Correct-by-construction distributed control for multi-vehicle transport systems

Elzbieta Roszkowska[1] and Ida Goral[2]

Abstract— **In this work we deal with a system of autonomous mobile vehicles with limited communication range and consider the problem of the formally correct distributed control, i.e., the design of the local control for each particular agent so that it ensures globally correct behavior of the system. A particular problem is the fact that the knowledge about the system state possessed by the agents is partial and never accurate, as during the time of information acquisition (by querying other agents) the state of the agents can change. The paper capitalizes on the results of [1] and contributes with: 1) further development of the theoretical idea toward the distributed supervisory control and the concept of robustness, 2) application of the results in the design of a transport agent, and 3) development of a demonstrator, i.e., a computer application that enables the user to specify a considered transport system and its tasks, creates a virtual environment, where each particular vehicle is represented by a particular software agent implementing the control model of (2), and allows observation of the agents concurrent operation and quantitative assessment of the system performance.**

I. INTRODUCTION

As pointed out by the editors of [2], robots often fail to do what they are expected to do; they go the wrong way, they drive into obstacles, and sometimes they just stay in place until one figures out what went wrong. The growing awareness of the insufficiency of the conventional engineering approach for dealing with this situation has motivated the research toward formal control synthesis methods and development of correct-by-construction hardware and software systems. The problem of ensuring the required behavior becomes even more complex if the object to control is not a single autonomous device, but a system composed of such devices as, in particular, a multi-vehicle system. In the prevailing approaches to modeling such objects, 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 [3], [4], [5], [6], [7], [8], [9] and a higher-level but more comprehensive description of the pursued methods can be found in [10]. However, while many of these works will guarantee collision freedom, very few of them have actually considered the issue of motion liveness. This mainly results from the continuous-time nature of the underlying models and their inability to provide in a scalable way formal liveness guarantees. Hence, in an effort to address these computational and analytical challenges, the relevant research was directed toward the employment of hybrid models [2]. In particular, works [11] and [1], underlying the developments of this work, propose such an approach to modeling and control of a multi-vehicle system.

In this paper we consider a transport system, where a number of autonomous mobile vehicles executes concurrently specific transport tasks, or missions, assigned to them in real time by the dispatching module. The motion process of each particular vehicle is viewed as a sequence of stages executed under the control of a conventional continuous-time model, while the transitions between these stages are governed by the logics of a discrete-event model. The role of the latter is to supervise the vehicle motion enforced by the former control level so that the behavior of the system as a whole be *collision-free* and *live*. That is, for the correctness of the system, it is respectively required that at any given moment of its operation: i) the space fragments occupied by any two vehicles be disjoint, and ii) each vehicle have the potential to terminate its currently executed mission in finite time and be ready to accept a new assignment. A particular problem is the fact that the enforcement of the system liveness requires a global model that conditions the admissibility of state transitions on the state of the system as a whole, while the knowledge about the system state possessed by the agents is partial and never accurate. The agents can acquire information by querying other agents, but during the time of its acquisition the state of the dynamic system can change. Consequently, an additional mechanism is required to ensure the robustness of the supervisory control, i.e., independence of the control decisions' results on the state changes occurring during the time of acquiring the necessary data and calculating the control decisions.

The specific objectives of the presented research are: 1) further development of the theoretical idea toward the distributed supervisory control and the concept of robustness, 2) application of the results in the design of a transport agent, and 3) development of a demonstrator, i.e., a computer application that enables the user to specify a transport system and its tasks, creates a virtual environment, where each particular vehicle is represented by a particular software agent implementing the control model of (2), and allows observation of the agents concurrent operation and quantitative assessment of the system performance. The following section describes the theoretic concepts underlying the construction of the considered supervisory control, and Section III presents their application in a computer system.

[1]Elzbieta Roszkowska is with Institute of Computer Engineering, Control and Robotics, Wroclaw University of Technology, Poland. `elzbieta.roszkowska@pwr.wroc.pl`

[2]Ida Góral is a graduate student at Department of Electronics, Wroclaw University of Technology, Poland. `goral.ida@gmail.com`

II. DFSA MODEL OF THE CORRECT SYSTEM BEHAVIOR

In this section we present a DFSA (*Deterministic Finite State Automaton (DFSA)* [12]) model that abstracts safe and robust behavior of a multi-vehicle-system with distributed control, that will provide the basis for the agents' control logic. Section II-A recaptures selected concepts of [1], and Section II-B further develops the model and proves its formal correctness.

