Control of an Idealized Adaptive Cruise Control System

Gabriele Sarti
Department of Mathematics and Geoscience
University of Trieste, International School for Advanced Studies

1 Introduction

Adaptive cruise control (ACC) is a cyber-physical system used to assist drivers in the longitudinal control of their vehicle during motorway driving. The system controls the accelerator, engine powertrain and vehicle brakes to accomplish two main goals: first and foremost, the vehicle provided with an ACC system (*ego car*) should maintain a desired safe distance δ_{set} with the vehicle ahead (*lead car*); secondly, if a lead car is not present or the relative distance δ between the ego car and the lead car allows it, a target speed ν_{set} should be reached and maintained by the ego car. Figure 1 presents three possible scenarios for the ACC system, including objectives and measurements.

This work will present the layout and discrete dynamics of an idealized ACC system. After a brief presentation of its components including sensors and actuators, the focus will be put on system requirements and on designing the *ACC control module*. I will conclude by presenting the results of an ACC simulation leveraging *model predictive control* (MPC) and discussing some perspectives in the extending the use of formal verification in RL for safety-critical settings such as ACC.

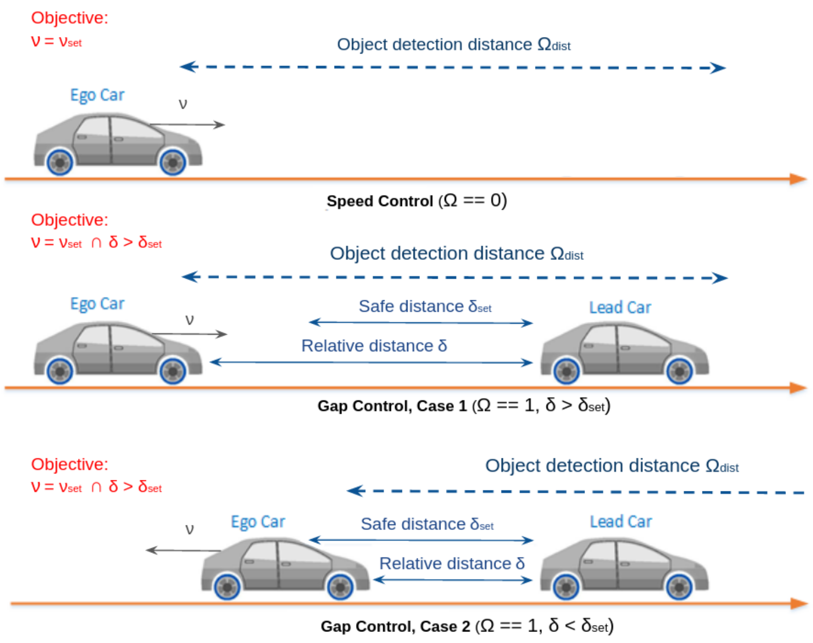

Figure 1: Visual representation of the main components and sensed information from *speed and gap control* modes of an adaptive cruise control (ACC) system. Adapted from MathWorks illustration.

2 Idealized ACC System Configuration

This section describes the components of a plant for an idealized ACC system, as visualized by Figure 2. The idealized system is designed to control the acceleration $a(t)$ of an simulated mass-less vehicle driving in a two-dimensional settings.

