Pedestrian Protection Hybrid Automaton Formalization

The hybrid automaton formalized here captures the decision logic governing pedestrian-related interventions. Its structure reflects a separation between discrete locations, encoding the high-level behavioral states of the controller, and a continuous component responsible for maintaining timing information and buffering perception-related quantities. The following sections collect all formal ingredients used by the automaton, including the thresholds that parameterize the guards and invariants, the predicates over buffered observations, and the reset and sensing functions.

TABLE I
AUTOMATON THRESHOLDS

name	semantic	value
n	dimension of the automaton	10
D_FREQ	detection frequency (ms)	100
RT_H	human reaction time (ms)	700
TH_C	threshold for valid detection	0.4
STALE	upper bound for stale detections (ms)	300
TH_TTC_s	threshold for safe TTC (ms)	3500
TH_TTC_r	threshold for risky TTC (ms)	2000
TH_TTC_c	threshold for critical TTC (ms)	1000
S_CONS	% for for safe TTC classification	0.8
SR_CONS	% for safe-risky TTC classification	0.6
RC_CONS	% for risky-critical TTC classification	0.4
C_CONS	% for critical TTC classification	0.2
CONS	% of agreeing frames	0.8

The numerical parameters in Table I regulate how perception evidence is aggregated and how fast stale information is discarded. They are intentionally chosen so that reaction-relevant quantities (e.g., TTC estimates) are evaluated over a short, recent observation window, allowing the controller to remain responsive while filtering short-lived outliers. Based on these parameters, we now introduce the predicates used in the invariants and guards.

Definition 1 (Discrete Locations)**.** The finite set of discrete locations L is defined as follows.

$$L = \{\mathsf{Normal}, \mathsf{Throttle}, \mathsf{SoftBrk}, \mathsf{EmergencyBrk}\},$$

Definition 2 (State Variables)**.** The automaton state variable sequence is defined as follows.

$$X = (C_0, \ldots, C_{n-1}, TTC_0, \ldots, TTC_{n-1}, cs_0, \ldots, cs_{n-1}, s_d, s_c, t)$$

where:

- n: the automaton dimension;
- C_i, with $i = 0, \ldots, n-1$: confidence of the pedestrian detection in the i-th frame;
- TTC_i with $i = 0, \ldots, n-1$: estimated time-to-collision in the i-th frame;
- cs_i with $i = 0, \ldots, n-1$: pedestrian crossing indicator in the i-th frame;
- s_d, s_c: timers for detection staleness, crossing staleness, respectively;
- t: time since the last sensor update.

Definition 3 (Initial State)**.** Being $n = |X|$, i.e., the number of variables in X, for any $v \in \mathbb{R}^n$, Init holds if v is:

- $C_i = 0, cs_i = 0,\ 0 \le i < n$;
- $TTC_i = \mathsf{NO\_TTC},\ 0 \le i < n$;
- $s_d = \mathsf{STALE}, s_c = \mathsf{STALE}$;
- $t = 0$.

NO_TTC and STALE are sentinel values to denote a safe TTC and stale data.

Definition 4 (Continuous evolution)**.** For any $l \in L$, the flow condition $f(l)$ is the same and is defined as:

- $\dot{C}_i = 0,\ \dot{TTC}_i = 0, \dot{cs}_i = 0,\ 0 \le i < n$;
- $\dot{s_d} = 1, \dot{s_c} = 1, \dot{t} = 1$.

The predicates below formalize the conditions under which a sequence of buffered measurements provides sufficiently strong evidence of a detection or a crossing event. Each predicate follows the same structural pattern: it inspects the most recent entries of the buffer and checks for temporal consistency, thereby avoiding spurious mode switches due to isolated sensor noise.

Definition 5 (Detection Predicate)**.** The predicate detected is defined as follows.

$$\mathsf{detected} = \forall i \in [0, \lceil \frac{\mathsf{RT\_H}}{2 \cdot \mathsf{D\_FREQ}} \rceil] \,.\, C_i \ge \mathsf{TH\_C}$$

Definition 6 (Crossing predicate)**.** The predicate crossing is defined as follows.

$$\mathsf{crossing} = \forall i \in [0, \lceil \frac{\mathsf{RT\_H}}{2 \cdot \mathsf{D\_FREQ}} \rceil] \,.\, cs_i = 1$$

Threat classification relies on estimating TTC over the buffered frames. To mitigate the influence of occasional misestimates, each TTC-level predicate uses a consensus requirement over a prefix of the buffer, scaled according to the severity of the corresponding threat class. Lower TTC thresholds naturally require fewer agreeing observations, reflecting the need to react quickly as risk increases.