A. DFSA model of the multi-vehicle system

We will view a multi-vehicle system as a set of processes sharing a set of common resources, and model it with a class of *Resource Allocation Systems (RAS)* [13]. The set of resources abstracts the set of *cells* obtained by a partition of the motion space into a grid of squares. A process abstracts in a discrete way the motion of a vehicle along its path. To model such systems, we will employ a RAS class, FREE-RANGE*-RAS, 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. 2) $C : \mathcal{R} \to \mathbb{Z}^+$ 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) $D : \Xi = \{\Xi_{ij} \mid i = 1, \ldots, n; j = 1, \ldots, l(i)\} \to \mathcal{R}$ 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 $D(\Xi_{ij})$. Then, the behavioral dynamics of FREE-RANGE*-RAS can be represented as follows.

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) $\forall i \in \{1, \ldots, n\}$, $0 \le \sigma_i \le l(i) + 1$, and (b) $\forall R \in \mathcal{R}$, $a(\sigma, R) = |\{\sigma_i \mid D_{i,\sigma_i} = R\}| \le C(R)$. 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
$$\sigma'_k = \begin{cases} \sigma_k + 1 & \text{if } \; k = i \;\wedge\; 1 \le j \le l(i) + 1 \\ & \qquad\qquad\quad \wedge\; \sigma_k = j - 1 \\ \sigma_k & \text{otherwise} \end{cases}$$
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.

Definition 2: Consider the DFSA $G(\Phi)$ of a FREE-RANGE*-RAS Φ.

1) state $\sigma' \in \Sigma$ is (resp., is not) reachable from state $\sigma \in \Sigma$ if there exists (resp., there does not exist) a feasible sequence of events that drives the automaton from state σ to state σ'. State $\sigma \in \Sigma$ is simply *reachable* if it is reachable from the initial state σ_0.
2) 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.
3) State σ is characterized as *safe iff* the marked state σ_M is reachable from state σ.

Definition 3: Consider the DFSA $G(\Phi)$ of a FREE-RANGE*-RAS Φ. The *resource allocation graph* 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, i.e., processes $P_i \in \mathcal{P}$ that in state σ execute a stage Ξ_{ij} s.t. $1 \ge j \le l(i) + 1$. Each such a process is represented by edge $(D_{ij}, D_{i,j+1})$, if $j < l(i)$, and by edge $(D_{i,l(i)}, R_\infty)$ otherwise, indicating the currently possessed resource and the resource required next.

The proofs of the following properties can be found in [1].

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 $p = R^1, R^2, \ldots, R^{l_p}$, $l_p > 1$, 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.

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.

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.

B. The correct system behavior

It can be proved that in a system of vehicles with limited communication range, it is impossible to enforce the property of liveness if the resources are of unit capacity. Therefore our further consideration will be limited to the sub-class FREE-RANGE*-2+RAS of FREE-RANGE*-RAS, distinguished by that the capacity of each resource R satisfies $C(R) > 1$, $\forall R \in \mathcal{R}$. The following theorem gives the necessary and sufficient condition for safe state transition in the considered RAS class. For the sake of a canonic form of the safety condition, we will assume that completed processes P_i, $i \in 1, \ldots, n$, hold resource $D_{i,l(i)+1} = R_\infty$ and are represented in graph $F(\sigma)$ by the self-loop (R_∞, R_∞).

Theorem 1: Consider the DFSA $G(\Phi)$ of a FREE-RANGE*-2+RAS, a reachable safe state σ, and an event e_{ij} such that the next state $\sigma' = f(\sigma, e_{ij})$ is defined. Then, state σ' is safe iff in graph $F(\sigma')$, there exists a path $q = R^1, R^2, \ldots, R^{l_q}$, $l_q \geq 1$ from resource $R^1 = D_{ij}$, to a resource R^{l_q} that has a free unit[1].

