Why the Robot Said No: Knowledge-Graph-Grounded Explanations for LLM-Based Robot Planning

Nur Keleşoğlu[1,*], Łukasz Sobczak[1]

[1]Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Bałtycka 5, 44-100, Gliwice, Poland

Abstract

Large language models are increasingly used as high level planners in robotics, but when such a system refuses or misinterprets an instruction it cannot say why in terms a human can verify. Existing reliability methods compress grounding into a scalar confidence score and ask the model to certify its own grounding, creating a circular arrangement that, we find, produces fabricated justifications. We instead treat grounding reliability as a property of query evaluation over a knowledge graph. The environment is compiled into an RDF scene graph; a language model performs a single map-blind extraction of the entities an instruction refers to, and graph queries then adjudicate them: hallucination appears as an empty answer set, semantic conflict as an empty conjunction of individually satisfiable patterns, and ambiguity as a non-unique solution mapping. Each signal carries the triples that produced it, so every decision is explained by the computation that made it. On TRUST-NAV the symbolic verifier reaches 99.5% decision accuracy against 92.7% for the strongest conventional LLM baseline, and the same machinery makes the faithfulness of free-text explanations symbolically checkable.

Keywords

knowledge graphs, explainable AI, semantic grounding, SPARQL, LLM-based robot planning

1. Introduction

Large Language Models (LLMs) are increasingly used as decision-making components in embodied systems, where they translate open-ended natural language instructions into executable robot behaviour. Recent work has applied them to language-guided navigation [1, 2, 3], embodied reasoning [4] and task planning [5, 6, 7, 8, 9]. As these systems move into human-centred environments, however, a robot that refuses or misinterprets an instruction must be able to explain *why* in terms that a human operator can inspect and verify. This requires treating trustworthy planning not only as a plan-generation problem, but also as a *grounding verification* problem.

Existing approaches improve planning reliability through uncertainty estimation or safety assessment. KnowNo calibrates predictive uncertainty using conformal prediction [10]; introspective planning asks the model to assess its own uncertainty [11]; SafeAgentBench and MADRA evaluate instruction safety prior to execution [12, 13]. These methods, however, compress grounding reliability into scalar confidence scores. Such scores provide little insight into which entity was missing, which constraint was violated, or why a decision was made. More fundamentally, they rely on self-assessment: the same model whose grounding may be incorrect is asked to justify it. As we show experimentally, incorrect decisions are often accompanied by fabricated scene facts.

Our key observation is that the required external reference already exists in the environment representation itself, provided it is expressed as a knowledge graph. When the environment is modelled as a Resource Description Framework (RDF) graph queried with SPARQL [14], the three canonical grounding failures become observable properties of graph-query evaluation:

- **Hallucination:** an instruction refers to an entity type or room that has no counterpart in the graph, producing an empty answer set.

XAI-KG@ISWC2026 - 2nd International Workshop on eXplainable AI and Knowledge Graphs, co-located with ISWC 2026, October 25-29, 2026, Bari, Italy

*Corresponding author.

nkelesoglu@iitis.pl (N. Keleşoğlu); lsobczak@iitis.pl (Ł. Sobczak)

0000-0002-0306-7281 (N. Keleşoğlu); 0000-0001-9439-1812 (Ł. Sobczak)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-07

CEUR-WS.org/Vol-4318/paper8.pdf

- **Semantic conflict:** the referenced entities exist individually, but the required relation between them does not, producing an empty conjunctive pattern.
- **Ambiguity:** multiple graph bindings satisfy all constraints of the instruction, yielding a non-unique solution mapping.

These signatures are deterministic and constructive: they identify the specific graph evidence responsible for each decision, making the explanation identical to the reasoning process rather than a post-hoc rationalisation.

Building on this observation, we propose a knowledge-graph-grounded architecture for trustworthy LLM-based robot planning. The environment is compiled into an RDF scene graph, while an LLM performs a single *map-blind* extraction of target mentions without observing the environment itself. Graph queries then verify these mentions against the RDF data, producing *execute*, *clarify*, or *reject* decisions together with the supporting or violated graph patterns. The division of labour is deliberate: **the language model interprets, but the graph adjudicates.**

We evaluate the framework on TRUST-NAV [15], a publicly available benchmark of 206 natural-language instructions spanning standard navigation tasks and three families of risk-inducing scenarios, introduced in our earlier work [16]. Graph-grounded verification reaches 99.5% decision accuracy with perfect scores on all three risk categories, compared with 92.7% for the strongest conventional LLM planner and 85.4% for an LLM-scored variant. Because graph-based risk signals are binary and evidence-backed, they require no threshold calibration and expose even benchmark annotation errors through checkable graph witnesses.

The main contributions of this work are as follows:

1. We compile a smart home environment into an RDF scene graph while keeping lexical knowledge outside the graph, preventing graph level matching from silently repairing unsupported instructions.
2. We reformulate grounding risk estimation as graph query semantics, mapping hallucination, semantic conflict and ambiguity onto empty answer sets, empty conjunctions of satisfiable patterns, and non-unique solution mappings.
3. We introduce ante-hoc explanations in which the supporting and violated graph patterns responsible for a decision are returned directly as its explanation.
4. We make explanation faithfulness measurable by converting LLM free-text risk flags into typed claims that are verified symbolically against the knowledge graph.
5. We analyse the contribution of each symbolic component through ablation and show how graph evidence improves decision reliability while remaining fully traceable.

The remainder of this paper is organised as follows. Section 2 reviews related work on knowledge graphs for explainability, semantic validation, and LLM-based planning. Section 3 presents the scene knowledge graph and the proposed architecture. Section 4 describes the experimental setup and reports the results. Section 5 concludes the paper and discusses its limitations.

2. Related Work

This work sits at the intersection of three lines of research: knowledge graphs as vehicles for explanation, semantic validation of structured knowledge, and the grounding of LLM-based planners. We review each in turn and then position our contribution.

