Watchdogs and Oracles: Runtime Verification Meets Large Language Models for Autonomous Systems

Angelo Ferrando
University of Modena and Reggio Emilia
Modena, Italy
angelo.ferrando@unimore.it

Assuring the safety and trustworthiness of autonomous systems is particularly difficult when learning-enabled components and open environments are involved. Formal methods provide strong guarantees but depend on complete models and static assumptions. Runtime verification (RV) complements them by monitoring executions at run time and, in its predictive variants, by anticipating potential violations. Large language models (LLMs), meanwhile, excel at translating natural language into formal artefacts and recognising patterns in data, yet they remain error-prone and lack formal guarantees. This vision paper argues for a symbiotic integration of RV and LLMs. RV can serve as a guardrail for LLM-driven autonomy, while LLMs can extend RV by assisting specification capture, supporting anticipatory reasoning, and helping to handle uncertainty. We outline how this mutual reinforcement differs from existing surveys and roadmaps, discuss challenges and certification implications, and identify future research directions towards dependable autonomy.

1 Introduction

Autonomous systems are becoming increasingly pervasive, from self-driving vehicles navigating complex traffic environments, to collaborative industrial robots, to adaptive cyber-physical infrastructures. Their decision-making blends symbolic control with machine learning, and often relies on large-scale perception and reasoning components. While such systems promise unprecedented levels of autonomy and efficiency, they also pose acute challenges for safety, reliability, and accountability. Failures may lead not only to costly disruptions, but to physical harm and erosion of public trust.

Formal methods (FMs) have long provided mathematically rigorous tools to specify, model, and verify systems before deployment. However, these techniques typically assume complete and accurate models, and they encounter fundamental limitations when faced with opaque neural controllers, distributional shift, or highly dynamic and uncertain environments. To bridge this gap, *runtime verification* (RV) [2] has emerged as a complementary technique: instead of proving all properties ahead of time, RV monitors executions against temporal specifications and can detect or even enforce safety at run time [8]. Recent advances in predictive and anticipatory RV extend this capability further by forecasting property violations before they occur, enabling proactive interventions [18, 10, 1].

In parallel, *large language models* (LLMs) have transformed natural language processing and increasingly influence software engineering and formal methods. LLMs can translate informal requirements into formal artefacts, propose invariants, ranking functions, or even sketch verification algorithms [4, 5, 15, 17]. By lowering the barrier to expressing formal properties, they democratise access to formal verification. Yet LLMs remain brittle: they hallucinate, misinterpret context, and provide no formal guarantees about their outputs.

This tension between the strengths and weaknesses of RV and LLMs motivates our vision. We argue that RV and LLMs can be integrated in a symbiotic way. RV can serve as a *guardrail* for LLM-driven

Matt Luckcuck, Maike Schwammberger & Mengwei Xu (Eds.):
Seventh International Workshop on Formal Methods
for Autonomous Systems (FMAS 2025)
EPTCS 436, 2025, pp. 80–87, doi:10.4204/EPTCS.436.8

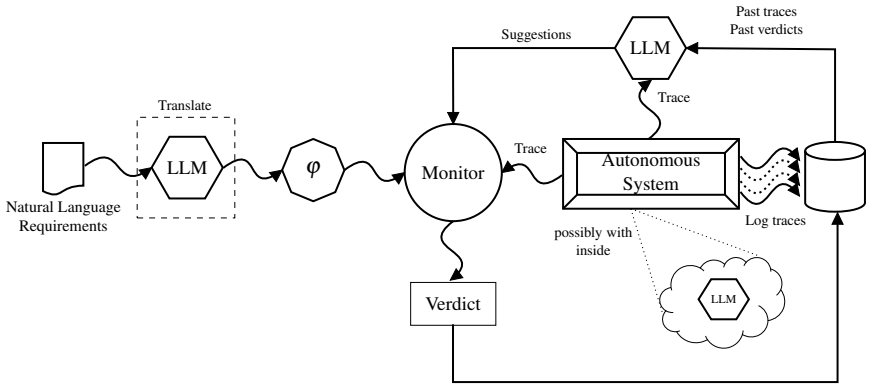

Figure 1: Envisioned architecture for RV+LLM integration. LLMs contribute as (i) enablers, translating informal requirements into candidate specifications, (ii) collaborators, providing predictive continuations to support anticipatory RV, and (iii) subjects, whose outputs are supervised by RV monitors to ensure safety. This dual role illustrates the bidirectional nature of our vision.