Proof: To show the necessity part of the theorem, notice that if path q does not exist then $j < l(i) + 1$, hence, by Property 1, process P_i is dead. Thus, by Property 3, state σ' is not safe. To prove the sufficiency, notice that under the theorem's assumptions, Property 3 and Property 2 imply that state σ' is safe if process P_i is not dead in σ'. If $j = l(i)+1$, the conclusion is obvious. Otherwise, assume that process P_i is dead in σ'. Then, by Property 1, in graph $F(\sigma')$, there is no path q such that: (*) $p = R^1, R^2, \ldots, R^{l_p}$, $R^1 = D_{ij}$, $l_p > 1$, $R^2 = D_{i,j+1}$, and R^{l_p} has a free unit. However, since there is no dead process in state σ, in graph $F(\sigma)$, there exists a simple path $p^* = R^2, \ldots, R^n$ such that $R^2 = D_{i,j+1}$, $n \geq 1$, and R^n has a free unit. If $R^n \in \{D_{i,j-1}, R_\infty\}$ or $R \neq D_{ij}$ then in $F(\sigma')$, exists path $p = D_{ij}, p^*$ that satisfies (*). Otherwise path $p' = D_{ij}, p^*$ is a cycle in $F(\sigma')$. Hence, if Theorem 1 holds, there exists path $p = p', q$ that satisfies (*). Since in both cases path p exists, process P_i is not dead in σ', and consequently state σ' is safe. ■

Theorem 1 defines the most permissive *liveness enforcing supervisor (LES)* for FREE-RANGE*-2+RAS. The supervisory control consists of checking the number of free units of the required resource and avoidance of unsafe states, which reduces to: (i) calculating the new state $\sigma' = (\sigma, e)$, (ii) checking the existence of the path in graph $F(\sigma')$, specified in Theorem 1, and (iii) in the case of a negative answer, forbidding the state transition from state σ to σ'.

[1]We note that $l_q \geq 1$, i.e., $q = R^1 = R^{l_q}$ is also considered as path.

However, a direct implementation of this policy in a distributed system is impossible due to the lack of synchronization among the agents. To make the control decision, an agent needs to acquire information about the state of the system, which may be no more up-to-date when it reaches the querying agent. Consequently, an additional mechanism is required to ensure the *robustness* of the supervisory control, i.e., independence of the decision result on the state changes occurring during the time of acquiring the necessary data and calculating the decision.

Let $G^*(\Phi) = (\Sigma, E, \Gamma^*, f, \sigma_0, \Sigma_M)$ be the live restriction of $G(\Phi)$, i.e., the DFSA obtained from such $G(\Phi)$ by restricting the active-event function Γ to Γ^* such that $e \in \Gamma^*(\sigma)$ iff $e \in \Gamma(\sigma)$ and the safety condition of Theorem 1 holds. An event e will be called *feasible* (*enabled*), in state σ if, respectively, $e \in \Gamma(\sigma)$ ($e \in \Gamma^*(\sigma)$). Then, the following theorem establishes a mechanism to enforce robustness in $G^*(\Phi)$.

Theorem 2: For the DFSA $G^*(\Phi)$ of a FREE-RANGE*-2+RAS, consider an event $e_{kl} \in E$, a reachable state σ such that e_{kl} is feasible, and state σ^v reachable from $\sigma^0 = \sigma$ through an event sequence $s = e_{i_1 j_1}, e_{i_2 j_2}, \ldots, e_{i_v j_v}$, $v \geq 1$, such that for each state $\sigma^r = f(\sigma^{r-1}, e_{i_r j_r})$, $r = 1, \ldots, v$, it is true that: i) D_{kl} has a free unit and ii) if the safety condition of Theorem 1 is satisfied by path $q = R^1, R^2, \ldots, R^{l_q}$ in graph $F(\sigma^r)$ then $R^{l_q} \neq D_{kl}$ or $R^{l_q} = D_{kl}$ and resource D_{kl} has two free units. Then, if event e_{kl} is enabled in state σ, it is also enabled in each state σ^r, $r = 1, \ldots, v$.

