Conflict Resolution in Free-Ranging Multi-Vehicle Systems: A Resource Allocation Paradigm

Spyros A. Reveliotis and Elzbieta Roszkowska

***Abstract*—We propose a novel paradigm for conflict resolution in multi-vehicle traffic systems where a number of mobile agents move freely in a finite area, each agent following a pre-specified motion profile. The key idea behind the proposed method is the tesselation of the underlying motion area in a number of cells, and the treatment of these cells as resources that must be acquired by the mobile agents for the execution of their motion profiles, according to an appropriate resource allocation protocol. We capitalize upon the existing literature on the real-time management of sequential resource allocation systems, and develop such protocols that can formally guarantee the safe and live operation of the underlying traffic system, while they remain scalable with respect to the number of the moving agents. Collective past experience with the considered policies indicates that they also provide a pretty large coverage of the RAS behavioral space that characterizes its safe and live operation. Finally, we also establish that the aforementioned approach is applicable even in traffic systems where all vehicles must be in perpetual motion until their retirement.**

I. Introduction

Conflict resolution in multi-vehicle systems is a problem that has received extensive attention in the literature, as it concerns the safe, robust and yet efficient operation of the transport systems employed in different application contexts. As remarked in [1], the problem appears in many different flavors, that result from the operational characteristics of the underlying application contexts, but, in our opinion, they also express the “culture” and the technical strengths and mindsets of the researchers and the technical experts in the relevant areas. A version of the problem that has received particular attention in the last decade concerns the establishment of collision-free routing for autonomous or semi-autonomous (e.g., human-piloted) vehicles in busy areas. Typical examples are provided by air-traffic management systems, especially for traffic close to congested airports, small boat-traffic near to the harbor, and the traffic generated by a large number of mobile robots working on different tasks in a small area. In the prevailing approaches to this problem, each vehicle is abstracted to a *mobile agent*, occupying a certain area and being able to generate a motion that is governed by a specified set of laws. Furthermore, the area occupied by each agent is frequently represented by a *disk* or *sphere* of a certain radius, selected in a way that guarantees a desired *degree of separation* between any pair of agents. The agent motion dynamics are typically modeled in *continuous time*, but they can vary with respect to the assumptions made regarding the *controllability* of the motion. A third attribute that defines another dimension of classification of these models and studies, is the *centralized* or *decentralized* nature of the adopted control scheme, and in the case of the latter, the extent and pattern of the *inter-agent communication* that is necessitated for its implementation. Once the assumptions highlighted above have been detailed, the problem reduces to defining the *sensing capabilities*, *communication protocols* and the *(feedback-based) motion control laws* that will enable each agent to complete its mission trip while avoiding potential collisions with the remaining agents and any other present obstacles. Some indicative examples of this line of research can be found in [1], [2], [3], [4], [5], [6], [7] while a higher-level but more comprehensive description of the pursued methods can be found in [8].

Yet, a closer examination of the results presented in the aforementioned references will reveal that, by focusing on the continuous-time dynamics of the vehicle motion, they tend to suffer from a very high computational complexity, and therefore, their scalability to environments requesting the coordination of a large number of vehicles can become a challenging issue.[1] Furthermore, as remarked in [1], while many of the above works will guarantee *motion safety*, very few of them have actually considered the issue of *motion liveness*, i.e., the ability of each agent to reach successfully its destination in finite time. When it comes to this last problem, most of the past research approaches in this area have addressed it in a haphazard manner, e.g., by reducing the liveness problem to deadlock handling in specific environments, such as narrow corridors [9], or by proposing specific road designs, such as intersection points with buffering areas [10]. In some other cases [11], deadlocks are not prevented, but they are allowed to occur, then they are detected and, if possible, resolved. Finally, among these earliest initiatives, there are also a few works, like those presented in [12], [13], that have tried to take a more systematic approach to the problem, and address it in a more rigorous manner, employing models and concepts borrowed from the supervisory control theory [14].

Motivated by the limited scalability of the aforementioned methods and their apparent incompleteness with respect to liveness, this work will focus on an alternative approach to the

S. A. Reveliotis is with the School of Industrial & Systems Engineering, Georgia Institute of Technology, email: `spyros@isye.gatech.edu`

E. Roszkowska is with the Institute of Computer Engineering, Control and Robotics, Wroclaw University of Technology, email: `ekr@pwr.wroc.pl`

The first author was partially supported by NSF grants CMMI-0619978 and CMMI-0928231.

[1]For instance, the results presented in [6], that constitute one of the most comprehensive and structured approaches to coordinated and optimized motion planning in multi-robot systems, present a computational complexity of, at least, $O(Q^N)$ where N is the number of robots and Q is the number of decision stages per robot that result from a time-based discretization of its motion.

traffic coordination of free-ranging multi-vehicle systems, that seeks to *discretize* the vehicle motion into a number of *stages* and establish the safety and the liveness of the vehicle motion through a *hybrid control* scheme that is defined on the basis of this additional structure. More specifically, under the proposed regime, the motion of a vehicle within a particular stage is controlled by one of the typically used *time-driven* models. On the other hand, the vehicle transitioning among the different stages induces an additional set of *event-driven* dynamics, that evolve in the *discrete state space* which is defined by the potential vehicle allocations to their corresponding stages. In the resulting representation, two stages belonging to the motion processes of two different vehicles are said to be *in conflict*, if their simultaneous execution by the corresponding vehicles can compromise the posed safety requirements. Clearly, conflicting stages should not be allowed to occur simultaneously. But this restriction, when combined with the arbitrary structure of the underlying motion profiles, arises the potential of *deadlock*, and it can compromise the system liveness. Hence, there is a need for a more extensive and systematic coordination of the vehicle transitions among the different stages of their trips.

More specifically, it should be clear from the above discussion that, in the operational regime of the considered hybrid control models, the safety and the liveness of the vehicle motion must be established through a *restriction* of the vehicle transitions among their consecutive stages, that will ensure the following two properties for any reachable state: (i) no two vehicles execute motion stages that are in mutual conflict, and (ii) there exists a feasible sequence of stage transitions that will enable every vehicle to complete its motion. An additional concern is that the restriction imposed upon the vehicle motion is the minimum possible, so that it does not impair unnecessarily other performance attributes of the system. The resulting problem corresponds to a rather classical problem in the theory of concurrent processes, which has already been studied in the context of computer operating systems [15], [16], [17], automated production systems [18], [19], [20], [21] and other workflow management systems [22], [23], and more recently, it has been more generally addressed by the theory of *real-time management of resource allocation systems (RAS)* [24]. The main contribution of this work is *a complete and formal reduction of the considered vehicle safety and liveness problem to the problem of the RAS liveness-enforcing supervision addressed in [24].*

From a more conceptual standpoint, the problem considered in this work generalizes and extends to the domain of free-ranging multi-vehicle systems, techniques that were developed in the past for zone-controlled Automated Guided Vehicle (AGV) systems [25], [26], [27]. AGV systems constitute a class of industrial automated material handling systems where a fleet of mobile robots facilitates the material transfer among a set of processing and buffering stations, while moving on a physically or virtually defined guidepath network that interconnects the considered stations. In order to avoid collisions and ensure the vehicle safety, the links of the guidepath network are split into zones and it is stipulated that every zone can be be allocated to at most one vehicle at a time. Hence, the vehicle trips can be discretized into a sequence of stages that is defined by the sequence of the traversed zones. At the same time, a liveness-enforcing supervisor coordinates the allocation of the different zones to the contesting vehicles so that deadlocks are avoided and every vehicle can proceed successfully to its destination.

Furthermore, the methodology developed in this manuscript is in line with some recent developments that advocate the use of hybrid approaches to vehicle motion planning and control as a means to manage the complexity and enhance the robustness of the underlying control function [28], [29]. In fact, as we shall establish in the following, our methods can take advantage of approaches like those developed in [29] in order to support the motion control of the system vehicles through their allocated stages. At the same time, the proposed methods complement those past results with a complete theoretical framework that is able to support in an effective and computationally efficient manner the vehicle interaction and coordination, as they move through the shared regions of the underlying motion space.

In the light of all the above discussion, the main contributions of this work can be summarized as follows: (i) It proposes a complete novel approach for dealing with the problems of safety and liveness in free-ranging multi-vehicle systems that is based on the discretization of the vehicular dynamics and the development of a hybrid control scheme that is less computationally demanding and, therefore, more scalable to large-scale applications of these systems. (ii) It provides a complete and rigorous characterization of the discretizing model mentioned above and details the process of its abstraction from the continuous dynamics of the underlying traffic system. Instrumental in this abstraction are (a) a tesselation of the motion area and (b) a notion of "resource allocation" that it is induced by this tesselation. (iii) The presented work also provides complete and rigorous characterizations of the problems of (motion) safety and liveness in the context of the aforementioned resource allocation function and the new operational regime that it defines for the underlying vehicle system, and effective and computationally efficient solutions to these problems. These solutions are obtained by customizing to the considered application domain results borrowed from the emerging broader area of real-time management of resource allocation systems. (iv) The manuscript also discusses issues that concern the practical implementation of the aforementioned results, and identifies additional opportunities for their strengthening and extension. (v) Finally, it establishes the rather surprising result that this new framework can be applicable even in the case of vehicles that must be in perpetual motion until they reach their destination and retire (and therefore, they cannot come to a halt while contesting for the additional "resources" – i.e., access to new areas – that are necessary for their advancement).

The rest of the paper is organized as follows: Section II introduces the RAS concept of [24], the problem of deadlock avoidance arising in it, and their formalization through the framework of Deterministic Finite State Automata [14]. Subsequently, Section III employs the developments of Section II in order to provide a detailed, complete statement of the problem considered in this work. Section IV addresses the abstraction of the continuous-time dynamics of the considered traffic systems to a motion-coordinating RAS, while Section V deals with the notion of deadlock and the problem of deadlock avoidance arising in this particular RAS. More specifically, Section V provides a customized instantiation of Banker's algorithm [15] for the considered problem context. Section VI extends the aforementioned results to traffic systems where all the vehicles must be in perpetual motion until their retirement. Section VII considers briefly some more practical aspects of the implementation of the presented results and highlights future research issues that can complement and extend the proposed approach. Finally, Section VIII concludes the paper epitomizing its major contributions.