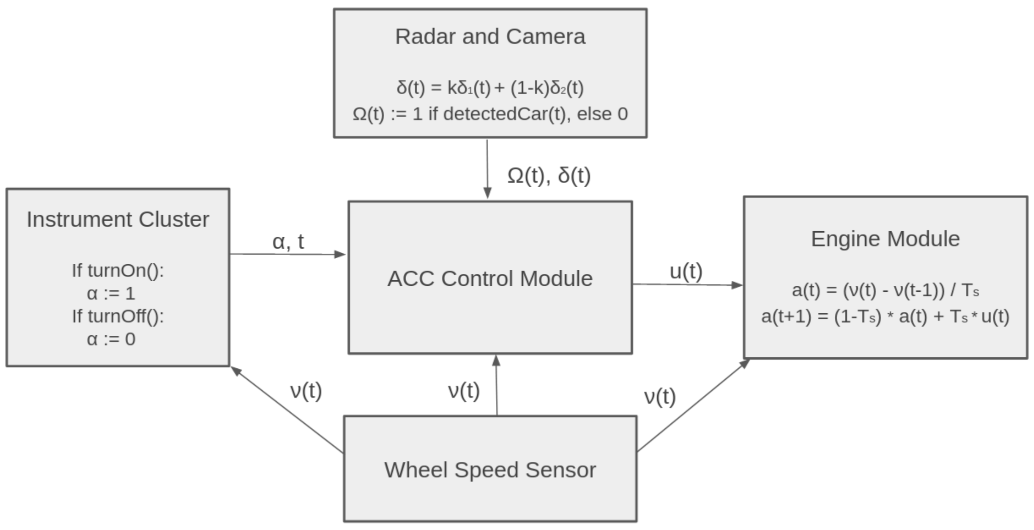

Figure 2: The idealized physical configuration of an ACC-centric vehicle plant, inspired by Group et al. (2005) layout of Figure 5. Arrows represent information flows throughout the system.

2.1 Radar and Camera

The radar and camera components include sensors able to estimate the actual distance δ separating the ego car from the lead car. In particular, if we describe the measurements of the radar as $\delta_1(t) = \hat{\delta}(t) + \epsilon_1$ and the ones of the camera as $\delta_2(t) = \hat{\delta}(t) + \epsilon_2$ (with $\epsilon_i \sim N(\mu_i, \sigma_i^2)$, $i \in \{1, 2\}$ representing respectively radar and camera noise), the final estimate for $\delta(t)$ passed to the ACC control module is the product of weighted average sensor fusion:

$$\delta(t) = k\delta_1(t) + (1-k)\delta_2(t), \text{ with } k = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}$$

Moreover, the radar informs the ACC control module about the presence of a lead car by means of the boolean variable $\Omega(t)$. This detection is performed over a distance range $\Omega_{dist}(t)$ (see Figure 1) and is used as an indicator for mode switching in the ACC controller.

2.2 Instrument Cluster

The instrument cluster includes the global clock providing the ACC module with discrete timesteps t with sampling period T_s. Timesteps are used for synchronous execution of all components in the system, assuming negligible sensing and actuation times. The switching on/off of the ACC system performed by the driver is communicated to the ACC module by means of an event-triggered boolean variable α. At each timestep, the cluster is informed by the wheel speed sensors about current speed $\nu(t)$ and presents the information to the driver. While in real-life settings we would presume that safe distance δ_{set} and target speed ν_{set} are both manually adjustable by drivers through the instrument cluster, here we assume they are fixed internal variables of the system.

2.3 Engine Module

The primary function of the engine module is to receive the desired acceleration from the ACC module and set the vehicle's acceleration based on this information. First, the module estimates the current acceleration $a(t)$ from current and previous velocity $\nu(t), \nu(t-1)$ and using discrete differential formula (Guo et al., 2020). Then, the current acceleration estimate is mixed with desired acceleration $u(t)$ to set the acceleration for next timestep using a discrete kinematics equation.

$$a(t) = \frac{\nu(t) - \nu(t-1)}{T_s} \qquad a(t+1) = (1 - T_s)a(t) + T_s u(t)$$

2.4 Wheel Speed Sensor

The wheel speed sensor is part of the brake control module (excluded here for simplicity), and its purpose is to estimate the speed ν at each timestep t from rotation counts and wheel radius. The sensor provides the ACC control module, the engine module and the instrument cluster with speed estimates to control the system acceleration and inform drivers. In a realistic ACC system, the engine would communicate sharp decreases in acceleration to activate braking. Here, we assume that acceleration can entirely be acted upon optimally by the engine module alone.

3 System Requirements

The following definitions are set, using ε, ζ as error margins to avoid chattering, to define situations respectively below and above the speed/distance thresholds for ACC:

$$\textit{thresh}_B(t) \equiv (\nu(t) < \nu_{set} - \varepsilon \wedge \delta_{pred}(t) > \delta_{set} + \zeta)$$
$$\textit{thresh}_A(t) \equiv (\nu(t) > \nu_{set} + \varepsilon \vee \delta_{pred}(t) < \delta_{set} - \zeta)$$

I now proceed to define the following requirements using STL:

- If the ACC is on, the desired acceleration $u(t)$ generated by the ACC control module must be positive when the system is below thresholds, and negative when it is above to adjust the driving behavior to current conditions:

  $$\varphi_1 \equiv \mathcal{G}_{[0,inf]}(\alpha \wedge \textit{thresh}_B(t) \Rightarrow u(t) > 0) \qquad \varphi_2 \equiv \mathcal{G}_{[0,inf]}(\alpha \wedge \textit{thresh}_A(t) \Rightarrow u(t) < 0)$$

- The acceleration produced must stay in the interval $[a_{min}, a_{max}]$ to ensure a comfortable driving experience for users. If the ACC is on and a lead car is detected, both threshold conditions will be respected at all times:

  $$\varphi_3 \equiv \mathcal{G}_{[0,inf]}(a(t) \in [a_{min}, a_{max}]) \qquad \varphi_4 \equiv \mathcal{G}_{[0,inf]}(\alpha \wedge \Omega(t) \Rightarrow \neg \textit{thresh}_A)$$

- If the ACC is on and a lead car is not detected, the set speed should be respected and will eventually be reached and held until a lead car is detected.

  $$\varphi_5 \equiv \mathcal{G}_{[0,inf]}(\alpha \wedge \neg\Omega(t) \Rightarrow \nu(t) \leq \nu_{set} + \varepsilon \wedge \mathcal{FG}(\nu \in [\nu_{set} - \varepsilon, \nu_{set} + \varepsilon])\mathcal{U}\Omega(t))$$

4 ACC Control Module Design

We can model our simple ACC control module as an extended state machine receiving speed and distance estimates (ν, δ) and object detection information Ω at each timestep t, and eventually receiving the event-triggered α variable when the system is turned on/off.

The ACC control module comprises three possible modes:

- **Off mode:** The system does not produce any desired acceleration output $u(t)$. Acceleration is entirely controlled by the driver through the accelerator and brake pedals.
- **Speed Control mode:** When the system is turned on but no lead car is detected ($\Omega(t) = 0$), the desired acceleration is estimated as a function of the distance between current speed ν and target speed ν_{set}:
  $$u(t) = K_\nu(\nu(t) - \nu_{set})$$
- **Gap Control mode:** When the system is turned on and a lead car is detected ($\Omega(t) = 1$), the desired acceleration is a function of both the distance between current and target speeds $\nu(t) - \nu_{set}$ and the distance between predicted and safe distance $(\delta_{pred}(t) - \delta_{set})$. The value

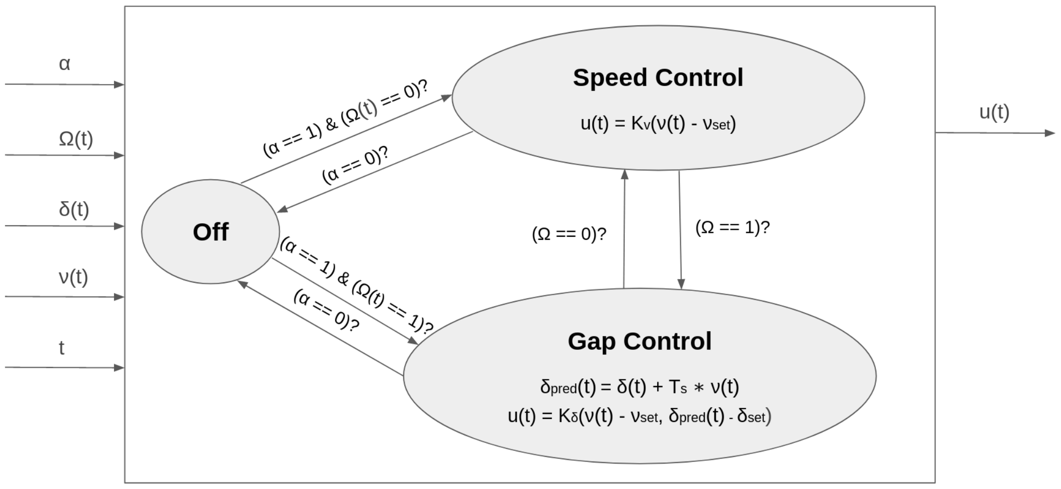

Figure 3: Model for the ACC control module. The module produces a desired acceleration $u(t)$ at each timestep t, given input velocity ν, distance δ and mode transition parameters α and Ω.

of $\delta_{pred}(t)$ is computed by the controller at each timestep from the currently estimated distance and speed $\delta(t), \nu(t)$:

$$\delta_{pred}(t) = \delta(t) + T_s \times \nu(t)$$
$$u(t) = K_\delta(\nu(t) - \nu_{set},\, \delta_{pred}(t) - \delta_{set})$$

K_ν, K_δ are functions whose parameters can either be set by the designer or learned by optimizing the robustness value over system requirements (see Section 5). While here a proportional controller is used, the predictive behavior of the PD component inside a digital PID controller would allow to satisfy the requirements φ_4, φ_5 and prevent overshooting.

