Chapter 5

Example 1: Overspeed Protection System

In this chapter, we present a protector that prevents the vehicles of the VEHICLES automaton from exceeding a prespecified speed limit. In an actual system, speed limits may vary from one region of the track to another; in this thesis, we assume a single global speed limit $\dot{c}_{max}$. We define a protector, called OS-PROT, that enforces the speed limit on all vehicles, provided that they do not collide among themselves. This protector is defined as the composition of n separate copies of another protector called OS-PROT-SOLO$_i$, one copy for each vehicle $i \in I$. Each of the OS-PROT-SOLO$_i$ protectors, for $i \in I$, is an implementation of a particular instantiation of the abstract protector automaton of Section 3.2 and guarantees that the vehicle i does not exceed the speed limit.

5.1 Protection System OS-PROT-SOLO$_i$

The OS-PROT-SOLO$_i$ automata, for $i \in I$, are vehicle-wise overspeed protectors, each of which individually guarantees that the vehicle i, for which it is responsible, does not exceed the speed limit $\dot{c}_{max}$, provided that no collisions among the vehicles occur. Each of the OS-PROT-SOLO$_i$ protectors, for $i \in I$, 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 VEHICLES automaton of Figure 4.1. The port j and the sampling period d are defined to be the port and sampling period with which the protector OS-PROT-SOLO$_i$ communicates with the VEHICLES automaton. They are assumed arbitrary and are fixed for the rest of the chapter. The set R is defined to be the set $P_{not\text{-}collided}$ defined in Section 4.2. This definition restricts the reachable states of the VEHICLES automaton to states in which no collisions among the vehicles have occurred. The set of "good" states G is defined to be the set of states in which the vehicle i is at or below the speed limit, *i.e.*, $G = VALID - P_{overspeed(i)}$. The set of states S is defined to be the set $safe_{PP,R,G,j}$ 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. In Section 3.2.1, the definition of *safe* depended on the automaton *PP*, the sets R and G, and the port j which, at the time, were arbitrary. Here, they are defined to be the automaton VEHICLES, the sets $P_{not\text{-}collided}$ and $VALID - P_{overspeed(i)}$, and the port j, respectively; that is, we have specialized the definition of *safe* for these particular definitions of the automaton *PP*, the sets R and G, and the port j. In this chapter, we will use the notation R_i, G_i, and S_i to refer to the above definitions of the sets R, G, and S.

The OS-PROT-SOLO$_i$ protector automaton is an implementation of the abstract protector automaton $Abs(\text{VEHICLES}, S_i, R_i, G_i, j, d)$. As is the case for the abstract protector automaton Abs_j, we define the OS-PROT-SOLO$_i$ 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 5.1.

It is important to note that the abstract protector automaton $Abs(\text{VEHICLES}, S_i, R_i, G_i, j, d)$ complies with the assumptions made about the abstract protector in Section 3.2.1. In particular, since the vehicle velocity variables are output variables of the VEHICLES automaton, the set *safe* is Y_{VEHICLES}-determinable and actions that guarantee safety can be determined from the output variables Y_{VEHICLES} of the VEHICLES automaton (Axioms 3.2.4 and 3.2.5, respectively). Moreover, the sets R_i and G_i are Y_{VEHICLES}-determinable (Axioms 3.2.6 and 3.2.7, respectively) and the set of start states S_i is a subset of the set *safe* (Axiom 3.2.8), since S_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 plant *PP* remains within G starting from S given R. Similarly, the OS-PROT-SOLO$_i$ automaton guarantees that VEHICLES remains within G_i starting from S_i given R_i. This is shown in the following section.

5.2 Correctness of OS-PROT-SOLO$_i$

The main result to be shown is that OS-PROT-SOLO$_i \leq Abs(\text{VEHICLES}, S_i, R_i, G_i, j, d)$. However, since both OS-PROT-SOLO$_i$ and $Abs(\text{VEHICLES}, S_i, R_i, G_i, j, d)$ 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 OS-PROT-SOLO$_i$ of Figure 5.1 implements the discrete controller automaton $DC(\text{VEHICLES}, S_i, R_i, G_i, j, d)$ of

Figure 5.1 Discrete controller automaton for the protector OS-PROT-SOLO$_i$.