autonomy, ensuring safety and trustworthiness in execution. Conversely, LLMs can act as *enablers* for RV, by generating specifications and monitors from natural language, predicting event patterns that extend beyond traditional logics, and assisting engineers in scaling monitoring infrastructures. Our focus in this paper is not on generic integration of LLMs and formal methods, which has already been surveyed elsewhere [11, 19], but specifically on the intersection of RV and LLMs in the context of autonomous systems.

We position this contribution as a vision paper: synthesising emerging evidence, highlighting lessons learned from recent experiments, and sketching promising avenues for future research. In particular, we emphasise (i) online and predictive monitoring of autonomous systems, (ii) LLM-assisted synthesis of specifications and monitors, and (iii) implications for certification and trustworthy deployment. By articulating this agenda, we aim to stimulate discussion in the formal methods community and to guide the development of practical, dependable solutions for learning-enabled autonomy.

To ground this agenda, Figure 1 depicts the interplay we envision between runtime verification and large language models. The diagram illustrates how the two technologies complement each other within a unified architecture. LLMs operate at three levels: at the front end, translating natural language into candidate specifications; alongside monitoring, contributing predictive continuations, gap-filling, and uncertainty handling; and within the system itself, where RV acts as a guardrail over their outputs. This dual perspective—LLMs as both enablers and subjects of monitoring—captures the essence of our vision: RV supplies formal assurance, while LLMs expand accessibility and foresight. Importantly, in the role of subjects, LLMs are treated as opaque components whose outputs are supervised by RV monitors. Even though their internal reasoning cannot be verified, RV ensures that their observable behaviour complies with safety constraints. The model illustrated here will serve as a reference throughout the remainder of the paper.

2 Related Work

The integration of LLMs and formal methods has attracted growing attention. Huang et al. present a comprehensive survey of techniques for assessing the safety and trustworthiness of LLMs, covering

static analysis, dynamic validation, testing frameworks, and monitoring mechanisms [11]. Their scope is intentionally broad: runtime monitoring appears as one tool among many, alongside interpretability, robustness evaluation, and adversarial testing.

Complementing this, Zhang et al. propose a roadmap for combining formal methods and LLMs to build trustworthy AI agents [19]. A more recent survey specifically examines the use of LLMs for specification capture and analysis [3]. Together, these surveys discuss challenges across deductive verification, theorem proving, model checking, and certification, with a largely technology-agnostic and domain-independent perspective. By contrast, our contribution is narrower but deeper: we focus specifically on runtime verification in the context of autonomous systems, emphasising predictive monitoring and imperfect-information settings.

More specialised efforts have examined concrete intersections of LLMs and runtime verification. Cohen and Peled show how LLMs can aid monitor construction by translating natural-language requirements into formal specifications [5]. Such work demonstrates how LLMs can reduce the entry barrier to RV, but it remains focused on static monitor synthesis rather than online monitoring.

In parallel, predictive runtime verification research investigates how monitors can conclude verdicts with *less observation* than classical semantics would require, by exploiting system models or reasoning over partial trace prefixes [10, 1, 18]. This anticipatory capability is critical in autonomous domains, where delayed detection can lead to unsafe behaviour. Here we see a natural role for LLMs: they could help infer approximate system models from informal descriptions, or reduce the set of future events that must still be considered (for instance, by recognising that certain actions will not reoccur). Integrating such anticipatory monitoring with LLM-assisted modelling could yield monitors that are both timely and practical in complex, data-rich environments.

