CPS-Guard：基于 AI 和 LLM 的网络物理系统可靠性保障框架

1st Trisanth Srinivasan
Department of AI Deployment and Safety
Cyrion Labs
Dallas, USA
trisanth@cyrionlabs.org

1st Santosh Patapati
Department of AI Deployment and Safety
Cyrion Labs
Dallas, USA
santosh@cyrionlabs.org

2nd Himani Musku
School of Computer Science
Carnegie Mellon University
Pittsburgh, USA
hmusku@andrew.cmu.edu

3rd Idhant Gode
College of Engineering
Cornell University
Ithaca, USA
iag32@cornell.edu

4th Aditya Arora
ACM AI
University of California San Diego
San Diego, USA
a7arora@ucsd.edu

5th Samvit Bhattacharya
Department of R & D
Cyrion Labs
San Roman, USA
samvitb@cyrionlabs.org

6th Abubakr Nazriev
Sentinel DE
University of Montana
Missoula, USA
abu.nazriev@umconnect.umt.edu

7th Sanika Hirave
Dept. of Computer Science & Engineering
Oakland University
Rochester, USA
sanikahirave@oakland.edu

8th Zaryab Kanjiani
School of Applied Economics and Management
Cornell University
Ithaca, USA
zk226@cornell.edu

9th Srinjoy Ghose
School of Science
University of North Texas
Denton, USA
srinjoyghose@my.unt.edu

10th Srinidhi Shetty
University of Texas at Dallas
Dallas, USA
sshetty@cyrionlabs.org

***Abstract*—网络-物理系统（CPS）越来越依赖先进的 AI 技术在关键应用中运行。然而，传统的验证和确认方法通常难以处理 AI 组件的不可预测和动态特性。在本文中，我们介绍了 CPS-Guard ，这是一种采用多角色编排来自动化 AI 驱动的 CPS 的迭代保证过程的新框架。通过在模拟环境中为专用代理分配专业角色（例如，安全监控、安全评估、故障注入和恢复计划），CPS-Guard 持续评估和改进 AI 行为，以满足一系列可靠性要求。我们通过一个涉及 AI 驱动的自主车辆在交叉路口导航的案例研究展示了该框架。我们的结果表明，CPS-Guard 有效地检测漏洞，管理性能影响，并支持自适应恢复策略，从而为安全和安全关键系统的严格 V & V 提供了一个结构化和可扩展的解决方案。**

***Index Terms*—Cyber-Physical Systems (CPS), Verification and Validation (V & V), Artificial Intelligence (AI), Safety-Critical Systems, Large Language Models (LLM), Autonomous Driving**

I. 介绍

网络物理系统（CPS）将计算算法与物理过程相结合。CPS 正在被应用于多个领域，如交通、能源、制造、医疗保健和农业。确保这些系统的安全性、可靠性、可信性和及时性至关重要，因为故障可能导致严重后果。因此，验证和验证（V & V）对于建立对 CPS 满足其要求并按预期运行的信任至关重要。V & V 在大多数领域仍然是一项复杂且代价高昂的工作 [1] 。

当人工智能（AI）被整合到 CPS 中时，V & V 的挑战加剧。这一难题随着越来越复杂的 AI 技术的应用而加剧，例如在这些系统中部署的深度神经网络（DNNs）和大型语言模型（LLMs）。AI 的行为往往对环境中意外变化 [2] 很敏感。这种不可预测性，加上对敌对攻击 [3] 的脆弱性，使得传统的 V & V 方法——依赖可预测行为和详尽状态探索 [4] 的方法——在评估基于 AI 的组件时效果不佳。

解决这些问题需要专门为基于 AI 的 CPS 设计的 V & V 框架。这样的框架应能在隔离状态下验证 AI 模型，以及它们在运行时与环境和其他系统组件的交互方式。我们需要系统化的结构化方法来：

- 根据多样化的可靠性标准评估 AI 在情境中的表现
- 评估对噪声、故障和安全威胁的鲁棒性
- 促进 AI、物理系统及其环境之间相互作用的分析
- 支持迭代优化
- 提供可追溯的证据，以适合建立保证案例并支持认证 [5]

为满足这些需求，我们提出了 CPS-Guard ，一个以多角色编排为中心的新框架，以迭代方式确保 CPS 中 AI 组件的可靠性。CPS-Guard 为各种代理（AI 模型、算法、形式检查器等）分配角色，以在模拟的 CPS 环境中协同工作。代理根据定义的可靠性要求不断评估、挑战和优化主要 AI 组件的性能。

所提出框架的贡献如下：

1) 多角色 V & V 架构：定义了专门的角色（例如，“生成器”、“安全监控器”、“安全评估者”、“性能预言机”、“故障注入器”、“恢复规划器”），以适应全面的可靠性评估。
2) 迭代保证循环：实现一个受控的反馈循环，其中来自监测/评估角色的 V & V 发现为后续行动、测试生成或适应性规划提供信息。
3) 模拟集成 & 状态管理：提供与标准 CPS 模拟器连接的接口，并管理上下文 V & V 所需的共享状态信息。
4) 可扩展性：设计为模块化，允许用户使用不同的 AI 模型、正式技术或过程逻辑来定义新的角色或定制现有角色。
5) 关注可靠性的指标：收集与安全违规、检测到的安全漏洞、性能遵循性以及恢复效果专门相关的指标。