Actions:	Input:	e, the environment action (stuttering)
		$\texttt{snapshot}(y)_j$, for each valuation y of Y_{VEHICLES}
	Output:	$\texttt{brake}(i)_j$
		$\texttt{unbrake}(i)_j$
Variables:	Internal:	$send_j \in \{\texttt{brake}, \texttt{unbrake}, null\}$, initially $null$

Discrete Transitions:

$\texttt{snapshot}(y)_j$
Eff: if $(y.\dot{x}_i \leq \dot{c}_{max} - d\ddot{c}_{max})$ then
$send_j := \texttt{unbrake}$
else
$send_j := \texttt{brake}$

$\texttt{brake}(i)_j$
Pre: $send_j = \texttt{brake}$
Eff: $send_j := null$

$\texttt{unbrake}(i)_j$
Pre: $send_j = \texttt{unbrake}$
Eff: $send_j := null$

Trajectories:
$w.send_j \equiv null$

Figure 3.3. According to Theorem 2.6.1, this follows by showing that there exists a simulation relation between the states of the discrete controller automaton of OS-PROT-SOLO$_i$ and DC(VEHICLES, S_i, R_i, G_i, j, d). We first give some useful set definitions, then prove some lemmas, and finally show the existence of such a simulation relation.

In this section, we use the notation *future*$_i$, *safe*$_i$, *very-safe*$_i$, and *delay-safe*$_i$ 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 VEHICLES, the sets R_i and G_i, and the port j of the OS-PROT-SOLO$_i$ protector. Moreover, since the environment action of the 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 OS-PROT-SOLO$_i$. For reference, their formal definitions appear in Table 5.1.

Let W_i be the set of states of the VEHICLES automaton in which none of the vehicles have collided and the vehicle i is at or below the speed limit; that is, $W_i = R_i \cap G_i$. Let V_i be the set of states of the VEHICLES automaton in which none of the vehicles have collided, the vehicle i is at or below the speed limit, and the protector j is requesting the

Table 5.1 Sets used in the correctness proof of OS-PROT-SOLO$_i$.

$W_i \subseteq \mathit{VALID}$, for $i \in I$, defined by

$$W_i = R_i \cap G_i$$

$V_i \subseteq \mathit{VALID}$, for $i \in I$, defined by

$$V_i = R_i \cap G_i \cap P_{B_{ij}}$$

$T_i \subseteq \mathit{VALID}$, for $i \in I$, defined by

$$T_i = \{p \in R_i \cap G_i \mid p.\dot{x}_i \leq \dot{c}_{max} - d\ddot{c}_{max}\}$$

vehicle i to brake; that is, $V_i = R_i \cap G_i \cap P_{B_{ij}}$. Furthermore, let T_i be the set of states of the VEHICLES automaton in which none of the vehicles have collided, the vehicle i is at or below the speed limit, and the condition $\dot{x}_i \leq \dot{c}_{max} - d\ddot{c}_{max}$ is satisfied; that is, $T_i = \{p \in R_i \cap G_i \mid p.\dot{x}_i \leq \dot{c}_{max} - d\ddot{c}_{max}\}$.

In the following lemma, we show that if we restrict the states of the VEHICLES automaton to the set R_i and consider a state in which the vehicle i is at or below the speed limit and is being requested to brake by the protector j, then, provided that no new protective actions are issued by the protector j, the vehicle i remains at or below the speed limit thereafter.

Lemma 5.2.1 *$\mathit{future}_i(V_i, \mathbb{R}^{\geq 0}) \subseteq G_i$.*

Proof: Let α be an execution fragment of the VEHICLES automaton of n steps and trajectories, where $n \in \mathbb{N}$, that: starts in a state in V_i, is only comprised of states in R_i, and involves no input actions on port j. Letting p_{init} and p_{final} be the initial and final states of α, respectively, we must show that $p_{\mathit{final}} \in G_i$. 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, $p_{\mathit{final}} = p_{init}$. Since $p_{init} \in V_i$ and $V_i \subseteq G_i$, it follows that $p_{\mathit{final}} \in G_i$.

