A Distributed Protocol for Motion Coordination in Free-Range Vehicular Systems

Elzbieta Roszkowska
Institute of Computer Engineering, Control, and Robotics
Wroclaw University of Technology
ekr@pwr.wroc.pl

Spyros Reveliotis
School of Industrial & Systems Engineering
Georgia Institute of Technology
spyros@isye.gatech.edu

Abstract

This work extends our research on motion coordination of free-range vehicular systems based on concepts and results borrowed from resource allocation systems (RAS) theory, to vehicular systems with limited communication range among the vehicles. Similar to the earlier work, the employed model assumes the tessellation of the motion plane into cells, which are allocated to the traveling vehicles in a controlled manner that ensures collision-free and live motion. On the other hand, the limited communication range of the vehicles implies that full synchronization of their access to the considered cells is not possible any more, and yields new challenges for the deployed supervisory control policies. To enable the development of supervisory policies capable of providing the necessary partial synchronization of the cell allocation, we modify the structure of the adopted tessellation by allowing the concurrent occupation of a cell by up to two vehicles at a time, instead of only one, that was assumed earlier. This modification renders polynomially computable the relevant maximally permissive cell allocation policy, and it enables the implementation of this policy in the form of a distributed protocol that is feasible in the context of the communication constraints that are considered in this work.

Keywords: Intelligent vehicular systems, motion liveness, motion safety, multi-agent systems, supervisory control, decentralized control, deadlock avoidance.

1 Introduction

The problem addressed in this paper concerns the traffic coordination for a fleet of vehicles that are moving concurrently in a finite area, so that (i) there are no collisions among these vehicles, and (ii) each vehicle reaches its destination in finite time. The second of these two requirements is of a teleological nature, and it defines a notion of "liveness" for the vehicle motion. On the other hand, the first requirement is clearly motivated by safety concerns. Yet, in this work we will differentiate the terms "safe" and "safety" from the notions of collision freedom and avoidance, since, by following some standard literature, we will need to use these two terms for concepts and properties that relate more directly to the notion of liveness discussed above.

It can be generally argued that the establishment of collision-free and live vehicle motion is a problem well-recognized in the existing literature, and it has been studied for a number of traffic systems. In the prevailing approaches, each vehicle is abstracted to a "mobile agent", and its dynamics is described with models whose state evolves in continuous time. Some representative works of this line of research can be found in [18, 3, 25, 14, 11, 13, 8] while a higher-level but more comprehensive description of the pursued methods can be found in [12]. However, as remarked in [18], while many of these works will guarantee collision freedom, very few of them have actually considered the issue of motion liveness. And the few works that have addressed the problem of liveness might suffer either from an inability to provide formal liveness guarantees, or from scalability problems, due to the continuous-time nature of the underlying modeling.

Hence, more recently, and in an effort to address these computational and analytical challenges, the relevant research community has pursued the analysis and control of the considered traffic systems through representations borrowed from the burgeoning field of hybrid dynamical systems. Under this paradigm, the overall motion control problem is "decomposed" to a number of subproblems that concern the system operation under different configurations – or "modalities" – while a higher-level coordinator ensures that the system transitions among these modalities generate a global behavior that is in line with the posed specifications and take place in a stable and seamless manner. Typically, the control subproblem that pertains to each of the aforementioned modalities is of a more "local", and therefore, simpler nature. On the other hand, the more "global" problem of the higher-level coordination is dealt through pertinent abstractions that retain the information that is essential for the formal analysis and verification of the sought properties, and for the synthesis of the necessary control logic, while concealing all the operational details that might be irrelevant to that level of decision making. In this way, the underlying problem complexity is effectively addressed in a "divide & conquer" manner. Some indicative examples of such lines of work are those presented in [2, 5], while a more comprehensive exposition of the state-of-art of hybrid-system-based methods for the design, formal verification and the control of many contemporary applications, including vehicular systems

similar to those considered in this work, can be found in [1].

In line with the developments discussed in the previous paragraph, in [20] we proposed a hybrid control scheme that establishes collision-free and live motion of free-range vehicular systems through the tesselation of the motion area into a number of cells. These cells are treated as "resources" that must be acquired by the traveling vehicles in order to execute their designated trips. Collision freedom of the vehicle motion is ensured by preventing the simultaneous occupation of any given cell by more than one vehicle. On the other hand, establishing the liveness of the vehicle motion necessitates the coordination of the cell allocation to the contesting vehicles, in a way that ensures that all vehicles are able to access the requested cells and complete their motion. Hence, under the hybrid control scheme of [20], the original (continuous) paths that specify the target motion for each vehicle are segmented into a number of sub-paths, with each sub-path being defined by a particular cell occupation pattern by the traveling vehicle. The faithful and successful execution of any given sub-path by a traveling vehicle is the task of a local controller that is possessed by the vehicle itself and drives its continuous motion. On the other hand, in [20], the control function that coordinates the cell allocation to the contesting vehicles is supported by an external controller that communicates asynchronously with the traveling vehicles, receiving requests by them to advance on their next sub-path and granting the corresponding permissions. The synthesis of the necessary control logic for this central coordinator leverages concepts and results from the area of liveness-enforcing supervision of resource allocation systems (RAS) [19]. More specifically, in [20], the resulting resource allocation problem is solved by an adaptation of Banker's algorithm [7] that establishes an efficient trade-off between operational flexibility for the underlying traffic and computational tractability for the necessary control logic.

In this paper, we seek to adapt the control paradigm of [20] so that it applies to free-range vehicular systems with limited communication capabilities. In the operational regime considered in this paper, vehicles can communicate only locally, within the range of a few cells centered around their current location, and therefore, the assumption of a central controller that monitors and controls the entire cell allocation is not plausible any more. Hence, in this new regime, vehicles must coordinate the allocation of the contested cells at a more local basis, while eliciting from the surrounding vehicles additional information that might be necessary in order to ensure collision-free motion and to assess the liveness retention of any contemplated allocation. This information must be captured by means of a communication protocol that propagates a series of inquiries to the vehicles occupying surrounding cells that are critical for assessing the liveness retention of the attempted allocation and the collision freedom of the resultant motion, and returns the obtained responses to the inquiring vehicle. The design and validation of such a communication protocol adds an entirely new level of complexity to the control problem under consideration, and it is the main focus of this work.

More specifically, in the presented work, the design and deployment of the communication protocol outlined in the previous paragraph is supported through the redefinition of the tesselation and the induced resource allocation scheme that were employed in [20]. Some key elements of the new resource allocation model are defined by the requirements that (i) a cell must be specified large enough to support collision-free travel of up to two vehicles in it, and that (ii) the generated (continuous) paths must be such that the vehicle does not occupy more than two cells at a time, except for some transient phases that correspond to the initiation and termination of the vehicle motion. In the next section, we shall show that under the aforestated conditions, the entire trip of any given vehicle can be decomposed to a number of sub-paths that correspond to the traversal of each of the consecutive cells that define the vehicle route from its origin to its destination, and that the local cell traffic generated by these sub-paths is effectively controllable for collision avoidance and liveness by results already available in the literature. Subsequently, the rest of the paper will focus on the higher-level problem of coordinating the cell access among the contesting vehicles, introducing formally the new, distributed control scheme, establishing its capabilities, and detailing the various functions and protocols that are necessary for its effective deployment. *To the best of our knowledge, this is the first set of results to provide a (non-trivial) complete, provably correct solution to the combined problem of collision avoidance and liveness-enforcement in free-range vehicular systems, that furthermore remains scalable with respect to the underlying fleet size and admits a natural distributed implementation across the traveling vehicles.*

The detailed organization of the remaining part of this document is as follows: Section 2 introduces the free-range vehicular systems of interest in this work, describes the proposed hybrid control framework by defining the various control sub-problems to be addressed in it, and also it outlines a set of solutions for the "lower-level / local" control subproblems of this hybrid framework by building upon a set of results in the current literature. Section 3 characterizes formally the cell allocation function taking place in the considered traffic systems as a new RAS model, to be called the FREE-RANGE*-2-RAS model, and it employs this characterization towards a formal definition of the notions of liveness and liveness enforcement for these systems. It also presents a series of analytical results that facilitate the development of a distributed liveness-enforcing supervisor (LES). The detailed design of such a supervisor is considered in Section 4. Special emphasis is placed on the communication protocol that is necessary for decentralized implementation. This section also establishes the correctness of the proposed control scheme with respect to the posed problem, and it demonstrates the dynamics of this scheme and its supporting communication protocol through illustrative examples. Finally, Section 5 concludes the paper and provides some directions for future work. Closing this introductory section, we notice, for completeness, that a preliminary version of the presented results, without the detailed specification of the proposed protocol and many of the proofs of

Sections 3 and 4, was presented in [23].

2 The considered vehicular system and the overall structure of the proposed control scheme

We consider a set of autonomous vehicles that move in a finite planar area $\mathcal{A} \subset \mathbf{R}^2$. Each vehicle is represented by a disk of radius ρ, and its motion is specified and controlled by specifying and controlling the path to be followed by the center of the aforementioned disk; the latter is typically defined in some parametric form: $x^c = x^c(t)$, $y^c = y^c(t)$, $t \in [0, T]$. For greater specificity, in the subsequent developments it is further assumed that, in terms of their motion capabilities, the vehicles act as "unicycles", possibly observing additional nonholonomic constraints that are frequently associated with the motion of this entity [17]; however, our results extend to other motion dynamics, like the "rear-drive car" dynamics analyzed in [17]. We also assume that the vehicles are familiar with their operational environment and they have the capability to localize themselves in it, e.g., through a combination of measurements that are obtained by an inertial measurement unit (IMU) and a GPS unit [9]. Finally, vehicles can communicate directly with each other, but only within a limited range. This range must support the communication requirements of the proposed control framework, and a lower bound for it will be determined by the detailed specification and analysis of the functionality of this framework.

From a higher, operational standpoint, the tasks / missions assigned to each vehicle can be perceived as trip requests between an origin and a destination point that are located in motion area $\mathcal{A}$. It is assumed that the vehicles stay off the system before they start their travel, and that they are retired from the system upon reaching their destination. However, during their concurrent motion in the system, the vehicles share the available space, and in order to avoid collisions, their motion profiles must be determined through some coordinating control logic. Similar to [20], this coordination will be achieved through a hybrid control scheme that is based on the *tessellation* of the motion plane into a number of areas, called *"cells"*.

More specifically, in the proposed control scheme, the motion area is tesselated through a grid of horizontal and vertical lines centered at the origin of the coordinate system (x, y). The resulting set of cells is 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 (finite) area $\mathcal{A}$, that supports the vehicle motion. For reasons that will be revealed in the following, we request that the cell edges are of a length $d \geq 4\rho$. Finally, given a point $(x, y) \in \mathcal{A}$ and a cell $w[i, j]$, we define

$$(x, y) \in w[i, j] \Longleftrightarrow (i-1) \cdot d \leq x \leq i \cdot d \ \wedge \ (j-1) \cdot d \leq y \leq j \cdot d \tag{1}$$

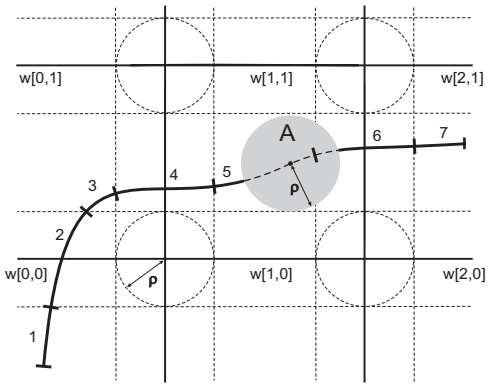


Figure 1: The tesselation of the motion area considered in this work, the induced mapping $\mathcal{W}(\cdot)$, and the resulting segmentation of the considered vehicle routes into stages (i.e., maximal segments of constant cell occupation).