通过协调这些角色，CPS-Guard 旨在为 CPS 中的 AI 提供比临时测试或孤立组件分析更为严格的 V & V 方法。

II. 相关工作

基于人工智能的 CPS 的保障借鉴了多个领域的研究。

A. CPS 的验证和确认

传统的 CPS 验证方法基于形式化方法，如模型检测和定理证明 [6] 、基于仿真的测试 [7] 、硬件在环验证、运行时验证 [8], [9] 以及故障或攻击注入 [10] 。这些方法构成了 CPS 保障的骨干。然而，在应用于复杂的 AI 组件时，它们面临着挑战，因为存在可扩展性的问题、测试覆盖不足以及难以建模 AI 不可预测行为的困难 [2] 。

B. AI/ML 系统的验证和确认

关于 AI & V 的研究主要集中在对扰动和对抗样本 [11], [12] 进行稳健性测试，以及使用 SMT 求解器或抽象解释 [13], [14] 等技术进行网络属性的形式验证。同时也有努力提高可解释性 [15], [16] ，并评估机器学习系统中的公平性 [17] 。基于模拟的测试生成也常用于补充离线分析 [18], [19] 。然而，这些方法往往针对特定的单个属性，不能捕捉动态 CPS 环境中的运行时交互的全部复杂性，例如连锁反应事件 [20] 。

C. 运行时保障与监控

运行时保证 (RA) 技术旨在通过使用监控器和安全控制器来确保操作期间的安全性 [21], [22] 。例如，Simplex 架构 [21] 根据需要在复杂的主控制器和经过验证的简单安全控制器之间切换。运行时验证方法使用诸如 LTL、MTL 或 STL 等时态逻辑根据形式化规格评估执行轨迹 [23], [24] 。与这些方法相比，CPS-Guard 在闭环保证过程中利用多个协调角色。这不仅仅是切换到备用控制器。

D. 用于模拟和验证的多智能体系统 (MAS)

多智能体系统已被用于模拟复杂系统，并在 CPS [25] 中测试控制策略。基于智能体的建模已被用于模拟 CPS 行为，重点是自动驾驶场景。这些方法仅仅利用智能体交互进行模拟。另一方面，CPS-Guard 分配了特定的 V & V 角色，其目标是确保可靠性，而不是用于通用模拟或涌现行为分析。

E. LLM 和 AI 编排框架

最近的框架，如 LangChain [26] 和 LlamaIndex [27] ，展示了如何将 LLM 调用与外部工具和数据结合来构建应用程序。AutoGen [28] 进一步支持涉及多个 LLM 代理的复杂工作流程。尽管这些编排框架表现出相当大的能力，它们主要是为应用程序开发和一般的 AI 协作而设计的。与 CPS-Guard 的专门设计相比，它们在形式需求检查、CPS 模拟和结构化保障循环方面的能力有限。这常常迫使研究人员从头开发定制的保障循环和 V & V 步骤，这非常耗时。早期的工作，如 LLMOrchestrator [29] ，提供了一个基本的生成器-验证器结构，CPS-Guard 通过引入多个 V & V 角色显著扩展了这一结构。

F. 定位 CPS-Guard

CPS-Guard 建立在前述领域的理念基础上，同时确立了新的定位。它采用了来自 AI 框架的协调范式，专门用于可靠性 V & V。该框架采用多角色架构，灵感来自多智能体系统。它分配的角色侧重于安全监控、安全性测试、性能评估、故障注入和恢复计划。与仅依赖运行时监控和简单控制器切换的方法不同，CPS-Guard 实现了一个迭代的闭环过程，该过程与 CPS 仿真环境紧密结合，是解决 AI 驱动的 CPS 中 V & V 复杂挑战的强大方案。

III. CPS-Guard 框架

CPS-Guard 是一个基于 Python 的框架，旨在为模拟 CPS 环境中的 AI 组件构建和自动化迭代 V & V 过程。它采用多角色编排模式，其中称为角色的专用计算代理协同工作，以根据可靠性要求评估主要 AI 组件被测试对象（AUT）的行为。

A. 核心架构

CPS-Guard 架构，如图 1 所示，围绕一个编排控制器展开。该控制器管理不同角色之间的交互，连接到 CPS 模拟器的环境接口，维护共享状态的状态管理器，以及可靠性指标追踪器。