The inductive step involves showing that if α is an execution fragment of length $n = k+1$, for some $k \in \mathbb{N}$, then $p_{\mathit{final}} \in G_i$. 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 $p'_{\mathit{final}} \in G_i$. 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 discrete actions by cases:

1. the actions $\texttt{brake}(i)_j$ and $\texttt{unbrake}(i)_j$ 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. Therefore, it trivially follows that $p_{\mathit{final}} \in G_i$.
3. the actions $\texttt{colliding-pair}(i', i'')$, for $i', i'' \in I, i' \neq i''$, and $\texttt{collision-effects}(i''')$, for $i''' \in I$, are not enabled because α is only comprised of states in R_i; recall that $R_i = P_{\mathit{not\text{-}collided}}$.
4. the actions $\texttt{brake}(i')_{j'}$, $\texttt{unbrake}(i')_{j'}$, for $i' \in I, j' \in J, j' \neq j$, and $\texttt{brick-wall}(i'')$, for $i'' \in I, i'' \neq i$, do not affect the velocity of the vehicle i; that is, $p_{\mathit{final}}.\dot{x}_i = p'_{\mathit{final}}.\dot{x}_i$. From the induction hypothesis we have that $p'_{\mathit{final}} \in G_i$ and, therefore, it follows that $p_{\mathit{final}} \in G_i$.

In the case of a trajectory, since the execution fragment α starts in a state in $V_i \subseteq P_{B_{ij}}$ and the only action that can set the *brake-req*(i,j) variable to $\texttt{False}$ is not enabled throughout α, all states in α are in $P_{B_{ij}}$; that is, the vehicle i keeps braking throughout the execution fragment α. Therefore, since the vehicle i in state p'_{final} is in G_i, *i.e.*, at or below the speed limit, and the vehicle i is braking throughout the trajectory from p'_{final} to p_{final}, it trivially follows that the velocity of the vehicle i in p_{final} will be at or below the speed limit; that is, $p_{\mathit{final}} \in G_i$. ∎

In the following two lemmas, we use Lemma 5.2.1 to show that $V_i \subseteq \mathit{very\text{-}safe}_i$ and $V_i \subseteq \mathit{delay\text{-}safe}_i(t)$, for any $t \in \mathbb{R}^{\geq 0}$, respectively.

Lemma 5.2.2 $V_i \subseteq \mathit{very\text{-}safe}_i$.

Proof: From the definition of *very-safe* in Section 3.2.1, we must show that the condition $\mathit{future}_i(V_i, \mathbb{R}^{\geq 0}) \subseteq G_i$ is satisfied. This follows directly from Lemma 5.2.1. ∎

Lemma 5.2.3 *For any $t \in \mathbb{R}^{\geq 0}$, it is the case that $V_i \subseteq \mathit{delay\text{-}safe}_i(t)$.*

Proof: Follows directly from Lemma 5.2.2 and Lemma 3.2.5, part 1. ∎

In the following two lemmas and the subsequent corollary, we show that the sets W_i and safe_i are equal. First, we show that $W_i \subseteq \mathit{safe}_i$ and $\mathit{safe}_i \subseteq W_i$. Then the fact that $W_i = \mathit{safe}_i$ follows trivially.

Lemma 5.2.4 $W_i \subseteq \mathit{safe}_i$.

Proof: From the definition of *safe* in Section 3.2.1, we must show that any state $p \in W_i$ satisfies: (i) $\mathit{future}_i(p, 0) \subseteq G_i$, and (ii) there exists some action π such that for every $p', p'' \in R_i$ satisfying $p' \in \mathit{future}_i(p, 0)$ and $p' \xrightarrow{\pi} p''$, it is the case that $p'' \in \mathit{very\text{-}safe}_i$.

For the first condition, let α be an execution fragment of the VEHICLES automaton of n steps, where $n \in \mathbb{N}$, that: starts in a state in W_i, is only comprised of states in R_i, involves no input actions on port j, and has a limit time equal to zero. Letting p_{init} and p_{final} be the initial and final states of α, respectively, we must show that $p_{final} \in G_i$.

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 $p_{init} \in W_i$, it follows that $p_{final} \in G_i$.

