On the liveness of guidepath-based, zone-controlled dynamically routed, closed traffic systems

Elzbieta Roszkowska and Spyros A. Reveliotis

***Abstract*—Zone-controlled, guidepath-based, dynamically routed, closed traffic systems constitute the modelling abstraction for a large set of industrial and public transport systems, including automated guided vehicle and monorail-based material handling systems, railway and urban monorail systems, and also, the complex elevator systems envisioned for the future residential compounds. An important requirement for the traffic flow of these systems is that the vehicles maintain their ability to access every location in the underlying guidepath-network, throughout the entire, presumably infinite, length of the system operation. States in which the system preserves the aforementioned property are said to be *live*. The work presented in this paper provides a novel structural characterization of state liveness for the considered traffic systems that (i) enables the identification of live states while foregoing an extensive enumeration of the underlying behavioral space and (ii) facilitates the design of computationally efficient liveness-enforcing supervisors.**

I. Introduction

Guidepath-based, zone-controlled, dynamically routed, closed traffic systems and their behavioral modeling: As indicated by its title, this paper deals with the characterization of the liveness of a particular class of traffic systems characterized as *guidepath-based, zone-controlled, dynamically routed* and *closed*. In its general definition, a *guidepath-based traffic system* consists of a number of vehicles that travel among a number of locations while following some predetermined paths; these paths are either physically implemented through the system hardware or virtually enforced through the controlling software, and they form a connected *guidepath network*. Links of this guidepath network can be traversed in both directions, but the motion of the vehicles on these links is unidirectional, i.e., a vehicle cannot reverse the direction of its motion while on any certain link. The traffic experienced in such a system is driven by a set of emerging requests for vehicle trips between various pairs of locations in the guidepath network. Vehicles can travel between these location pairs either by following pre-specified routes in the guidepath network, or by developing their route in real-time, based on the prevailing congestion conditions in the network. In the former case, the resultant traffic system is characterized as *"statically routed"*; in the latter, it is said to be *"dynamically routed"*. Another classification of these systems takes place on the basis of the vehicle behavior between two consecutive trip assignments: According to a first scheme, idle vehicles will retire to a particular location of the guidepath network known as the system docking station, and the resulting guidepath-based traffic system is characterized as *"open"*. In the opposite case, vehicles remain in the guidepath network during their idling period, either idling on some guidepath link or moving among various links in order to clear the way for some other vehicles; such a guidepath-based traffic system is characterized as *"closed"*. Finally, in order to avoid the physical collision of the various vehicles, the entire guidepath network is partitioned into a number of segments, and only one vehicle is allowed in any segment at any point in time; in the relevant terminology, these segments are known as *"zones"* and the resulting guidepath-based traffic system is said to be *"zone-controlled"*.

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

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

The work of the first author was supported by the Polish Ministry of Scientific Research and Information Technology under grant no. 5-T12C-02625.

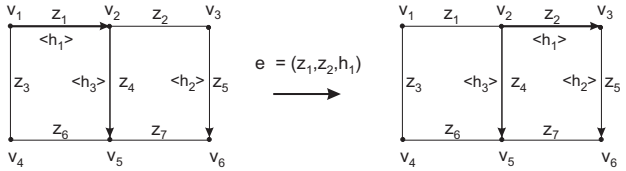

Fig. 1. Transition of vehicle h_1 from zone z_1 to zone z_2.

Under the operational scheme described above, vehicles will always reside in some exclusively allocated zone of the guidepath network, and they will abandon this zone only in order to transition to a neighboring free zone. Furthermore, the vehicle travel through the node of the guidepath network corresponding to the physical intersection between these two zones is only a transient event in the overall evolution of the system traffic. Hence, controlling the traffic flow in such a system in order to maintain liveness can be abstracted to the problem of managing the allocation of the guidepath zones to the various vehicles circulating in it. To facilitate the characterization and the management of this allocation, in the following we shall represent the state s of the considered traffic systems with a *labeled, partially directed graph (PDG)*, where (i) the graph topology is determined by the zoning of the underlying guidepath network, (ii) directed edges correspond to occupied zones and the edge directions indicate the sense of the vehicle motion in these zones, and (iii) the labeling function $l_s()$ is defined on the set of directed edges and it labels each such edge with the identity of the corresponding vehicle. Under this representation, the system state can change through the advancement of a vehicle h from its currently allocated zone, z, to a neighboring unoccupied zone, z'. This advancement will be characterized by the event $e = (z, z', h)$, and it is exemplified in Figure 1. The set of states that can be reached from a given state s through an event sequence $< e_i = (z_{[i]}, z'_{[i]}, h_{[i]}), i = 1, 2, \ldots, >$ will be denoted by $R(s)$, and it will be characterized as the set of *reachable states from state* s. The behavioral subspace induced by $R(s)$ admits a graphical representation where the graph nodes correspond to the states in $R(s)$ and the graph edges correspond to the feasible transitions among these states; we shall denote this graph by $RG(s)$ and we shall refer to it as the *reachability graph* of s.

Characterizing liveness of the considered traffic systems: In this paper, we are especially interested in the characterization of those zone-allocation schemes under which *every* vehicle $h \in H$ maintains its capability to reach *any* zone $z \in Z$ of the guidepath network throughout a presumably *infinite* horizon of the system operation. Zone allocations possessing this property correspond to *live* system states. A complete formal characterization of this concept is as follows:

Definition 1: A state $s \in S$ is *live* if and only if (iff) the reachability graph $RG(s)$ has a strongly connected component that for each vehicle $h \in H$ and each zone $z \in Z$ contains a vertex s' such that $l_{s'}(z) = h$.

The following property establishes that the notion of state liveness provided in Definition 1 is in agreement with the more intuitive characterization of liveness provided at the beginning of this paragraph. It also offers an alternative characterization of liveness that will be useful in the subsequent developments.