A different strand of work treats the LLM itself as the subject of monitoring. Pro2Guard, for instance, abstracts the behaviour of an LLM agent into a Discrete-Time Markov Chain (DTMC) and applies probabilistic model checking to anticipate unsafe outcomes [16]. In contrast, Statistical Runtime Verification for LLMs adopts a black-box view, using robustness estimation to bound confidence under perturbations [13]. Although these approaches differ in technique, they share the same perspective: RV acts as an external safeguard around the LLM, but the LLM does not contribute to the monitoring process. Our vision inverts this relationship. We propose that LLMs should not only be monitored but also play an active role in enhancing RV itself—supporting anticipatory reasoning and helping monitors cope with imperfect information.

In summary, existing surveys and roadmaps consider the broad landscape of LLMs and formal methods, while specific contributions address either monitor synthesis or the monitoring of LLMs as opaque systems. By contrast, our vision focuses on the synergy of LLMs and runtime verification within *autonomous systems*. Domains such as medical robotics, autonomous driving, and adaptive industrial systems pose distinctive challenges of real-time monitoring, incomplete information, and safety-critical decision making. We argue that combining RV's formal rigour and online guarantees with the adaptive and predictive capabilities of LLMs offers a promising path toward trustworthy, anticipatory supervision for autonomy—distinct from broader, technology-agnostic surveys and roadmaps in the literature.

3 LLMs in Support of Runtime Verification

One clear opportunity for synergy is using LLMs to lower the barrier to RV. In industrial practice, requirements are overwhelmingly expressed in natural language, and translating them into temporal logic is labour-intensive and error-prone. LLMs can assist here: Cohen and Peled demonstrated that monitors

can be synthesised from plain-English descriptions via an iterative prompt-and-check loop [4], while follow-up work showed how LLMs may even propose whole verification algorithms in pseudocode, to be validated and refined by experts [5]. Similarly, tools such as ClassInvGen [15] and Lemur [17] exploit LLMs to suggest invariants or ranking functions, with static analysers providing rigorous validation.

Where LLMs may offer a more transformative role is in *predictive RV*. Unlike classical monitoring—checking satisfaction on an observed trace—predictive RV ventures into the realm of uncertainty: anticipating likely future events and estimating violation probabilities. The approach by Zhang, Leucker and Dong [18] formalises predictive semantics for runtime verification, enabling monitors to look ahead beyond the current execution prefix. Relatedly, work on imperfect-information monitoring [9] addresses how to reason when only partial observations of the system are available. LLMs could support such approaches by learning from traces to propose plausible continuations (filling gaps) or high-level predictions that, when fed into formally synthesised monitors, enable earlier and more informed interventions.

LLMs could significantly extend these predictive capabilities. By learning from historical traces or domain-specific corpora, they can recognise subtle patterns of behaviour and propose likely continuations of execution. Coupled with formal monitors, such insights enable anticipatory interventions: not merely flagging a violation once it occurs, but estimating its probability ahead of time and prompting earlier corrective action. For instance, an LLM trained on driving logs might infer that after observing a pedestrian near a crossing, a braking event is highly likely. Feeding this continuation into a predictive RV monitor allows the system to assess the probability of a safety violation sooner than observation alone would permit. Such foresight is particularly valuable in safety-critical domains where avoiding unsafe conditions requires anticipation as much as detection.

Thus, while we maintain that *monitor synthesis and enforcement must remain formal*, LLMs offer a promising role in the uncertain terrain of predictive RV—extrapolating, filling gaps, and proposing continuations where purely logic-based approaches struggle. When such predictions are unreliable or misleading, the RV framework can fall back to a conservative mode of operation: discarding unverifiable suggestions and reverting to classical monitoring semantics. This safeguard ensures that correctness is never compromised, even when LLM assistance fails. Exploring this balance between formal guarantees and predictive support represents, in our view, one of the most exciting directions for future research.

4 Runtime Verification to Guard LLMs

The complementary direction treats LLM-driven components themselves as subjects of monitoring and enforcement. Unlike traditional software modules, LLMs are stochastic, non-deterministic, and continually updated. Their internal reasoning processes are opaque, making pre-deployment formal verification impractical. This opacity and adaptivity make LLMs natural candidates for oversight by RV, which can provide on-line assurances even when static guarantees are unattainable.