We shall say that a vehicle (with its disk) centered at (x^c, y^c) *occupies* cell $w[i, j]$ if and only if (*iff*) there exists $(x, y) \in w[i, j]$ with $||(x, y) - (x^c, y^c)|| \leq \rho$, where $||\cdot||$ denotes the Euclidean norm. This definition induces a further mapping, $\mathcal{W}$, from the motion area, $\mathcal{A}$, to the powerset of W, 2^W, that maps to any point $(x, y) \in \mathcal{A}$, the cell subset $\mathcal{W}(x, y) \in 2^W$ consisting of the cells occupied by a vehicle centered at (x, y). A graphical illustration of this mapping $\mathcal{W}$ is given in Figure 1. In this figure, the adopted tessellation is defined by the grid of the solid horizontal and vertical lines, and the vehicle is depicted by the grey disk in it. It is not hard to notice that a vehicle can occupy from one cell to four neighboring cells at a time. The number of cells occupied by a vehicle is effectively determined by the relative positioning of its center point (x^c, y^c) with respect to another partitioning of the motion plane, that is induced by the original tessellation scheme and the vehicle geometry. In Figure 1, this induced partitioning is defined by the depicted dashed lines.[1]

As discussed in the introductory section, the proposed control paradigm tries to establish collision-free and live motion for the system vehicles, by controlling the vehicle access to the various cells that are established by the tesselation of the motion area described in the previous paragraphs. In particular, it is enforced that a cell cannot be occupied by any more than two vehicles at any point in time. On the other hand, vehicle trips are specified as a sequence of cells to be traversed by the traveling vehicle while advancing from its origin to its destination. In face of the aforementioned restriction for the cell occupancy, the cells defining a vehicle trip can be perceived as "resources" that must be acquired and released by the vehicle for the execution of the corresponding part of its trip. In the case of the example path depicted in Figure 1, the cell sequence that defines the corresponding trip has the following

[1] An analytical characterization of this partitioning can be found in [20]; we forego the relevant details due to space limitations.

form: $w[0,0], w[0,1], w[1,1], w[2,1]$.

The specification of the vehicle trips by a sequence of cells, as discussed in the previous paragraph, has a number of fundamental implications for the dynamics of the generated traffic and for the control function that must be imposed upon this traffic in order to ensure collision freedom and liveness. For a start, the reader should notice that the specification of the vehicle trips by a sequence of cells further implies that, with the exception of the motion-initiating and terminating phases, a vehicle cannot occupy more than two cells at a time during its entire trip. In particular, the overall trip decomposes into a series of stages where the vehicle initially idles in its current cell trying to secure access to the next cell in its route, and as soon as this access is secured, it crosses the boundary between the two cells in a way that it does not occupy any further cell during this crossing, and subsequently it advances within this new cell up to a point where it will have to wait until it gets access to the subsequent cell. The boundaries that define and restrict the motion are induced by the mapping $\mathcal{W}(\cdot)$ that was discussed above. In Figure 1, the observation of these boundaries by the depicted route is manifested by the fact that this route stays clear of the square regions around the cell corners that are defined by: $(id-\rho, jd+\rho)$, $(id+\rho, jd+\rho)$, $(id-\rho, jd-\rho)$, $(id-\rho, jd+\rho)$, for $i \in \{-\underline{I}, \ldots, -1, 0, 1, \ldots, \overline{I}\}$, $j \in \{-\underline{J}, \ldots, -1, 0, 1, \ldots, \overline{J}\}$. On the other hand, since we want to retain the vehicle capability to access any point in the motion area, we allow the origin and the destination of any given trip to lie anywhere in this area, and we augment the vehicle trips with two additional stages that correspond to the initiation and termination of the vehicle motion. These stages are handled as "special events" by the proposed control framework, in a way that is described in the next paragraph.

The stage-based decomposition of the vehicle trips discussed above, defines also the path-planning and the corresponding motion control problem that must be addressed by each vehicle as it executes any of these stages. As explained above, with the potential exception of the initiation and termination stages, any other stage of the vehicle trip corresponds to its advancement from a "boundary" location in its current cell to another "boundary" location in its next cell, where it will have to negotiate its access to the subsequent cell. Furthermore, this advancement must be performed in a way that (i) restricts the vehicle cell occupancy to its current and the next acquired cell only, and (ii) avoids collision with any other vehicles that might be present in these two cells. We also remind the reader that the spatial boundaries that enforce the above restrictions are defined with respect to the partitioning of the motion area that is induced by mapping $\mathcal{W}(\cdot)$. Then, it can easily be checked that under the assumption that the cell dimension d is at least equal to 4ρ, there exist feasible paths able to support the simultaneous traversal of a cell by two vehicles moving between any pair of the cell edges. Furthermore, setting $d \geq 4\rho$ (i) enables two vehicles occupying any given cell to rest side by side on the same cell boundary while contesting their access to the same next cell, and (ii) ensures that two vehicles advancing

between a pair of cells, but in opposite directions, can cross each other. It is also easy to see that the paths to be followed by the traveling vehicles during their traversal of any given cell can be defined smooth enough so that they can be approximated to any degree of accuracy by the assumed vehicle dynamics; in particular, such an accurate path following can be achieved by appropriately controlling the vehicle speed ("slowing down" these vehicles) [16]. One can envision a variety of possible ways that these paths can be specified and coordinated among the different vehicles, but space limitations do not allow an expansive treatment of this topic in this work. Furthermore, under the assumptions for the vehicle dynamics and their localization capabilities that were discussed in the opening paragraph of this section, and the additional assumption that any pair of vehicles can communicate directly when located in the vicinity of two neighboring cells, the aforestated control problem of the vehicle navigation across any given cell can be addressed by results in the existing literature; a particular methodology for dynamic path planning and collision avoidance that fits perfectly the above problem specification is that presented in [4]. This method allows a set of vehicles with "unicycle"-type of dynamics to reach their destination while avoiding collisions among themselves and observing additional constraints that are imposed upon their motion, by combining the steering method of dipolar navigation functions [24] with model predictive control [15]. In fact, the same method can address the local coordination problem that corresponds to the motion initiation or termination phase of any given vehicle. The higher-level objective of such a stage will be to enable the corresponding vehicle that initiates or terminates its motion, to reach its target location within its current cell, while any neighboring vehicles must reposition themselves within their current cells so that they "clear" the way for the former vehicle.[2]

Having described the vehicle dynamics and the control problem that coordinates the vehicle motion during any single stage of their overall trip, in the rest of this document we shift attention to the higher-level problem of coordinating the vehicle transition across their different stages, and we detail the necessary protocols that will effect this coordination. We start in the next section, by abstracting and analyzing this coordination problem as a resource allocation problem, using concepts and results from the resource allocation system theory [19]. The theoretical developments of the next section will provide the formal basis for the specification and the analysis of the sought protocols in Section 4.

3 The FREE-RANGE*-RAS and an analysis of the corresponding "state safety" problem

In this section, we introduce the FREE-RANGE*-2-RAS, that will formally model the resource allocation dynamics taking place in the considered traffic system, and will also provide the

[2]The requirement that $d \geq 4\rho$ is sufficient for ensuring the feasibility of this maneuvering.

basis for the specification of the proposed control logic and communication protocol. We start, however, with the introduction of the more general RAS class of FREE-RANGE*-RAS. This class is formally defined by the quadruple $\Phi = (\mathcal{R}, C, \mathcal{P}, \mathcal{D})$, where: 1) $\mathcal{R}$ is the set of the system *"resources"*, and corresponds to the set of cells defined by the imposed tessellation. 2) $C : \mathcal{R} \rightarrow \mathbb{Z}^+$ is the *"resource capacity function"*, that determines the maximal number of vehicles that can occupy a particular cell at a time. 3) $\mathcal{P} = \{P_1, \ldots, P_n\}$ is the set of *"processes"* that abstract the transitions of the system vehicles through the cells that define their routes. In particular, each process P_i, $i = 1, \ldots, n$, consists of $\Xi_{i1}, \Xi_{i2}, \ldots, \Xi_{il(i)}$ consecutive *"processing stages"*, that represent the motion of vehicle A_i while traversing each of its route cells. 4) $D : \Xi = \{\Xi_{ij} \mid i = 1, \ldots, n; j = 1, \ldots, l(i)\} \rightarrow \mathcal{R}$ is the *"resource allocation function"* specifying the cells that are occupied by the vehicles at the various stages of their motion process. For the sake of simplicity, in the sequel we shall use interchangeably the notation D_{ij} and $D(\Xi_{ij})$.

The *FREE-RANGE*-k-RAS*, where $k \in \mathbb{Z}^+$, is obtained from the FREE-RANGE*-RAS with the additional restriction that $C(R) = k$, $\forall k$. Furthermore, the behavioral dynamics of FREE-RANGE*-RAS can be represented by a *Deterministic Finite State Automaton (DFSA)* (c.f. [10]), that is defined next.

Definition 1 *The DFSA $G(\Phi) = (\Sigma, E, \Gamma, f, \sigma_0, \Sigma_M)$ abstracting the feasible dynamics of a FREE-RANGE*-RAS $\Phi = (\mathcal{R}, \mathcal{C}, \mathcal{P}, D)$ 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: (a) $\forall i \in \{1, \ldots, n\}$, $0 \leq \sigma_i \leq l(i) + 1$, and (b) $\forall R \in \mathcal{R}$, $a(\sigma, R) = |\{\sigma_i \mid D_{i,\sigma_i} = R\}| \leq C(R)$. Each component σ_i of σ indicates the current stage of process P_i. In particular, $\sigma_i = 0$ indicates that process P_i has not been initiated yet, while $\sigma_i = l(i) + 1$ indicates that process P_i has been completed (and retired from the motion plane). For each $R \in \mathcal{R}$, $a(\sigma, R)$ indicates the number of units of resource R that are allocated in state σ (or, equivalently, the number of processes that hold R in state σ).*

2. *The* event set $E = \{e_{ij} \mid i = 1, \ldots, n; j = 1, \ldots, l(i) + 1\}$, *where for every $i = 1, \ldots, n$: (a) the event e_{i1} represents the initiation of process P_i by the allocation of the resource D_{i1}; (b) the events e_{ij}, $j = 2, \ldots, l(i)$, represent the advancement of process P_i from processing stage $\Xi_{i,j-1}$ to processing stage Ξ_{ij} through the corresponding adjustment of its resource allocation; and (c) the event $e_{i,l(i)+1}$ represents the termination of process P_i and the release of the currently held resource $D_{i,l(i)}$.*

3. *For each pair (σ, e_{ij}), the* state transition function f *returns the new state $\sigma' = f(\sigma, e_{ij})$, whose components σ'_k, $k = 1, \ldots, n$, are given by*

$$\sigma'_k = \begin{cases} \sigma_k + 1 & \text{if } k = i \ \wedge \ 1 \leq j \leq l(i) + 1 \ \wedge \ \sigma_k = j - 1 \\ \sigma_k & \text{otherwise} \end{cases}$$

4. *Function f is defined for a pair (σ, e)* iff $e \in \Gamma(\sigma)$*, where the* set of active events $\Gamma(\sigma)$ *is defined by* $\Gamma(\sigma) \equiv \{e \in E : \sigma' = f(\sigma, e) \in \Sigma\}$.