2.1. Knowledge Graphs for Explainable AI

The use of structured knowledge to make opaque models intelligible is a well-established programme within explainable AI. Tiddi and Schlobach [17] survey the field and distinguish between using knowledge graphs to *generate* explanations, to *enrich* them with background knowledge, and to *evaluate* them;

Lécué [18] argues that the semantics carried by knowledge graphs is what allows explanations to be both machine-processable and human-interpretable. Concrete instantiations include ontology-driven surrogate models, where domain concepts replace raw features as the vocabulary of explanation, an approach shown in controlled user studies to yield measurably more understandable explanations [19, 20]. A parallel line explains predictions made *over* knowledge graphs, for instance through levelwise filtering and graph summarisation of link predictions [21], and has begun to address the harder question of how such explanations should be evaluated [22, 23].

Our setting inverts the usual direction in two respects. First, rather than explaining a model by projecting its behaviour onto a knowledge graph after the fact, we place the graph *upstream* of the decision, so that the evidence retrieved during verification is identical to the evidence presented as explanation; the explanation is not an approximation of the decision procedure, it is the decision procedure made visible. Second, where LP-DIXIT [22] enlists a language model to judge the quality of graph-derived explanations, we enlist the graph to judge the quality of language-model-derived ones: free-text risk flags are decomposed into typed claims and checked against the triples they implicitly assert. Explanation quality thereby becomes a symbolically decidable property rather than a matter of human judgement.

2.2. Ontologies, Constraints and Semantic Validation

Establishing that a description is inconsistent with a schema is a classical concern of knowledge representation. OWL reasoning supports satisfiability and consistency checking, while SHACL [24] validates graph data against declarative shapes and returns machine-readable reports identifying the focus node and the violated constraint. Provenance vocabularies such as PROV-O [25] offer complementary means of recording how a conclusion was reached. Our verifier occupies a deliberately restricted point in this design space, with conflicts detected as empty conjunctive graph patterns rather than through a dedicated constraint language, and two of the commitments behind that restriction deserve comment, since both run against default Semantic Web practice.

First, we read the scene graph under a closed-world assumption: a type or room absent from the graph is treated as non-existent rather than as unknown. This departs from the open-world semantics of OWL, and the departure is essential rather than incidental. The graph is a complete map of a bounded indoor environment produced by the robot's own mapping pipeline, not an open fragment of the Web; under an open-world reading, hallucination could not be detected at all, because every missing referent would merely be unrecorded rather than absent. Trustworthy grounding in a closed environment requires the ability to conclude that something is not there.

Second, we keep lexical knowledge outside the graph rather than encoding it as subsumption. It would be natural to relate *television* to *tv*, or *table* to its subtypes, through a class hierarchy and let entailment resolve the instruction. We deliberately do not: an ontology that makes aliases indistinguishable at query time also erases the distinction between a legitimate alias and an unsupported reference, and that distinction is exactly what a trustworthy system must preserve. Inference that silently repairs an ill-posed instruction defeats the purpose of verifying it. The lexical layer is therefore a separate, inspectable component whose contribution we measure independently (Section 4.4).

For smart home environments, mature vocabularies already exist, including DogOnt [26], SAREF [27] for smart appliances, and the Building Topology Ontology [28] for spatial containment. Our schema is intentionally minimal and is not currently aligned to them; aligning it would make the resulting graph reusable beyond our benchmark and let deployment-specific constraints be expressed as SHACL shapes over standard vocabulary, which we regard as the natural next step rather than as an accomplished contribution.

2.3. Knowledge and Scene Graphs for Robot Planning

Semantic mapping augments metric maps with object, region and relational information. Early systems combined SLAM with object detection to label observed scenes [29]; subsequent work pursued

compact structured representations [30] and semantic organisation supporting consistent indoor navigation [31]. More recently, open-vocabulary 3D scene graphs have exposed object-level spatial relations directly to language models: ConceptGraphs constructs such graphs for perception and planning [32], while SayPlan uses hierarchical 3D scene graphs to make large-scale environments tractable for LLM planners [33].

These systems use the graph as a *search space*: a structure to be traversed in order to find a plan, on the assumption that a plan is what should be produced. We use it as a *verification and explanation substrate*: a structure against which a candidate grounding is tested before planning is permitted, and from which the justification of that test is extracted. The representational choice follows from this. Bespoke scene-graph structures are adequate for traversal, but expressing the environment in RDF gives us a standard query semantics in which the three failure modes have uniform, declarative characterisations, and a clear path towards reusing established domain vocabularies and constraint languages. It also makes the evidence attached to a decision a set of triples that any external tool can re-check, rather than an artefact of one system's internal data structures.

2.4. LLM-Based Task Planning in Robotics

LLMs have become popular as high-level planners that decompose tasks into executable steps. Early prompting studies showed that pre-trained models produce usable zero-shot plans for embodied agents [4]. SayCan grounds the model in robot affordances by scoring candidate actions for both goal relevance and feasibility [5]; Inner Monologue injects environment and execution feedback into the prompt [6]; ProgPrompt and Code as Policies elicit structured programs or executable policy code [7, 8]; and PaLM-E integrates perception and language in a single model [9]. Splitting high-level task planning from low-level motion control is itself a long-standing design principle [34]. Across this literature, however, the instruction is generally assumed to be well-posed and its referents assumed to exist; the question is how to plan, not whether planning is warranted.

2.5. Uncertainty-Aware and Trustworthy LLM Planning

The reliability of LLM planners has become a research topic in its own right. KnowNo applies conformal prediction to calibrate uncertainty and trigger help-seeking when the prediction set remains ambiguous [10]. Introspective planning instead prompts the model to evaluate its own uncertainty and choose between acting and asking [11]. SafeAgentBench shows that embodied agents frequently accept unsafe commands [12], and MADRA introduces a training-free debate module that assesses instruction safety prior to execution [13].