The inductive step involves showing that if α is an execution fragment of length $n = k + 1$, for some $k \in \mathbb{N}$, then $p_{final} \in G_i$. 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 $p'_{final} \in G_i$. 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 $\mathtt{brake}(i)_j$ and $\mathtt{unbrake}(i)_j$ are not enabled because α involves no input actions on port j.
2. the $\mathtt{brick\text{-}wall}(i)$ action sets the velocity of the vehicle i to zero. Therefore, it trivially follows that $p_{final} \in G_i$.
3. the actions $\mathtt{colliding\text{-}pair}(i', i'')$, for $i', i'' \in I, i' \neq i''$, and $\mathtt{collision\text{-}effects}(i''')$, for $i''' \in I$, are not enabled because α is only comprised of states in R_i; recall that $R_i = P_{not\text{-}collided}$.
4. the actions $\mathtt{brake}(i')_{j'}$, $\mathtt{unbrake}(i')_{j'}$, for $i' \in I, j' \in J, j' \neq j$, and $\mathtt{brick\text{-}wall}(i'')$, for $i'' \in I, i'' \neq i$, do not affect the velocity of the vehicle i; that is, $p_{final}.\dot{x}_i = p'_{final}.\dot{x}_i$. However, from the induction hypothesis, it is the case that $p'_{final} \in G_i$. Therefore, it trivially follows that $p_{final} \in G_i$.

For the second condition, consider the action $\pi = \mathtt{brake}(i)_j$. The effect of this action is to set the internal variable $\mathit{brake\text{-}req}(i, j)$ to $\mathtt{True}$. Therefore, it is the case that $p'' \in P_{B_{ij}}$. From the proof of the first condition, it is the case that $p' \in G_i$, and since the $\mathtt{brake}(i)_j$ action does not affect the velocity of the vehicle i, it is also the case that $p'' \in G_i$. From the above conditions and the fact that $p'' \in R_i$, it follows that $p'' \in V_i$. Finally, Lemma 5.2.2 implies that $p'' \in \mathit{very\text{-}safe}_i$, as needed. ∎

Lemma 5.2.5 $\mathit{safe}_i \subseteq W_i$.

Proof: From Lemma 3.2.4, part 1, and the definition of *safe* in Section 3.2.1, it is the case that $safe_i \subseteq G_i$ and $safe_i \subseteq R_i$, respectively. It trivially follows that $safe_i \subseteq W_i$. ■

Corollary 5.2.6 $W_i = safe_i$.

Proof: Follows directly from Lemmas 5.2.4 and 5.2.5. ■

In the next three lemmas, we show that any state p in the set T_i is in the set $delay\text{-}safe_i(d)$; that is, any state R_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 and any state R_i-reachable from the state p in exactly an amount of time d through an execution fragment that involves no input actions on port j, is in the set $safe_i$.

Lemma 5.2.7 $future_i(T_i, [0, d]) \subseteq G_i$.

Proof: Let α be an execution fragment of the VEHICLES automaton of n steps and trajectories, where $n \in \mathbb{N}$, that: starts in a state in T_i, is only comprised of states in R_i, involves no input actions on port j, and has a limit time t that lies in the interval $[0, d]$. Letting p_{init} and p_{final} be the initial and final states of α, respectively, we must show that $p_{final} \in G_i$.

We use induction on the length n of the execution fragment α and the assertion $p_{final}.\dot{x}_i \leq p_{init}.\dot{x}_i + t\ddot{c}_{max}$ to show that $p_{final} \in G_i$.

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, $p_{final} = p_{init}$ and $p_{final}.\dot{x}_i = p_{init}.\dot{x}_i$. Moreover, since $t = 0$, it is the case that $t\ddot{c}_{max} = 0$. It trivially follows that $p_{final}.\dot{x}_i \leq p_{init}.\dot{x}_i + t\ddot{c}_{max}$.

