ONE HYPOTHESIS IS NOT ENOUGH: ABDUCTIVE REASONING WITH AGENTIC HYPOTHESIS REFINEMENT OVER KNOWLEDGE GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Abductive reasoning over knowledge graphs (KGs) seeks a first-order logic hypothesis whose answer set explains a given set of observed entities. Since many hypotheses can explain the same observations, controllable hypothesis generators condition generation on entities, relations, or logical patterns, but they treat generation as a single step. A generated hypothesis may be well-formed and satisfy the given conditions yet still fail to explain the observations, and the model has no mechanism to detect or correct this mismatch. Closing the abductive cycle requires revising a discrete, structured hypothesis, which large language models cannot do reliably, even though they iterate readily in natural language. We propose HypoAgent, an agentic hypothesis refinement framework that treats condition signals not only as expressions of user intent but also as operators that steer generation. A Hypothesis Proposal Agent calls a small trained generator to propose a hypothesis. Root-cause analysis then uses fragment-level coverage to locate branches worth inspecting and gathers neighborhood evidence from the training graph. A Hypothesis Refiner Agent combines this diagnosis with previously generated hypotheses to produce updated conditions and directly revised hypotheses. HypoAgent outperforms one-shot generation in single-turn and multi-turn settings on BioKG, PharmKG8k, and DBpedia50, and in the unconditional setting on DBpedia50.

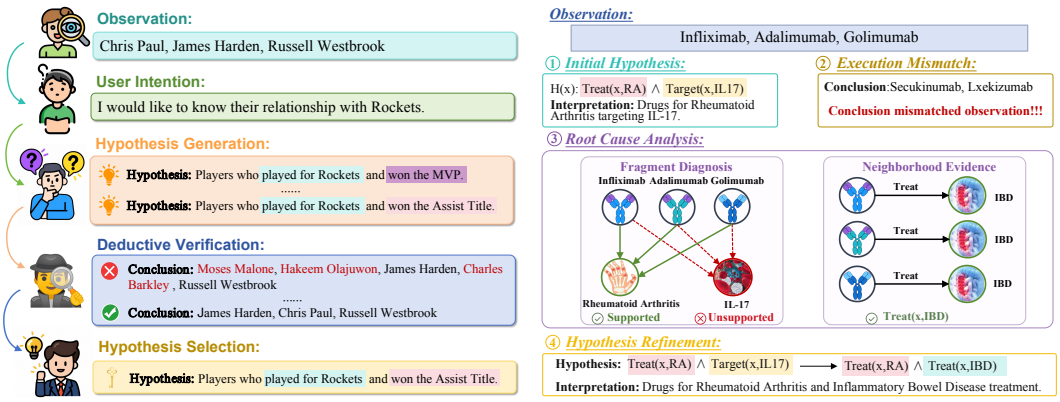
1 INTRODUCTION

Abductive reasoning (Paul, 1993) is one of the three classical modes of reasoning, alongside deductive (Johnson-Laird, 1999) and inductive reasoning (Hayes et al., 2010). It reasons from evidence to hypotheses, seeking the most plausible explanation for observed phenomena. It is widely used in clinical diagnosis (Pukancová & Homola, 2015; Martini, 2023), where physicians infer latent diseases or mechanisms from symptoms and patient profiles, and in scientific discovery (Engelschalt et al., 2023; Wackerly, 2021; Duede & Evans, 2021; Upmeier zu Belzen et al., 2021), where researchers propose and revise hypotheses to explain empirical observations.

Knowledge graphs (KGs) provide a natural substrate for abductive reasoning, as they represent domain knowledge as interconnected entities and relations. Over a KG, abductive reasoning (Bai et al., 2024) aims to find a first-order logic hypothesis whose answer set explains a given set of observed entities. For example, given several diseases, a system may hypothesize about their shared mechanisms, treatments, or susceptible populations. Such logical hypotheses are interpretable, since their answer sets and supporting paths can be inspected directly on the KG. However, the task is underdetermined: a single observation set may admit many plausible hypotheses, especially in large and densely connected KGs. Recent work (Gao et al., 2025) addresses this issue with controllable hypothesis generation, where users guide generation through explicit conditions, such as target entities, relations, or logical patterns, to obtain hypotheses that match their interests.

