# Enforcing Liveness in Autonomous Traffic Management

## Appendix: Proofs of the Theorems *


### Abstract

This file contains the proofs of the theorems in a paper entitled "Enforcing Liveness in Autonomous Traffic Management" that appears in the proceedings of Twenty-Fifth Conference on Artificial Intelligence (AAAI-11).


## Liveness of the Batch Policy with Locking

**Theorem 1** *When an intersection manager in AIM uses the batch policy with locking, a vehicle will eventually get a reservation if every vehicle resends a request message after receiving a reject message.*

*Proof.* The cost function we used in the batch policy is

$$f(wait) = a \times (wait)^b,$$

where $a$ and $b$ are positive constants and *wait* is the estimated amount of time the vehicle has been waiting to enter the intersection. This cost function is a monotonically increasing function. Therefore, the cost strictly increases with the time a vehicle spends waiting at the intersection. If a vehicle $v$ keeps waiting to enter the intersection, the cost of $v$ will eventually exceed the given threshold and get locked.

The batch policy, when the locking mechanism is activated, grants a reservation to the locked vehicle with the highest cost first in a batch, and then grant reservations to a few other vehicles whose trajectories do not conflict with the trajectory of the vehicle with the highest cost as well as each other. If (a) $v$ keep sending requests; (b) all vehicles in front of $v$ keep sending requests; and (c) all vehicles with higher costs on the other incoming roads also keep sending requests. Then the batch policy grants at least one reservation to any locked vehicle in each batch processing. Since the number of vehicles with higher costs than $v$ is finite, $v$ will eventually get a reservation and enter the intersection after a finite number of batch processing. Since no vehicle can stay inside the intersection forever in AIM, $v$ will eventually leave the intersection as well. □


* Author: Tsz-Chiu Au (chiu@cs.utexas.edu). Department of Computer Science, The University of Texas at Austin.



## Nonexistence of Hard Gridlock

**Theorem 2** *If the body of a discretized road network $G$ is strongly connected (every position is reachable from every other position), there can be no hard gridlock in $G$.*

*Proof.* Let $G_{\mathsf{body}} = (P_{\mathsf{body}}, E_{\mathsf{body}})$ be the body of $G$ where $P_{\mathsf{body}} = P + \setminus (P_{\mathsf{src}} \cup P_{\mathsf{dst}})$. Suppose there exists a hard gridlock $G' = (P', E')$ in $G$. Clearly, $(P' \cap P_{\mathsf{dst}}) = \emptyset$ because all positions in $P'$ are occupied while all destinations in $P_{\mathsf{dst}}$ are unoccupied, since any vehicle that entered a destination in the last time step has been removed from the network. Therefore, $P' \subseteq (P_{\mathsf{body}} \cup P_{\mathsf{src}})$. Without loss of generality, consider a path $\tau$ from a position $p_1 \in P'$ to a destination $p_n \in P_{\mathsf{dst}}$. $\tau$ exists because (1) by the definition of destinations there exists $p_{n-1} \in P_{\mathsf{body}}$ such that $p_n \in \mathsf{next}(p_{n-1})$, and (2) either $p_1 = p_{n-1}$ or there exists a directed path from $p_1$ to $p_{n-1}$ since either (a) if $p_1 \in P_{\mathsf{body}}$, $G_{\mathsf{body}}$ is strongly connected; or (b) if $p_1 \in P_{\mathsf{src}}$, there exists $p_2 \in P_{\mathsf{body}}$ such that $p_2 \in \mathsf{next}(p_1)$ by the definition of sources, and there exists a directed path from $p_2$ to $p_{n-1}$ since $G_{\mathsf{body}}$ is strongly connected. Let $\tau$ be $\langle p_1, p_2, \ldots, p_n \rangle$, where $n \geq 2$. On $\tau$ there exists at least one position not in $P'$ (e.g., $p_n$). Let $p_k$ be the first node on $\tau$ that is not in $P'$, where $k \geq 2$ (i.e., $p_i \in P'$ for all $1 \leq i < k$ but $p_k \notin P'$). Then we have $p_k \in \mathsf{next}(p_{k-1}) \subseteq P'$ (by the definition of hard gridlock) but $p_k \notin P'$. Therefore, a contradiction occurred and no hard gridlock exists. □

## Liveness of Road Network without SGDC

In this section, we assume *all* vehicle controllers are deterministic: a vehicle $v$ at a position $p$ will always choose one and only one next position from $(p)$ (i.e., $|\Pi_v(p)| = 1$). A consequence of this assumption is that when a vehicle $v$ is spawned it has already chosen the path towards its destination (we exclude the deterministic controllers that do not lead the vehicles to the destinations). Let $\tau_{\mathsf{chosen}} = \langle p_1, p_2, \ldots, p_n \rangle$ be the *chosen path*, where $p_1 = \mathsf{src}(v)$ and $p_n = \mathsf{dst}(v)$, such that the controller of $v$ only chooses the positions on this path to move into (i.e., $\Pi_v(p_i) = \{p_{i+1}\}$ for $1 \leq i < n$). We assume chosen paths are finite. Since the choices made by deterministic controllers are time-independent, let $\mathsf{chosen}_v(t) = \mathsf{chosen}_v(p_i) = p_{i+1}$ be the *next chosen position* at $p_i$, for $1 \leq i < n$ and any time $t$.

