Chapter 8

Example 4: Collision Avoidance on a General Graph of Tracks

In this chapter, we consider a general track topology involving binary merges and diverges. We first augment the model of the PRT 2000™ to involve a track topology consisting of multiple branches interconnected by Y-shaped merges and diverges — the new model is referred to as the GRAPH-VEHICLES automaton. Then we define the protector GRAPH-PROT that guarantees that none of the vehicles of the GRAPH-VEHICLES automaton collide, assuming that they are all abiding by the speed limit. The GRAPH-PROT protector is defined as the composition of $n(n-1)/2$ separate copies of another protector called GRAPH-PROT-PAIR$_{\{i,i'\}}$, one copy for each unordered pair $\{i, i'\}$ of vehicles of the GRAPH-VEHICLES automaton, for $i, i' \in I, i \neq i'$. Each of these GRAPH-PROT-PAIR$_{\{i,i'\}}$ protectors, for $i, i' \in I, i \neq i'$, is an implementation of a particular instantiation of the abstract protector automaton of Section 3.2 and guarantees that the vehicles i and i' do not collide into each other.

8.1 Augmented Physical Plant: GRAPH-VEHICLES

In this section we augment the model for the system of n vehicles to involve a track topology involving binary merges and diverges. This is done by extending the definition of the location of a vehicle to support a graph of tracks and by introducing an additional internal discrete action which is used to update the location variables of the vehicles as they cross the junction points in the track topology.

The track topology is represented by a directed graph $G = (V, E)$, where V and E denote the sets of vertices and edges of the graph G, respectively. The vertices and edges of the

graph G correspond, respectively, to the junctions and branches of the track topology. Any edge e of the graph G is specified by an ordered pair of vertices that denote the initial and the final vertices of the directed edge e, *i.e.*, $e = \langle v_{init}, v_{final} \rangle$. We use the notation $e.v_{init}$ and $e.v_{final}$ to denote the initial and final vertices of the edge e, respectively. The function $length : E \rightarrow \mathbb{R}^{\geq 0}$ maps an edge to its length. Moreover, the functions $in(v)$, $out(v)$, and $e(v)$ map the vertex v of the graph G to its sets of incoming edges, outgoing edges, and both incoming and outgoing edges, respectively; that is, $in : V \rightarrow \mathcal{P}(E)$, $out : V \rightarrow \mathcal{P}(E)$, and $e : V \rightarrow \mathcal{P}(E)$, with $e(v) = in(v) \cup out(v)$, for all $v \in V$.

The graph G, as defined above, is assumed to satisfy the following conditions:

- All the edges of the graph G are of sufficient length to rule out collisions among vehicles that are neither on identical, nor on contiguous edges; that is, if d_{max} is the maximum sampling period of all the protectors under consideration, the length of each edge in the graph G is greater than $\Delta x_{max} = \dot{c}_{max} d_{max} - \dot{c}_{max}^2 / 2\ddot{c}_{brake}$ — the maximum distance a vehicle can travel if left free for d_{max} time units and instructed to brake thereafter, under the assumption that the vehicle does not collide and is abiding by the speed limit. This restriction rules out the possibility of a vehicle having a d_{max} time unit claim overlap with a vehicle that is more than one edge upstream or downstream.
- All the merges and diverges of the graph G are Y-shaped; that is, for each vertex v in the graph G, it is the case that $\langle |in(v)|, |out(v)| \rangle \in \{\langle 1,1 \rangle, \langle 2,1 \rangle, \langle 1,2 \rangle\}$.
- All cycles must contain at least three edges. This condition ensures that the ordering of the locations of vehicles traveling on successive branches of the track topology is well defined.

Any point on the graph G is represented by a pair consisting of the directed edge of the graph G and the distance of the particular point from the initial vertex of the directed edge. The formal definition of the set L of locations is as follows:

$$L = \{\langle e, x \rangle \mid e \in E \text{ and } x \in [0, length(e)]\}$$

The set of locations is constrained by the length of the edges of the graph G; that is, for $l \in L$ and $l = \langle e, x \rangle$, it is the case that $x \in [0, length(e)]$. We use the notation $l.e$ and $l.x$ to denote the edge and position components of the location l, respectively. It is important to note that, in this representation scheme, the vertices of the graph G have non-unique representations; that is, for all edges $e, e' \in E$, with $e.v_{final} = e'.v_{init}$, it is the case that the location $l = \langle e, length(e) \rangle$ is identical to the location $l' = \langle e', 0 \rangle$. Finally, two locations in L are *comparable* if they are locations either on identical, or on successive edges, *i.e.*, the locations $l, l' \in L$ are comparable only if either $l.e = l'.e$, or $l.e.v_{final} = l'.e.v_{init}$ or $l.e.v_{init} = l'.e.v_{final}$.

Addition of a non-negative scalar y to a location $l \in L$, where $l = \langle e, x \rangle$, maps the location l to the set of locations that can be reached from the location l by traveling a distance y downstream. The set $\langle e, x \rangle + y$ always exists and is defined to be either the singleton $\{\langle e, x + y \rangle\}$, if $x + y \leq \mathit{length}(e)$, or the set $\bigcup_{\varepsilon \in \mathit{out}(e.v_{\mathit{final}})} (\langle \varepsilon, 0 \rangle + (x + y - \mathit{length}(e)))$, otherwise. This definition handles the cases in which the locations $\langle e, x \rangle + y$ extend past a single split or merge, or even multiple splits and/or merges in the track topology.

It is important to note that addition of a location l with a non-negative scalar that is bounded by the minimum distance from the location l to the closest second junction downstream results in a set of locations in which each location l' is comparable to the location l and satisfies the inequality $l \leq l'$; that is, for all $l \in L$, where $l = \langle e, x \rangle$, and $y \in [0, \mathit{length}(e) - x + \min_{\varepsilon \in \mathit{out}(e.v_{\mathit{final}})} \mathit{length}(\varepsilon)]$, the location l is comparable to all locations in $l + y$ and, moreover, $l \leq l'$, for all $l' \in l + y$. In particular, since the length of each edge of the graph G is assumed to be greater than Δx_{max}, addition of a location l with a non-negative scalar $y \leq \Delta x_{max}$ results in a set of locations in which each location l' is comparable to the location l and satisfies the inequality $l \leq l'$.

A closed interval in L is specified with an ordered pair of comparable locations and contains all locations between them, *e.g.*, $[\langle e_1, x_1 \rangle, \langle e_2, x_2 \rangle]$. The partial ordering on comparable locations in L is as follows: $\langle e_1, x_1 \rangle \leq \langle e_2, x_2 \rangle$ if and only if either $x_1 \leq x_2$ and $e_1 = e_2$, or $e_1.v_{\mathit{final}} = e_2.v_{\mathit{init}}$.

Due to the fact that the extent of a vehicle may extend beyond a split in the track topology, we redefine the notion of the section of the track occupied by a particular vehicle as the union of the intervals extending from the current position of the vehicle to a point on the track that is a distance c_{len} downstream; that is, the extent of a vehicle $i \in I$ is the set $E_i = \bigcup_{l_i' \in l_i + c_{len}} [l_i, l_i']$.

In view of breaking the right-of-way symmetry when vehicles approach a merge in the track topology, we must define a prioritization scheme. In Chapter 7, the prioritization was based on the configuration of the merge; namely, the vehicle traveling on the `right` branch of the merge had priority over a vehicle traveling on the `left` branch. In the case of the graph of tracks, the notion of either `left`, or `right` is not well defined. Therefore, we associate a unique priority index to each edge of the graph and give priority to vehicles traveling on the edge whose priority index is greater. Let the function *priority* be an injection from the set of edges E of the graph G, to the set of natural numbers $\mathbb{N}$; that is, $\mathit{priority} : E \to \mathbb{N}$, where for any $e, e' \in E, e \neq e'$, it is the case that $\mathit{priority}(e) \neq \mathit{priority}(e')$.

The new model of the physical system, called GRAPH-VEHICLES, is presented in Figure 8.1. The GRAPH-VEHICLES automaton is the result of augmenting the MERGE-VEHICLES automaton of Chapter 7 so as to involve a general track topology consisting of Y-shaped merges and diverges. Each of the `reset-location`(i) actions, for $i \in I$, is enabled when the vehicle i has reached the final point of the directed edge on which it is traveling, *i.e.*, the vehicle i

Figure 8.1 The GRAPH-VEHICLES automaton.

Actions:

Input:
e, the environment action (stuttering)
$\texttt{protect}(C)_j$, for all $C \in \mathcal{P}(I), j \in J$

Internal:
$\texttt{colliding-pair}(i,i')$, for all $i,i' \in I, i' \neq i$
$\texttt{collision-effects}(i)$, for all $i \in I$
$\texttt{brick-wall}(i)$, for all $i \in I$
$\texttt{reset-location}(i)$, for all $i \in I$

Variables

Internal:
$\ddot{x}_i \in \mathbb{R}$, for all $i \in I$, initially $\ddot{x}_i \in \mathbb{R}$
$\mathit{brake}(i) \in \texttt{Bool}$, for all $i \in I$, initially $\texttt{False}$
$\mathit{brake\text{-}req}(i,j) \in \texttt{Bool}$, for all $i \in I, j \in J$, initially $\texttt{False}$
Output:
$l_i \in L$, for all $i \in I$, initially $l_i \in L$
$\dot{x}_i \in \mathbb{R}$, for all $i \in I$, initially $\dot{x}_i \in \mathbb{R}$
$\mathit{collided}(i,i') \in \texttt{Bool}$, for all $i,i' \in I, i' \neq i$, initially $\texttt{False}$
subject to *VALID*

Discrete Transitions:

$\texttt{protect}(C)_j$
Eff: for all $i \in C$
$\mathit{brake\text{-}req}(i,j) := \texttt{True}$
if $\neg \mathit{brake}(i)$ then
$\mathit{brake}(i) := \texttt{True}$
if $\dot{x}_i = 0$ then $\ddot{x}_i := 0$
else $\ddot{x}_i := \ddot{c}_{brake}$
for all $i \in I - C$
$\mathit{brake\text{-}req}(i,j) := \texttt{False}$
if $\mathit{brake}(i) \wedge (\neg \vee_{k \in J}\ \mathit{brake\text{-}req}(i,k))$ then
$\mathit{brake}(i) := \texttt{False}$
$\ddot{x}_i :\in [\ddot{c}_{min}, \ddot{c}_{max}]$

$\texttt{colliding-pair}(i,i')$
Pre: $\neg \mathit{collided}(i,i')$
$\wedge (E_i \cap E_{i'} \neq \emptyset)$
$\wedge (l_i < \min(E_i \cap E_{i'}))$
Eff: $\mathit{collided}(i,i') := \texttt{True}$
if $(l_i.e \neq l_{i'}.e)$
$\wedge (l_i.e.v_{final} = l_{i'}.e.v_{final})$
then
$\mathit{collided}(i',i) := \texttt{True}$

$\texttt{collision-effects}(i)$
Pre: $\mathit{collided}(*,i,*)$
Eff: $\dot{x}_i :\in \mathbb{R}^{\geq 0}$
$\ddot{x}_i :\in \mathbb{R}$

$\texttt{reset-location}(i)$
Pre: $l_i.x = \mathit{length}(l_i.e)$
Eff: $l_i.e :\in \mathit{out}(l_i.e)$
$l_i.x := 0$

$\texttt{brick-wall}(i)$
Pre: $\texttt{True}$
Eff: $\dot{x}_i := 0$
if $\mathit{brake}(i)$ then $\ddot{x}_i := 0$
else $\ddot{x}_i :\in [0, \ddot{c}_{max}]$

Trajectories:

for all $i,i' \in I, i \neq i'$, $\mathit{collided}(i,i')$ is constant throughout w
for all $i \in I$ and $j \in J$, $\mathit{brake}(i)$ and $\mathit{brake\text{-}req}(i,j)$ are constant throughout w
for all $i,i' \in I, i \neq i'$
the function $w.\ddot{x}_i$ is integrable
for all $t \in T_I$
$w(t).\dot{x}_i = w(0).\dot{x}_i + \int_0^t w(s).\ddot{x}_i\ ds$
$w(t).l_i.x = w(0).l_i.x + \int_0^t w(s).\dot{x}_i\ ds$
if $\neg w.\mathit{collided}(i,i')$
$\wedge (w(t).E_i \cap w(t).E_{i'} \neq \emptyset)$
$\wedge (w(t).l_i < \min(w(t).E_i \cap w(t).E_{i'}))$
then
$t = w.ltime$
if $w(t).l_i.x = \mathit{length}(w(t).l_i.e)$ then
$t = w.ltime$
subject to *VALID*

is located on a vertex of the graph G. At that point in time, its location is nondeterministically set to the initial point of an arbitrary outgoing edge of the vertex on which the vehicle i is located.

The remaining state variables, derived variables, and discrete actions of either the VEHICLES automaton of Chapter 4, or the MERGE-VEHICLES automaton of Chapter 7 as well as the notational shorthand *collided*$(i,*)$, *collided*$(*,i)$, and *collided*$(*,i,*)$, for all $i \in I$, defined for the VEHICLES automaton in Section 4.1, carry over to the GRAPH-VEHICLES automaton unchanged.