II. Resource allocation systems

As explained in the Introduction, this work will seek to establish the safety and liveness of the considered multi-vehicle systems by abstracting and managing a resource allocation system (RAS). In most general terms, a RAS is a dynamical system consisting of a set of concurrently executing processes that compete for their access to shared resources. A formalism particularly suitable to represent the interactive dynamics of these processes from the standpoint of the aforestated concerns of safety and liveness, is the *Deterministic Finite State Automaton (DFSA)* [14]. In this section, first we provide a systematic introduction of the RAS model to be employed in this work and its further abstraction to a DFSA. Then, in the second part of the section, we employ the considered RAS and its DFSA-based representation in order to formally characterize some concepts and properties relating to the liveness of its behavior. Finally, for reasons of completeness, we also provide an appendix with a brief introduction to the DFSA modeling framework and some additional concepts in that framework that are useful for the developments presented in this work.

A. L-CON-RAS and their DFSA-based modeling

In this subsection, first we review the RAS concept, as defined in [24], and subsequently we introduce the further conditions that will define the RAS sub-class considered in this work. Also, in the rest of this manuscript, the notation $\mathbb{Z}$, $\mathbb{Z}_0^+$ and $\mathbb{Z}^+$ will respectively denote the set of integers, non-negative integers and strictly positive integers.

Definition 1: [24] A *(sequential) resource allocation system (RAS)* is defined as a 4-tuple $\Phi = < \mathcal{R}, C, \mathcal{P}, \mathcal{A} >$[2] where:

1) $\mathcal{R} = \{R_1, \ldots, R_m\}$ is the set of the system *resource types*.
2) $C : \mathcal{R} \rightarrow \mathbb{Z}^+$ is the system *capacity* function, with $C(R_i) \equiv C_i$ characterizing the number of identical units from resource type R_i that are available in the system. Resources are considered to be *reusable*, i.e., they are engaged by the various processes according to an allocation/de-allocation cycle, and each such cycle does not affect their functional status or subsequent availability.
3) $\mathcal{P} = \{J_1, \ldots, J_n\}$ is the set of the system *process types* supported by the considered system configuration. Each process type J_j is a composite element itself; in particular, $J_j = < \mathcal{S}_j, \mathcal{G}_j >$, where:
   a) $\mathcal{S}_j = \{\Xi_{j1}, \ldots, \Xi_{j,l(j)}\}$ is the set of *processing stages* involved in the definition of process type J_j, and
   b) $\mathcal{G}_j$ is a data structure that defines the sequential logic over the set of processing stages $\mathcal{S}_j$, that governs the execution of any process instance of type J_j.
4) $\mathcal{A} : \bigcup_{j=1}^{n} \mathcal{S}_j \rightarrow \prod_{i=1}^{m} \{0, \ldots, C_i\}$ is the *resource allocation function*, which associates every processing stage Ξ_{jk} with a *resource allocation request* $\mathcal{A}(j,k) \equiv \mathcal{A}_{jk}$. More specifically, each $\mathcal{A}_{jk}$ is an m-dimensional vector, with its i-th component indicating the number of resource units of resource type R_i necessary to support the execution of stage Ξ_{jk}. Obviously, in a well-defined RAS, $\mathcal{A}_{jk}(i) \leq C_i, \ \forall j,k,i$. Also, it is assumed that $\mathcal{A}_{jk} \neq \mathbf{0}$, i.e., every processing stage requires at least one resource unit for its execution.

For complexity considerations, we also define the quantity $|\Phi| \equiv |\mathcal{R}| + |\bigcup_{j=1}^{n} \mathcal{S}_j| + \sum_{i=1}^{m} C_i$ as the *"size"* of RAS Φ. It is clear from the above that Definition 1 encompasses an entire taxonomy of RAS, obtained by the further specification of the underlying process structure (cf. item 3(b)), the capacities of the various resource types (cf. item 2), and the applied resource allocation protocol (cf. item 4). Next we focus on a particular RAS class that is known as the class of *Linear Conjunctive (L-CON-) RAS* [24] and is of special interest in the resource allocation to be considered in this work. This class is defined by the following two conditions regarding the aforementioned items:

Condition 1: In L-CON-RAS, the data structure $\mathcal{G}_j$ that defines the sequential logic of process type J_j, $j = 1, \ldots, n$, corresponds to a total ordering of the processing stage set $\mathcal{S}_j$. Without loss of generality, in the following we shall assume that this total ordering is defined by the index k of Ξ_{jk} with $k \in \{1, \ldots, l(i)\}$. The implication of the imposed total ordering is that any process instance of J_j will execute the processing stages in $\mathcal{S}_j$ sequentially, starting from Ξ_{j1} and terminating with $\Xi_{j,l(j)}$.

Condition 2: In L-CON-RAS, the resource allocation requests $\mathcal{A}_{jk}$, $j = 1, \ldots, n$, $k = 1, \ldots, l(j)$, are "conjunctive", i.e., a processing stage Ξ_{jk} requests a nonempty and possibly non-singleton subset of the system resources for its execution. Furthermore, a process instance executing processing stage Ξ_{jk} will be able to advance to its successor processing stage $\Xi_{j,k+1}$, only after it is allocated the resource differential $(\mathcal{A}_{j,k+1} - \mathcal{A}_{jk})^+$; and it is only upon this advancement that the process will release the resource units $|(\mathcal{A}_{j,k+1} - \mathcal{A}_{jk})^-|$, that are not needed anymore.[3]

[2]The complete definition of a RAS, according to [24], involves an additional component that characterizes the time-based – or *quantitative* – dynamics of the RAS, but this component is not relevant in the modeling and analysis to be pursued in the following developments, and therefore, it is omitted.

[3]We remind the reader that for any given $a \in \mathbb{Z}$, $a^+ \equiv \max\{a, 0\}$ and $a^- \equiv \min\{a, 0\}$.

Next we discuss how the dynamics of the L-CON-RAS can be further formalized in the DFSA modeling framework. Without any loss of generality, and for reasons that will become clear in the subsequent developments of this paper, in the following we shall assume that each process instance defines a distinct process type.

Definition 2: The DFSA $G(\Phi) = (\Sigma, E, \Gamma, f, \sigma_0, \Sigma_M)$ abstracting the feasible dynamics of an L-CON-RAS $\Phi =< \mathcal{R}, C, \mathcal{P}, \mathcal{A} >$ is defined as follows:

1) The *state set* Σ consists of all vectors $\sigma = [\sigma_1, \sigma_2, \ldots, \sigma_n] \in \mathbb{Z}^n$ such that:
$$\forall j \in \{1, \ldots, n\}, \quad 0 \leq \sigma_j \leq l(j) + 1 \tag{1}$$
and
$$\forall i \in \{1, \ldots, m\}, \quad \sum_{j=1}^{n} \mathcal{A}_{j,\sigma_j}(i) \leq C_i \tag{2}$$
Each component σ_j of σ indicates the current stage of process J_j. In particular, $\sigma_j = 0$ indicates that process J_j has not been initiated yet, while $\sigma_j = l(j) + 1$ indicates that process J_j has been completed. Furthermore, in Equation 2 it is implicitly assumed that $\forall j, i, \ \mathcal{A}_{j0}(i) = \mathcal{A}_{j,l(j)+1}(i) = 0$.
2) The *event set* $E = \{e_{jk} \mid j = 1, \ldots, n; k = 1, \ldots, l(j) + 1\}$, where for every $j = 1, \ldots, n$: (a) the event e_{j1}, represents the initiation of process J_j by the allocation of the resource set $\mathcal{A}_{j1}$; (b) the events e_{jk}, $k = 2, \ldots, l(j)$, represent the advancement of process J_j from processing stage $\Xi_{j,k-1}$ to processing stage Ξ_{jk} through the corresponding adjustment of its resource allocation; and (c) the event $e_{j,l(j)+1}$ represents the termination of process J_j and the release of all the resources currently held by it.
3) For each pair (σ, e_{jk}) we define $\sigma' = f(\sigma, e_{jk})$ such that the components σ'_q, $q = 1, \ldots, n$, of σ' are given by
$$\sigma'_q = \begin{cases} \sigma_q + 1 & \text{if } q = j \wedge \\ & (1 \leq k \leq l(j) + 1 \ \wedge \ \sigma_q = k - 1) \\ \sigma_q & \text{otherwise} \end{cases}$$
4) For each state $\sigma \in \Sigma$, $\Gamma(\sigma) = \{e \in E : \sigma' = f(\sigma, e) \in \Sigma\}$.
5) The *initial state* $\sigma_0 = \mathbf{0}$, which corresponds to the situation where no process has been initiated, and therefore, all the system resources are free.
6) The *set of marked states* Σ_M is the singleton $\{s_M = [l(1)+1, \ldots, l(n)+1]\}$, and it expresses the requirement for complete process runs.

B. Deadlocks and deadlock avoidance in L-CON-RAS

A major concern in the logical control of L-CON-RAS is the establishment of *live* – or *deadlock-free* or *non-blocking* – behavior. *Deadlocks* are defined as RAS states where there is a set of processes such that each of them, in order to advance, requests the allocation of resources currently held by some other process(es) in the considered set. Their development results from (i) the fact that processes will hold upon their allocated resources in a non-preemptive manner and (ii) the arbitrary resource requirements of the process stages, as expressed by function $\mathcal{A}(j, k)$, that can give rise to cyclical patterns of resource requests among the various executing processes.

In the DFSA abstraction of the L-CON-RAS operation, the presence of deadlocks is manifested by the presence of states $\sigma \in RS(\sigma_0)$ from which there is no path to the marked state σ_M, in the transition graph TG of automaton $G(\Phi)$. Such states σ will be characterized as *non-live* in the following. This characterization of the state liveness further implies that a *correct Deadlock Avoidance Policy (DAP)* must restrict the system operation to an *acyclic* subgraph of TG that contains the initial state σ_0 as the single "source" node and the marked state s_M as the single "terminal" node. In the representation of the DFSA model $G(\Phi) = (\Sigma, E, \Gamma, f, \sigma_0, \Sigma_M)$, the development of a DAP is equivalent to establishing a *liveness-enforcing supervisor (LES)* Γ^x.[4] Such a LES is characterized as *optimal*, and denoted by Γ^*, if the corresponding transition graph TG^* is the *maximal* subgraph of TG that satisfies the correctness property stated above. The set of states reachable in $G^*(\Phi)$ includes all the reachable live states of $G(\Phi)$ and leaves out all those states of $G(\Phi)$ that are not live.