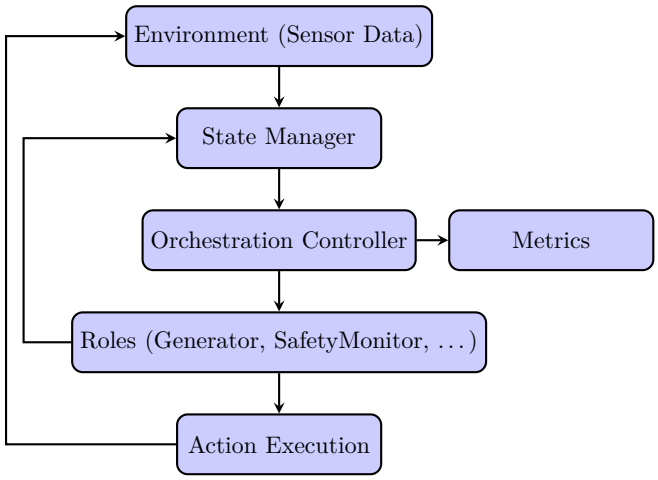

Fig. 1. CPS-Guard 架构概述。来自 CPS 的传感器数据由环境接口收集，并由状态管理器组织。编排控制器协调专门的角色，这些角色通过专用的动作执行模块生成和优化动作。这些动作被反馈到系统中以完成一个闭环的保证过程，同时指标被追踪以进行持续的验证和确认。

B. 关键组件

CPS-Guard 由五个关键组件组成。

1) 编排控制器： 这个核心组件负责管理整个执行流程。它初始化角色，管理迭代 V & V 循环，根据依赖性或触发条件排序角色执行，促进角色之间的通信（通过状态管理器），并根据预定义标准终止过程（例如，迭代次数、测试完成、检测到违规）。

2) 角色： 角色代表了 V & V 过程中的一个特殊功能。它是一个定义标准接口的抽象基类。用户可以实现或配置具体的角色子类。角色通过状态管理器间接交互。关键且可扩展的预定义角色包括：

- 生成器：表示正在测试的主要 AI 组件或为其生成输入/场景的组件。获取当前状态/上下文，生成动作、计划或输出。
- SafetyMonitor：根据安全规则或不变量检查状态、建议的行动或预测的结果。可以使用基于规则的逻辑、形式规范（例如，通过集成监视器如 RTAMT [30] 的 STL 检查）或甚至另一个为安全评估训练的 AI 模型。返回安全判决（例如，安全、不安全、警告）以及潜在的量化评分。
- SecurityAssessor：评估系统的安全态势。可以根据当前状态或 AI 输出分析潜在的漏洞，或者指导 FaultInjector。这可能涉及检查已知攻击模式或安全策略。
- PerformanceOracle：监控性能指标是否符合要求（例如，响应时间、资源使用情况、控制精度、任务完成指标）。
- FaultInjector：根据指令（例如，来自 SecurityAssessor 或预定义的测试计划）引入故障或干扰到模拟中。可以模拟传感器噪声/故障、通信延迟/丢失、GPS 欺骗或对 AI 输入的对抗性扰动。
- RecoveryPlanner：在检测到安全/安全性违规或关键故障时激活。提出恢复措施或适应方案（例如，切换到安全模式，重新规划轨迹，触发警报）。可以基于规则或使用规划算法/人工智能。

3) 环境接口： 提供了一个抽象层，用于与外部 CPS 模拟器（例如，CARLA, AirSim, Gazebo [31]–[33] ）进行通信。它处理：

- 将命令/动作（来自生成器或恢复计划器）发送到模拟器。
- 从模拟器接收传感器数据和状态更新。
- 在模拟器的数据格式和内部状态表示之间进行转换。
- 可能控制模拟时间步长或场景加载。

4) 状态管理器： 维护所有角色可以访问的共享状态。这包括：

- 从环境接口接收的当前状态（例如，车辆位置、传感器读数）。
- 当前迭代中各角色产生的输出（例如，Generator 的建议行动，SafetyMonitor 的裁定，FaultInjector 的主动故障）。
- 如果需要进行时间分析，可以使用历史状态信息。

确保在一个迭代中所有角色都能一致地查看系统状态。

5) 可靠性度量： 收集和记录整个协调过程中关键指标，例如：

- 检测到的安全/保密违规的数量和类型。
- 随时间变化的性能指标值。
- 来自监测器的鲁棒性得分（如适用）。
- 故障注入成功/影响。
- 恢复行动成功率。
- 每个角色/迭代的处理时间。