Abductive reasoning, however, is inherently iterative: hypotheses are proposed, validated, and refined. Controllable hypothesis generation (Gao et al., 2025) instead formalizes the task as one-shot conditional generation: given an observation set and a condition set, the model produces a hypothesis and terminates, executing only the first step of this cycle. A generated hypothesis may be

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Two hypotheses that both satisfy the condition; only one explains the observations.

(b) Fragment diagnosis and neighborhood evidence guide hypothesis repair.

Figure 1: Conditioning decides the direction of generation, not whether the hypothesis is correct.

well-formed and satisfy the given conditions, yet still explain the observations poorly, and there is no mechanism to detect or correct the mismatch. As shown in Fig. 1a, given three NBA players as observations and a condition related to the Rockets, the model may hypothesize that they are players who played for the Rockets and won the MVP award. Although this hypothesis looks plausible and satisfies the condition, its answer set does not match the observed entities. Conditioning determines the direction of generation, but not whether the resulting hypothesis is correct.

Closing this cycle requires two capabilities. First, execution feedback should guide targeted revision rather than undirected regeneration: fragment-level coverage indicates which branches to inspect, and neighborhood evidence suggests alternative anchors. Fig. 1b illustrates such a repair, where a target-based query is replaced by a treatment-based query that better matches the observations on the graph. Second, previous hypotheses should inform later attempts, since even an imperfect query may contain useful structures or anchors. Together, these capabilities call for an execution-guided search that reuses information from earlier candidates.

LLM agents can invoke tools and interpret execution feedback, but prior work shows that they struggle to generate strongly structured hypotheses (Bai et al., 2024; Zheng et al., 2025). We therefore separate revision planning from structured hypothesis generation: the LLM expresses the intended change as conditions for a specialized generator, while direct rewriting remains available as a complementary route. This division of labor lets the LLM focus on deciding what to change and the generator on producing well-formed logical queries.

Building on this analysis, we propose **HypoAgent**, an agentic hypothesis refinement framework for abductive reasoning over KGs. Given a set of observed entities and a condition set, a Hypothesis Proposal Agent first invokes a small trained generator to produce an initial hypothesis. Root-cause analysis then examines this hypothesis at a fine-grained level from two complementary perspectives. Internally, the hypothesis is decomposed into executable fragments, whose coverage of the observations indicates which parts to inspect and which to reuse. Externally, the agent explores the KG neighborhoods of the observed entities to identify relations and entity anchors supported by the graph. The resulting diagnosis, together with previously generated hypotheses, is passed to a Hypothesis Refiner Agent, which decides how to revise the current hypothesis. Our key insight is that condition signals can not only encode user intent but also steer the direction of refinement. Accordingly, the Refiner both revises the logical form directly and produces modified conditions that encode the desired correction; these conditions are passed back to the Proposal Agent to obtain new structured proposals. Among the complete candidates, the system prioritizes adherence to the user conditions and then training-graph Jaccard similarity.

Our contributions are summarized as follows:

- **Condition-mediated refinement.** We use execution feedback to turn a one-shot conditional generator into the proposal component of a refinement loop, with direct rewriting as a complementary route.
- **Root-cause analysis.** We combine fragment-level coverage statistics with KG neighborhood probing, guiding hypothesis revision with complementary internal and external evidence.
- **Empirical evidence.** HypoAgent outperforms one-shot generation in single-turn and multi-turn settings on BioKG, PharmKG8k, and DBpedia50, and in the unconditional setting on DBpedia50. On DBpedia50 single-turn tasks, it also requires fewer remote LLM calls and lower inference cost than ReAct-style agents.

