RvLLM: LLM Runtime Verification with Domain Knowledge

Yedi Zhang[*]
National University of Singapore
Singapore

Sun Yi Emma
National University of Singapore
Singapore

Annabelle Lee Jia En
National University of Singapore
Singapore

Jin Song Dong
National University of Singapore
Singapore

Abstract

Large language models (LLMs) have emerged as a dominant AI paradigm due to their exceptional text understanding and generation capabilities. However, their tendency to generate inconsistent or erroneous outputs challenges their reliability, especially in high-stakes domains requiring accuracy and trustworthiness. Existing research primarily focuses on detecting and mitigating model misbehavior in general-purpose scenarios, often overlooking the potential of integrating domain-specific knowledge. In this work, we advance misbehavior detection by incorporating domain knowledge. The core idea is to design a general specification language that enables domain experts to customize domain-specific constraints in a lightweight and intuitive manner, supporting later runtime monitoring of LLM outputs. To achieve this, we design a novel specification language ESL and introduce a runtime verification framework RvLLM to validate LLM output against domain-specific constraints defined in ESL. RvLLM operates in two main stages: interpretation and reasoning. During interpretation, it derives interpretations of the specification based on the context, which then guide the reasoning process to identify inconsistencies. When new knowledge is derived, RvLLM issues a follow-up query to the LLM to further verify the consistency. We evaluate RvLLM on three representative tasks: violation detection against Singapore Rapid Transit Systems Act, numerical comparison, and inequality solving. Experimental results show that RvLLM effectively detects erroneous outputs across various LLMs in a lightweight and flexible manner. The results reveal that despite their impressive capabilities, LLMs remain prone to low-level errors due to a lack of formal guarantees during inference, and our framework offers a potential long-term solution by leveraging expert domain knowledge to rigorously and efficiently verify LLM outputs.

1 Introduction

Unlike rule-based systems [52] operating on predefined and deterministic rules, large language models (LLMs) [1, 23, 62, 43] learn data representation and processing automatically from the training datasets, achieving human-like or even superhuman performance, and have driven significant advancements in various practical applications [10, 33, 55, 44, 24]. However, their non-deterministic and unpredictable nature sometimes leads to inconsistent or erroneous outputs [3, 49], posing significant risks in safety-critical or knowledge-intensive domains. Recent studies [61, 5] have shown

[*]Corresponding author

that such misbehavior in LLMs cannot be fully eliminated, underscoring the urgent need for dedicated verification and validation methodologies to enhance the reliability of LLM-generated outputs.

LLM testing [67, 27, 30, 68] primarily aims to establish comprehensive benchmarks to evaluate the overall model performance against domain-agnostic criteria–such as accuracy, coherence, and fairness–in alignment with the intended application. While these approaches effectively assess general behavior and reveal edge cases that may provoke unexpected responses, they are limited to predefined benchmarks and lack the specificity needed to address domain-specific assessment needs. LLM verification, instead, may serve as a complementary mechanism to LLM testing by providing formal guarantees on model behavior. Despite substantial progress over the past decade in neural network verification [8, 19, 29, 34, 59, 48, 65], existing methods exhibit notable limitations: they are primarily tailored to simpler model architectures–such as deep or convolutional networks–and classification tasks, struggling to scale to the complexity of modern AI models and their diverse functionalities. Furthermore, these approaches target general properties such as robustness and fairness, making them ill-suited for domain-specific properties. Therefore, developing a dedicated verification paradigm tailored to LLMs is crucial for ensuring certified and reliable outputs.

On the other hand, the decline of rule-based expert systems [25, 31, 22] in the late 20th century can be attributed to their inability to handle incomplete information and poor scalability in real-world complexity. Defining comprehensive rule sets for open domains has been proven infeasible [52, 28]. We argue that such rules can only be statistically approximated–a capability exemplified by LLMs. However, this approximation cannot substitute for explicit rule encoding, and learning-based models remain inherently unreliable, particularly in edge cases demanding strict adherence to domain-specific requirements. This highlights the need for verifying explicitly defined rules in a lightweight and incremental manner for practical deployment and maintainability.

This paper. Building on these foundations, we propose runtime verification with domain knowledge as a sustainable solution to ensuring reliable LLM behavior, motivated by two key insights: i) Runtime verification offers a lightweight yet rigorous means to bridge testing and formal verification by assessing system behavior against formal properties during execution; ii) Existing work primarily focuses on generic misbehavior detection while often overlooking the critical role of domain-specific expertise. Integrating such knowledge is crucial for handling edge cases and enhancing LLM reliability in specialized tasks, where domain knowledge can significantly improve performance.

To achieve this, we introduce RvLLM, an innovative runtime verification framework tailored to LLMs to ensure reliable outputs, particularly against axiomatic domain specifications. The idea is to automatically check whether the LLM-generated responses adhere to domain-specific constraints expressed in a simple, adaptable specification language. This simplicity and flexibility empower domain experts to encode their knowledge and formally define constraints that capture the expected behavior of LLMs in specialized applications. To support this, we introduce ESL, a general language tailored for encoding rule-based domain expertise. ESL integrates natural language with formal logic to ensure the adaptability to the natural language settings of LLMs while enabling the structured and formal representation of properties to verify. This design allows for the rigorous specification of domain-specific criteria across diverse LLM applications.

Figure 1 presents an overview of our proposed framework. Given an ESL specification provided by domain experts, RvLLM first extracts relevant information from the LLM's context and outputs to interpret the specification, generating a set of propositional formulae along with truth assignments for the corresponding propositions. Following a normalization step, these formulae are transformed into a standard form and subsequently validated using a forward chaining procedure. This process either detects inconsistencies due to logical contradictions or infers new knowledge. Once new knowledge is inferred, the query generation module issues targeted queries, evaluates the LLM responses, and detects inconsistencies when outputs contradict the previously inferred knowledge.

Experimental results. We evaluate the effectiveness of RvLLM through experiments on three representative tasks: violation detection against Singapore Rapid Transit Systems Act [51], numerical comparison, and inequality solving, using a diverse set of LLMs. The experimental results show that RvLLM significantly enhances the reliability of LLM output in these domain-specific tasks. In the violation detection task, employing RvLLM as a complementary mechanism increases the true positive rate (TPR) by 15.7% to 50.2% across various models. In the numerical comparison task, RvLLM detects nearly all errors in the LLM responses across 100 randomly generated questions. In the inequality solving task, RvLLM facilitates a systematic analysis of the step-by-step reasoning

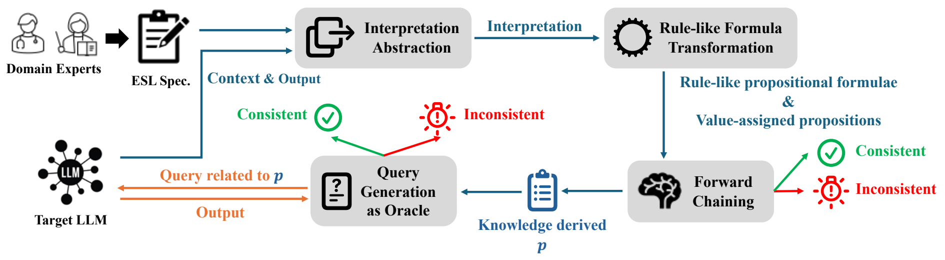


Figure 1: An overview of RvLLM. Given an LLM's context and outputs, RvLLM first extracts relevant propositions and translates the ESL specification into normalized propositional logic formulae. These are then processed through forward chaining to detect inconsistencies and infer new knowledge, which subsequently guides follow-up queries and consistency checks across successive LLM responses.

process from LLMs, effectively identifying its deviations from expert-defined specifications. These specifications are primarily grounded in fundamental algebraic inequality properties typically covered at the junior college level. Notably, despite the incomplete domain knowledge in the inequality solving task, RvLLM still achieves a TPR of 50% in detecting erroneous solutions. We argue that such a runtime verification approach can function as a critical safeguard for LLMs, particularly in rule-based or reasoning-intensive domains where adherence to specific constraints is essential. We further compare RvLLM with existing runtime hallucination-detection methods. Results show that RvLLM demonstrates superior performance by maintaining both high true positive and negative rates, whereas existing methods often exhibit a non-negligible trade-off between these metrics.