We share the objective but differ in kind rather than degree. These methods produce a *probabilistic estimate* of reliability, derived from the planner's own internal signals; we produce a *symbolic justification*, derived from an external, inspectable knowledge graph. The practical consequences are threefold. First, the decision is auditable: a rejection can be traced to a named relation between named nodes, and an operator who disagrees can contest the graph rather than the model. We find direct empirical support for why this matters. When the LLM assessor reaches a wrong decision, it also fabricates the scene fact that justifies it, so a system whose explanations come from the same model they are meant to explain cannot be audited at all (Section 4.3). Second, the decision is deterministic given the graph and the extracted mentions. Identical inputs therefore yield identical outcomes and identical explanations, and no threshold calibration is required. Third, the failure surface is relocated: errors now arise from an incomplete or incorrect knowledge graph rather than from unquantifiable model behaviour. This is a substantially more tractable form of error, since a graph can be inspected, corrected, and versioned in a way that model behaviour cannot.

3. Methodology

We propose a risk-aware semantic grounding framework in which a scene knowledge graph acts both as the *verifier* of a natural-language instruction before any plan is generated and as the *source of the explanation* attached to every decision. The framework decides, for each instruction, whether the robot should `execute` it, `clarify` it with the user, or `reject` it, and it justifies non-execution decisions with evidence triples drawn from the graph.

3.1. Problem Formulation

Let $\mathcal{M} = (\mathcal{O}, \mathcal{R})$ be the semantic map of an indoor environment, where each object $o_i \in \mathcal{O}$ is a tuple $(id_i, type_i, pos_i, props_i)$ and each room $r_j \in \mathcal{R}$ is a tuple $(id_j, type_j, O_j)$ with $O_j \subseteq \mathcal{O}$. Given a natural-language instruction q and a start position p_0, the system must output a decision $d \in \{\texttt{execute}, \texttt{clarify}, \texttt{reject}\}$, a navigation plan π (an ordered list of object identifiers, non-empty only for `execute`), and an explanation α. We model grounding reliability along three interpretable risk dimensions: *ambiguity* s_a, *hallucination* s_h, and *semantic conflict* s_c, each in $[0, 1]$.

3.2. Scene Knowledge Graph

The semantic map is produced by the robot perception pipeline and contains all recognised objects together with their room assignments and semantic attributes. The map $\mathcal{M}$ is compiled into an RDF graph $\mathcal{G}$ (181 triples for our environment, 36 object instances, 8 rooms, 12 predicates, 14 object classes). The compilation is a deterministic one-to-one mapping from the semantic map to RDF. Each object instance becomes an RDF resource identified by its map identifier, object categories become RDF classes, room membership is translated into home:locatedIn relations, and all object attributes are represented as datatype properties. The compilation introduces no inferred semantic entities or relations; every triple corresponds directly to information already present in the semantic map.

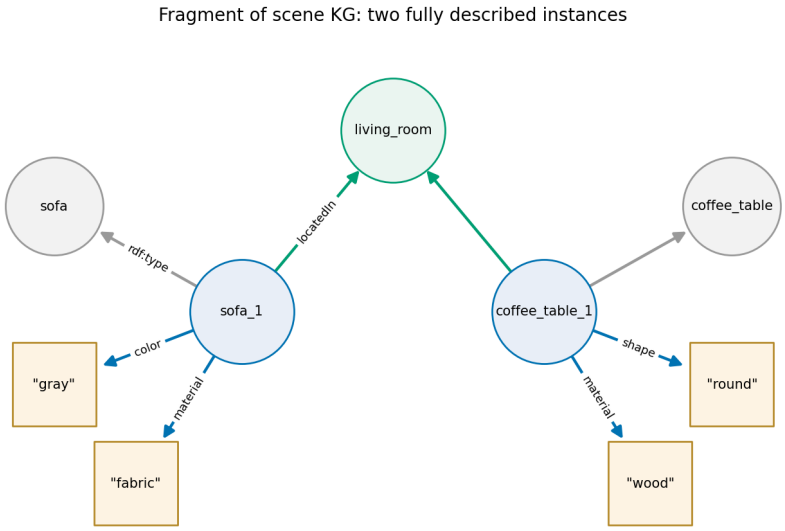


Figure 1: Excerpt of the scene knowledge graph $\mathcal{G}$: the complete description of two instances. Instance nodes (blue) are linked to their room by `home:locatedIn` (green), to type nodes (grey) by `rdf:type`, and to literal attribute values (amber squares) by flat predicates such as `home:color` and `home:material`.

The resulting graph contains the following schema: every object is an instance of its type node (`home:sink_bath_1 rdf:type home:sink`), containment is a single relation (`home:sink_bath_1 home:locatedIn home:bathroom`), and object attributes are flat literal predicates (`home:color`, `home:material`, `home:size`, ...). Rooms are typed `home:Room` and carry human-readable labels.

Fig. 1 shows a fragment of the resulting graph: the complete description of two instances from the living room. The graph is serialized as RDF/Turtle and queried with SPARQL [14]; in the implementation, verification uses direct triple pattern matching. RDF is not merely a storage format. It provides declarative query semantics and interoperability with existing ontologies, making explanations independently verifiable by third-party tools.

A design principle of the KG is that it contains *only facts present in the map*. Lexical knowledge—exact synonyms (*television*→`tv`, *couch*→`sofa`) and broader terms (*table*→{`dining_table`, `coffee_table`})—is kept in a separate normalisation layer outside the RDF data. This separation prevents graph-level inference from silently "repairing" instructions that a trustworthy system should question: a broader term legitimately widens the candidate set (and may therefore cause ambiguity), whereas an unknown word must surface as a hallucination signal rather than being fuzzily matched to the nearest concept.