2 Related Work

Abductive Reasoning. Abductive reasoning infers plausible explanations from observations and has been studied in a variety of general reasoning settings. In natural language inference, α-NLI (Bhagavatula et al., 2020) framed abduction as a commonsense reasoning task in which systems select the most plausible explanation for an observed event sequence. Subsequent studies improve abductive inference through decoding, prompting, self-consistency, and uncommon-sense reasoning (Qin et al., 2021; Kadiķis et al., 2022; Chan et al., 2023; Zhao et al., 2024). Beyond commonsense scenarios, abductive reasoning has also been explored in formal textual reasoning benchmarks such as ProofWriter (Tafjord et al., 2021), open-world reasoning with LLMs (Zhong et al., 2023; Del & Fishel, 2023; Thagard, 2024), abstract logical reasoning (Liu et al., 2024; Zheng et al., 2025), and neuro-symbolic abductive learning (Zhou, 2019; Camposampiero et al., 2024; Hu et al., 2025).

Abductive Reasoning on Knowledge Graphs. Abductive reasoning over knowledge graphs generates logical hypotheses that explain a set of observed entities or facts. AbductiveKGR (Bai et al., 2024) first introduced this problem, formulating KG abduction as Transformer-based hypothesis generation in which a model produces candidate logical explanations for the observations. CtrlH-Gen (Gao et al., 2025) then introduced controllable abductive hypothesis generation by conditioning the generator on user-specified control signals, so that generated hypotheses better satisfy the desired semantic or structural constraints. DARK (Gao et al., 2026b) unified deductive and abductive reasoning over KGs by modeling their bidirectional relationship: it treats hypotheses and conclusions as two sides of a sequence and uses masked diffusion to reason in both directions. Together, these efforts have advanced neural graph database reasoning (Bai et al., 2025; Li et al., 2026; Xie et al., 2026; Gao et al., 2026a) by improving hypothesis generation and controllability and by better capturing user intent.

3 Preliminary

Notation. We define a knowledge graph as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations, and $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is the set of observed triples. A triple $(v_i, r, v_j) \in \mathcal{T}$ indicates that relation r holds between entities v_i and v_j. Under the open-world assumption (Drummond & Shearer, 2006), the KG is incomplete, and a missing edge does not imply that the corresponding real-world relation is false; we therefore evaluate queries by executing them on fixed graph snapshots. In this work, a hypothesis H is represented as a first-order logic formula over the entities, variables, and relations in $\mathcal{G}$, with existential quantifiers and logical connectives such as $\wedge$, $\vee$, and $\neg$. For example, a conjunctive hypothesis can be written as

$$H(X) = \exists Z_1, \ldots, Z_k : \; a_1 \wedge a_2 \wedge \cdots \wedge a_m,$$

where X is the target variable, $Z_1, \ldots, Z_k$ are existentially quantified intermediate variables, and each a_i is a relational literal, i.e., either a relational predicate $r(u, v)$ or its negation $\neg r(u, v)$. Here, r is a relation in $\mathcal{G}$, and u, v can be entities, the target variable X, or existential variables from $\{Z_1, \ldots, Z_k\}$. Given a hypothesis H, we denote by $\mathcal{A}_{\mathcal{G}}(H)$ the answer set obtained by executing H on $\mathcal{G}$. That is, $\mathcal{A}_{\mathcal{G}}(H) \subseteq \mathcal{E}$ contains the entities that, when assigned to the target variable X, make $H(X)$ hold on $\mathcal{G}$. Negation follows the standard graph-query semantics: $\mathcal{A}_{\mathcal{G}}(\neg H) = \mathcal{E} \setminus \mathcal{A}_{\mathcal{G}}(H)$.