As in the case of the MERGE-VEHICLES automaton, the set of input actions of the GRAPH-VEHICLES automaton includes the actions $\texttt{protect}(C)_j$, for $C \in \mathcal{P}(I)$ and $j \in J$; that is, the GRAPH-VEHICLES automaton allows each protector j, for $j \in J$, to brake any subset of the vehicles. However, it is often the case that a protector j, for some $j \in J$, need not schedule but a subset of the actions $\texttt{protect}(C)_j$, for $C \in \mathcal{P}(I)$. In such cases, the protector j is specified as having only the output actions that it is capable of scheduling and the remaining input actions of the GRAPH-VEHICLES automaton on port j are ignored.

The *VALID* set of the GRAPH-VEHICLES automaton is the redefinition of the *VALID* set of the VEHICLES automaton to account for the new track topology representation.

VALID $\subseteq$ *states*(GRAPH-VEHICLES), defined as the set of states of the GRAPH-VEHICLES automaton that satisfy the following conditions:

1. $\nexists\ i, i' \in I, i \neq i'$, such that the set $E_i \cap E_{i'}$ *contains* a positive length closed interval of L.
2. $\dot{x}_i \geq 0$, for all $i \in I$.
3. If $\neg$ *collided*$(*,i,*)$ then $\ddot{x}_i \in [\ddot{c}_{min}, \ddot{c}_{max}]$, for all $i \in I$.
4. If $\neg$ *collided*$(*,i,*) \wedge$ *brake*(i) then if $\dot{x}_i = 0$ then $\ddot{x}_i = 0$ else $\ddot{x}_i = \ddot{c}_{brake}$, for all $i \in I$.

The GRAPH-VEHICLES automaton complies with the assumptions made about the *PP* automaton in Section 3.2.1. The GRAPH-VEHICLES automaton has neither input variables, nor output actions, on any of its ports (Axioms 3.2.1 and 3.2.2, respectively). Moreover, each of the actions $\texttt{protect}(C_j)_j$, for $j \in J$ and $C_j = \{i \mid$ *brake-req*$(i,j) = \texttt{True}\}$, is a no-op input action on port j for any $R \subseteq$ *VALID*. Therefore, the set of no-op input actions on each port $j \in J$ and any $R \subseteq$ *VALID* is non-empty (Axiom 3.2.3).

8.2 Auxiliary Derived Variables and Auxiliary Sets for the GRAPH-VEHICLES Automaton

In this section, we define auxiliary derived variables and sets for the GRAPH-VEHICLES automaton. Most of these variables and sets carry over from either the VEHICLES, or the MERGE-VEHICLES automata. In such cases, the variables and sets are redefined only when their extension to the GRAPH-VEHICLES automaton is not obvious.

As in Chapter 7, we assume that the variables $stop\text{-}dist_i$, $max\text{-}range_i(t)$, and $max\text{-}vel_i(t)$, defined for the VEHICLES automaton in Section 4.3, extend to involve location instead of position variables in the obvious way.

As in the case of the extents of the vehicles of the GRAPH-VEHICLES automaton, we redefine the sections of track owned and claimed by the vehicles in the GRAPH-VEHICLES automaton. While their formal definitions appear in Table 8.1, their informal interpretations follow.

O_i, for $i \in I$, is the section of track that the vehicle i "owns". A vehicle i owns all track intervals that extend from the current position of the vehicle i to the points on the track that the vehicle i can reach even if it is braked immediately. Due to the possibility of such track intervals extending beyond a split in the track topology, the variable O_i is the union of all the intervals that the vehicle i owns.

$C_i(t)$, for $i \in I$ and $t \in \mathbb{R}^{\geq 0}$, is the section of track that the vehicle i "claims" within t time units. A vehicle i claims within t time units all track intervals that extend from the current position of the vehicle i to the points on the track that the vehicle i can reach if braked after t time units and assuming worst-case vehicle behavior up to the point in time when it is braked. Due to the possibility of such track intervals extending beyond a split in the track topology, the variable $C_i(t)$ is the union of all the intervals that the vehicle i claims within t time units.

Henceforth, we assume that the sets $disjoint\text{-}extents(i, i')$, $disjoint\text{-}owned\text{-}tracks(i, i')$, and $disjoint\text{-}claimed\text{-}tracks(i, i', t)$, for $i, i' \in I, i \neq i'$ and $t \in \mathbb{R}^{\geq 0}$, defined for the VEHICLES automaton in Section 4.3, have been extended to the GRAPH-VEHICLES automaton to incorporate the redefinitions of the derived variables used in their definitions. Moreover, we assume that the Lemmas 4.4.1, 4.4.2, 4.4.3, 4.4.4, and 4.4.5 extend to the GRAPH-VEHICLES automaton in the obvious way.

Several auxiliary sets for the GRAPH-VEHICLES automaton are described below. Their formal definitions appear in Table 8.2.

$successive(i, i')$, for $i, i' \in I, i \neq i'$, is the subset of $VALID$ that consists of the states in which the vehicles i and i' are traveling in succession either on the same, or on

Table 8.1 Auxiliary derived variables for the GRAPH-VEHICLES automaton.

$O_i \subseteq L$, for all $i \in I$, defined by

$$O_i = \bigcup_{l_i' \in l_i + (stop\text{-}dist_i + c_{len})} [l_i, l_i']$$

$C_i(t) \subseteq L$, for all $i \in I$ and $t \in \mathbb{R}^{\geq 0}$, defined by

$$C_i(t) = \bigcup_{l_i' \in l_i + (max\text{-}range_i(t) - max\text{-}vel_i(t)^2/(2\ddot{c}_{brake}) + c_{len})} [l_i, l_i']$$

successive directed edges; that is, states in which either the vehicle i is downstream of the vehicle i', or the vehicle i' is downstream of the vehicle i.

adjacent(i, i'), for $i, i' \in I, i \neq i'$, is the subset of *VALID* that consists of the states in which the vehicles i and i' are traveling on different tracks that lead to the same junction; that is, the edges on which the vehicles i and i' are traveling are distinct and have the same final vertex.

proximate(i, i'), for $i, i' \in I, i \neq i'$, is the subset of *VALID* that consists of the states in which the vehicles i and i' are traveling either in succession as defined by the set *successive*(i, i'), or on adjacent tracks as defined by the set *adjacent*(i, i').

remote(i, i'), for $i, i' \in I, i \neq i'$, is the subset of *VALID* that consists of the states in which the vehicles i and i' are traveling neither in succession as defined by the set *successive*(i, i'), nor on adjacent tracks as defined by the set *adjacent*(i, i').

yield-successive(i, i'), for $i, i' \in I, i \neq i'$, is the subset of *successive*(i, i') that consists of the states in which, in the case of a claim overlap among the vehicles i and i', the vehicle i must yield to the vehicle i'. When the vehicles i and i' are traveling in succession, the vehicle i must yield to the vehicle i' if the vehicle i is trailing the vehicle i'. The vehicle i is said to be trailing the vehicle i' if the location of the vehicle i is strictly less than the location of the vehicle i'.

yield-adjacent(i, i'), for $i, i' \in I, i \neq i'$, is the subset of *adjacent*(i, i') that consists of the states in which, in the case of a claim overlap among the vehicles i and i', the vehicle i must yield to the vehicle i'. When the vehicles i and i' are traveling on adjacent incoming tracks, the vehicle i must yield to the vehicle i' if either *only* the vehicle i' owns the upcoming merge point, or the vehicle i' has priority and neither or both vehicles own the merge point.

yield(i, i'), for $i, i' \in I, i \neq i'$, is the subset of *VALID* that consists of the states in which,

Table 8.2 Auxiliary sets for the GRAPH-VEHICLES automaton.

$successive(i,i') \subseteq VALID$, for $i,i' \in I, i \neq i'$, defined by

$$successive(i,i') = \{p \in VALID \mid (p.l_i.e = p.l_{i'}.e) \vee (p.l_i.e.v_{final} = p.l_{i'}.e.v_{init}) \vee (p.l_{i'}.e.v_{final} = p.l_i.e.v_{init})\}$$

$adjacent(i,i') \subseteq VALID$, for $i,i' \in I, i \neq i'$, defined by

$$adjacent(i,i') = \{p \in VALID \mid (p.l_i.e \neq p.l_{i'}.e) \wedge (p.l_i.e.v_{final} = p.l_{i'}.e.v_{final})\}$$

$proximate(i,i') \subseteq VALID$, for $i,i' \in I, i \neq i'$, defined by

$$proximate(i,i') = successive(i,i') \cup adjacent(i,i')$$

$remote(i,i') \subseteq VALID$, for $i,i' \in I, i \neq i'$, defined by

$$remote(i,i') = VALID - proximate(i,i')$$

$yield\text{-}successive(i,i') \subseteq VALID$, for $i,i' \in I, i \neq i'$, defined by

$$yield\text{-}successive(i,i') = \{p \in successive(i,i') \mid p.l_i < p.l_{i'}\}$$

$yield\text{-}adjacent(i,i') \subseteq VALID$, for $i,i' \in I, i \neq i'$, defined by

$$\begin{aligned} yield\text{-}adjacent(i,i') = \{p \in adjacent(i,i') \mid\ & (\langle p.l_i.e, length(p.l_i.e)\rangle \notin p.O_i \\ & \wedge \langle p.l_{i'}.e, length(p.l_{i'}.e)\rangle \in p.O_{i'}) \\ & \vee (\langle p.l_i.e, length(p.l_i.e)\rangle \in p.O_i \\ & \quad \wedge \langle p.l_{i'}.e, length(p.l_{i'}.e)\rangle \in p.O_{i'} \\ & \quad \wedge priority(p.l_i.e) < priority(p.l_{i'}.e)) \\ & \vee (\langle p.l_i.e, length(p.l_i.e)\rangle \notin p.O_i \\ & \quad \wedge \langle p.l_{i'}.e, length(p.l_{i'}.e)\rangle \notin p.O_{i'} \\ & \quad \wedge priority(p.l_i.e) < priority(p.l_{i'}.e))\} \end{aligned}$$

$yield(i,i') \subseteq VALID$, for $i,i' \in I, i \neq i'$, defined by

$$yield(i,i') = yield\text{-}successive(i,i') \cup yield\text{-}adjacent(i,i')$$

in the case of a claim overlap among the vehicles i and i', the vehicle i must yield to the vehicle i' in order to prevent a potential collision between the vehicles i and i'.

The following lemma describes some properties of the sets defined above.

Lemma 8.2.1 *For all $i, i' \in I, i \neq i'$, the following hold:*

1. $VALID = proximate(i, i') \cup remote(i, i')$.
2. $proximate(i, i') \cap remote(i, i') = \emptyset$.
3. $successive(i, i') = yield\text{-}successive(i, i') \cup yield\text{-}successive(i', i)$.
4. $yield\text{-}successive(i, i') \cap yield\text{-}successive(i', i) = \emptyset$.
5. $adjacent(i, i') = yield\text{-}adjacent(i, i') \cup yield\text{-}adjacent(i', i)$.
6. $yield\text{-}adjacent(i, i') \cap yield\text{-}adjacent(i', i) = \emptyset$.

Proof: We prove each of the conditions separately:

1. The condition that $VALID = proximate(i, i') \cup remote(i, i')$, for each $i, i' \in I, i \neq i'$, follows from the definition of the sets $proximate(i, i')$ and $remote(i, i')$.
2. As for the first condition, the condition that $proximate(i, i') \cap remote(i, i') = \emptyset$, for each $i, i' \in I, i \neq i'$, follows from the definition of the sets $proximate(i, i')$ and $remote(i, i')$.
3. For all $i, i' \in I, i \neq i'$, the sets $yield\text{-}successive(i, i')$ and $yield\text{-}successive(i', i)$ are both subsets of the set $successive(i, i')$. Therefore, it suffices to show that any state p in the set $successive(i, i')$, for some $i, i' \in I, i \neq i'$, is either in the set $yield\text{-}successive(i, i')$, or in the set $yield\text{-}successive(i', i)$.

   Let the state p be any state in $successive(i, i')$, for some $i, i' \in I, i \neq i'$. Since $successive(i, i') \subseteq VALID$, it is the case that $p \in VALID$. Therefore, the sections of the track occupied by the vehicles i and i' do not have a positive length closed interval overlap. It follows that it is not possible for their locations to coincide; that is, for any $p \in successive(i, i')$, it is the case that $p.l_i \neq p.l_{i'}$. Therefore, regarding the ordering of the locations of the vehicles i and i', there are only two viable cases:

   (a) $p.l_i < p.l_{i'}$. In this case, $p \in yield\text{-}successive(i, i')$.
   (b) $p.l_{i'} < p.l_i$. In this case, $p \in yield\text{-}successive(i', i)$.

4. If $p \in yield\text{-}successive(i, i')$ then it is the case that $p.l_i < p.l_{i'}$. It follows that $p \notin yield\text{-}successive(i', i)$. Similarly, if $p \in yield\text{-}successive(i', i)$ then it is the case that $p.l_{i'} < p.l_i$. It follows that $p \notin yield\text{-}successive(i, i')$. This suffices.

5. For all $i, i' \in I, i \neq i'$, the sets *yield-adjacent*(i, i') and *yield-adjacent*(i', i) are both subsets of the set *adjacent*(i, i'). Therefore, it suffices to show that any state p in the set *adjacent*(i, i'), for some $i, i' \in I, i \neq i'$, is either in the set *yield-adjacent*(i, i'), or in the set *yield-adjacent*(i', i).