3.3. Map-Blind Mention Extraction

Grounding an instruction in the KG requires knowing *what the instruction refers to*. A single structured LLM call converts q into a list of target mentions, each with the fields ⟨*object_phrase*, *object_type*, *room*, *room_modifier*, *properties*, *spatial_relation* ⟩; for example, *"the office desk in Bedroom 1"* yields *object_type*=`desk`, *room_modifier*=`office`, *room*=`bedroom 1`. Crucially, the extractor is *map-blind*: it never sees the environment, so it performs a purely linguistic task and cannot leak world knowledge into the risk estimate. All grounding, and therefore all risk evidence, originates from the KG. This division of labour is what makes the resulting explanations faithful by construction (Section 3.6).

3.4. Symbolic Risk Verification on the KG

Each mention is grounded by deterministic graph queries, and the three risk dimensions correspond directly to three failure patterns of that grounding:

Hallucination. The mentioned type (after lexical normalisation) matches no type node, or the mentioned room matches no room node: the pattern ($?x$, `rdf:type`, `home:`t) has no solution. Then $s_h = 1$.

Semantic conflict. The type exists, but the containment or attribute edge required by the mention does not: e.g. the pattern ($?x$ `rdf:type home:bookshelf`$\wedge ?x$ `home:locatedIn home:kitchen`) is empty although bookshelf instances exist elsewhere. Then $s_c = 1$.

Ambiguity. More than one candidate instance satisfies all constraints of the mention and no discriminating cue is present. Then $s_a = 1$.

Fig. 2 makes the three checks concrete by expressing them as SPARQL queries over $\mathcal{G}$ for three benchmark instructions. The implementation evaluates the equivalent triple patterns directly through the graph API; in either formulation, the query solutions (or their absence) are exactly the evidence attached to the resulting decision.

Candidate sets are narrowed by a small collection of symbolic resolution rules before ambiguity is declared: (i) room constraints from both the explicit *"in the* ⟨*room*⟩*"* phrase and nominal room modifiers (*office desk*); contradictory constraints empty the intersection and raise s_c; (ii) attribute filters over literal predicates (*wooden*, *double*); (iii) superlative spatial cues (*closest*, *furthest*), which identify a unique referent geometrically and thus discriminate; (iv) identifier-level lexical matches in multi-target instructions; and (v) *tour context*: in a multi-target instruction, a generic follow-up mention (*"…then the desk"*) resolves to the instance located in a room already visited by another, uniquely resolved target, including targets appearing later in the sentence. Scores are intentionally binary: a fact either holds in $\mathcal{G}$ or it does not, and every raised signal carries its witness.

(a) Hallucination — *"Go to the piano."*

```
ASK { ?x rdf:type home:piano }
# no solution: the type node does not exist => s_h = 1 (reject)
```

(b) Semantic conflict — *"Go to the bookshelf in the kitchen."*

```
ASK { ?x rdf:type home:bookshelf ;
      home:locatedIn home:kitchen }
# false, although ASK { ?x rdf:type home:bookshelf } holds:
# the containment edge is missing => s_c = 1 (reject)
```

(c) Ambiguity — *"Visit the wooden table."*

```
SELECT ?x ?room WHERE {
 VALUES ?t { home:dining_table home:coffee_table } # lexical layer: "table"
 ?x rdf:type ?t ;
   home:material "wood" ;
   home:locatedIn ?room }
# 3 solutions, no discriminating constraint => s_a = 1 (clarify);
# the bindings are the candidates cited in the explanation
```

Figure 2: SPARQL formulation of the three risk checks for concrete benchmark instructions (prefix declarations omitted). Each raised signal is the observable outcome of a query: an empty ASK on the type pattern (a), an empty ASK on the joint type–containment pattern while the type alone matches (b), or a SELECT returning multiple undiscriminated bindings (c).

3.5. Risk Assessment Variants and Decision Gate

We compare three variants of the risk assessment stage, differing in the source used to estimate ambiguity, hallucination, and semantic conflict. LLM Risk Assessment (LLM-RA) follows the risk assessment approach introduced in our previous work [16], in which an LLM receives the instruction and a textual map summary and directly outputs the three risk scores and free-text risk flags. Knowledge Graph Risk Assessment (KG-RA) uses the deterministic graph-based verification described in Section 3.4. Hybrid combines both conservatively, per dimension:

$$s_x^{\text{hyb}} = \max\left(s_x^{\text{LLM}},\, s_x^{\text{KG}}\right), \qquad x \in \{a, h, c\}, \tag{1}$$

so that whenever the KG raises a dimension above the LLM estimate, the attached evidence triples are the literal cause of any decision change. All variants share the same aggregate score $R = w_a s_a + w_h s_h + w_c s_c$ with $(w_a, w_h, w_c) = (0.30, 0.40, 0.30)$, and the same deterministic decision gate we defined in our earlier study [16]:

$$d(q) = \begin{cases} \texttt{reject} & \text{if } s_h > \delta_h \text{ or } s_c > \delta_c, \\ \texttt{clarify} & \text{if } s_a > \delta_a, \\ \texttt{execute} & \text{if } R \le \tau_e, \\ \texttt{clarify} & \text{if } \tau_e < R \le \tau_c, \\ \texttt{reject} & \text{if } R > \tau_c, \end{cases} \tag{2}$$

with τ_e=0.10, τ_c=0.70, δ_a=0.75, δ_h=0.80, δ_c=0.85. Because the KG scores are binary, they interact with the gate without any threshold tuning: a single violated pattern is sufficient for rejection or clarification. Only when the gate outputs `execute` is a downstream tool-augmented LLM planner invoked to produce π.

3.6. KG-Grounded Explanations and Their Faithfulness

Every decision is accompanied by a structured explanation. For each mention, the verifier records an *evidence record* containing the supporting triples (e.g. `(home:sink_bath_1, home:locatedIn,`

home:bathroom)), the violated graph pattern when a signal is raised, and the surviving candidate set. The decision gate additionally reports which rule of Eq. (2) fired. Because the extractor is map-blind and the verifier only ever cites triples it actually matched (or failed to match) in $\mathcal{G}$, these explanations are *faithful by construction*: the cited evidence is the computation that produced the decision, not a post-hoc rationalisation.