Recent approaches illustrate this trend. Zhang et al. propose *RvLLM* [20], which augments LLMs with domain knowledge and enforces correctness by checking responses against safety constraints at run time, retrying or abstaining upon violations. Levy et al. introduce a *statistical* RV framework that interprets LLM confidence scores through probabilistic monitoring, providing robustness estimates when deterministic guarantees are out of reach [13]. These efforts highlight two central insights: (i) RV can act as a lightweight safety layer around untrusted models, and (ii) monitoring must be adapted to cope with inherently probabilistic behaviour.

A particularly promising avenue is the integration of *predictive RV*. Anticipatory monitors can compute lookahead satisfaction probabilities and pre-empt unsafe actions [10], while predictive prefixes al-

low violations of temporal logic properties to be detected earlier [1]. When applied to LLMs, these techniques could detect signs of drift or hallucination before erroneous outputs are delivered, e.g. flagging an unsafe instruction in dialogue before it is executed by a robotic agent.

The challenge becomes even sharper in settings with *distribution shift* or *incomplete information*, which are common in LLM deployments. Distributionally robust and conformal prediction monitors [21, 14] quantify uncertainty when the data distribution changes, a scenario directly relevant to LLMs exposed to novel prompts or adversarial contexts. Similarly, techniques for runtime verification with imperfect information [9] suggest that monitoring LLM behaviour under partial observability is not only possible but essential: the full reasoning state of an LLM is inaccessible, and only outputs (and limited internal signals such as probabilities) can be observed.

Taken together, these approaches indicate that RV can serve as a multi-layered guard for LLMs:

- **Syntactic monitoring**, filtering outputs for forbidden tokens or patterns.
- **Semantic monitoring**, enforcing domain-specific temporal or safety constraints.
- **Predictive monitoring**, anticipating unsafe continuations, hallucinations, or distribution shifts before they manifest.

This layered perspective also connects to certification. Since LLM internals cannot be fully verified, regulators may instead require runtime assurance frameworks that combine statistical confidence measures with formal monitors. We envision RV not merely as a reactive guardrail but as an anticipatory partner, enabling dependable use of LLMs within safety-critical autonomous systems.

To situate our vision within the broader landscape, Table 1 contrasts three existing lines of work—LLM-assisted specification, monitoring of LLMs, and predictive RV—with the perspective advanced in this paper. While Figure 1 illustrated how RV and LLMs can interact within a unified architecture, the table highlights how prior research has addressed only fragments of this picture. Our contribution differs in combining these strands: positioning LLMs not just as objects of monitoring or front-end translators, but as active components that can enhance predictive and uncertainty-aware runtime verification.

Table 1: Landscape of RV+LLM research and this vision.

Line of work	**Role of LLM**	**Role of RV**	**Limitation**
LLM → spec translation (e.g. Cohen & Peled)	NL → logic/specs; front-end assistant	Monitor synthesis; offline validation	Static; no anticipation; requires human vetting
Monitoring LLMs (e.g. Pro2Guard; Statistical RV)	LLM = monitored object; black-box outputs	Guardrail at runtime; enforcement / risk checks	LLM does not assist monitor; external, not bidirectional
Predictive RV (formal) (e.g. Hipler et al.; Ang & Mathur; Zhang–Leucker–Dong)	–	Anticipation via models/prefixes; early verdicts with less observation	No LLM-aided anticipation; no uncertainty resolution
This vision paper	**Active aide: prediction, modelling, gap-filling**	**Formal guardrail + anticipatory partner**	**Validation and certification challenges remain**

Citations (not in cells): LLM→spec: [5, 4]. Monitoring LLMs: [16, 13]. Predictive RV (formal): [10, 1, 18].

5 Challenges and Certification Implications

The integration of RV and LLMs holds great promise, but also exposes a set of difficult open problems:

- **Trust.** LLM-generated artefacts—whether properties, invariants, or predictions—cannot be accepted at face value. Without formal validation, subtle flaws may undermine the very safety guarantees that RV seeks to provide.