2 ESL: a simple way of specifying domain-specific properties

In this section, we introduce a general language, Expert Specification Language (ESL), which can be customized with domain-specified predicates by experts to impose behavioral constraints on LLMs.

2.1 Design guidance

In the following, we give our high-level requirements for designing a specification language that encapsulates LLM behavioral constraints from the perspective of domain experts. The core objective is to strike a balance between user-friendliness and expressiveness. We illustrate this with the following regulation from Singapore Rapid Transit System Act [51]:

> *"No person shall consume or attempt to consume any chewing gum or bubble gum while in or upon any part of the railway premises."*

Accessibility for domain experts. The primary goal of this specification language is to empower domain experts, even those with limited programming or formal methods experience, to effectively define and enforce constraints on the LLM's behavior. To balance expressiveness and user-friendliness, we base the specification rule on predicate logic [53, 20], restricted to prenex normal forms [53, 20] and consisting solely of universal quantifiers. We further constrain its quantifier-free part to a rule-like form (or implication), yielding a simplified variant of predicate logic expressible as $\forall \vec{x}.(f(\vec{x}) \Rightarrow g(\vec{x}))$[2]. Specifically, both the left-hand side (LHS) and right-hand side (RHS) of the rule-like part are quantifier-free predicate formulae composed of predicates applied to terms that may include variables (e.g., x), constants (e.g., "bubble gum"), and functions of terms (e.g., $\sin(x)$). For example, $\mathsf{ChewGum}(x)$ is an atomic predicate with a single variable argument x, and $\mathsf{IsGreater}(\mathsf{Square}(x), 0)$ is also an atomic predicate with two arguments: the function term $\mathsf{Square}(x)$ and the constant term 0.

Ease of customization. The specification language should be designed to support seamless customization, enabling adaptation to diverse domain-specific requirements. To facilitate this, we require each predicate in the specification to be associated with a natural language description that aligns with domain expertise. These descriptions allow customized predicates to remain clear, semantically grounded, and adaptable to specific application contexts. Additionally, built-in terms such as

[2]We use $\vec{x}$ instead of x as there could be a sequence of variables.

arithmetic functions can be provided to streamline constraint formulation and reduce the need for extensive customization. For example, a domain expert can define their specialized atomic predicates $\mathsf{ChewGum}(x)$ and $\mathsf{InRailway}(x)$ associated with a natural language description as follows:

"$\mathsf{ChewGum}(x) :=$ a person x consumes or attempts to consume chewing gum or bubble gum."
"$\mathsf{InRailway}(x) :=$ a person x is in some part of railway premises."

Efficient interpretation. The language should also support an efficient interpretation procedure to obtain propositional formulae from the specification rule for subsequent logical verification. To this end, we remove the universal quantifier from the standard predicate formulas. For example, the regulation previously mentioned, typically expressed as $\forall x.(\mathsf{InRailway}(x) \Rightarrow \neg\mathsf{ChewGum}(x))$, will be simplified to $\mathsf{InRailway}(x) \Rightarrow \neg\mathsf{ChewGum}(x)$.

2.2 Formalization

In this section, we formalize our specification language, ESL. We begin with the definition of Deductive Normal Form (DeNF), which serves as a foundation for the remainder of this paper.

Definition 1. *Given a propositional formula $\psi_1 \Rightarrow \psi_2$, if ψ_1 is a disjunctive norm form [53] and ψ_2 is a conjunctive norm form [53], then we call $\psi_1 \Rightarrow \psi_2$ is a deductive normal form (DeNF).*

Definition 2. *An* ESL *rule is a DeNF where each proposition is substituted by a predicate.*

An ESL specification $\mathcal{E}$ comprises three components: a variable set $\mathcal{V}$, a predicate set $\mathcal{P}$, and an ESL rule set $\mathcal{R}_\mathcal{E}$. For each predicate $f \in \mathcal{P}$, an associated natural language description is required. For instance, the following ESL specification encodes the regulation aforementioned in Section 2.1:

```
"Variables":    {"x"},
"Predicates":   {"ChewGum(x) := a person x consumes or attempts to consume chewing gum or bubble gum.",
                 "InRailway(x) := a person x is in some part of railway premises"},
"Rules":        {"InRailway(x) => not ChewGum(x)"}
```

Interpretation of ESL **rules**. An interpretation of an ESL rule r assigns objects to variables (i.e., variable binding) in the predicate of r and evaluates the truth value (True, False, or Unknown) of the resulting proposition after substitution. Note that we adopt the open-world assumption [17] to better accommodate the open-ended nature of LLMs. In this work, interpretations are derived from the context and LLM outputs, serving as the domain of discourse [26]. Now, consider the following contextual scenario:

"In a crowded MRT train, Alex nervously chews gum to ease stress before an interview. Suddenly, the train jolts, and the gum flies out, landing on a stranger's shirt. Awkward glances turn into laughter as apologies spill out, diffusing tension in the confined space."

The interpretation of the rule $\mathsf{InRailway}(x) \Rightarrow \neg\mathsf{ChewGum}(x)$ is $\{\mathsf{InRailway}(o_1) = \mathsf{True}$, $\mathsf{InRailway}(o_2) = \mathsf{True}$, $\mathsf{ChewGum}(o_1) = \mathsf{True}$, $\mathsf{ChewGum}(o_2) = \mathsf{Unknown}\}$, where $o_1 =$ 'Alex' and $o_2 =$ 'stranger'.

3 Methodologies of RvLLM

As illustrated in Figure 1, RvLLM conducts runtime verification through four stages: interpretation abstraction, rule normalization, forward chaining, and query generation. In this section, we detail the methodology of each stage and demonstrate its application using an example given in Figure 2.

3.1 Interpretation abstraction

Given a context and LLM outputs (the union denoted as σ) as the domain of discourse D and an ESL specification $\mathcal{E} = \langle \mathcal{V}, \mathcal{P}, \mathcal{R}_\mathcal{E} \rangle$, we first leverage a perception agent to obtain all possible objects from D and propositionalize as much as possible of the predicates defined in σ. We call this the *perception process*. Then, for each rule $r \in \mathcal{R}_\mathcal{E}$, we bind the variables in each predicate to all possible objects in a search-and-replace operation, to generate the propositional formulae set. Note that, given a