这些数据对于事后分析和生成保证报告至关重要。

C. 迭代编排工作流程

可以根据需要定制。一个典型的执行周期如下：

1) 初始化：控制器加载配置，初始化角色，通过环境接口连接到模拟器，并通过状态管理器获取初始状态。
2) 迭代开始：控制器根据顺序或依赖关系触发角色。
3) 状态更新：环境接口向状态管理器提供当前的世界状态。
4) 生成或动作提案：生成器（AUT）根据当前状态提出一个动作。
5) 可靠性评估：
   - 安全监控器根据安全规则评估所提议的动作/状态。
   - SecurityAssessor 评估安全态势；可能会指导 FaultInjector。
   - PerformanceOracle 检查性能指标。
   - FaultInjector 可能会根据指令或测试计划引入故障/攻击。
6) V & V 反馈处理：控制器通过 StateManager 收集来自评估角色的裁决/输出。
7) 决策与适应：
   - 如果检测到违规：控制器可能会停止、记录详细信息或激活 RecoveryPlanner。RecoveryPlanner 提出替代措施。
   - 如果没有违规：控制器批准发电机的操作（或更改后的操作）。
8) 动作执行：控制器通过环境接口将最终批准或恢复的动作发送到模拟器。
9) 指标记录：通过 DependabilityMetrics 记录相关数据用于迭代。
10) 循环/终止：控制器检查终止条件（例如，时间限制、场景结束、严重故障）。如果不满足，则继续下一次迭代（步骤 2）。

这个循环允许连续评估和适应，在模拟中形成一个迭代保证过程。角色的配置、它们的交互逻辑（依赖和触发）以及与模拟环境的连接定义了使用该框架进行的特定 V & V 实验。

D. 可扩展性

CPS-Guard 被设计为具有可扩展性。用户可以：

- 使用自定义 Python 代码实现新的角色子类，集成不同的 AI 模型（LLMs、DNNs）、形式验证工具（通过包装）或标准算法。
- 定义角色之间的复杂交互协议和触发条件。
- 开发新的 EnvironmentInterface 子类，以支持不同的模拟器或硬件在环设置。
- 自定义 DependabilityMetrics 集合。

这允许根据特定的 CPS 领域、AI 组件和 V & V 需求定制该框架。

Fig. 2. 一个 CARLA 3D 场景示例。使用第三人称视角和传感器读数作为我们 Llama 3.2 11B 模型的输入。[31], [34], [35]

IV. 用例：自动驾驶车辆

为了展示 CPS-Guard 的能力，我们将其应用于一个具有挑战性的 VMATHXAD V 场景：确保配备基于 AI 规划模块的自动驾驶车辆（AV）在未设信号灯的城市交叉路口安全和可靠地导航。

A. 场景描述

自动驾驶车辆必须导航穿过一个潜在与其他车辆（模拟背景交通）和行人共享的四向交叉口。主要的人工智能测试组件是一个基于 LLM 的战术规划器。根据传感器输入（对象列表、来自模拟感知的位置信息和速度）和一个高层次的目标（例如，“直行”），LLM 规划器生成机动决策（例如，“等待”、“加速”、“礼让”、“谨慎前进”）。关键的可靠性要求包括：

- 安全：避免与其他车辆和行人发生碰撞。保持安全的跟车距离。隐式遵守交通规则（例如，优先通行权，尽管没有明确编程）。
- 安全性：能够抵御伪造的传感器数据（例如，虚假的障碍物、错误的轨迹）带来的危害性行为或交通阻塞。
- 性能：在不造成过度延误（避免过度保守）或舒适性违反（冲击/加速度）的情况下通过交叉路口。

1) 基于 LLM 的规划器的原理： 现有的自动驾驶规划系统使用特定领域的规则集。虽然这有效，但它限制了我们在各种人工智能用例中对 CPS-Guard 进行压力测试的能力。因此，我们故意使用基于 LLM 的规划器，该规划器在这方面相对较弱，以便 1）展示该框架可以包装任何黑箱决策模块，2）揭示传统框架无法发现的失败模式，以及 3）展示当人工智能的内部逻辑不透明时，DURA-CPS 的不同角色如何相互作用。

B. CPS-Guard 配置

我们为此场景配置了 CPS-Guard ，并实例化了以下角色（图 3 ）：

- 生成器（AUT）：一个大型语言模型（经过微调的 Llama3.2 11B 变体 [34] ），接收到当前感知的世界状态和目标后，输出一个战术机动决策。该大型语言模型提供了少量示例和一个思维链（CoT）提示。表 I 详述了大型语言模型接收的传感器输入。
- SafetyMonitor：通过几何检查和简化的交通规则实现。它验证所提议的机动是否在基于预测轨迹的所有感知动态物体中保持最低安全距离。如果预测到违反，则标记为“不安全”。
- 安全评估器：监控传入的传感器数据模式。对于这个用例，它指示故障注入器定期引入特定攻击。
- FaultInjector：基于安全评估触发器模拟两种攻击类型：
  1）幽灵障碍注入：向提供给生成器的感知状态中添加一个不存在的动态障碍。
  2）轨迹欺骗：修改真实检测到的车辆的预测速度或路径，使其看起来比实际更危险。
- PerformanceOracle：跟踪交叉口清除时间和最大纵向/横向加速度/冲击。如果超过阈值则标记“performance_fail”。
- RecoveryPlanner：一个简单的基于规则的代理。它使用与 SafetyMonitor 相同的几何检查，标记不安全情况的检查。如果检测到不安全情况，它会用“emergency_brake”替代生成模块的决定。