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* $\{\sigma_M = [l(1) + 1, \ldots, l(n) + 1]\}$*, and it expresses the requirement for complete process runs.*

The reader should notice that the combination of items (1) and (4) of Definition 1 ensures that, in the considered RAS dynamics, no cell is over-allocated with respect to its capacity. In the sequel, we will use the expression "state σ' is (resp., is not) reachable from state σ" to describe the fact that there exists (resp., there does not exist) a feasible sequence of events that drives the automaton from state σ to state σ'. In particular, a state $\sigma \in \Sigma$ that is reachable from the initial state σ_0 will be simply characterized as *reachable*. The next definition introduces some fundamental concepts that are necessary for reasoning about the liveness of the considered traffic systems:

Definition 2 *Consider a FREE-RANGE*-RAS specified by a quadruple $\Phi = (\mathcal{R}, C, \mathcal{P}, D)$, and a state $\sigma \in \Sigma$ of the corresponding DFSA $G(\Phi)$.*

1. *A process instance executing stage Ξ_{ij} is* dead *in state $\sigma_d \in \Sigma$* iff *function $f(\sigma, e_{i,j+1})$ is not defined for any state σ reachable from σ_d, i.e., the process can never advance to its next stage.*

2. *State σ is characterized as* safe iff *the marked state σ_M is reachable from state σ.*[3]

3. *The* RAS state safety problem *is the decision problem that, upon input Φ and σ, addresses the question of whether or not state σ of RAS Φ is safe.*

Clearly, a process instance will never become dead if it runs alone in the system. Its progress can only be impaired by the presence of other processes, and the direct reason for that is the formation of a *deadlock*, i.e., the development of a set of processes such that each of them in order to advance to its next stage requests a resource unit that is currently held by some other process in the set. In this paper, rather than detecting deadlocks, we will focus on testing whether or not a state transition caused by the advancement of a process to its next state renders this process dead. This will allow us to develop a distributed liveness-enforcing supervisory control policy

[3] As noticed in the introductory section, in the context of this work "(state) safety" is a concept that pertains to the liveness of the resource allocation function; this term has been employed extensively in the relevant RAS literature, and therefore, we decided to maintain it in spite of the fact that in the context of the coordination of vehicle systems, safety typically implies collision avoidance.

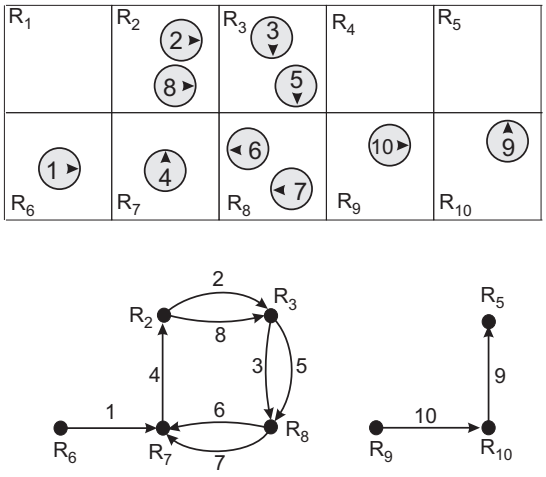


Figure 2: Graphical representation of an example state in FREE-RANGE*-2-RAS. The upper part of the figure depicts the distribution of the vehicles as well as the direction of their motion in state σ, and the lower part shows the corresponding graph $F(\sigma)$.

for vehicular systems with limited communication range. The proposed approach is based on [22] and requires depicting state σ in the form of a *resource allocation graph* $F(\sigma) = (V, H)$.

Definition 3 *Consider the DFSA $G(\Phi)$ that models the resource allocation dynamics of a FREE-RANGE*-RAS Φ. The* resource allocation graph *representing a state σ of $G(\Phi)$ is a graph $F(\sigma) = (V, H)$ such that:*

- *The set of vertices is defined by the extended set of resources $V = \mathcal{R} \cup \{R_\infty\}$, where R_∞ is a dummy resource of infinite capacity.*
- *The set of edges H is defined by the set of active processes $\mathcal{P}'$, i.e., the set of processes $P_i \in \mathcal{P}$ that in state σ execute a stage Ξ_{ij} s.t. $j \in \{1, \ldots, l(i)\}$. In particular, the edge (corresponding to process) P_i goes from vertex $R \in \mathcal{R}$ to vertex $R' \in \mathcal{R}$* iff*, at state σ, process P_i has been allocated resource R and for its next stage it requires resource R'. Edge P_i goes from vertex $R \in \mathcal{R}$ to vertex R_∞ if, at state σ, process P_i executes its last stage,*[4] *$\Xi_{i,l(i)}$.*

An example of the state representation in the form of a resource allocation graph is given in Figure 2. Also, in the sequel, for any resource $R \in \mathcal{R}$, we shall use the notation $Suc(R; \sigma)$ to denote the (not necessarily immediate) successors of R in graph $F(\sigma)$. Then, using the above introduced concepts, we can establish the following property for FREE-RANGE*-RAS.

[4]Here, as well as in the sequel, by 'last stage' of process P_i we understand $\Xi_{i,l(i)}$, i.e., the last active stage of P_i.

Property 1 *Consider a FREE-RANGE*-RAS specified by the quadruple* $\Phi =< \mathcal{R}, C, \mathcal{P}, D >$, *a state* $\sigma \in \Sigma$ *of the corresponding DFSA* $G(\Phi)$, *and its graphical representation* $F(\sigma) = (V, H)$. *Then, a process* $P_i \in H$ *executing stage* Ξ_{ij} *is not dead* iff *in graph* $F(\sigma)$ *there exists a path* $p = \langle R^1, R^2, \ldots, R^{l_p} \rangle$, $l_p > 1$, *from resource* $R^1 = D(\Xi_{ij})$ *to a resource* $R^{l_p} \in \mathcal{R} \cup \{R_\infty\}$ *that has a free unit of capacity, and edge* (R^1, R^2) *corresponds to the advancement of process* P_i *from stage* Ξ_{ij} *to its next stage.*

Proof: To prove Property 1, assume first that the specified path p exists. Then, there exists a feasible sequence of events $s = e^{l_{p-1}}, \ldots, e^1$ such that event e^k, $k = 1, \ldots, l_{p-1}$ corresponds to a transition of a process instance executing stage Ξ^k with $D(\Xi^k) = R^k$ to its next stage. Since $\Xi^1 = \Xi_{ij}$, the considered process instance is not dead. To prove the reverse implication, assume that the required path p does not exist; i.e., each resource that can be reached by a path p starting with edge $(D_{ij}, D_{i,j+1})$ is fully allocated. The latter implies that the dummy resource R_∞ is not an element of any such path p. Hence, no process instance executing any stage on a resource that is reachable by the considered paths p can ever leave the system or advance to its next stage. Consequently, there exists no state σ' reachable from σ that enables event e_{ij}, and so process P_i is dead in σ. ■

The next property is a technical result that will be very useful in many of the subsequent developments.

Property 2 *Consider the DFSA* $G(\Phi)$ *of a FREE-RANGE*-RAS* Φ, *and a state* $\sigma \in \Sigma$ *such that (i) no process* $P \in \mathcal{P}$ *is dead in* σ, *and (ii) for some process* P_i, *the next state* $\sigma' = f(\sigma, e_{i,j+1})$ *is defined. Then, if process* P_i *is not dead in state* σ', *no other process is dead in that state.*

Proof: To prove Property 2, let $R' = D_{ij}$ denote the resource currently allocated to process P_i; for $j = 0$, $R' = R_\infty$. Let R^* denote the resource required for the next stage of P_i, i.e., $R^* = D_{i,j+1}$ if $j < l(i)$, and $R^* = R_\infty$ otherwise. Moreover, if $j < l(i) - 1$, define $R'' = D_{i,j+2}$, and let $R'' = R_\infty$ otherwise.

Consider any process $P_k \neq P_i$ and assume that P_k is at its r-th stage. Since P_k is not dead in state σ, then, by Property 1, in graph $F(\sigma)$ there exists a successor R of D_{kr} that has a free unit. Assume now that in state σ' process P_k is dead. This requires that in state σ' either a) R is no more a successor of D_{kr}, or b) R has no more a free unit.

The assumptions of Property 2 when combined with the definition of R', R^*, and R'', imply that the only difference in the structure of graph $F(\sigma')$ with respect to graph $F(\sigma)$ is that in $F(\sigma')$ edge (R', R^*) is replaced by edge (R^*, R''). But then, case (a) can only happen if, in $F(\sigma)$, each path from D_{kr} to R includes R'. Hence, in state σ', R' is still a successor of D_{kr}, and furthermore, it has a free unit of capacity (the unit released by the advancement of P_i). But then, process P_k cannot be dead in σ'.

On the other hand, since resource allocation state σ' differs from resource allocation state σ only in that resource R' has one more free unit in it and resource R^* one less free unit, case (b) can only happen if $R = R^*$. However, if process P_i is not dead in σ', then there exists a successor of R^* with a free unit; hence process P_k cannot be dead either.

Since in both of the two possible cases discussed above the assumption that P_k is dead results in contradiction, the claim of Property 2 is true. ■

The next property expresses the fact that in the considered RAS class, state safety is equivalent to the absence of partial deadlock.[5] The result is stated and proven in a way that aligns with the broader developments of this section.

Property 3 *Consider the DFSA $G(\Phi)$ of a* FREE-RANGE*-RAS Φ *such that the capacity of each resource R satisfies $C(R) > 1$. Then, a reachable state σ of $G(\Phi)$ is safe* iff *no process is dead in σ.*

Proof: By Definition 2, state σ is safe *iff* the marked state σ_M (i.e., the RAS state when each process has finished its run) is reachable from state σ. Clearly, if there exists a dead process in σ, then σ is not safe. Thus, to prove Property 3, it is sufficient to show that: for any state $\sigma \neq \sigma_M$ such that no process is dead, (*) there exists an event e such that function $\sigma' = f(\sigma, e)$ is defined and there is no dead process in state σ' (since, then, the repetitive invocation of this result a finite number of times, further implies the existence of an event sequence leading from σ to σ_M). To establish this last result, consider the DFSA $G(\Phi)$ described in the assumptions of Property 3 and its set of active events $\Gamma(\sigma)$. Since no process is dead in σ, and $\sigma \neq \sigma_M$, set $\Gamma(\sigma)$ is not empty. There are only two possible cases of σ: a) no event in $\Gamma(\sigma)$ renders any process dead in state σ', and b) there exists event $e_{i,j+1}$ that renders process P_i dead in σ'. In case (a), claim (*) is clearly true. For case (b) we shall show that there must exist another process P_k that remains not dead when advancing to its next stage, and therefore, by Property 2, there will be no dead process in the state σ' that results from this advancement.

Let $R' = D_{ij}$, $R^* = D_{i,j+1}$ and $R'' = D_{i,j+2}$ (i.e., R', R^* and R'' denote respectively the resources held by process P_i in states σ and σ' and also the resource requested by it in σ'). Consider the subgraph $\mathcal{G}$ of $F(\sigma')$ that is induced by the node subset $\{R''\} \cup Suc(R''; \sigma')$. The specification of $\mathcal{G}$, combined with the fact that process P_i is dead in σ', imply that $\mathcal{G}$ must contain at least one strongly connected component, $\mathcal{G}'$, with all the nodes of $\mathcal{G}'$ corresponding to resources that are allocated to capacity. Hence, R' and R_∞ are not nodes of $\mathcal{G}'$. On the other hand, $\mathcal{G}'$ must contain the node corresponding to resource R^*, since otherwise, it would also be present in $F(\sigma)$ (we remind the reader that the only resources that have their allocation

[5] We emphasize that this equivalence between state safety and the absence of partial deadlock does not hold true, in general. Examples of deadlock-free and yet unsafe RAS states in the context of the considered application are provided in [20].

altered during the transition from σ to σ' are resources R' and R^*). But then, state σ would not be free of dead processes (all the processes allocated to resources in $\mathcal{G}'$ would be dead).