The inductive step involves showing that if α is an execution fragment of length $n = k+1$, for some $k \in \mathbb{N}$, then $p_{final}.\dot{x}_i \leq p_{init}.\dot{x}_i + t\ddot{c}_{max}$. 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, and $t' \in [0, t]$ is the limit time of α', then it is the case that $p'_{final}.\dot{x}_i \leq p'_{init}.\dot{x}_i + t'\ddot{c}_{max}$. 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 $\texttt{brake}(i)_j$ and $\texttt{unbrake}(i)_j$ 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 since all vehicle velocities are restricted to be non-negative, it follows that $p_{\mathit{final}}.\dot{x}_i \leq p'_{\mathit{final}}.\dot{x}_i$. Moreover, from the induction hypothesis, we have $p'_{\mathit{final}}.\dot{x}_i \leq p'_{init}.\dot{x}_i + t'\ddot{c}_{max}$. Since $p_{init} = p'_{init}$ and $t = t'$, it follows that $p_{\mathit{final}}.\dot{x}_i \leq p_{init}.\dot{x}_i + t\ddot{c}_{max}$.
3. the actions $\texttt{colliding-pair}(i', i'')$, for $i', i'' \in I, i' \neq i''$, and $\texttt{collision-effects}(i''')$, for $i''' \in I$, are not enabled because α is only comprised of states in R_i; recall that $R_i = P_{\mathit{not\text{-}collided}}$.
4. the actions $\texttt{brake}(i')_{j'}$, $\texttt{unbrake}(i')_{j'}$, for $i' \in I, j' \in J, j' \neq j$, and $\texttt{brick-wall}(i'')$, for $i'' \in I, i'' \neq i$, do not affect the velocity of the vehicle i; that is, $p_{\mathit{final}}.\dot{x}_i = p'_{\mathit{final}}.\dot{x}_i$. Moreover, from the induction hypothesis we have $p'_{\mathit{final}}.\dot{x}_i \leq p'_{init}.\dot{x}_i + t'\ddot{c}_{max}$. Since $p_{init} = p'_{init}$ and $t = t'$, it follows that $p_{\mathit{final}}.\dot{x}_i \leq p_{init}.\dot{x}_i + t\ddot{c}_{max}$.

In the case of a trajectory, since the change in velocity is equal to the integral of the acceleration and the acceleration is bounded from above by the quantity $\ddot{c}_{max}$, it is the case that $p_{\mathit{final}}.\dot{x}_i \leq p'_{\mathit{final}}.\dot{x}_i + (t - t')\ddot{c}_{max}$. Moreover, from the induction hypothesis we have $p'_{\mathit{final}}.\dot{x}_i \leq p'_{init}.\dot{x}_i + t'\ddot{c}_{max}$. Since $p_{init} = p'_{init}$, it follows that $p_{\mathit{final}}.\dot{x}_i \leq p_{init}.\dot{x}_i + t\ddot{c}_{max}$. This result completes the induction.

Since $p_{init} \in T_i$, it is the case that $p_{init}.\dot{x}_i \leq \dot{c}_{max} - d\ddot{c}_{max}$. Moreover, from the above induction we have $p_{\mathit{final}}.\dot{x}_i \leq p_{init}.\dot{x}_i + t\ddot{c}_{max}$. Therefore, $p_{\mathit{final}}.\dot{x}_i \leq \dot{c}_{max} - (d - t)\ddot{c}_{max}$, and since $d - t \geq 0$ and $\ddot{c}_{max} > 0$, it follows that $p_{\mathit{final}}.\dot{x}_i \leq \dot{c}_{max}$; that is, $p_{\mathit{final}} \in G_i$, as needed. ■

Lemma 5.2.8 $\mathit{future}_i(T_i, 0) \subseteq T_i$.

Proof: From Lemma 5.2.4 and the definition of T_i it is the case that $T_i \subseteq \mathit{safe}_i$. Therefore, from Lemma 3.2.4, part 2, it follows that $\mathit{future}_i(T_i, 0) \subseteq \mathit{safe}_i$. Moreover, Lemma 5.2.5 implies that $\mathit{future}_i(T_i, 0) \subseteq W_i$. It remains to be shown that for all $p, p' \in R_i$ such that $p \in T_i$ and $p' \in \mathit{future}_i(p, 0)$, it is the case that $p'.\dot{x}_i \leq \dot{c}_{max} - d\ddot{c}_{max}$.

