*, T. Samad and G. Balas, Eds., IEEE Press.

2. Baillieul J. and Antsaklis P. (2007) Control and communication challenges in networked real-time systems. *Proceedings of the IEEE*, Vol. 95, 1, pp. 9 – 28

3. Benoit G. and Merchand H. (2007) An efficient modular method for the control of concurrent discrete event systems: A language based approach. *Discrete Event Dynamic Systems* 17 pp. 179-209

4. Blouin S., Guay M. and Rudie K. (2000) An application of discrete event theory to truck dispatching, Dept. of Computing and Information Science, Queen's University, Ontario, Canada, *TR#2000-440*

5. Bourdon S. Lawford M. and Wonham W. (2002) Robust nonblocking supervisory control of discrete event systems. *Proceedings of the American Control Conference*, Anchorage, AK

6. Chen P. Guzman J., Ng T., Poo A. and Chan C. (2002) Supervisory control of an unmanned land vehicle, *Proceedings on IEEE International Symposium on Intelligent Control*, Vancouver, Canada, pp. 580 -585

7. Fabian M. and Kumar R. (1997) Mutually nonblocking supervisory control of discrete event systems. *IEEE Conference on Decision and Control*, p 2970--2975, San Diego, CA.

8. Grigorov, L. and Rudie K. (2006) Near optimal online control of dynamic discrete event systems. *Discrete Event Dynamic Systems* 16 pp. 419-449

9. Hruz B. and Zhou M. (2007) Modeling and control of discrete event dynamic systems. *Springer Engineering*, 341 pg.

10. Jonoska N. (2007) Algebraic automata theory. *USF lecture notes*.

11. Lafortune S., Rohloff K. and Yoo T. (2001) Recent advances on the control of partially observed discrete event systems. Chapter in *Synthesis and Control of Discrete Event Systems,* pp. 3-17, Kluwer Academic Publishers.

12. Lee S. and Wong K. (1997) Decentralized control of concurrent discrete event systems with non-prefix closed local specifications. *Proceedings of the 36th Conference on Decision & Control,* San Diego, CA., pp. 2958-2963

13. Lin F. and Wonham W. (1988) On Supervisory Control of Real-Time Discrete-Event Systems. *Information Sciences 44,* pp.159-183

14. Wong K. and Wonham W. (1998) Modular control and coordination of discrete event systems, *Discrete Event Dynamic Systems: Theory and Applications,* 8 pp. 247-297

15. Yoo T. and Lafortune S. (2002) Decentralized supervisory control: A new architecture with a dynamic decision fusion rule, *Proceedings of the Sixth International Workshop on Discrete Event Dynamic Systems* (*WODES '02*)

About the Author

Daniel Yankov received a Bachelor's Degree in Mechanical Engineering from Technical University, Sofia, Bulgaria in 1993 and a M.S. in Industrial Engineering from Rochester Institute of Technology, Rochester, NY in 2004. He has wide engineering experience in project management, tool design and quality control. Daniel entered the Ph.D. program at the University of South Florida in 2004.

While in the Ph.D. program at the University of South Florida, Mr. Yankov was actively involved in a project for simulation and optimization of security check points at major commercial airports. He also made a paper presentation at Informs '07 Annual meeting in Seattle, WA.