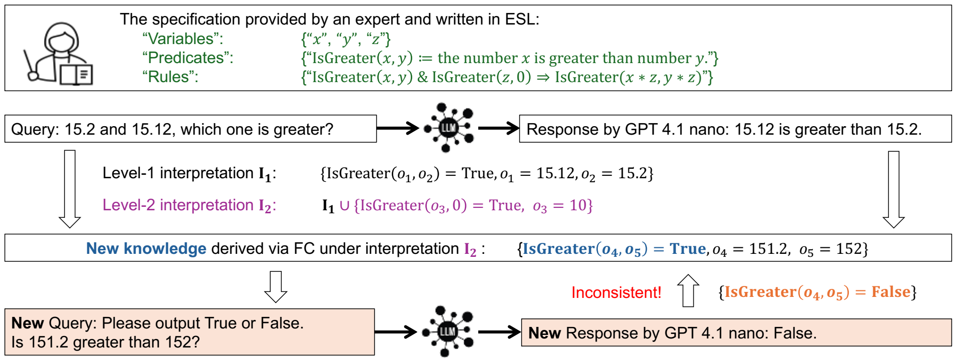

The specification provided by an expert and written in ESL:
"Variables": {"x", "y", "z"}
"Predicates": {"IsGreater(x, y) := the number x is greater than number y."}
"Rules": {"IsGreater(x, y) & IsGreater$(z, 0)$ ⇒ IsGreater$(x * z, y * z)$"}
Query: 15.2 and 15.12, which one is greater?
LLM
Response by GPT 4.1 nano: 15.12 is greater than 15.2.
Level-1 interpretation $\mathbf{I}_1$: {IsGreater(o_1, o_2) = True, o_1 = 15.12, o_2 = 15.2}
Level-2 interpretation $\mathbf{I}_2$: $\mathbf{I}_1$ ∪ {IsGreater$(o_3, 0)$ = True, o_3 = 10}
New knowledge derived via FC under interpretation $\mathbf{I}_2$: {IsGreater(o_4, o_5) = True, o_4 = 151.2, o_5 = 152}
Inconsistent!
{IsGreater(o_4, o_5) = False}
New Query: Please output True or False.
Is 151.2 greater than 152?
LLM
New Response by GPT 4.1 nano: False.

Figure 2: Runtime verification of GPT 4.1 nano by RvLLM for a number comparison task.

variable binding for a rule $r \in \mathcal{R}_{\mathcal{E}}$, the interpretation has two different results: i) all predicates in the LHS of r are propositionalized successfully with assigned truth value under the binding, which we call a complete binding in this work, or ii) only part of predicates are propositionalized due to the insufficient information from the domain of discourse, which we call a partial binding.

Given this distinction, we introduce two levels of interpretation in this work: Level-1 and Level-2. Specifically, Level-1 interpretation disregards all the partial variable binding and operates only on the complete variable bindings. Given a partial binding, Level-2 interpretation, in contrast, employs a perception agent to instantiate all the variables that are not assigned with an object in the binding, in a way such that the resultant proposition returns True given the domain of discourse.

Example 1. Consider the example in Figure 2, which illustrates our approach using a simple numerical comparison question with a specification encoding a basic algebraic property of multiplication, the interpretation abstraction process first gets all possible objects and propositionalizes the predicate IsGreater as follows: $\{p_0 = \mathsf{IsGreater}(o_1, o_2) = \mathsf{True}, p_1 = \mathsf{IsGreater}(o_2, o_1) = \mathsf{False}\}$ with $o_1 = 15.2$ and $o_2 = 15.12$. Then, it can obtain all possible variable binding $\{\flat_1, \flat_2\}$ for the rule ψ where $\flat_1 = \{x \mapsto o_1, y \mapsto o_2\}$ and $\flat_2\{x \mapsto o_2, y \mapsto o_1\}$. $\flat_1$ results in a formula $p_0 \wedge \mathsf{IsGreater}(z, 0) \Rightarrow \mathsf{IsGreater}(15.2 * z)$ for ψ, which is an partial binding. Similarly, $\flat_2$ is also partial. Consequently, for the Level-1 interpretation, no propositional formula is successfully generated, and the checking procedure exits with no inconsistency detected. While for the Level-2 interpretation, it successfully instantiates the variable z for the partial bindings $\flat_1$ and $\flat_2$ with an object $o_3 = 10$, and obtain an additional proposition $p_2 = \mathsf{IsGreater}(o_3, 0) = \mathsf{True}$. Then, the updated bindings are: $\flat'_1 = \{x \mapsto o_1, y \mapsto o_2, z \mapsto o_3\}, \flat'_2 = \{x \mapsto o_2, y \mapsto o_1, z \mapsto o_3\}$. By applying $\flat'_1$, we can obtain a new proposition $p_3 = \mathsf{IsGreater}(o_1 * o_3, o_2 * o_3) = \mathsf{IsGreater}(15.2 * 10, 15.12 * 10)$ but with unknown truth value. Similarly, we obtain $p_4 = \mathsf{IsGreater}(o_2 * o_3, o_1 * o_3) = \mathsf{IsGreater}(15.12 * 10, 15.2 * 10) =$ Unknown by $\flat'_2$. Finally, we generate two propositional formulae: $\psi_1 = p_0 \wedge p_2 \Rightarrow p_3$ via $\flat'_1$ and $\psi_2 = p_1 \wedge p_2 \Rightarrow p_4$ via $\flat'_2$ for the following component.

3.2 Rule-like propositional formula transformation

Given a DeNF formula $\psi = D_1 \vee D_2 \vee \cdots \vee D_m \Rightarrow C_1 \wedge C_2 \wedge \cdots \wedge C_n$, where each D_i is a conjunction of literals $l^D_{i1} \wedge l^D_{i2} \wedge \cdots \wedge l^D_{im_i}$ and each C_j is a disjunction of literals $l^C_{j1} \vee l^C_{j2} \vee \cdots \vee l^C_{jn_j}$, the formula transformation module is designed to reformulate ψ into a set of propositional formulae Γ_ψ in a rule-like form, i.e., an implication, such that the LHS of each formula in Γ_ψ is a literal conjunction and the RHS is a single literal.

To achieve this, we first transform ψ to a formula set $\Gamma'_\psi = \{D_i \Rightarrow C_j \mid i \in [m], j \in [n]\}$. Then, for each single formula $\psi_{i,j} = D_i \Rightarrow C_j = l^D_{i1} \wedge l^D_{i2} \wedge \cdots \wedge l^D_{im_i} \Rightarrow l^C_{j1} \vee l^C_{j2} \vee \cdots \vee l^C_{jn_j}$, we reformulate it as the implicant formula set $\Gamma'_{\psi_{i,j}}$, following the idea of unit propagation [63]:

$$\Gamma'_{\psi_{i,j}} = \{l^D_{i1} \wedge l^D_{i2} \wedge \cdots \wedge l^D_{im_i} \wedge (\bigwedge_{k \in [n_j] \wedge k \neq t} \neg l^C_{jk}) \Rightarrow l^C_{jt} \mid t \in [n_j]\}$$

Finally, given a DeNF ψ, we obtain the corresponding reformulated rule-like propositional formula set $\Gamma_\psi = \bigcup_{i \in [m], j \in [n]} \Gamma'_{\psi_{i,j}}$, and given a DeNF set $\mathcal{R}$, we use $\Gamma_{\mathcal{R}}$ to denote the union of all rule-like form formulae transformed by each formula in $\mathcal{R}$, i.e., $\Gamma_{\mathcal{R}} = \bigcup_{\psi \in \mathcal{R}} \Gamma_{\psi}$. In the remainder of this paper, we use $\mathtt{Lit}_{\mathcal{R}}$ (resp. $\mathtt{Lit}_{\psi}$) to denote the set of all the literals that appeared in $\mathcal{R}$ (resp. ψ). The rules ψ_1 and ψ_2 obtained in Example 1 are already expressed in the rule-like form.