   Let the state p be any state in *adjacent*(i, i'), for some $i, i' \in I, i \neq i'$, and without loss of generality let the vehicle i' be the vehicle traveling on the incoming edge of greater priority, *i.e.*, *priority*$(p.l_i.e) <$ *priority*$(p.l_{i'}.e)$. Regarding the ownership of the merge point by each of the vehicles, there are four cases:

   (a) $\langle \texttt{out}, 0 \rangle \in p.O_i \wedge \langle \texttt{out}, 0 \rangle \in p.O_{i'}$. In this case, $p \in$ *yield-adjacent*(i, i') and $p \notin$ *yield-adjacent*(i', i).

   (b) $\langle \texttt{out}, 0 \rangle \notin p.O_i \wedge \langle \texttt{out}, 0 \rangle \notin p.O_{i'}$. Similarly to above, $p \in$ *yield-adjacent*(i, i') and $p \notin$ *yield-adjacent*(i', i).

   (c) $\langle \texttt{out}, 0 \rangle \notin p.O_i \wedge \langle \texttt{out}, 0 \rangle \in p.O_{i'}$. In this case, $p \in$ *yield-adjacent*(i, i') and $p \notin$ *yield-adjacent*(i', i).

   (d) $\langle \texttt{out}, 0 \rangle \in p.O_i \wedge \langle \texttt{out}, 0 \rangle \notin p.O_{i'}$. In this case, $p \notin$ *yield-adjacent*(i, i') and $p \in$ *yield-adjacent*(i', i).

6. This condition follows from the analysis in the proof of condition 5.

■

8.3 Protection System GRAPH-PROT-PAIR$_{\{i,i'\}}$

The GRAPH-PROT-PAIR$_{\{i,i'\}}$ automata, for $i, i' \in I, i \neq i'$, are vehicle-pair collision protectors and guarantee that the vehicles i and i' do not collide into each other, provided that all the vehicles are abiding by the speed limit and the vehicles of all other vehicle pairs do not collide between themselves. Each of the GRAPH-PROT-PAIR$_{\{i,i'\}}$ protectors, for $i, i' \in I, i \neq i'$, is an implementation of the abstract protector of Section 3.2 specialized to particular definitions of the parameters *PP*, S, R, G, j, and d.

The physical plant automaton, *PP*, is defined to be the GRAPH-VEHICLES automaton of Figure 8.1. The port j and the sampling period d are defined to be the port and sampling period with which the protector GRAPH-PROT-PAIR$_{\{i,i'\}}$ communicates with the GRAPH-VEHICLES automaton. While the port j is assumed arbitrary, the sampling period d is restricted to the set $(0, d_{max}]$, where d_{max} is the maximum protector sampling period presented in Section 8.1. The set of "good" states G is defined to be the set of states in which the vehicles i and i' have not collided into each other, *i.e.*, $G = VALID - P_{collided(i,i')} - P_{collided(i',i)}$. In this chapter, we use the notation $G_{\{i,i'\}}$ to denote the definition of G that is specific to the GRAPH-PROT-PAIR$_{\{i,i'\}}$ protector. The set R is defined to be the

set $R = P_{not\text{-}overspeed} \bigcap \left(\bigcap_{i'',i''' \in I, i'' \neq i''', \{i'',i'''\} \neq \{i,i'\}} G_{\{i'',i'''\}} \right)$. This definition restricts the states of the GRAPH-VEHICLES automaton to states in which all the vehicles are abiding by the speed limit and in which the vehicles of all other vehicle pairs $\{i'', i'''\}$, for $i'', i''' \in I, i'' \neq i''', \{i'', i'''\} \neq \{i, i'\}$, have not collided into each other. The set S is defined to be the set *safe* defined in Section 3.2.1; that is, the set of states of the PP automaton for which a single input action of PP on port j can guarantee that, provided no new input actions on port j are allowed, all subsequently R-reachable states will be in G. Once again, the definition of the set *safe* is specialized to the above definitions of the automaton PP, the sets R and G, and the port j. In this chapter, we use the notation $R_{\{i,i'\}}$ and $S_{\{i,i'\}}$ to refer to the above definitions of the sets R and S.

The GRAPH-PROT-PAIR$_{\{i,i'\}}$ protector automaton is an implementation of the abstract protector automaton Abs(GRAPH-VEHICLES, $S_{\{i,i'\}}, R_{\{i,i'\}}, G_{\{i,i'\}}, j, d$). More precisely, as is the case for the abstract protector Abs_j, we define the GRAPH-PROT-PAIR$_{\{i,i'\}}$ automaton to be the composition of a sensor and a discrete controller automaton. These automata are implementations of their abstract equivalents of Figures 3.2 and 3.3 specialized, however, to the above definitions of the parameters PP, S, R, G, j, and d. The sensor automaton is precisely the specialization of the sensor automaton of Figure 3.2 to the above definitions of the parameters PP, etc. The discrete controller automaton is defined in Figure 8.2.

The braking strategy of the GRAPH-PROT-PAIR$_{\{i,i'\}}$ protector is as follows. The protector is allowed to brake the vehicles i and i' only if the sections of the track they claim in d time units overlap. Given that the vehicles i and i' are indeed involved in such a claim overlap, there are two possible scenarios depending on whether the vehicles i and i' are traveling in succession, or on adjacent tracks. If the vehicles are traveling in succession, then the vehicle i is instructed to brake if it trails the vehicle i'; otherwise, the vehicle i' is instructed to brake. On the other hand, if the vehicles i and i' are traveling on adjacent edges, the vehicle i is instructed to brake either if *only* the vehicle i' owns the merge point, or if both or neither vehicles own the merge point and the vehicle i' is traveling on the edge of greater priority; otherwise, the vehicle i' is instructed to brake.

It is important to note that the abstract protector automaton Abs(GRAPH-VEHICLES, $S_{\{i,i'\}}$, $R_{\{i,i'\}}, G_{\{i,i'\}}, j, d$) complies with the assumptions made about the abstract protector in Section 3.2.1. In particular, since the vehicle location variables, the vehicle velocity variables, and the *collided* variables are output variables of the GRAPH-VEHICLES automaton, the set *safe* is $Y_{\text{GRAPH-VEHICLES}}$-determinable and actions that guarantee safety can be determined from the output variables of the GRAPH-VEHICLES automaton (Axioms 3.2.4 and 3.2.5, respectively). Moreover, the sets $R_{\{i,i'\}}$ and $G_{\{i,i'\}}$ are $Y_{\text{GRAPH-VEHICLES}}$-determinable (Axioms 3.2.6 and 3.2.7, respectively) and the set of start states $S_{\{i,i'\}}$ is a subset of the set *safe* (Axiom 3.2.8), since $S_{\{i,i'\}}$ is defined to be the set *safe*.

In Section 3.1 it was shown that the abstract protector Abs_j guarantees that the physical

Figure 8.2 Discrete controller automaton for the protector GRAPH-PROT-PAIR$_{\{i,i'\}}$.

Actions:	Input:	e, the environment action (stuttering)
		$\texttt{snapshot}(y)_j$, for each valuation y of $Y_{\text{GRAPH-VEHICLES}}$
	Output:	$\texttt{protect}(C)_j$, for $C \in \mathcal{P}(\{i, i'\})$
Variables:	Internal:	$send_j \in \mathcal{P}(\{i, i'\}) \cup null$, initially $null$

Discrete Transitions:

$\texttt{snapshot}(y)_j$
Eff: if $y \notin$ *disjoint-claimed-tracks*(i, i', d) then
if $y \in$ *yield*(i, i') then
$send_j := \{i\}$
else
$send_j := \{i'\}$
else
$send_j := \emptyset$

$\texttt{protect}(C)_j$
Pre: $send_j = C$
Eff: $send_j := null$

Trajectories:
$w.send_j \equiv null$

plant PP remains within G starting from S given R. Similarly, the GRAPH-PROT-PAIR$_{\{i,i'\}}$ automaton guarantees that the GRAPH-VEHICLES automaton remains within $G_{\{i,i'\}}$ starting from $S_{\{i,i'\}}$ given $R_{\{i,i'\}}$. This is shown in the following section.

8.4 Correctness of GRAPH-PROT-PAIR$_{\{i,i'\}}$

The main result to be shown is that GRAPH-PROT-PAIR$_{\{i,i'\}} \leq Abs$(GRAPH-VEHICLES, $S_{\{i,i'\}}$, $R_{\{i,i'\}}, G_{\{i,i'\}}, j, d$). Since both GRAPH-PROT-PAIR$_{\{i,i'\}}$ and Abs(GRAPH-VEHICLES, $S_{\{i,i'\}}$, $R_{\{i,i'\}}, G_{\{i,i'\}}, j, d$) involve the composition of the same sensor automaton with distinct discrete controller automata, Theorem 2.7.4 applies. Therefore, it suffices to show that the discrete controller automaton of the protector GRAPH-PROT-PAIR$_{\{i,i'\}}$ of Figure 8.2 implements the discrete controller automaton DC(GRAPH-VEHICLES, $S_{\{i,i'\}}, R_{\{i,i'\}}, G_{\{i,i'\}}, j, d$) of Figure 3.3. From Theorem 2.6.1, this follows by showing that there exists a simulation relation between the states of the discrete controller automaton of GRAPH-PROT-PAIR$_{\{i,i'\}}$

Table 8.3 Sets used in the correctness proof of GRAPH-PROT-PAIR$_{\{i,i'\}}$.

$W_{\{i,i'\}} \subseteq VALID$, for $i, i' \in I, i \neq i'$, defined by

$$W_{\{i,i'\}} = R_{\{i,i'\}} \cap G_{\{i,i'\}} \cap \textit{disjoint-owned-tracks}(i,i')$$

$B_{\{i,i'\}} \subseteq VALID$, for $i, i' \in I, i \neq i'$, defined by

$$B_{\{i,i'\}} = W_{\{i,i'\}} \cap P_{B_{ij}} \cap P_{B_{i'j}}$$

$V_{(i,i')} \subseteq VALID$, for $i, i' \in I, i \neq i'$, defined by

$$V_{(i,i')} = \{p \in W_{\{i,i'\}} \cap P_{B_{ij}} \mid (p \in \textit{successive}(i,i') \wedge p.l_i < p.l_{i'}) \vee (p \in \textit{adjacent}(i,i') \wedge p.O_i \subseteq [p.l_i, \langle p.l_i.e, \textit{length}(p.l_i.e)\rangle])\}$$

$V_{\{i,i'\}} \subseteq VALID$, for $i, i' \in I, i \neq i'$, defined by

$$V_{\{i,i'\}} = V_{(i,i')} \cup V_{(i',i)}$$

$T_{\{i,i'\}}(t) \subseteq VALID$, for $i, i' \in I, i \neq i'$, and $t \in \mathbb{R}^{\geq 0}$, defined by

$$T_{\{i,i'\}}(t) = R_{\{i,i'\}} \cap G_{\{i,i'\}} \cap \textit{disjoint-claimed-tracks}(i,i',t)$$

and the discrete controller automaton DC(GRAPH-VEHICLES, $S_{\{i,i'\}}, R_{\{i,i'\}}, G_{\{i,i'\}}, j, d$). We first give some set definitions, then prove some lemmas, and finally show the existence of such a simulation relation.

In this section, we use the notation *future*$_{\{i,i'\}}$, *safe*$_{\{i,i'\}}$, *very-safe*$_{\{i,i'\}}$, and *delay-safe*$_{\{i,i'\}}$ to denote the specialization of the function *future*, the sets *safe* and *very-safe*, and the function *delay-safe*, which are defined in Section 3.2.1, to the automaton GRAPH-VEHICLES, the sets $R_{\{i,i'\}}$ and $G_{\{i,i'\}}$, and the the port j of the GRAPH-PROT-PAIR$_{\{i,i'\}}$ protector. Moreover, since the environment action of the GRAPH-VEHICLES automaton is stuttering, its consideration is omitted in all inductive proofs involving the PP automaton.

We proceed by defining several sets that are used in the correctness proof of the protector GRAPH-PROT-PAIR$_{\{i,i'\}}$. For reference, their formal definitions appear in Table 8.3.

Let $W_{\{i,i'\}}$ be the subset of $R_{\{i,i'\}} \cap G_{\{i,i'\}}$ comprised of the states in which the section of the track owned by the vehicle i does not overlap the section of track owned by the vehicle i'; that is, $W_{\{i,i'\}} = R_{\{i,i'\}} \cap G_{\{i,i'\}} \cap$ *disjoint-owned-tracks*(i,i').

Let $B_{\{i,i'\}}$ be the subset of of $W_{\{i,i'\}}$ comprised of the states in which the vehicles i and i' are both being instructed to brake by the protector j; that is, $B_{\{i,i'\}} = W_{\{i,i'\}} \cap P_{B_{ij}} \cap P_{B_{i'j}}$.

Let $V_{(i,i')}$ be the subset of $W_{\{i,i'\}}$ comprised of the states in which the vehicle i is being instructed to brake by the protector j and either the vehicles i and i' are traveling in succession and $l_i < l_{i'}$, *i.e.*, the vehicle i is trailing the vehicle i', or the vehicles i and i' are adjacent and the section of the track owned by the vehicle i is entirely upstream of the merge point $\langle \texttt{out}, 0 \rangle$. Moreover, let $V_{\{i,i'\}}$ be defined as $V_{\{i,i'\}} = V_{(i,i')} \cup V_{(i',i)}$.

