Hybrid Systems: Computation and Control
Berkeley, California
April 13–15, 1998

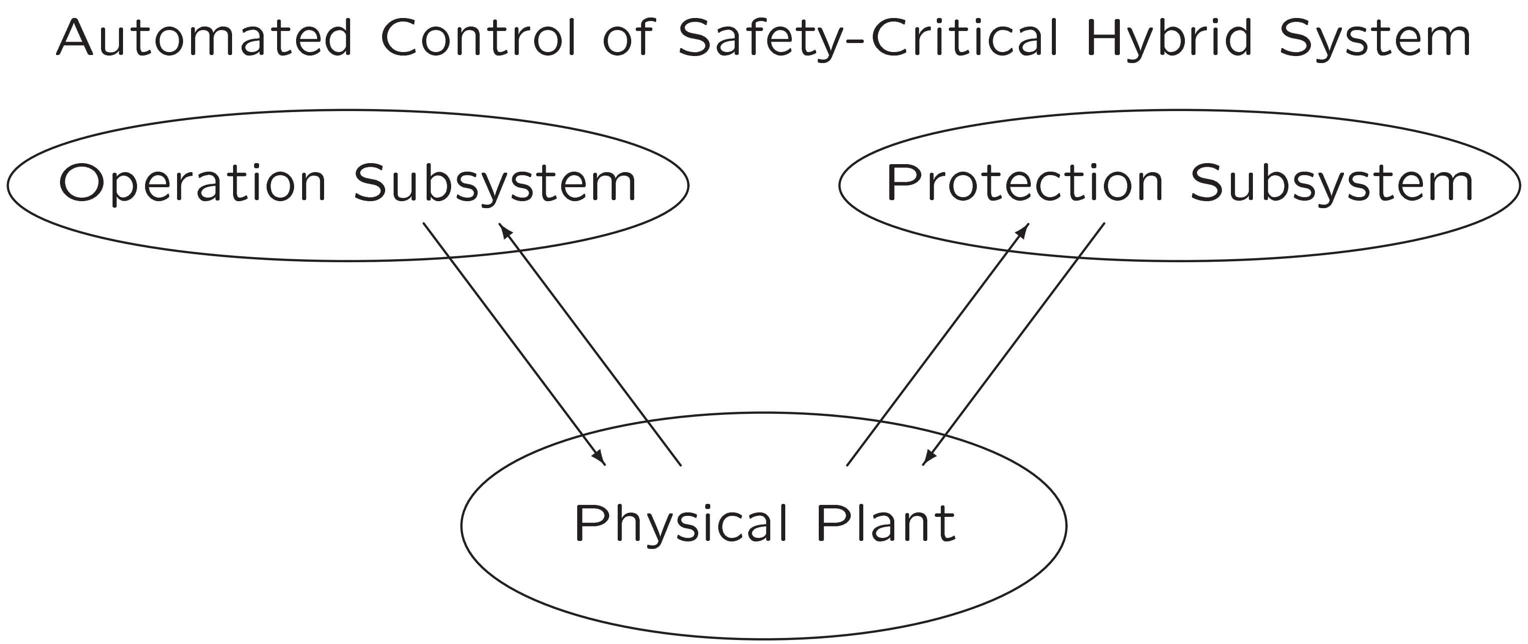
Formal Verification of Safety-Critical Hybrid Systems

Carl Livadas and Nancy A. Lynch

April 14, 1998

Theory of Distributed Systems
Laboratory for Computer Science
Massachusetts Institute of Technology

Automated Operation and Protection of Safety-Critical Hybrid Systems



Operation Subsystem

- complex
- normal operation

Goal: Performance

Protection Subsystem

- simple and reliable
- emergency operation

Goal: Safety

Why use Formal Methods?

Motivation:

- Safety-critical nature of hybrid systems.
- Inadequacy of current validation techniques.

Formal Approach:

- Obtain clear and precise system specifications.
- Take advantage of modular (de)composition.
- Use levels of abstraction / model refinement.
- Inherit proof and reasoning techniques from CS.

Goal:

- Address safety concerns through formal reasoning.