In the L-CON-RAS operational context, the optimal LES Γ^* is well-defined and it is effectively computable through an *one-step lookahead* scheme that admits a tentative feasible event *iff* the resulting state is live. However, the corresponding problem of assessing the state liveness in L-CON-RAS is NP-complete [30]. In the light of this result, the research community has pursued two alternative solutions: 1) The identification of special L-CON-RAS structure that can admit the deployment of the optimal DAP in polynomial complexity with respect to the RAS size. 2) The synthesis of sub-optimal DAPs that are implementable in polynomial complexity with respect to the underlying RAS size, and yet, efficient, i.e., they manage to admit a large part of TG^*. The basic mechanism for developing this second class of policies is through the identification of some surrogate condition to state liveness, $H(s)$, such that (i) $H()$ is polynomially testable on any given RAS state s and (ii) the application of this condition in an one-step-lookahead control policy, while starting from the RAS initial state s_0, will lead to a correct DAP (where correctness should be specified as in the previous paragraphs). We shall further concretize the concepts and techniques underlying the development of this last class of policies in Section V, where we present a particular policy from this class that is suitable for the management of the resource allocation function considered in this document.

III. Problem statement

Having provided, in the previous section, a formal characterization of the concept of resource allocation system and of the need for liveness-enforcing supervision that arises in these environments, next we proceed to a more complete description of the problem that is addressed in this work and of the proposed methodology. Hence, in this section we first detail the multi-vehicle system that is the focus of our study, and subsequently we highlight the proposed approach and

[4]A formal definition of the concept of liveness-enforcing supervisor for any given DFSA is provided in the appendix.

the specific tasks that need to be fulfilled for its successful implementation.

The multi-vehicle system to be considered in the following consists of a set of autonomous mobile agents that move in a finite planar motion area $\mathcal{U} \subset \mathbf{R}^2$. Each agent is represented by a disk of radius ρ, and its center follows a pre-specified path that is given in the parametric form

$$x^c = x^c(t), \quad y^c = y^c(t), \quad t \in [0, T] \tag{3}$$

Each of the agent paths is of bounded *curvature*, $K(t)$, with

$$K(t) = \frac{\dot{x}^c(t)\ddot{y}^c(t) - \ddot{x}^c(t)\dot{y}^c(t)}{(\dot{x}^c(t)^2 + \dot{y}^c(t)^2)^{3/2}} = \frac{1}{R(t)} \tag{4}$$

$R(t)$ denotes the path *radius* at time t. At any time point t, the agent is described by its *configuration* $a(t) = (x^c(t), y^c(t), \Theta(t))$, where $\Theta(t)$ is the angle defining the orientation of the agent at time t, and it is measured with respect to the x-axis. Starting from an initial configuration $a(0) = (x^c(0), y^c(0), \Theta(0))$, an agent will realize its designated path, specified by Equation 3, if its orientation $\Theta(t)$, linear velocity $v(t)$, and angular velocity $\omega(t)$ change according to the model

$$\left\{ \begin{array}{rcl} \dot{x}^c(t) & = & v(t)\ cos(\Theta(t)) \\ \dot{y}^c(t) & = & v(t)\ sin(\Theta(t)) \\ \dot{\Theta}(t) & = & \omega(t)\ =\ v(t)\ K(t) \end{array} \right. \tag{5}$$

We assume that the path of each agent starts and ends at *"private"* locations, i.e., the areas occupied by the agents at these points are disjoint, and the agents are retired from the system upon reaching their destination. However, during their concurrent motion in the system, the agents share the available space, and in order to avoid collisions, they may need to modify their path and velocity profiles. For such a system, the basic problem that we are trying to solve can be stated as follows:

Find a mechanism to dynamically modify the initially assumed motion control of the system agents, so that: 1) in a finite time interval, all the agents will have attained their destination locations, and 2) at each moment of this time interval, the areas occupied by any given pair of the agents are disjoint. An additional concern is that the developed mechanism compromises the underlying system performance to a minimum possible extent, where the latter is determined according to a pertinently selected performance index.

In the above statement, by the "initially assumed motion control", we imply the control law derived from Equations 3-5, that would enable each agent to realize its designated path without the disturbance of the remaining agents. The requirement for "dynamic modification" means that changes in the motion profile of the different agents must be determined on-line, based on available information about the system state, where the latter is defined by the agent configurations $a(t)$. As remarked in the introductory section, the satisfaction of the aforestated requirement in a way that scales well with respect to the number of the moving agents, has been a challenging task. In this work we address this challenge with a hybrid control scheme that: a) partitions the agents' continuous motion processes into stages, and b) assumes an independent, continuous control of each agent's motion within a stage, while stage transitions are executed under a discrete, RAS-based control model.

The aforementioned discretization of the vehicle motion processes into stages is attained by means of the *tesselation* of the motion plane into a number of areas that will be referred to as *"cells"*. An agent is said to *"occupy"* a certain cell if its disk overlaps with the area corresponding to that cell. The defining characteristic of the proposed methodology is the stipulation that at any point in time, each cell can be occupied by at most one agent. Hence, the cells defined by the adopted tesselation constitute fictitious *"resources"* that must be acquired and released by any vehicle during its trip. Furthermore, the entire vehicle motion is naturally segmented in a sequence of *"stages"*, where each stage is defined by a maximal path segment with constant cell (i.e., resource) occupation. The exclusive allocation to a vehicle of all the cells that correspond to a certain stage of its trip implies that the vehicle can execute the corresponding motion segment in a safe and undisrupted manner. On the other hand, the vehicle transition from a certain stage to the next requires the negotiation of any extra cells that are engaged in the execution of the new stage. This last effect necessitates the deployment of a *resource allocation protocol* that will facilitate the aforementioned negotiation among the vehicles and will ensure the liveness of the entire system.

In order to provide a complete realization of the approach described in the previous paragraph we must systematically address the following four issues:

1) the detailed specification of the tesselation mentioned above, i.e., the discretization of the motion area $\mathcal{U}$ and of the agent motion paths;
2) the development of the control logic to be followed by an agent during the acquisition and release of the necessary cells for the execution of its different process stages, that will guarantee mutually exclusive cell usage by the different agents;
3) the development of additional control logic that will guarantee the deadlock freedom and the liveness of the induced resource allocation system; and
4) the eventual integration of the continuous-time dynamics describing the agent motion and the discrete-event dynamics induced by the imposed resource allocation system into a hybrid agent control model.

In the following, we provide a systematic treatment of items #1 – #3 in the above list. Item #4 in this list, i.e., the control of the continuous vehicle motion so that it adheres to the control logic imposed by the RAS LES that allocates the system cells and coordinates the vehicle advancement among the different stages of their motion plans, is rather straightforward and it can be formally addressed through techniques similar to those presented in [29]. Closing this section we notice, for completeness, that some earlier works that present conceptual affinity to the considered problem, and also a preliminary version of the results presented in this paper, can be found in [31], [32], [33], [34].

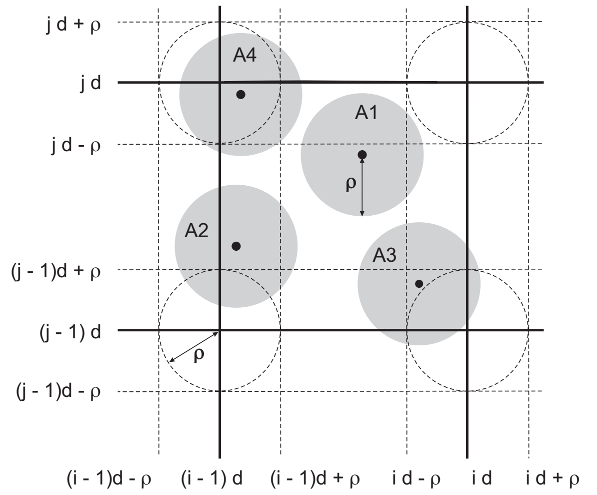


Fig. 1. The proposed tesselation, the mappings $\mathcal{W}$ and $\mathcal{W}^{-1}$, and the further partitioning of the motion space induced by them.

IV. The proposed tesselation of the motion plane and the induced resource allocation system

The tesselation of the agent motion space that will lead to the discretization of their motion and to the abstraction of the relevant RAS outlined in the previous section, can take place in many different ways. In general, the selection of such a tesselation scheme should be driven by (i) the tractability of the necessary calculations for determining the resource requirements of the different agents, and (ii) the efficiencies attained by the resulting resource allocation. The evaluation of these efficiencies can be based on a number of (possibly conflicting) measures, with some typical examples being (a) the delays that are experienced by the different agents as a result of the induced resource allocation, (b) the rate of the trip completion, and (c) the resulting space occupancy.

When it comes to criterion (i) in the above list, one of the simplest tesselation schemes is provided by a grid of horizontal and vertical lines spaced at a distance $d \geq 2\rho$ and centered at the origin of a coordinate system, (x, y), that is superimposed on the motion plane. The resulting cells will be denoted by $W = \{w[i,j] \; : \; i \in \{-\underline{I}, \ldots, -1, 0, 1, \ldots, \overline{I}\}, \; j \in \{-\underline{J}, \ldots, -1, 0, 1, \ldots, \overline{J}\}\}$, where $-\underline{I}$, $\overline{I}$, $-\underline{J}$, and $\overline{J}$ are taken large enough to encompass the entire area $\mathcal{U}$, that supports the agent motion. Then, given a point $(x, y) \in \mathcal{U}$ and a cell $w[i, j]$, we define

$$\begin{aligned} (x, y) \in w[i, j] &\Longleftrightarrow \\ (i-1) \cdot d \leq x \leq i \cdot d \; &\wedge \; (j-1) \cdot d \leq y \leq j \cdot d \end{aligned} \tag{6}$$

The size d of the grid, that defines the length of the cell edges, should be selected by considering the efficiency criteria mentioned above. In general, a smaller value of d can accommodate a larger number of agents, and therefore, can lead to a higher space occupancy, but at the same time, it will lead to more disruption of the agent travels by the superimposed resource allocation process, and possibly to more congested traffic and longer delays.