**Lemma 1** *A vehicle $v$ will eventually obtain the right of way of its chosen next position $p = \mathsf{chosen}_v(\mathsf{pos}(v))$ if*

*1) the traffic control mechanism $\Psi_p$ for $p$ is open; and*
*2) $p$ is unoccupied repeatedly until $v$ gets the right of way of $p$ (i.e., the following statement holds until $v$ get the right of way of $p$: at any given time $t$, there exists a time $t' \geq t$ such that $p$ is unoccupied at time $t'$).*

*Proof.* Let $t_0$ be the initial time at which $v$ starts to choose $p$. One of the following four cases occurs:

**Case 1** $p$ is occupied. $v$ cannot move into $p$ and does not have the right of way of $p$;

**Case 2** $p$ is unoccupied and $p$ is unmanaged. $v$ has the right of way to enter $p$;

**Case 3** $p$ is unoccupied, $p$ is managed and $\Psi_p$ gives $v$ the right of way; and

**Case 4** $p$ is unoccupied, $p$ is managed but $\Psi_p$ does not give $v$ the right of way.

In Case 2 and Case 3, the vehicle obtains the right of way at $t_0$. In Case 1 and Case 4, the vehicle does not get the right of way at $t_0$ and has to wait until the next time step $(t_0+1)$ to check whether Case 2 and Case 3 occur. Basically, the checking process repeats until either Case 2 or Case 3 occurs, and the question is whether neither Case 2 nor Case 3 occurs in every time step. Since $p$ is unoccupied repeatedly, Case 1 would not occur indefinitely. Let $t_1, t_2, \ldots$ be the time at which $p$ is unoccupied (before $v$ get the right of way of $p$). If $p$ is unmanaged, $v$ gets the right of way as soon as $p$ is unoccupied at $t_1$ (i.e., Case 2 occurs). If $p$ is managed, $\Psi_p$ will check to see if it can grant the right of way to $v$ at $t_1$, $t_2$, .... Since $\Psi_p$ is open, there exists $i \in \mathbb{N}$ such that $v$ gets the right of way at $t_i$ (i.e., Case 3 occurs and Case 4 would not occur indefinitely). Thus there exists a time at which $v$ obtains the right of way of $p$. □

**Lemma 2** *A vehicle $v$ will eventually move into its chosen next position $p = \mathsf{chosen}_v(\mathsf{pos}(v))$ if*
*1) the coordination mechanism $\Lambda_p$ at $p$ is fair; and*
*2) $v$ gets the right of way of $p$ repeatedly until $v$ moves into $p$ (i.e., the following statement holds until $v$ moves into $p$: at any given time $t$, there exists a time $t' \geq t$ such that $v \in \mathsf{permitted}(p, t')$).*

*Proof.* Let $t_0$ be the initial time at which $v$ starts to choose $p$. Let $t_1, t_2, \ldots$ be the time steps at which $v \in \mathsf{permitted}(p, t_i)$ before $v$ moves into $p$, for $i = 1, 2, \ldots$. At time $t_1$, one of the following three cases occurs for $i = 1$:

**Case 1** $v$ has no competing vehicle and $v$ can enter $p$.

**Case 2** $v$ has competing vehicles and the coordination mechanism $\Lambda_p$ chooses $v$ to move into $p$ (i.e., $\Lambda_p(V_p(t_i)) = v$, where $V_p(t_i)$ is the set of competing vehicles at $p$ at time $t_i$).

**Case 3** $v$ has competing vehicles but $\Lambda_p$ does not choose $v$ to move into $p$.

In Case 1 and Case 2, the vehicle successfully moves into $p$. In Case 3, however, the vehicle has to wait until $t_2$ to see if it gets an opportunity to move into $p$. Basically, the checking process repeats until either Case 1 or Case 2 occurs, and the question is whether neither Case 1 nor Case 2 occurs indefinitely in every time step (that means that Case 3 occurs indefinitely for all $i \in \mathbb{N}$). Since the coordination mechanism $\Lambda_p$ is fair, $\Lambda_p$ would not deny $v$ from moving into $p$ indefinitely, and there exists $j \in \mathbb{N}$ such that $v \in \mathsf{permitted}(p, t_j)$ and $\Lambda_p(V_p(t_j)) = v$. Therefore, Case 3 would not occur indefinitely and $v$ will eventually move into $p$. □

**Lemma 3** *A vehicle $v$ will eventually move into its chosen next position $p = \mathsf{chosen}_v(\mathsf{pos}(v))$ if*
*1) the traffic control mechanism $\Psi_p$ for $p$ is open;*
*2) the coordination mechanism $\Lambda_p$ at $p$ is fair; and*
*3) $p$ is unoccupied repeatedly until $v$ moves into $p$ (i.e., the following statement holds until $v$ moves into $p$: at any given time $t$, there exists a time $t' \geq t$ such that $p$ is unoccupied at time $t'$).*

*Proof.* $v$ can move into $p$ only if $v$ gets the right of way of $p$; thus Condition 3 implies Condition 2 in Lemma 1. Therefore, according to Lemma 1, $v$ will eventually obtain the right of way of $p$ given Conditions 1 and 3. Suppose $v$ obtains its first right of way of $p$ at time $t_1$. One of the following cases occurs at $i = 1$:

**Case 1** $v$ can move into $p$ at $t_i$; and

**Case 2** $v$ cannot move into $p$ at $t_i$.

In Case 2, $v$ will eventually obtain the right of way of $p$ at another time $t_2$. if Case 2 occurs again at $t_2$, $v$ will eventually obtain the right of way of $p$ at another time $t_3$. Thus, $v$ gets the right of way of $p$ repeatedly until Case 1 occurs, and this constitutes Condition 2 in Lemma 2. Whenever $v$ gets the right of way of $p$, This fact, in conjunction with Condition 2 ($\Lambda_p$ is fair), implies that $v$ will eventually move into $p$ according to Lemma 2 (i.e., Case 2 would not occur indefinitely for all $i \in \mathbb{N}$). □

If all traffic control mechanisms are open and all coordination mechanisms are fair, Lemma 3 provides us the key condition to prove that a vehicle can make progress on its chosen path towards its destination, namely whether the chosen next position is unoccupied repeatedly until the vehicle moves into it. It is easy to show that if SGDCs occur, vehicles on the SGDCs cannot make progress because their chosen next positions are always occupied. An important question for us, however, is that if SGDCs do *not* occur at all times (due to the topology of the road network and/or the traffic control mechanisms), is it possible for some vehicles to fail to make any progress on its chosen path? We will show that no vehicle could fail to make any progress in Lemma 4 and Lemma 5 below.