Since R^* belongs to $\mathcal{G}'$, it is filled to capacity in σ', and since $C(R^*) > 1$, there must exist at least one process P_l that holds a unit of it in state σ. Furthermore, P_l must be dead in σ' since, otherwise, process P_i would not be dead, either (the developments of the previous paragraph imply that, in $F(\sigma')$, the resources reachable from P_l are also reachable from P_i). The deadness of P_l in σ' implies the existence of another strongly connected subgraph of $F(\sigma')$, $\mathcal{G}''$, such that (i) all the nodes of $\mathcal{G}''$ belong to $Suc(R^*; \sigma')$; (ii) they correspond to resources allocated to capacity (and therefore R' is not one of them); while (iii) R^* is one of the nodes of $\mathcal{G}''$.

Hence, there must exist a process $P_k \neq P_i$ that in state σ' is allocated some resource R of $\mathcal{G}''$ and it requests resource R^* for its advancement. Clearly, this request is also present in state σ and its satisfaction constitutes a feasible event of σ. Furthermore, since $\mathcal{G}''$ manifests the deadness of the process P_l, there must be a path π in $F(\sigma)$ leading from R^* to R. The presence of path π in σ further implies that advancing process P_k instead of process P_i will lead to a state σ'' in which process P_l is not dead. But then, process P_k is also not dead (since, reasoning as in the case of the advancement of process P_i above, if P_k were dead, resource R^* should belong in a subgraph $\mathcal{G}'''$ of $F(\sigma'')$ consisting of nodes corresponding to resources allocated to capacity to dead processes). ■

Next we leverage the results of Properties 1–3 in order to articulate the safety conditions that will define the logics of the sought supervisor.

Theorem 1 *In* FREE-RANGE*-RAS *with* $C(R) > 1$, $\forall R$, *allocation of the available resource* R^* *to process* P_i *executing stage* Ξ_j *in a safe state* σ *and with* $D(\Xi_j) = R'$, *leads to a safe state* σ' iff *any of the following statements holds true in state* σ *(or, equivalently, in graph* $F(\sigma)$*):*

1. $j + 1 = l(i)$, *i.e., the next stage of* P_i *is its last one.*
2. *Allocation of resource* R^* *to process* P_i *does not fill* R^* *to capacity.*
3. *Resource* R^* *contains a process at its last stage.*
4. *If* $j < l(i) - 1$, *the resource* $R'' = D_{i,j+2}$ *has a free unit or it contains a process at its last stage.*
5. $R'' = R'$.
6. *Resource* R' *is a successor of* R^* *or of* R''.
7. *There exists a successor* $R \neq R^*$ *of* R'' *that has a free unit or it contains a process at its last stage.*

8. *There exists a successor R of R^* that has a free unit or it contains a process at its last stage.*

Proof: By Property 3, to prove this theorem, it is sufficient to show that no process is dead in state σ' *iff* any of the conditions (1–8) holds. Furthermore, under the assumptions of Theorem 1, Property 2 asserts that there will be no dead processes in state σ' as long as the advanced process P_i is not dead in σ'. Hence, to prove the sufficiency part of Theorem 1, it suffices to show that any of the conditions (1-8) will imply that process P_i is not dead in σ'. For conditions (1), (4), (5) and (7) this implication is immediate, when taking also into consideration Property 1. For condition (2), (3), (6) and (8), notice that according to the argument developed in the proof of Property 3, the deadness of P_i in σ' would imply that R^* is part of a strongly connected component of $F(\sigma')$ consisting of nodes corresponding to resources allocated to capacity. But it is easy to see that all these four conditions negate the development of the aforementioned structure.

To prove the necessity part of Theorem 1, we will show that if none of conditions (1–8) holds then process P_i must be dead in σ'. Indeed, it can be easily checked that if none of conditions (1–8) holds, the resource set $R'' \cup Suc(R''; \sigma')$ will consist of resources filled to capacity with processes requesting some other resource in this set. But then, Property 1 implies that process P_i is dead. ■

We close the developments of this section with some complexity considerations regarding the tests proposed in Theorem 1.

Theorem 2 *The evaluation of conditions (1–8) of Theorem 1 can be performed in time $O(|\mathcal{P}|)$, where $|\mathcal{P}|$ denotes the number of active processes in the underlying DFSA $G(\Phi)$.* [6]

Proof: Clearly, conditions (1–5) of Theorem 1 can be assessed in time $O(1)$. Regarding conditions (6–8), we first notice that the only part of the graph $F(\sigma)$ that is relevant for the assessment of these conditions is the subgraph $F'(\sigma)$ determined by the resources allocated to the various processes, the resources that constitute immediate requests for these processes, and the edges that represent the posed requests. Clearly, the size of $F'(\sigma)$ is $O(|\mathcal{P}|)$, and furthermore, under pertinent storage of the necessary information, the time that is required for the construction of $F'(\sigma)$ is also $O(|\mathcal{P}|)$. Once $F'(\sigma)$ has been constructed, the computation of the successor sets of R^* and R'' can be performed in linear time with respect to the graph size [6], which further implies that conditions (5–7) can be assessed in $O(|\mathcal{P}|)$. ■

In the next section, we discuss how the above results can facilitate the deployment of a liveness-enforcing supervisor (LES) for the vehicular systems considered in this work that is maximally permissive and can be implemented in a distributed manner.

[6] We remind the reader that in the considered application context, the active RAS processes abstract the running vehicle trips, and therefore, $|\mathcal{P}|$ is practically bounded by the fleet size.

4 Distributed implementation of the maximally permissive LES

4.1 Some assumptions and stipulations for the proposed control framework

The developments of the previous section have established that under the resource allocation dynamics induced by the tesselation scheme of Section 2 and formalized through the automaton of Definition 1, the advancement of a vehicle A_i from its currently held cell R' to the next requested cell R^* is admissible *iff* (i) cell R^* is currently allocated to less than two vehicles and (ii) at least one of the eight conditions of Theorem 1 holds true. In the subsequent discussion, condition (i) defines a notion of *feasibility* (w.r.t. the considered resource allocation policy) for the contemplated vehicle advancements. On the other hand, condition (ii) defines the notion of *safety* for these advancements (i.e., the preservation of the liveness of the underlying traffic).

In the context of the distributed control scheme to be presented in this section, vehicle A_i needs to resolve the two issues of feasibility and safety for any contemplated advancement to a next cell through a communication protocol that will enable it to collect all the necessary information while leveraging the communication capabilities and the information that is possessed by the other vehicles. If the outcome of this communication reveals that the aforestated two conditions are satisfied, vehicle A_i will allocate R^* to itself and it will advance to that cell, eventually releasing its current cell R'. If, on the other hand, it is found that some of these two conditions is violated, vehicle A_i will go into a waiting mode and attempt its advancement at a later time. For the reasons to be explained in the sequel, this time is randomly selected from an exponential distribution with some rate λ.

Next, we present a series of assumptions and stipulations that provide further specificity to the operational scheme that is outlined in the previous paragraph.

1. Each vehicle A_i has only knowledge of its own *local state* $s(A_i)$, that includes the parameters characterizing the state of the vehicle in, both, the continuous and the DFSA model that characterizes the cell allocation. The information about the state of any other vehicle, A_j, can be obtained by querying A_j, that replies with the information reflecting its state at the time of handling the query.

2. A vehicle can query directly only the vehicles that are currently within its communication range. The structure of the considered protocol requires that this communication range is no less than $\sqrt{4(d+\rho)^2+(d-2\rho)^2}$. To obtain information from beyond this area, the query must be propagated from less to more distant vehicles. The communication protocol that facilitates this interaction must ensure that each query is handled in finite time.

The last part of Assumption 2 above ensures that a query initiated by a vehicle A_i will be resolved in finite time. This resolution implies that either (i) vehicle A_i has obtained an

explicit response to its question, or (ii) it has inferred a response through the observation of the overall communication process initiated by this query, or (iii) it was not able to obtain a definite response to the question underlying its query, and therefore, it will have to repeat the query at a subsequent time.

Furthermore, as explained in the opening part of this section, the queries introduced in Assumption 2 above intend to help a vehicle to assess whether a contemplated cell advancement is admissible or not. But the potential communication latency that is suggested by this assumption, when combined with the asynchronous operation of the proposed control scheme, further imply that by the time vehicle A_i receives some information about the state of some other vehicle A_j, the state of vehicle A_j might have actually changed. Clearly, the safety assessment employed by the proposed allocation policy must be robust to such changes. We will demonstrate in the sequel that the LES determined by Theorem 1, and implemented by the proposed control scheme, satisfies this requirement.

An issue that arises by the asynchronous operation of the system vehicles, and needs particular attention in the proposed control framework, concerns the resolution of the potential conflict between two (or even a larger number of) vehicles that try to simultaneously resolve the safety of the allocation of the same cell R to themselves. In the context of centrally coordinated resource allocation, this issue is immediately resolved by the serialization of the decision-making process that results from the vehicle interaction with the central coordinator. In the considered operational scheme, this conflict must be explicitly addressed by the proposed control scheme. Hence, we further stipulate the following:

3. At any time point, only one vehicle can make a decision about the allocation of a particular cell. That is, through an appropriate communication protocol, a vehicle should obtain first the *testing rights*, i.e., the exclusive rights to test the possibility of the allocation of the next-needed resource R^* to itself.

4. During the time of calculating the decision, the vehicle having the testing rights *tentatively allocates* the resource in question to itself. Depending on the obtained result, this tentative resource allocation either turns into a *stable* one or it is canceled.

5. During the period of the tentative allocation of R^* discussed in item #4 above, the vehicle responds to queries regarding the availability of R^* as if its allocation of R^* was stable.

6. During its transitional phase from a cell D_{ij} to $D_{i,j+1}$, vehicle A_i responds to queries regarding the *feasibility* of the allocation of cells D_{ij} and $D_{i,j+1}$ as if it is an occupant of both cells. On the other hand, vehicle A_i responds to queries regarding the *safety* of the allocation of cells D_{ij} and $D_{i,j+1}$ as if it is an occupant of cell $D_{i,j+1}$ only.

Stipulations (5) and (6) are necessary for establishing the validity of the proposed protocol; their particular role will be revealed in the next section, that discusses the detailed protocol specification and its dynamics.

4.2 The distributed control scheme and its supporting protocol

This section details the distributed control scheme that is deployed at the top layer of the hybrid controllers that govern the motion of the traveling vehicles, and seeks to ensure the safety of the resource allocation that is manifested by the vehicle advancement among the cells that define their trips. From the discussion of the previous sections, this control layer must also handle (i) the vehicle communication that will provide the information that is necessary for the control function, and (ii) the supervision of the underlying motion processes; this last task involves the authorization of the execution of the various motion segments that (de-)compose the entire vehicle trip, and the monitoring of the completion of these segments. Figure 3 depicts the basic functionality implemented at this layer of the control architecture that is embedded in any single vehicle. In the depicted protocol, the vehicle operation is driven by the *handle_events* and *handle_messages* control blocks, which work in parallel. The blocks denoted with bold names realize procedures that will be explained in more detail in the sequel. Furthermore, in the following discussion, we will use the symbols R', R^*, and R'' as in the statement of Theorem 1; i.e., R' will denote the cell occupied by vehicle A_i for the execution of its current stage Ξ_{ij}, R^* is the cell required by A_i for its next stage, $\Xi_{i,j+1}$, and R'' is the cell required for stage $\Xi_{i,j+2}$.