Because of the non-negative constraint on the vehicle velocities, the only discrete action that could potentially increase the velocity of the vehicle i is the $\texttt{collision-effects}(i)$ action. However, the $\texttt{collision-effects}(i)$ action is not enabled because the function $\mathit{future}_i(p, 0)$ only considers R_i-reachable states. If follows that $p'.\dot{x}_i \leq p.\dot{x}_i$. Moreover, since $p \in T_i$, it is the case that $p.\dot{x}_i \leq \dot{c}_{max} - d\ddot{c}_{max}$. It trivially follows that $p'.\dot{x}_i \leq \dot{c}_{max} - d\ddot{c}_{max}$, as needed. ■

Lemma 5.2.9 $T_i \subseteq \mathit{delay\text{-}safe}_i(d)$.

Proof: We must show that $\mathit{future}_i(T_i, [0, d]) \subseteq G_i$ and $\mathit{future}_i(T_i, d) \subseteq \mathit{safe}_i$. The first condition follows directly from Lemma 5.2.7. For the second condition, from Lemma 3.2.1, part 1, we have that $\mathit{future}_i(T_i, d) \subseteq \mathit{future}_i(T_i, [0, d])$. Therefore, from Lemma 5.2.7 it follows that $\mathit{future}_i(T_i, d) \subseteq G_i$. Moreover, since $\mathit{future}_i(T_i, d)$ restricts the reachable states to the set R_i, it is the case that $\mathit{future}_i(T_i, d) \subseteq R_i$. Therefore, it is the case that $\mathit{future}_i(T_i, d) \subseteq W_i$ and from Lemma 5.2.4 it follows that $\mathit{future}_i(T_i, d) \subseteq \mathit{safe}_i$, as needed. ■

In the following lemma, we show that the OS-PROT-SOLO$_i$ protector implements the protector $\mathit{Abs}(\text{VEHICLES}, S_i, R_i, G_i, j, d)$. Since the protector automata OS-PROT-SOLO$_i$ 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 OS-PROT-SOLO$_i$ implements the discrete controller automaton $\mathit{DC}(\text{VEHICLES}, S_i, R_i, G_i, j, d)$.

Lemma 5.2.10 OS-PROT-SOLO$_i \leq \mathit{Abs}(\text{VEHICLES}, S_i, R_i, G_i, j, d)$.

Proof: As noted above, both the OS-PROT-SOLO$_i$ 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 OS-PROT-SOLO$_i$ implements DC_j. This is shown by a simulation from the discrete controller automaton of OS-PROT-SOLO$_i$ to DC_j.

The mapping between the states of the discrete controller automaton of OS-PROT-SOLO$_i$ and DC_j is almost the identity. In the discrete controller automaton of OS-PROT-SOLO$_i$, the variable send_j is equal to either one of the labels `brake` and `unbrake`, or the value *null*. In the abstract discrete controller automaton, these valuations simply map to either the actions $\texttt{brake}(i)_j$ and $\texttt{unbrake}(i)_j$, or the value *null*, respectively.

The start states for the discrete controller automaton of OS-PROT-SOLO$_i$ and DC_j are the states in which $\mathit{send}_j = \mathit{null}$. These are mapped 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 VEHICLES automaton that corresponds to y, *i.e.*, $p \in \mathit{VALID}$ and $p \lceil Y_{\text{VEHICLES}} = y$.

1. The $\texttt{snapshot}(y)_j$ action of the implementation sets send_j to `brake`, or `unbrake`. 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 $p', p'' \in R_i$ such that $p' \in \mathit{future}_i(p, 0)$ and $p' \xrightarrow{\pi} p''$, it is the case that $p'' \in \mathit{delay\text{-}safe}_i(d)$, if $p \in \mathit{safe}_i$, and (ii) is an arbitrary output action of the implementation, otherwise.

   First, consider the case in which $p \in \mathit{safe}_i$. Since Corollary 5.2.6 implies that $p \in W_i$, the discrete controller automaton of OS-PROT-SOLO$_i$ sets the variable $send_j$ according to whether the state p is in T_i, or not.