We shall say that an agent (with its disk) centered at (x^c, y^c) *lies in* cell $w[i, j]$ iff $(x^c, y^c) \in w[i, j]$.[5] On the other hand, following the discussion of the previous section, we shall say that an agent centered at (x^c, y^c) *occupies* cell $w[i, j]$ *iff* there exists $(x, y) \in w[i, j]$ with $||(x, y) - (x^c, y^c)|| \leq \rho$, where $|| \cdot ||$ denotes the Euclidean norm.[6] Clearly, this definition induces a mapping $\mathcal{W}$ from the motion area, $\mathcal{U}$, to the powerset of W, 2^W, that maps to any point $(x, y) \in \mathcal{U}$ the cell subset $\mathcal{W}(x, y) \in 2^W$ consisting of the cells occupied by an agent centered at (x, y). A graphical illustration of this mapping $\mathcal{W}$ is given in Figure 1. More specifically, in Figure 1, the adopted tesselation is defined by the grid of the solid horizontal and vertical lines, and the mobile agents are depicted by the grey disks in it. The reader should notice that an agent can occupy one cell (as in the case of $A1$), two neighboring cells (as in the case of $A2$), three neighboring cells (as in the case of $A3$), or four neighboring cells (as in the case of $A4$).

Next we show that for the tesselation scheme considered in this work, the number of cells occupied by a mobile agent that is located at (x^c, y^c) is effectively determined by the relative positioning of (x^c, y^c) with respect to another partitioning of the motion plane, that is induced by the original tesselation scheme and the agent geometry. In Figure 1, this induced partitioning is defined by the depicted dashed lines. In order to develop a formal characterization for it, it is instructive to consider the *"inverse"* mapping of $\mathcal{W}$, $\mathcal{W}^{-1}$, that is defined for any $C \in 2^W$ by $\mathcal{W}^{-1}(C) \equiv \{(x, y) \in \mathcal{U} : \mathcal{W}(x, y) = C\}$. In more natural terms, for each set of cells $C \in 2^W$, $\mathcal{W}^{-1}(C)$ is the set of all points $(x, y) \in \mathcal{U}$ such that the disk of an agent centered at (x, y) overlaps each of the cells contained in C, and only these cells. Figure 2 depicts $\mathcal{W}^{-1}(C)$ in the particular case that C is the singleton $\{w[i, j]\}$, while Figure 3 depicts the structure of $\mathcal{W}^{-1}(C)$ in the cases where the set C consists of two, three or four neighboring cells. Furthermore, it is clear that in the case that C contains two or more non-neighboring cells, $\mathcal{W}^{-1}(C) = \emptyset$.

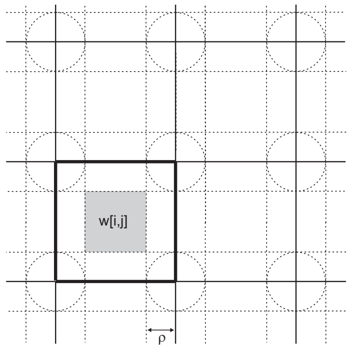


Fig. 2. The "inverse" mapping $\mathcal{W}^{-1}(C)$ for a single cell, $C = w[i, j]$.

[5]It should be noticed that according to Equation 6, an agent can lie in more than one cells at the same time. Especially, in the (rather singular) case that the agent center is located at the intersecting point of two grid lines, the agent will lie in all four neighboring cells.

[6]In order to maintain a simple notation, in the entire discussion of this manuscript we have assumed that the system agents are homogeneous with respect to their disk size. If, however, this is not the case, but each agent Ξ_k occupies a disk of distinct radius, ρ_k, the concepts and structures defined in the rest of this section still apply, but they are customized for each agent through their parameterization by the agent radius ρ_k.

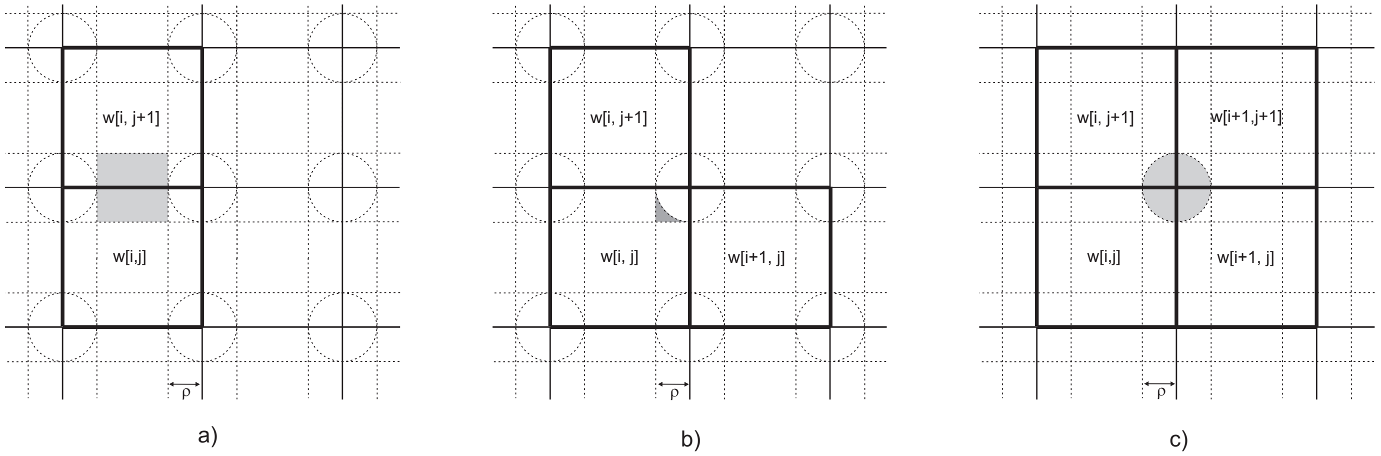


Fig. 3. The "inverse" mapping $\mathcal{W}^{-1}(C)$ in the case that set C consists of: a) two cells, b) three cells, and c) four cells; the corresponding cells are annotated by thicker lines.

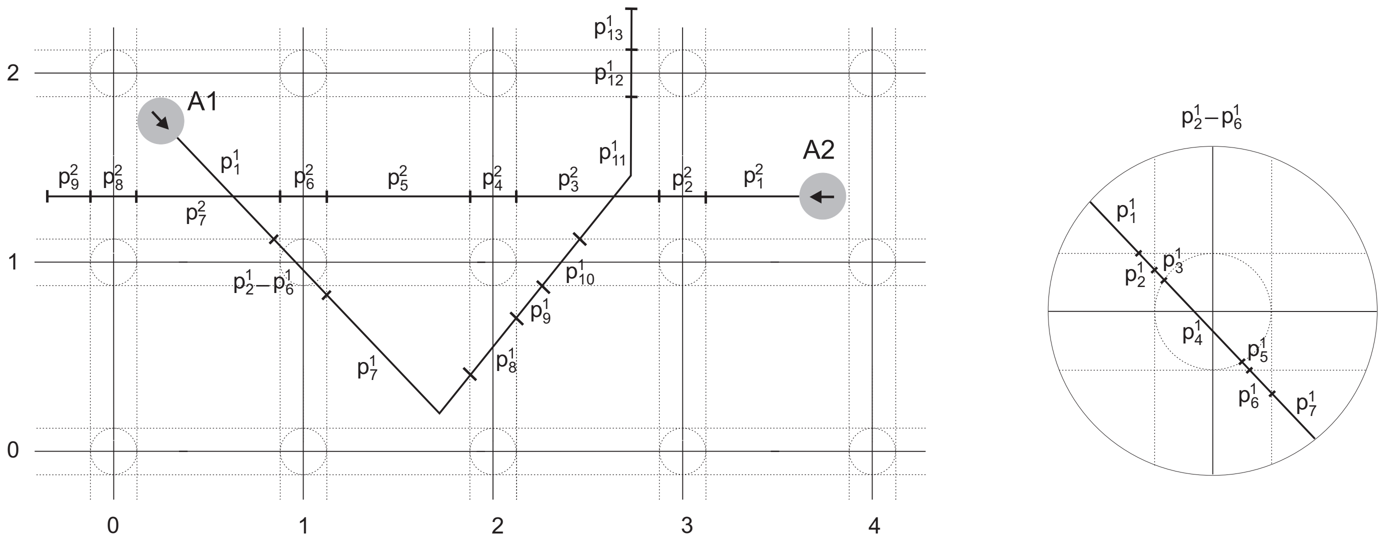


Fig. 4. Example paths for two mobile agents, and the corresponding resource allocation profiles that are defined by the path partitioning into maximal segments with the same cell occupation. The right part of the figure details the profile obtained for agent $A1$.

TABLE I

THE RESOURCE ALLOCATION INDUCED BY THE PATH SEGMENTATION OF FIGURE 4

Agent A_1

Stage No. j	Required resources $\mathcal{A}(1,j)$
1	$w[0,1]$
2	$w[0,0], w[0,1]$
3	$w[0,0], w[0,1], w[1,1]$
4	$w[0,0], w[0,1], w[1,0], w[1,1]$
5	$w[0,0], w[1,0], w[1,1]$
6	$w[0,0], w[1,0]$
7	$w[1,0]$
8	$w[1,0], w[2,0]$
9	$w[2,0]$
10	$w[2,0], w[2,1]$
11	$w[2,1]$
12	$w[2,1], w[2,2]$
13	$w[2,2]$

Agent A_2

Stage No. j	Required resources $\mathcal{A}(2,j)$
1	$w[3,1]$
2	$w[3,1], w[2,1]$
3	$w[2,1]$
4	$w[2,1], w[1,1]$
5	$w[1,1]$
6	$w[1,1], w[0,1]$
7	$w[0,1]$
8	$w[0,1], w[-1,1]$
9	$w[-1,1]$

Next, we consider the binary relation $\mathcal{R}$ that is defined on the motion plane $\mathcal{U}$ by

$$\forall\ ((x_1,y_1),(x_2,y_2)) \in \mathcal{U}^2, \quad \mathcal{R}((x_1,y_1),(x_2,y_2)) \iff \mathcal{W}(x_1,y_1) = \mathcal{W}(x_2,y_2) \quad (7)$$

Clearly, $\mathcal{R}$ is an *equivalence* relation on $\mathcal{U}$. The *equivalence classes* of $\mathcal{R}$ are defined by