Formal Model: Hybrid I/O Automata

Lynch, Segala, Vaandrager, and Weinberg

Hybrid I/O Automaton: $A = (U, X, Y, \Sigma^{in}, \Sigma^{int}, \Sigma^{out}, \Theta, \mathcal{D}, \mathcal{W})$

- Variables V (Input U/Internal X/Output Y)
- Discrete Actions Σ (Input/Internal/Output)
- Initial States Θ
- Discrete Transitions $\mathcal{D}$ (discrete behavior)
- Trajectories $\mathcal{W}$ (continuous behavior)

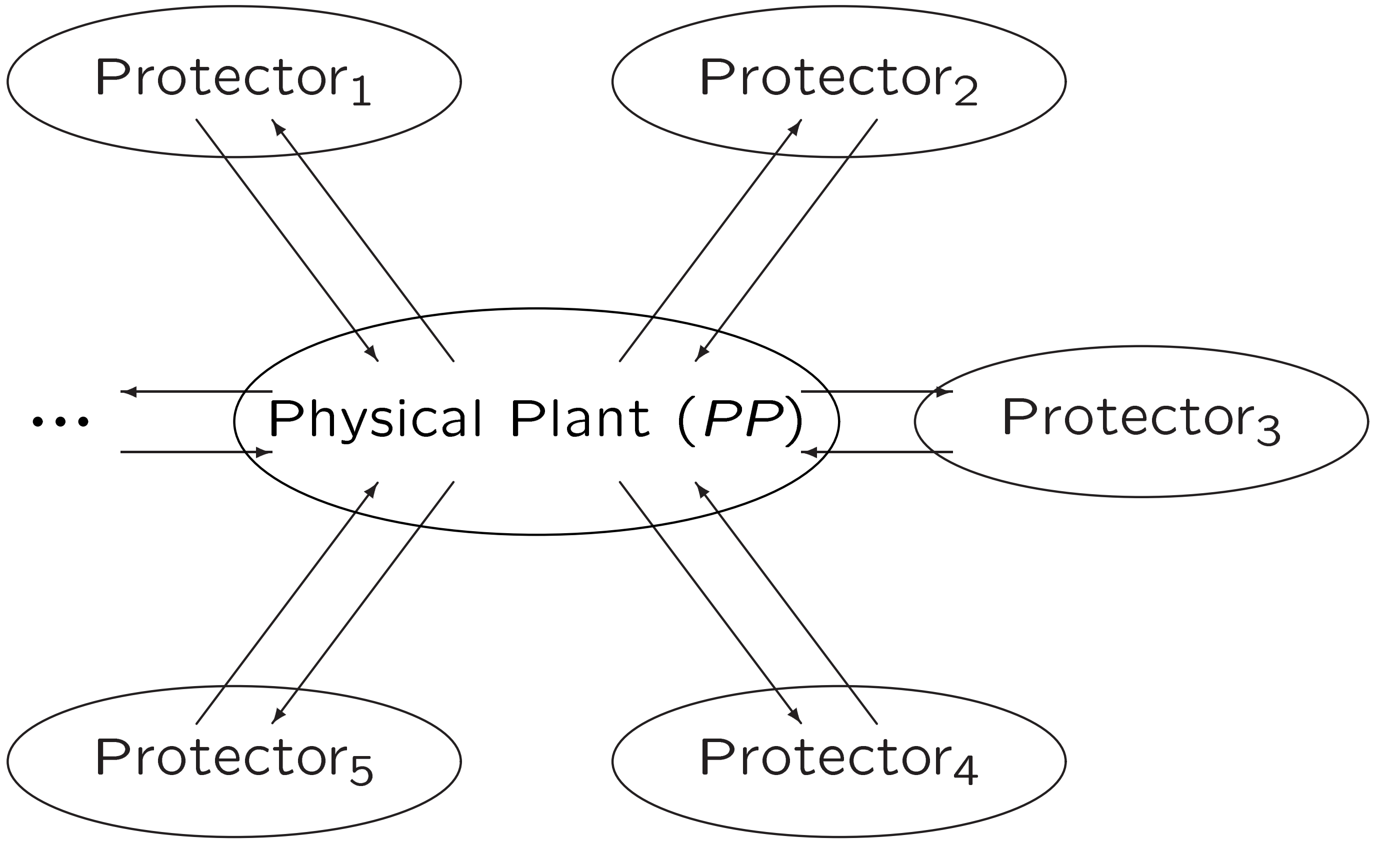
Hybrid Executions/Execution Fragments (HIOA Behavior)

$\alpha = w_0 a_1 w_1 a_2 w_2 \cdots$
where $w_i \in \mathcal{W}$ and $a_i \in \Sigma$ [α.*fstate* $\in \Theta$].

Hybrid Traces (HIOA External Behavior)

$\gamma = (w_0 \downarrow E)\mathit{vis}(a_1)(w_1 \downarrow E)\mathit{vis}(a_2)(w_2 \downarrow E) \cdots$
where $E = U \cup Y$.

Protected Physical Plant Model



Assumption: Protector$_i$ guarantees G_i in *PP* from S_i given R_i.
$G_i \equiv$ Guarantee, $R_i \equiv$ Reliance, $S_i \equiv$ Initial.

Goal: Composition of the Protector$_i$ automata, for all i, guarantees $G = \bigcap_i G_i$ from S given R.

Questions: Does composition hold? What are S and R?