Property 1: State $s \in S$ is *live* iff (i) for each vehicle $h \in H$ and each zone $z \in Z$ there exists a state $s' \in R(s)$ such that $l_{s'}(z) = h$, and (ii) state s' is live.

Proof: First consider a state $s \in S$ that is live. Then, according to Definition 1, the reachability graph $RG(s)$ contains a strongly

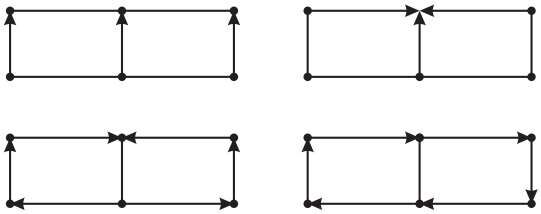

Fig. 2. Examples of an impending deadlock, a deadlock, a livelock, and a live state (clockwise from left top corner).

connected component that for each vehicle $h \in H$ and each zone $z \in Z$ contains a vertex s' such that $l_{s'}(z) = h$. Furthermore, each of these states s' is live, since $RG(s')$ contains the strongly connected component contained in $RG(s)$. Next, consider a state $s \in S$ such that (i) for each vehicle $h \in H$ and each zone $z \in Z$ there exists a state $s' \in R(s)$ with $l_{s'}(z) = h$, and (ii) state s' is live. Since s' is live, $RG(s')$ will contain a strongly connected component that for each vehicle $h \in H$ and each zone $z \in Z$ contains a vertex s'' with $l_{s''}(z) = h$. But $RG(s')$ is a subgraph of $RG(s)$, and therefore, state s is also live. ■

If the traffic system starts its operation in or transitions to a state that is not live, then the ability of its vehicles to visit all the zones of the guidepath network is compromised. The natural cause for this loss of liveness are the phenomena of *deadlock* and *livelock*. A deadlock occurs when two or more vehicles permanently block each other, with the resulting effect that no event associated with any of these vehicles will ever be enabled in any state reachable from the current one. Furthermore, we shall say that a state s is an *impending deadlock*, if it is a deadlock-free state, but there is no infinite event sequence emanating from it that will lead the system only through deadlock-free states; i.e., deadlock is unavoidable from s. On the other hand, a *livelock* is a state s where there are infinite event sequences emanating from it that allow each vehicle $h \in H$ to circulate *ad infinitum* in some specific subset of zones, Z_h, but any attempt by h to enter a zone in $Z \backslash Z_h$ results in a (impending) deadlock. Examples of a live state and of states that are not live are given in Figure 2.

Motivation of the presented research: It should be clear from the above discussion that the liveness of the guidepath-based, zone-controlled traffic systems can be studied through the tools offered by *qualitative discrete event systems (DES) theory* [1], and its specialization in the modelling framework of *sequential resource allocation systems (RAS)* [2], [3]. Indeed a series of results concerning the characterization of liveness and the liveness-enforcing supervision of *open* guidepath-based, zone-controlled traffic systems have already been developed in, e.g., [4], [5], [6]. The open structure of these environments enables the effective characterization of their liveness through the reachability of a target *"home"* state, namely, the state where all vehicles have successfully completed all their missions and are parked in the system docking station. This characterization subsequently enables the episodic decomposition of the system behavior based on its consecutive visits to this "home" state, and the application of results concerning the liveness and the liveness-enforcing supervision of RAS with finite processes [2], [3]. On the other hand, *closed*, zone-controlled, dynamically routed, guidepath-based traffic systems do not possess such easily identifiable "home" states. The perpetual circulation of the various vehicles in the guidepath network renders the identification of a target "home" state that would be appropriate for this class of systems, a challenging, if not a vacuous, proposition. Hence, the characterization of liveness for closed traffic systems must be based on novel concepts and techniques, that will extend the existing RAS theory so that it can model, analyze and eventually control the behavior of RAS with non-terminating processes. The work presented in this paper addresses the aforementioned gap and sets the basis for a formal RAS theory for *closed, zone-controlled, dynamically routed, guidepath-based traffic systems*. At the same time, the key result of this work can be interpreted as an extension of the notion of "home" state to the considered class of traffic systems. From a methodological standpoint, the presented results build upon concepts and techniques concerning the study and manipulation of the PDG structure that was proposed as a natural representational framework of the underlying system dynamics. Hence, in the next section we introduce the PDG theory that is necessary for the subsequent developments. The main body of these developments are presented in Section III, which provides the new criterion for state liveness of the considered class of traffic systems. Finally, Section IV concludes the paper, by summarizing its key contributions and briefly discussing their implications for the design of effective and computationally efficient liveness-enforcing supervisors for these environments. Closing this introductory section, we also notice that preliminary versions of (some of) the presented results can be traced in [7], [8].[1]

II. Partially directed graphs

In this section we formally introduce the concept of the *partially directed graph (PDG)*, and establish some of its properties that will be necessary in the subsequent developments.

Definition 2: A partially directed graph G is formally defined by the triplet $(V, Z, \mathcal{D})$, where: (i) V is the set of the graph *vertices*; (ii) Z is the set of the graph *edges*, with each edge $z \in Z$ being a multi-set of V with cardinality equal to two; and (iii) $\mathcal{D}$ is a partial function on Z such that $\mathcal{D}(z)$ is an element of the multi-set that defines z, for every z that the function is defined.

Edges $z \in Z$ belonging to the domain of function $\mathcal{D}()$ are *directed*, with their direction pointing to the vertex $\mathcal{D}(z)$; the remaining edges are *undirected*. Furthermore, in the following we shall let $U = (V, Z)$ denote the undirected graph induced by PDG G.