Let $T_{\{i,i'\}}(t)$, where $t \in \mathbb{R}^{\geq 0}$, be the subset of $R_{\{i,i'\}} \cap G_{\{i,i'\}}$ comprised of the states in which the section of the track claimed in time t by the vehicle i does not overlap the section of the track claimed in time t by the vehicle i'; that is, $T_{\{i,i'\}}(t) = R_{\{i,i'\}} \cap G_{\{i,i'\}} \cap$ *disjoint-claimed-tracks*(i, i', t).

The following lemma defines the relation among the sets $G_{\{i,i'\}}$, $W_{\{i,i'\}}$, $B_{\{i,i'\}}$, $V_{\{i,i'\}}$, and $T_{\{i,i'\}}(t)$, for $t \in \mathbb{R}^{\geq 0}$.

Lemma 8.4.1 *For all $t, t' \in \mathbb{R}^{\geq 0}$, $t \leq t'$, the following hold:*

1. $T_{\{i,i'\}}(t) \subseteq W_{\{i,i'\}} \subseteq G_{\{i,i'\}}$.
2. $V_{\{i,i'\}} \subseteq W_{\{i,i'\}} \subseteq G_{\{i,i'\}}$.
3. $B_{\{i,i'\}} \subseteq W_{\{i,i'\}} \subseteq G_{\{i,i'\}}$.
4. $T_{\{i,i'\}}(t') \subseteq T_{\{i,i'\}}(t)$.
5. $T_{\{i,i'\}}(0) = W_{\{i,i'\}}$.

Proof: Follow directly from the definitions of the sets $W_{\{i,i'\}}$, $B_{\{i,i'\}}$, $V_{\{i,i'\}}$, and $T_{\{i,i'\}}(\tau)$, where $\tau \in \mathbb{R}^{\geq 0}$, and Lemma 4.4.2. ■

In the next two lemmas, we show that any state p in the set $B_{\{i,i'\}}$ is in the set *very-safe*$_{\{i,i'\}}$; that is, any state $R_{\{i,i'\}}$-reachable from p through an execution fragment that involves no input actions on port j, is in the set $G_{\{i,i'\}}$. In the first lemma, we show that any state that is $R_{\{i,i'\}}$-reachable from p through an execution fragment that involves no input actions on port j, is in the set $W_{\{i,i'\}}$. In the second lemma, we show that $B_{\{i,i'\}} \subseteq$ *very-safe*$_{\{i,i'\}}$.

Lemma 8.4.2 *future*$_{\{i,i'\}}(B_{\{i,i'\}}, \mathbb{R}^{\geq 0}) \subseteq W_{\{i,i'\}}$.

Proof: Let α be an execution fragment of the GRAPH-VEHICLES automaton of n steps and trajectories, where $n \in \mathbb{N}$, that: starts in a state in $B_{\{i,i'\}}$, is only comprised of states in $R_{\{i,i'\}}$, and involves no input actions on port j. Let p_{init} and p_{final} be the initial and final states of α, respectively. By induction on the length n of the execution fragment α, we show that $p_{final} \in W_{\{i,i'\}}$.

For the base case, consider the execution fragment α of length $n = 0$; that is, α is an execution fragment that consists of a single point trajectory and, therefore, $p_{final} = p_{init}$. Since $p_{init} \in B_{\{i,i'\}}$, Lemma 8.4.1, part 3, implies that $p_{final} \in W_{\{i,i'\}}$.

The inductive step involves showing that if α is an execution fragment of length $n = k+1$, for some $k \in \mathbb{N}$, then $p_{final} \in W_{\{i,i'\}}$. Let α' be the part of the execution fragment α comprised of the first k steps and trajectories. The induction hypothesis involves the assertion that if p'_{final} is the final state of α', then it is the case that $p'_{final} \in W_{\{i,i'\}}$. Since the final state of α is reached from the final state of α' by a single step, the inductive step involves the consideration of all possible steps and trajectories leading from p'_{final} to p_{final}.

In the case of a step, we consider all possible discrete actions by cases:

1. the actions $\texttt{protect}(C)_j$, for $C \in \mathcal{P}(\{i, i'\})$, are not enabled because α involves no input actions on port j.

2. the $\texttt{brick-wall}(i)$ action sets the velocity of the vehicle i to zero and affects neither the velocity of the vehicle i', nor the *collided*(i, i') and *collided*(i', i) variables.

   From the induction hypothesis, it is the case that $p'_{final} \in W_{\{i,i'\}} \subseteq G_{\{i,i'\}}$. Therefore, since the $\texttt{brick-wall}(i)$ action does not affect the *collided*(i, i') and *collided*(i', i) variables, it follows that $p_{final} \in G_{\{i,i'\}}$.

   Since the vehicle velocities are restricted to be non-negative, it is the case that $p_{final}.\dot{x}_i \leq p'_{final}.\dot{x}_i$. Moreover, since the $\texttt{brick-wall}(i)$ action does not affect the velocity of the vehicle i', it is the case that $p_{final}.\dot{x}_{i'} = p'_{final}.\dot{x}_{i'}$. From Lemma 4.4.3, part 1, it follows that $p_{final}.O_i \subseteq p'_{final}.O_i$ and $p_{final}.O_{i'} \subseteq p'_{final}.O_{i'}$. Therefore, since $p'_{final} \in W_{\{i,i'\}} \subseteq$ *disjoint-owned-tracks*(i, i'), it follows that $p_{final} \in$ *disjoint-owned-tracks*(i, i').

   Finally, since all states in α are, by definition, restricted to the set $R_{\{i,i'\}}$, it follows that $p_{final} \in W_{\{i,i'\}}$.

3. the $\texttt{brick-wall}(i')$ action sets the velocity of the vehicle i' to zero and affects neither the velocity of the vehicle i, nor the *collided*(i, i') and *collided*(i', i) variables.

   From the induction hypothesis, it is the case that $p'_{final} \in W_{\{i,i'\}} \subseteq G_{\{i,i'\}}$. Therefore, since the $\texttt{brick-wall}(i')$ action does not affect the *collided*(i, i') and *collided*(i', i) variables, it follows that $p_{final} \in G_{\{i,i'\}}$.

   Since the vehicle velocities are restricted to be non-negative, it is the case that $p_{final}.\dot{x}_{i'} \leq p'_{final}.\dot{x}_{i'}$. Moreover, since the $\texttt{brick-wall}(i')$ action does not affect the velocity of the vehicle i, it is the case that $p_{final}.\dot{x}_i = p'_{final}.\dot{x}_i$. From Lemma 4.4.3, part 1, it follows that $p_{final}.O_{i'} \subseteq p'_{final}.O_{i'}$ and $p_{final}.O_i \subseteq p'_{final}.O_i$. Therefore, since $p'_{final} \in W_{\{i,i'\}} \subseteq$ *disjoint-owned-tracks*(i, i'), it follows that $p_{final} \in$ *disjoint-owned-tracks*(i, i').

   Finally, since all states in α are, by definition, restricted to the set $R_{\{i,i'\}}$, it follows that $p_{final} \in W_{\{i,i'\}}$.

4. the actions $\texttt{protect}(C)_{j'}$, for $C \in \mathcal{P}(I)$ and $j' \in J, j' \neq j$, $\texttt{brick-wall}(i'')$, for $i'' \in I - \{i, i'\}$, and $\texttt{reset-location}(i''')$, for $i''' \in I$, affect neither the velocities of the vehicles i and i', nor the *collided*(i, i') and *collided*(i', i) variables.

From the induction hypothesis, it is the case that $p'_{final} \in W_{\{i,i'\}} \subseteq G_{\{i,i'\}}$. Therefore, since the actions $\texttt{protect}(C)_{j'}$, for $C \in \mathcal{P}(I)$ and $j' \in J, j' \neq j$, $\texttt{brick-wall}(i'')$, for $i'' \in I - \{i, i'\}$, and $\texttt{reset-location}(i''')$, for $i''' \in I$, do not affect the *collided*(i, i') and *collided*(i', i) variables, it follows that $p_{final} \in G_{\{i,i'\}}$.

Moreover, since the input actions $\texttt{protect}(C)_{j'}$, for $C \in \mathcal{P}(I)$ and $j' \in J, j' \neq j$, and the internal actions $\texttt{brick-wall}(i'')$, for $i'' \in I - \{i, i'\}$, and $\texttt{reset-location}(i''')$, for $i''' \in I$, do not affect the velocities of the vehicles i and i', it is the case that $p_{final}.\dot{x}_i = p'_{final}.\dot{x}_i$ and $p_{final}.\dot{x}_{i'} = p'_{final}.\dot{x}_{i'}$. From Lemma 4.4.3, part 1, it follows that $p_{final}.O_i \subseteq p'_{final}.O_i$ and $p_{final}.O_{i'} \subseteq p'_{final}.O_{i'}$. Therefore, since $p'_{final} \in W_{\{i,i'\}} \subseteq$ *disjoint-owned-tracks*(i, i'), it is the case that $p_{final} \in$ *disjoint-owned-tracks*(i, i').

Finally, since all states in α are, by definition, restricted to the set $R_{\{i,i'\}}$, it follows that $p_{final} \in W_{\{i,i'\}}$.

5. the internal actions $\texttt{colliding-pair}(i'', i''')$, for $i'', i''' \in I, i'' \neq i'''$, and the internal actions $\texttt{collision-effects}(i'''')$, for $i'''' \in I$, are not enabled because α is only comprised of states in $R_{\{i,i'\}}$ and $p'_{final} \in W_{\{i,i'\}}$.

Since, $p_{init} \in B_{(i,i')} \subseteq P_{B_{ij}} \cap P_{B_{i'j}}$ and the execution fragment leading from p_{init} to p'_{final} involves no input actions on port j, it follows that $p'_{final} \in P_{B_{ij}} \cap P_{B_{i'j}}$. Therefore, in the case of a trajectory from p'_{final} to p_{final}, Lemma 4.4.4, part 1, implies that $p_{final}.O_i \subseteq p'_{final}.O_i$ and $p_{final}.O_{i'} \subseteq p'_{final}.O_{i'}$. Therefore, since $p'_{final} \in W_{\{i,i'\}} \subseteq$ *disjoint-owned-tracks*(i, i'), it follows that $p_{final} \in$ *disjoint-owned-tracks*(i, i'). Moreover, since $p'_{final} \in W_{\{i,i'\}} \subseteq G_{\{i,i'\}}$ and the variables *collided*(i, i') and *collided*(i', i) remain constant throughout the trajectory, it follows that $p_{final} \in G_{\{i,i'\}}$. Finally, since all states in α are, by definition, restricted to the set $R_{\{i,i'\}}$, it follows that $p_{final} \in W_{\{i,i'\}}$, as needed. ■

Lemma 8.4.3 $B_{\{i,i'\}} \subseteq$ *very-safe*$_{\{i,i'\}}$.

Proof: Follows directly from Lemma 8.4.2, Lemma 8.4.1, part 3, and the definition of *very-safe* in Section 3.2.1. ■

In the next three lemmas and the subsequent corollary, we show that the sets $W_{\{i,i'\}}$ and *safe*$_{\{i,i'\}}$ are equal. First, we show that any state that is $R_{\{i,i'\}}$-reachable from a state p in $W_{\{i,i'\}}$ through an execution fragment that involves no input actions on port j and has a limit time equal to zero, is in the set $W_{\{i,i'\}}$. Then, we show that $W_{\{i,i'\}} \subseteq$ *safe*$_{\{i,i'\}}$ and *safe*$_{\{i,i'\}} \subseteq W_{\{i,i'\}}$. Finally, the subsequent corollary states that $W_{\{i,i'\}} =$ *safe*$_{\{i,i'\}}$.

Lemma 8.4.4 *future*$_{\{i,i'\}}(W_{\{i,i'\}}, 0) \subseteq W_{\{i,i'\}}$.

Proof: Let α be an execution fragment of the GRAPH-VEHICLES automaton of n steps, where $n \in \mathbb{N}$, that: starts in a state in $W_{\{i,i'\}}$, is only comprised of states in $R_{\{i,i'\}}$, involves

no input actions on port j, and has a limit time equal to zero. Let p_{init} and p_{final} be the initial and final states of α, respectively. By induction on the length n of the execution fragment α, we show that $p_{final} \in W_{\{i,i'\}}$.

For the base case, consider the execution fragment α of length $n = 0$; that is, α is an execution fragment that consists of no steps and, therefore, $p_{final} = p_{init}$. Since $p_{init} \in W_{\{i,i'\}}$, it follows that $p_{final} \in W_{\{i,i'\}}$.

The inductive step involves showing that if α is an execution fragment of length $n = k+1$, for some $k \in \mathbb{N}$, then $p_{final} \in W_{\{i,i'\}}$. Let α' be the part of the execution fragment α comprised of the first k steps. The induction hypothesis involves the assertion that if p'_{final} is the final state of α', then it is the case that $p'_{final} \in W_{\{i,i'\}}$. Since the final state of α is reached from the final state of α' by a single step, the inductive step involves the consideration of all possible steps leading from p'_{final} to p_{final}.

To complete the induction, we consider all possible discrete actions by cases:

1. the actions $\texttt{protect}(C)_j$, for $C \in \mathcal{P}(\{i, i'\})$, are not enabled because α involves no input actions on port j.

2. the $\texttt{brick-wall}(i)$ action sets the velocity of the vehicle i to zero and affects neither the velocity of the vehicle i', nor the $\mathit{collided}(i, i')$ and $\mathit{collided}(i', i)$ variables.