Definition 7 (Safe distance predicate)**.** The predicate s_TTC is defined as follows.

$$\mathsf{s\_TTC} = \sum_{i=0}^{k} P_i \ge \lceil k \cdot \mathsf{CONS} \rceil$$

where $P_i = 1$ if $TTC_i > \mathsf{TH\_TTC\_s}$, otherwise $P_i = 0$ and $k = \lceil n \cdot \mathsf{S\_CONS} \rceil$.

Definition 8 (Safe-Risky TTC predicate)**.** The predicate sr_TTC is defined as follows.

$$\mathsf{sr\_TTC} = \sum_{i=0}^{k} P_i \geq \lceil k \cdot \mathsf{CONS} \rceil$$

where $P_i = 1$ if $TTC_i \leq \mathsf{TH\_TTC\_s}$, otherwise $P_i = 0$ and $k = \lceil n \cdot \mathsf{SR\_CONS} \rceil$.

Definition 9 (Risky-Critical TTC predicate)**.** The predicate rc_TTC is defined as follows.

$$\mathsf{rc\_TTC} = \sum_{i=0}^{k} P_i \geq \lceil k \cdot \mathsf{CONS} \rceil$$

where $P_i = 1$ if $TTC_i \leq \mathsf{TH\_TTC\_r}$, otherwise $P_i = 0$ and $k = \lceil n \cdot \mathsf{RC\_CONS} \rceil$.

Definition 10 (Critical TTC predicate)**.** The predicate c_TTC is defined as follows.

$$\mathsf{c\_TTC} = \sum_{i=0}^{k} P_i \geq \lceil k \cdot \mathsf{CONS} \rceil$$

where $P_i = 1$ if $TTC_i \leq \mathsf{TH\_TTC\_c}$, otherwise $P_i = 0$ and $k = \lceil n \cdot \mathsf{C\_CONS} \rceil$.

Definition 11 (No TTC consensus)**.** The predicate doubt as defined as follows.

$$\begin{aligned}\mathsf{doubt} = &\neg \mathsf{s\_TTC} \wedge \neg \mathsf{sr\_TTC} \wedge \\ &\neg \mathsf{rc\_TTC} \wedge \neg \mathsf{c\_TTC}\end{aligned}$$

The mode-dependent invariants express admissible continuous states while the automaton resides in a given discrete location. These invariants enforce that the controller remains in a state only as long as the corresponding perception conditions remain compatible with that mode. In particular, they ensure that stale information cannot lock the automaton in a mode indefinitely, and that a new sensor update (captured by the timer t) eventually forces a transition through the sensing edge. For clarity, in the auxiliary predicates used in Definition 12, we omit writing the assignment $X := x$.

Definition 12 (State Invariants)**.** The function $\mathsf{Inv}(l)$ is defined as follows.

$$\begin{aligned}\mathsf{Inv}(\mathsf{Normal}) &= t < \mathsf{D\_FREQ} \wedge \\ &\quad (s_d \geq \mathsf{STALE} \vee s_c \geq \mathsf{STALE} \vee \mathsf{s\_TTC} \vee \mathsf{doubt}) \\ \mathsf{Inv}(\mathsf{Throttle}) &= t < \mathsf{D\_FREQ} \wedge s_c < \mathsf{STALE} \wedge \\ &\quad \big((\mathsf{sr\_TTC} \wedge \neg \mathsf{rc\_TTC} \wedge \neg \mathsf{c\_TTC}) \vee \mathsf{doubt}\big) \\ \mathsf{Inv}(\mathsf{SoftBrk}) &= t < \mathsf{D\_FREQ} \wedge s_c < \mathsf{STALE} \wedge \\ &\quad \big((\mathsf{rc\_TTC} \wedge \neg \mathsf{c\_TTC}) \vee \mathsf{doubt}\big) \\ \mathsf{Inv}(\mathsf{EmergencyBrk}) &= t < \mathsf{D\_FREQ} \wedge s_c < \mathsf{STALE}\end{aligned}$$

The transition structure is fully enumerated by the edges in Definition 13. Each edge corresponds to a possible qualitative change in the controller's behaviour, ranging from maintaining the current mode to escalating the intervention level. The resulting graph is acyclic except for the intentional self-loops, which allow the automaton to accommodate new measurements without generating unnecessary mode changes.