Definition 3: Given a partially directed graph $G = (V, Z, \mathcal{D})$:

1) A *path* in G is a sequence $\pi = v_0, z_1, v_1, z_2, \ldots, v_{n-1}, z_n, v_n$, $n \geq 0$, such that, for each $i \in 1, \ldots, n$, $z_i = \{v_{i-1}, v_i\}$, and either edge z_i is undirected or it is directed and points to $\mathcal{D}(z_i) = v_i$. A path is *simple* if all vertices on the path, except possibly for the first and last vertices, are distinct.
2) A *cycle* in G, denoted by c, is a simple path such that $n > 0$ and $v_0 = v_n$.
3) A *joint* between two cycles c and c' is a simple path that is a sub-path of both c and c'.
4) A *pass* between two cycles c and c' is a simple path such that its first vertex lies on c, its last vertex on c', and all of its edges are undirected and are not components of either c or c'.
5) A *chain* in G is the subgraph defined by a sequence $ch =< c_1, \pi_2, c_2, \pi_3, \ldots, \pi_n, c_n >$, $n \geq 1$, such that (i) c_i, $i \in 1, \ldots, n$, are cycles, (ii) π_i, $i = 2, \ldots, n$, are simple paths, and (iii) each path π_i is a joint or a pass between cycles c_{i-1} and c_i.

The concepts introduced by Definition 3 are exemplified in Figure 3. The reader should notice that, according to this definition, all

[1]It should also be noticed that, in principle, the liveness criterion of Definition 1 can be assessed through *model checking* algorithms [9]. However, the complexity of these algorithms is *polynomial in the size of the underlying state space*, and in spite the fact that *symbolic* model checking has enabled their execution on very large state spaces, they do not seem to be particularly amenable to the considered application context, where (i) the underlying state spaces explode extremely fast and (ii) they must be executed in real time.

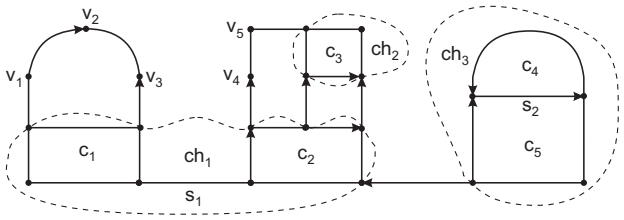

Fig. 3. Chains in an example PDG G.

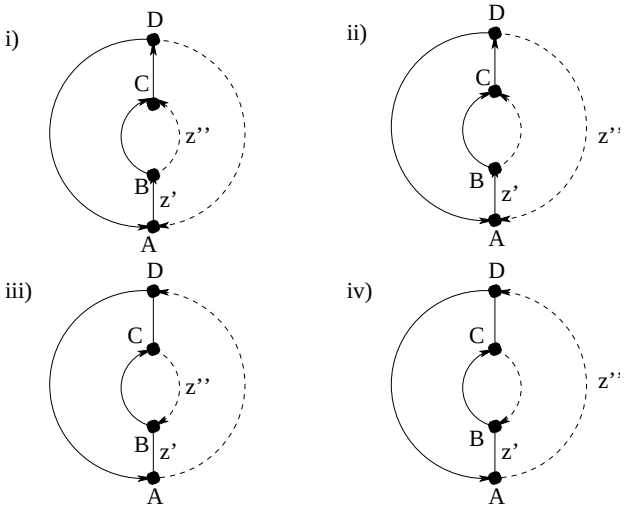

Fig. 4. The sub-case enumeration considered in the proof of Property 2.

the directed edges in a PDG path, π, must point in the same direction, which constitutes the path direction. Also, a vertex can be considered as a *singular* path of zero length. Furthermore, the notion of chain establishes a connectivity relation on the edges of graph G.

Definition 4: Two edges $z, z' \in Z$ in graph G are *chain-connected* or *chained* iff there exists a chain that contains both z and z'. Graph G is *chained* iff every two edges $z, z' \in Z$ are chained.

Chained PDG's have the following properties.

Property 2: The chain-connectivity relation is symmetric and transitive.

Proof: The symmetry of the chain-connectivity follows directly from its definition. To show its transitivity, consider three edges z, z', z'' of a PDG G such that z and z' are chained, and z' and z'' are chained. Then, by Definition 4, there exist two chains in G: $ch = c_1, \pi_2, \ldots, \pi_n, c_n$ and $ch' = c'_1, \pi'_2, \ldots, \pi'_m, c'_m$ such that z is contained by c_1 or π_2, z' is contained by c_n or π_n and by c'_1 or π'_2, and z'' is contained by c'_m or π'_m. Thus, z' can only occur in one of the following configurations: (a) c_n and c'_1 or (b) c_n and π'_2 or (c) π_n and c'_1 or (d) π_n and π'_2. In each of these cases, we can build a chain that contains both z and z''. Next we detail this construction for case (a); the other three cases can be addressed by similar arguments and they are left to the reader.

For case (a), first notice that if edge z'' happens to belong in the intersection of cycles c_n and c'_1, then, the sought chain is actually the original chain ch. The opposite case can be analyzed through the four sub-cases enumerated in Figure 4. In all of the presented sub-cases, cycle c_n is depicted in solid lines, while cycle c'_1 is depicted in dashed lines, except for its parts that intersect with cycle c_n. Furthermore, the line segments and half-circles depicted in this figure should be interpreted as *sequences* of guidepath edges that contain the edges annotated next to them.[2] Then, for sub-cases (i) and (ii), the sought chain can be obtained by appending cycle c'_1 to chain ch, through the joint AB shared by cycles c_n and c'_1. For the sub-cases (iii) and (iv), first notice that the intersecting parts of cycles c_n and c'_1 must consist of undirected edges. Furthermore, the external cycle ADA and the internal cycle BCB are directed. Hence, for sub-case (iii), the sought chain can be obtained by first replacing cycle c_n in chain ch by cycle ADA, and subsequently appending cycle BCB, connected to ADA through any of the passes AB or DC. For sub-case (iv), the sought chain is obtained by replacing cycle c_n in ch by cycle ADA. Finally, the proof for case (a) concludes by noticing that the further sub-case where edge z'' does not belong on c'_1 is effectively covered by constructions similar to those in sub-cases (ii) and (iv). ■

Property 3: In a chained PDG $G = (V, Z, \mathcal{D})$, each edge $z \in Z$ lies on a cycle iff z is not a bridge[3].