Example 2. Consider a DeNF $\psi = (a_1 \wedge b_1) \vee a_2 \Rightarrow (c_1 \vee d_1) \wedge c_2$. We first transform ψ into an equivalent formula set $\Gamma'_{\psi} = \{\psi_{1,1}, \psi_{1,2}, \psi_{2,1}, \psi_{2,2}\}$, where $\psi_{1,1} = a_1 \wedge b_1 \Rightarrow c_1 \vee d_1$, $\psi_{1,2} = a_1 \wedge b_1 \Rightarrow c_2$, $\psi_{2,1} = a_2 \Rightarrow c_1 \vee d_1$, and $\psi_{2,2} = a_2 \Rightarrow c_2$. For each single formula in Γ'_{ψ}, we then reformulate it as follows: $\Gamma'_{\psi_{1,1}} = \{a_1 \wedge b_1 \wedge \neg c_1 \Rightarrow d_1, a_1 \wedge b_1 \wedge \neg d_1 \Rightarrow c_1\}$, $\Gamma'_{\psi_{1,2}} = \{a_1 \wedge b_1 \Rightarrow c_1\}$, $\Gamma'_{\psi_{2,1}} = \{a_1 \wedge \neg c_1 \Rightarrow d_1, a_1 \wedge \neg d_1 \Rightarrow c_1\}$, $\Gamma'_{\psi_{2,2}} = \{a_2 \wedge c_2\}$, and finally obtain the set of rule-like propositional formulae as $\Gamma_{\psi} = \Gamma'_{\psi_{1,1}} \cup \Gamma'_{\psi_{1,2}} \cup \Gamma'_{\psi_{2,1}} \cup \Gamma'_{\psi_{2,2}}$.

3.3 Forward chaining

Given a rule set $\Gamma_{\mathcal{R}}$ obtained as above with corresponding literal set as $\mathtt{Lit}_{\mathcal{R}}$ and a subset of literals $\mathtt{Lit} \subseteq \mathtt{Lit}_{\mathcal{R}}$ with predetermined truth values, the forward chaining (FC) procedure is designed to: i) verify consistency, and ii) infer truth values for the undefined literal $l \in \mathtt{Lit}_{\mathcal{R}} \backslash \mathtt{Lit}$. The truth value inference of undefined literals is operated based on the inference rule of modus ponens [14], which is closely aligned with the traditional forward chaining algorithm [52] utilized in rule-based systems. However, compared to the traditional FC algorithm, which requires each rule to be strictly a Horn Clause, a significant distinction between our algorithm and the traditional one is: we require that all negative literals appearing in $\mathtt{Lit}_{\mathcal{R}}$ are also included in the forward chaining graph. Such an extension enables us to detect the inconsistency by identifying truth values for undefined literals. For instance, consider a rule set $\{a \wedge b \Rightarrow c\}$ and literals defined as $a = b = \mathsf{True}$ and $c = \mathsf{False}$. In this case, the truth value of literal c is *inconsistent* with the inference result derived from the rule set.

To achieve it, given a rule set $\Gamma_{\mathcal{R}}$ and an initially defined literals $\mathtt{Lit}$ with truth values, we first construct an initial graph G through the following steps:

- Step 1 (Literal Nodes): Create a node for each literal that appears in the rule set $\Gamma_{\mathcal{R}}$.
- Step 2 (LHS Node): Create an LHS node for each rule-like propositional formula in $\Gamma_{\mathcal{R}}$.
- Step 3 (Edges): For each formula in $\Gamma_{\mathcal{R}}$, such as $l_1 \wedge \cdots \wedge l_n \Rightarrow l_{n+1} \in \Gamma_{\mathcal{R}}$, add a directed edge from each literal node l_i ($i \in [n]$) to the corresponding LHS node, and add a directed edge from this LHS node to the literal node l_{n+1}.

Next, according to the initially defined literals, we mark all the literal nodes valued True as $\mathtt{Lit}^{\uparrow}$ and mark all the literal nodes valued False as $\mathtt{Lit}^{\downarrow}$, based on the truth value of literals from $\mathtt{Lit}$. Then, we perform forward chaining procedure on the graph G and update $\mathtt{Lit}^{\uparrow}$ and $\mathtt{Lit}^{\downarrow}$ as follows until no more update can be done or an inconsistency is detected by $\mathtt{Lit}^{\uparrow} \cap \mathtt{Lit}^{\downarrow} \neq \emptyset$:

- Update of $\mathtt{Lit}^{\uparrow}$ and $\mathtt{Lit}^{\downarrow}$: For each LHS node encoding the rule-like formula (or implication) $l_1 \wedge \cdots \wedge l_n \Rightarrow l_{n+1}$, if $\{l_1, \ldots, l_n\} \subseteq \mathtt{Lit}^{\uparrow}$, then we have $\mathtt{Lit}^{\uparrow} = \mathtt{Lit}^{\uparrow} \cup l_{n+1}$ and $\mathtt{Lit}^{\downarrow} = \mathtt{Lit}^{\downarrow} \cup \neg l_{n+1}$.

Example 3. We now illustrate the graph construction and the corresponding FC procedure for the rule set $\Gamma_{\psi} = \{\psi_1, \psi_2\}$ under the Level-2 interpretation given in Example 1. We first obtain all the literals as $\mathtt{Lit}_{\mathcal{R}} = \{p_0, p_1, p_2, p_3, p_4\}$ with $\mathtt{Lit}^{\uparrow} = \{p_0, p_2\}$ and $\mathtt{Lit}^{\downarrow} = \{p_1\}$, construct the literal nodes correspondingly, and construct the LHS node set as $\{p_0 \& p_2, p_1 \& p_2\}$. Then, directed edges are added as follows: $p_0 \to p_0 \& p_2$, $p_2 \to p_0 \& p_2$, $p_0 \& p_2 \to p_3$, $p_1 \to p_1 \& p_2$, $p_2 \to p_1 \& p_2$, $p_1 \& p_2 \to p_4$, resulting an initial graph. Next, we check the logic consistency. For the LHS node $p_0 \& p_2$, since $\{p_0, p_2\} \subseteq \mathtt{Lit}^{\uparrow}$, we update $\mathtt{Lit}^{\uparrow} = \mathtt{Lit}^{\uparrow} \cup p_3 = \{p_0, p_2, p_3\}$ and $\mathtt{Lit}^{\downarrow} = \mathtt{Lit}^{\downarrow} \cup \neg p_3 = \{p_1, \neg p_3\}$, where "$p_3 = \mathsf{True}$" is the new knowledge derived. For the LHS node $p_1 \& p_2$, since $p_1 \notin \mathtt{Lit}^{\uparrow}$, there is no update on $\mathtt{Lit}^{\uparrow}$ and $\mathtt{Lit}^{\uparrow}$, hence no new knowledge inferred.

3.4 Query generation

Once RvLLM obtains the newly inferred knowledge, the query generation module generates a concrete query to the target LLM related to the knowledge, and requires the LLM to analyze its truth value. For the inferred knowledge $p_3 = \mathsf{IsGreater}(151.2, 152) = \mathsf{True}$ in Example 3, we generate the corresponding query as "*Is 151.2 greater than 152?*", as shown in Figure 2. Since the target LLM returns False, an inconsistency is detected, indicating a diagnostic error in the target LLM.

Clarification. The soundness of this work—the ability to detect inconsistencies with respect to domain-specific constraints—depends on the accurate and faithful encoding of domain knowledge by the expert. Thus, the validity of RvLLM hinges on the alignment between the specification given in ESL and the underlying domain knowledge, and any misrepresentations in this encoding may compromise the soundness of the method.

4 Experiments

To evaluate the effectiveness of RvLLM, we apply it to the runtime verification of LLMs across three representative tasks that require domain-specific knowledge: violation detection against Singapore Rapid Transit Systems Act [51], numerical comparison, and inequality solving. For a comprehensive evaluation, we include both SOTA and non-SOTA LLMs as our benchmark, including Qwen 2.5 (max, plus, turbo, 7B, 14B, 32B, 72B), GPT (4.1, 4.1 mini, 4.1 nano), Gemini 2.0 (Flash, Flash Lite), and DeepSeek-V3. All experiments are conducted on a machine with an Intel (R) Xeon(R) w7-2475X processor. A dedicated guideline for designing interpretation abstraction prompts for the perception agent is provided in the appendices, along with additional experimental details and descriptions of the datasets and specifications used throughout the experiments.