The same machinery yields a KG-based *faithfulness metric* for free-text explanations produced by the LLM assessor. Each LLM risk flag (e.g. *"Bedroom 2 contains a library, not a bookshelf"*) is converted by a structured LLM call into typed claims—*type missing*, *type–room conflict*, *property conflict*, *multiple candidates*—and each claim is then checked symbolically against $\mathcal{G}$. A flag is *verified* if its checkable claims hold in the graph, *refuted* if contradicted, and *uncheckable* if it makes no claim about the scene. This turns explanation quality, usually assessed by human judgement, into a measurable property.

4. Results and Discussion

4.1. Experimental Setup

We evaluate on a benchmark of 206 natural-language instructions over a single-floor smart-home environment (8 rooms, 36 objects). Instructions fall into five categories: *single-step* (34) and *multi-step* (61) navigation tasks whose ground-truth decision is `execute`, *ambiguous* instructions (41, ground truth `clarify`), and *hallucination* (40) and *semantic-conflict* (30) scenarios (ground truth `reject`). All LLM components use `gpt-5.6-luna`; the symbolic verifier itself is LLM-free and costs one mention-extraction call per instruction.

We compare against two conventional LLM planners introduced in our earlier work [16], which receive the same model and map information but perform no explicit risk estimation: **SLLmP**, which is given the full semantic map (without coordinates) in its prompt and directly outputs a decision and plan, and **TA-LLmPA**, a tool-augmented agent that queries the map iteratively through function calls before answering. Both may decide to clarify or reject on their own initiative, so they are genuine baselines for trustworthy decision making. We report decision accuracy (DA) overall and per category; on the ambiguous, hallucination, and semantic-conflict subsets, per-category DA corresponds to the ambiguity detection rate (ADR), hallucination rejection rate (HRR), and semantic-conflict rejection rate (SCR).

All material required to reproduce the experiments is released on the `xai-kg-codebase`[1] branch of our repository: the scene knowledge graph in Turtle, the lexical normalisation dictionaries, the verifier and its resolution rules, the TRUST-NAV instructions with their ground-truth decisions, and the evaluation scripts producing Tables 1–3. The knowledge graph is released as a standalone Turtle file so that the scene, the constraints and the explanations can be inspected independently of our implementation.

4.2. Decision Reliability

Table 1 summarises decision accuracy and Fig. 3 shows the corresponding confusion matrices. KG-RA attains 99.5% overall and the best score in every category, including a perfect ADR, HRR, and SCR. The strongest conventional baseline, SLLmP, reaches 92.7% but misses one in eight ambiguous instructions, precisely the cases where committing to an arbitrary grounding is least desirable.

The errors of the purely neural LLM-RA variant are not random but stem from graded, uncalibrated scores meeting fixed thresholds: it under-commits on true failures (mean hallucination score of 0.88 on hallucinated instructions, occasionally below the rejection ceiling δ_h) and leaks risk mass onto valid instructions (non-zero ambiguity on multi-step tasks), producing the spurious clarification requests visible in Fig. 3. Binary, evidence-backed KG signals have neither defect. The Hybrid variant shows that the KG can repair part of this behaviour even when the LLM remains in the loop: the KG raised a dimension above the LLM estimate for 51 mention-level signals, which changed 13 final decisions, 12

[1]Scene knowledge graph, verifier, benchmark and evaluation scripts: https://github.com/iitis/Risk-Aware-Semantic-Grounding/tree/xai-kg-codebase

Table 1
Decision accuracy (%) per benchmark category. ADR, HRR, and SCR columns correspond to the ambiguous, hallucination, and semantic-conflict subsets.

Method	Single	Multi	ADR	HRR	SCR	Overall
SLLmP	**100.0**	83.6	87.8	**100.0**	**100.0**	92.7
TA-LLmPA	91.2	52.5	82.9	**100.0**	93.3	80.1
LLM-RA	97.1	73.8	92.7	75.0	**100.0**	85.4
KG-RA (ours)	**100.0**	**98.4**	**100.0**	**100.0**	**100.0**	**99.5**
Hybrid (ours)	97.1	73.8	97.6	**100.0**	**100.0**	91.3

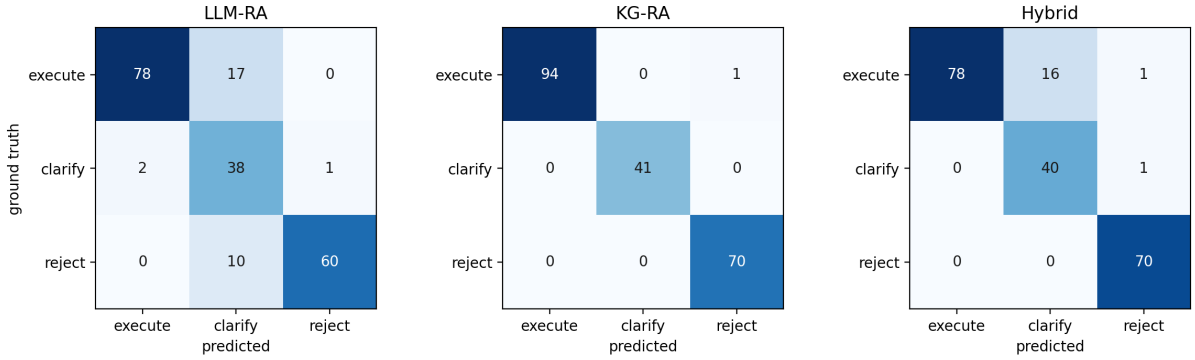


Figure 3: Decision confusion matrices (ground truth × predicted) for the three risk-assessment variants.

of them correctly and because the combination in Eq. (1) is a per-dimension maximum, every one of these changes is traceable to the specific evidence triples that caused it. What the Hybrid cannot do is retract the LLM's false alarms on executable instructions (it inherits them by construction), which is why KG-RA alone remains the strongest configuration in this closed-world setting.