Definition 13 (Edges)**.** $E \subseteq L \times L$ is defined as follows

$$\begin{aligned}e_1 &= (\mathsf{Normal}, \mathsf{Normal}) \\ e_2 &= (\mathsf{Normal}, \mathsf{Throttle}) \\ e_3 &= (\mathsf{Normal}, \mathsf{SoftBrk}) \\ e_4 &= (\mathsf{Normal}, \mathsf{EmergencyBrk}) \\ e_5 &= (\mathsf{Throttle}, \mathsf{Normal}) \\ e_6 &= (\mathsf{Throttle}, \mathsf{Throttle}) \\ e_7 &= (\mathsf{Throttle}, \mathsf{SoftBrk}) \\ e_8 &= (\mathsf{Throttle}, \mathsf{EmergencyBrk}) \\ e_9 &= (\mathsf{SoftBrk}, \mathsf{Normal}) \\ e_{10} &= (\mathsf{SoftBrk}, \mathsf{Throttle}) \\ e_{11} &= (\mathsf{SoftBrk}, \mathsf{SoftBrk}) \\ e_{12} &= (\mathsf{SoftBrk}, \mathsf{EmergencyBrk}) \\ e_{13} &= (\mathsf{EmergencyBrk}, \mathsf{Normal}) \\ e_{14} &= (\mathsf{EmergencyBrk}, \mathsf{EmergencyBrk})\end{aligned}$$

Guard conditions determine when a discrete transition may fire. They combine perception predicates with timing constraints on t, s_d and s_c. Structurally, the guards partition all admissible states in such a way that the automaton remains deterministic whenever a transition is enabled. For clarity, in the auxiliary predicates used in Definition 14, we omit writing the assignment $X := x$.

Definition 14 (Guards)**.** For $e \in E$, we define guard predicates as follows.

$$\begin{aligned}G(e_1) &= t \geq \mathsf{D\_FREQ} \\ G(e_2) &= t < \mathsf{D\_FREQ} \wedge s_c < \mathsf{STALE} \wedge \\ &\quad \mathsf{sr\_TTC} \wedge \neg \mathsf{rc\_TTC} \wedge \neg \mathsf{c\_TTC} \\ G(e_3) &= t < \mathsf{D\_FREQ} \wedge s_c < \mathsf{STALE} \wedge \\ &\quad \mathsf{rc\_TTC} \wedge \neg \mathsf{c\_TTC} \\ G(e_4) &= t < \mathsf{D\_FREQ} \wedge s_c < \mathsf{STALE} \wedge \mathsf{c\_TTC} \\ G(e_5) &= t < \mathsf{D\_FREQ} \wedge \\ &\quad \big(s_d \geq \mathsf{STALE} \vee (s_c < \mathsf{STALE} \wedge \mathsf{s\_TTC})\big) \\ G(e_6) &= G(e_1) \\ G(e_7) &= G(e_3) \\ G(e_8) &= G(e_4) \\ G(e_9) &= G(e_5) \\ G(e_6) &= G(e_1) \\ G(e_7) &= G(e_3) \\ G(e_8) &= G(e_4) \\ G(e_9) &= G(e_5) \\ G(e_{10}) &= G(e_2) \\ G(e_{11}) &= G(e_1) \\ G(e_{12}) &= G(e_5) \\ G(e_{13}) &= t < \mathsf{D\_FREQ} \wedge s_c \geq \mathsf{STALE} \\ G(e_{14}) &= G(e_1)\end{aligned}$$

The predicates update and reset govern how the continuous state is updated upon taking an edge. The former incorporates a fresh sensor measurement and shifts the history buffers, while the latter keeps the existing measurements but selectively resets the staleness counters. Together, they ensure that the automaton cleanly separates the arrival of new data from pure time-driven maintenance steps.

Definition 15 (Sense function)**.** The update predicate, being $C \in [0, 1]$,

$$TTC = \begin{cases} \mathsf{NO\_TTC} & \text{if } C < \mathsf{TH\_C}, \\ val \in \mathbb{R}_{\geq 0} & \text{otherwise}, \end{cases}$$

$$cs = \begin{cases} 0 & \text{if } C < \mathsf{TH\_C}, \\ val \in \{0, 1\} & \text{otherwise}, \end{cases}$$

holds if:

- $C'_0 = C, TTC'_0 = TTC, cs'_0 = cs$;
- $C'_{i+1} = C_i, TTC'i + 1 = TTC_i, cs'_{i+1} = cs_i$, for $0 \leq i < n - 1$;
- $s'_d = s_d, s'_c = s_c, t' = t$.

Definition 16 (Reset timers function)**.** The predicate reset holds if

$$s'_d = \begin{cases} 0 & \text{if detected} \\ s_d & \text{otherwise} \end{cases}$$

$$s'_c = \begin{cases} 0 & \text{if crossing} \\ s_c & \text{otherwise} \end{cases}$$

while all other variables remain unchanged.

Definition 17 (Reset)**.** For any $e \in E$, the function $R(e)$ is defined as follows.

$$R(e) = \begin{cases} \mathsf{update} & \text{if } e \in \{e_1, e_6, e_{11}, e_{14}\} \\ \mathsf{reset} & \text{otherwise} \end{cases}$$

Combining the discrete locations, continuous dynamics, invariants, guards, and reset maps yields a fully specified hybrid automaton that captures the entire control logic of the pedestrian-protection system. This formalisation is suitable both for implementation-level reference and for verification tasks such as reachability analysis or mode-transition correctness proofs.