Proof: To see the necessity part of this statement, notice that, if z is a bridge, then there is no cycle in the undirected graph U that contains z, so there is no cycle in G either. To prove the sufficiency part, notice that, if z is not a bridge, then there are two possible cases: (i) z is a directed edge, and (ii) z is an undirected edge. Since graph G is chained, then, for each edge $z \in Z$, there exists a chain ch that contains z. If edge z is directed, then it is not an element of a pass in ch, thus it lies on a cycle. If edge z is undirected, then, since z is not a bridge, there exists a cycle c_u in U that contains z. Moreover, the chained structure of G implies that each edge z' that belongs to c_u and is directed in G, lies on a cycle $c_{z'}$ in G. But then, one can identify a cycle in G consisting of edge z, all undirected edges of c_u, and the parts of each $c_{z'}$ that go in the same direction. ■

Property 4: Let b and n denote respectively the number of the bridge edges and the number of the directed edges in a PDG $G = (V, Z, \mathcal{D})$. If G is chained then $n \leq |Z| - b$.

Proof: The definitions of a chain and of a bridge imply that, in any chained graph G, a bridge edge will lie on a pass between two cycles. As a pass consists of undirected edges, there is no chained PDG with $n > |Z| - b$ directed edges. ■

The symmetry and transitivity of the chain-connectivity established by Property 2, further imply that, for any given PDG G, we can distinguish its maximally chained subgraphs, called the *chained components* of G, and establish the notion of *condensation*, $\mathcal{C} = \mathcal{C}(G)$.

Definition 5: A *chained component* of a PDG, G, is a connected subgraph of G, $G_c = (V_c, Z_c, \mathcal{D})$, such that each edge $z \in Z_c$ is chained with edge $z' \in Z$ iff $z' \in Z_c$. The PDG $\mathcal{C} = \mathcal{C}(G)$ that is obtained from graph G by replacing each chained component with a single vertex, will be called the *condensation* of G. Vertices of $\mathcal{C}(G)$ that correspond to chained components will be characterized as the *nodes* of $\mathcal{C}(G)$, while the remaining vertices will be characterized as *simple*.

It follows that the condensed graph $\mathcal{C}(G)$ consists of all the edges of G that do not lie on a chain, and two types of vertices: (i) nodes, that substitute the chained components of G, and (ii) simple vertices, $v \in V$, that are only incident to edges which do not lie on any chain in G. Figure 5 exemplifies these concepts by providing the condensation of the PDG depicted in Figure 3. To understand the structure of

[2]Also, the segment CD in the depicted drawings emphasizes the fact that the (undirected) cycles c_n and c'_1 might intersect through more than one subsets of their edges. The cases where c_n and c'_1 intersect through only one or through more than two segments are covered by trivial modifications of the provided arguments.

[3]We remind the reader that, according to the standard graph-theoretic terminology, a bridge of an *undirected* graph is an edge whose removal disconnects the remaining subgraph. Equivalently, an edge is a bridge if and only if it is not contained in any cycle.

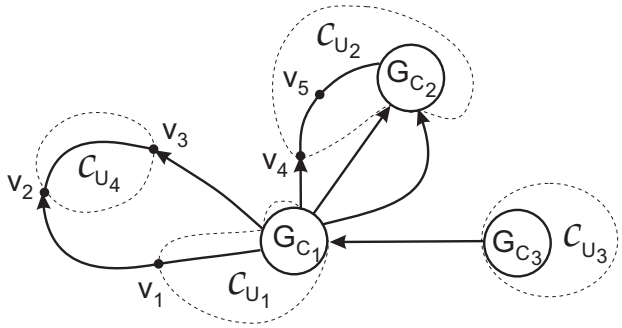

Fig. 5. The condensation, $\mathcal{C}(G)$, of the PDG G depicted in Figure 3, and the u-components of $\mathcal{C}(G)$.

the PDG depicted in Figure 5, notice that each of the three chains depicted in Figure 3 is also a chained component of that PDG; hence, the resulting condensation consists of the five simple vertices, v_1 - v_5, and the three nodes, G_{c_1}, G_{c_2}, and G_{c_3} that condense the three chained components. Next we establish some additional properties and structure exhibited by condensation graphs.

Property 5: Condensation $\mathcal{C} = \mathcal{C}(G)$ is an acyclic graph. Furthermore, each non-singular path in $\mathcal{C}$ that starts and ends with a node contains a directed edge.

Proof: If two edges in G lie on a common cycle, then they are covered by some node of $\mathcal{C}$; thus, $\mathcal{C}$ has no cycles. If two chained subgraphs of G are connected by a path that has only undirected edges, then there exists a chain that includes elements that belong to both subgraphs. Thus, in condensation $\mathcal{C}$ they are covered by the same node. Consequently, each non-singular path in $\mathcal{C}$ that starts and ends with a node contains a directed edge. ■

We will distinguish the subgraphs of condensation $\mathcal{C}(G)$ that (i) contain no directed edges and (ii) are connected to the complement part of $\mathcal{C}(G)$ by directed edges only.

Definition 6: An undirected component (or *u-component*) in condensation $\mathcal{C}(G)$ is a maximal connected subgraph $\mathcal{C}_u$ that contains no directed edges. The edges of $\mathcal{C}(G)$ that point to $\mathcal{C}_u$ are the *inputs* of $\mathcal{C}_u$, and those that point from $\mathcal{C}_u$, are the *outputs* of $\mathcal{C}_u$. $\mathcal{C}_u$ is a *source* if it has no inputs, and a *sink* if it has no outputs. $\mathcal{C}_u$ is a *complex* u-component if it contains a node, and a *simple* u-component otherwise.