**Definition 1** *A* dependency list *starting from an occupied position $p_1$ is a non-empty, finite sequence $\langle p_1, p_2, \ldots, p_n \rangle$ of occupied positions such that $\mathsf{chosen}_{v_i}(p_i) = p_{i+1}$ for $1 \leq i < n$ and $p_{n+1} = \mathsf{chosen}_{v_n}(p_n)$ is unoccupied, where $n \geq 1$ and $v_i = \mathsf{veh}(p_i)$ is the vehicle occupying $p_i$ for $1 \leq i \leq n$.*

**Lemma 4** *In the absence of SGDCs, all dependency lists in a road network $G = (P, E)$ are finite and their lengths are at most $|P_{\mathsf{body}}| + 1$, where $G_{\mathsf{body}} = (P_{\mathsf{body}}, E_{\mathsf{body}})$ is the body of $G$.*

*Proof.* Since the in-degrees of sources are zero, at most one position in a dependency list is a source node. Also, no dependency list contains any destinations since vehicles are removed immediately upon arriving at destinations. Suppose

there exists a dependency list $\tau$ whose length is greater than $|P_{\mathsf{body}}|+1$. The number of positions in $\tau$ that belong to the body of $G$ is greater than $|P_{\mathsf{body}}|$. By the pigeonhole principle, at least one position in $P_{\mathsf{body}}$ must appear at least twice in $\tau$. Let $p$ be one such position. Then the sublist between the two occurrences of $p$ in $\tau$ forms a SGDC. Therefore, if no SGDC exists, there is no dependency list whose length is greater than $|P_{\mathsf{body}}|+1$. □

**Lemma 5** *A vehicle will eventually move into its chosen next position if*

1) *the traffic control mechanism $\Psi_p$ is open, for all $p \in (P \setminus P_{\mathsf{src}})$;*
2) *the coordination mechanism $\Lambda_p$ is fair, for all $p \in (P \setminus P_{\mathsf{src}})$; and*
3) *there is no SGDC at any time.*

*Proof.* We are going to prove that the vehicle at any occupied position $p_1$ can move into its chosen next position. Let $\tau_{\mathsf{init}} = \langle p_1, p_2, \ldots, p_n \rangle$, where $p_{k+1} = \mathsf{chosen}_{\mathsf{veh}(p_k)}(p_k)$ for $1 \leq k < n$ and $p_{n+1} = \mathsf{chosen}_{\mathsf{veh}(p_n)}(p_n)$ is unoccupied, be the *initial* dependency list starting from $p_1$.

Let $\mathscr{T}$ be the set of all *possible* dependency lists starting from $p_1$. While we can define a possible dependency list in several different ways, we focus on the set that can be constructed as follows. We first set $\mathscr{T}$ as empty. Then we find a set of non-cyclic directed paths by an exhaustive depth-first search starting from $p_1$ and backtracking when the search process reaches a node on the current path (a cycle is detected) or a terminal node (the destinations). Whenever the search process reaches a destination $p_n$, it inserts all proper prefixes of the current path into $\mathscr{T}$ (i.e., $\langle p_1, p_2, \ldots, p_k \rangle$ for $1 \leq k < n$, where $\langle p_1, p_2, \ldots, p_n \rangle$ is the current path). When the research process finishes, we obtain a set $\mathscr{T}$ of directed paths. Each directed path $\langle p_1, p_2, \ldots, p_k \rangle$ in $\mathscr{T}$ can be considered as a dependency list starting from $p_1$, with a "hypothetical" vehicle $v_i$ at each $p_i$ choosing $p_{i+1}$ as its chosen next position, for $1 \leq i < k$.

Notice that not every dependency list in $\mathscr{T}$ is realizable due to existing vehicles on the road network and traffic control mechanisms; however $\mathscr{T}$ does include all possible dependency lists starting from $p_1$ that would possibly occur under the dynamics of the road network, including the initial dependency list $\tau_{\mathsf{init}}$.

Since there is no SGDC at any time, the length of the longest dependency list in $\mathscr{T}$ is at most $|P_{\mathsf{body}}|+1$ according to Lemma 4. We partition $\mathscr{T}$ according to the length of the dependency list, such that $\mathscr{T} = \cup_{1 \leq i \leq |P_{\mathsf{body}}|+1} \mathscr{T}_i$, where $\mathscr{T}_i$ is the set of all dependency lists in $\mathscr{T}$ those length is $i$. Formally, $\mathscr{T}_i = \{\tau \in \mathscr{T} : |\tau| = i\}$.

We are going to show by backward induction that the following statement on $k$ is true for $1 \leq k \leq |P_{\mathsf{body}}|+1$: for all $\tau = \langle p_1, p_2, \ldots, p_k \rangle \in \mathscr{T}_k$, the vehicle $v_k$ at $p_k$ will eventually move into its chosen next position $p_{k+1} = \mathsf{chosen}_{v_k}(p_k)$.

**Base case** $(k = |P_{\mathsf{body}}|+1)$**:** If $\mathscr{T}_{|P_{\mathsf{body}}|+1}$ is empty, the statement is true when $k = |P_{\mathsf{body}}|+1$. If $\mathscr{T}_{|P_{\mathsf{body}}|+1}$ is not empty, let $\tau = \langle p_1, p_2, \ldots, p_{|P_{\mathsf{body}}|+1} \rangle$ be a dependency list in $\mathscr{T}_{|P_{\mathsf{body}}|+1}$. The chosen next position $p'$ of $v = \mathsf{veh}(p_{|P_{\mathsf{body}}|+1})$ must be a destination since all positions in the body of $G$ are occupied by the vehicles on $\tau$. According to Lemma 3, $v$ will eventually move into $p'$ because 1) $\Psi_{p'}$ is open; 2) $\Lambda_{p'}$ is fair; and 3) destinations are always unoccupied.