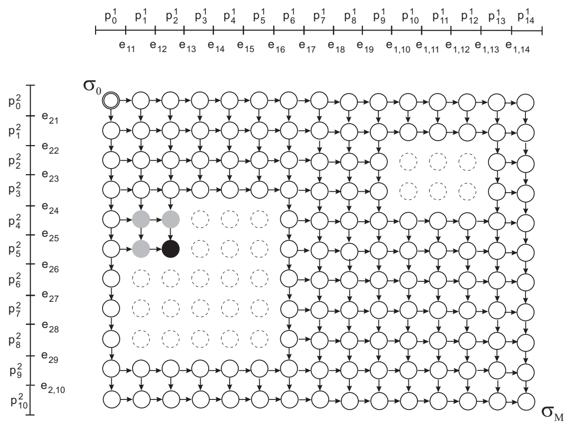
$$\{\mathcal{W}^{-1}(C): \quad C \in 2^W \ \wedge\ \exists\ (x,y) \in \mathcal{U}:\ \mathcal{W}(x,y) = C\} \quad (8)$$

From a more geometric standpoint, the equivalence classes of $\mathcal{R}$ establish a *partitioning* of the motion plane $\mathcal{U}$ into the regions identified by Equation 8.[7] When combined with the continuity of the agent motion, this partitioning of $\mathcal{U}$ enables the specification of a *"resource allocation profile"* for any given agent, that is induced by its motion profile (or *"path"*) and expresses the cell occupation and release during the evolution of this motion. Figure 4 exemplifies the abstracting notion of the resource allocation profile, by applying it on the motion profiles, p^1 and p^2, of two agents, A_1 and A_2. Path p^1 consists of thirteen (maximal) segments p^1_1 - p^1_{13}, and path p^2 consists of nine such segments, p^2_1 - p^2_9. Also, Table I specifies the cells occupied by the two agents at the various stages of their route. The motion of an agent can, thus, be viewed as a sequence of stages, each of which requires exclusive access to a particular subset of resources (i.e., cells). Consequently, the system of free ranging agents, can be considered as an L-CON-RAS.

However, the aforementioned L-CON-RAS that models the cell occupancy by the different agents as they progress through their motion profile presents additional structure, that renders the resulting RAS a proper sub-class of L-CON-RAS. The main attributes that define this new RAS class and differentiate it from any other element of the broader L-CON-RAS class, stem from the following two facts:

1) The resource allocation and/or de-allocation that takes place during the transition between two consecutive processing stages, must observe a "resource proximity" relation that is defined by the adopted tesselation. More specifically, in the considered RAS systems, the allocation corresponding to a particular processing stage must be interpretable as the occupation of a number of neighboring cells by the corresponding mobile agent, while the variation of the allocations between two consecutive processing stages must be interpretable as the occupation of some new neighboring cells and/or the release of some previously held ones, during the agent motion.
2) Furthermore, as stated in the previous section, in this prototypical introduction of the presented method, we assume that each cell can be occupied by at most one agent at a time, which implies a unit capacity for the system resources.

[7]In the mathematical theory of binary relations, this partitioning is known as the *equivalence kernel* of function $\mathcal{W}$, and it is denoted by $ker\mathcal{W}$. The equivalence classes of $ker\mathcal{W}$, described by Equation 8, are known as the *fibers* of $\mathcal{W}$. [35]



p^1_0 p^1_1 p^1_2 p^1_3 p^1_4 p^1_5 p^1_6 p^1_7 p^1_8 p^1_9 p^1_{10} p^1_{11} p^1_{12} p^1_{13} p^1_{14}
e_{11} e_{12} e_{13} e_{14} e_{15} e_{16} e_{17} e_{18} e_{19} $e_{1,10}$ $e_{1,11}$ $e_{1,12}$ $e_{1,13}$ $e_{1,14}$
p^2_0 p^2_1 p^2_2 p^2_3 p^2_4 p^2_5 p^2_6 p^2_7 p^2_8 p^2_9 p^2_{10}
e_{21} e_{22} e_{23} e_{24} e_{25} e_{26} e_{27} e_{28} e_{29} $e_{2,10}$
σ_0
σ_M

Fig. 5. The transition graph for the DFSA Φ that represents the dynamics of the FREE-RANGE-RAS abstracted from the two motion profiles depicted in Figure 4.

The sub-class of L-CON-RAS that possesses the aforementioned additional features will be characterized as FREE-RANGE-RAS. Next, we characterize the DFSA Φ that formalizes the dynamics of a FREE-RANGE-RAS according to the spirit of Section II. This automaton is obtained by the synchronous composition of the simple DFSAs that represent the resource allocation profiles of the various agents, under the further constraint that a certain state σ and the transitions leading into it are feasible *iff* it satisfies the condition of Equation 2. Foregoing the formal characterization of this concept, as it would require a lengthy but rather pedantic sequence of definitions, we provide a systematic exposition of it in Figure 5. More specifically, the DFSA Φ depicted in Figure 5 represents the dynamics of the FREE-RANGE-RAS that is abstracted from the two motion profiles depicted in Figure 4. The states of the automaton Φ are arranged in an array with state $\sigma[l,k]$ corresponding to the state where agent A_1 is executing its l-th, stage and agent A_2 is executing its k-th stage. The exact range of the state components l and k has been determined from Table I and the state semantics for the L-CON-RAS introduced in Section II-A. At the same time, the state of the agent trips uniquely defines the resource allocation state; each of the agents A_1 and A_2 is allocated the set of cells respectively defined by $\mathcal{A}(1,l)$ and $\mathcal{A}(2,k)$ (cf. Table I). All the rows of arcs in the depicted digraph represent state transitions associated with agent A_1, and are labeled with the events given over the first row. All the columns of arcs represent state transitions associated with agent A_2 and are labeled with the events given next to the first column. The left upper node represents the initial state $\sigma_0 = [0,0]$, where no agent motion has been initiated, and therefore, no resources are allocated to the agents. Thus, any allocation request of any of the agents can be satisfied, i.e., $\Gamma(\sigma_0) = \{e_{11}, e_{21}\}$, and the transition function f is defined for both (σ_0, e_{11}) and (σ_0, e_{21}). The occurrence of event e_{11} causes the transition to state $\sigma' = [1,0]$, and the occurrence of event e_{21} causes the transition to state $\sigma'' = [0,1]$. On the other hand, the conflicting resource requirements posed by the two agents when they execute concurrently certain pairs of their stages, imply that not all states σ in the aforementioned array are reachable. For example, in state $\sigma = [3, 3]$, only the event e_{14}, associated with agent A_1, is feasible. The event e_{24}, that is associated with agent A_2, would lead to state $\sigma = [3, 4]$ and it can be seen from Table I that the two agents would conflict regarding the occupancy of cell $w[1, 1]$. Working in this manner, we can verify that the plausible resource allocation space for the two-agent system depicted in Figure 4 is the connected part of the digraph of Figure 5. The remaining nodes, representing the states that are not reachable from the initial state σ_0, are depicted with a dashed line. Finally, in the digraph of Figure 5, we also distinguish the black and the grey nodes from which there is no path to the marked state σ_M. These two node categories represent the *deadlock* and the *impending-deadlock* states, respectively. In the following section, we draw our attention to these two classes of nodes and discuss their prevention from the system behavior.

V. Deadlock and Deadlock Avoidance for the considered multi-vehicle systems

A. Interpreting the motion non-liveness through the formation of deadlocks

It is clear from the discussion of Section IV that, in the proposed regime, the agent transition from one state of their resource allocation profile to the next must be coordinated with the rest of the system so that conflicting (global) states of Φ are avoided. Practically, this means that upon reaching a boundary of an equivalence class defined by Equation 8 of Section IV, the agent must have to wait until all the cells in the set C that corresponds to that class are clear from any other agents, and only then it can proceed to its next stage. However, this advancement mechanism can lead to *circular dependencies* among a number of agents, where each agent waits upon the release of some cell(s) that are held by some other agent in this group. Hence, the further advancement of all these agents will be stalled indefinitely, and we shall say that these agents are (entangled) in a *deadlock*. The formation of deadlock and its insidious role in the development of non-live behavior by the considered multi-vehicle systems are exemplified more concretely through the vehicle motion that underlies the DFSA dynamics depicted in Figure 5. In particular, the reader should consider the motion dynamics corresponding to the state that is depicted in black in this figure; let us denote this state by σ for further reference. As indicated in Figure 5, in the considered state σ, agent A_1 executes the second stage of its resource allocation profile, and, according to the information provided in Table I, it occupies cells $w[0, 0]$ and $w[0, 1]$ of the tesselation applied on the motion plane. On the other hand, agent A_2 executes the fifth stage of its resource allocation profile and it occupies cell $w[1, 1]$ of the applied tesselation. Table I also reveals that agent A_1 requests cell $w[1, 1]$ for its further advancement to its next state, and agent A_2 requests cell $w[0, 1]$. Hence, these two agents block each other, and the resulting deadlock renders state σ a non-live state in the dynamics depicted in Figure 5.

The reader should notice that the aforementioned formation of deadlock in σ, incurs the non-liveness not only of state σ itself, but also of any other state from which state σ is unavoidable (under the discretized motion restrictions imposed by safety). In Figure 5, these additional non-live states are the three states depicted in grey. Hence, in order to maintain liveness, the system controller must guard not only against the formation of deadlock, but also against the transition into deadlock-free non-live states. This last remark is especially important when it comes to the computational complexity of the required liveness-enforcing supervision. More specifically, in the proposed operational regime, deadlock-containing states can be recognized by algorithms of polynomial complexity with respect to the underlying system size, where this size is defined by the number of the circulating agents, the number of cells in the adopted tesselation scheme, and the number of stages in the resource allocation profiles that are induced by this tesselation (c.f., [24]; Chpt. 2). On the other hand, the recognition of states that are themselves deadlock-free but from which deadlock is unavoidable, is an NP-hard problem [36]. Hence, as discussed in Section II, liveness enforcement in the considered system will typically be attained by a sub-optimal supervisor that will seek to establish a trade-off between the policy permissiveness and the computational complexity involved in its design and implementation.