The progress of a vehicle A_i on its designated trip is supervised by the following control loop:

1. Set $j = 0$.
2. Try to obtain the exclusive rights to attempt allocation of cell R^*, required for the next stage, $\Xi_{i,j+1}$.
3. If these rights are granted, i.e., if there is no conflict with another vehicle in accessing R^*, attempt the allocation. Otherwise, go into a random delay and repeat step (2) at the end of this delay.
4. If the allocation succeeded, advance, by entering the cell R^* corresponding to stage $\Xi_{i,j+1}$, and de-allocating R'. If $\Xi_{i,j+1}$ is the last stage, complete the entire trip and deallocate R^*. Otherwise set $j := j + 1$, authorize the motion segment corresponding to the transition of cell R^*, and go to step (2) once notified that this motion segment has been completed.
5. If the allocation attempted in steps (2-3) did not succeed, go into a random delay and repeat step (2).

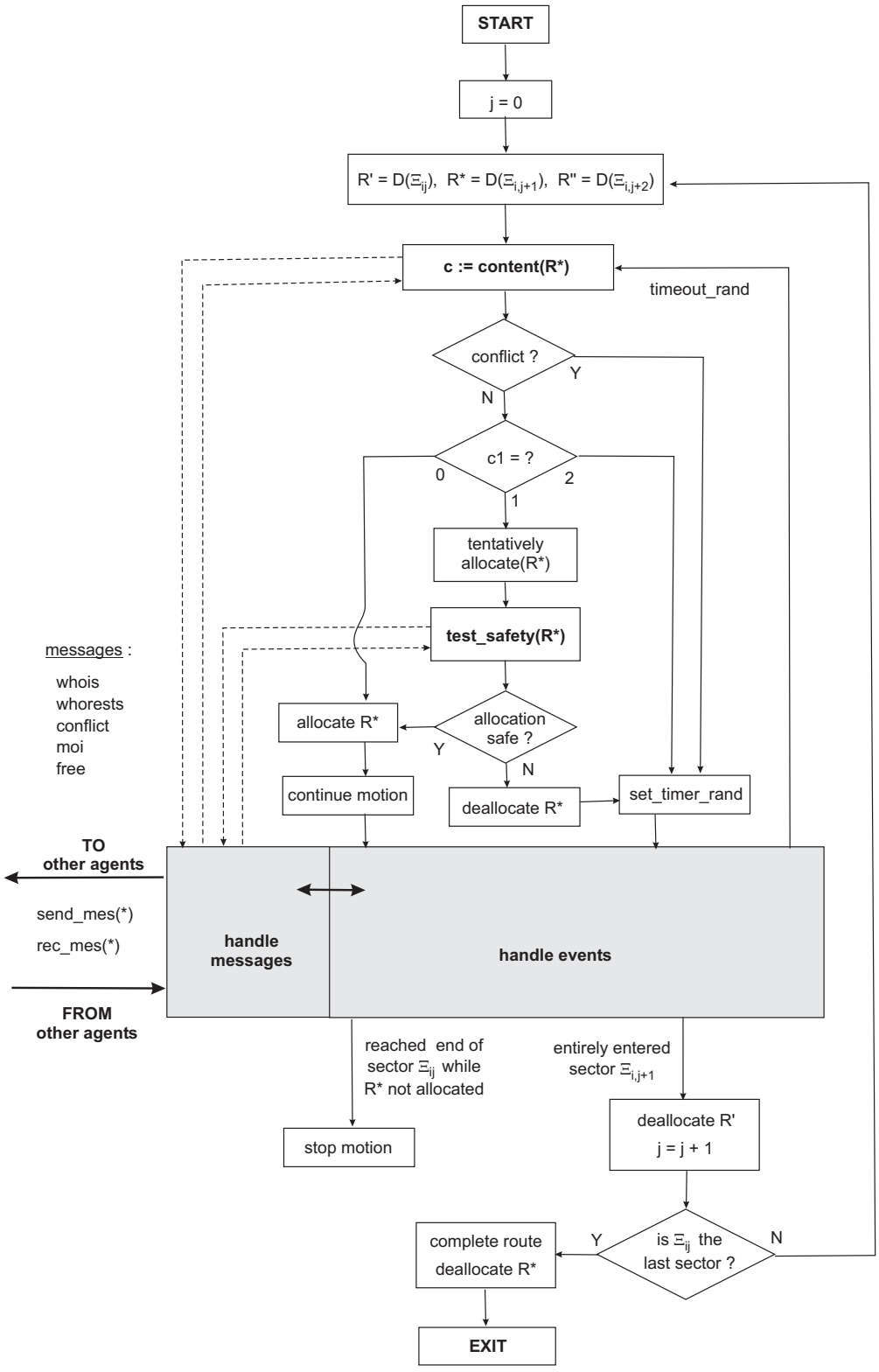


Figure 3: The basic structure of the control logic that is implemented at the top layer of the vehicle controller and coordinates its advancement among the various cells that define its trip.

6. When the last cell is deallocated, exit.

Next we detail the control logic that determines the admissibility of a contemplated resource allocation and the communication protocol that supports the execution of this logic. The allocation procedure is based on Theorem 1 and the communication protocol implements the assumptions presented in the previous section. From a more physical standpoint, vehicle communication is based on the broadcasting of various types of messages that are received and processed by any vehicle that is located within the communication range of the broadcasting vehicle. The protocol for handling those messages is depicted in Figure 4, and it is detailed in the subsequent discussion. Furthermore, the structure of these messages is depicted in the bottom-left part of Figure 4. More specifically, a message consists of the header and the message body, where the former part includes the information on the type of the message, addressed cell, sender cell, message creator ID, and time-stamp, and the latter contains data dependent on the message type.

To make an allocation decision, vehicle A_i proceeds as follows:

1. A_i first checks the feasibility of the allocation, i.e., the number of vehicles that occupy cell R^*. To this end, A_i executes procedure $content(R^*)$ (see Figure (5a)), which sends a message of type $whois$, addressed to the vehicles occupying cell R^*, and sets the timer. A recipient of this message, say vehicle A_k, is supposed to answer it with a message of type moi that conveys the information about its status. More specifically, the content of this message is the pair $c(moi) = (c1(moi), c2(moi)) \in \{(1,1),(1,0)\}$, that is interpreted as follows: $c1(moi) = 1$ denotes the fact that the disk of vehicle A_k overlaps with R^*; $c2(moi) = 0$ represents the fact that A_k is about to leave R^*, i.e., either it has already been allocated the cell needed for the next stage or its current stage is the last one; finally, $c2(moi) = 1$ if neither of the last two conditions is true. If, while awaiting the response, vehicle A_i receives a query $whois$ from another vehicle that targets the same cell R^*, A_i signals $conflict$ and exits. The same happens when, as a response to its own message, vehicle A_i receives message $conflict$ from another vehicle. If no answer is received in a specified time window, A_i assumes that cell R^* is free, which implies that the safety condition (2) is true, and the vehicle allocates the cell to itself. If A_i receives two answers (i.e., the component c1=2, in the c data structure depicted in the computation of Figure (5a)), then resource R^* is filled to capacity and the attempted allocation fails.

2. If vehicle A_i receives only one response – i.e., $c1 = 1$ and, therefore, one unit of R^* is free – then, A_i tentatively allocates a unit of R^* to itself and executes procedure $test\_safety$ (see Figure 6), to test the conditions (1–8) of Theorem 1. If any of these conditions holds true, then vehicle A_i allocates R^* to itself in a stable manner. Otherwise, the allocation test fails and A_i de-allocates R^*. During the period of tentative allocation of R^*, vehicle

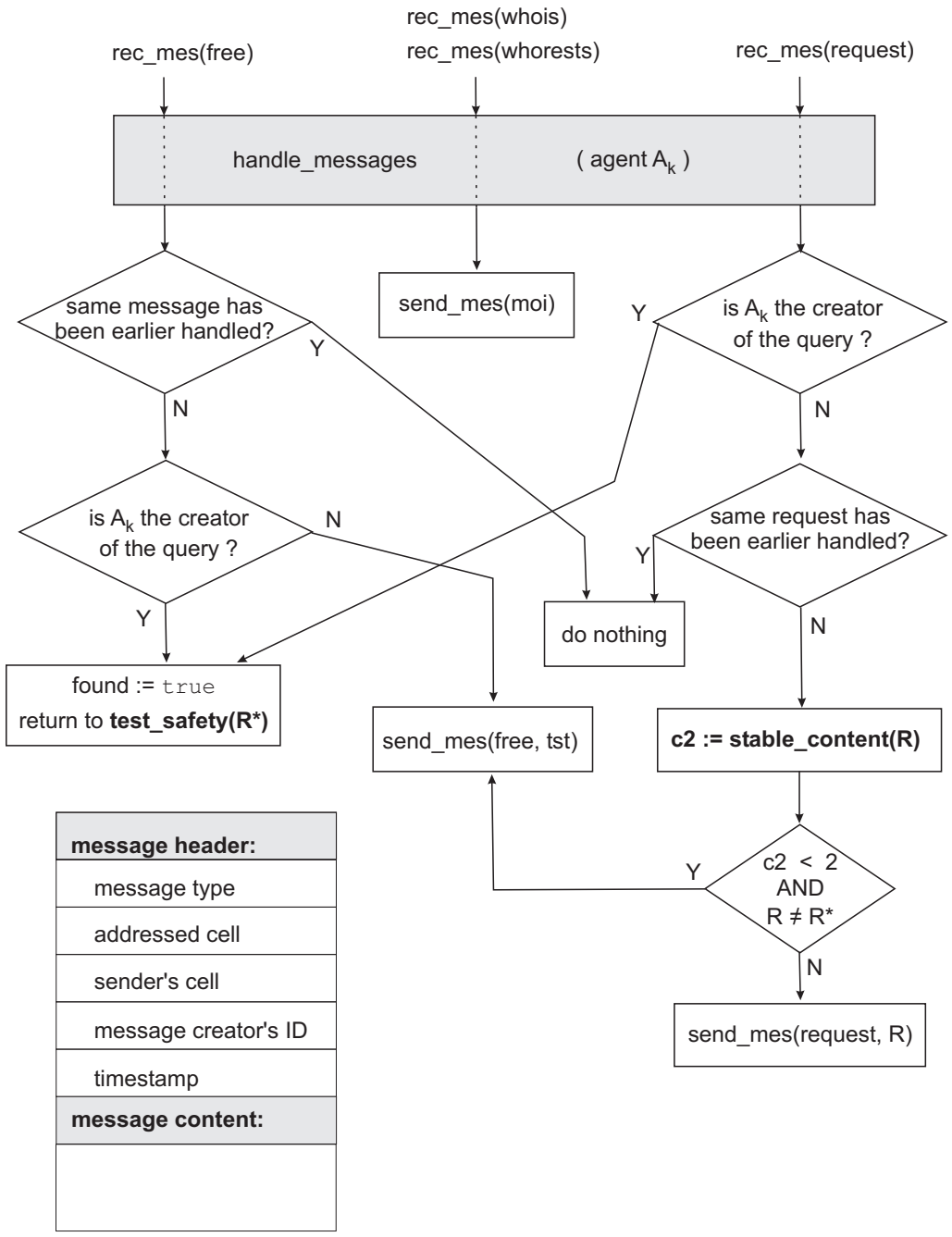


Figure 4: Message structure (bottom left), and the protocol for handling the various received messages.