1) 积分： CARLA 模拟器 [31] 通过自定义 CPS-Guard CarlaInterface（图 2 ）进行使用。StateManager 跟踪感知到的对象列表（来自 CARLA 的传感器）、建议的动作以及相关的 CoT 解释。

2) 编排逻辑： 控制器在每个模拟时间步骤（100 毫秒）内顺序执行角色，其中处理按照模拟时间的 100 毫秒对齐，顺序如下：环境更新、生成器、安全监控器、安全评估者、故障注入器（条件），性能预测，决策（控制器在不安全时激活恢复计划器），动作执行（图 1 ）。

C. 测试场景

我们设计了具有不同复杂性的模拟场景并注入了故障/攻击：

1）名义：交通畅通，通行权明确。
2）拥堵：交通密度中等，需要仔细让行和选择间隙。
3）冲突交通：车辆同时从多个方向驶来，测试导航逻辑。
4）幽灵障碍攻击：正常场景 + FaultInjector 在交叉口入口附近添加一个幽灵障碍。
5）轨迹欺骗攻击：拥挤场景 + FaultInjector 欺骗迎面而来的汽车的轨迹，使其看起来具有攻击性。
6）行人过街：模拟行人穿过自动驾驶车辆预期路径的情景。

每个场景运行了 15 次，交通模式和时间有所变化。

D. 预期结果和指标

我们使用了 CPS-Guard 进行了评估：

- 安全违规：SafetyMonitor 标记为“不安全”操作的频率（基于几何/规则指示潜在的碰撞）。CARLA 记录的实际碰撞作为真实情况的确认。
- 安全弹性：当生成器（LLM）遇到注入故障时的反应。是否表现出行为不稳定、冻结（死锁），或按照伪造数据引发的不安全指令行事？攻击期间 SafetyMonitor 激活的频率。
- 性能下降：在拥堵或受到攻击的情况下，交叉口清除时间增加或舒适性违规（抖动/加速度）。
- 恢复效果：当由 SafetyMonitor 激活时，RecoveryPlanner（紧急刹车）在防止实际碰撞方面的成功率。

指标是由 DependabilityMetrics 组件收集的。

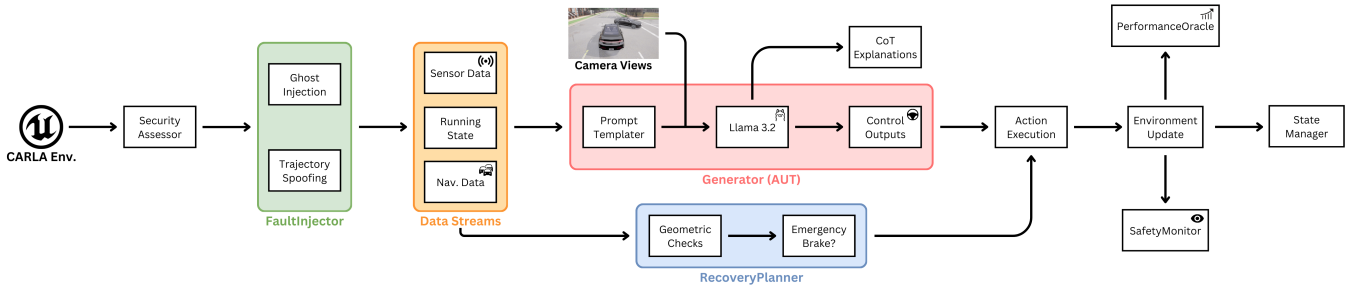

Fig. 3. 在人机交互系统中，高级协调 AI 生成器和其他角色。CARLA 环境提供导航（地图、交通、路点）和传感器数据。SecurityAssessor 可以通过 FaultInjector 注入故障。这些数据流以及运行状态被送入提示模板生成器，以为 Llama 3.2 11B 模型生成文本表示。相机视图直接传递给 LLM。Llama 3.2 同时生成控制输出和相应的解释。动作执行模块然后将这些输出应用于环境中。与此同时，RecoveryPlanner 使用几何检查并决定是否使用紧急制动，以覆盖所有其他动作。运行状态通过 StateManager 更新，以包括过去的动作和相关的 CoT 解释。PerformanceOracle 和 SafetyMonitor 负责跟踪性能。

TABLE I
用例架构的 CARLA 传感器输入

Sensor Input	Description
LiDAR-based Obstacle Summary	Textual summary of obstacles extracted from the LiDAR. Instead of using raw 3D data, the CarlaInterface aggregates nearby objects (vehicles, pedestrians, static obstacles) with positions & dimensions.
Radar Summary	A text summary of radar detections that includes each object' s range and relative radial velocity.
Front RGB Camera	An RGB image captured from the front-facing camera passed directly to the LLM.
Third-Person View Camera	An RGB image providing a broader, third-person perspective of the intersection. This image delivers contextual clues about background traffic and environmental layout.
IMU Summary	A text-based summary of inertial measurements that includes linear acceleration, angular velocity, and heading. This information succinctly describes the vehicle' s motion dynamics.
Vehicle Speed	A numerical value representing the current speed of the vehicle, extracted from vehicle odometry.
HD Map & Waypoint Data	A structured list of upcoming way points or lane center coordinates derived from a high-definition map. This input supports high-level route planning and navigation.
Traffic Controls Status	A concise textual report detailing the state of nearby traffic signals and the presence of key road signs.