B. Efficient deadlock avoidance policies for the considered class of multi-vehicle systems

Clearly, in resource allocation systems where each process is defined by a finite allocation sequence, the most straight-forward way to resolve state liveness is through a search procedure for a feasible sequence of process advancing events that brings every process to completion. We shall refer to such a sequence as a *process terminating (event) sequence*. In the context of this search for process terminating sequences, the non-polynomial complexity of the state liveness problem is manifested by the need to *backtrack* to a previously encountered state every time that the search reaches a deadlock. The policy presented next seeks to constrain the search for a terminating sequence over a subset of such sequences so that it can be performed in a *"greedy"* manner, i.e., without the need for backtracking. As a result, the complexity of this search remains polynomial. On the other hand, live states with no terminating sequences in the identified subset will have to be rejected; this is the price that must be paid for ensuring the computational efficiency and the scalability of the policy.

The basic concept that underlies the conceptual and computational definition of the proposed policy is that of an *ordered* state:

Definition 3: We shall say that state σ of the DFSA Φ modeling a FREE-RANGE-RAS is ordered *iff* there exists a terminating sequence that advances and terminates the vehicles activated in that state one at a time.

In other words, state σ is ordered *iff* we can order its activated vehicles $A_1, A_2, \ldots, A_n$ so that all the cells that must be occupied by vehicle A_i until the completion of its trip, are either free or they are currently occupied by a vehicle A_j that precedes vehicle A_i in the aforementioned order (and therefore they will be free by the time that vehicle A_i is picked for advancement). Clearly, every ordered state is live.

```
FUNCTION ordered $(k, \mathcal{A})$ : BOOLEAN

Input: (i) $k = [k_1, ..., k_n]$, where $k_i$ is the index of the current
        stage of active agent $A_i$ in the stage sequence $p^i_1, \ldots,$
        $p^i_{l(i)}$ defining the resource allocation profile of the agent
        (ii) function $\mathcal{A} = \mathcal{A}(i, j),\ i = 1, ..., n,\ j = 1, ..., l(i)$,
        that returns the set of cells that are occupied
        by agent $A_i$ in its stage $p^i_j$
Output: A decision of whether the resource allocation state
        described by the input is ordered or not.

BEGIN
  $occupied := \bigcup_{i=1}^{n} \mathcal{A}(i, k_i)$;
  FOR $i \in 1, \ldots, n$ DO $remain[A_i] := \bigcup_{j=k_i}^{l(i)} \mathcal{A}(i, j)$;
  $\mathcal{X} := \{A_i : i = 1, \ldots, n\}$;
  REPEAT
    $\mathcal{X}' := \mathcal{X}$;
    FOR $A_i \in \mathcal{X}$
      IF $remain[A_i] \cap occupied = \mathcal{A}(i, k_i)$ THEN
        $\mathcal{X} := \mathcal{X} \setminus \{A_i\}$;
        $occupied := occupied \setminus \mathcal{A}(i, k_i)$;
  UNTIL $\mathcal{X}' = \mathcal{X}$ OR $\mathcal{X} = \emptyset$;
  IF $\mathcal{X} = \emptyset$ THEN $ordered := TRUE$;
  ELSE $ordered := FALSE$;
END
```

Fig. 6. An implementation of Banker's algorithm for the resource allocation that takes place in the operational regime for multi-vehicle systems proposed in this work.

Furthermore, by confining the system operation to the subspace of ordered states we substitute the search for a vehicle terminating sequence with a search for a vehicle ordering that possesses the aforementioned property. But this search can be performed in a greedy manner, since every step that is taken in it increases monotonically the set of free cells in the underlying system, and therefore it can only improve the potential of the remaining vehicles to terminate. The resulting policy is typically known as the *Banker's algorithm* [15] for resource allocation, and its detailed instantiation for the considered class of systems is presented in Figure 6. Furthermore, recalling that under the proposed tesselation scheme an agent cannot occupy more than 4 cells at any point of its trip, it is easy to verify that the complexity of this algorithm is no more than $O(n^2 \cdot L)$, where $L \equiv \max_i\{l(i)\}$. Figure 7 depicts the subspace of the DFSA Φ that is admitted by the Banker's algorithm of Figure 6 when applied to the multi-vehicle system of Figure 4.

Figure 6 reveals that the Banker's implementation in the considered example is also quite efficient as it fails to admit only a very small part of the underlying live sub-space. Collective experience with the implementation of this class of policies in RAS abstractions resulting from other application domains indicates that this is a typical attribute of these policies, i.e., these policies they will provide a pretty large coverage of the target behavioral space. Yet, in the rest of this section we discuss a variation of the Banker's implementation of Figure 6 that can lead to ever more enhanced performance.

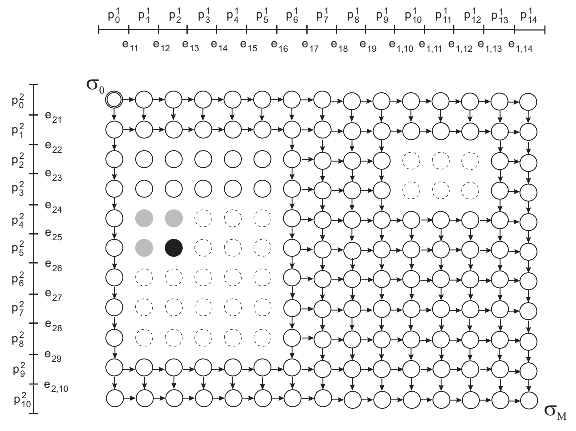

p^1_0 p^1_1 p^1_2 p^1_3 p^1_4 p^1_5 p^1_6 p^1_7 p^1_8 p^1_9 p^1_{10} p^1_{11} p^1_{12} p^1_{13} p^1_{14}
e_{11} e_{12} e_{13} e_{14} e_{15} e_{16} e_{17} e_{18} e_{19} $e_{1,10}$ $e_{1,11}$ $e_{1,12}$ $e_{1,13}$ $e_{1,14}$
p^2_0 p^2_1 p^2_2 p^2_3 p^2_4 p^2_5 p^2_6 p^2_7 p^2_8 p^2_9 p^2_{10}
e_{21} e_{22} e_{23} e_{24} e_{25} e_{26} e_{27} e_{28} e_{29} $e_{2,10}$
σ_0
σ_M

Fig. 7. The FSA modeling the behavior of the agent system of Figure 4 when operated under the original implementation of Banker's algorithm, provided in Figure 6.

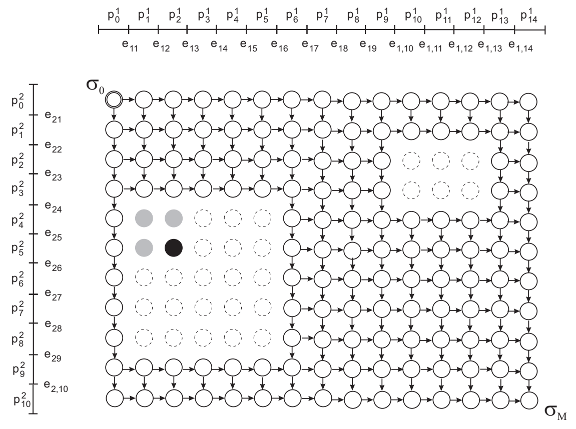

p^1_0 p^1_1 p^1_2 p^1_3 p^1_4 p^1_5 p^1_6 p^1_7 p^1_8 p^1_9 p^1_{10} p^1_{11} p^1_{12} p^1_{13} p^1_{14}
e_{11} e_{12} e_{13} e_{14} e_{15} e_{16} e_{17} e_{18} e_{19} $e_{1,10}$ $e_{1,11}$ $e_{1,12}$ $e_{1,13}$ $e_{1,14}$
p^2_0 p^2_1 p^2_2 p^2_3 p^2_4 p^2_5 p^2_6 p^2_7 p^2_8 p^2_9 p^2_{10}
e_{21} e_{22} e_{23} e_{24} e_{25} e_{26} e_{27} e_{28} e_{29} $e_{2,10}$
σ_0
σ_M

Fig. 8. The FSA modeling the behavior of the agent system of Figure 4 when operated under the modified implementation of Banker's algorithm.

This new variation of Banker's algorithm takes advantage of the fact that, in many cases, there will be areas of the motion plane $\mathcal{U}$ that are used exclusively by a single agent among the agents that are active in the evaluated state. More specifically, there will be agents A_i with stages p^i_j in their resource allocation profile such that the cells contained in $\mathcal{A}(i, j)$ will not be occupied by any other agent at any stage of its trip. We shall refer to stages in the agent resource allocation profiles that present this exclusivity of the engaged cells, as *"private"* stages. Furthermore, it is easy to see that the ability to bring each agent that is active in a state σ of the DFSA Φ to one of its private stages, implies the liveness of state σ (since the remaining profile segments of each agent can be executed one at a time, in any order). Hence, the search for a terminating sequence can be replaced by a search for an agent advancing sequence that collects all active agents to their next private stage.[8] This last observation further motivates the substitution of the original concept of the ordered state with the more relaxed concept of the *"p-ordered"* state:

Definition 4: We shall say that state σ of the DFSA Φ modeling a FREE-RANGE-RAS is p-ordered *iff* there exists a

[8]Obviously, agents that do not possess private stages must be led to completion by the sought sequences.

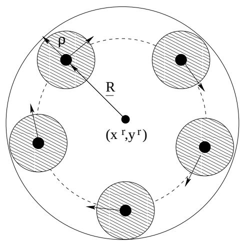


Fig. 9. The expanded disk reserved for each agent (modified from [1]).

vehicle advancing sequence that brings every vehicle activated in state σ to its next private stage or to the completion of its trip, if such a private stage does not exist, while advancing one vehicle at a time.

The resolution of this new property for any given state σ of Φ can be performed by a modified version of the algorithm of Figure 6, where the value of the variables $remain[A_i]$ are redefined as

$$\text{FOR } i \in 1,\ldots,n \ \text{ DO } \ remain[A_i] := \bigcup_{j=k_i}^{p_i(k_i)} \mathcal{A}(i,j);$$

In the above expression, the index $p_i(k_i)$ indicates the first private stage in the remaining (i.e., for $j \geq k_i$) resource allocation profile of agent A_i. The determination of this index from the provided input data is quite straightforward and its details are left to the reader. Figure 8 depicts the subspace of the DFSA Φ that is admitted by the aforementioned modification of the Banker's algorithm of Figure 6 when applied to the multi-vehicle system of Figure 4. As expected, the suggested modification increased significanty the admitted state space; in fact, in the considered case, the proposed modification led to the maximal permissiveness of the applied LES.