The single KG-RA error is itself instructive: for *"…and double bed in bedroom 2"* the verifier rejects because the triple (`home:bed_double_1`, `home:locatedIn`, `home:bedroom_2`) does not exist (the double bed is in Bedroom 1) while the benchmark labels the instruction as executable. The symbolic verifier thus surfaced an annotation error in the benchmark itself, a useful side effect of evidence-backed verification: every disagreement with the ground truth comes with a checkable witness.

4.3. Explanation Quality and Faithfulness

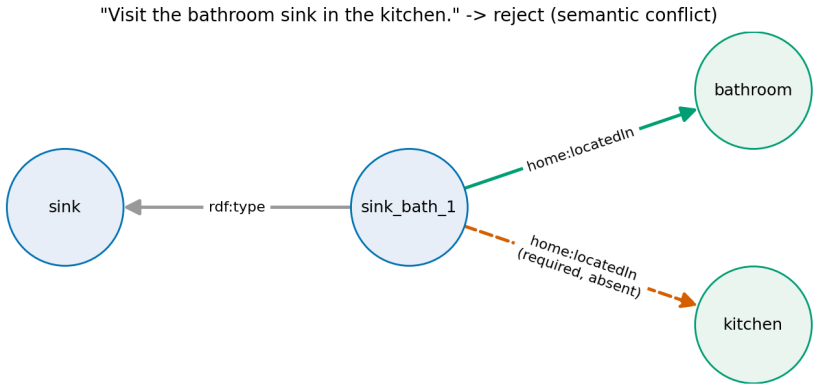


Figure 4: Evidence subgraph attached to a rejection. Solid edges are supporting triples found in the scene KG; the dashed edge is the containment relation required by the instruction but absent from the graph.

Figs. 4 and 5 show two complete explanations produced by KG-RA, one per non-execute decision type. For the rejected instruction *"Visit the bathroom sink in the kitchen"* (Fig. 4), the supporting triple locates the only bathroom sink in the bathroom, and the violated pattern states the containment edge the instruction would require but which is absent from the graph: the decision is explained by a *missing* edge. The clarification request for *"Visit the wooden table"* (Fig. 5) is the dual case: the generic word *table* expands through the lexical layer to two concrete types, the attribute filter `home:material` = "wood" is satisfied by all three instances, and their containment triples place them in two different rooms—every constraint holds, for *too many* referents, so the system asks the user to choose and can list the three candidates with their locations verbatim from the cited triples. Across the benchmark, non-execute decisions cite on average 2.7 triples or patterns, i.e. the explanations are compact enough to be shown verbatim to an end user.

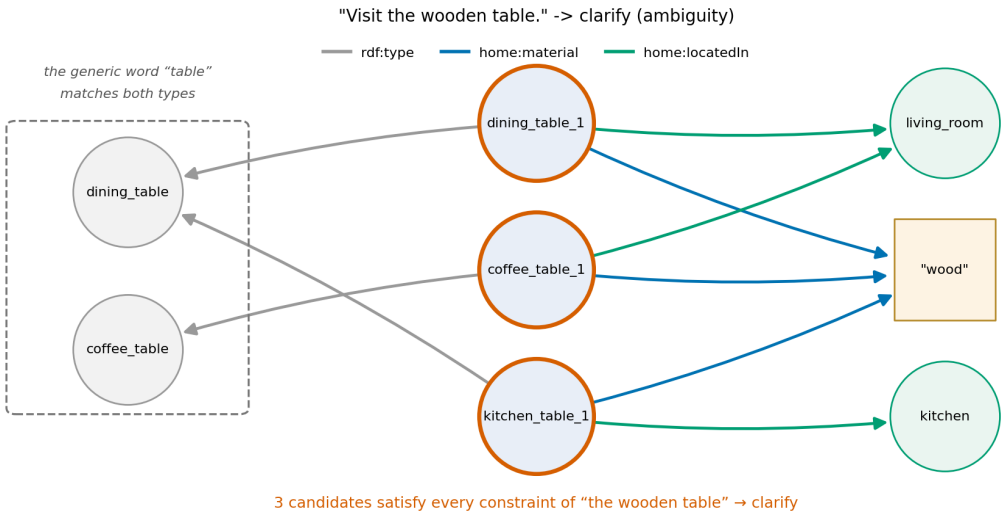


Figure 5: Evidence subgraph attached to a clarification request. The generic word *table* matches two concrete types (dashed box), the attribute constraint *wooden* is satisfied by all three instances, and no constraint discriminates between the highlighted candidates.

Table 2 reports the faithfulness of the LLM assessor's free-text flags under symbolic verification. Overall, 93.0% of checkable flags are supported by the graph, but 10 flags state scene facts that are simply false, and a further 24 are too vague to check. Notably, the two refuted flags on single-step instructions accompany exactly the false alarms discussed above: when the LLM makes a wrong decision, it also fabricates a wrong justification for it. This is the central argument for KG-grounded explanations in trustworthy robotics: a system whose explanations are generated by the same fallible model it is supposed to explain cannot be audited, whereas evidence triples can be re-checked against the scene at any time.

Table 2
Faithfulness of free-text LLM risk flags, verified symbolically against the scene KG. KG-RA explanations are faithful by construction (100%).