- **Efficiency.** Predictive monitors must operate within strict real-time budgets, especially in domains such as driving or medical robotics where delays are unacceptable. LLM queries may exceed these constraints, making lightweight distilled models or offline predictors necessary. Determining the boundary of feasible deployment remains an open question.
- **Human–AI collaboration.** Interfaces and workflows must be designed so that LLM-assisted RV remains intelligible, supporting rather than obscuring engineers' reasoning about system safety.
- **Robustness.** Both monitors and LLMs must continue to function under distribution shift or even adversarial manipulation.
- **Ethics and accountability.** When system behaviour results from LLM suggestions filtered through monitors, the locus of responsibility must remain transparent to maintain trust and assign liability.

These challenges feed directly into the problem of *certification*. Standards such as ISO 26262 for automotive functional safety [12], DO-178C for avionics software certification [6], and the forthcoming EU AI Act regulating trustworthy AI [7] demand structured evidence of safety, traceability, and compliance.

RV+LLM pipelines could provide certification evidence in novel ways. *LLM-assisted specification capture* can establish transparent links from informal requirements to executable monitors, improving traceability. *Predictive RV enhanced by LLM-based modelling or forecasting* can generate runtime assurance artefacts that complement offline analyses even in real-time environments, offering regulators a dynamic view of system safety. Finally, *monitor-assisted explanations of rejected LLM outputs* can increase auditability, clarifying to both engineers and auditors why certain behaviours were deemed unsafe. For this potential to materialise, however, certification frameworks themselves must evolve to recognise dynamic, runtime evidence as a legitimate complement to traditional verification artefacts.

Notably, existing predictive-RV frameworks rely on formal abstractions to anticipate violations, but none exploit LLMs to assist anticipation or to reduce the set of future observations needed for early conclusions. Similarly, imperfect-information monitoring has been studied through logical and automata-theoretic techniques, but not with LLMs acting as uncertainty resolvers. This gap underscores the novelty of our proposal: to treat LLMs not merely as translators of specifications, but as *active components of runtime monitors*, augmenting anticipatory reasoning and strengthening monitoring under uncertainty.

6 Conclusions and Future Work

Addressing the challenges of integrating RV and LLMs calls for progress on several interconnected fronts. A first priority is *hybrid predictive RV*, where LLM-based sequence prediction is combined with formally synthesised monitors to deliver earlier verdicts than traditional logics permit. A second is *domain-specific adaptation*: tailoring LLMs with corpora of safety standards and certification artefacts can constrain their outputs, yielding more precise and auditable specifications. Equally important are *interactive toolchains* that integrate generation, human refinement, and formal validation into a single loop, making monitor construction both collaborative and transparent. Finally, certification practices must evolve to embrace these innovations, formally recognising runtime assurance evidence from RV+LLM systems as part of assurance cases.

Our focus here is on RV, yet similar bidirectional opportunities may exist for other formal techniques: LLMs could assist in invariant discovery for model checking or proof search in theorem proving, while those methods in turn could provide stronger guarantees for LLM outputs. By contrast, much current research frames RV as a safeguard *around* LLMs. Our perspective inverts this relationship: LLMs themselves can act as *enablers* of novel forms of RV, supporting anticipatory reasoning and helping

to resolve uncertainty. This inversion opens a distinct research trajectory that complements existing monitoring approaches.

Taken together, these priorities suggest a compelling agenda. LLMs show promising ability to assist in translating natural language into formal artefacts, but remain imperfect and error-prone; RV provides enforcement and formal assurance. Their combination can lower the barrier to formalising requirements, enable anticipatory supervision of learning-enabled components, and enhance the trustworthiness of autonomous systems. Unlike broader surveys and roadmaps [19, 11], our focus is specifically on real-time monitoring, predictive semantics, and domain-specific deployment. Realising this vision will require advances in predictive reasoning, robust integration with human workflows, and stronger alignment with certification practices. Yet the potential benefits—dependable autonomy in medicine, transport, and generally in safety-critical infrastructures—make this a timely and urgent research direction.