V. 结果与分析

本节展示了在定义的场景中执行自主交叉口导航用例与 CPS-Guard 的结果（每个场景运行 15 次，总共 90 次运行）。

A. 安全评估

SafetyMonitor 主动检查 LLM Generator 提出的操作。表 II 总结了 SafetyMonitor 标记至少一个“不安全”预测的运行百分比和在 CARLA 中观察到的实际碰撞率。

TABLE II
安全监控激活次数与碰撞率

Scenario Type	Monitor Flags "Unsafe" (%)	Collision Rate (%)
Nominal	6.7 % (1/15)	0.0 % (0/15)
Congested	20.0 % (3/15)	6.7 % (1/15)
Conflicting Traffic	33.3 % (5/15)	13.3 % (2/15)
Ghost Obstacle Attack	86.7 % (13/15)	6.7 % (1/15)
Trajectory Spoof Attack	60.0 % (9/15)	20.0 % (3/15)
Pedestrian Crossing	26.7 % (4/15)	6.7 % (1/15)
Overall Avg.	38.9 %	8.9 %

观察结果：

- LLM 规划器在正常情况下表现出合理的安全性，但在复杂情况下（拥挤的、冲突的交通）越来越显得力不从心。它暴露出的弱点验证了我们多角色循环的必要性。
- 安全攻击显著触发了 SafetyMonitor。幽灵障碍攻击经常导致 LLM 提出突然刹车或转向的建议，这些行为被监测系统视为不安全。轨迹欺骗常常使 LLM 出现不必要的让步或犹豫，有时会导致后来被监测系统标记为冲突的情况。
- 实际碰撞次数低于监控标志次数，主要是因为当 SafetyMonitor 触发时，RecoveryPlanner（紧急刹车）经常能成功介入。在监控标志出现但仍然发生碰撞的情况下，通常涉及很短的碰撞时间，从而使刹车不足或复杂的多车辆交互。

CPS-Guard 有效识别了 LLM 计划器提出潜在不安全行动的情形。

B. 安全评估与韧性

故障注入器在 SecurityAssessor 的指导下成功引入了故障。分析集中在 LLM 的反应上：

- 幽灵障碍：尽管视觉输入与传感器输入相矛盾，LLM 仍持续对幽灵障碍作出反应。它通常建议立即刹车或显著减速，将其视为真实障碍。这常常导致性能问题（突然停下，增加通行时间），并且如果刹车过于突然，常常被安全监控标记出来。

- 轨迹欺骗：对于被欺骗的激进轨迹，大型语言模型表现出敏感性，通常选择让步或等待的时间比实际需要的长得多，从而显著影响了性能。在 3 次运行中（20 %），这种过度谨慎导致自动驾驶车辆变得“卡住”，无法找到一个自认为安全的空隙，最终导致了只有通过模拟超时才能打破的僵局场景。

CPS-Guard 的角色允许系统地注入和观察 AI 对传感器数据操控的脆弱性。

C. 性能影响

PerformanceOracle 跟踪了交叉口通过时间和舒适性指标。图 4 显示了平均通过时间。

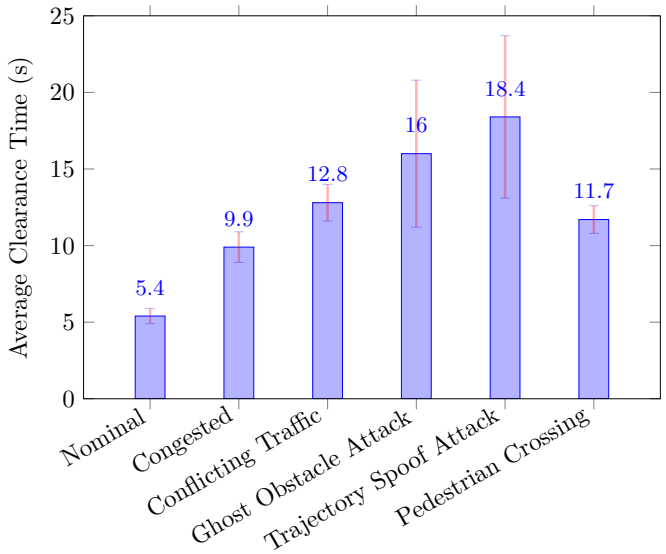

Fig. 4. 不同情境下的平均路口清空时间。误差线表示在 15 次模拟运行中的标准偏差。

正如预期的那样，拥堵和冲突的交通增加了清理时间。安全攻击具有重大影响：幻影障碍物导致急刹车，有时由于需要恢复而增加了时间，而轨迹欺骗则显著增加了等待时间，这是由于 LLM 对假设的攻击性行为反应过于谨慎造成的。舒适性违例（高变速/加速度）最常发生在恢复刹车和应对幻影障碍物时。