Category	Verified	Refuted	Uncheckable	Faithfulness (%)
Single-step	0	2	1	0.0
Multi-step	19	2	13	90.5
Ambiguous	44	4	7	91.7
Hallucination	40	0	0	100.0
Semantic conflict	29	2	3	93.5
Total	132	10	24	93.0

4.4. Ablation of the Symbolic Components

Table 3 isolates the contribution of the two main symbolic components introduced in Sections 3.2 and 3.4. Removing the lexical normalisation layer, the dictionaries that map instruction vocabulary to graph types, both exact synonyms (*television*→`tv`) and broader terms (*table*→{`dining_table`, `coffee_table`}), costs 9.7 points. Legitimate aliases are then reported as hallucinations, confirming that lexical knowledge is necessary, but, per the design principle of Section 3.2, it must live outside the graph, since the rejection categories rely on unknown words *not* being fuzzily matched. Removing tour-context resolution, rule (v), which grounds a generic mention through the room established by another target of the same instruction, costs 6.3 points. They are all on multi-step instructions: a wardrobe mentioned in a tour that starts at the double bed unambiguously means the Bedroom 1 wardrobe, but without the rule every such follow-up mention (*"…then the desk"*) triggers a spurious clarification request. With both removed, accuracy drops to 83.5%—i.e. most of the advantage of the symbolic verifier comes from resolution rules that are themselves interpretable and individually testable.

Table 3
Ablation of the KG verifier (decision accuracy, %).

Configuration	Single	Multi	ADR	HRR	SCR	Overall
Full KG-RA	100.0	98.4	100.0	100.0	100.0	99.5
w/o lexical layer	97.1	77.0	85.4	100.0	100.0	89.8
w/o tour context	100.0	77.0	100.0	100.0	100.0	93.2
w/o both	97.1	55.7	85.4	100.0	100.0	83.5
w/o KG (LLM-RA)	97.1	73.8	92.7	75.0	100.0	85.4

The last row of Table 3 removes the graph altogether: without a KG, risk assessment can only fall back to the LLM assessor (LLM-RA, Section 3.5). Its overall accuracy (85.4%) lands in the same range as the fully stripped symbolic verifier (83.5%), but the two fail in qualitatively different ways. The stripped verifier loses only its ability to *resolve* references: its errors are excess clarification requests, while HRR and SCR remain at 100% and every raised signal still carries checkable evidence. The LLM instead under-rejects hallucinated instructions (HRR 75.0%) and, as shown in Section 4.3, occasionally justifies its decisions with unfaithful flags. In a trustworthiness setting these failure modes are not equivalent: an unnecessary question costs a dialogue turn, whereas an executed hallucination and a fabricated explanation cost user trust.

4.5. Discussion and Limitations

When the environment admits an explicit KG representation, the grounding-risk estimation that is commonly delegated to an LLM reduces to triple-pattern matching, and performing it symbolically involves no trade-off: it is simultaneously more accurate (Table 1) and more explainable (Section 4.3). Moreover, the KG remains useful even when an LLM assessor is retained: it repairs miscalibrated decisions with attributable evidence (Hybrid), quantifies the faithfulness of the LLM's own explanations, and audits the benchmark annotations themselves.

The main limitation is the closed-world assumption: the verifier can only reason about facts representable in the scene graph, and its perfect scores are relative to a benchmark whose failure modes are, by design, grounding failures. Risks that live outside the map, such as unsafe physical actions, temporal constraints, and perception errors that make the map itself wrong, are invisible to it; retaining an LLM assessor in the Hybrid configuration is a pragmatic hedge against this blind spot, and extending the graph with affordance and safety relations is the natural next step. Finally, our environment is a single simulated apartment; validating the approach on larger scenes and on maps built by real perception pipelines remains future work.

5. Conclusion

We have argued that trustworthy LLM-based robot planning should be treated as a grounding verification problem, and that when the environment is available as a knowledge graph the verification reduces to query evaluation. Hallucination, semantic conflict and ambiguity are not latent properties of the language model requiring calibration; they are an empty answer set, an empty conjunction of individually satisfiable patterns, and a non-unique solution mapping. Each is decidable, and each carries the triples that produced it.

The empirical consequence is that explainability and accuracy do not trade off here. The symbolic verifier reaches 99.5% decision accuracy against 92.7% for the strongest conventional LLM baseline, and it does so while returning, for every non-execution decision, the graph evidence that caused it. The comparison with the LLM assessor shows why: graded self-assessment under-commits on genuine failures and leaks risk onto valid instructions, and when it decides wrongly it fabricates the scene fact that justifies the decision. A system whose explanations are produced by the model they are meant to explain cannot be audited. Evidence triples can be re-checked against the scene at any time, as they were here when the verifier's single disagreement with the ground truth turned out to be a mislabelled benchmark item.

Portability and outlook. Porting the framework to a new environment is uneven in cost. Compiling a map into RDF is mechanical, and the symbolic resolution rules are environment-independent. The lexical layer is the real expense: learning it from data while preserving the principle that unknown words must surface rather than be fuzzily repaired is an open problem. Beyond that, aligning the schema with established smart-home vocabularies would let deployment-specific constraints be stated as SHACL shapes over standard terms, and extending the graph with affordance and safety relations would bring risks currently outside its reach into the same verifiable regime. So that the robot can say not only why it did not know where to go, but why it should not go there.

Acknowledgments

This work was supported by COST Action CA24121 – Knowledge Graphs in the Era of Large Language Models (KGELL).

Declaration on Generative AI

During the preparation of this work, the authors used Claude (Anthropic) in order to: Paraphrase and reword, improve writing style, and check grammar and spelling. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, A. Van Den Hengel, Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 3674–3683.

[2] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, D. Fox, Alfred: A benchmark for interpreting grounded instructions for everyday tasks, in: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, 2020, pp. 10737–10746.

[3] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, N. Roy, Understanding natural language commands for robotic navigation and mobile manipulation, in: Proceedings of the AAAI conference on artificial intelligence, volume 25, 2011, pp. 1507–1514.