References

[1] Zhendong Ang & Umang Mathur (2024): *Predictive Monitoring with Strong Trace Prefixes*. In Arie Gurfinkel & Vijay Ganesh, editors: *Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part II*, *Lecture Notes in Computer Science* 14682, Springer, pp. 182–204, doi:10.1007/978-3-031-65630-9_9.

[2] Ezio Bartocci, Yliès Falcone, Adrian Francalanza & Giles Reger (2018): *Introduction to Runtime Verification*. In Ezio Bartocci & Yliès Falcone, editors: *Lectures on Runtime Verification - Introductory and Advanced Topics*, *Lecture Notes in Computer Science* 10457, Springer, pp. 1–33, doi:10.1007/978-3-319-75632-5_1.

[3] Arshad Beg, Diarmuid P. O'Donoghue & Rosemary Monahan (2025): *Leveraging LLMs for Formal Software Requirements - Challenges and Prospects*. *CoRR* abs/2507.14330, doi:10.48550/ARXIV.2507.14330. arXiv:2507.14330.

[4] Itay Cohen & Doron Peled (2023): *End-to-End AI Generated Runtime Verification from Natural Language Specification*. In Bernhard Steffen, editor: *Bridging the Gap Between AI and Reality - First International Conference, AISoLA 2023, Crete, Greece, October 23-28, 2023, Selected Papers*, *Lecture Notes in Computer Science* 14129, Springer, pp. 362–384, doi:10.1007/978-3-031-73741-1_23.

[5] Itay Cohen & Doron Peled (2024): *LLM-Based Scheme for Synthesis of Formal Verification Algorithms*. In Bernhard Steffen, editor: *Bridging the Gap Between AI and Reality - Second International Conference, AISoLA 2024, Crete, Greece, October 30 - November 3, 2024, Proceedings*, *Lecture Notes in Computer Science* 15217, Springer, pp. 167–182, doi:10.1007/978-3-031-75434-0_11.

[6] Dewi Daniels (2011): *Position Paper: DO-178C/ED-12C and Object-Orientation for Critical Systems*. In Alexander B. Romanovsky & Tullio Vardanega, editors: *Reliable Software Technologies - Ada-Europe 2011 - 16th Ada-Europe International Conference on Reliable Software Technologies, Edinburgh, UK, June 20-24, 2011. Proceedings*, *Lecture Notes in Computer Science* 6652, Springer, pp. 211–213, doi:10.1007/978-3-642-21338-0_19.

[7] European Commission (2021): *Proposal for a Regulation of the European Parliament and of the Council Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act)*. https://artificialintelligenceact.eu/. European Commission, COM/2021/206 final.

[8] Yliès Falcone, Laurent Mounier, Jean-Claude Fernandez & Jean-Luc Richier (2011): *Runtime enforcement monitors: composition, synthesis, and enforcement abilities*. *Formal Methods Syst. Des.* 38(3), pp. 223–262, doi:10.1007/S10703-011-0114-4.

[9] Angelo Ferrando & Vadim Malvone (2022): *Runtime Verification with Imperfect Information Through Indistinguishability Relations*. In Bernd-Holger Schlingloff & Ming Chai, editors: *Software Engineering and Formal Methods - 20th International Conference, SEFM 2022, Berlin, Germany, September 26-*

30, 2022, Proceedings, *Lecture Notes in Computer Science* 13550, Springer, pp. 335–351, doi:10.1007/978-3-031-17108-6_21.

[10] Raik Hipler, Hannes Kallwies, Martin Leucker & César Sánchez (2024): *General Anticipatory Runtime Verification*. In Arie Gurfinkel & Vijay Ganesh, editors: *Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part II*, *Lecture Notes in Computer Science* 14682, Springer, pp. 133–155, doi:10.1007/978-3-031-65630-9_7.