Note that a particular case of a simple u-component is a single vertex t such that each edge incident to t is directed. Figure 5 also shows the undirected components of the condensation $\mathcal{C}(G)$ obtained from the graph depicted in Figure 3. Component $\mathcal{C}_{u_4}$ is a simple sink, $\mathcal{C}_{u_2}$ is a complex sink, and $\mathcal{C}_{u_3}$ is a complex source. Two useful properties of u-components are as follows:

Property 6: Each u-component, $\mathcal{C}_u$, is an undirected tree and contains at most one node. Moreover, the u-components are partially ordered by the directed edges of condensation $\mathcal{C}(G)$.

Proof: Since, by Property 5, each non-singular path between any two nodes contains a directed edge, and all the edges of any u-component are undirected, it must contain at most one node. Since condensation $\mathcal{C} = \mathcal{C}(G)$ is an acyclic graph, any subgraph of $\mathcal{C}$ is also acyclic. An undirected, connected graph is acyclic *iff* it is a tree. Since each path between any two u-components consists of directed edges only and $\mathcal{C}$ is acyclic, the u-components are partially ordered. ■

We conclude this section on PDG concepts and properties by outlining an algorithm for assessing the chain connectivity of a given PDG $G = (V, Z, \mathcal{D})$. This algorithm proceeds as follows:

1) First, it converts the given PDG, G, to a digraph, G', by replacing each undirected edge z of it with two directed edges z' and z'' of the opposite directions.
2) Subsequently, it extracts the strongly connected components G'_1, G'_2, …, G'_n of G'.
3) Next, it converts each digraph G'_i computed in Step 2 to a PDG G_i by replacing each edge pair (z', z''), introduced in Step 1, by a single undirected edge z.
4) Finally, at each graph G_i, it removes iteratively all one-degree vertices and their incident edges, until no such vertex is left. The resulting PDGs are the chained components of the original PDG G.

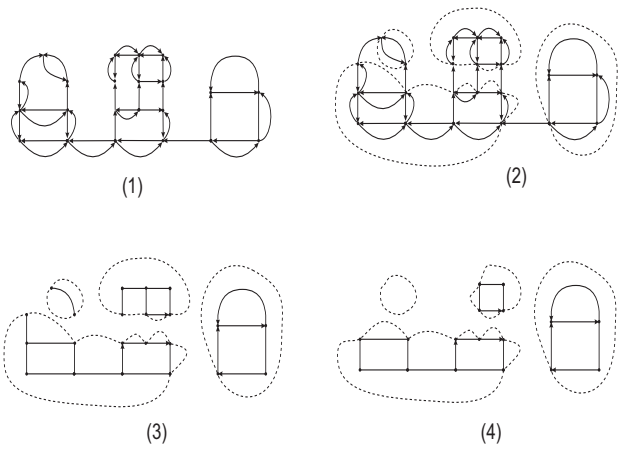

Fig. 6. Assessing the chain connectivity of the PDG of Figure 3.

Figure 6 depicts the application of the above algorithm to the PDG of Figure 3. To see the algorithm correctness, it suffices to notice that each PDG G_i obtained in Step 3 above is the subgraph of PDG G that corresponds to an undirected component of the condensation $\mathcal{C}(G)$, according to Definition 6. Subsequently, Step 4 derives the node of each detected u-component, which essentially constitutes its chained part. Obviously, a given PDG G is chained *iff* the execution of Step 2 in the above algorithm returns only a single strongly connected component $G'_1 = G'$, and the subsequent execution of Step 4 removes no edges from PDG G_1. Furthermore, since the strongly connected components of digraph G' can be calculated with polynomial complexity [10], it is clear from the above that the chain connectivity of a given PGD G can be resolved in polynomial time with respect to the size of G.

The next section builds upon the previously introduced PDG concepts and properties in order to provide an alternative, structural characterization of state liveness for the considered traffic systems, that, in general, will be more easily testable than the characterization provided by Definition 1. The derivation of this new characterization will also suggest a novel and computationally efficient policy for enforcing liveness in the considered class of traffic systems.

III. Structural characterizations of state liveness

The results presented in this section establish that, under the PDG-based representation of the system dynamics that was introduced in the opening section, the liveness of any given state s is strongly related to the reachability from s of another state s' such that the corresponding PDG graph $G(s')$ is chained. The next definition provides a formal characterization of this target structure:

Definition 7: In the considered traffic systems, state $s \in S$ is *chained* iff the condensation $\mathcal{C}(s)$ of the corresponding PDG graph G(s) is a single node (i.e., if $G(s)$ is chained).

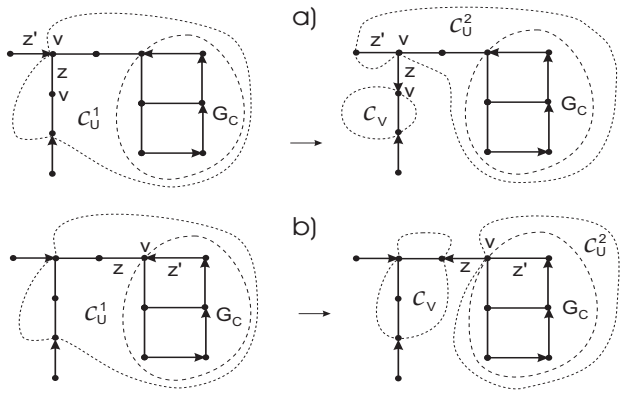

Fig. 7. Illustration of Lemma 2 – in both cases, the transition from state s to state s' results in a simple sink.

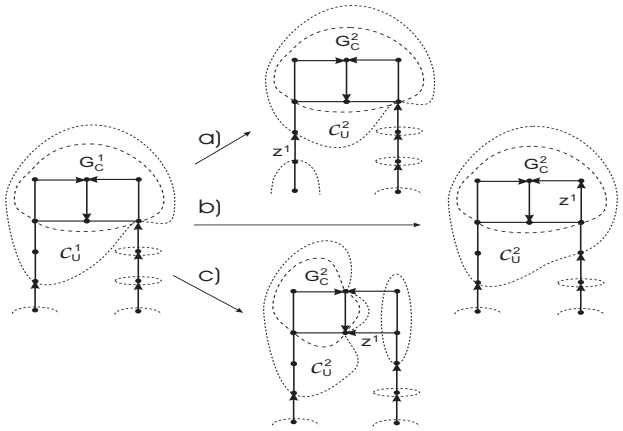

Fig. 8. Illustration of Theorem 1 – in cases (a) and (c) the number of u-components does not change, and in case (b) this number decreases.