To ensure a comprehensive evaluation, we also compare RvLLM with other runtime approaches, including SelfCheckGPT [47] and a prompt-based augmentation method that explicitly encodes domain-specific constraints within prompts. Due to the computational overhead and API costs associated with large-scale evaluations, we restrict our comparison to cost-efficient model variants. Additional experimental results and missing implementation details/justifications can be found in [64].

4.1 Case study 1: violation detection against Singapore Rapid Transit Systems Act

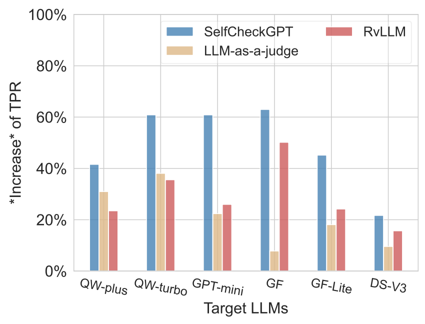
For this case study, we apply RvLLM under the Level-1 interpretation to evaluate the LLM's responses in detecting violations within contextual scenarios against the Singapore Rapid Transit Systems Act. In this capacity, RvLLM can be regarded as a complementary mechanism to enhance the LLM's capability in detecting the law violations in this study. We conduct experiments using contextual scenarios derived from [58], consisting of 304 cases in total–281 labeled as involving violations (unsafe) and 23 as not (safe). To incorporate relevant domain knowledge, we carefully encode 31 of the 53 regulations from Singapore Rapid Transit System Act into ESL specifications. It took a research scientist three days to interpret the law and develop corresponding specifications–a one-time effort applicable for verifying any LLM application against these regulations. In this study, the same model serves both as the target LLM and as the perception agent.

Results. Table 1 reports the violation detection results by various LLMs, with and without our framework as a complementary analysis method, where TPR and TNR denote the true positive rate and true negative rate for the violation detection, i.e., a violation is a positive example. The last column gives the average execution time for each scenario analysis. In the standalone LLM setting, a true positive is defined as a case where the model successfully identifies a violation in an unsafe scenario, and a true negative corresponds to a case where the model confirms the absence of violations in a safe scenario. In the combined LLM+RvLLM setting, a true positive is recorded if either the LLM or RvLLM detects a violation in an unsafe scenario, whereas a true negative is defined as a case that both the LLM and RvLLM determine the absence of any violation in a safe scenario. The results show that integrating RvLLM significantly enhances the TPR, improving the target LLM's capability to identify violations. However, we also observe that the integration of RvLLM can reduce the TNR in some models. This decline is primarily due to inaccuracies or incompleteness in the interpretation procedure by the perception agent. We also find that, among all evaluated models, DeepSeek-V3 achieves the highest TPR in both standalone and integrated settings, demonstrating superior detection capabilities in this domain-specific task and good performance on language understanding.

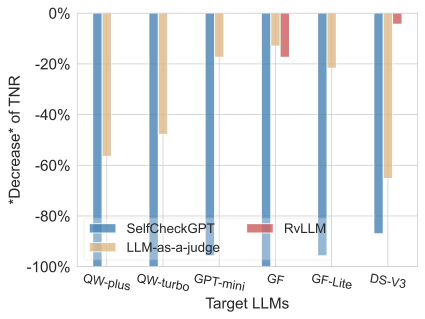
Figure 3 gives the comparison results of RvLLM against other baseline methods. We observe that SelfCheckGPT often yields the highest TPR gains, as it labels nearly all outputs as "hallucinations", resulting in near-zero TNR. This outcome is expected since SelfCheckGPT relies solely on output stability without leveraging domain knowledge, making it more suitable for open-domain generation. The prompt augmentation method (also known as LLM-as-a-judge) achieves TPR performance comparable to RvLLM but accompanied by a non-negligible TNR decrease. In contrast, RvLLM achieves a better balance between TPR and TNR, yielding more reliable and interpretable results.

Table 1: Violation detection results of LLMs with and without RvLLM against the Singapore Rapid Transit Systems Act [51]. All percentage values are reported rounded to one decimal place.

Target LLMs	LLM		LLM + RvLLM		Performance of RvLLM		Time (s)
	TPR	TNR	TPR	TNR	TPR	TNR	
Qwen max [62]	56.2%	91.3%	86.1%	91.3%	+29.9%	0	3.00
Qwen plus [62]	58.4%	1	81.9%	1	+23.5%	0	5.32
Qwen turbo [62]	39.1%	1	74.7%	1	+35.6%	0	1.64
Qwen 2.5 (7B) [62]	16.0%	1	61.2%	87.0%	+45.2%	-13.0%	2.28
Qwen 2.5 (14B) [62]	38.8%	95.7%	80.4%	87.0%	+41.6%	-8.7%	3.11
Qwen 2.5 (32B) [62]	56.6%	95.7%	87.5%	95.7%	+31.0%	0	2.55
Qwen 2.5 (72B) [62]	57.3%	1	80.4%	95.7%	+23.1%	-4.3%	2.57
GPT 4.1 [1]	57.7%	95.7%	81.1%	91.3%	+23.5%	-4.3%	3.11
GPT 4.1 mini [1]	39.1%	95.7%	65.1%	95.7%	+26.0%	0	3.94
GPT 4.1 nano [1]	11.7%	1	29.9%	1	+18.1%	0	1.82
Gemini 2.0 Flash [23]	37.0%	1	87.2%	82.6%	+50.2%	-17.4%	2.69
Gemini 2.0 Flash Lite [23]	54.8%	95.7%	79.0%	95.7%	+24.2%	0	1.72
DeepSeek-V3 [43]	**78.3%**	87.0%	**94.0%**	82.6%	+15.7%	-4.3%	15.84



(a) The **increase** of TPR across models.



(b) The **decrease** of TNR across models.

Figure 3: Performance comparison of different methods in the violation detection task, where QW, GF/GF-Lite, and DS-V3 denote Qwen, Gemini 2.0 Flash/Flash Lite, and DeepSeek-V3, respectively.

4.2 Case study 2: numerical comparison problems

For this case study, we apply RvLLM under Level-2 interpretation to verify the numerical comparison results produced by various LLMs. It is important to note that RvLLM is inherently designed without mathematical solving capabilities and relies solely on logical inference. Without loss of generality, we randomly generate 100 questions following the guideline for increasing the likelihood of incorrect comparison results by LLMs. The specification file used here is the one shown in Figure 2. Again, we use the same model to serve both the target LLM and the perception agent.

Results. Table 2 gives the verification results by RvLLM on numerical comparison tasks across various LLMs. Columns Con. and Incon. report the number of cases where RvLLM detects no inconsistencies and where at least one inconsistency is detected, following the process in Figure 2. The values in parentheses in Column Con. represent cases where the target LLM produces an erroneous comparison result on the original comparison question, yet RvLLM fails to identify any inconsistencies–typically due to incomplete interpretation by the perception agents. In contrast, the values in parentheses in Column Incon. corresponds to cases where the target LLM returns a correct comparison result on the original question, but RvLLM indeed detects some inconsistencies. This usually arises from two main reasons: i) incomplete interpretation, and ii) the target LLM producing an incorrect diagnosis on newly-inferred knowledge. Overall, RvLLM effectively identifies erroneous diagnoses by the target LLM in most cases, with exceptions observed in only 3 cases for GPT 4.1 nano and mini. Among all tested models, Gemini 2.0 Flash Lite and DeepSeek-V3 show the weakest diagnostic performance, and Qwen 2.5 (7B) has the worst language understanding capabilities, failing all interpretation abstraction. Table 3 presents the performance comparison across methods. Consistent with the findings in case study 1, we find that RvLLM achieves the best balance, maintaining both high true positive and high true negative rates among all evaluated approaches.