5 Verification and Experiments

The system behavior depends on the current mode of the controller defined by signals $\alpha, \Omega(t)$ and by sensed inputs $\nu(t), \delta(t)$. Given that the 2D ACC settings are easily simulated, a possibility would be to use the domain estimation algorithm to verify the validity of specifications by iteratively sampling trajectories from a Gaussian process with increasing probability of belonging to a set B of trajectories where the robustness has negative value.

It will be especially important to test extreme settings (e.g. activating ACC when $\nu(t)$ or $\delta(t)$ are below safety thresholds) to ensure a proper behavior of the control module in those situations. Another important aspect is to ensure smooth transitions between modes and states: Figure 4 provides an example of the behavior of an simulated ACC systems with model predictive control (MPC) based on the Simulink framework. We can see how, when presented with a leading vehicle moving with sinusoidal acceleration, the ego car is able to enforce both the safe distance δ_{set} and the target speed ν_{set}, constraints at all times. In the case of MPC the control module needs to keep a running estimate of the relative speed ν_{rel} between the two vehicles and treat the front vehicle acceleration as a noise factor when modeling the dynamical system, as suggested by Guo et al. (2020).

The parameter synthesis step is also an important one in the context of our idealized ACC system, provided that the control output is influenced by the output of K_ν, K_δ which depends on parametrization. To ensure a smooth acceleration we can squeeze it between user-defined bounds by passing $a(t)$ through a logistic function inside the engine module:

$$a^*(t) = \frac{|a_{max}| + |a_{min}|}{1 - e^{-ka(t)}}$$

In this case, an additional parameter k could be estimated to control the steepness of the curve.

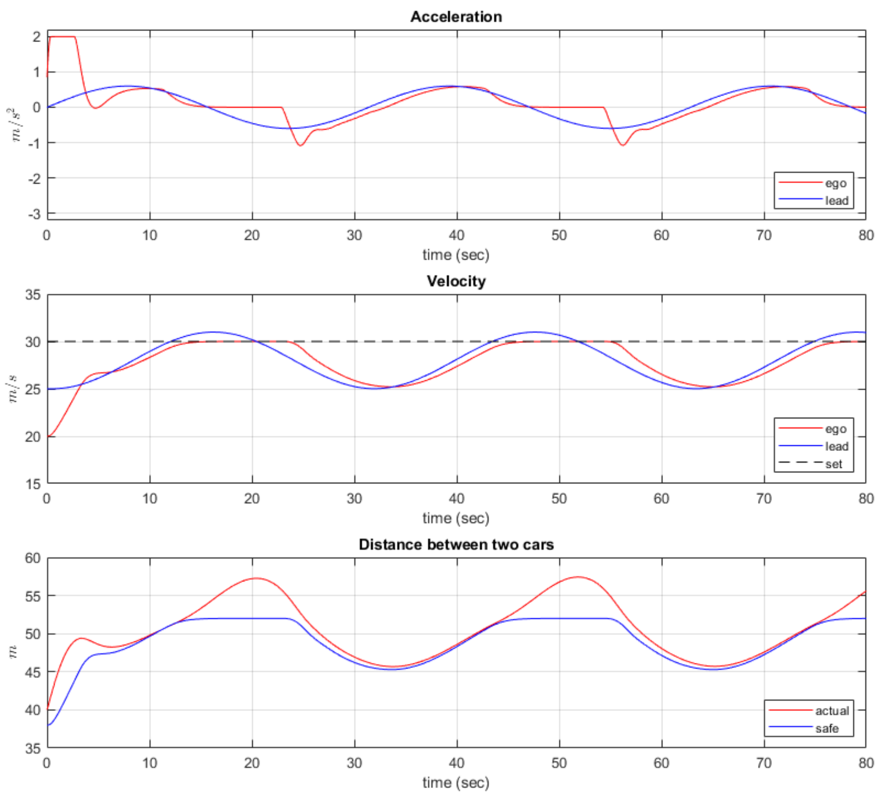

Figure 4: Simulation of ego car's behavior in gap control mode in the case of sinusoidal lead car acceleration. Results taken from MathWorks ACC tutorials.