D. 恢复效果

当 SafetyMonitor 标记为“不安全”时，简单的 RecoveryPlanner（紧急制动）就会被触发。

- 在启用该系统的情况下，它成功地防止了碰撞，否则在模拟接近事故场景时，很可能会发生碰撞。
- 通常在不安全情况发展过快，单凭刹车不足以应对或涉及复杂的侧面碰撞情景时，可能会发生故障。

这凸显了监控-恢复循环的重要性，并且也表明在未来的工作中需要比简单制动更复杂的恢复策略。

E. *CPS-Guard* 贡献分析

此用例说明了 CPS-Guard 如何促进 V & V：

- 多角色设置允许评估 LLM 计划器的安全性、保密性和性能方面。
- 故障注入器和安全评估器使得针对特定攻击向量的系统化测试成为可能。
- 闭环协调揭示了交互作用，例如安全攻击导致安全监控触发，或者恢复动作影响性能。
- DependabilityMetrics 提供定量数据，总结了跨运行和场景的复杂行为。

该框架提供了一种系统的方法来挑战 AI 组件并评估其可靠性。

VI. 讨论

自动交叉路口导航用例的结果提供了有关在复杂的 CPS 中确保 AI 可靠性所面临挑战的见解，并展示了 CPS-Guard 框架多角色协调方法的效用。

A. 对人工智能/大语言模型在网络物理系统中保障的影响

实验表明，即使是复杂的 AI 模型如大型语言模型（LLM）在面对复杂情境和威胁时也表现出脆弱性。LLM 规划者在拥堵、目标冲突，特别是在模拟攻击下表现出安全性和性能的下降。它对伪造数据（鬼影障碍和操纵轨迹）的敏感性是一个显著的担忧，这表明单靠 AI 感知的世界模型而没有稳健的验证机制是高风险的。

安全攻击直接影响安全性（通过诱发不安全反应或犹豫导致二次冲突）和性能（通过引起过度谨慎或交通瘫痪）。这表明需要像 CPS-Guard 这样的 V & V 框架，它们可以同时评估这些属性并分析它们之间的交互，而不是将它们孤立地对待。简单恢复机制的部分成功表明了运行时保证循环的重要性，但同时也表明了需要针对特定故障模式的更高级的监控和恢复策略。

B. *CPS-Guard* 的有效性和作用

CPS-Guard 被证明在构建这个复杂的 V & V 任务中是有效的。其关键优势在使用案例中包括：

- 结构化评估：将具体的可靠性问题（安全性、保密性、性能）分配给不同的角色提供了清晰的认识，并允许模块化地实施检查和攻击。
- 系统性故障/攻击注入：FaultInjector 角色使得控制性引入安全威胁成为可能，从而允许对 AI 的抗风险能力进行系统评估。
- 闭环分析：该框架捕捉了动态反馈回路，其中 AI 行动影响环境，进而影响后续的 AI 输入和 V & V 评估，包括恢复行动。
- 扩展潜力：虽然这里使用了简单的监控和恢复，但基于角色的结构可以很容易地适应更复杂的实现（例如，基于 STL 的监控器、基于 AI 的安全评估器）。

CPS-Guard 充当“环中 V & V 协调器”，使工程师能够配置一个虚拟团队，由专业代理组成，这些代理不断探测和评估其模拟操作环境中的 AI AUT。这种方法有助于识别弱点，理解可靠性方面之间的交互，并评估如监视器和恢复规划器之类的保证机制的有效性。

这项工作存在若干限制，提示了未来研究的方向：

未来的方向集中在：

1) 开发一个更丰富的预定义 V & V 角色库，结合多种技术（STL 监控、简化的形式化方法、基于 ML 的异常检测）。一种方法可能是根据问题和约束，使用 LLM 自动生成 V & V 角色。

2) 将 CPS-Guard 与硬件在环（HIL）设置相结合，以评估其有效性。
3) 将 CPS-Guard 应用于其他领域，包括工业机器人和可靠的农业自动化（例如，智能农业中安全的人机协作）。
4) 研究优化并消除限制 CPS-Guard 在模拟外测试中有效性的瓶颈。
5) 开发专门的评估指标，针对大语言模型（LLM）特定的失效模式，如幻觉，并将这些指标整合到 CPS-Guard 框架中，以提高基于 LLM 的组件保障。

VII. 结论

本文介绍了 CPS-Guard ，这是一种利用多角色编排来构建和自动化基于 AI 的 CPS 迭代保障过程的框架。通过在模拟的 CPS 环境中为角色分配专门的 V & V 函数，CPS-Guard 系统地评估 AI 行为是否满足一系列可靠性要求。其迭代闭环的方法允许分析复杂的交互作用并对运行时保障机制进行持续评估。