Table 2: Verification results by RvLLM over various LLMs for numerical comparison tasks.

Target LLMs	Con.	Incon.	Fail	Time (s)
Qwen max [62]	93	7	0	5.03
Qwen plus [62]	99	1	0	7.01
Qwen turbo [62]	94	6(2)	0	2.93
Qwen 2.5 (7B) [62]	0	0	100	N/A
Qwen 2.5 (14B) [62]	90	10(7)	0	5.09
Qwen 2.5 (32B) [62]	98	2	0	4.98
Qwen 2.5 (72B) [62]	96	4	0	6.31
GPT 4.1 [1]	98	2	0	4.84
GPT 4.1 mini [1]	92(1)	8	0	5.37
GPT 4.1 nano [1]	90(3)	10	0	3.73
Gemini 2.0 Flash [23]	93	7	0	4.94
Gemini 2.0 Flash Lite [23]	52	44	4	4.35
DeepSeek-V3 [43]	69	31(5)	0	30.87

Table 3: Performance comparison of different methods for numerical comparison tasks.

Target LLMs	Methods	TPR	TNR
Qwen plus [62]	SelfCheckGPT	0	96.7%
	LLM-as-a-judge	100%	0
	RvLLM	100%	100%
GPT 4.1 mini [1]	SelfCheckGPT	100%	0
	LLM-as-a-judge	0	100%
	RvLLM	88.9%	100%
Gemini 2.0 Flash [23]	SelfCheckGPT	87.5%	0
	LLM-as-a-judge	0	100%
	RvLLM	100%	100%

4.3 Case study 3: inequality solving problems

For this case study, we employ RvLLM under Level-1 interpretation to verify the step-by-step reasoning process of LLMs in inequality-solving tasks. We compile a dataset of 40 inequality questions sourced from A-Level H2 Mathematics examination papers [32] and carefully design three specifications to serve as domain knowledge that RvLLM uses for verification. These specifications consist of basic algebraic inequality properties taught at the junior college level, targeting three common LLM reasoning errors in inequality solving: incorrect factorization, flawed interval analysis, and omission of endpoint or critical point checks. Given the higher interpretation complexity in this case study and the strong language processing capabilities of DeepSeek-V3, we employ it as the perception agent for all LLMs evaluated and Qwen 2.5 (32B) as an alternative for comparison.

Results. Given the strong performance of RvLLM in detecting true positives, we focus the evaluation on the questions that the target LLMs answered incorrectly. The results are summarized in Table 4. Column 2 lists the total number of questions for which the target LLM produces incorrect solutions. Columns labeled DS-V3 and QW-32B (spanning Columns 3 to 11) show the number of cases where RvLLM successfully identifies inconsistencies based on the corresponding specifications using DeepSeek-V3 and Qwen 2.5 (32B) as the perception agent, respectively. Values in parentheses indicate false positives–cases where the detected inconsistency does not stem from an actual reasoning error but from inaccuracies of the perception agent during the interpretation abstraction process. Column GT gives the ground truth number of incorrect responses attributable to the rule specifications. It is important to note that we define only three specifications as domain knowledge for inequality-solving tasks, which is insufficient to cover all reasoning patterns required for such problems comprehensively. As a result, it is expected that RvLLM fails to detect a portion of the errors due to the lack of knowledge. The final two columns give the true positive rate of RvLLM with three specifications. We find that RvLLM achieves a TPR of up to 50% when using DeepSeek-V3 as the perception agent. This result underscores both RvLLM's potential to detect reasoning errors—even with limited domain knowledge—and the critical role of a robust perception agent in boosting overall performance. Although TPR remains below 50% for most models, they rise markedly when restricted to ground truth cases (Column GT), suggesting substantial gains are possible with richer domain knowledge in inequality solving.

5 Related work

LLM testing. Research in this area aims to develop comprehensive benchmarks and systematic evaluation frameworks to assess LLM across general attributes such as robustness, fairness, and factual consistency [45, 41, 57]. Robustness studies examine model resilience to adversarial input, paraphrasing, or noise [9, 56, 66], while fairness evaluations assess bias with respect to sensitive attributes like gender or race [39, 66]. Factual consistency detection, or hallucination detection, focuses on identifying instances in which LLMs produce plausible yet factually incorrect or unsupported claims [42, 38, 40]. Although large-scale benchmarks like HELM [41] and BIG-bench [54] offer task suites and metrics for performance evaluation, a critical challenge remains—the dynamic

Table 4: Verification results by RvLLM in inequality solving tasks with three specifications, defining rules related to factorization, interval analysis, and endpoints/critical points checking.

Target LLMs	Incorrect	Factorization Error			Interval Error			Endpoints Error			TPR	
		DS-V3	QW-32B	GT	DS-V3	QW-32B	GT	DS-V3	QW-32B	GT	DS-V3	QW-32B
Qwen max [62]	12	1	1	1	1	0	1	2	2	3	33.3%	25%
Qwen plus [62]	10	1	1	2	0	0	0	0	1	2	10%	20%
Qwen turbo [62]	22	0	0	1	1	0	1	3	2	4	18.2%	9.1%
Qwen 2.5 (7B) [62]	25	2	1	3	7	2	7	3	2	5	48%	20%
Qwen 2.5 (14B) [62]	18	0	0	1	3	2(1)	3	3(1)	2	3	27.8%	16.7%
Qwen 2.5 (32B) [62]	16	1	0	1	3	2	4	4	2	5	50%	25%
Qwen 2.5 (72B) [62]	15	1	0	1	4	3	4	0	0	1	33.3%	20%
GPT 4.1 [1]	5	1	1(1)	1	0	0	0	0	0	0	20%	0
GPT 4.1 mini [1]	2	0	0	0	0	0	0	1	0	1	50%	0
GPT 4.1 nano [1]	10	0	0	0	1	0	1	1	0	1	20%	0
Gemini 2.0 Flash [23]	5	0	0	0	1	0	1	0	0	0	20%	0
Gemini 2.0 Flash Lite [23]	6	1	1	2	0	0	0	0	0	1	16.7%	16.7%

and open-ended nature of real-world deployments complicates the construction of exhaustive and representative test suites.

Runtime verification of LLMs. Runtime verification [6, 21] is a dynamic analysis technique that ensures certified system behavior during execution by monitoring traces against formal specifications. Owing to its lightweight nature, runtime verification has inspired multiple efforts to verify LLM-generated outputs at runtime [13, 7, 47, 11, 37, 46, 36]. However, existing approaches center on open-domain settings, where specifications are often ambiguous and informal. For instance, SelfCheckGPT [47] and CHECKEMBED [7] operationalize specifications in terms of output stability. More recently, [12] proposes a fairness monitoring approach leveraging precisely defined properties in formal specifications. These specifications, informed by linear temporal logic [50] and its bounded metric variant [2], signal a shift towards formal methods for more accurate and dependable runtime verification of LLM behaviors. However, these works primarily focus on general properties and are not well-suited to domain-specific constraints for specialized tasks.