Protector Composition

Composition Theorem I (Independent Protector Composition)

Suppose that A_1 and A_2 are protector automata for *PP*. If A_1 guarantees G_1 from S_1 given R_1 and A_2 guarantees G_2 from S_2 given R_2, then $A_1 \times A_2$ guarantees $G_1 \cap G_2$ from $S_1 \cap S_2$ given $R_1 \cap R_2$.

Composition Theorem II (One-Way Protector Dependence)

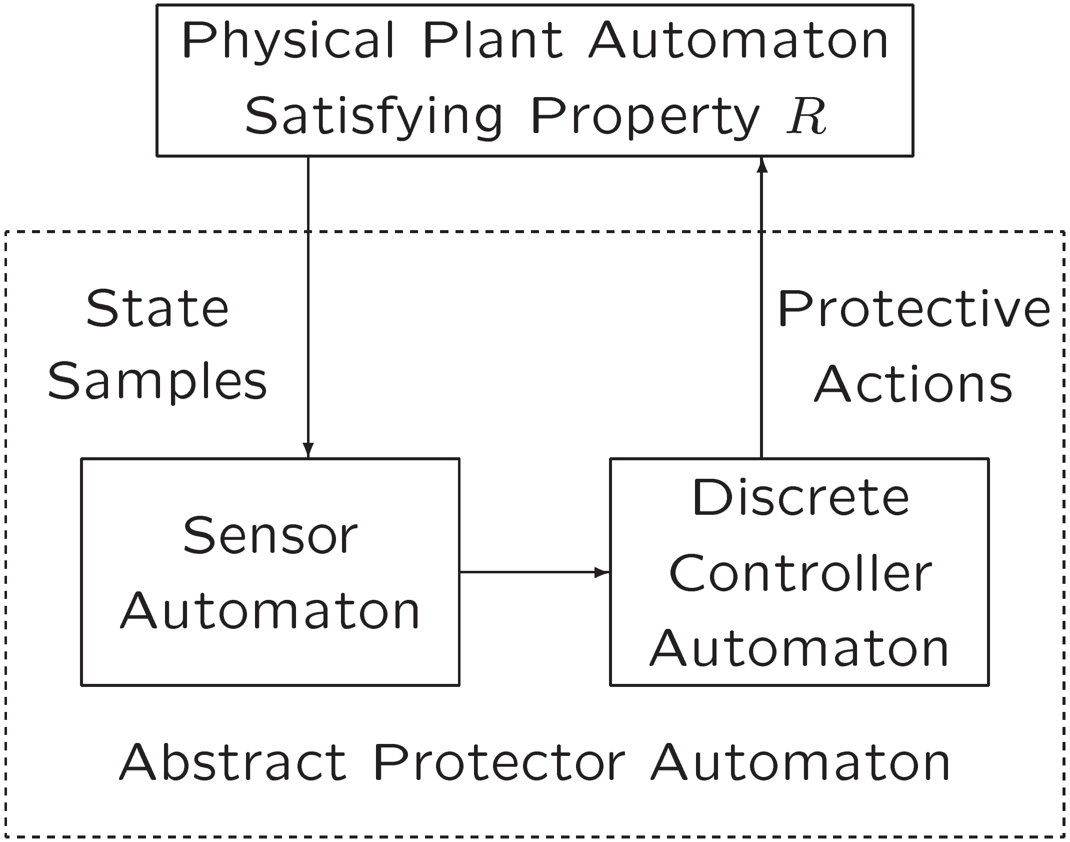
Suppose that A_1 and A_2 are protector automata for *PP*. If A_1 guarantees G_1 from S_1 given R_1 and A_2 guarantees G_2 from S_2 given $R_2 \cap G_1$, then $A_1 \times A_2$ guarantees $G_1 \cap G_2$ from $S_1 \cap S_2$ given $R_1 \cap R_2$.

Composition Theorem III (Two-Way Protector Dependence)

Suppose that A_1 and A_2 are protector automata for *PP*, such that A_1 guarantees G_1 from S_1 given $R_1 \cap G_2$ and A_2 guarantees G_2 from S_2 given $R_2 \cap G_1$. If α is any execution of the system $PP \times A_1 \times A_2$, starting from a state in $S_1 \cap S_2$ and in which all states are in $R_1 \cap R_2$, then, either

1. Every state in α is in $G_1 \cap G_2$, or
2. (Informally) G_1 and G_2 are both invalidated at, or after, a particular instant in time within α.

Abstract Parameterized Protector Automaton



Parameterized wrt:

- Physical Plant (*PP*)
- Sets R, S, and G
- Port j
- Sampling Interval d

Abstract Protector: Guarantees a property G in *PP* starting from S given R.

Actual Protector: A particular implementation of an instantiation of the abstract parameterized protector.

Correctness Proof: Actual protector **implements** instantiation of abstract parameterized protector.