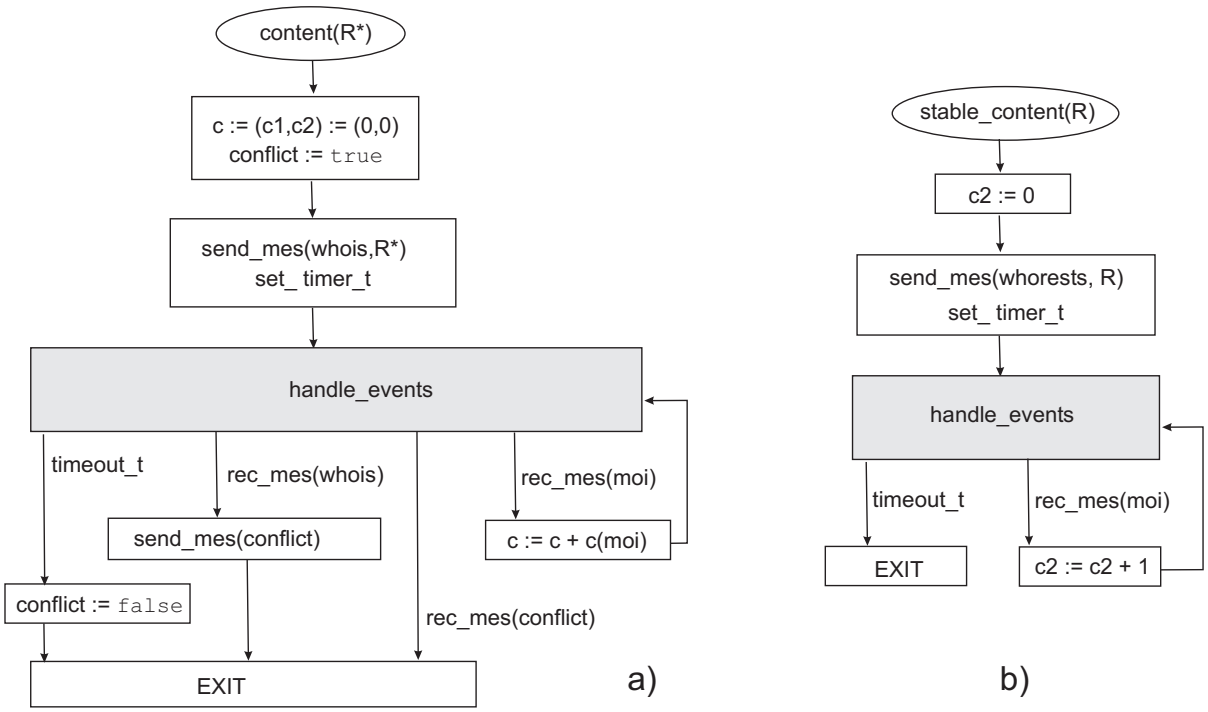


Figure 5: (a) Procedure *content*(R^*) that checks the allocation state of cell R^* in a mutually exclusive mode, and (b) procedure *stable_content*(R) that returns the number of vehicles in R that are not about to leave it (i.e., neither at their last stage nor already allocated the resource for their next stage).

A_i responds to queries of other vehicles as if it is a stable holder of the resource. Since conditions (1) and (5) refer to the route of vehicle A_i, they can be immediately checked locally. Also, conditions (2) and (3) will be true or not depending on whether the value of component $c2$, that was returned by the execution of procedure *content*(R^*), is equal to 0 or to 1. If none of conditions (1–3) and (5) are true, vehicle A_i checks conditions (4) and (6-8), which involve the more complicated task of assessing the existence of a cell $R \neq R^*$ that is not fully occupied and lies on a path that starts from R'' or R^*.

3. To test condition (4), vehicle A_i executes first the procedure *stable_content*(R'') (see Figure (5b)). This procedure sends the message *whorests* to the vehicles located in cell R'', awaits their answer through message *moi*, as in the case of the query *whois*, and checks the sum $c2$ of the values in the corresponding fields of the received responses. If $c2 < 2$, i.e., if R'' is not fully occupied or one of its occupants is about to leave, procedure *test_safety* sets the value $safe = true$ and exits.[7] Otherwise A_i proceeds to

[7]We also notice that it is the support of the procedure *stable_content*(R'') that sets the lower bound of the vehicle communication range that was provided in Section 4.1. More specifically, this bound is obtained by the need to reach a vehicle that has allocated cell R'' to itself but it has not entered this cell yet.

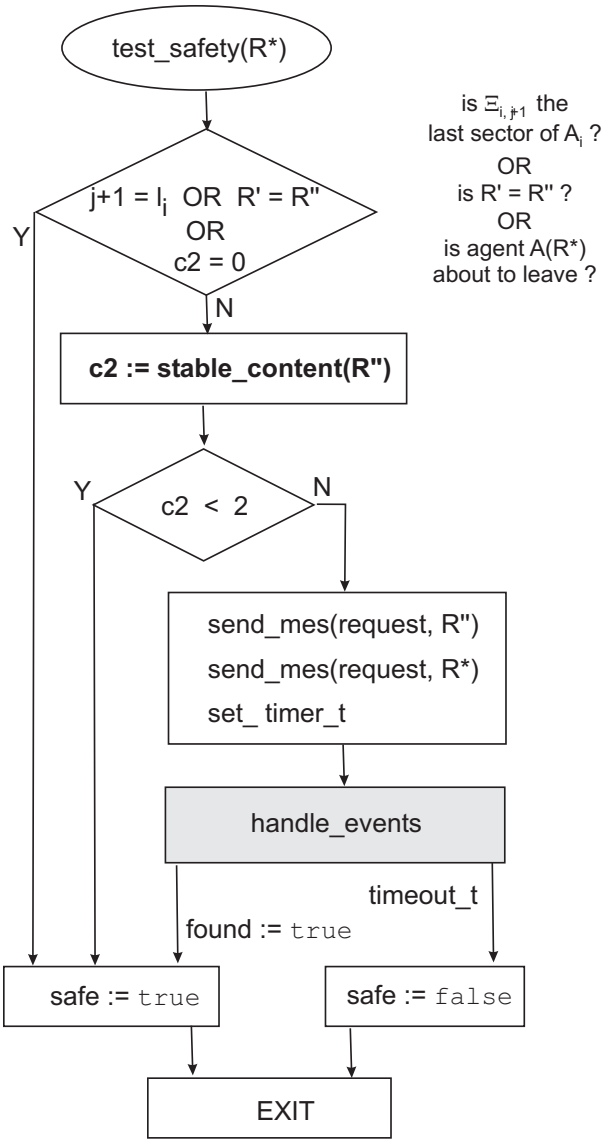


Figure 6: Procedure $test\_safety(R^*)$ that establishes whether or not allocation of R^* to vehicle A_i is safe, according to the (transition) safety characterizations of Theorem 1

test conditions (6–8) and sends a message of type *request* to cells R'' and R^*. A recipient of *request*, say vehicle A_k, initiates a recursive procedure (see Figure 4) that performs the following operations:

(a) Checks the creator ID and time-stamp of the received message to see whether the same *request* message has been received earlier through a different path. If so, it exits.

(b) Checks the stable content, $c2$, of the vehicles located in cell R that is required for the next stage of vehicle A_k.

(c) If $c2 = 2$ (i.e., a free unit cannot be guaranteed in cell R), A_k sends a *request* message to the vehicles occupying R. Otherwise, i.e., if $c2 < 2$, a resource (cell) with a free unit is found. This is communicated back to vehicle A_l, that had sent the request message to A_k, by a message of type $free$.

(d) As already observed in the earlier discussion, if vehicle A_i receives a *request* message addressed to the holders of the contested cell R^*, it responds as if it was one of the current holders of that cell, following steps (a-c) above. On the other hand, upon the reception of a *request* message addressed to the holders of resource R' and pertaining to the allocation request initiated by itself, vehicle A_i infers the safety of the contemplated allocation, setting $safe = true$ in procedure $test\_safety$.

In the next section we provide some examples that will demonstrate and concretize the control logic and the communication protocol that were presented in the previous paragraphs. The correctness of this control scheme with respect to the stated objective of ensuring collision-free and live motion for the underlying traffic system is established in Section 4.4.

4.3 Illustrative examples

To illustrate the operation of the distributed control logic and the protocol that were defined in the previous section, let us consider the state σ of the FREE-RANGE*-2-RAS depicted in Figure 2. For each vehicle A_i, $i \in 1, \ldots, 10$, the figure shows the resource $R'(i)$, currently allocated to the vehicle, and the next-required resource $R^*(i)$. The resource $R''(i) = D_{i,j+2}$, for each vehicle A_i, is as follows: $R''(1) = R_2$, $R''(2) = R_4$, $R''(3) = R_\infty$, $R''(4) = R_3$, $R''(5) = R_7$, $R''(6) = R_6$, $R''(7) = R_2$, $R''(8) = R_8$, $R''(9) = R_4$, $R''(10) = R_\infty$. Also, as can be noticed in Figure 2, for each vehicle A_i (or, equivalently, process P_i), there exists a path in graph $F(\sigma)$ that starts from the edge labeled with i and ends with a resource that has a free unit of capacity (R_7 in the case of $i \in 1, \ldots, 8$, R_5 for $i = 9$, and R_{10} for $i = 10$). Thus, by Property 1, no process is dead in the depicted state σ, and by Property 3, state σ is safe. Since the vehicles operate in parallel and asynchronously, there are various possible directions for the evolution

of state σ. Below, we consider a few possible scenaria. In their description, we will use the notation '$A_i \to R_j$' to express the fact that vehicle A_i attempts to transition to cell $R_j = R^*(i)$ and tests the feasibility and safety of allocating to itself (a capacity unit of) that cell.

1) $A_9 \to R_5$. To perform this test, in the main loop of its motion control procedure (c.f. Figure 3), vehicle A_9 sets $R' = R_{10}$, $R^* = R_5$, $R'' = R_4$, and initiates procedure $content(R_5)$ (c.f. Figure 4(a)). Since no other vehicle attempts to allocate R_5 to itself, there occurs no conflict, and since there is no vehicle in cell R_5, the query produced by $send\_mes(whois, R_5)$ is not responded. Hence, procedure *content* eventually times out with the values $c = (c1, c2) = (0, 0)$ and $conflict = false$. This allows A_9 to allocate R_5 to itself and continue its motion into this cell.

2) $A_4 \to R_2$. As in the previous scenario, the considered resource is contested by vehicle A_4 only, and therefore, no conflict occurs in acquiring the testing rights. However, when testing the content of R_2, vehicle A_4 receives two messages of the type *moi*, from vehicle A_2 and from vehicle A_8, both with $c = (1, 1)$. Hence, the procedure *content* exits with the value $c1 = 2$, indicating current unavailability of R_2. This makes vehicle A_4 set a timer and suspend any further allocation attempt until the timeout, when the testing will be repeated.

3) $A_6 \to R_7$, $A_7 \to R_7$ and $A_1 \to R_7$. In this case, each of the three vehicles attempts to allocate to itself the same cell, which may produce a conflict when the procedure $content(R_7)$ is executed at the same time by two or three vehicles. However, due to the random delay in the repetition of the procedure, eventually one of the vehicles will exit $content(R_7)$ with the values $conflict = false$ and $c1 = 1$ (the latter value is set as a result of the answer *moi* from A_4).[8] The winning vehicle tentatively allocates R_7 to itself and begins the execution of the $test\_safety(R_7)$ procedure (c.f. Figure 6). Moreover, from this point on, in the case of a query *whois* or *whorests* addressed to cell R_7 by any of the other vehicles, this vehicle responds as if it was a stable holder of resource R_7, i.e., with the value $c = (1, 1)$ sent in the message *moi*. The remaining part of this scenario depends on which vehicle has been the winner of the testing rights for R_7. Let us consider the following cases:

3.1) $A_6 \to R_7$. Since none of the conditions in the first box of $test\_safety(R_7)$ holds, the process control is passed to function $stable\_content(R_6)$ (c.f. Figure 4(b)) that tests the number $c2$ of vehicles in $R''(6) = R_6$ that are not yet able to leave it. To do this, vehicle A_6 sends a query *whorests* to R_6, receives answer *moi* with data $(c1, c2) = (1, 1)$ from A_4, and so it finds out that $c2 = 1$. Thus, $test\_safety(R_7)$ exits with the value $safe = true$, which allows vehicle A_6 to change the tentative allocation of R_7 to a stable one, and to proceed further in the same way as vehicle A_9 in scenario (1).