6 Conclusion

In this work, I presented a simple ACC system acting over discrete time steps t to vary the acceleration of a 2-dimensional mass-less vehicle. The fundamental ACC control module found inside the system was described in detail to highlight switching state transitions and how desired acceleration outputs $u(t)$ are produced. System requirements were presented, along with suggestions for verification strategies and an experimental result taken from MathWorks ACC tutorials.

In Appendix B, I provide an overview of the Verifiably Safe Reinforcement Learning paradigm(Hunt et al., 2020) which performs a synthesis over the control theory and RL perspectives which is especially interesting in the context of safety-critical systems like ACC.

References

US Software System Safety Working Group et al. Adaptive cruise control system overview. In *5th Meeting of the US Software System Safety Working Group*, 2005.

Lie Guo, Pingshu Ge, Dachuan Sun, and Yanfu Qiao. Adaptive cruise control based on model predictive control with constraints softening. *Applied Sciences*, 10(5):1635, 2020.

Nathan Hunt, Nathan Fulton, Sara Magliacane, Nghia Hoang, Subhro Das, and Armando Solar-Lezama. Verifiably safe exploration for end-to-end reinforcement learning, 2020.

MathWorks. Adaptive cruise control system. URL `https://www.mathworks.com/help/mpc/ref/adaptivecruisecontrolsystem.html`.

A ACC SYSTEM PHYSICAL LAYOUT

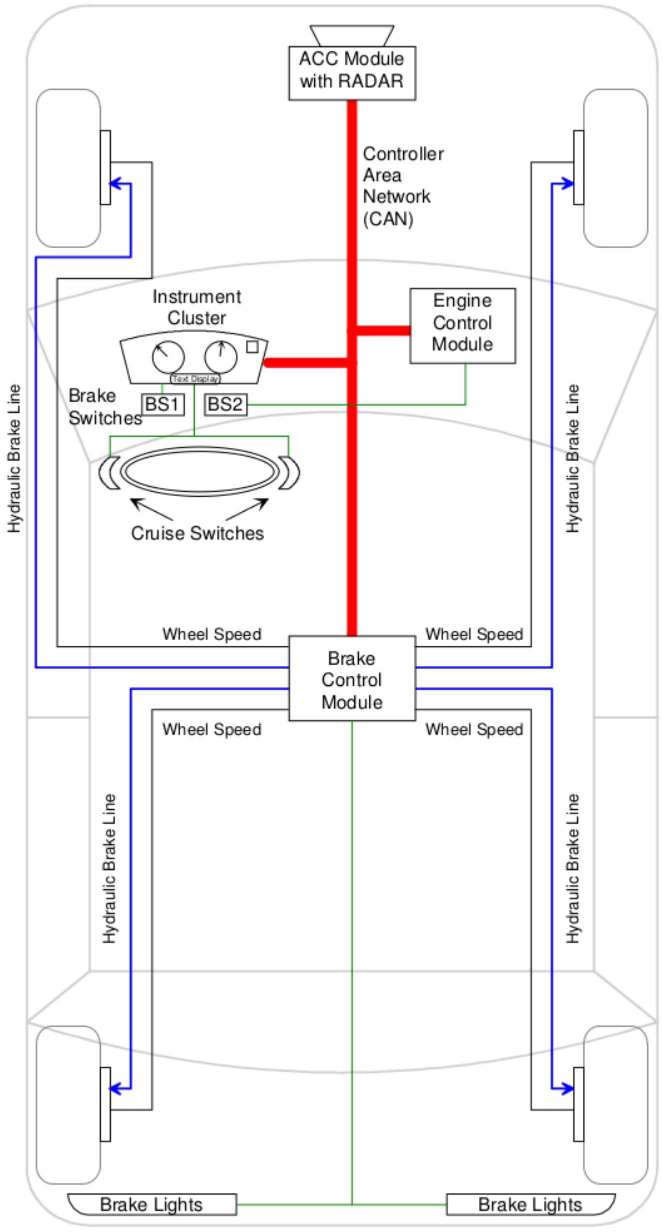

Figure 5: A detailed vehicle physical layout with highlighted ACC modules from Group et al. (2005). The layout presented in Figure 2 provides an idealized abstraction of this setup.