Proof: By Theorem 1, if event e_{kl} is enabled in state σ then in graph $F(\sigma')$, $\sigma' = f(\sigma, e_{kl})$, there exists path q from resource D_{kl} to a resource R^l_q that has a free unit. Notice that it is equivalent to the condition that: a) $D_{k,l}$ has two free units in state σ, or b) in graph $F(\sigma)$ there exists path p from D_{kl} to a resource $R^l_p \neq D_{k,l}$ that either has a free unit or $R^l_p = D_{k,l-1}$. Thus, if event e_{kl} is enabled in state σ either (a) or (b) holds. The occurrence of event $e_{i_r j_r}$, $r \in 1, \ldots, v$ may cause that condition (a) becomes no more true, but then, by assumption (ii), in state σ^r condition (b) is true, i.e., there exists the required path p. If path p exists then the occurrence of event $e_{i_r j_r}$), $r \in 1, \ldots, v$ may: 1) leave path p intact, 2) break it due to the deallocation of a unit of resource $D_{i_r j_r - 1}$ from process P_{i_r}, or 3) cause that the end vertex of p has no more a free unit. However, in case (2), vertex $D_{i_r j_r - 1}$ becomes the end of path p, and in case (3), assumptions (i-ii) ensure that path p becomes extended to a vertex $R^{l_q} \neq D_{kl}$ with a free unit, or condition (a) becomes true. Thus, assumptions (i-ii) ensure that event e_{kl} is admissible in each state σ^r, $r = 1, \ldots, v$. Since assumption (i) also ensures that e_{kl} is feasible in each of these states, the theorem is true. ■

Theorem 2 defines constraints on the dynamics of $G^*(\Phi)$ that, as long as they hold, ensure the preservation of the feasibility and safety of a particular event e_{kl}. Thus, temporary enforcement of these constraints for the time required to acquire and process the information necessary to decide whether or not event e_{kl} is enabled, allows the test procedure to be realized concurrently to the operation of other agents, and still guarantee the correct allocation of resource D_{kl}.

An essential feature of this approach is that the constraints given by Theorem 2 can be implemented in the distributed way. That is, when testing the admissibility of the allocation of the next required resource, D_{kl}, agent P_k (i.e., the agent executing process P_k) can force other agents to respect the defined constraints, in particular through *temporary allocation* of D_{kl} to itself [1]. More specifically, in an attempt to progress, agent P_k needs to:

1) Check if there is a free unit of resource D_{kl} and if so, temporarily allocate it to itself.
2) If queried by other agents, respond in both cases, of $D_{k,l-1}$ and of D_{kl}, as holder of the resource.
3) Initiate the process of querying other agents in order to determine whether or not a path q that conditions the safety of stable allocation of resource D_{kl} exists.
4) If information positively verifying the safety test is received, allocate stably D_{kl} (which implies that in finite time $D_{k,l-1}$ will be released).
5) If information negatively verifying the safety test is received, or if after some assumed time period, no or insufficient information for making the decision has been received (this also makes the system robust to packet loss), deallocate D_{kl}. Wait for a period of time and repeat the whole process.

As can be noticed, when following the above protocol, agent P_k makes a unit of resource D_{kl} not available for other agents, and thus enforces the respect of the other agents to the constraints of Theorem 2. However, since the executions of step 1 by different agents can overlap in time, an additional mechanism is needed to prevent the potential allocation of a particular resource by more agents than available free units. This requires that step 1 be executed as a critical section, i.e., only one agent among those attempting to allocate a particular resource can execute step 1 at a time. In a distributed system, a critical section can be implemented as the *Aloha protocol* [14] or its extensions.

III. Multi-Vehicle transport system

The formal concept of the distributed supervisory control, presented in the previous section, was used in the development of the model of a *transport agent*. This model was subsequently employed in a multi-thread computer application (each agent implemented as a single thread) that provides an environment for modeling and simulation of correct-by-construction multi-vehicle transport systems.

A. System representation

The assumed abstraction views the transport system as a collection of agents that execute tasks assigned to them by the dispatcher. The agents share a common motion area that is represented by a grid of white and black square cells (or sectors). White sectors represent the available space, and black sectors represent obstacles. The agents that are currently not active remain in the parking lot, outside of the system. An assignment (or task, or mission) is defined by a sequence of (not necessarily adjacent) target sectors that an agent has to visit before it can leave the system. Each agent knows the map of the motion area, and having received a new assignment, plans a path that avoids obstacles, and allows the agent to visit the targets and retire in the parking lot. The path is specified by a sequence of sectors, where each two subsequent sectors are horizontally or vertically adjacent. It is assumed that each cell can accommodate up to two agents at a time, and that the concurrent motion of two agents in a cell is coordinated by a sensor-based protocol that allows the agents to avoid collisions with each other and return to their planned paths. When moving between sectors, the agents follow the cell-allocation protocol implementing the model described in the previous sections. The agents are equipped in communication devices, which serve them to acquire information about the system state, necessary to make the control decisions.