3.2) $A_7 \to R_7$. Vehicle A_7 begins in the same way as in scenario (3.1), but in this case, the execution of $stable\_content(R_2)$ returns the value $c2 = 2$, since, in the depicted state, re-

[8] The claim in this sentence is formally established in the next section; c.f. Proposition 2.

source $R''(7) = R_2$ is occupied in a stable manner by two vehicles. Consequently, the procedure $test\_safety(R_7)$ follows the right path emanating from its second conditional block (c.f. Figure 6), i.e., initiates a propagation of message *request* along the paths starting from $R^*(7) = R_7$ and from $R''(7) = R_2$, in order to find out if any of these paths contains resource $R'(7) = R_8$ or a resource R such that $c2 = stable\_content(R) < 2$. Thus, vehicle A_7 sends two *requests*, addressed to the vehicles in cells R_7 and R_2. The recipients of these messages follow the procedure depicted in Figure 4, that is:

- A_4 checks $c2 = stable\_content(R_2)$, and since $c2 = 2$, it relays *request* to R_2.

- A_2 and A_8, occupying cell R_2, receive the same *request* twice, from A_7 and A_4, yet undertake the appropriate action only after receiving the first message, while the next one is ignored. The vehicles check $c2 = stable\_content(R_3)$, and since $c2 = 2$, each of them sends *request* to R_3.

- A_3 and A_5, occupying cell R_3, receive the two *request* messages issued by A_2 and A_8, responding to the first and ignoring the second. In response to the first *request*, they check $c2 = stable\_content(R_8)$, and since $c2 = 2$, each of them sends a *request* message to R_8.

- A_6, occupying cell R_8, receives the two *request* messages issued by A_3 and A_5, and ignores the second one. In response to the first, it checks $c2 = stable\_content(R_7)$, and it obtains an outcome $c2 = 2$, since both vehicles A_4 and A_7 declare themselves as stable holders of this resource. Hence, A_6 proceeds with the relay of the *request* to the holders of resource R_7.

- On the other hand, when receiving the first *request* from A_3 or A_5, vehicle A_7 recognizes itself as the initiator of this request. In response to this event, vehicle A_7 exits procedure $test\_safety$ with the value $safe = true$, changes the allocation status of R_7 from tentative to stable, and proceeds further as in scenario (1).

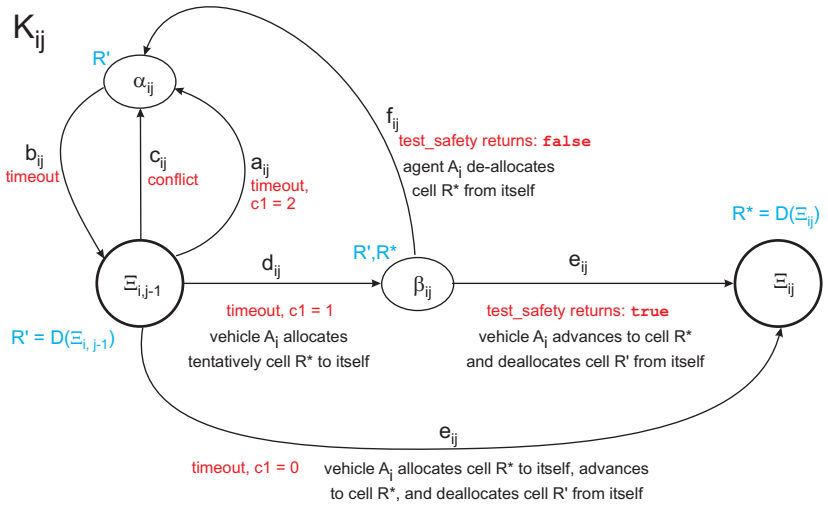
3.3) $A_1 \rightarrow R_7$. This scenario starts similarly to scenario (3.2). Vehicle A_6 finds out that $c2 = stable\_content(R_2) = 2$, and therefore, it initiates the propagation of message *request*, that is first sent to $R^*(1) = R_7$ and $R''(1) = R_2$, and then relayed, through R_3 and R_8, back to R_7. Yet, in contrast to case (3.2), neither a resource $R \neq R^*$ with $stable\_content < 2$ nor cell R_6, that is currently occupied by A_1, is reached through the aforementioned relay. Thus, procedure $test\_safety$ will time out and exit with the value $safe = false$. Consequently, vehicle A_1 cancels its temporary allocation of resource R_7 and suspends its activity for some random time, after which it repeats the same step $A_1 \rightarrow R_7$. The reader should notice that the above result indeed prevents vehicle A_1 from becoming dead; allocation of R_7 to A_1 in the cell allocation depicted in Figure 2 would produce a deadlock involving vehicles $A_1, A_2, \ldots, A_8$.

4.4 Establishing the correctness of the distributed control scheme and the supporting communication protocol

In this section we establish the correctness of the proposed control scheme and the supporting communication protocol with respect to the stated objective of ensuring collision avoidance and liveness for the underlying traffic system. Hence, in line with the opening remarks of Section 4.1, we need to establish that under the proposed control scheme, the advancement of a vehicle A_i from its currently held cell R' to the next requested cell R^* is admissible *iff* (i) cell R^* is currently allocated to less than two vehicles, and (ii) at least one of the eight conditions of Theorem 1 holds true. Furthermore, we need to prove that the communication protocol itself does not create any additional dynamics that would permanently stall the advancement of some vehicles.

A particular structure that will facilitate the development of the aforementioned results is the automaton K_{ij} depicted in Figure 7. This automaton abstracts the dynamics of a vehicle A_i that is in cell R', corresponding to the $(j-1)$-st cell in its trip, and seeks the allocation of the next cell, R^*, according to the logic of the proposed control scheme. From the standpoint of our analysis, this process involves four key states, $\Xi_{i,j-1}$, β_{ij}, α_{ij} and Ξ_{ij}. The semantics of those states, and of the available transitions among these states, are defined in Figure 7 itself. Furthermore, the resource allocation dynamics that are experienced by vehicle A_i as it tries to secure and traverse the entire sequence of cells that define its trip, can be captured by a "cascade" of automata similar to that depicted in Figure 7, for $j = 1, \ldots, l(j)+1$, that are merged through the corresponding states Ξ_{ij}; we shall refer to the resulting automaton as the automaton K_i. Finally, the dynamics of the entire traffic system can be abstracted through an automaton K that results from the composition of the automata K_i in a way that further observes the structure of the underlying RAS and the control and communication logic that is outlined in Section 4.2. In particular, the state σ_K of this automaton is defined by the state of each vehicle A_i with respect to the corresponding automaton K_i.

For the needs of the subsequent developments, it is also important to notice that while in states $\Xi_{i,j-1}$, β_{ij} and α_{ij} of automaton K_{ij}, vehicle A_i is actually allocated to cell $R' = D(\Xi_{i,j-1})$, and only when transitioning to state Ξ_{ij} does A_i advance, initially logically and subsequently also physically, to cell $R^* = D(\Xi_{ij})$. Hence, in a DFSA-based representation that traces only the advancements of vehicle A_i among the various cells that define its trip (i.e., the execution of the events e_{ij} in the corresponding automaton K_i), the three states $\Xi_{i,j-1}$, β_{ij} and α_{ij} of each automaton K_{ij} can be aggregated to a single state; let us denote this state also by $\Xi_{i,j-1}$, and the automaton that results from K_i through the aforementioned aggregation by G_i. The composition of the automata G_i, corresponding to all vehicles A_i, in a way that respects the requirement that no cell is allocated to more than two vehicles in any (global) state, is the automaton $G(\Phi)$ of Definition 1 in Section 3. In particular, the events e_{ij} in automata K_{ij} are



state $\Xi_{i,j-1}$ - vehicle A_i holds R' and it attempts to obtain the testing rights and to determine the value of c = (c1,c2) = content(R*).

state α_{ij} - vehicle A_i holds R', sets its timer to a randomly selected value, and awaits timeout.

state β_{ij} - vehicle A_i holds stably R' and tentatively R*, and it executes the procedure test_safety(R*) in order to determine the safety of its advancement to the next cell.

state Ξ_{ij} - vehicle A_i holds stably R* and executes its motion segment corresponding to stage Ξ_{ij}.

Figure 7: The automaton K_{ij} abstracting the dynamics of vehicle A_i's efforts to secure its transition from the $(j-1)$-st cell to the j-th cell of its route.

in direct correspondence with the elements of the event set E introduced in Definition 1.[9]

In view of the formal abstractions that were defined in the previous paragraphs, the first of the two key results that were outlined in the beginning of this section can be stated and proven as follows:

Proposition 1 *Consider a state σ_K of automaton K and its corresponding state σ in automaton $G(\Phi)$. Then, if σ is a safe state of $G(\Phi)$, the occurrence of an event e_{ij} in K will lead to a state σ' of $G(\Phi)$ that is also safe.*

Proof: First we notice that from the protocol description in Section 4.2, it is clear that no vehicle will try to allocate a new cell to itself while that cell is occupied by two vehicles. Furthermore, a vehicle A_i that has acquired the testing rights for its next requested cell R^* can rest assured that the number of vehicles occupying this cell may increase only through its own decision to allocate the cell to itself. Hence, the decision by a vehicle to allocate a cell to itself through a type-e event will never violate the stipulation that a cell is never allocated to more than two vehicles, and σ' is indeed a valid state of the automaton $G(\Phi)$.

Next, we establish the safety of such allocations. As remarked in Section 4.2 (and further stated in Figure 7), the decision of a vehicle A_i to turn the request for its next cell into a stable

[9]This correspondence should also be evident from the characterization of the events e_{ij} in Figure 7.

allocation (in other words, the execution of a type-e event) is triggered by the inference reached by vehicle A_i that one of the conditions of Theorem 1 holds true. A complication for this part of the proof arises from the potential latency between the time that such a condition is detected in the system and the time that the vehicle is informed about the occurrence of this condition and determines to act with a type-e event. This remark is especially true for conditions (4), (6), (7) and (8) of Theorem 1. In particular, a condition of type (4), (6), (7) or (8) might have been detected in a state $\hat{\sigma}$ that precedes the current state σ, and this particular condition might not be true anymore in state σ. To deal with this complication, next we use an inductive argument that is based on the number of transitions, ν, that have taken place in the evolution of the automaton $G(\Phi)$.

The base case of this induction is defined by $\nu = 0$. Then, irrespective of what is the particular condition that triggers the next event e_i, $\hat{\sigma} = \sigma$, and the result of Proposition 1 follows immediately from Theorem 1.

Next, we assume that the first ν transitions of $G(\Phi)$ have maintained the safety of its state σ, and we establish that the $(\nu + 1)$-st transition will also lead to a safe state σ'. First, let us consider the case where the next transition is triggered by the inference of conditions (1) or (5) by vehicle A_i. From the content of these two conditions, it is clear that $\hat{\sigma} = \sigma$, and therefore, the safety of the next state σ' follows immediately from Theorem 1. The equation $\hat{\sigma} = \sigma$ holds true even in the case that the inferred condition is condition (2) or (3); more specifically, this equation results from the content of these two conditions and the remark provided in the opening part of this proof that the number of vehicles occupying cell R^* can increase only through the decision of vehicle A_i to allocate this cell to itself in a stable manner. Hence, conditions (2) and (3) are immediately resolved through Theorem 1, as well.