   From the induction hypothesis, it is the case that $p'_{final} \in W_{\{i,i'\}} \subseteq G_{\{i,i'\}}$. Therefore, since the $\texttt{brick-wall}(i)$ action does not affect the $\mathit{collided}(i, i')$ and $\mathit{collided}(i', i)$ variables, it follows that $p_{final} \in G_{\{i,i'\}}$.

   Since the vehicle velocities are restricted to be non-negative, it is the case that $p_{final}.\dot{x}_i \leq p'_{final}.\dot{x}_i$. Moreover, since the $\texttt{brick-wall}(i)$ action does not affect the velocity of the vehicle i', it is the case that $p_{final}.\dot{x}_{i'} = p'_{final}.\dot{x}_{i'}$. From Lemma 4.4.3, part 1, it follows that $p_{final}.O_i \subseteq p'_{final}.O_i$ and $p_{final}.O_{i'} \subseteq p'_{final}.O_{i'}$. Therefore, since $p'_{final} \in W_{\{i,i'\}} \subseteq \mathit{disjoint\text{-}owned\text{-}tracks}(i, i')$, it follows that $p_{final} \in \mathit{disjoint\text{-}owned\text{-}tracks}(i, i')$.

   Finally, since all states in α are, by definition, restricted to the set $R_{\{i,i'\}}$, it follows that $p_{final} \in W_{\{i,i'\}}$.

3. the $\texttt{brick-wall}(i')$ action sets the velocity of the vehicle i' to zero and affects neither the velocity of the vehicle i, nor the $\mathit{collided}(i, i')$ and $\mathit{collided}(i', i)$ variables.

   From the induction hypothesis, it is the case that $p'_{final} \in W_{\{i,i'\}} \subseteq G_{\{i,i'\}}$. Therefore, since the $\texttt{brick-wall}(i')$ action does not affect the $\mathit{collided}(i, i')$ and $\mathit{collided}(i', i)$ variables, it follows that $p_{final} \in G_{\{i,i'\}}$.

   Since the vehicle velocities are restricted to be non-negative, it is the case that $p_{final}.\dot{x}_{i'} \leq p'_{final}.\dot{x}_{i'}$. Moreover, since the $\texttt{brick-wall}(i')$ action does not affect the velocity of the vehicle i, it is the case that $p_{final}.\dot{x}_i = p'_{final}.\dot{x}_i$. From Lemma 4.4.3, part 1,

it follows that $p_{final}.O_{i'} \subseteq p'_{final}.O_{i'}$ and $p_{final}.O_i \subseteq p'_{final}.O_i$. Therefore, since $p'_{final} \in W_{\{i,i'\}} \subseteq$ *disjoint-owned-tracks*(i,i'), it follows that $p_{final} \in$ *disjoint-owned-tracks*(i,i').

Finally, since all states in α are, by definition, restricted to the set $R_{\{i,i'\}}$, it follows that $p_{final} \in W_{\{i,i'\}}$.

4. the actions $\texttt{protect}(C)_{j'}$, for $C \in \mathcal{P}(I)$ and $j' \in J, j' \neq j$, $\texttt{brick-wall}(i'')$, for $i'' \in I - \{i,i'\}$, and $\texttt{reset-location}(i''')$, for $i''' \in I$, affect neither the velocities of the vehicles i and i', nor the *collided*(i,i') and *collided*(i',i) variables.

   From the induction hypothesis, it is the case that $p'_{final} \in W_{\{i,i'\}} \subseteq G_{\{i,i'\}}$. Therefore, since the actions $\texttt{protect}(C)_{j'}$, for $C \in \mathcal{P}(I)$ and $j' \in J, j' \neq j$, $\texttt{brick-wall}(i'')$, for $i'' \in I - \{i,i'\}$, and $\texttt{reset-location}(i''')$, for $i''' \in I$, do not affect the *collided*(i,i') and *collided*(i',i) variables, it follows that $p_{final} \in G_{\{i,i'\}}$.

   Moreover, since the input actions $\texttt{protect}(C)_{j'}$, for $C \in \mathcal{P}(I)$ and $j' \in J, j' \neq j$, and the internal actions $\texttt{brick-wall}(i'')$, for $i'' \in I - \{i,i'\}$, and $\texttt{reset-location}(i''')$, for $i''' \in I$, do not affect the velocities of the vehicles i and i', it is the case that $p_{final}.\dot{x}_i = p'_{final}.\dot{x}_i$ and $p_{final}.\dot{x}_{i'} = p'_{final}.\dot{x}_{i'}$. From Lemma 4.4.3, part 1, it follows that $p_{final}.O_i \subseteq p'_{final}.O_i$ and $p_{final}.O_{i'} \subseteq p'_{final}.O_{i'}$. Therefore, since $p'_{final} \in W_{\{i,i'\}} \subseteq$ *disjoint-owned-tracks*(i,i'), it is the case that $p_{final} \in$ *disjoint-owned-tracks*(i,i').

   Finally, since all states in α are, by definition, restricted to the set $R_{\{i,i'\}}$, it follows that $p_{final} \in W_{\{i,i'\}}$.

5. the internal actions $\texttt{colliding-pair}(i'',i''')$, for $i'',i''' \in I, i'' \neq i'''$, and the internal actions $\texttt{collision-effects}(i'''')$, for $i'''' \in I$, are not enabled because α is only comprised of states in $R_{\{i,i'\}}$ and $p'_{final} \in W_{\{i,i'\}}$.

■

Lemma 8.4.5 $W_{\{i,i'\}} \subseteq$ *safe*$_{\{i,i'\}}$.

Proof: From the definition of *safe* in Section 3.2.1, we must show that any state $p \in W_{\{i,i'\}}$ satisfies: (i) *future*$_{\{i,i'\}}(p,0) \subseteq G_{\{i,i'\}}$, and (ii) there exists some input action π on port j such that for every $p', p'' \in R_{\{i,i'\}}$ satisfying $p' \in$ *future*$_{\{i,i'\}}(p,0)$ and $p' \xrightarrow{\pi} p''$, it is the case that $p'' \in$ *very-safe*$_{\{i,i'\}}$.

(i) Since $p \in W_{\{i,i'\}}$, the first condition follows from Lemma 8.4.4 and Lemma 8.4.1, part 1.

(ii) For the second condition, let π be the action $\texttt{protect}(\{i,i'\})_j$.

From Lemma 8.4.4, it follows that $p' \in W_{\{i,i'\}}$. Now, considering the step from p' to p'', since the $\texttt{protect}(\{i,i'\})_j$ action affects neither the velocity of any of the vehicles, nor any of the *collided* variables, it follows that $p'' \in R_{\{i,i'\}}$, $p'' \in G_{\{i,i'\}}$, $p''.\dot{x}_i = p'.\dot{x}_i$, and $p''.\dot{x}_{i'} = p'.\dot{x}_{i'}$. Therefore, Lemma 4.4.3, part 1, implies that $p''.O_i \subseteq p'.O_i$ and $p''.O_{i'} \subseteq p'.O_{i'}$. Since

$p' \in W_{\{i,i'\}}$, it follows that $p'' \in$ *disjoint-owned-tracks*(i, i'). From the above conditions, it follows that $p'' \in W_{\{i,i'\}}$.

Moreover, since the $\texttt{protect}(\{i, i'\})_j$ action sets the internal variables *brake-req*(i, j) and *brake-req*(i', j) to `True`, it is the case that $p'' \in P_{B_{ij}} \cap P_{B_{i'j}}$. Since $p'' \in W_{\{i,i'\}}$, it follows that $p'' \in B_{\{i,i'\}}$.

Finally, Lemma 8.4.3 implies that $p'' \in$ *very-safe*$_{\{i,i'\}}$, as needed. ■

Lemma 8.4.6 *For any $p \in R_{\{i,i'\}}$, if $p \in$ safe$_{\{i,i'\}}$ then $p \in W_{\{i,i'\}}$.*

Proof: We show the contrapositive; that is, for any $p \in R_{\{i,i'\}}$, if $p \notin W_{\{i,i'\}}$ then $p \notin$ *safe*$_{\{i,i'\}}$. Since $W_{\{i,i'\}} = R_{\{i,i'\}} \cap G_{\{i,i'\}} \cap$ *disjoint-owned-tracks*(i, i') and $p \in R_{\{i,i'\}}$, we consider the conditions $p \notin G_{\{i,i'\}}$ and $p \notin$ *disjoint-owned-tracks*(i, i') separately.

1. $p \notin G_{\{i,i'\}}$.

   From Lemma 3.2.4, part 1, it is the case that *safe*$_{\{i,i'\}} \subseteq G_{\{i,i'\}}$. Since $p \notin G_{\{i,i'\}}$, it follows that $p \notin$ *safe*$_{\{i,i'\}}$.

2. $p \notin$ *disjoint-owned-tracks*(i, i').

   We must show that $p \notin$ *safe*$_{\{i,i'\}}$. In order for the state $p \in R_{\{i,i'\}}$ to be in the set *safe*$_{\{i,i'\}}$ there must exist some input action π on port j such that for every $p', p'' \in R_{\{i,i'\}}$ satisfying $p' \in$ *future*$_{\{i,i'\}}(p, 0)$ and $p' \xrightarrow{\pi} p''$, it is the case that $p'' \in$ *very-safe*$_{\{i,i'\}}$. Therefore, it suffices to show that for any input action π on port j, there exist $p', p'' \in R_{\{i,i'\}}$ satisfying $p' \in$ *future*$_{\{i,i'\}}(p, 0)$ and $p' \xrightarrow{\pi} p''$, such that $p'' \notin$ *very-safe*$_{\{i,i'\}}$.

   Using similar analyses to those presented in the proofs of Lemmas 6.2.9 and 7.4.10, it can be shown that for any $p \in R_{\{i,i'\}}$ and any input action π on port j, there exist $p', p'' \in R_{\{i,i'\}}$ satisfying $p' \in$ *future*$_{\{i,i'\}}(p, 0)$ and $p' \xrightarrow{\pi} p''$, such that $p'' \notin$ *very-safe*$_{\{i,i'\}}$. It follows that $p \notin$ *safe*$_{\{i,i'\}}$, as needed.

■

Corollary 8.4.7 $W_{\{i,i'\}} =$ *safe*$_{\{i,i'\}}$.

Proof: Follows directly from Lemmas 8.4.5 and 8.4.6. ■

In the following three lemmas, we show that any state $R_{\{i,i'\}}$-reachable from a state in $V_{(i,i')}$ through an execution fragment that involves no input actions on port j and has a limit time that lies in the interval $[0, d_{max}]$, is in the set $W_{\{i,i'\}}$. In the first lemma, we show that if the final state of such an execution fragment is in $G_{\{i,i'\}}$ and the section of track owned by the

vehicle i has not grown since the beginning of the execution fragment, then the final state of the execution fragment is in $W_{\{i,i'\}}$. In the second lemma, we show that the final state of any such execution fragment is in $G_{\{i,i'\}}$ and the section of track owned by the vehicle i does not grow throughout the execution fragment. The third lemma states the desired property which follows directly from the first two lemmas.

Lemma 8.4.8 *Let $p \in V_{(i,i')}$ and $p' \in \mathit{future}_{\{i,i'\}}(p, [0, d_{max}])$. If $p' \in G_{\{i,i'\}}$ and $p'.O_i \subseteq p.O_i$, then $p' \in W_{\{i,i'\}}$.*

Proof: We need to show that $p' \in W_{\{i,i'\}}$; that is, we need to show that the state p' is in the set $R_{\{i,i'\}} \cap G_{\{i,i'\}} \cap \mathit{disjoint\text{-}owned\text{-}tracks}(i, i')$. Since $p' \in G_{\{i,i'\}}$, by assumption, it remains to be shown that $p' \in R_{\{i,i'\}}$ and $p' \in \mathit{disjoint\text{-}owned\text{-}tracks}(i, i')$. We consider these two conditions by cases:

1. $p' \in R_{\{i,i'\}}$.

   This is the case because the function $\mathit{future}_{\{i,i'\}}(p, \mathbb{R}^{\geq 0})$ only considers $R_{\{i,i'\}}$-reachable states.

2. $p' \in \mathit{disjoint\text{-}owned\text{-}tracks}(i, i')$.

   Since $p \in V_{(i,i')}$, there are two possible scenarios: (i) $p \in \mathit{successive}(i, i')$ and $p.l_i < p.l_{i'}$, (ii) $p \in \mathit{adjacent}(i, i')$ and $p.O_i \subseteq [p.l_i, \langle p.l_i.e, \mathit{length}(p.l_i.e)\rangle]$.

   In the first case, it is as if the vehicle i is trailing the vehicle i' on a single track. Since $p \in V_{(i,i')} \subseteq W_{\{i,i'\}}$, the sections of the track owned by the vehicles i and i' in state p are disjoint. Now, consider the section of track owned by the vehicle i in the state p'. Since $p'.O_i \subseteq p.O_i$, it follows that $p.l_i = \min(p.O_i) \leq p'.l_i = \min(p'.O_i)$ and there exist locations in $p.O_i$ that are at least as downstream as any of the locations in $p'.O_i$. Next, consider the section of track owned by the vehicle i' in the state p'. Because of the non-negative constraint on the vehicle velocities it follows that the location $p'.l_{i'} = \min(p'.O_{i'})$ is either equal to, or downstream of the location $p.l_{i'} = \min(p.O_{i'})$. Moreover, the sections of track owned by the vehicle i' in state p' could only range from the location $p'.l_{i'}$ up to the locations that are a distance Δx_{max} downstream from the location $p.l_{i'}$. Therefore, because of the constraint on the length of the edges in the track topology and the constraint on the minimum number of edges comprising a cycle in the track topology, it follows that $p' \in \mathit{disjoint\text{-}owned\text{-}tracks}(i, i')$.