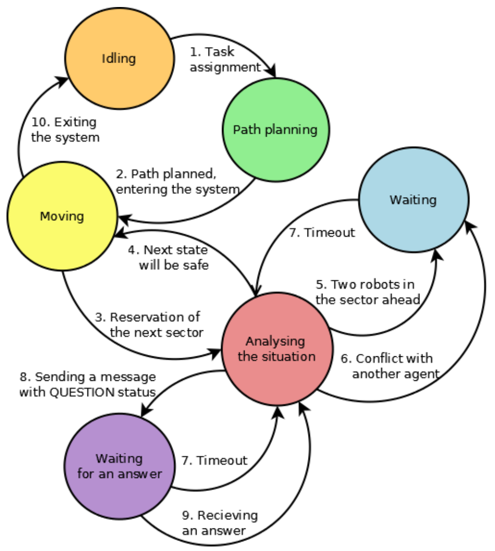

Idling
1. Task assignment
Path planning
10. Exiting the system
2. Path planned, entering the system
Waiting
Moving
7. Timeout
4. Next state will be safe
5. Two robots in the sector ahead
3. Reservation of the next sector
Analysing the situation
6. Conflict with another agent
8. Sending a message with QUESTION status
Waiting for an answer
7. Timeout
9. Recieving an answer

Fig. 1. The DFSA representing the functioning of the transport agent.

B. Transport agent

The abstraction of the transport system characterized above is consistent with the assumptions of the FREE-RANGE*-2+RAS. This allows us to use the $G^*(\Phi)$ model together with the two protocols: the temporary resource allocation (for the time of the testing the safety of its stable allocation), and an aloha-like algorithm for eliminating overlaps in the execution of the critical paths of different agents.

The logic of the operation of each transport agent is given by the DFSA presented in Figure 1. There are five specified states pertaining to an agent, representing five distinguished types of its activity: Idling, Path planning, Moving, Analyzing the situation, Waiting (for availability of the required resource), and Waiting for an answer. Each of the states is correlated with the events that may occur while the agent remains in that state (outgoing arcs) or whose occurrence results in robot's transition to that state (ingoing arcs). When having no current assignment, the agent is in the IDLING state, which corresponds to its presence in the parking lot, and awaits a message from the dispatcher, specifying its new task. The agent remains in this state until the occurrence of the 'task assignment' event, which results in the state transition to PATH PLANNING. This state corresponds to the act of task analysis and planning a path that ensures the accomplishment of the task. In the current implementation of the system, the planning criterion is the shortest path, which is calculated with the employment of the A^* search algorithm [15].

The successful completion of the planning phase enables the agent to enter the motion area, and transit to the MOVING state, where it executes its motion process and handles messages from/to other agents. In this state two different events may occur. The agent may infer that it has completed its task and is currently located at the exit. In such a case it will retire from the system and return to the „parking lot", which corresponds to the *Exiting the system* event). Another event, whose occurrence is possible in this state, is the *Reservation of the next sector*. This event occurs when the agent approaches the next sector at some assumed distance, and describes transition to the ANALYZING THE SITUATION state, where the agent will try to obtain the permission to move to the next sector. To do it, as described in the previous section, the agent will try first to allocate temporarily to itself, or to *reserve*, a unit of this sector in a mutually exclusive manner, using the principles of the Aloha protocol [14]. The agent checks the current allocation state of the next cell by sending a message with a query, and if the cell is fully occupied or there occurs a conflict with another agent, also attempting the allocation of the same resource, the agent transits to the WAITING state, and after some random time returns again to the analysis of the situation.