VI. Extending the proposed resource allocation paradigm to systems with agents in perpetual motion

In this section we extend the results of Sections IV–V to the case of multi-vehicle systems involving agents that must be in perpetual motion until they reach their destination and retire; a cruising airplane is a typical example of such an agent. The presented results are motivated by and capitalize upon some recent developments presented in [1]. More specifically, (slightly) generalizing the model of [1], we assume that the motion of any given agent is constrained by

$$\forall t, \quad v(t) \geq \underline{v} > 0 \ \wedge \ \omega(t) \in [-\overline{\omega}, \overline{\omega}] \tag{9}$$

Then, it is clear from Equation 5 that the minimum path radius $\underline{R}$ is attained by setting $v(t) = \underline{v}$ and $\omega(t) = \pm\overline{\omega}$. In particular, by setting $v(t) = \underline{v}$ and $\omega(t) = -\overline{\omega}$, the agent will keep moving on a *stationary* circle of minimal radius $\underline{R}$. The center $(x^r(t), y^r(t))$ of this circle can be derived from the agent configuration $a(t) = (x^c(t), y^c(t), \Theta(t))$ through the following equation:

$$(x^r(t), y^r(t)) \ = \ (x^c(t) + \underline{R}\cdot sin(\Theta(t)), y^c(t) - \underline{R}\cdot cos(\Theta(t))) \tag{10}$$

Furthermore, under this operational regime, the separation of the considered agent from the other agents in the traffic system can be ensured by exclusively allocating to it the entire disk centered at $(x^r(t), y^r(t))$, and having a radius $\rho^r = \underline{R} + \rho$, where ρ is the agent disk radius defined in Section III (c.f. Figure 9). Finally, by setting $\Theta^r(t) = \Theta(t)$, we can define a configuration $a^r(t) = (x^r(t), y^r(t), \Theta^r(t))$ for this expanded reservation area, with a structure similar to the configuration $a(t)$ of the original agent. The dynamics of configuration $a^r(t)$ are induced by the dynamics of the agent configuration $a(t)$, according to the following equations:

$$\begin{aligned}
\dot{x}^r(t) &= \dot{x}^c(t) + \underline{R}\cdot cos(\Theta(t))\cdot\dot{\Theta}(t) \\
&= (v(t) + \underline{R}\cdot\omega(t))\cdot cos(\Theta(t)) && (11)\\
\dot{y}^r(t) &= \dot{y}^c(t) + \underline{R}\cdot sin(\Theta(t))\cdot\dot{\Theta}(t) \\
&= (v(t) + \underline{R}\cdot\omega(t))\cdot sin(\Theta(t)) && (12)\\
\dot{\Theta}^r(t) &= \dot{\Theta}(t) \ = \ \omega(t) && (13)
\end{aligned}$$

Equations 11-13 reveal that configuration $a^r(t)$ is much more controllable than the original configuration $a(t)$. In particular, Equations 11 and 12 verify that by setting $v(t) = \underline{v}$ and $\omega(t) = \overline{\omega}$, configuration $a^r(t)$ can be stopped. Also, once stopped, $a^r(t)$ can be moved in any desired direction Θ^r, by letting the agent to rotate around the stationary point $(x^r(t), y^r(t))$, until $\Theta(t) = \Theta^r$, and then setting $\omega(t) = 0$. But then, one can ensure conflict-free motion for the considered traffic system by profiling the motion of the configurations $a^r(t)$ instead of the motion of the original agent configurations $a(t)$, and controlling this motion through the imposition upon the motion area $\mathcal{U}$ of a tesselation with a grid size $d \geq 2\rho^r$. The implementational details are straightforward and they are left to the reader.

VII. Practical Implementation and Further Research

In this section we briefly discuss the control "architecture" and the communication protocol that can facilitate the implementation of the control logic developed in the previous parts of this manuscript. The main intention of this discussion is (i) to provide a more concrete perspective regarding the practical applicability of the presented results, and also (ii) to highlight further research directions that can extend the applicability of these results by further alleviating the computational and / or the communication complexity of the control function.

It is generally true that the supervisory control problem of establishing non-blocking behavior in the context of various DES models necessitates a comprehensive / holistic view of the system behavior. As a result, the solutions that are developed for such supervisory control problems are typically characterized by a centralized structure; i.e., the various decisions are effected by a central controller that maintains a detailed representation of the state of the entire system and interacts with the various agents operating in the system through an appropriate communication protocol. This is also the typical control structure that is assumed by the RAS supervisory control theory. Hence, the most straightforward implementation of the control logic developed in the previous sections will involve a central controller – or supervisor – that maintains a characterization of the global resource allocation state in the form of the DFSA Φ that was introduced in Section IV. This supervisor has also knowledge of (i) the imposed tesselation on the motion area $\mathcal{U}$, and of (ii) the continuous motion profiles executed by every agent; therefore, it can infer the evolution of the future cell requirements for every agent. On the other hand, each mobile agent is aware of the applied tesselation and of the further partitioning of the motion area $\mathcal{U}$ that it is induced by it (cf. Figure 1 and the accompanying discussion in Section IV). In particular, every time that the agent is about to cross the boundary of one of the regions defined by this induced partitioning, the agent will experience an "interrupt" that will cause it to signal the corresponding event to the supervisor and get into a waiting mode until the supervisor grants it permission to proceed. The supervisor response to the reception of such a boundary-crossing signal by a mobile agent is structured as follows: First it assesses the admissibility of the requested transition by assessing the admissibility of the resulting RAS state by the deadlock avoidance policy detailed in Section V. If the applied DAP rejects the requested transition, the supervisor archives it as a "blocked request". In the opposite case, the supervisor updates accordingly the maintained resource allocation state and signals its permission to the requesting agent. In this second case, the supervisor also revisits the list of blocked requests, and re-assesses them in the context of the new resource allocation state. The blocked requests must be re-assessed one request at a time, according to a priority scheme that is aligned to the performance criteria of the traffic system. A similar set of procedures will regulate the initiation of a new trip by an agent and the retirement of an agent that has completed its trip.

We want to point out that, in spite of its centralized nature, the control architecture outlined above will be practically implementable for traffic systems that involve a very large number of vehicles due to the discretized / abstracting nature of the applied control logic, and the low (polynomial) computational complexity of the proposed DAP. In fact, it is possible to further reduce the complexity of the computation performed by this policy during the assessment of the admissibility of a requested advancement by any given vehicle, by processing the vehicles in the set $\mathcal{X}$ of the algorithm in Figure 6 in a way that seeks to terminate the advancing vehicle first and returns with an accepting outcome as soon as this vehicle has been removed from set $\mathcal{X}$; the relevant details are rather straightforward and they are omitted due to space limitations.

Other directions that can be investigated in an effort to further alleviate and control the computational and communication complexity of the underlying control function involve the hierarchical and/or decentralized implementation of this function. In general, such distributed implementations of the control function are based on "special structure" in the controlled plant that enables the localization of the underlying decision making process. In the context of the RAS-based paradigm that is proposed in this work, a possibility for decentralized liveness-enforcing supervision is facilitated by well known results from supervisory control theory that specify conditions under which the concepts of deadlock and non-live states become equivalent. In the resulting regime one needs to guard only against transitions to deadlock states, and this test is much more amenable to decentralization than the test for non-liveness. A systematic investigation of these ideas and a preliminary set of results on them can be found in [37].

VIII. Conclusion

This paper has proposed a novel paradigm for establishing safe and live motion in free-ranging multi-vehicle systems. The defining idea of the proposed method is the tesselation of the underlying motion plane in a number of cells of a certain shape and size, and the treatment of these cells as resources that must be acquired by the moving agents for the execution of the corresponding segments of their motion profiles. The paper established that the resulting resource allocation problem can be effectively addressed by leveraging results provided by the burgeoning area of real-time management of resource allocation systems. It also highlighted further research directions that can extend these results and their practical implementation in the considered application context. In fact, we believe that the paper defines a new research domain for the underlying application area, and many of the more open topics highlighted in earlier sections of the manuscript constitute part of our current investigations.

Appendix

Deterministic Finite State Automata

In this appendix, we introduce the concept of *Deterministic Finite State Automaton (DFSA)* and we discuss some of its properties that are important for the developments presented in the main part of the paper. A more extensive discussion on this material can be found in [14].

Definition 5: A deterministic finite state automaton (DFSA) is a 6-tuple $G = (\Sigma, E, \Gamma, f, \sigma_0, \Sigma_M)$, where:

1) Σ is the set of *states*.
2) E is the set of *events*. The occurrence of an event causes a state transition in G.
3) $\Gamma : \Sigma \rightarrow 2^E$ is the *feasible-event function*. Event $e \in E$ is feasible (i.e., can occur) in state $\sigma \in \Sigma$ iff $e \in \Gamma(\sigma)$.
4) $f : \Sigma \times E \rightarrow \Sigma$ is the *transition function*, i.e., a partial function defined for pairs (σ, e) such that $e \in \Gamma(\sigma)$, and with $\sigma' = f(\sigma, e)$ defining the state that results from the occurrence of event e in state σ.
5) $\sigma_0 \in \Sigma$ is the *initial state*.
6) $\Sigma_M \subseteq \Sigma$ is the *set of marked states*.

For the sake of convenience, function f is often extended from domain $\Sigma \times E$ to domain $\Sigma \times E^*$, where E^* denotes the set that contains all the finite-length strings consisting of elements of E and the empty string ϵ. For any given $z \in E^*$ and $e \in E$, the extension of f is defined in the following recursive way:

$$\begin{aligned} f(\sigma, \epsilon) &= \sigma \\ f(\sigma, ze) &= f(f(\sigma, z), e) \end{aligned} \tag{14}$$

We will say that state σ' is reachable from state σ iff there exists string $z \in E^*$ such that $f(\sigma, z) = \sigma'$. The set of all states reachable from σ is denoted by $RS(\sigma)$ and it is called the *reachability set* of σ.

An alternative way to define a DFSA is through its *transition graph* $TG = (RS(\sigma_0), F)$, i.e., a labelled, directed multi-graph such that (i) $RS(\sigma_0)$ is the vertex set of TG, (ii) $F \subseteq \Sigma \times \Sigma \times E$ is the set of edges, where each edge $d = (\sigma, \sigma', e)$ is directed from vertex σ to vertex σ' and labelled by e, and (iii) $f = (\sigma, \sigma', e) \in F$ iff $e \in \Gamma(\sigma)$ and $\sigma' = f(\sigma, e)$.