   In the second case, since $p.O_i \subseteq [p.l_i, \langle p.l_i.e, \mathit{length}(p.l_i.e)\rangle]$, the section of the track owned by the vehicle i in state p is strictly within the incoming directed edge $p.l_i.e$. Since $p'.O_i \subseteq p.O_i$, the same is true for the section of track owned by the vehicle i in state p'. Similarly to above, because of the non-negative constraint on the vehicle velocities it follows that the location $p'.l_{i'} = \min(p'.O_{i'})$ is either equal to, or downstream of the location $p.l_{i'} = \min(p.O_{i'})$. Moreover, the sections of track owned by

the vehicle i' in state p' could only range from the location $p'.l_{i'}$ up to the locations that are a distance Δx_{max} downstream from the location $p.l_{i'}$. Therefore, because of the constraint on the length of the edges in the track topology, the constraint on the minimum number of edges comprising a cycle in the track topology, the fact that the vehicles are traveling on adjacent tracks in state p, and the fact that the section of track owned by the vehicle i remains within the incoming branch, it follows that $p' \in$ *disjoint-owned-tracks*(i, i').

■

Lemma 8.4.9 *If $p \in V_{(i,i')}$ and $p' \in future_{\{i,i'\}}(p, [0, d_{max}])$, then it is the case that $p' \in G_{\{i,i'\}}$ and $p'.O_i \subseteq p.O_i$.*

Proof: Let α be an execution fragment of the GRAPH-VEHICLES automaton of n steps and trajectories, where $n \in \mathbb{N}$, that: starts in a state in $V_{(i,i')}$, is only comprised of states in $R_{\{i,i'\}}$, involves no input actions on port j, and has a limit time that lies in the interval $[0, d_{max}]$. Letting p_{init} and p_{final} be the initial and final states of α, respectively, we must show that $p_{final} \in G_{\{i,i'\}}$ and $p_{final}.O_i \subseteq p_{init}.O_i$. The proof is by induction on the length n of the execution fragment α.

For the base case, consider the execution fragment α of length $n = 0$; that is, α is an execution fragment that consists of a single point trajectory and, therefore, $p_{final} = p_{init}$. From Lemma 8.4.1, part 2, and the fact that $p_{init} \in V_{(i,i')} \subseteq V_{\{i,i'\}}$, it follows that $p_{final} \in G_{\{i,i'\}}$. Moreover, the fact that $p_{final}.O_i \subseteq p_{init}.O_i$ is trivially true.

The inductive step involves showing that if α is an execution fragment of length $n = k+1$, for some $k \in \mathbb{N}$, then $p_{final} \in G_{\{i,i'\}}$ and $p_{final}.O_i \subseteq p_{init}.O_i$. Let α' be the part of the execution fragment α comprised of the first k steps and trajectories. The induction hypothesis involves the assertion that if p'_{init} and p'_{final} are the initial and final states of α', respectively, then it is the case that $p'_{final} \in G_{\{i,i'\}}$ and $p'_{final}.O_i \subseteq p'_{init}.O_i$. Moreover, from Lemma 8.4.8 it follows that $p'_{final} \in W_{\{i,i'\}}$. Since the final state of α is reached from the final state of α' by a single step or trajectory, the inductive step involves the consideration of all possible steps and trajectories leading from p'_{final} to p_{final}.

In the case of a step, we consider all possible actions by cases:

1. the actions `protect`$(C)_j$, for $C \in \mathcal{P}(\{i, i'\})$, are not enabled because α involves no input actions on port j.
2. the `brick-wall`(i) action sets the velocity of the vehicle i to zero and does not affect the *collided*(i, i') and *collided*(i', i) variables.

   From the induction hypothesis, it is the case that $p'_{final} \in G_{\{i,i'\}}$. Therefore, since the

`brick-wall`(i) action does not affect the *collided*(i, i') and *collided*(i', i) variables, it follows that $p_{final} \in G_{\{i,i'\}}$.

Moreover, since the vehicle velocities are restricted to be non-negative, it is the case that $p_{final}.\dot{x}_i \leq p'_{final}.\dot{x}_i$. From Lemma 4.4.3, part 1, it follows that $p_{final}.O_i \subseteq p'_{final}.O_i$. However, from the induction hypothesis it is the case that $p'_{final}.O_i \subseteq p'_{init}.O_i$. Therefore, since $p_{init} = p'_{init}$, it follows that $p_{final}.O_i \subseteq p_{init}.O_i$, as needed.

3. the actions `protect`$(C)_{j'}$, for $C \in \mathcal{P}(I)$ and $j' \in J, j' \neq j$, `brick-wall`(i''), for $i'' \in I, i'' \neq i$, and `reset-location`(i'''), for $i''' \in I$, affect neither the velocity of the vehicle i, nor the *collided*(i, i') and *collided*(i', i) variables.

   From the induction hypothesis, it is the case that $p'_{final} \in G_{\{i,i'\}}$. Therefore, since the actions `protect`$(C)_{j'}$, for $C \in \mathcal{P}(I)$ and $j' \in J, j' \neq j$, `brick-wall`(i''), for $i'' \in I, i'' \neq i$, and `reset-location`(i'''), for $i''' \in I$, do not affect the *collided*(i, i') and *collided*(i', i) variables, it follows that $p_{final} \in G_{\{i,i'\}}$.

   Moreover, since the input actions `protect`$(C)_{j'}$, for $C \in \mathcal{P}(I)$ and $j' \in J, j' \neq j$, and the internal actions `brick-wall`(i''), for $i'' \in I, i'' \neq i$ and `reset-location`(i'''), for $i''' \in I$, do not affect the velocity of the vehicle i, it is the case that $p_{final}.\dot{x}_i = p'_{final}.\dot{x}_i$. From Lemma 4.4.3, part 1, it follows that $p_{final}.O_i \subseteq p'_{final}.O_i$. However, from the induction hypothesis it is the case that $p'_{final}.O_i \subseteq p'_{init}.O_i$. Therefore, since $p_{init} = p'_{init}$, it follows that $p_{final}.O_i \subseteq p_{init}.O_i$, as needed.

4. the internal actions `colliding-pair`(i'', i'''), for $i'', i''' \in I, i'' \neq i'''$, and the internal actions `collision-effects`(i''''), for $i'''' \in I$, are not enabled because α is only comprised of states in $R_{\{i,i'\}}$ and $p'_{final} \in W_{\{i,i'\}}$.

Since $p'_{init} \in V_{(i,i')} \subseteq P_{B_{ij}}$ and the execution fragment leading from p'_{init} to p'_{final} involves no input actions on port j, it follows that $p'_{final} \in P_{B_{ij}}$. Therefore, in the case of a trajectory from p'_{final} to p_{final}, Lemma 4.4.4, part 1, implies that $p_{final}.O_i \subseteq p'_{final}.O_i$. However, from the induction hypothesis it is the case that $p'_{final}.O_i \subseteq p'_{init}.O_i$. Since $p_{init} = p'_{init}$, it follows that $p_{final}.O_i \subseteq p_{init}.O_i$. Moreover, since $p'_{final} \in G_{\{i,i'\}}$ and the variables *collided*(i, i') and *collided*(i', i) remain constant throughout the trajectory, it follows that $p_{final} \in G_{\{i,i'\}}$, as needed. ■

Lemma 8.4.10 $future_{\{i,i'\}}(V_{(i,i')}, [0, d_{max}]) \subseteq W_{\{i,i'\}}$.

Proof: Follows directly from Lemmas 8.4.8 and 8.4.9. ■

In the following lemma, we extend the result of Lemma 8.4.10 to the set $V_{\{i,i'\}}$.

Lemma 8.4.11 $future_{\{i,i'\}}(V_{\{i,i'\}}, [0, d_{max}]) \subseteq W_{\{i,i'\}}$.

Proof: Follows directly from Lemma 8.4.10 and the fact that $V_{\{i,i'\}} = V_{(i,i')} \cup V_{(i',i)}$. ■

The following lemma states that any state p in the set $V_{\{i,i'\}}$ is in the set *delay-safe*$_{\{i,i'\}}(d)$; that is, any state $R_{\{i,i'\}}$-reachable from p within an amount of time d through an execution fragment that involves no input actions on port j, is in the set $G_{\{i,i'\}}$ and any state $R_{\{i,i'\}}$-reachable from p in exactly an amount of time d through an execution fragment that involves no input actions on port j, is in the set *safe*$_{\{i,i'\}}$.

Lemma 8.4.12 $V_{\{i,i'\}} \subseteq$ *delay-safe*$_{\{i,i'\}}(d)$.

Proof: Follows from Lemma 3.2.4, part 1, Lemma 8.4.11, Corollary 8.4.7, and the fact that $d \leq d_{max}$. ■

In the next few lemmas, we show that any state p in the set $T_{\{i,i'\}}(t)$, for any $t \in \mathbb{R}^{\geq 0}$, is in the set *delay-safe*$_{\{i,i'\}}(t)$; that is, any state $R_{\{i,i'\}}$-reachable from p within an amount of time t through an execution fragment that involves no input actions on port j, is in the set $G_{\{i,i'\}}$ and any state $R_{\{i,i'\}}$-reachable from p in exactly an amount of time t through an execution fragment that involves no input actions on port j, is in the set *safe*$_{\{i,i'\}}$.

Lemma 8.4.13 *Let $p \in T_{\{i,i'\}}(\tau)$, where $\tau \in \mathbb{R}^{\geq 0}$, and $p' \in$ future$_{\{i,i'\}}(p,t)$, where $t \in [0,\tau]$. If $p' \in G_{\{i,i'\}}$, $p'.C_i(\tau - t) \subseteq p.C_i(\tau)$, and $p'.C_{i'}(\tau - t) \subseteq p.C_{i'}(\tau)$, then $p' \in T_{\{i,i'\}}(\tau - t)$.*

Proof: We need to show that $p' \in R_{\{i,i'\}} \cap G_{\{i,i'\}} \cap$ *disjoint-claimed-tracks*$(i,i',\tau - t)$. Since $p' \in G_{\{i,i'\}}$, by assumption, it remains to be shown that $p' \in R_{\{i,i'\}}$ and $p' \in$ *disjoint-claimed-tracks*$(i,i',\tau - t)$. We consider these two conditions by cases:

1. $p' \in R_{\{i,i'\}}$.

   This is the case because the function *future*$_{\{i,i'\}}(p,t)$ only considers $R_{\{i,i'\}}$-reachable states.

2. $p' \in$ *disjoint-claimed-tracks*$(i,i',\tau - t)$.

   Since $p \in T_{\{i,i'\}}(\tau)$, it is the case that $p \in$ *disjoint-claimed-tracks*(i,i',τ). Therefore, since $p'.C_i(\tau - t) \subseteq p.C_i(\tau)$ and $p'.C_{i'}(\tau - t) \subseteq p.C_{i'}(\tau)$, it follows that $p' \in$ *disjoint-claimed-tracks*$(i,i',\tau - t)$, as needed.

■

Lemma 8.4.14 *For all $p \in T_{\{i,i'\}}(\tau)$, where $\tau \in \mathbb{R}^{\geq 0}$, and $p' \in$ future$_{\{i,i'\}}(p,t)$, where $t \in [0,\tau]$, it is the case that $p' \in G_{\{i,i'\}}$, $p'.C_i(\tau - t) \subseteq p.C_i(\tau)$, and $p'.C_{i'}(\tau - t) \subseteq p.C_{i'}(\tau)$.*

Proof: Let $\tau \in \mathbb{R}^{\geq 0}$ and α be an execution fragment of the GRAPH-VEHICLES automaton of n steps and trajectories, where $n \in \mathbb{N}$, that: starts in a state in $T_{\{i,i'\}}(\tau)$, is only comprised of states in $R_{\{i,i'\}}$, involves no input actions on port j, and has a limit time t that lies in the interval $[0, \tau]$. Letting p_{init} and p_{final} be the initial and final states of α, respectively, we must show that $p_{final} \in G_{\{i,i'\}}$, $p_{final}.C_i(\tau - t) \subseteq p_{init}.C_i(\tau)$, and $p_{final}.C_{i'}(\tau - t) \subseteq p_{init}.C_{i'}(\tau)$. The proof is by induction on the length n of the execution fragment α.

For the base case, consider the execution fragment α of length $n = 0$; that is, α is an execution fragment that consists of a single point trajectory and, therefore, $p_{final} = p_{init}$ and $\alpha.ltime = 0$, *i.e.*, $t = 0$. From Lemma 8.4.1, part 1, and the fact that $p_{init} \in T_{\{i,i'\}}(\tau)$, it follows that $p_{final} \in G_{\{i,i'\}}$. Moreover, since $t = 0$, the facts that $p_{final}.C_i(\tau - t) \subseteq p_{init}.C_i(\tau)$ and $p_{final}.C_{i'}(\tau - t) \subseteq p_{init}.C_{i'}(\tau)$ are trivially true.