**Inductive step:** Assume the statement is true for all $k = j$ where $i \leq j \leq (|P_{\mathsf{body}}|+1)$ for some $2 \leq i \leq |P_{\mathsf{body}}|+1$. We are going to show that the statement is true for $k = i-1$. Consider a dependency list $\tau = \langle p_1, p_2, \ldots, p_{i-1} \rangle \in \mathscr{T}_{i-1}$. Let $v = \mathsf{veh}(p_{i-1})$ be the vehicle at $p_{i-1}$. Let $p_i = \mathsf{chosen}_v(p_{i-1})$ be the chosen next position of $v$. Since $p_i$ is unoccupied, one of the following cases would occur:

**Case 1** $p_i$ is unmanaged, $v$ has no competing vehicle at $p_i$, and $v$ can move into $p_i$;

**Case 2** $p_i$ is unmanaged, $v$ has competing vehicles at $p_i$, and $\Lambda_{p_i}$ chooses $v$ to move into $p_i$;

**Case 3** $p_i$ is unmanaged, $v$ has competing vehicles at $p_i$, and $\Lambda_{p_i}$ does not choose $v$ to move into $p_i$;

**Case 4** $p_i$ is managed, $\Psi_{p_i}$ gives $v$ the right of way of $p_i$, $v$ has no competing vehicle at $p_i$, and $v$ can move into $p_i$;

**Case 5** $p_i$ is managed, $\Psi_{p_i}$ gives $v$ the right of way of $p_i$, $v$ has competing vehicles at $p_i$, and $\Lambda_{p_i}$ chooses $v$ to move into $p_i$;

**Case 6** $p_i$ is managed, $\Psi_{p_i}$ gives $v$ the right of way of $p_i$, $v$ has competing vehicles at $p_i$, and $\Lambda_{p_i}$ does not choose $v$ to move into $p_i$; and

**Case 7** $p_i$ is managed, $\Psi_{p_i}$ does not give $v$ the right of way of $p_i$, and $v$ cannot move into $p_i$.

Thus, there are three possible outcomes:

**Outcome 1** The outcome in Case 1, 2, 4, and 5 is that $v$ moves into $p_i$;

**Outcome 2** The outcome in Case 3 and 6 and one possible outcome in Case 7 is that $v$ cannot move into $p_i$ but some other vehicle moves into $p_i$;

**Outcome 3** Another possible outcome in Case 7 is that $v$ cannot move into $p_i$ and $p_i$ remains unoccupied since $\Psi_{p_i}$ does not give the right of way to any vehicle.

In Outcome 1, $v$ moves into $p_i$ and the inductive statement is true for $k = i-1$.

In Outcome 2, $v$ cannot move into $p_i$ and $p_i$ becomes occupied. Since $p_i$ is occupied, $\tau$ is no longer a dependency list because it is "extended". The extended dependency list, denoted by $\tau'$, is a concatenation of $\tau$ and the dependency list starting at $p_i$. Since the length of $|\tau'|$ is at least $i$ and at most $|P_{\mathsf{body}}|+1$, $\tau'$ belongs to $\mathscr{T}_{|\tau'|}$. By the inductive assumption, the last vehicle in $\tau'$ will eventually move into its chosen next position. Likewise, the second to the last vehicle will also move into its chosen next position and so on, until $p_i$ becomes unoccupied again.

In Outcome 3, $v$ cannot move into $p_i$ but $p_i$ remains unoccupied. Thus, in both Outcome 2 and Outcome 3, $p_i$ eventually becomes unoccupied again, and the possible outcome in the next time step after $p_i$ is once again one of the above three outcomes. After that, if Outcome 1 occurs, the inductive statement is true for $k = i-1$. But if Outcome 2 or

Outcome 3 occur again, $p_i$ will eventually become unoccupied again. Hence, the question is whether the Outcome 2 and Outcome 3 occurs indefinitely in every time step and Outcome 1 does not occur. The answer is no according to Lemma 4: $v$ will eventually move into its chosen next position $p_i$ since 1) $\Psi_{p_i}$ is open; 2) $\Lambda_{p_i}$ is fair; and 3) $p_i$ is unoccupied repeatedly before Case 2 occurs. Therefore, the inductive statement is also true for $k = i-1$ in Outcome 2 and Outcome 3.

**Conclusion:** By strong backward induction, the statement is true for all $k = j$ where $1 \leq j \leq |P'|+1$. When $k = 1$, the vehicle at $p_1$ will eventually move into its chosen next position. □

**Lemma 6** *If a vehicle $v$ can always eventually move into its chosen next position $\mathsf{chosen}_v(p)$ at any position $p \in \tau_{\mathsf{chosen}}$ on its chosen path, $v$ will eventually reach its destination $p_n = \mathsf{dst}(v)$.*

*Proof.* Let the chosen path be $\langle p_1, p_2, \ldots, p_n \rangle$. $v$ is spawned at the source $p_1$ and then moves along the chosen path since it is always able to move into its chosen next position. Since chosen paths are finite, $v$ will eventually arrive at $p_n$. □

**Theorem 3** *Every spawned vehicle will eventually reach its destination if*
*1) all vehicle controllers are deterministic;*
*2) all traffic control mechanisms $\Psi_p$ are open, for all $p \in (P \setminus P_{\mathsf{src}})$;*
*3) all coordination mechanisms $\Lambda_p$ are fair, for all $p \in (P \setminus P_{\mathsf{src}})$; and*
*4) there is no SGDC at any time.*

*Proof.* Let us consider a vehicle $v$ that has just spawned at a source $p_1$. Condition 1 is the assumption we made throughout this section; this implies that $v$ has to follow its chosen path to reach its destination. Conditions 2–4 are the conditions in Lemma 5; thus, according to Lemma 5, $v$ can always eventually move into its chosen next position. Then, by Lemma 6, $v$ will eventually reach its destination by moving along its chosen path. Therefore, every spawned vehicle will eventually reach its destination. □

## Liveness of Road Network without SGSC

In this section, we assume the controllers of *all* vehicles are stochastic. This means that the number of relevant positions in $\Pi_v(p)$ of a vehicle $v$ at a position $p$ can be greater than 1.