On one hand, if $p \notin T_i$ then the discrete controller automaton of OS-PROT-SOLO$_i$ sets the variable $send_j$ to `brake` and the $\texttt{brake}(i)_j$ action is enabled. However, since $p \in safe_i$, Lemma 3.2.4, part 2, implies that $p' \in safe_i$ and from Corollary 5.2.6 it follows that $p' \in W_i$. Moreover, since the $\texttt{brake}(i)_j$ action sets the *brake-req*(i, j) variable to `True` and affects neither the velocity of any of the vehicles, nor any of the *collided* variables, it is the case that $p'' \in R_i \cap G_i \cap P_{B_{ij}}$, *i.e.*, $p'' \in V_i$. Finally, from Lemma 5.2.3, it follows that $p'' \in delay\text{-}safe_i(d)$, as needed.

On the other hand, if $p \in T_i$ then the discrete controller automaton of OS-PROT-SOLO$_i$ sets the variable $send_j$ to `unbrake` and the $\texttt{unbrake}(i)_j$ action is enabled. From Lemma 5.2.8, it follows that $p' \in T_i$. Moreover, since the $\texttt{unbrake}(i)_j$ action sets the *brake-req*(i, j) variable to `False` and affects neither the velocity of the vehicle i, nor any of the *collided* variables, it is the case that $p'' \in T_i$. Finally, from Lemma 5.2.9, it follows that $p'' \in delay\text{-}safe_i(d)$, as needed.

Next, consider the case in which $p \notin safe_i$. In this case, the $\texttt{snapshot}(y)_j$ action of the discrete controller automaton of OS-PROT-SOLO$_i$ sets the variable $send_j$ to either `brake` or `unbrake` and, subsequently, enables either the action $\texttt{brake}(i)_j$ or the action $\texttt{unbrake}(i)_j$. However, when $p \notin safe_i$, 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 OS-PROT-SOLO$_i$ 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 $\texttt{brake}(i)_j$ and $\texttt{unbrake}(i)_j$ actions have identical effects in both discrete controller automata. When the $send_j$ variable matches the labels `brake` and `unbrake`, or the actions $\texttt{brake}(i)_j$ and $\texttt{unbrake}(i)_j$, the respective action is performed and the $send_j$ variable is set to the value *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 OS-PROT-SOLO$_i$ and the DC_j automaton prior to and succeeding the execution of the environment action remains the same.

■

Corollary 5.2.11 *The protector* OS-PROT-SOLO$_i$ *guarantees* G_i *in the* VEHICLES *automaton starting from* S_i *given* R_i.

Proof: Follows directly from Lemma 5.2.10 and Theorem 3.2.9. ■

Table 5.2 Formal definitions of OS-PROT, $G_{\text{OS-PROT}}$, $S_{\text{OS-PROT}}$, and $R_{\text{OS-PROT}}$.

$$\text{OS-PROT} \equiv \prod_{i \in I} \text{OS-PROT-SOLO}_i$$

$$G_{\text{OS-PROT}} \equiv \bigcap_{i \in I} G_i$$

$$S_{\text{OS-PROT}} \equiv \bigcap_{i \in I} S_i$$

$$R_{\text{OS-PROT}} \equiv P_{not\text{-}collided}$$

5.3 Protection System OS-PROT

We now define the overspeed protector OS-PROT. As in the vehicle-wise case, we restrict the states of the VEHICLES automaton to the set $P_{not\text{-}collided}$ as defined in Section 4.2, *i.e.*, $R_{\text{OS-PROT}} = P_{not\text{-}collided}$. Let $G_{\text{OS-PROT}}$ and $S_{\text{OS-PROT}}$ be the intersection of the sets G_i and S_i, for all $i \in I$, respectively, and OS-PROT be the composition of the protectors OS-PROT-SOLO$_i$, for all $i \in I$. The protector OS-PROT guarantees that the VEHICLES automaton remains within $G_{\text{OS-PROT}}$ starting from $S_{\text{OS-PROT}}$ given $R_{\text{OS-PROT}}$. For reference, The formal definition of the OS-PROT automaton and of the sets $G_{\text{OS-PROT}}$, $S_{\text{OS-PROT}}$, and $R_{\text{OS-PROT}}$ are shown in Table 5.2.

Corollary 5.3.1 *The protector* OS-PROT *guarantees* $G_{\text{OS-PROT}}$ *in the* VEHICLES *automaton starting from* $S_{\text{OS-PROT}}$ *given* $R_{\text{OS-PROT}}$.

Proof: Follows directly from Corollary 5.2.11 and Theorem 3.1.4. ■