Expert systems with NLP. Rule-based expert systems [15, 16, 18] have effectively tackled domain-specific problems by coupling knowledge bases with inference engines to emulate expert reasoning for decades of years. To enhance usability, in the 70s, researchers integrated natural language processing (NLP), enabling interaction via simplified language, e.g., SHRDLU [60] in virtual environments and MedLEE [35] in clinical text analysis. These systems relied on symbolic parsing and controlled vocabularies but suffered from fragile parsing, limited lexical coverage, and ambiguity, often requiring constrained input or extensive user guidance. Nevertheless, integrating NLP with rule-based reasoning provides a key proof-of-concept for interactive, user-friendly AI, shaping later knowledge-based systems and natural language interfaces. Recently, the rise of LLMs has reinvigorated this paradigm. In this work, we employ a perception LLM to translate the target LLM's context and outputs into formal representations for backend reasoning. An improved version could explore other extensions of Horn clauses beyond those considered here to enhance expressiveness and completeness [4].

6 Conclusion

In this work, we present the first runtime verification framework RvLLM for LLMs, which allows the incorporation of domain knowledge. To support this, we design a general specification language, ESL, which enables domain experts to formally encode constraints in a lightweight yet expressive manner, supporting the rigorous verification of LLM behavior. We implement the proposed framework as a tool and evaluate its effectiveness on three representative tasks where domain knowledge plays a critical role. For each task, we develop tailored specifications grounded in relevant domain expertise. Experimental results indicate that RvLLM effectively leverages expert-defined domain knowledge to enable precise runtime verification of LLMs and strike a good balance between TPR and TNR, compared to existing runtime methods. However, despite its good performance, the current framework suffers from limited expressiveness by ESL, restricting its ability to encode more complex domain knowledge and behavioral constraints for LLMs. Future work will focus on enriching the specification language with additional operators to improve its expressiveness and generalization.

7 Acknowledgments

This work was partially funded by the Ministry of Education, Singapore, under its Academic Research Fund Tier 3 (MOET32020-0003). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

References

[1] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)

[2] Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. Journal of the ACM (JACM) **43**(1), 116–146 (1996)

[3] Armstrong, K.: Chatgpt: Us lawyer admits using ai for case research. `https://www.bbc.com/news/world-us-canada-65735769` (2023)

[4] Babka, M., Balyo, T., Čepek, O., Gurskỳ, Š., Kučera, P., Vlček, V.: Complexity issues related to propagation completeness. Artificial Intelligence **203**, 19–34 (2013)

[5] Banerjee, S., Agarwal, A., Singla, S.: Llms will always hallucinate, and we need to live with this. arXiv preprint arXiv:2409.05746 (2024)

[6] Bartocci, E., Falcone, Y., Francalanza, A., Reger, G.: Introduction to runtime verification. Lectures on Runtime Verification: Introductory and Advanced Topics pp. 1–33 (2018)

[7] Besta, M., Paleari, L., Kubicek, A., Nyczyk, P., Gerstenberger, R., Iff, P., Lehmann, T., Niewiadomski, H., Hoefler, T.: Checkembed: Effective verification of llm solutions to open-ended tasks. arXiv preprint arXiv:2406.02524 (2024)

[8] Bunel, R., Lu, J., Turkaslan, I., Torr, P.H.S., Kohli, P., Kumar, M.P.: Branch and bound for piecewise linear neural network verification. J. Mach. Learn. Res. **21**, 42:1–42:39 (2020)

[9] Cai, R., Song, Z., Guan, D., Chen, Z., Li, Y., Luo, X., Yi, C., Kot, A.: Benchlmm: Benchmarking cross-style visual capability of large multimodal models. In: European Conference on Computer Vision. pp. 340–358. Springer (2024)

[10] Cascella, M., Montomoli, J., Bellini, V., Bignami, E.: Evaluating the feasibility of chatgpt in healthcare: an analysis of multiple clinical and research scenarios. Journal of medical systems **47**(1), 33 (2023)

[11] Chen, J., Kim, G., Sriram, A., Durrett, G., Choi, E.: Complex claim verification with evidence retrieved in the wild. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). pp. 3569–3587 (2024)

[12] Cheng, C.H., Wu, C., Ruess, H., Zhao, X., Bensalem, S.: Formal specification, assessment, and enforcement of fairness for generative ais. arXiv preprint arXiv:2404.16663 (2024)

[13] Cohen, R., Hamri, M., Geva, M., Globerson, A.: Lm vs lm: Detecting factual errors via cross examination. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 12621–12640 (2023)

[14] Copi, I.M., Cohen, C., McMahon, K.: Introduction to logic. Routledge (2016)

[15] Daniel, M., Hájek, P., Nguyen, P.H.: Cadiag-2 and mycin-like systems. Artificial Intelligence in Medicine **9**(3), 241–259 (1997)

[16] Denning, P.J.: Towards a science of expert systems. IEEE Intelligent Systems **1**(02), 80–83 (1986)

[17] Drummond, N., Shearer, R.: The open world assumption. In: eSI workshop: the closed world of databases meets the open world of the semantic web. vol. 15, p. 1 (2006)

[18] Durkin, J.: Expert systems: a view of the field. IEEE Intelligent Systems **11**(02), 56–63 (1996)

[19] Elboher, Y.Y., Gottschlich, J., Katz, G.: An abstraction-based framework for neural network verification. In: Proceedings of the 32nd International Conference on Computer Aided Verification. pp. 43–65 (2020)

[20] Enderton, H.B.: A mathematical introduction to logic. Elsevier (2001)

[21] Falcone, Y., Havelund, K., Reger, G.: A tutorial on runtime verification. Engineering dependable software systems pp. 141–175 (2013)

[22] Forsyth, R.: Expert systems: principles and case studies. Chapman & Hall, Ltd. (1984)

[23] Gemini, T.: Gemini: A family of highly capable multimodal models (2024), `https://arxiv.org/abs/2312.11805`

[24] Guha, N., Nyarko, J., Ho, D., Ré, C., Chilton, A., Chohlas-Wood, A., Peters, A., Waldon, B., Rockmore, D., Zambrano, D., et al.: Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models. Advances in Neural Information Processing Systems **36**, 44123–44279 (2023)

[25] Hayes-Roth, F., Waterman, D.A., Lenat, D.B.: Building expert systems. Addison-Wesley Longman Publishing Co., Inc. (1983)

[26] Hein, J.L.: Discrete mathematics. Jones & Bartlett Learning (2003)

[27] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., Steinhardt, J.: Measuring massive multitask language understanding. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021 (2021)

[28] Hernández-Orallo, J.: The measure of all minds: evaluating natural and artificial intelligence. Cambridge University Press (2017)

[29] Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: Proceedings of the 29th International Conference on Computer Aided Verification. pp. 3–29 (2017)

[30] Huang, Y., Bai, Y., Zhu, Z., Zhang, J., Zhang, J., Su, T., Liu, J., Lv, C., Zhang, Y., Fu, Y., et al.: C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. Advances in Neural Information Processing Systems **36** (2024)

[31] Jackson, P.: Introduction to expert systems (1990)

[32] Junior college test papers in level: A-level. `https://www.testpapersfree.com/junior-college/alevel/` (2024)

[33] Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., et al.: Chatgpt for good? on opportunities and challenges of large language models for education. Learning and individual differences **103**, 102274 (2023)

[34] Katz, G., Barrett, C.W., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks. In: Proceedings of the 29th International Conference on Computer Aided Verification. pp. 97–117 (2017)

[35] Krauthammer, M., Hripcsak, G.: A knowledge model for the interpretation and visualization of nlp-parsed discharged summaries. In: Proceedings of the AMIA Symposium. p. 339 (2001)

[36] Laban, P., Kryściński, W., Agarwal, D., Fabbri, A.R., Xiong, C., Joty, S., Wu, C.S.: Llms as factual reasoners: Insights from existing benchmarks and beyond. arXiv preprint arXiv:2305.14540 (2023)