Otherwise, that is, if the reservation is successful, the agent remains in the ANALYZING THE SITUATION state and begins the process of assessing the safety of the system's state resulting from the stable allocation of the considered cell. The assessment is based on testing the condition of Theorem 1, i.e., the existence of the required path from the resource it attempts to allocate to either the resource that it currently occupies (as it will have a free unit in state σ') or to a resource that currently has a free unit. The agent tries first to evaluate positively this fact based on the messages of the agents that it can contact directly. The positive result corresponds to event (4), which changes its state to MOVING, and proceeds to the new allocated sector. the agent allocates stably the reserved cell and proceeds to it. If the positive evaluation is impossible, the agent initiates a process that prompts other agents to propagate the query and changes state for WAITING FOR AN ANSWER. The message is passed forward the paths of the allocation graph and if a cell with a free unit is found, this information is propagated back to the querying agent. Having received a message, or after some assumed period of time, the agent returns to the ANALYZING THE SITUATION state. In the latter case, the agent deallocates from itself the temporarily allocated cell and after a certain period of time repeats the attempt of its allocation. In the former case, similar as in the case of the positive result of the safety test based on the direct communication, the agent changes again its state to MOVING, allocates stably the reserved cell and proceeds to it. Having entirely entered the new sector, the agent deallocates the previously possessed cell and repeats the same procedure with respect to the next required cell. Although the process of testing the allocation admissibility takes time, and the state of the other agents can change, Theorem 2 ensures that the result of the test is immune to such system state changes. Figure 2 presents the discussed protocol in the form of a block diagram, underlying the subsequent development of the code.

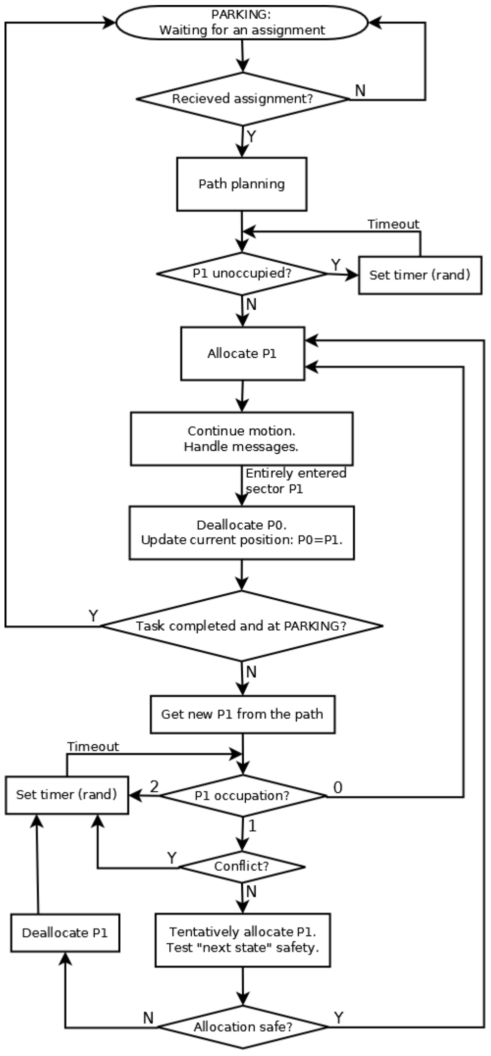
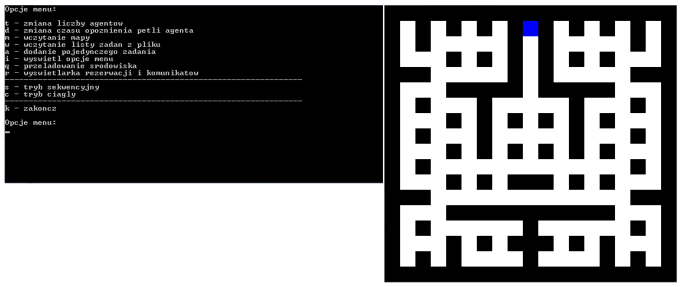

PARKING: Waiting for an assignment
Recieved assignment?
N
Y
Path planning
Timeout
P1 unoccupied?
Y
Set timer (rand)
N
Allocate P1
Continue motion. Handle messages.
Entirely entered sector P1
Deallocate P0. Update current position: P0=P1.
Y
Task completed and at PARKING?
N
Get new P1 from the path
Timeout
Set timer (rand)
2
P1 occupation?
0
1
Y
Conflict?
N
Deallocate P1
Tentatively allocate P1. Test "next state" safety.
N
Allocation safe?
Y

Fig. 2. The basic structure of the agent's control logic.