Abstract Protector Assumptions

1. $U_{PP_j} = \emptyset$, for all $j \in J$.
2. $\Sigma^{out}_{PP_j} = \emptyset$, for all $j \in J$.
3. $\mathit{no\text{-}op}_{PP,R,j}(p) \neq \emptyset$, for all $p \in R$.
4. $\mathit{safe}_{PP,R,G,j}$ is Y_{PP}-determinable.
5. There exists a function, $\mathtt{decision}$, from valuations of Y_{PP} to $\Sigma^{in}_{PP_j}$ such that for any $y \in \mathbf{Y_{PP}}$ and $p \in R$ satisfying $p \lceil Y_{PP} = y$, it is the case that if $y \in \mathit{safe}_{PP,R,G,j}$ then $p \in \mathit{safe}_{PP,R,G,j}(\mathtt{decision}(y))$.
6. R and G are Y_{PP}-determinable.
7. $S \subseteq \mathit{safe}_{PP,R,G,j}$.

Protector Terminology

future$_{PP,R,j} : \mathcal{P}(R) \times \mathcal{P}(\mathbb{R}^{\geq 0}) \to \mathcal{P}(R)$, defined by:

$p \in$ *future*$_{PP,R,j}(P,T)$, where $P \subseteq R$ and $T \subseteq \mathbb{R}^{\geq 0}$, if and only if p is R-reachable from some $p' \in P$ via a finite execution fragment α of *PP* with no input actions on port j and with α.*ltime* $\in T$.

very-safe$_{PP,R,G,j} \subseteq R$, defined by:

$p \in$ *very-safe*$_{PP,R,G,j}$ if and only if *future*$_{PP,R,j}(p, \mathbb{R}^{\geq 0}) \subseteq G$.

safe$_{PP,R,G,j} : \Sigma^{in}_{PP_j} \to \mathcal{P}(R)$, defined by:

$p \in$ *safe*$_{PP,R,G,j}(\pi)$ if and only if both of the following hold:

1. *future*$_{PP,R,j}(p, 0) \subseteq G$.
2. For every $p', p'' \in R$ such that $p' \in$ *future*$_{PP,R,j}(p,0)$ and $p' \xrightarrow{\pi}_{PP} p''$, it is the case that $p'' \in$ *very-safe*$_{PP,R,G,j}$.

delay-safe$_{PP,R,G,j} : \mathbb{R}^{\geq 0} \to \mathcal{P}(R)$, defined by:

$p \in$ *delay-safe*$_{PP,R,G,j}(t)$ if and only if both of the following hold:

1. *future*$_{PP,R,j}(p, [0,t]) \subseteq G$.
2. *future*$_{PP,R,j}(p, t) \subseteq$ *safe*$_{PP,R,G,j}$.

Abstract Sensor Automaton

Actions:

Input: e, the environment action

Output: $\mathtt{snapshot}(y)_j$, for each valuation y of Y_{PP}

Variables:

Input: $u \in \mathit{type}(u)$, for all $u \in Y_{PP}$,
initially $u \in \mathit{type}(u)$, for each $u \in Y_{PP}$

Internal: $\mathit{now}_j \in \mathbb{R}^{\geq 0}$, initially 0
$\mathit{next\text{-}snap}_j \in \mathbb{R}^{\geq 0}$, initially 0

Discrete Transitions:

e
Eff: $Y_{PP} :\in \boldsymbol{Y}_{\mathbf{PP}}$

$\mathtt{snapshot}(y)_j$
Pre: $\mathit{next\text{-}snap}_j = \mathit{now}_j$
y is current valuation of Y_{PP}
Eff: $Y_{PP} :\in \boldsymbol{Y}_{\mathbf{PP}}$
$\mathit{next\text{-}snap}_j := \mathit{now}_j + d$

Trajectories:

for all $u \in Y_{PP}$
u assumes arbitrary values in $\mathit{type}(u)$ throughout w
$\mathit{next\text{-}snap}_j$ is constant throughout w
for all $t \in T_I$
$w(t).\mathit{now}_j = w(0).\mathit{now}_j + t$
$w(t).\mathit{now}_j \leq w(t).\mathit{next\text{-}snap}_j$

Abstract Discrete Controller Automaton

Actions:

Input: e, the environment action (stuttering)

$\mathtt{snapshot}(y)_j$, for each valuation y of Y_{PP}

Output: π, for all $\pi \in \Sigma^{in}_{PP_j}$

Variables:

Internal: $send_j \in \Sigma^{in}_{PP_j} \cup \{null\}$, initially $null$

Discrete Transitions:

e

Eff: None

π

Pre: $send_j = \pi$

Eff: $send_j := null$

$\mathtt{snapshot}(y)_j$

Eff: if $y \in safe_{PP,R,G,j}$ then

$send_j :\in \{\phi \in \Sigma^{in}_{PP_j} \mid$
$\forall\, p, p', p'' \in R$ such that
$p \lceil Y_{PP} = y$, $p' \in future_{PP,R,j}(p, 0)$,
and $p' \xrightarrow{\phi}_{PP} p''$,
it is the case that
$p'' \in delay\text{-}safe_{PP,R,G,j}(d)\}$

else

$send_j :\in \Sigma^{in}_{PP_j}$

Trajectories:

$w.send_j \equiv null$

The PRT 2000™

Characteristics:

- Point-to-point Rapid Transit
- Y-shaped Merges and Diverges
- Automated Operation and Protection
- Stringent Safety Requirements

Modeling Assumptions:

- Brick wall stop (conservative)
- Full state sampling (uncertain?)
- Binary brakes (either on or off) [$\ddot{c}_{brake}$]
- Pairwise collisions (realistic?)
- Non-negative velocities (realistic?)
- Acceleration in [$\ddot{c}_{min}, \ddot{c}_{max}$]

Physical Plant Automaton

The *PP* Automaton:

Actions:

Input:

$\texttt{brake}(i)_j$, $\forall\ i \in I, j \in J$
$\texttt{unbrake}(i)_j$, $\forall\ i \in I, j \in J$

Internal:

$\texttt{brick-wall}(i)$, $\forall\ i \in I$
$\texttt{colliding-pair}(i, i')$, $\forall\ i, i' \in I, i' \neq i$
$\texttt{collision-effects}(i)$, $\forall\ i \in I$

Variables

Output:

$x_i \in \mathbb{R}$, $\forall\ i \in I$, initially $x_i \in \mathbb{R}$
$\dot{x}_i \in \mathbb{R}$, $\forall\ i \in I$, initially $\dot{x}_i \in \mathbb{R}$
$\ddot{x}_i \in \mathbb{R}$, $\forall\ i \in I$, initially $\ddot{x}_i \in \mathbb{R}$

Internal:

brake$(i) \in$ `Bool`, $\forall\ i \in I$,
initially `False`
brake-req$(i, j) \in$ `Bool`, $\forall\ i \in I, j \in J$,
initially `False`
collided$(i, i') \in$ `Bool`, $\forall\ i, i' \in I, i' \neq i$,
initially `False`

subject to *VALID*

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

1. $\nexists\ i, i' \in I, i \neq i'$, s.t. the set $E_i \cap E_{i'}$ is a positive length closed interval of $\mathbb{R}$.
2. $\dot{x}_i \geq 0$, $\forall i \in I$.
3. If $\neg$*collided*$(*, i, *)$ then $\ddot{x}_i \in [\ddot{c}_{min}, \ddot{c}_{max}]$, $\forall i \in I$.
4. If $\neg$*collided*$(*, i, *) \wedge$ *brake*(i) then if $\dot{x}_i = 0$ then $\ddot{x}_i = 0$ else $\ddot{x}_i = \ddot{c}_{brake}$, $\forall i \in I$.

Physical Plant Automaton

Discrete Transitions:

$\texttt{brake}(i)_j$
Eff: *brake-req*(i,j) := $\texttt{True}$
if $\neg$*brake*(i) then
brake(i) := $\texttt{True}$
if $\dot{x}_i = 0$ then $\ddot{x}_i := 0$
else $\ddot{x}_i := \ddot{c}_{brake}$

$\texttt{unbrake}(i)_j$
Eff: *brake-req*(i,j) := $\texttt{False}$
if *brake*(i)
$\wedge(\neg \vee_{k \in J}$ *brake-req*$(i,k))$
then
brake(i) := $\texttt{False}$
$\ddot{x}_i :\in [\ddot{c}_{min}, \ddot{c}_{max}]$

$\texttt{colliding-pair}(i,i')$
Pre: $\neg$*collided*(i,i')
$\wedge(E_i \cap E_{i'} \neq \emptyset)$
$\wedge(x_i < \min(E_i \cap E_{i'}))$
Eff: *collided*(i,i') := $\texttt{True}$

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

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

Physical Plant Automaton

Trajectories:

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

Derived Variables:

E_i, for $i \in I$, is the section of the track that the vehicle i "occupies".

O_i, for $i \in I$, is the section of the track that the vehicle i "owns".

$C_i(t)$, for $i \in I$ and $t \in \mathbb{R}^{\geq 0}$, is the section of the track that the vehicle i "claims" within t time units.

Overspeed Protection

Structure: One protector per vehicle.

Assumption: No vehicle collisions occur.

Strategy: Issue emergency brake command when vehicle is capable of exceeding the speed limit within d time units.