The inductive step involves showing that if α is an execution fragment of length $n = k + 1$, for some $k \in \mathbb{N}$, with $\alpha.ltime = t$, where $t \in [0, \tau]$, then $p_{final} \in G_{\{i,i'\}}$, $p_{final}.C_i(\tau - t) \subseteq p_{init}.C_i(\tau)$, and $p_{final}.C_{i'}(\tau - t) \subseteq p_{init}.C_{i'}(\tau)$. Let α' be the part of the execution fragment α comprised of the first k steps and trajectories and let $\alpha'.ltime = t'$, where $t' \in [0, t]$. The induction hypothesis involves the assertion that if p'_{init} and p'_{final} are the initial and final states of α', respectively, then it is the case that $p'_{final} \in G_{\{i,i'\}}$, $p'_{final}.C_i(\tau - t') \subseteq p'_{init}.C_i(\tau)$, and $p'_{final}.C_{i'}(\tau - t') \subseteq p'_{init}.C_{i'}(\tau)$. Moreover, from Lemma 8.4.13 it follows that $p'_{final} \in T_{\{i,i'\}}(\tau - t')$. Since the final state of α is reached from the final state of α' by a single step or trajectory, the inductive step involves the consideration of all possible steps and trajectories leading from p'_{final} to p_{final}.

In the case of a step, keeping in mind that the limit times of α' and α are equal, *i.e.*, $t' = t$, we consider all possible discrete actions by cases:

1. the actions $\texttt{protect}(C)_j$, for $C \in \mathcal{P}(\{i, i'\})$, are not enabled because α involves no input actions on port j.
2. the $\texttt{brick-wall}(i)$ action sets the velocity of the vehicle i to zero and affects neither the velocity of the vehicle i', nor the *collided*(i, i') and *collided*(i', i) variables.

   From the induction hypothesis, it is the case that $p'_{final} \in G_{\{i,i'\}}$. Therefore, since the $\texttt{brick-wall}(i)$ action does not affect the *collided*(i, i') and *collided*(i', i) variables, it follows that $p_{final} \in G_{\{i,i'\}}$.

   Since the vehicle velocities are restricted to be non-negative, it is the case that $p_{final}.\dot{x}_i \leq p'_{final}.\dot{x}_i$. Moreover, since the $\texttt{brick-wall}(i)$ action does not affect the velocity of the vehicle i', it is the case that $p_{final}.\dot{x}_{i'} = p'_{final}.\dot{x}_{i'}$. From Lemma 4.4.3, part 2, it follows that $p_{final}.C_i(\tau - t) \subseteq p'_{final}.C_i(\tau - t')$ and $p_{final}.C_{i'}(\tau - t) \subseteq p'_{final}.C_{i'}(\tau - t')$. However, from the induction hypothesis it is the case that $p'_{final}.C_i(\tau - t') \subseteq p'_{init}.C_i(\tau)$ and $p'_{final}.C_{i'}(\tau - t') \subseteq p'_{init}.C_{i'}(\tau)$. Therefore, since $p_{init} = p'_{init}$, it follows that $p_{final}.C_i(\tau - t) \subseteq p_{init}.C_i(\tau)$ and $p_{final}.C_{i'}(\tau - t) \subseteq p_{init}.C_{i'}(\tau)$, as needed.

3. the $\texttt{brick-wall}(i')$ action sets the velocity of the vehicle i' to zero and affects neither the velocity of the vehicle i, nor the *collided*(i, i') and *collided*(i', i) variables.

   From the induction hypothesis, it is the case that $p'_{final} \in G_{\{i,i'\}}$. Therefore, since the $\texttt{brick-wall}(i')$ action does not affect the *collided*(i, i') and *collided*(i', i) variables, it follows that $p_{final} \in G_{\{i,i'\}}$.

   Since the vehicle velocities are restricted to be non-negative, it is the case that $p_{final}.\dot{x}_{i'} \leq p'_{final}.\dot{x}_{i'}$. Moreover, since the $\texttt{brick-wall}(i')$ action does not affect the velocity of the vehicle i, it is the case that $p_{final}.\dot{x}_i = p'_{final}.\dot{x}_i$. From Lemma 4.4.3, part 2, it follows that $p_{final}.C_i(\tau - t) \subseteq p'_{final}.C_i(\tau - t')$ and $p_{final}.C_{i'}(\tau - t) \subseteq p'_{final}.C_{i'}(\tau - t')$. However, from the induction hypothesis it is the case that $p'_{final}.C_i(\tau - t') \subseteq p'_{init}.C_i(\tau)$ and $p'_{final}.C_{i'}(\tau - t') \subseteq p'_{init}.C_{i'}(\tau)$. Therefore, since $p_{init} = p'_{init}$, it follows that $p_{final}.C_i(\tau - t) \subseteq p_{init}.C_i(\tau)$ and $p_{final}.C_{i'}(\tau - t) \subseteq p_{init}.C_{i'}(\tau)$, as needed.

4. the actions $\texttt{protect}(C)_{j'}$, for $C \in \mathcal{P}(I)$ and $j' \in J, j' \neq j$, $\texttt{brick-wall}(i'')$, for $i'' \in I - \{i, i'\}$, and $\texttt{reset-location}(i''')$, for $i''' \in I$, affect neither the velocities of the vehicles i and i', nor the *collided*(i, i') and *collided*(i', i) variables.

   From the induction hypothesis, it is the case that $p'_{final} \in G_{\{i,i'\}}$. Therefore, since the actions $\texttt{protect}(C)_{j'}$, for $C \in \mathcal{P}(I)$ and $j' \in J, j' \neq j$, $\texttt{brick-wall}(i'')$, for $i'' \in I - \{i, i'\}$, and $\texttt{reset-location}(i''')$, for $i''' \in I$, do not affect the *collided*(i, i') and *collided*(i', i) variables, it follows that $p_{final} \in G_{\{i,i'\}}$.

   Moreover, since the input actions $\texttt{protect}(C)_{j'}$, for $C \in \mathcal{P}(I)$ and $j' \in J, j' \neq j$, and the internal actions $\texttt{brick-wall}(i'')$, for $i'' \in I - \{i, i'\}$, and $\texttt{reset-location}(i''')$, for $i''' \in I$, do not affect the velocities of the vehicles i and i', it is the case that $p_{final}.\dot{x}_i = p'_{final}.\dot{x}_i$ and $p_{final}.\dot{x}_{i'} = p'_{final}.\dot{x}_{i'}$. From Lemma 4.4.3, part 2, it follows that $p_{final}.C_i(\tau - t) \subseteq p'_{final}.C_i(\tau - t')$ and $p_{final}.C_{i'}(\tau - t) \subseteq p'_{final}.C_{i'}(\tau - t')$. However, from the induction hypothesis it is the case that $p'_{final}.C_i(\tau - t') \subseteq p'_{init}.C_i(\tau)$ and $p'_{final}.C_{i'}(\tau - t') \subseteq p'_{init}.C_{i'}(\tau)$. Therefore, since $p_{init} = p'_{init}$, it follows that $p_{final}.C_i(\tau - t) \subseteq p_{init}.C_i(\tau)$ and $p_{final}.C_{i'}(\tau - t) \subseteq p_{init}.C_{i'}(\tau)$, as needed.

5. the internal actions $\texttt{colliding-pair}(i'', i''')$, for $i'', i''' \in I, i'' \neq i'''$, and the internal actions $\texttt{collision-effects}(i'''')$, for $i'''' \in I$, are not enabled because α is only comprised of states in $R_{\{i,i'\}}$ and $p'_{final} \in T_{\{i,i'\}}(\tau - t')$.

In the case of a trajectory, Lemma 4.4.4, part 2, implies that $p_{final}.C_i(\tau - t) \subseteq p'_{final}.C_i(\tau - t')$ and $p_{final}.C_{i'}(\tau - t) \subseteq p'_{final}.C_{i'}(\tau - t')$. However, from the induction hypothesis it is the case that $p'_{final}.C_i(\tau - t') \subseteq p'_{init}.C_i(\tau)$ and $p'_{final}.C_{i'}(\tau - t') \subseteq p'_{init}.C_{i'}(\tau)$. Therefore, since $p_{init} = p'_{init}$, it follows that $p_{final}.C_i(\tau - t) \subseteq p_{init}.C_i(\tau)$ and $p_{final}.C_{i'}(\tau - t) \subseteq p_{init}.C_{i'}(\tau)$. Moreover, since $p'_{final} \in G_{\{i,i'\}}$ and the *collided*(i, i') and *collided*(i', i) variables remain constant throughout the trajectory, it follows that $p_{final} \in G_{\{i,i'\}}$, as needed. ■

Lemma 8.4.15 *For $\tau \in \mathbb{R}^{\geq 0}$ and $t \in [0, \tau]$, it is the case that $future_{\{i,i'\}}(T_{\{i,i'\}}(\tau), t) \subseteq T_{\{i,i'\}}(\tau - t)$.*

Proof: Follows directly from Lemmas 8.4.13 and 8.4.14. ■

Corollary 8.4.16 *For any $t \in \mathbb{R}^{\geq 0}$, it is the case that $future_{\{i,i'\}}(T_{\{i,i'\}}(t), 0) \subseteq T_{\{i,i'\}}(t)$.*

Proof: Follows directly from Lemma 8.4.15. ■

Lemma 8.4.17 *For any $t \in \mathbb{R}^{\geq 0}$, it is the case that $T_{\{i,i'\}}(t) \subseteq$ delay-safe$_{\{i,i'\}}(t)$.*

Proof: From the definition of *delay-safe* in Section 3.2.1, we must show that:

1. $future_{\{i,i'\}}(T_{\{i,i'\}}(t), [0, t]) \subseteq G_{\{i,i'\}}$, and
2. $future_{\{i,i'\}}(T_{\{i,i'\}}(t), t) \subseteq safe_{\{i,i'\}}$.

The first condition follows directly from Lemma 8.4.15 and Lemma 8.4.1, part 1. Moreover, Lemma 8.4.15 and Lemma 8.4.1, part 5, imply that $future_{\{i,i'\}}(T_{\{i,i'\}}(t), t) \subseteq W_{\{i,i'\}}$. Therefore, the second condition follows from Lemma 8.4.5. ■

In the following lemma, we show that the protector GRAPH-PROT-PAIR$_{\{i,i'\}}$ implements the protector Abs(GRAPH-VEHICLES, $S_{\{i,i'\}}, R_{\{i,i'\}}, G_{\{i,i'\}}, j, d$). Since the protector automata GRAPH-PROT-PAIR$_{\{i,i'\}}$ and Abs_j involve the composition of the same sensor automaton with distinct controller automata, it suffices to show that the discrete controller automaton of the protector GRAPH-PROT-PAIR$_{\{i,i'\}}$ implements the discrete controller automaton DC(GRAPH-VEHICLES, $S_{\{i,i'\}}, R_{\{i,i'\}}, G_{\{i,i'\}}, j, d$).

Lemma 8.4.18 GRAPH-PROT-PAIR$_{\{i,i'\}} \leq Abs$(GRAPH-VEHICLES, $S_{\{i,i'\}}, R_{\{i,i'\}}, G_{\{i,i'\}}, j, d$).

Proof: Both the GRAPH-PROT-PAIR$_{\{i,i'\}}$ and the Abs_j protectors involve the composition of the same sensor automaton with distinct controller automata. From Theorem 2.7.4, it suffices to show that the discrete controller automaton of GRAPH-PROT-PAIR$_{\{i,i'\}}$ implements DC_j. This is shown by a simulation from the discrete controller automaton of GRAPH-PROT-PAIR$_{\{i,i'\}}$ to DC_j.

The mapping between the states of the discrete controller automaton of the protector GRAPH-PROT-PAIR$_{\{i,i'\}}$ and DC_j is almost the identity. In the discrete controller automaton of GRAPH-PROT-PAIR$_{\{i,i'\}}$, the variable $send_j$ is equal to either a member of $\mathcal{P}(\{i, i'\})$, or the value *null*. In DC_j, these valuations simply map to either the actions `protect`$(C)_j$, where

C is the member of $\mathcal{P}(\{i, i'\})$ that corresponds to the the valuation of the variable $send_j$ of the discrete controller automaton of GRAPH-PROT-PAIR$_{\{i,i'\}}$, or the value *null*, respectively.

The start states for the discrete controller automaton of GRAPH-PROT-PAIR$_{\{i,i'\}}$ and DC_j are the states in which $send_j = null$. These are related to each other according to the mapping discussed above.

Furthermore, since the trajectories in both discrete controller automata are identical, we need only consider their discrete transitions. We analyze the actions of the implementation by cases, letting p denote any complete state of the GRAPH-VEHICLES automaton that corresponds to the output state y, *i.e.*, $p \in VALID$ and $p\lceil Y_{\text{GRAPH-VEHICLES}} = y$.

1. The $\texttt{snapshot}(y)_j$ action of the implementation sets $send_j$ to an element of $\mathcal{P}(\{i, i'\})$. In order to show that the behavior of the implementation is allowed by the specification, we must show that the input action $\texttt{snapshot}(y)_j$ of the implementation sets the value of the $send_j$ variable in such a way that the subsequently enabled action π of the implementation (i) guarantees that for all $p', p'' \in R_{\{i,i'\}}$ such that $p' \in future_{\{i,i'\}}(p, 0)$ and $p' \xrightarrow{\pi} p''$, it is the case that $p'' \in delay\text{-}safe_{\{i,i'\}}(d)$, if $p \in safe_{\{i,i'\}}$, and (ii) is an arbitrary output action of the implementation, otherwise.

   First, consider the case in which $p \in safe_{\{i,i'\}}$. Since Corollary 8.4.7 implies that $p \in W_{\{i,i'\}}$, the discrete controller automaton of GRAPH-PROT-PAIR$_{\{i,i'\}}$ sets the variable $send_j$ according to whether the state p is in $T_{\{i,i'\}}(d)$, or not.