C. Computer application

Figure 3 presents the interface of the application with an example map of the motion space area. It consists of two collaterally operating windows — the text console (user



```
Opcje menu:

t - zmiana liczby agentow
d - zmiana czasu opoznienia petli agenta
m - wczytanie mapy
w - wczytanie listy zadan z pliku
a - dodanie pojedynczego zadania
i - wyswietl opcje menu
q - przeladowanie srodowiska
r - wyswietlarka rezerwacji i komunikatow
-----------------------------------------------------------------
s - tryb sekwencyjny
c - tryb ciagly
-----------------------------------------------------------------
k - zakoncz

Opcje menu:
_
```

Fig. 3. The interface of the application. The user interface – text console (left), and the graphical visualization of the vehicle motion area (right). The blue sector on the map represents the entrance to the „parking lot".

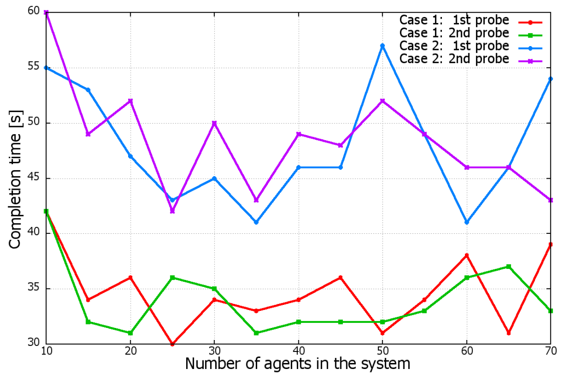


Fig. 4. Completion time of all the assignments in function of the maximal number of agents allowed in the system at a time.

interface) and the window used for the visualization of the agents' concurrent motion. Two more windows are also available - for tracking the resource allocation and message exchange among the agents. The system allows the user to model the transport system by specifying its different parameters. Moreover, the user is presented with several options such as the possibility of: loading a previously designed map, changing the maximal number of vehicles allowed in the system, loading the assignment list from a file and specifying tasks manually from the console, manually from the console, activating or dis-activating the additional visualization modes, adjusting simulation speed, switching between two different types of the visualization - continuous or step by step.

The architecture of the system reflects the asynchronous character of the vehicle system. Each agent, as well as the dispatcher are implemented as independent threads. The task dispatching algorithm assumes the assignment of a task to all available agents, first at the start of the system operation, and then each time when an agent completes its current mission. The system collects the history of the agents' activity and saves it in the format that allows its subsequent processing. The variety of the parameters, modifiable through the interface, provides the user with the opportunity to influence the performance of the system.

To provide an example of the system's operation, two models with their respective maps and task lists were defined. Each list consisted of a hundred of tasks. The employed maps represent two different magazine areas, each having 18 sectors. Figure. 4 depicts the time of completion of all the tasks in function of the maximal number of vehicles allowed to enter the system at a time. For each system (case 1, 2) we performed two independent simulations (probe 1, 2), run on a PC with a 3.4 GHz processor.

IV. Conclusions

The paper further develops the results of [1] towards the provably correct distributed control for multiple vehicle systems. Of a particular interest is the problem of the robust control enforcement, that is the design of a protocol that ensures the correctness of the supervisory control, defined as a function of the system state, in the case when the knowledge of the system is partial and inaccurate.

At the present stage, the discussed solutions and computer application only capture the supervisory layer of the assumed in [1] hybrid control model for the system of autonomous vehicles. Further research is assumed on the communication protocols and motion control for the agents sharing a cell.

References

[1] E. Roszkowska and S. Reveliotis, "A distributed protocol for motion coordination in free-range vehicular systems," *Automatica*, vol. 49, no. 6, pp. 1639–1653, 2013.

[2] *IEEE Robotics & Automation Magazine*, vol. 18, no. 3, 2011, special issue on "Ensuring Correct Behavior. Formal Methods for Hardware and Software Systems".

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

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

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

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

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

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

[9] 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*, vol. 42, pp. 229–243, 2006.

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

[11] S. Reveliotis and E. Roszkowska, "Conflict resolution in free-ranging multivehicle systems: A resource allocation paradigm," *IEEE Trans. Robotics*, vol. 27, no. 2, pp. 283 – 296, 2011.

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

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

[14] N. Abramson, "The aloha system: another alternative for computer communications," in *Proceedings of the fall joint computer conference*, ser. AFIPS '70, 1970, pp. 281–285.

[15] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.