**Lemma 7** *The following statement holds until a vehicle $v$ moves into one of its relevant next position: $v$ will eventually obtain the right of way of a relevant next position $p$ if*
*1) the traffic control mechanism $\Psi_p$ for $p$ is open;*
*2) $p$ is unoccupied repeatedly; and*
*3) the controller of $v$ is opportunistic.*

*Proof.* Since $v$ is opportunistic, $v$ will eventually choose a relevant next position $p$ if $p$ is unoccupied repeatedly (until $v$ moves into one of its relevant position, which is not necessarily $p$, and is no longer at its current position $\mathsf{pos}(v)$). Suppose $v$ chooses $p$ at time $t_1$. One of the following three cases will occur at time $t_1$:

**Case 1** $p$ is unmanaged and $v$ has the right of way to move into $v$;

**Case 2** $p$ is managed and $\Psi_p$ gives $v$ the right of way; and

**Case 3** $p$ is managed but $\Psi_p$ does not give $v$ the right of way.

In Case 1 and Case 2, $v$ obtains the right of way of $p$. But in Case 3, $v$ does not obtain the right of way of $p$, and $v$ has to wait until the next time at which $v$ chooses $p$ again. Since $v$ is opportunistic and $p$ will continue to be unoccupied repeatedly after $t_1$, $v$ will eventually choose $p$ again. Basically, the check process repeats until Case 2 occurs, and the question is whether Case 2 occurs eventually. Since $\Psi_p$ is open, whenever $v$ repeatedly chooses to move into $p$ when $p$ is unoccupied, $\Psi_p$ will eventually give $v$ the right of way to enter $p$ and thus Case 2 will eventually occur. □

**Lemma 8** *The following statement holds until a vehicle $v$ moves into one of its relevant next positions: $v$ will eventually move into a relevant position $p$ if*
*1) the traffic control mechanism $\Psi_p$ for $p$ is open;*
*2) the coordination mechanism $\Lambda_p$ at $p$ is fair;*
*3) $p$ is unoccupied repeatedly; and*
*4) the controller of $v$ is opportunistic.*

*Proof.* According to Lemma 7, $v$ will eventually obtain the right of way of $p$ given Condition 1, 3 and 4. Suppose $v$ obtains the right of way of $p$ at time $t_1$. Then either (1) $v$ can move into $p$ at $t_1$ or (2) $v$ cannot move into $p$ at $t_1$. In the latter case, $v$ will eventually obtain the right of way of $p$ at another time $t_2$, and the vehicle may or may not be able to move into $p$ at $t_2$. Basically, the checking process repeats until $v$ moves into $p$, and before that $v$ gets the right of way of $p$ repeatedly. Thus this constitues Condition 2 in Lemma 2. Given that the coordination mechanism $\Lambda_p$ at $p$ is fair, $v$ will eventually move into $p$ according to Lemma 2. □

Assume all traffic control mechanisms are open, all coordination mechanisms are fair, and all vehicle controllers are opportunistic. Lemma 8 provides us the key condition to prove that a vehicle can make progress by moving into one of its relevant next positions: there is some relevant next position that becomes unoccupied repeatedly. We can easily show that if SGSCs occur it is impossible for vehicles on the SGSCs to make any progress because all of their relevant next positions are occupied at all times. An interesting question, however, is whether vehicles can always make progress in the absence of SGSCs. We will show that it is indeed the case in Lemma 12.

The proof of Lemma 12 relies on a concept called a *dependency tree*, which is like a dependency list except that the children of a position on a dependency tree are relevant next positions of the position.

**Definition 2** *A* dependency tree *starting from a position $p_1$ is a tree $T = (P', E')$ where*
*1) $P' \subseteq (P \setminus P_{\mathsf{dst}})$ is a set of occupied positions;*
*2) the children of any position $p \in P'$ in $T$, denoted by $\mathsf{children}(p)$, is the set of all occupied relevant next positions of the vehicle $v$ at $p$ (i.e., $\mathsf{children}(p) \subseteq \Pi_v(p)$, $p'$ are occupied for all $p' \in \mathsf{children}(p)$, and $p''$ are unoccupied for all $p'' \in (\Pi_v(p) \setminus \mathsf{children}(p))$);*

*3)* $E' = \{(p_1, p_2) \in E : p_1, p_2 \in P'\}$*; and*
*4)* $p_1$ *is the root of the tree.*

If $(\Pi_\nu(p) \setminus \mathsf{children}(p))$ is a nonempty set, we say $p$ is *open*. In other words, an open position $p$ on $T$ is one with a nonempty set of unoccupied, relevant next positions. Since if a relevant next position $p'$ is not in $P'$, $p'$ must be unoccupied according to the Condition 2 in Definition 2. We can simplify the definition of open positions as follows:

**Definition 3** *A position $p$ on a dependency tree $T = (P', E')$ is* open *if there exists a relevant next position $p'$ of the vehicle $\nu$ at $p$ that is not in $P'$.*

**Lemma 9** *In the absence of SGSCs, all dependency trees in a road network $G = (P, E)$ have an open position.*

*Proof.* Clearly, if a dependency tree $T$ has no open position, $T$ is a SGSC, since all relevant positions of every vehicle on $T$ are occupied. Conversely, if $T$ is not a SGSC, there must exist an open position in $T$. Therefore, in the absence of SGSCs, all dependency trees on $G$ are not SGSC, and all dependency trees have at least one open position. □

The *depth* of a position $p$ in a dependency tree $T$ is the length of the path from the root of $T$ to $p$. The root of $T$ has a depth of zero. We are interested in the *shallowest* open positions whose depths are the smallest among all open positions in a dependency tree. Notice that a dependency tree can have more than one shallowest open positions since several open positions can have the same depth. The *trunk height* of a dependency tree $T$, denoted by $H(T)$, is the depth of the shallowest open positions in $T$. If $T$ has no open position, we define $H(T) = \infty$.