我们在一个关于使用基于 LLM 的规划器进行自动驾驶车辆交叉路口导航的案例研究中展示了 CPS-Guard 的效用。该案例研究展示了 CPS-Guard 如何有效地引入故障，检测安全和安全漏洞，评估性能影响，以及评估恢复措施。CPS-Guard 提供了定量的保障指标。

总之，CPS-Guard 为在安全和安全性至关重要的 CPS 中的 AI V & V 提供了一种实用的方法。进一步的工作应关注已识别的挑战，如规范工作的努力和弥合从模拟到现实的差距。

References

[1] G. Klein, J. Andronick, K. Elphinstone, T. Murray, T. Sewell, R. Kolanski, and G. Heiser, "Comprehensive formal verification of an os microkernel," *ACM Trans. Comput. Syst.*, 2014.
[2] K. Lekadir, R. Osuala, C. Gallina, N. Lazrak, and K. e. a. Kushibara, "Future-ai: Guiding principles and consensus recommendations for trustworthy artificial intelligence in medical imaging," *arXiv preprint arXiv:2109.09658*, 2024.
[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2015.
[4] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. MIT Press, 1999.
[5] R. Hawkins, T. Kelly, J. Knight, and P. Graydon, "A new approach to creating clear safety arguments," in *Advances in Systems Safety*, 2011.
[6] R. Alur, "Principles of cyber-physical systems," *MIT Press*, 2015.
[7] C. Birchler, S. Khatiri, P. Rani, T. Kehrer, and S. Panichella, "A roadmap for simulation-based testing of autonomous cyber-physical systems: Challenges and future direction," *arXiv preprint*, 2024.
[8] M. Leucker and C. Schallhart, "A brief account of runtime verification," *Journal of Logic and Algebraic Programming*, 2009.
[9] C. Sanchez and G. e. a. Schneider, "A survey of challenges for runtime verification from advanced application domains (beyond software)," *Formal Methods in System Design*, 2019.
[10] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer, "Fault injection techniques and tools," *Computer*, 1997.
[11] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy (SP)*, 2017.
[12] M.-I. Nicolae, M. Sinn, M. N. Tran, A. Rawat, M. Wistuba, and V. e. a. Zittel, "Adversarial robustness toolbox v1.0.0," *arXiv preprint arXiv:1807.01069*, 2018.
[13] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *Computer Aided Verification*, 2017.
[14] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "AI^2: Safety and robustness certification of neural networks with abstract interpretation," in *IEEE Symposium on Security and Privacy (SP)*, 2018.
[15] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.
[16] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Neural Information Processing Systems 30*, 2017.
[17] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM Computing Surveys (CSUR)*, 2021.
[18] V. Landersheim, M. Jurisch, R. Bartolozzi, G. Stoll, R. Möller, and H. Dann, "Simulation-based testing of subsystems for autonomous vehicles at the example of an active suspension control system," 2022.
[19] D. Humeniuk, F. Khomh, and G. Antoniol, "Ambiegen: A search-based framework for autonomous systems testing," in *arXiv preprint arXiv:2301.01234*, 2023.
[20] R. Uuk, C. I. Gutierrez, D. Guppy, L. Lauwaert, A. Kasirzadeh, and L. e. a. Velasco, "A taxonomy of systemic risks from general-purpose AI," 2024, arXiv preprint.
[21] S. Sheikhi, U. Mehmood, S. Bak, S. A. Smolka, and S. D. Stoller, "The black-box simplex architecture for runtime assurance of multi-agent cps," *Innovations in Systems and Software Engineering*, 2024.
[22] M. Abate, E. Feron, and S. Coogan, "Monitor-based runtime assurance for temporal logic specifications," *arXiv preprint*, 2019.
[23] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modeling and Analysis of Timed and Fault-Tolerant Systems*, 2004.
[24] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *Computer Aided Verification*, 2010.
[25] M. Ahmed, O. Kazar, A. Ratnayake, and S. Harous, "Cyber-physical system model based on multi-agent system," *IET Cyber-Physical Systems: Theory & Applications*, 2024.
[26] H. Chase, "Langchain," 2022, gitHub repository.
[27] J. Liu, "Llamaindex (gpt index)," 2022, gitHub repository.
[28] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Li, and E. e. a. Zhu, "Autogen: Enabling next-gen llm applications via multi-agent conversation framework," *arXiv preprint arXiv:2308.08155*, 2023.
[29] T. Srinivasan and S. Patapati, "Llmorchestrator: A multi-model llm orchestration framework for reducing bias and iterative reasoning," 2025.
[30] D. Nickovic, A. Pant, and G. Schneider, "Rtamt: Online robustness monitors from signal temporal logic," in *Runtime Verification*, 2016.
[31] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017.
[32] S. Shah and Dey, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017.
[33] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2004.
[34] A. Grattafiori, A. Dubey, A. Jauhri, and A. e. a. Pandey, "The llama 3 herd of models," 2024, arXiv preprint.
[35] X. Li, A. Kumar, and S. et al., "Dialogue-based generation of self-driving simulation scenarios using large language models," 2023, arXiv preprint.