   On one hand, if $p \notin T_{\{i,i'\}}(d)$ then the discrete controller automaton of the protector GRAPH-PROT-PAIR$_{\{i,i'\}}$ sets the variable $send_j$ to either $\{i\}$, or $\{i'\}$ according to the strategy described in Section 8.3. Therefore, the $\texttt{snapshot}(y)_j$ action enables either the $\texttt{protect}(\{i\})_j$ action, or the $\texttt{protect}(\{i'\})_j$ action. Since $p \in W_{\{i,i'\}}$, Lemma 8.4.4 implies that $p' \in W_{\{i,i'\}}$. Moreover, since the $\texttt{protect}(\{i\})_j$ and $\texttt{protect}(\{i'\})_j$ actions affect neither the velocity of any of the vehicles, nor any of the *collided* variables, it follows that $p'' \in R_{\{i,i'\}}$, $p'' \in G_{\{i,i'\}}$, $p''.\dot{x}_i = p'.\dot{x}_i$, and $p''.\dot{x}_{i'} = p'.\dot{x}_{i'}$. Therefore, since $p' \in W_{\{i,i'\}}$, Lemma 4.4.3, part 1, implies that $p'' \in disjoint\text{-}owned\text{-}tracks(i, i')$. From the above conditions, it follows that $p'' \in W_{\{i,i'\}}$. Moreover, since the $\texttt{protect}(\{i\})_j$ and $\texttt{protect}(\{i'\})_j$ actions set the $brake\text{-}req(i, j)$ and $brake\text{-}req(i', j)$ variables, respectively, to `True`, it follows that $p'' \in V_{\{i,i'\}}$. Finally, Lemma 8.4.12 implies that $p'' \in delay\text{-}safe_{\{i,i'\}}(d)$, as needed.

   On the other hand, if $p \in T_{\{i,i'\}}(d)$ then the discrete controller automaton of the protector GRAPH-PROT-PAIR$_{\{i,i'\}}$ sets the variable $send_j$ to $\emptyset$ and the $\texttt{protect}(\emptyset)_j$ action is enabled. Since $p \in T_{\{i,i'\}}(d)$, Corollary 8.4.16 implies that $p' \in T_{\{i,i'\}}(d)$. Moreover, since the $\texttt{protect}(\emptyset)_j$ action affects neither the velocity of any of the vehicles, nor any of the *collided* variables, it follows that $p'' \in R_{\{i,i'\}}$, $p'' \in G_{\{i,i'\}}$, $p''.\dot{x}_i = p'.\dot{x}_i$, and $p''.\dot{x}_{i'} = p'.\dot{x}_{i'}$. Therefore, since $p' \in T_{\{i,i'\}}(d)$, Lemma 4.4.3, part 2, implies

that $p'' \in$ *disjoint-claimed-tracks*(i, i', d). From the above conditions, it follows that $p'' \in T_{\{i,i'\}}(d)$. Finally, Lemma 8.4.17 implies that $p'' \in$ *delay-safe*$_{\{i,i'\}}(d)$, as needed.

Next, consider the case in which $p \notin$ *safe*$_{\{i,i'\}}$. In this case, the $\texttt{snapshot}(y)_j$ action of the discrete controller automaton of GRAPH-PROT-PAIR$_{\{i,i'\}}$ sets the variable $send_j$ to either $\{i\}$, $\{i'\}$, or $\emptyset$ and, subsequently, enables either the $\texttt{protect}(\{i\})_j$ action, the $\texttt{protect}(\{i'\})_j$ action, or the $\texttt{protect}(\emptyset)_j$ action, respectively. However, when $p \notin$ *safe*$_{\{i,i'\}}$, the DC_j automaton sets the variable $send_j$ arbitrarily and, subsequently, enables an arbitrary output action. Therefore, the behavior of the discrete controller automaton of the protector GRAPH-PROT-PAIR$_{\{i,i'\}}$ is allowed by that of the DC_j automaton.

Therefore, the effects of the $\texttt{snapshot}(y)_j$ action of the implementation are allowed by its specification.

2. The $\texttt{protect}(C)_j$ actions, for $C \in \mathcal{P}(\{i, i'\})$, have identical effects in both discrete controller automata. When the $send_j$ variable matches either the set C, or the $\texttt{protect}(C)_j$ action, the action $\texttt{protect}(C)_j$ is executed and the $send_j$ variable is set to *null* in both discrete controller automata.
3. The environment action in both discrete controller automata is stuttering. It follows that the mapping between the states of the discrete controller automaton of GRAPH-PROT-PAIR$_{\{i,i'\}}$ and the DC_j automaton prior to and succeeding the execution of the environment action remains the same.

■

Corollary 8.4.19 *The protector* GRAPH-PROT-PAIR$_{\{i,i'\}}$ *guarantees that the automaton* GRAPH-VEHICLES *remains within* $G_{\{i,i'\}}$ *starting from* $S_{\{i,i'\}}$ *given* $R_{\{i,i'\}}$.

Proof: Follows directly from Lemma 8.4.18 and Theorem 3.2.9. ■

8.5 Protection System GRAPH-PROT

We now define the collision protector GRAPH-PROT. While considering the automaton GRAPH-PROT, we restrict the states of the GRAPH-VEHICLES automaton to $P_{not\text{-}overspeed}$ as defined in Section 4.2, *i.e.*, $R_{\text{GRAPH-PROT}} = P_{not\text{-}overspeed}$. Let $G_{\text{GRAPH-PROT}}$ and $S_{\text{GRAPH-PROT}}$ be the intersection of $G_{\{i,i'\}}$ and $S_{\{i,i'\}}$, for all $\{i, i'\}$, where $i, i' \in I, i \neq i'$, respectively, and GRAPH-PROT be the composition of GRAPH-PROT-PAIR$_{\{i,i'\}}$, for all $\{i, i'\}$, where $i, i' \in I, i \neq i'$. The protector GRAPH-PROT guarantees that GRAPH-VEHICLES remains within $G_{\text{GRAPH-PROT}}$ starting from $S_{\text{GRAPH-PROT}}$ given $R_{\text{GRAPH-PROT}}$. For reference, the formal definitions of the GRAPH-PROT automaton and the sets $G_{\text{GRAPH-PROT}}$, $S_{\text{GRAPH-PROT}}$, and $R_{\text{GRAPH-PROT}}$ are shown in Table 8.4.

Table 8.4 Formal definitions of GRAPH-PROT, $G_{\text{GRAPH-PROT}}$, $S_{\text{GRAPH-PROT}}$, and $R_{\text{GRAPH-PROT}}$.

$$\text{GRAPH-PROT} \equiv \prod_{i,i' \in I, i \neq i'} \text{GRAPH-PROT-PAIR}_{\{i,i'\}}$$

$$G_{\text{GRAPH-PROT}} \equiv \bigcap_{i,i' \in I, i \neq i'} G_{\{i,i'\}}$$

$$S_{\text{GRAPH-PROT}} \equiv \bigcap_{i,i' \in I, i \neq i'} S_{\{i,i'\}}$$

$$R_{\text{GRAPH-PROT}} \equiv P_{not\text{-}overspeed}$$

Lemma 8.5.1 *The protector* GRAPH-PROT *guarantees that the* GRAPH-VEHICLES *automaton remains within* $G_{\text{GRAPH-PROT}}$ *starting from* $S_{\text{GRAPH-PROT}}$ *given* $R_{\text{GRAPH-PROT}}$.

In the following proof, we show that all the states of an execution of $PP \times$ GRAPH-PROT starting from $S_{\text{GRAPH-PROT}}$ given $R_{\text{GRAPH-PROT}}$ are in $G_{\text{GRAPH-PROT}}$. This is done by applying Theorem 3.1.8 and showing that the second condition of the theorem does not hold.

Proof: Let α be any execution of the system $PP \times$ GRAPH-PROT starting from a state in $S_{\text{GRAPH-PROT}}$ and in which all states are in $R_{\text{GRAPH-PROT}}$.

From Theorem 3.1.8, one of the following holds:

1. Every state in α is in $G_{\text{GRAPH-PROT}} = \bigcap_{i,i' \in I, i \neq i'} G_{\{i,i'\}}$.
2. α can be written as $\alpha_1 \frown \alpha_2$, where
   (a) All state occurrences in α_1 except possibly the last state occurrence are in the set $G_{\text{GRAPH-PROT}} = \bigcap_{i,i' \in I, i \neq i'} G_{\{i,i'\}}$.
   (b) If the last state occurrence in α_1 is in $\overline{G_{\{i,i'\}}}$, for some $i, i' \in I, i \neq i'$, then there exists $i'', i''' \in I, i'' \neq i''', \{i'', i'''\} \neq \{i, i'\}$, such that the last state occurrence in α_1 is in $\overline{G_{\{i'',i'''\}}}$.
   (c) All state occurrences in α_2 except possibly the first state occurrence are in the set $\bigcap_{\{i'',i'''\} \in N} past(\overline{G_{\{i'',i'''\}}}, \alpha)$, for some $N \subseteq \{\{i, i'\} \mid i, i' \in I, i \neq i'\}$, where $|N| \geq 2$.

We proceed by showing that it is not possible to decompose α as $\alpha_1 \frown \alpha_2$ while satisfying the three aforementioned conditions.

The violation of $\bigcap_{i,i' \in I, i \neq i'} G_{\{i,i'\}}$ can only occur through the violation of at least one of the conditions $G_{\{i,i'\}}$, where $i, i' \in I, i \neq i'$. Moreover, each of these conditions are violated only through the execution of a `colliding-pair` action. Without loss of generality, suppose that the first condition that is violated in α is the condition $G_{\{i,i'\}}$, for some $i, i' \in I, i \neq i'$, and that such a violation has resulted through a `colliding-pair`(i, i') action. Let p and p' be the states of the system prior to and succeeding this `colliding-pair`(i, i') action, *i.e.*, $p, p' \in R_{\text{GRAPH-PROT}}$ such that $p \xrightarrow{\pi} p'$, where $\pi =$ `colliding-pair`(i, i'). Since the `colliding-pair`(i, i') action only sets the *collided*(i, i') variable to `True`, it follows that the state p' is in the set $\overline{G_{\{i,i'\}}} \bigcap \left(\bigcap_{i'',i''' \in I, i'' \neq i''', \{i'',i'''\} \neq \{i,i'\}} G_{\{i'',i'''\}} \right)$. Now, we attempt to decompose α as $\alpha_1 \frown \alpha_2$:

1. Suppose we split α at any state preceding the state p. Then the state p is in α_2. Since p' is the first state in which one of the conditions $G_{\{i'',i'''\}}$, for $i'', i''' \in I, i'' \neq i'''$, is violated, it is the case that $p \in \bigcap_{i'',i''' \in I, i'' \neq i'''} G_{\{i'',i'''\}}$ and there does not exist $N \subseteq \{\{i'', i'''\} \mid i'', i''' \in I, i'' \neq i'''\}$ such that $|N| \geq 2$ and $p \in \bigcap_{\{i'',i'''\} \in N} \mathit{past}(\overline{G_{\{i'',i'''\}}}, \alpha)$. Therefore, the third condition is violated and this decomposition of α is not valid.
2. Suppose we split α at the state p. Then the state p' is in α_2. Since p' is the first state in which one of the conditions $G_{\{i'',i'''\}}$, for $i'', i''' \in I, i'' \neq i'''$, is violated and since the state p' is in $\overline{G_{\{i,i'\}}} \bigcap \left(\bigcap_{i'',i''' \in I, i'' \neq i''', \{i'',i'''\} \neq \{i,i'\}} G_{\{i'',i'''\}} \right)$, it follows that there does not exist $N \subseteq \{\{i'', i'''\} \mid i'', i''' \in I, i'' \neq i'''\}$ such that $|N| \geq 2$ and $p' \in \bigcap_{\{i'',i'''\} \in N} \mathit{past}(\overline{G_{\{i'',i'''\}}}, \alpha)$. Therefore, the third condition is violated and this decomposition of α is not valid.
3. Suppose we split α at the state p'. Then p' is the last state of α_1 and the first state of α_2. However, $p' \in \overline{G_{\{i,i'\}}} \bigcap \left(\bigcap_{i'',i''' \in I, i'' \neq i''', \{i'',i'''\} \neq \{i,i'\}} G_{\{i'',i'''\}} \right)$. Therefore, the second condition is violated and this decomposition of α is not valid.
4. Suppose we split α at any state succeeding p'. Then the state p' is in α_1. Since $p' \in \overline{G_{\{i,i'\}}} \bigcap \left(\bigcap_{i'',i''' \in I, i'' \neq i''', \{i'',i'''\} \neq \{i,i'\}} G_{\{i'',i'''\}} \right)$, it follows that the state p' is not in the set $\bigcap_{i'',i''' \in I, i'' \neq i'''} G_{\{i'',i'''\}}$. Therefore, the first condition is violated and this decomposition of α is not valid.

Therefore, the execution α cannot be decomposed into any such α_1 and α_2. It follows that the first clause of Theorem 3.1.8 must hold; that is, every state in α is in $G_{\text{GRAPH-PROT}}$. This implies that the protector GRAPH-PROT guarantees $G_{\text{GRAPH-PROT}}$ in the GRAPH-VEHICLES automaton starting from $S_{\text{GRAPH-PROT}}$ given $R_{\text{GRAPH-PROT}}$. ■