Furthermore, in the following it is natural to assume that every vertex $v \in V$ in the underlying guidepath network has a degree at least equal to two, i.e., there are at least two vehicle zones incident to that vertex. The subsequent developments proceed in two stages: In the first stage, we derive the sought liveness characterization in an inductive form, which is very general, but not easily amenable to computation. Hence, in the second stage, we further operationalize the original characterization, by investigating its implications for the state liveness of traffic systems belonging in various sub-classes of practical interest; the derived results are of non-inductive nature, and they enable an effective and straightforward evaluation. The last part of the section also provides a brief discussion on how the obtained results enable the effective and computationally efficient liveness-enforcing supervision of the considered class of traffic systems.

A general structural condition for state-liveness in the considered traffic systems: The next two lemmas are the necessary stepping stones for establishing the key result of this section.

Lemma 1: In the considered traffic systems, if state $s \in S$ is live, then there are no simple sinks in condensation $\mathcal{C}(s)$.

Proof: Let $s \in S$ be a state such that the condensation $\mathcal{C}(s)$ contains a simple sink $\mathcal{C}_u$. Since $\mathcal{C}_u$ contains no nodes, it is a subgraph of $G(s)$, let's say $G_u = \mathcal{C}_u$. Then, according to Property 6, G_u is an undirected tree, and therefore, the vehicles on the directed edges pointing to C_u are destined to an unavoidable deadlock. ■

Lemma 2: Let $\mathcal{C}_u$ be a complex sink in the condensation $\mathcal{C}(s)$ corresponding to the state $s \in S$. Let G_c be the node of $\mathcal{C}_u$, and z' be a directed edge in $G(s)$ that points to the vertex v that is shared with an edge z in $\mathcal{C}_u$. Consider the event $e = (z', z, h)$, where $h = l_s(z')$, and note that it can occur in state s. Then, if there is no path $\pi = v, z, \ldots, G_c$ in the graph $\mathcal{C}(s)$, the state s' that results from the execution of event e in state s, is not live.

Proof: Let $\mathcal{C}_u = \mathcal{C}_u^1$, and note that the occurrence of event e causes the transition of vehicle h from zone z' to zone z. There are two possible cases: (a) edge z' is an input of $\mathcal{C}_u^1$, and (b) z' lies in the node of $\mathcal{C}_u^1$ (see Figure 7 for some corresponding examples). Since, by Property 6, any u-component is an undirected tree, if there is no path $\pi = v, z, \ldots, G_c$ in graph $\mathcal{C}(s)$, then, in both cases, there will be a simple sink, $\mathcal{C}_v$, in graph $\mathcal{C}(s')$. Thus, by Lemma 1, state s' is not live. ■

Note that Lemma 2 implies that a vehicle located on an input of a sink, $\mathcal{C}_u$, can only head towards its node, G_c, and a vehicle located in the node of a sink cannot leave it, if the resulting state is going to be live. Since the number of undirected edges in node G_c is finite, only a finite number of vehicles can be additionally accommodated in G_c. Thus, the only possibility for the vehicles to proceed is by enlarging the sink node so that it covers a bigger subgraph of graph U. This only happens when a path between the sink $\mathcal{C}_u$ and a preceding u-component, say $\mathcal{C}_{u'}$ with node $G_{c'}$, becomes undirected. Then, in the condensation $\mathcal{C}(s')$, of the new state s', $\mathcal{C}_u$ and $\mathcal{C}_{u'}$ become one u-component $\mathcal{C}_{uu'}$ with node $G_{cc'}$ that covers at least the area of G_c, G'_c, and the undirected path that connects them. The next theorem establishes that a necessary condition for any given state s to be live, is the reachability from state s of a *live* state s' in which all u-components become merged into one.

Theorem 1: In the considered traffic systems, state $s \in S$ is live iff the reachability set $R(s)$ contains a live chained state s'.

Proof: For the sufficiency part, simply notice that, if there exists a live state $s' \in R(s)$, then, by Definition 1, state s is also live. To prove the necessity part of this theorem, consider a live state $s \equiv s^1$. If condensation $\mathcal{C}(G(s^1))$ has only one u-component then the lemma holds. Otherwise, by Property 6, condensation $\mathcal{C}(s^1)$ has a nonempty set of sinks, each of which has at least one input edge z and, by Lemma 1, it is a complex u-component. From Lemma 2, no vehicle can leave the sink and the vehicles located on the inputs of the sink can only move towards its node, if the resulting state is going to be live. However, by Property 1, if state s is live, then a transition from s to a live state s' is always possible. Thus, there exists a sink $\mathcal{C}_u^1$ with an input zone occupied by a vehicle h, such that the transition of h to the zone z^1 that lies either in $\mathcal{C}_u^1$ or in the node G_c^1 of $\mathcal{C}_u^1$, is feasible. There are three possible cases of such a transition: (a) If edge z^1 lies in $\mathcal{C}_u^1$ (see Figure 8.a) then, in the resulting state s^2, the original sink reduces to $\mathcal{C}_u^2$ that is a subgraph of $\mathcal{C}_u^1$, and edge z^1 becomes an input of $\mathcal{C}_u^2$, while the structure of node G_c^1 does not change, that is, $G_c^2 = G_c^1$. It follows that, in state s^2, the number of u-components in graph $\mathcal{C}(s^2)$, $k(s^2)$, is equal to the number of u-components in $\mathcal{C}(s^1)$, $k(s^1)$. If edge z^1 lies in the node G_c^1, then, either (b) $G_c^2 = G_c^1$ (see Figure 8.b) or (c) the node G_c^1 gets split by edge z^1 into two subgraphs (see Figure 8.c). In case (b), since the edge left by the vehicle becomes undirected, the u-component that contained its tail in graph $\mathcal{C}(s)$ and $\mathcal{C}_u^1$ unite into one u-component $\mathcal{C}_u^2$. Consequently, $k(s^2) < k(s^1)$. In case (c), edge z^1 becomes an input to a new complex sink $\mathcal{C}_u^2$, which is a subgraph of $\mathcal{C}_u^1$. Hence, in this case, $k(s^2) = k(s^1)$. Since the number of edges in any sink is finite, a sink cannot get split indefinitely, and therefore, in graph $\mathcal{C}(s^1)$, there exists a sink, $\mathcal{C}_u^1$, such that, in some state s^n reachable from s^1, a vehicle enters its node according to case (b), and $k(s^n) < k(s^1)$. Then, it follows by induction that there exists a live state $s' \in R(s)$ such that the condensation $\mathcal{C}(s')$ has only one u-component. Since by Property 6, $\mathcal{C}_u$ is a tree, and it is assumed that graph U has no terminal vertices, condensation $\mathcal{C}(s')$ is a tree