Independent? YES (Composition Theorem I)

Overspeed Protector (Discrete Controller)

Actions:

Input: $\texttt{snapshot}(y)_j$, where $y \in Y_{\text{vehicles}}$

Output: $\texttt{brake}(i)_j$

$\texttt{unbrake}(i)_j$

Variables:

Internal: $\mathit{send}_j \in \{\texttt{brake}, \texttt{unbrake}, \mathit{null}\}$, initially *null*

Discrete Transitions:

$\texttt{snapshot}(y)_j$

Eff: if $(y.\dot{x}_i \leq \dot{c}_{max} - d\ddot{c}_{max})$ then

$\mathit{send}_j := \texttt{unbrake}$

else

$\mathit{send}_j := \texttt{brake}$

$\texttt{brake}(i)_j$

Pre: $\mathit{send}_j = \texttt{brake}$

Eff: $\mathit{send}_j := \mathit{null}$

$\texttt{unbrake}(i)_j$

Pre: $\mathit{send}_j = \texttt{unbrake}$

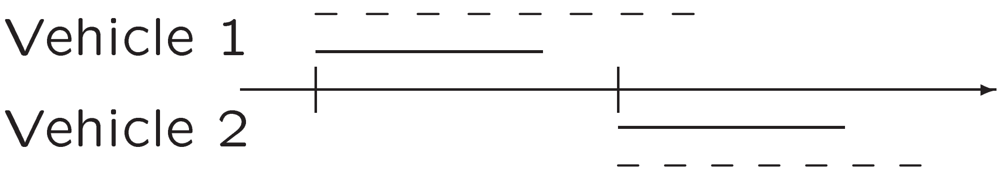
Eff: $\mathit{send}_j := \mathit{null}$

Trajectories:

$w.\mathit{send}_j \equiv \mathit{null}$

Collision Avoidance (Single Track)

Structure: Single track (one protector per vehicle)



——	Section of the track "owned" by the vehicle. If overlapping then collision is imminent.
– – –	Section of the track "claimed" by the vehicle. If overlapping then collision is possible.

Strategy: Issue emergency brake command when potential for collision exists.

Assumption:

- None of the vehicles exceed the speed limit.
- None of the other vehicles (vehicle pairs) collide into other vehicles (among themselves).

Independent? NO (Composition Theorem III)

Collision Avoidance (Single Track DC)

Actions:

Input: $\texttt{snapshot}(y)_j$, where $y \in Y_{\text{vehicles}}$

Output: $\texttt{brake}(i)_j$

$\texttt{unbrake}(i)_j$

Variables:

Internal: $\mathit{send}_j \in \{\texttt{brake}, \texttt{unbrake}, \mathit{null}\}$, initially *null*

Discrete Transitions:

$\texttt{snapshot}(y)_j$

Eff: if $\exists\, i' \in I, i' \neq i$ such that

$y \notin \mathit{disjoint\text{-}claimed\text{-}tracks}(i, i', d) \wedge (y.x_i < y.x_{i'})$

then

$\mathit{send}_j := \texttt{brake}$

else

$\mathit{send}_j := \texttt{unbrake}$

$\texttt{brake}(i)_j$

Pre: $\mathit{send}_j = \texttt{brake}$

Eff: $\mathit{send}_j := \mathit{null}$

$\texttt{unbrake}(i)_j$

Pre: $\mathit{send}_j = \texttt{unbrake}$

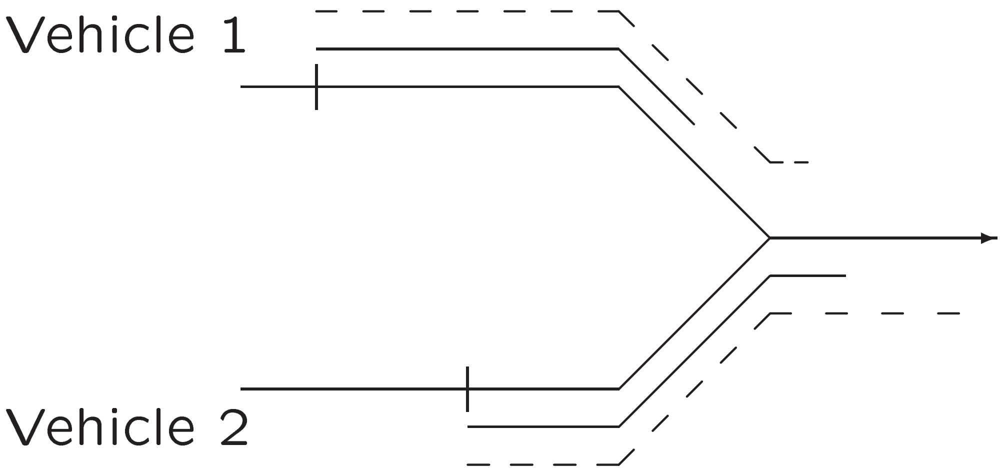
Eff: $\mathit{send}_j := \mathit{null}$

Trajectories:

$w.\mathit{send}_j \equiv \mathit{null}$

Collision Avoidance (Y-Shaped Merge)

Structure: Y-shaped Merge (one protector per vehicle pair)



—— Section of the track "owned" by the vehicle. If overlapping then collision is imminent.