**Lemma 10** *In the absence of SGSCs, the path $\tau = \langle p_1, p_2, \ldots, p_k \rangle$ from the root $p_1$ of a dependency tree $T$ to a shallowest open position $p_k$ in $T$ must contain no cycle (i.e., $p_i \neq p_j$ for all $1 \leq i < j \leq k$).*

*Proof.* By Lemma 9, $T$ must have an open position, thus $p_k$, one of the shallowest open positions in $T$, exists. Suppose there exists one node $p'$ that occurs at least twice in $\tau$. Let $p_i$ and $p_j$ be the two occurrences of $p'$ on $\tau$, where $1 \leq i < j \leq k$. Note that $p_i = p_j = p'$. Let $T'$ be the dependency trees starting from $p'$. Clearly $T'$ is a subtree of $T$; in fact, $T'$ is also a proper subtree of itself, because $p_j$ is a non-root position in the subtree $T'$ starting at $p_i$. We denote the copy of $T'$ starting at $p_i$ by $T'_i$ and the copy of $T'$ starting at $p_j$ by $T'_j$. Then we look at the depth of $p_k$ in $T'_i$ and $T'_j$. The depth of $p_k$ in $T'_i$ is $H(T) - (i-1)$, and the depth of $p_k$ in $T'_j$ is $H(T) - (j-1)$. Thus, the depth of $p_k$ in $T'_i$ is different from the depth of $p_k$ in $T'_j$ since $i \neq j$. An important fact is that $T'_i$ and $T'_j$ are the same subtree, which implies that 1) $p_k$ occurs again at the depth of $H(T) - (j-1)$ in $T'_i$, and 2) $p_k$ occurs again at the depth of $H(T) - (i-1)$ in $T'_j$. In the former case, the newly found instance of $p_k$ is at the depth of $H(T) - (j-i)$ in $T$, thus it is shallower than the original $p_k$ since $H(T) - (j-i) < H(T)$. But this contradicts the fact that $p_k$ is one of the shallowest open positions. Therefore, no position can occur twice in $\tau$. □

**Lemma 11** *In the absence of SGSCs, $H(T) \leq |P_{\mathsf{body}}|$ for every dependency tree $T$ in $G$.*

*Proof.* By Lemma 9, an open position must exist in a dependency tree $T$ in $G$, thus $H(T)$ is a finite number. Suppose $H(T) > |P_{\mathsf{body}}|$. Let $p_k$ be one of the shallowest open positions in $T$, where $k = H(T) + 1$. Consider path $\tau = \langle p_1, p_2, \ldots, p_k \rangle$ from the root $p_1$ to $p_k$, where $p_i \in (P_{\mathsf{src}} \cup P_{\mathsf{body}})$ for $1 \leq i \leq k$. Since $|\tau| = H(T) + 1 > |P_{\mathsf{body}}| + 1$, there exists a position $p_i \in \tau$ that occurs at least twice in $\tau$ by the pigeonhole principle. The path between the two occurrences of $p_i$ on $\tau$ forms a cycle. However, by Lemma 10, the path from the root of a dependency tree to a shallowest open position cannot contain any cycle. Therefore, $H(T) \leq |P_{\mathsf{body}}|$. □

**Lemma 12** *A vehicle $\nu$ at $p$ will eventually move into one of its relevant next positions in $\Pi_\nu(p)$ if*
*1) the traffic control mechanism $\Psi_{p'}$ is open, for all $p' \in (P \setminus P_{\mathsf{src}})$;*
*2) the coordination mechanism $\Lambda_{p'}$ is fair, for all $p' \in (P \setminus P_{\mathsf{src}})$;*
*3) there is no SGSC at any time; and*
*4) all vehicle controllers are opportunistic.*

*Proof.* We are going to prove that the vehicle $\nu_1$ at any occupied position $p_1$ can move into one of its relevant next positions. Let $\mathscr{T}$ be the set of all *possible* dependency trees in $G$ starting from $p_1$. While we can define a possible dependency tree in many different ways, we focus on the set that can be constructed as follows.

From now on, we will draw a distinction between the vehicle $\nu$ and the vehicle controller $\mu$, such that two different vehicles can have the same vehicle controller, causing both vehicles to behave in the same manner. We define a behavioral model of a vehicle controller as a tuple $(P_{\mathsf{src}}, P_{\mathsf{dst}}, \Pi)$, where $P_{\mathsf{src}}$ is the source at which the vehicle is spawned, $P_{\mathsf{dst}}$ is the destination of the vehicle, and $\Pi$ is a mapping from $(P_{\mathsf{src}} \cup P_{\mathsf{body}})$ to $2^{(P_{\mathsf{body}} \cup P_{\mathsf{dst}})}$ which defines the set $\Pi(p)$ of relevant next positions of the vehicle at each position $p$. Let $\mathscr{U}$ be the set of all possible vehicle controllers in a discretized road network $G$. A *configuration* of $G$ is a pair $(P', U)$ where $P' \subseteq (P_{\mathsf{src}} \cup P_{\mathsf{body}})$ is the set of all occupied positions in $G$ and $U$ is a mapping from $P'$ to $\mathscr{U}$ such that $U(p)$ is the vehicle controller of the vehicle at $p$. [1]

The dynamics of the road network can be captured by the set of next possible configurations of each configuration. Each configuration $C$ has a set of next possible configurations $\mathsf{next}(C)$, which is defined by 1) all possible ways vehicles move into their relevant next positions according to their vehicle controllers (including the cases in which vehicles do not move due to traffic controls or competitions among vehicles), and 2) the vehicle controllers spawned at the unoccupied sources (including the cases in which no vehicle controller is spawned at an unoccupied source). Let $C_1$ be the current configuration. Notice that the controller $\mu_1$ of $\nu_1$ is at $p_1$ in $C_1$. Let $\mathscr{C}$ be the set of all possible future configurations starting from $C_1$. More precisely, $\mathscr{C}$ is a closure of a set $\{C_1\}$ in an operation $\mathsf{next}$ on $\mathscr{C}$. Let $\mathscr{C}' = \{C \in \mathscr{C} : \nu_1 \text{ is at } p_1\}$.