without any edges, that is, a single node. Consequently, graph $G(s')$ is chained. ■

Further structural characterizations of state liveness – specific cases: In this paragraph we refine the state-liveness characterization of Theorem 1 by establishing a partition of the considered class of traffic systems into four sub-classes, distinguished on the basis of the relative size of the vehicle set H with respect to the number of the non-bridge edges in the underlying guidepath network. More specifically, letting b denote the number of the bridge edges in the graph $U = (V, Z)$ that models the guidepath network, the four considered sub-classes are defined by the following relationships: 1) $|H| > |Z| - b$, 2) $|H| = |Z| - b$, 3) $|H| = |Z| - b - 1$, and 4) $|H| < |Z| - b - 1$. Then, by inferring from Theorem 1 and Property 4, it can be easily seen that traffic systems satisfying Condition 1 have no live states, as no state $s \in S$ is chained. Similarly, a little reflection will reveal that traffic systems satisfying Condition 2 have no live states either, since any chained state $s \in S$ in them will not be live. In essence, Conditions 1 and 2 above characterize *over-congested* systems. On the other hand, for traffic systems satisfying Condition 3, it can be shown that some of their chained states can present livelocks, an element that complicates the assessment of the criterion of Theorem 1. However, it can be argued that traffic systems satisfying Condition 3 can be easily converted to systems satisfying Condition 4, by splitting one of their zones into two new ones; hence, these systems present rather little practical interest. Motivated by the above observations, in the rest of this section we focus on the study of state liveness for traffic systems satisfying Condition 4. The state liveness characterization developed for this system sub-class can be succinctly stated as follows:

Theorem 2: In a traffic system with $|H| < |Z| - b - 1$ vehicles, state $s \in S$ is live iff the reachability set $R(s)$ contains a chained state s'.

Proof: The necessity part of this theorem results immediately from Theorem 1. In order to prove the sufficiency part, we shall show that in a traffic system with $|H| < |Z| - b - 1$ vehicles, for every chained state $s' \in S$ and for each pair $(h, z) \in H \times Z$, there exist (i) a state $s'' \in R(s)$ such that $h = l_{s''}(z)$, and (ii) a chained state s''' reachable from s''. This result further implies that, starting from chained state s', each vehicle h can visit infinitely many times each zone z, and since the reachability set $R(s')$ is finite, this can happen if and only if the reachability graph $RG(s')$ has a strongly connected component that for each vehicle $h \in H$ and each zone $z \in Z$, contains a vertex s'' such that $l_{s''}(z) = h$. But then, Definition 1 implies that state s' is live, and since $s' \in R(s)$, state s is also live.

In order to show the intermediary result mentioned above, it is convenient to follow the movement of not only the vehicles, but also of the empty zones – or *"holes"* – in the *cycles* of the guidepath network of the underlying traffic system. These "holes" will be considered as indistinguishable, and they move in the direction opposite to the vehicles; i.e., when vehicle h moves from zone z to zone z', a "hole" moves from zone z' to zone z. Then, we can make the following observations:

1) Since state s' is chained and $|H| < |Z| - b - 1$, there exist at least two holes that are located on cycles in $G(s')$.
2) A hole located on a cycle c can be transfered to any edge of the cycle, while maintaining the structure of the cycle, by advancing an appropriately selected subset of the vehicles on the cycle. Furthermore, if cycle c is connected with cycle c' by a joint π, then the hole can move to π, and thus reach c'. Similarly, if cycle c is connected by a pass π to another cycle c', then, the hole can be transferred from c to c', and in the resulting configuration, both c and c' maintain their cyclic structure, and path π remains empty. Furthermore, an induction based on the above observations can easily establish that a hole in the chained state s' can move to any zone in the guidepath network so that the resulting state preserves the chain structure of $G(s')$.

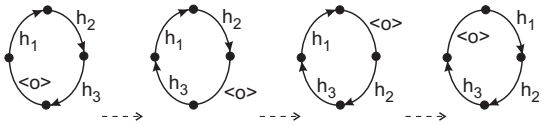

Fig. 9. Illustration of Theorem 2: Transfer of vehicle h_1 to a neighboring zone on a cycle with one hole.

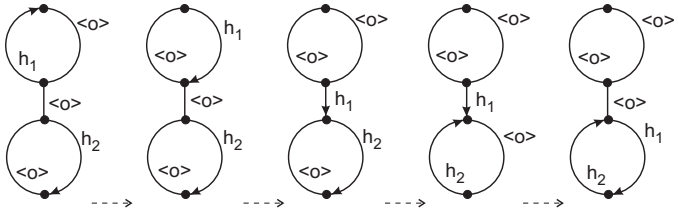

Fig. 10. Illustration of Theorem 2: Transfer of vehicle h_1 between two neighboring cycles, each of which has one hole.