– – – Section of the track "claimed" by the vehicle. If overlapping then collision is possible.

Strategy: Issue emergency brake command when potential for collision exists.

Assumption:
- None of the vehicles exceed the speed limit.
- None of the other vehicles (vehicle pairs) collide into other vehicles (among themselves).

Independent? NO (Composition Theorem III)

Collision Avoidance (Y-Shaped Merge DC)

Actions:

Input: $\texttt{snapshot}(y)_j$, where $y \in Y_{\text{merge-vehicles}}$
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*

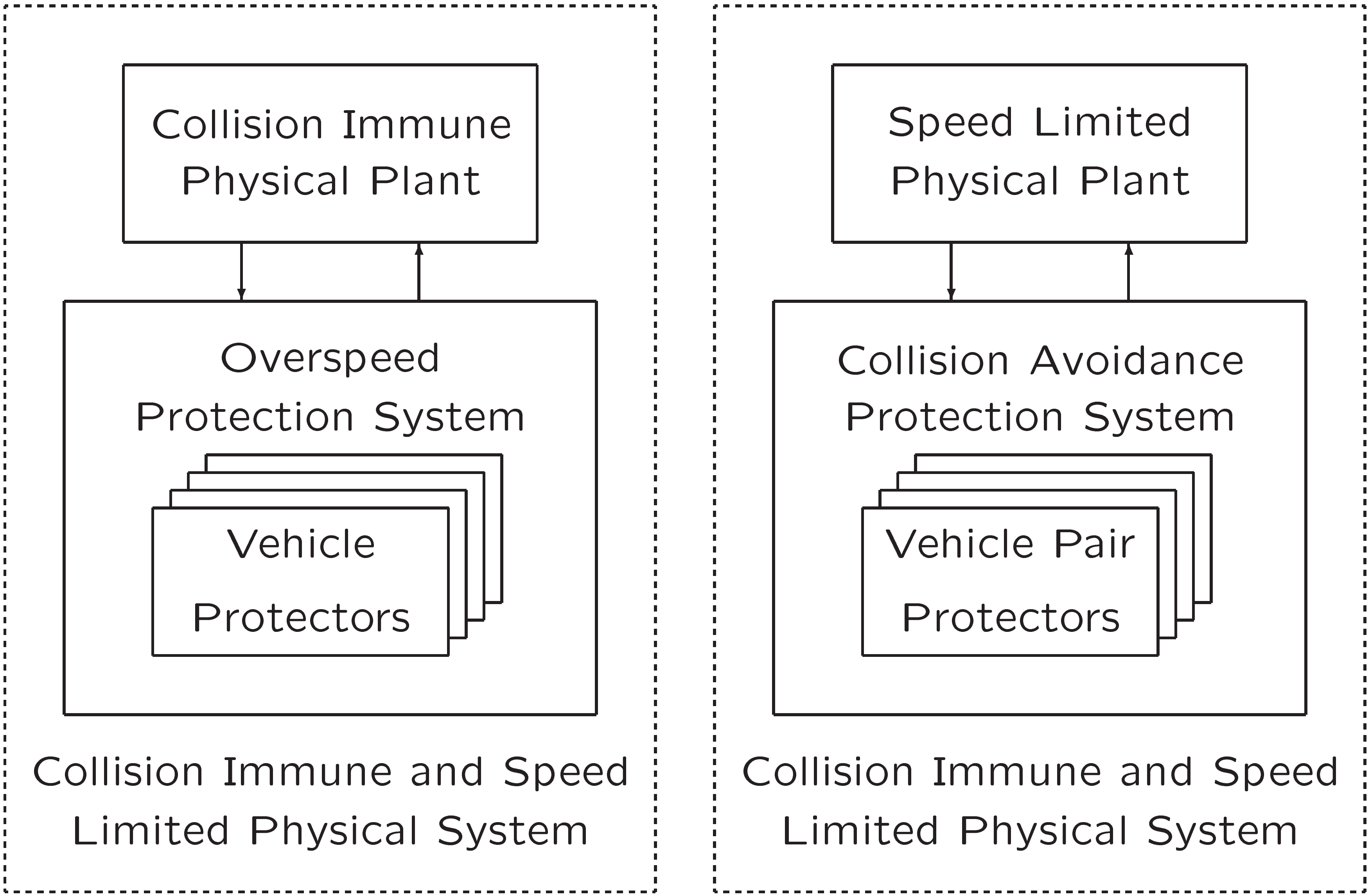
Discrete Transitions:

$\texttt{snapshot}(y)_j$
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$_j := \emptyset$

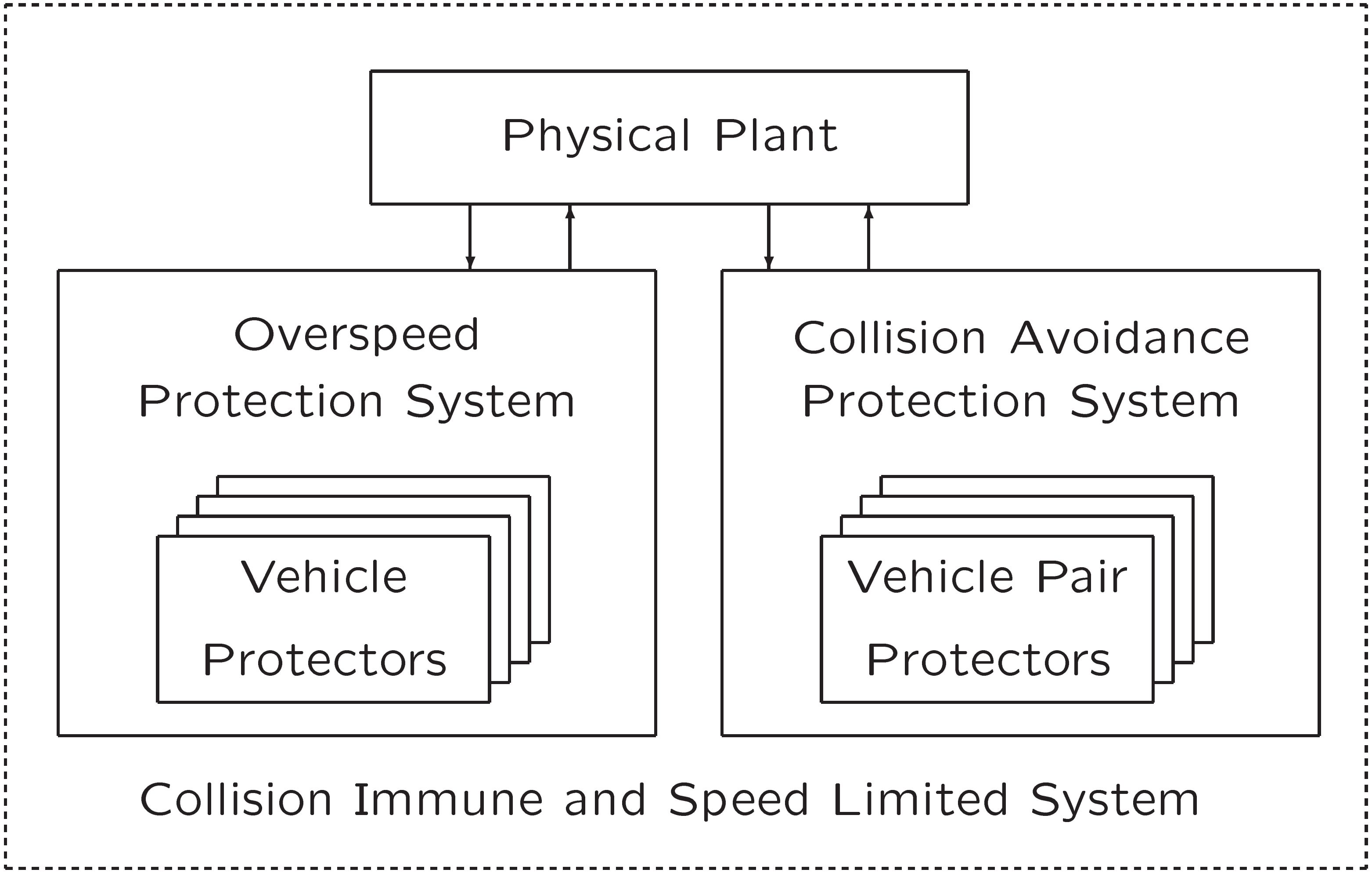
$\texttt{protect}(C)_j$
Pre: *send*$_j = C$
Eff: *send*$_j :=$ *null*

Trajectories:
$w.$*send*$_j \equiv$ *null*

Overspeed and Collision Avoidance Protection Systems



Protection System Composition



Conclusions and Future Work

Contributions

- Defined and Verified Protector Compositional Rules
- Defined and Abstract Protector Model
- Analyzed Overspeed and Collision Avoidance for PRT 2000^{TM}
- Showed Applicability of Formal Methods

Future Work

- Fault-Tolerance (uncertainty/delays/failures)
- Scalability and Tractability
- HIOA Tool Development