B ACC and Verifiably Safe Reinforcement Learning

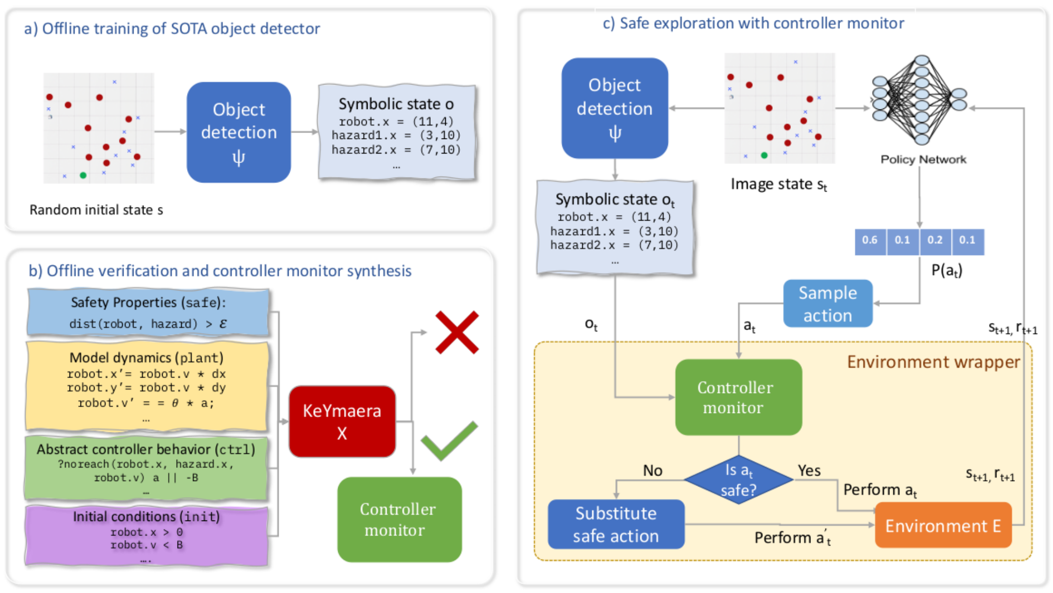

Figure 6: Visual representation of the VSRL approach by Hunt et al. (2020). VSRL can be used in safety-critical settings such as ACC to drastically reduce policy network violations through the enforcement of safety constraints produced by synthesized controller monitors.

With the rise of cheap graphical processors and machine learning frameworks, deep reinforcement learning (RL) algorithms have proven to be very effective in learning control policies aimed at maximizing some reward function in partially-observable scenarios (i.e. from raw sensed data). However, the process needed to learn and verify those policies requires disproportionate amounts of training examples, and can't be guaranteed to incur in safety constraints violations. This fact limits the application of such algorithms to real-life safety-critical settings such as ACC, where strong safety requirements represent a minimal baseline for deployment.

The VSRL approach (Hunt et al., 2020) builds upon the standard deep reinforcement learning methodology to ensure that constraints derived from formal verification are respected at all times. Figure 6 presents the steps required by the VSRL paradigm:

- An object detection system ψ is first trained to extract the position of safety-relevant object into a symbolic state o_t, without relying on internal simulator states to match real-life conditions.
- A controller monitor C_m is synthesized from model dynamics, initial conditions and abstract behaviors of the controller to ensure that safety properties are respected.
- The two components are added to a standard reinforcement learning exploratory procedure: action a_t sampled by a policy network over the current state are passed alongside the symbolic state o_t produced by ψ to C_m, which validates the safety of a_t and possibly replace it with an action a'_t which is deemed as safe by the monitor.

Hunt et al. (2020) show in Table 1 how the VSRL approach leads to significantly less safety constraints violations with respect to a standard deep RL approach in the context of an ACC 2-dimensional simulation very similar to those described in this work. The perspective of using inferred symbolic states to ensure safe exploration through formal verification promotes a synthesis between traditional control and modern reinforcement learning perspectives that may extend the application of deep reinforcement learning to safety-critical scenarios.