[1] Our definition of configurations does not include the states of traffic control mechanisms and coordination mechanisms. But this definition is sufficient for our proof.

Each configuration $C$ in $\mathscr{C}'$ has a dependency tree $T_C$ starting from $p_1$ (since $p_1$ is occupied by $v_1$ in all configurations in $\mathscr{C}'$). Let $\mathscr{T} = \{T_C : C \in \mathscr{C}'\}$ be the set of all dependency trees of all configurations in $\mathscr{C}'$. Given a dependency tree $T \in \mathscr{T}$, let $\mathsf{next}(T)$ be the set of all possible next dependency trees of $T$, where $\mathsf{next}(T) = \{T_{C'} : \forall C \in \mathscr{C}' \text{ s.t. } T_C = T \text{ and } \forall C' \in \mathsf{next}(C)\}$.

Since there is no SGSC at any time, every dependency tree in $\mathscr{T}$ must contain an open position according to Lemma 9. Furthermore, the trunk heights of all dependency trees in $\mathscr{T}$ are at most $|P_{\mathsf{body}}|$ according to Lemma 11. Therefore, we can partition $\mathscr{T}$ into a finite number of subsets according to the depth of the shallowest open positions: $\mathscr{T} = \cup_{0 \le i \le |P_{\mathsf{body}}|} \mathscr{T}_i$, where $\mathscr{T}_i$ is the set of all dependency trees in $\mathscr{T}$ whose shallowest open positions has a depth of $i$. Formally, $\mathscr{T}_i = \{T \in \mathscr{T} : H(T) = i\}$.

We are going to show by backward induction that the following statement on $k$ is true for $0 \le k \le |P_{\mathsf{body}}|$: for all $T \in \mathscr{T}_k$, at least one of the vehicles at the shallowest open positions will eventually move into one of its relevant next positions.

**Base case** ($k = |P_{\mathsf{body}}|$)**:** If $\mathscr{T}_{|P_{\mathsf{body}}|}$ is empty, the statement is true when $k = |P_{\mathsf{body}}|$. If $\mathscr{T}_{|P_{\mathsf{body}}|}$ is not empty, let $T$ be a dependency tree in $\mathscr{T}_{|P_{\mathsf{body}}|}$. Let $p_k$ be one of the shallowest open positions in $T$. By Lemma 10, the path $\tau$ from the root of $T$ to $p_k$ must contains no cycle. Since $|\tau| = H(T)+1 = |P_{\mathsf{body}}|+1$, $\tau$ contains every position in the body of $G$ as well as one source. Hence, all unoccupied relevant next positions of $p_k$ must be destinations. Since destinations are always unoccupied, the vehicle at $p_k$ will eventually move into one of the unoccupied relevant next positions of $p_k$ according to Lemma 8. Therefore, at least one of the vehicles at the shallowest open positions will eventually move into one of its relevant next positions.

**Inductive step:** Assume the statement is true for all $k = j$ where $i \le j \le (|P_{\mathsf{body}}|)$ for some $1 \le i \le |P_{\mathsf{body}}|$. We are going to show that the statement is also true for $k = i-1$.

Consider a dependency tree $T \in \mathscr{T}_{i-1}$. Let $P_{\mathsf{shallow}}$ be the set of all positions at depths $i-1$ in $T$ (including both open and non-open positions). Notice that 1) all open positions in $P_{\mathsf{shallow}}$ are the shallowest open positions; 2) there exists at least one (shallowest) open position in $P_{\mathsf{shallow}}$; and 3) $P_{\mathsf{shallow}}$ is a finite set. Let $P_{\mathsf{rel}} = \cup_{p \in P_{\mathsf{shallow}}} \Pi_{\mathsf{veh}(p)}(p)$ be the union of the sets of all relevant next positions of the vehicles at all positions in $P_{\mathsf{shallow}}$. Notice that $P_{\mathsf{rel}}$ is also finite and at least one position in $P_{\mathsf{rel}}$ is unoccupied. We are going to prove that the vehicle at one of the positions in $P_{\mathsf{shallow}}$ will eventually move into one of its relevant next positions in $P_{\mathsf{rel}}$.

According to Lemma 8, if there exists a position $p \in P_{\mathsf{shallow}}$ such that one of its next relevant position $p' \in \Pi_v(p) \subseteq P_{\mathsf{rel}}$ is unoccupied repeatedly, then the vehicle $v$ at $p$ will eventually move into $p'$. One way to interpret this statement is to assume there exists an integer $n$ such that $v$ will be able to move into $p'$ on or before the number of times $p'$ becomes unoccupied is $n$. Based on this interpretation, for every $p_j \in P_{\mathsf{rel}}$, let $n_j$ be the maximum number of times that $p_j$ needs to be unoccupied in order for the vehicle at its parent position in $P_{\mathsf{shallow}}$ to move into $p_j$.

By a generalization of the pigeonhole principle, if the total number of unoccupied positions in $P_{\mathsf{rel}}$ exceeds $K = \sum_{p_j \in P_{\mathsf{rel}}} \{n_j - 1\}$, there exists a position $p_{j^*} \in P_{\mathsf{rel}}$ such that the number of times $p_{j^*}$ is unoccupied must be at least $n_{j^*}$, and the vehicle at the parent of $p_{j^*}$ can move into $p_{j^*}$. Therefore, all we need to prove is that the total number of occupancies of the positions in $P_{\mathsf{rel}}$ will eventually exceed $K$.