[11] Xiaowei Huang, Wenjie Ruan, Wei Huang, Gaojie Jin, Yi Dong, Changshun Wu, Saddek Bensalem, Ronghui Mu, Yi Qi, Xingyu Zhao, Kaiwen Cai, Yanghao Zhang, Sihao Wu, Peipei Xu, Dengyu Wu, André Freitas & Mustafa A. Mustafa (2024): *A survey of safety and trustworthiness of large language models through the lens of verification and validation*. *Artif. Intell. Rev.* 57(7), p. 175, doi:10.1007/S10462-024-10824-0.

[12] International Organization for Standardization (2018): *ISO 26262: Road vehicles – Functional safety*. https://www.iso.org/standard/68383.html. International Organization for Standardization, first edition 2011, updated 2018.

[13] Natan Levy, Adiel Ashrov & Guy Katz (2025): *Statistical Runtime Verification for LLMs via Robustness Estimation*. In Bettina Könighofer & Hazem Torfah, editors: *Runtime Verification - 25th International Conference, RV 2025, Graz, Austria, September 15-19, 2025, Proceedings*, *Lecture Notes in Computer Science* 16087, Springer, pp. 457–476, doi:10.1007/978-3-032-05435-7_25.

[14] Lars Lindemann, Xin Qin, Jyotirmoy V. Deshmukh & George J. Pappas (2023): *Conformal Prediction for STL Runtime Verification*. In: *59th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2023, Monticello, IL, USA, September 26-29, 2023*, IEEE, p. 1, doi:10.1109/ALLERTON58177.2023.10313399.

[15] Chuyue Sun, Viraj Agashe, Saikat Chakraborty, Jubi Taneja, Clark W. Barrett, David L. Dill, Xiaokang Qiu & Shuvendu K. Lahiri (2025): *ClassInvGen: Class Invariant Synthesis using Large Language Models*. *CoRR* abs/2502.18917, doi:10.48550/ARXIV.2502.18917. arXiv:2502.18917.

[16] Haoyu Wang, Chris M. Poskitt, Jun Sun & Jiali Wei (2025): *Pro2Guard: Proactive Runtime Enforcement of LLM Agent Safety via Probabilistic Model Checking*. arXiv:2508.00500.

[17] Haoze Wu, Clark W. Barrett & Nina Narodytska (2024): *Lemur: Integrating Large Language Models in Automated Program Verification*. In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, OpenReview.net. Available at https://openreview.net/forum?id=Q3YaCghZNt.

[18] Xian Zhang, Martin Leucker & Wei Dong (2012): *Runtime Verification with Predictive Semantics*. In Alwyn Goodloe & Suzette Person, editors: *NASA Formal Methods - 4th International Symposium, NFM 2012, Norfolk, VA, USA, April 3-5, 2012. Proceedings*, *Lecture Notes in Computer Science* 7226, Springer, pp. 418–432, doi:10.1007/978-3-642-28891-3_37.

[19] Yedi Zhang, Yufan Cai, Xinyue Zuo, Xiaokun Luan, Kailong Wang, Zhe Hou, Yifan Zhang, Zhiyuan Wei, Meng Sun, Jun Sun, Jing Sun & Jin Song Dong (2024): *The Fusion of Large Language Models and Formal Methods for Trustworthy AI Agents: A Roadmap*. *CoRR* abs/2412.06512, doi:10.48550/ARXIV.2412.06512. arXiv:2412.06512.

[20] Yedi Zhang, Sun Yi Emma, Annabelle Lee Jia En & Jin Song Dong (2025): *RvLLM: LLM Runtime Verification with Domain Knowledge*. *CoRR* abs/2505.18585, doi:10.48550/ARXIV.2505.18585. arXiv:2505.18585.

[21] Yiqi Zhao, Emily Zhu, Bardh Hoxha, Georgios Fainekos, Jyotirmoy Deshmukh & Lars Lindemann (2025): *Distributionally Robust Predictive Runtime Verification under Spatio-Temporal Logic Specifications*. *ACM Trans. Cyber-Phys. Syst.*, doi:10.1145/3748818.