[4] W. Huang, P. Abbeel, D. Pathak, I. Mordatch, Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, in: International conference on machine learning, PMLR, 2022, pp. 9118–9147.
[5] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al., Do as i can, not as i say: Grounding language in robotic affordances, in: Conference on robot learning, PMLR, 2023, pp. 287–318.
[6] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, T. Jackson, N. Brown, L. Luu, S. Levine, K. Hausman, b. ichter, Inner monologue: Embodied reasoning through planning with language models, in: K. Liu, D. Kulic, J. Ichnowski (Eds.), Proceedings of The 6th Conference on Robot Learning, volume 205 of *Proceedings of Machine Learning Research*, PMLR, 2023, pp. 1769–1782.
[7] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, A. Garg, ProgPrompt: Generating situated robot task plans using large language models, in: IEEE International Conference on Robotics and Automation (ICRA), 2023, pp. 11523–11530.
[8] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, A. Zeng, Code as policies: Language model programs for embodied control, in: 2023 IEEE International conference on robotics and automation (ICRA), IEEE, 2023, pp. 9493–9500.
[9] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al., PaLM-E: An embodied multimodal language model, in: International Conference on Machine Learning (ICML), 2023, pp. 8469–8488.
[10] A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley, Z. Xu, D. Sadigh, A. Zeng, A. Majumdar, Robots that ask for help: Uncertainty alignment for large language model planners, in: Conference on Robot Learning (CoRL), 2023, pp. 661–682.
[11] K. Liang, Z. Zhang, J. F. Fisac, Introspective planning: Aligning robots' uncertainty with inherent task ambiguity, Advances in Neural Information Processing Systems 37 (2024) 71998–72031.
[12] S. Yin, X. Pang, Y. Ding, M. Chen, Y. Bi, Y. Xiong, W. Huang, Z. Xiang, J. Shao, S. Chen, Safeagentbench: A benchmark for safe task planning of embodied llm agents, arXiv preprint arXiv:2412.13178 (2024).
[13] J. Wang, L. Zhao, X. S. Zhang, Madra: Multi-agent debate for risk-aware embodied planning, in: Findings of the Association for Computational Linguistics: ACL 2026, 2026, pp. 6852–6876.
[14] S. Harris, A. Seaborne, SPARQL 1.1 query language, W3C Recommendation, 2013. https://www.w3.org/TR/sparql11-query/.
[15] Ł. Sobczak, N. Keleşoğlu, S. P. Nowak, TRUST-NAV: A benchmark for trustworthy language-guided robot planning, https://github.com/iitis/Risk-Aware-Semantic-Grounding, 2026. Dataset and source code.
[16] Łukasz Sobczak, N. Keleşoğlu, S. P. Nowak, Risk-aware semantic grounding for trustworthy llm-based robot planning, 2026. arXiv:2609.37554.
[17] I. Tiddi, S. Schlobach, Knowledge graphs as tools for explainable machine learning: A survey, Artificial Intelligence 302 (2022) 103627.
[18] F. Lecue, On the role of knowledge graphs in explainable ai, Semantic Web 11 (2020) 41–51.
[19] R. Confalonieri, T. Weyde, T. R. Besold, F. Moscoso del Prado Martín, Trepan reloaded: A knowledge-driven approach to explaining black-box models, in: ECAI 2020: 24th European Conference on Artificial Intelligence, 29 August–8 September 2020, Santiago de Compostela, Spain, 2020, pp. 2457–2464.
[20] R. Confalonieri, T. Weyde, T. R. Besold, F. M. del Prado Martín, Using ontologies to enhance human understandability of global post-hoc explanations of black-box models, Artificial Intelligence 296 (2021) 103471.
[21] R. Barile, C. d'Amato, N. Fanizzi, Explanation of link predictions on knowledge graphs via levelwise filtering and graph summarization, in: European Semantic Web Conference, Springer, 2024, pp. 180–198.
[22] R. Barile, C. d'Amato, N. Fanizzi, Lp-dixit: Evaluating explanations for link predictions on knowledge graphs using large language models, in: Proceedings of the ACM on Web Conference

2025, 2025, pp. 4034–4042.
[23] R. Barile, C. d'Amato, N. Fanizzi, Grainsack: a comprehensive software library for benchmarking explanations of link prediction tasks on knowledge graphs, arXiv preprint arXiv:2508.08815 (2025).
[24] H. Knublauch, D. Kontokostas, Shapes constraint language (SHACL), W3C Recommendation, 2017. https://www.w3.org/TR/shacl/.
[25] T. Lebo, S. Sahoo, D. McGuinness, PROV-O: The PROV ontology, W3C Recommendation, 2013. https://www.w3.org/TR/prov-o/.
[26] D. Bonino, F. Corno, Dogont-ontology modeling for intelligent domotic environments, in: International Semantic Web Conference, Springer, 2008, pp. 790–803.
[27] L. Daniele, F. den Hartog, J. Roes, Created in close interaction with the industry: the smart appliances reference (saref) ontology, in: International Workshop Formal Ontologies Meet Industries, Springer, 2015, pp. 100–112.
[28] M. H. Rasmussen, M. Lefrançois, G. F. Schneider, P. Pauwels, Bot: The building topology ontology of the w3c linked building data group, Semantic Web 12 (2020) 143–161.
[29] A. Nüchter, J. Hertzberg, Towards semantic maps for mobile robots, Robotics and Autonomous Systems 56 (2008) 915–926.
[30] Ł. Sobczak, P. Biernacki, J. Domańska, Visual encoding method for semantic mapping with federated learning concept, in: Proceedings of the Twenty-sixth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing, 2025, pp. 428–435.
[31] M. Halama, S. Nowak, K. Połys, Semantic orientation for indoor navigation system using large language models, Scientific Reports 15 (2025) 41519.
[32] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, et al., ConceptGraphs: Open-vocabulary 3D scene graphs for perception and planning, in: IEEE International Conference on Robotics and Automation (ICRA), 2024, pp. 5021–5028.
[33] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, N. Suenderhauf, Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning, in: J. Tan, M. Toussaint, K. Darvish (Eds.), Proceedings of The 7th Conference on Robot Learning, volume 229 of *Proceedings of Machine Learning Research*, PMLR, 2023, pp. 23–72.
[34] L. P. Kaelbling, T. Lozano-Pérez, Integrated task and motion planning in belief space, The International Journal of Robotics Research 32 (2013) 1194–1227.