Let $t_0$ be the initial time. Let $N(t)$ be the total number of unoccupied positions in $P_{\mathsf{rel}}$ at or before time $t$ since $t_0$. Notice that $N(t_0) \ge 1$ since at least one position in $P_{\mathsf{rel}}$ is unoccupied at the initial time. We are going to show that $N(t)$ is an increasing function (but not necessarily a strictly increasing function), and there exists a time $t^*$ such that either (a) $N(t^*) > K$, or (b) a vehicle at a position in $P_{\mathsf{shallow}}$ moves into one of its relevant next positions before time $t^*$.

At time $t_0+1$, there are three possible outcomes:

**Outcome 1** a vehicle at a position in $P_{\mathsf{shallow}}$ moves into one of its relevant next positions in $P_{\mathsf{rel}}$;

**Outcome 2** no vehicle in any position in $P_{\mathsf{shallow}}$ moves into one of its relevant next positions, and some position in $P_{\mathsf{rel}}$ is unoccupied (either some unoccupied position remains unoccupied, or some occupied position become unoccupied); and

**Outcome 3** no vehicle in any position in $P_{\mathsf{shallow}}$ moves into one of its relevant next positions, and all positions in $P_{\mathsf{rel}}$ becomes occupied (all unoccupied positions $P_{\mathsf{rel}}$ are occupied by some vehicles that are *not* on the dependency tree).

In Outcome 1, one of the vehicle at the shallowest open positions in $P_{\mathsf{shallow}}$ moves into one of its relevant next positions in $P_{\mathsf{rel}}$. Therefore, our inductive statement is true for $k = i-1$.

In Outcome 2, we have $N(t_0+1) > N(t_0)$ since the number of unoccupied positions in $P_{\mathsf{rel}}$ at time $t_0+1$ is greater than zero. In Outcome 3, we have $N(t_0+1) = N(t_0)$ because there is no unoccupied position in $P_{\mathsf{rel}}$ at time $t_0+1$.

In both Outcome 2 and Outcome 3, the system has to check to see what the outcome is at time $t_0+2$. The question is whether the checking process keeps going forever and Outcome 1 will not occur at any time. The answer is no because

**Case 1** whenever Outcome 2 occurs, $N(t)$ increases; and

**Case 2** whenever Outcome 3 occurs, some of the positions in $P_{\mathsf{rel}}$ will eventually become unoccupied again according to the inductive assumption, and Outcome 2 will eventually occur, causing $N(t)$ to increase.

To see why Case 2 is true, let's consider the trunk height of the new dependency tree when Outcome 3 occurs. The trunk height (i.e., the depth of the shallowest open positions) of the new dependency tree $T'$ starting from $p_1$ will be larger than $i-1$, and thus $T' \in \mathscr{T}_{i'}'$ for some $i' > i-1$. By our inductive assumption, at least one of the shallowest open positions in $T'$ will eventually move into one of its relevant next positions. If the new trunk height after the movement remains larger than $i-1$, one shallowest open positions will move again. Thus, eventually the trunk height will become $i-1$,

and one of the positions in $P_{\mathsf{rel}}$ will eventually become unoccupied again.

Thus in both Case 1 and Case 2, Outcome 2 occurs again and $N(t)$ increases. Unless Outcome 1 occurs, Outcome 2 will occur repeatedly and $N(t)$ will eventually exceed $K$ at some time $t^*$. Then at time $t^*+1$ Outcome 1 *must* occur according to Lemma 8, and one of the vehicle at the shallowest open positions in $P_{\mathsf{shallow}}$ moves into one of its relevant next positions in $P_{\mathsf{rel}}$. Thus our inductive statement is true for $k = i-1$.

**Conclusion:** By strong backward induction, the statement is true for all $k = j$ where $0 \leq j \leq |P_{\mathsf{body}}|$. Therefore, when $k = 0$, the vehicle $v_1$ at $p_1$ will eventually move into one of its relevant next positions. □

**Lemma 13** *If a vehicle $v$ with a progressive controller can always eventually move into one of its relevant next positions in $\Pi_v(p)$ for any position $p \in P$, $v$ will eventually reach its destination $p_n = \mathsf{dst}(v)$.*

*Proof.* Since $v$ is progressive, at any time $t$ the positions visited by $v$ constitute a prefix $\langle p_1, p_2, \ldots, p_i \rangle$ of at least one noncyclic path from the source to the destination, where $i \geq 1$ and $p_1 = \mathsf{src}(v)$, and $p_i = \mathsf{pos}(v)$ is the position of $v$ at time $t$. In fact, $\langle p_1, p_2, \ldots, p_i, p_{i+1} \rangle$ is also a prefix of at least one noncyclic path from the source to the destination, for any next relevant next position $p_{i+1} \in \Pi_v(p)$; otherwise, $p_{i+1}$ is not a relevant next position at $p_i$ since $\langle p_1, p_2, \ldots, p_i, p_{i+1} \rangle$ is not a prefix of a noncyclic path from the source to the destination and $v$ is not progressive.

Hence, if $v$ can always eventually move into one of its relevant next positions, $v$ is always moving on at least one noncyclic path from the source to the destination. Since the road network is finite, all noncyclic paths in $G$ are finite, and therefore $v$ will eventually arrive at $\mathsf{dst}(v)$. □

**Theorem 4** *Every spawned vehicle will eventually reach its destination if*
*1) all traffic control mechanisms are open;*
*2) all coordination mechanisms are fair;*
*3) there is no SGSC at any time;*
*4) all vehicle controllers are opportunistic; and*
*5) all vehicle controllers are progressive.*

*Proof.* By Conditions 1–4, all vehicles will eventually move into one of their relevant next positions at any reachable position, as discussed in Lemma 12. Therefore, all vehicles, equipped with progressive controllers, will eventually reach their destinations according to Lemma 13. □