[37] Laban, P., Schnabel, T., Bennett, P.N., Hearst, M.A.: Summac: Re-visiting nli-based models for inconsistency detection in summarization. Transactions of the Association for Computational Linguistics **10**, 163–177 (2022)

[38] Li, J., Cheng, X., Zhao, W.X., Nie, J.Y., Wen, J.R.: Halueval: A large-scale hallucination evaluation benchmark for large language models. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 6449–6464 (2023)

[39] Li, M., Li, L., Yin, Y., Ahmed, M., Liu, Z., Liu, Q.: Red teaming visual language models. In: Findings of the Association for Computational Linguistics ACL 2024. pp. 3326–3342 (2024)

[40] Li, N., Li, Y., Liu, Y., Shi, L., Wang, K., Wang, H.: Drowzee: Metamorphic testing for fact-conflicting hallucination detection in large language models. Proceedings of the ACM on Programming Languages **8**(OOPSLA2), 1843–1872 (2024)

[41] Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., et al.: Holistic evaluation of language models. arXiv preprint arXiv:2211.09110 (2022)

[42] Lin, S., Hilton, J., Evans, O.: Truthfulqa: Measuring how models mimic human falsehoods. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 3214–3252 (2022)

[43] Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al.: Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437 (2024)

[44] Liu, J., Xia, C.S., Wang, Y., Zhang, L.: Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. Advances in Neural Information Processing Systems **36**, 21558–21572 (2023)

[45] Liu, Y., Yao, Y., Ton, J.F., Zhang, X., Guo, R., Cheng, H., Klochkov, Y., Taufiq, M.F., Li, H.: Trustworthy llms: a survey and guideline for evaluating large language models' alignment. arXiv preprint arXiv:2308.05374 (2023)

[46] Luo, Z., Xie, Q., Ananiadou, S.: Chatgpt as a factual inconsistency evaluator for text summarization. arXiv preprint arXiv:2303.15621 (2023)

[47] Manakul, P., Liusie, A., Gales, M.: Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 9004–9017 (2023)

[48] Narodytska, N., Kasiviswanathan, S.P., Ryzhyk, L., Sagiv, M., Walsh, T.: Verifying properties of binarized deep neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 6615–6624 (2018)

[49] Payne, K.: An ai chatbot pushed a teen to kill himself, a lawsuit against its creator alleges. `https://apnews.com/article/chatbot-ai-lawsuit-suicide-teen-artificial-intelligence-9d48adc572100822fdbc3c90d1456bd0` (2024)

[50] Pnueli, A.: The temporal logic of programs. In: 18th annual symposium on foundations of computer science (sfcs 1977). pp. 46–57. ieee (1977)

[51] Rapid transit systems regulations (revised edition). `https://sso.agc.gov.sg/SL/RTSA1995-RG1` (1997)

[52] Russell, S.J., Norvig, P.: Artificial intelligence: a modern approach. pearson (2016)

[53] Shoenfield, J.R.: Mathematical logic. AK Peters/CRC Press (2018)

[54] Srivastava, A., Rastogi, A., Rao, A., Shoeb, A.A.M., Abid, A., Fisch, A., Brown, A.R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al.: Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. arXiv preprint arXiv:2206.04615 (2022)

[55] Thirunavukarasu, A.J., Ting, D.S.J., Elangovan, K., Gutierrez, L., Tan, T.F., Ting, D.S.W.: Large language models in medicine. Nature medicine **29**(8), 1930–1940 (2023)

[56] Tu, H., Cui, C., Wang, Z., Zhou, Y., Zhao, B., Han, J., Zhou, W., Yao, H., Xie, C.: How many unicorns are in this image? a safety evaluation benchmark for vision llms. arXiv preprint arXiv:2311.16101 (2023)

[57] Wang, B., Chen, W., Pei, H., Xie, C., Kang, M., Zhang, C., Xu, C., Xiong, Z., Dutta, R., Schaeffer, R., et al.: Decodingtrust: A comprehensive assessment of trustworthiness in gpt models. In: NeurIPS (2023)

[58] Wang, H., Zhang, A., Duy Tai, N., Sun, J., Chua, T.S., et al.: Ali-agent: Assessing llms' alignment with human values via agent-based evaluation. Advances in Neural Information Processing Systems **37**, 99040–99088 (2024)

[59] Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.J., Kolter, J.Z.: Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. Advances in Neural Information Processing Systems **34** (2021)

[60] Winograd, T.: Procedures as a representation for data in a computer program for understanding natural language. Tech. rep. (1971)

[61] Xu, Z., Jain, S., Kankanhalli, M.: Hallucination is inevitable: An innate limitation of large language models. arXiv preprint arXiv:2401.11817 (2024)

[62] Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al.: Qwen2. 5 technical report. arXiv preprint arXiv:2412.15115 (2024)

[63] Zhang, H., Stickely, M.E.: An efficient algorithm for unit propagation. Proc. of AI-MATH **96** (1996)

[64] Zhang, Y., Emma, S.Y., En, A.L.J., Dong, J.S.: Rvllm: Llm runtime verification with domain knowledge. arXiv preprint arXiv:2505.18585 (2025)

[65] Zhang, Y., Zhao, Z., Chen, G., Song, F., Chen, T.: BDD4BNN: A bdd-based quantitative analysis framework for binarized neural networks. In: Proceedings of the 33rd International Conference on Computer Aided Verification (CAV). pp. 175–200 (2021). https://doi.org/10.1007/978-3-030-81685-8_8

[66] Zhang, Y., Huang, Y., Sun, Y., Liu, C., Zhao, Z., Fang, Z., Wang, Y., Chen, H., Yang, X., Wei, X., et al.: Multitrust: A comprehensive benchmark towards trustworthy multimodal large language models. Advances in Neural Information Processing Systems **37**, 49279–49383 (2024)

[67] Zhong, W., Cui, R., Guo, Y., Liang, Y., Lu, S., Wang, Y., Saied, A., Chen, W., Duan, N.: Agieval: A human-centric benchmark for evaluating foundation models. In: Findings of the Association for Computational Linguistics: NAACL 2024. pp. 2299–2314 (2024)

[68] Zhou, K., Zhu, Y., Chen, Z., Chen, W., Zhao, W.X., Chen, X., Lin, Y., Wen, J.R., Han, J.: Don't make your llm an evaluation benchmark cheater. arXiv preprint arXiv:2311.01964 (2023)

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We made the claim that: i) we introduce a novel specification language ESL for experts to encode their domain-expertise into specification rule for checking the LLM behavior (design details are given in Section 2), and ii) design an innovative runtime verification framework RvLLM for LLMs with domain knowledge encoded in ESL (design details are given in Section 3). The experimental results are shown in Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: We do not include theoretical results in this paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We disclose detailed information for reproducing the main experimental results in the long-version preprint [64], provided as a separate file.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code and data will be open-sourced and made available on our website once they are properly organized. We also upload the preliminary version (of code) as a zip file as supplemental materials in the submission.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental settings and implementation details are described in Section 4 and in the long-version preprint [64], provided as a separate PDF.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For all main experiments, we executed RvLLM multiple times (at least twice) to assess potential variations in the results. Given that the proposed method employs a fixed framework with deterministic reasoning, any observed variability can only originate from the perception agent's outputs. Our analysis confirms that such variability is minimal in practice, to the extent that it can be considered negligible. We include such claim in the appendices in the long-version preprint [64].

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We discuss computing resources in Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: The potential societal impacts are discussed in appendices in the long-version preprint [64].

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not pose such risks in this paper.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: For all the data used in this work, we give the citation properly.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA] .

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We have describe our usage of LLMs throughout the experiments part in Section 4 as well as appendices in the long-version preprint [64].

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.