For conditions (4), (6), (7) and (8), the equation $\hat{\sigma} = \sigma$ might not be true anymore. Next, we argue the safety of state σ' for the case of condition (4); the remaining three cases can be argued in a similar manner. For condition (4), first we consider Case (a) where this condition was satisfied in state $\hat{\sigma}$ through the presence of a vehicle A_j in cell R'' executing its last stage. If, in state σ, vehicle A_j is not present in R'' any more, then, there are two possibilities regarding the allocation of the corresponding capacity unit of R'' in σ: (a-i) This capacity unit is free. (a-ii) This capacity unit is allocated to another vehicle A_l. In case (a-i), condition (4) is still true in state σ; hence, the execution of the considered event e_{ij} in it leads to a safe state σ'. In case (a-ii), the inductive hypothesis implies that vehicle A_l is not dead. Hence, by Property 1 of Section 3, there exists a path leading from R'' to a resource R of free capacity, possibly R_∞. Furthermore, this free unit is not the unit tentatively held by A_i on R^*, since the allocation of A_l to R'' took place at some state $\tilde{\sigma}$ that succeeded state $\hat{\sigma}$, and therefore, vehicle A_i had already performed that tentative allocation. Hence, the considered resource R will retain its free capacity in state σ', which further implies that vehicle A_i is not dead in σ'. But then, by

Property 2 of Section 3, state σ' is safe. Case (b), where condition (4) materialized in state $\hat{\sigma}$ through the presence of a free unit on resource R'', and this free unit is not available any more in state σ, can be handled by an argument similar to that used for case (a-ii) above. ■

The next result establishes that the proposed control scheme does not introduce additional dynamics that might lead to the indefinite stalling of some vehicle A_i in its current cell.

Proposition 2 *Consider a state σ_K of the aforementioned automaton K that corresponds to a safe state σ of the automaton $G(\Phi)$. Then, a type-e event will take place in finite time w. p. 1.*[10]

Proof: First , let us consider the case where state σ contains a vehicle A_i that has reached the terminal cell of its route and it is about to initiate its termination stage. Then, according to the previous description of the proposed control scheme (c.f. also Section 2), the vehicle can proceed immediately to the execution of the relevant motion while coordinating this maneuvering with any surrounding vehicles that might be affected by it. Hence, Proposition 2 is true in this case.

Next, we consider the more interesting case where, in state σ, every vehicle A_i needs to transition to a neighboring cell $R^* \neq R_\infty$. For the needs of the following argument, let $\mathcal{A}$ denote the set of the system vehicles, and set $|\mathcal{A}| = n$. Since state σ is safe, there will be a subset $\tilde{\mathcal{A}}$ of $\mathcal{A}$ that can advance to their next cell in a safe manner (i.e., the state σ' resulting from any such advancement will be safe). Also, Proposition 1 guarantees that any attempt to advancement by a vehicle in $\mathcal{A} \setminus \tilde{\mathcal{A}}$ will be blocked by the considered protocol. In the rest of this proof, we shall show that some vehicle A_i in $\tilde{\mathcal{A}}$ will advance to its next cell R^* in finite time, w. p. 1.

From the description of the protocol in Section 4.2 (c.f. also the abstracting automaton in Figure 7), a successful attempt by a vehicle $A_i \in \tilde{\mathcal{A}}$ to allocate the next cell R^* to itself must go through the following two phases:

(a) In a first phase, the vehicle attempts to get the testing rights for cell R^* by sending the corresponding message *whois* and then waiting for a finite time τ_1 before it can conclude that either R^* is currently empty, and therefore the allocation of this cell to itself is safe, or that it has obtained the testing rights and can proceed with the further testing of the safety of the contemplated allocation. On the other hand, the vehicle will have to abort during this phase only if there is a conflict with another vehicle A_j that also tries to secure the testing rights for the same cell (the remaining possibility that vehicle A_i is blocked by the presence of two vehicles in cell R^* is excluded by the fact that $A_i \in \tilde{\mathcal{A}}$). In the last case mentioned above, the vehicle will restart a new attempt at a later time point, defined by a random delay that

[10] We remind the reader that the notation 'w. p. 1' is an abbreviation of the expression 'with probability 1' and it implies that the stated result will hold true with the possible exception of a set of outcomes with zero total probability measure [21].

is selected from an exponential distribution with some rate λ. Furthermore, letting t_{i0} denote the initiation time of A_i's attempt, it should be clear that the aforementioned conflict will be avoided as long there is no initiation of an advancement attempt to cell R^* by any other vehicle A_j over the time interval $(t_{i0} - \tau_1, t_{i0} + \tau_1)$.

(b) Upon the successful completion of the first phase described above, vehicle A_i might have to go through a second phase where it will have to resolve the safety of the allocation of cell R^* to itself. From the protocol description in Section 4.2, this phase will last a bounded time τ_2. More specifically, by the end of this time interval the vehicle will have already verified the safety of its advancement, or it will have failed to do so, in which case, it will have to abort and re-attempt its advancement at a later time point, determined by a random delay with distribution $Exp(\lambda)$. Since A_i belongs in $\tilde{\mathcal{A}}$, a negative outcome for this phase will result only in the case that another vehicle A_k is simultaneously testing a tentative allocation of its own next cell to itself. Such a situation can be avoided as long as there is no initiation of an advancement attempt by another vehicle in the interval $(t_{i0}+\tau_1-(\tau_1+\tau_2), t_{i0}+\tau_1+\tau_2-\tau_1) = (t_{i0}-\tau_2, t_{i0}+\tau_2)$.

It is also natural to assume that $\tau_2 \geq \tau_1$, since the vehicle communication involved in phase (a) is of a more local nature than the vehicle communication involved in phase (b). Hence, combining the results obtained in the analyses of phases (a) and (b) above, we can conclude that vehicle A_i will be successful in the considered endeavor to allocate its next cell R^* to itself, as long as there is no initiation of an advancement attempt by another vehicle over the time interval $(t_{i0} - \tau_2, t_{i0} + \tau_2)$. Since each vehicle (not necessarily in $\tilde{\mathcal{A}}$) performs a new advancing attempt with a random delay drawn from $Exp(\lambda)$, the probability of the aforementioned event is bounded from below by $\hat{p} = e^{-((n-1)\lambda)(2\tau_2)}$ (i.e., by the probability that the time to the next event for an exponential race with $n-1$ independent random variables sharing the same instantaneous failure rate λ is greater than $2\tau_2$ [21]). But then, Proposition 2 results from the basic properties of a Bernoulli trial with a positive "success" probability p [21], and the fact that the time between two consecutive attempts for any vehicle $\mathcal{A}_i \in \tilde{\mathcal{A}}$ is finite. ■

Finally, Propositions 1 and 2 lead to the following result that establishes the correctness of the proposed control scheme.

Theorem 3 *Under the control scheme of Section 4.2, and the further assumption that the intra-cellular traffic is coordinated by a local control scheme that enables all the locally located vehicles to reach the next set-point for their local motion profile in finite time and without any collisions,*[11] *every vehicle will reach its final destination in finite time w. p. 1.*

Proof: Proposition 2, together with the assumed liveness of the local motion within any given cell, guarantee that the underlying traffic system will advance from any safe state σ of automaton $G(\Phi)$ to a neighboring state σ' in finite time w. p. 1. Furthermore, Proposition 1

[11] We outlined such control schemes in Section 2.

guarantees the safety of σ'. But then, the result of Theorem 3 follows immediately from the fact that the safe sub-space of $G(\Phi)$ is a connected acyclic digraph with the only terminal node being defined by state $\sigma_M = [l(1)+1, \ldots, l(n)+1]$. ■

5 Conclusions

In this paper we presented a new distributed control scheme and its supporting communication protocol that will enable a fleet of vehicles sharing a common motion area to execute their trips in a collision-free and live manner. The proposed controller segments the vehicle motion by means of a tesselation of the motion area, and refers the continuous-time control problem regarding the collision-free and live execution of a single motion segment, by any given vehicle, to results already existing in the literature. On the other hand, the coordination of the execution of the entire set of motion segments, across all vehicles, in a way that ensures the liveness of the overall traffic, was abstracted to a resource allocation problem, and the relevant results from resource allocation system theory were extended so that they can support a distributed implementation. The resulting hybrid control scheme was formally proven to be correct and complete in terms of the combined objective of collision avoidance and liveness enforcement. It is also scalable with respect to the underlying fleet size, since the continuous-time control problem is of a local nature and it always concerns no more than a few vehicles, while the scalability of the coordinator that addresses the resource allocation problem and of the associated protocol is guaranteed by Theorem 2 in Section 3. *To the best of our knowledge, this is the first set of results to provide a non-trivial, complete, scalable and distributed solution to the problem of collision avoidance and liveness-enforcing supervision that arises in free-ranging vehicular systems with limited communication capabilities.*

We also notice, for completeness, that the adopted abstraction of the vehicle entity to a disk that is controlled by controlling the motion of its center, enables the application of the proposed control scheme even in vehicular systems where the vehicles are expected to be in perpetual motion from the initiation of their trip until they retire at their destination; the reader is referred to [20] for the methodological details that enable this implementation. Furthermore, the proposed approach is implementable in higher-dimensional spaces, or under other tessellation patterns, as long as it is possible to establish the existence of control schemes that can support collision-free and live vehicle motion within the various cells of the adopted tessellation. In fact, under some obvious modifications, the control logic and the communication protocol of Section 4.2 can support collision-free and live cell allocation in FREE-RANGE*-k-RAS for $k > 2$. This capability establishes the possibility of a more relaxed control at the higher level of the proposed hybrid control framework, and a potentially higher occupancy of the motion area, at the expense of more complex control for the lower level. The practical

implications of this trade-off need to be systematically assessed and explored.

More generally, our future endeavors will focus on (i) a full-fledged implementation of the proposed control scheme in simulated and/or actual vehicular systems, and (ii) the embedding of the developed controller in broader control architectures that will also address issues pertaining more explicitly to the system performance.

References

[1] *IEEE Robotics & Automation Magazine*, 18(3), 2011.

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

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

[4] G. Chalouros, G. P. Roussos, J. Lygeros, and K. Kyriakopoulos. Mid and short term conflict resolution in autonomous aircraft operations. In *Proceedings of the 8th Innovative Research Workshop and Exhibition*, pages –, 2009.

[5] 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*, 25:136–146, 2009.

[6] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms, 2nd ed.* MIT Press, Boston, MA, 2001.

[7] E. W. Dijkstra. Cooperating sequential processes. Technical report, Technological University, Eindhoven, Netherlands, 1965.

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

[9] G. Dudek and M. Jenkin. Inertial sensors, GPS, and odometry. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, pages 477–490. Springer-Verlag, 2008.

[10] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, MA, 1979.

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

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

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

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

[15] J. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Upper Saddle, NJ, 2001.

[16] J. Minguez, F. Laminaux, and J.-P. Laumond. Motion planning and obstacle avoidance. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, pages 827–852. Springer-Verlag, 2008.

[17] P. Morin and C. Samson. Motion control of wheeled mobile robots. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, pages 799–826. Springer-Verlag, 2008.

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

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

[20] S.A. Reveliotis and E. Roszkowska. Conflict resolution in free-ranging multivehicle systems: A resource allocation paradigm. *IEEE Trans. Robotics*, 27(2):283 – 296, 2011.

[21] S. M. Ross and E. A. Peköz. *A second course in probability*. www.ProbabilityBookstore.com, Boston, MA, 2007.

[22] E. Roszkowska and J. Jentink. Minimal restrictive deadlock avoidance in FMS's. In *Proc. European Control Conf. ECC '93*, volume 2, pages 530–534, 1993.

[23] E. Roszkowska and S.A. Reveliotis. A distributed protocol for motion coordination in free-range vehicular systems. In *Proc. of 18th IFAC World Congress*, 2011.

[24] H. Tanner, S. Loizou, and K. Kyriakopoulos. Nonholonomic navigation and control of cooperating mobile manipulators. *IEEE Trans. on R&A*, 19:53–64, 2003.

[25] 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*, 43:509–521, 1998.