Next consider the vehicle h and the target zone z, and also let z' be the zone allocated to vehicle h in state s'. If zones z and z' belong to the same cycle c, then, according to Observation 1, there exists at least one hole in $G(s')$ and, according to Observation 2, this hole can be transferred to cycle c in a way that preserves the underlying chain structure of $G(s')$. But then, vehicle h can move to any zone in cycle h while preserving the cyclical structure of c, and therefore, it can reach zone z while the resulting state s'' is chained (see Figure 9). If zones z and z' do not belong to a common cycle c, then there must be a chain $< c_1, \pi_2, \ldots, c_{n-1}, \pi_n, c_n >$ such that z' belongs on c_1 and z belongs on c_n or on π_n. Consider first the case where $n = 2$. Then, according to Observation 1, there are at least two holes in state s', and according to Observation 2, these holes can be transferred so that one of them is located in cycle c_1 and the other on cycle c_2, while preserving the chain structure of s'. But then, the reader should be able to convince herself that the presence of a hole in cycle c_1 allows vehicle h to enter the path π_2 and, subsequently, the presence of a hole in c_2 enables it to enter c_2 (see Figure 10). If edge z belongs to the path π_2, it will have been reached by vehicle h by the time it enters c_2. Otherwise, once the vehicle is in c_2, the transfer of a hole back in c_2, if necessary, will allow it to reach zone z. Furthermore, the state s''' resulting from all the above steps maintains the chain structure of $G(s')$. The case where $n > 2$ is addressed by noticing that the above argument for $n = 2$ implies the ability to advance the vehicle between any two consecutive cycles in the chain, while maintaining the underlying chain structure, until the vehicle reaches the final cycle of the chain. ■

Liveness-enforcing supervision of the considered traffic systems: In principle, the result of Theorem 2 provides also a characterization of the *one-step-lookahead, maximally permissive, liveness-enforcing supervisor (LES)*, Δ^*, for the subclass of the considered traffic systems with $|H| < |Z| - b - 1$, which, as observed in the previous paragraph, is the subclass of main practical interest. However, the practical significance of this result is partially compromised by the fact that currently we lack an efficient algorithm for determining the reachability of a chained state from any arbitrary state s. On the other hand, a careful study of the arguments underlying the proof of Theorem 2 will reveal that it is possible to maintain the system liveness while constraining its operation only in those states that are chained or they are generated from chained states by transferring a

single vehicle between two cycles, c_i and c_{i+1}, connected by pass π_{i+1} in some given chain ch; states possessing this last structure are characterized as *semi-chained*. As discussed in Section II, chained states are polynomially recognizable with respect to the structure of the underlying PDG. Furthermore, the definition of the class of semi-chained states implies that for each such state there exists a chained state that can be reached from it through an event sequence that concerns the advancement of a *single* vehicle by no more than $\bar{l}$ zones, where $\bar{l}$ is the length of the maximal simple path in the underlying guidepath network. Therefore, the combined class of chained and semi-chained states can be efficiently (i.e., polynomially) recognized through a controlled partial search for reachable chained states, where the search depth is bounded by $\bar{l}$ and the search is confined to event sequences concerning the advancement of only one vehicle at a time. The resulting supervisor will be denoted by Δ^0. Furthermore, one can envision additional LES, Ψ^i, $i = 0, 2, \ldots$, that will enhance the discriminatory capability of LES Δ^0 by (i) removing the restriction of the single-vehicle-advancement from the search process applied by Δ^0, and (ii) increasing the depth of that search to $\bar{l} + i$. Obviously, the complexity of LES Ψ^i, $i = 0, 2, \ldots$, is super-polynomial with respect to the search depth $\bar{l} + i$. On the other hand, the parameter i defines a notion of *"order"* on the resulting LES class, and it enables some explicit control of the trade-off between operational efficiency and computational tractability.

IV. Conclusions

The key contribution of the work presented in this paper is a novel structural characterization of state liveness for the considered traffic systems, which enables the identification of live states while foregoing an extensive enumeration of the underlying behavioral space. When viewed from a more practical standpoint, the presented results also enable the design of computationally efficient liveness-enforcing supervisors for the considered environments.

Future work will seek (i) to formally analyze the complexity of the decision problem implied by Theorem 2, an important issue since it underlies the efficient implementation of *maximally permissive* supervision for the considered class of systems, (ii) to empirically assess the operational efficiency of the suboptimal supervisors introduced above, and (iii) to explore the implications of the results and the techniques developed in this paper for other classes of guidepath-based traffic systems.

References

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

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

[3] M. Zhou and M. P. Fanti (editors), *Deadlock Resolution in Computer-Integrated Systems*, Marcel Dekker, Inc., Singapore, 2004.

[4] S. A. Reveliotis, "Conflict resolution in AGV systems," *IIE Trans.*, vol. 32(7), pp. 647–659, 2000.

[5] M. P. Fanti, "Event-based controller to avoid deadlock and collisions in zone-controlled AGVS," *IJPR*, vol. 40, pp. –, 2002.

[6] N. Wu and M. Zhou, "Resource-oriented Petri nets in deadlock avoidance of AGV systems," in *Proceedings of the ICRA'01*. IEEE, 2001, pp. 64–69.

[7] E. Roszkowska, "Liveness of states in closed AGV systems with dynamic routing," in *Proceedings of MMAR'02*. IEEE, 2002, pp. 947–952.

[8] E. Roszkowska, "Undirected colored Petri nets for modelling and supervisory control of AGV systems," in *WODES'02*. IEEE, 2002, pp. 135–142.

[9] E. Clarke, O. Grumberg, and D. Peled, *Model Checking*, The MIT Press, 1999.

[10] A. Aho, J. Hopcroft, and J. Ullman, *The design and analysis of computer algorithms*, Adison-Wesley Publishing Company, Reading, Massachusetts, 1974.