From the application viewpoint, an important property of a DFSA is the demonstration of *non-blocking* behavior, which is defined as follows:

Definition 6: A DFSA G presents non-blocking behavior (or, more briefly, is non-blocking), iff $\forall \sigma \in RS(\sigma_0)$, $RS(\sigma) \cap \Sigma_M \neq \emptyset$.

In plain terms, a DFSA G is non-blocking if every string that originates at the initial state σ_0 can be extended so that it leads to one of the automaton marked states $\sigma \in \Sigma_M$. Hence, this definition is consistent with the general idea in DFSA-based modeling that marked states indicate the achievement of a "milestone" or a task completion, in the behavior generated by the automaton. In the context of the application considered in this manuscript, non-blocking behavior coincides with the concept of traffic *liveness*, and therefore, these two concepts are used inter-changeably. In a system that is not naturally live, one can consider enforcing this property by making the feasible event function Γ more restrictive. In more technical terms, the adopted supervisor filters out from the set of feasible events at every state, an appropriate subset of *admissible* events; a formal characterization of this idea is as follows:

Definition 7: Given a DFSA $G = (\Sigma, E, \Gamma, f, \sigma_0, \Sigma_M)$, a *restriction* of G is an automaton $G^x = (\Sigma, E, \Gamma^x, f, \sigma_0, \Sigma_M)$, such that for each $\sigma \in \Sigma$, $\Gamma^x(\sigma) \subseteq \Gamma(\sigma)$. If the resulting automaton G^x is live then Γ^x is called a *liveness enforcing supervisor* or an *LES*.

For the purposes of this paper, we also recall the concept of the parallel composition of two or more automata.

Definition 8: The *parallel composition* of automata G_1 and G_2 is the automaton $G_1 \parallel G_2 = (\Sigma_1 \times \Sigma_2, E_1 \cup E_2, \Gamma_{1\parallel 2}, f, (\sigma_{01}, \sigma_{02}), \Sigma_{M_1} \times \Sigma_{M_2})$ such that:

1) $\Gamma_{1\parallel 2}(\sigma_1, \sigma_2) = (\Gamma_1(\sigma_1) \cap \Gamma_2(\sigma_2)) \cup (\Gamma_1(\sigma_1) \setminus E_2) \cup (\Gamma_2(\sigma_2) \setminus E_1)$

2) $f((\sigma_1, \sigma_2), e) =$
$$\begin{cases} (f_1(\sigma_1, e), f_2(\sigma_2, e)) & \text{if } e \in \Gamma_1(\sigma_1) \cap \Gamma_2(\sigma_2) \\ (f_1(\sigma_1, e), \sigma_2) & \text{if } e \in \Gamma_1(\sigma_1) \setminus E_2 \\ (\sigma_1, f_2(\sigma_2, e)) & \text{if } e \in \Gamma_2(\sigma_2) \setminus E_1 \\ \text{undefined} & \text{otherwise} \end{cases}$$

In the parallel composition, feasible events that are "private" to each constituent automaton can occur independently in the automaton, while events shared by the two automata must occur in both of them in a synchronized manner. Finally, we also notice that, as an operator, parallel composition is associative and commutative; that is, $G_1 \parallel (G_2 \parallel G_3) = (G_1 \parallel G_2) \parallel G_3$ and $G_1 \parallel G_2 = G_2 \parallel G_1$.

REFERENCES

[1] L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli, "Decentralized cooperative policy for conflict resolution in multivehicle systems," *IEEE Trans. on Robotics*, vol. 23, pp. 1170–1183, 2007.

[2] A. Bicchi and L. Pallottino, "On optimal cooperative conflict resolution of air traffic management systems," *IEEE Trans. on Intelligent Transportation Systems*, vol. 1, pp. 221–232, 2000.

[3] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: a study in multiagent hybrid systems," *IEEE Trans. on Automatic Control*, vol. 43, pp. 509–521, 1998.

[4] J. Lygeros, D. N. Godbole, and S. Sastry, "Verified hybrid controllers for automated vehicles," *IEEE Trans. on Automatic Control*, vol. 43, pp. 522–539, 1998.

[5] G. Inalhan, D. M. Stipanovic, and C. J. Tomlin, "Decentralized optimization, with application to multiple aircraft coordination," in *Proc. of CDC'02*. IEEE, 2002, pp. 1147–1155.

[6] S. M. La Valle and S. A. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Trans. on Robotics & Automation*, vol. 14, pp. 912–925, 1998.

[7] D. V. Dimarogonas, S. G. Loizou, K. J. Kyriakopoulos, and M. M. Zavlanos, "A feedback stabilization and collision avoidance scheme for multiple independent non-poin agents," *Automatica*, vol. 42, pp. 229–243, 2006.

[8] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Trans. on Intelligent Transportation Systems*, vol. 1, pp. 179–189, 2000.

[9] Y. Lee, K. Gupta, and S. Payandeh, "Motion planning of multiple agents in virtual environments using coordination graphs," in *IEEE Int. Conf. Robotics and Automation, ICRA'05*, 2005, pp. 378–383.

[10] J. Wang and S. Premvuti, "Distributed traffic regulation and control for multiple autonomous mobile robots operating in discrete space," in *IEEE Int. Conf. Robotics and Automation, ICRA'95*, vol. 2, 1995, pp. 1619 – 1624.

[11] M. Jager and B. Nebel, "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems IROS'01*, vol. 3, 2001, pp. 1213 – 1219.

[12] F. Noreils, "Integrating multirobot coordination in a mobile-robot control system," in *IEEE Int. Workshop Intelligent Robots and Systems*, 1990, pp. 43–49.

[13] Y. Liu, S. Kuroda, T. Naniwa, H. Noborio, and S. Arimoto, "A practical algorithm for planning collision-free coordinated motion of multiple mobile robots," in *IEEE Int. Conf. Robot. Automat.*, 1989, pp. 1427–1432.

[14] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Boston, MA: Kluwer Academic Pub., 1999.

[15] E. W. Dijkstra, "Cooperating sequential processes," Technological University, Eindhoven, Netherlands, Tech. Rep., 1965.

[16] E. G. Coffman, M. J. Elphick, and A. Shoshani, "System deadlocks," *Computing Surveys*, vol. 3, pp. 67–78, 1971.

[17] R. D. Holt, "Some deadlock properties of computer systems," *ACM Computing Surveys*, vol. 4, pp. 179–196, 1972.

[18] Z. A. Banaszak and B. H. Krogh, "Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows," *IEEE Trans. on Robotics and Automation*, vol. 6, pp. 724–734, 1990.

[19] J. Ezpeleta, J. M. Colom, and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. on R&A*, vol. 11, pp. 173–184, 1995.

[20] S. A. Reveliotis and P. M. Ferreira, "Deadlock avoidance policies for automated manufacturing cells," *IEEE Trans. on Robotics & Automation*, vol. 12, pp. 845–857, 1996.

[21] M. P. Fanti, B. Maione, S. Mascolo, and B. Turchiano, "Event-based feedback control for deadlock avoidance in flexible production systems," *IEEE Trans. on Robotics and Automation*, vol. 13, pp. 347–363, 1997.

[22] W. Van der Aalst and K. Van Hee, *Workflow Management: Models, Methods and Systems*. Cambridge, MA: The MIT Press, 2002.

[23] J. Park, "A deadlock and livelock free protocol for decentralized internet resource coallocation," *IEEE Trans. on Systems, Man and Cybernetics, Part A*, vol. 34, pp. 123–131, 2004.

[24] S. A. Reveliotis, *Real-time Management of Resource Allocation Systems: A Discrete Event Systems Approach*. NY, NY: Springer, 2005.

[25] ——, "Conflict resolution in AGV systems," *IIE Trans.*, vol. 32(7), pp. 647–659, 2000.

[26] M. P. Fanti, "Event-based controller to avoid deadlock and collisions in zone-control AGVS," *Int. J. of Production Res.*, vol. 40, no. 6, pp. 1453–1478, 2002.

[27] E. Roszkowska and S. Reveliotis, "On the liveness of guidepath-based, zoned-controlled, dynamically routed, closed traffic systems," *IEEE Trans. on Automatic Control*, vol. 53, pp. 1689–1695, 2008.

[28] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot motion planning and control in polygonal environments," *IEEE Trans. on Robotics*, vol. 21, pp. 864–874, 2005.

[29] D. C. Conner, H. Choset, and A. A. Rizzi, "Flow-through policies for hybrid controller synthesis applied to fully actuated systems," *IEEE Trans. on Robotics*, vol. 25, pp. 136–146, 2009.

[30] T. Araki, Y. Sugiyama, and T. Kasami, "Complexity of the deadlock avoidance problem," in *2nd IBM Symp. on Mathematical Foundations of Computer Science*. IBM, 1977, pp. 229–257.

[31] E. Roszkowska, "Provably correct closed-loop control for multiple mobile robot systems," in *Proceedings of ICRA'05*. IEEE, 2005, pp. 2810–2815.

[32] ——, "High-level motion control for workspace sharing mobile robots," in *Robot Motion and Control*. Springer, LNCIS vol. 360, 2007, pp. 427 – 435.

[33] A. Kobetski, "Optimal coordination of flexible manufacturing systems with automatic generation of collision and deadlock-free working schedules," Ph.D. dissertation, Chalmers University of Technology, 2008.

[34] S. A. Reveliotis and E. Roszkowska, "Conflict resolution in multi-vehicle systems: a resource allocation paradigm," in *Proceedings of IEEE CASE 2008*. IEEE, 2008, pp. 115–121.

[35] W. M. Wonham, "Supervisory control of discrete event systems," Electrical & Computer Eng., University of Toronto, Tech. Rep. ECE 1636F / 1637S 2006-07, 2006.

[36] S. A. Reveliotis and E. Roszkowska, "On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems," *IEEE Trans. on Automatic Control*, vol. 55, pp. 1646–1651, 2010.

[37] E. Roszkowska and S. Reveliotis, "A maximally permissive distributed protocol for motion coordination in free-range vehicular systems," in *IFAC World Congress 2011*. IFAC (submitted), 2011, pp. –.