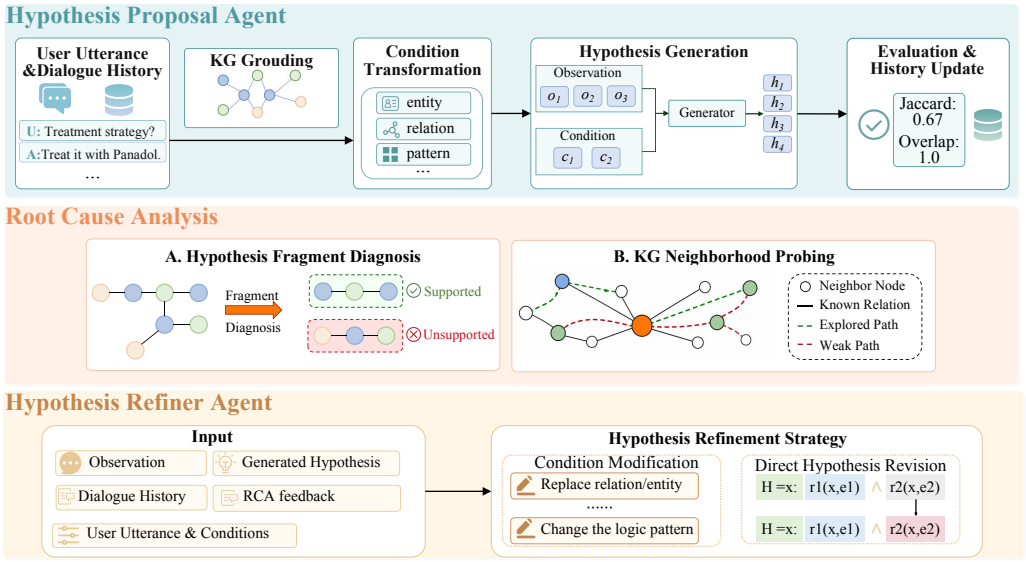


Figure 2: Overview of HypoAgent.

Task definition. Given a knowledge graph $\mathcal{G}$ and a set of observed entities $\mathcal{O} = \{o_1, o_2, \ldots, o_n\} \subseteq \mathcal{E}$, abductive hypothesis generation over KGs aims to infer a plausible first-order logic hypothesis H that explains the observations. Let $\mathcal{H}$ denote the space of candidate logical queries. The quality of a hypothesis is measured by the similarity between its answer set and $\mathcal{O}$:

$$H^* = \arg\max_{H \in \mathcal{H}} \text{sim}\left(\mathcal{A}_{\mathcal{G}}(H), \mathcal{O}\right), \tag{1}$$

where $\text{sim}(\cdot, \cdot)$ can be instantiated by set-based metrics such as the Jaccard similarity, Dice coefficient, and Overlap coefficient. Since different hypotheses may share the same answer set, the goal is to find a hypothesis that explains $\mathcal{O}$, not to recover a unique ground-truth formula. In controllable abductive hypothesis generation (Gao et al., 2025), the model is additionally given a user-specified control condition $\mathbf{c}$, and the goal is to generate a hypothesis H that both matches $\mathcal{O}$ and satisfies $\mathbf{c}$.

4 Method

4.1 Overview

We propose **HypoAgent**, an agentic hypothesis refinement framework for abductive reasoning over knowledge graphs, as illustrated in Fig. 2. Given a knowledge graph, observations, and a user utterance, the system searches for a logical query that matches the observations while prioritizing the requested conditions. The condition set is usually derived from the user's request; when none is given, the system induces proposal hints from an unconditional hypothesis (Appendix B.4). Each iteration consists of three sequential components. First, a Hypothesis Proposal Agent (Section 4.2) calls a trained generator to propose a hypothesis from the observations and the extracted conditions, and validates it on the training graph. If the hypothesis is satisfactory, the loop terminates. Otherwise, root-cause analysis (RCA; Section 4.3) inspects the hypothesis at a fine-grained level and produces a diagnosis. Finally, a Hypothesis Refiner Agent (Section 4.4) uses the diagnosis and previously generated hypotheses to produce revision candidates: a directly composed hypothesis and updated conditions for the Proposal Agent.

4.2 Hypothesis Proposal Agent

Within a dialogue turn, t indexes refinement rounds. At round t, the Proposal Agent generates a query under the current condition set $\mathbf{c}_t$, which may extend the user's conditions; the original user conditions stay fixed and serve as the selection criterion. Following Gao et al. (2025), each condition

set is a collection of key–value pairs $\mathbf{c}_t = \{(k_i, v_i)\}$, where each key k_i is one of five types: *relation*, *entity*, *relationnumber*, *entitynumber*, and *pattern*. The first two specify semantic anchors, and the last three specify structural constraints on the target hypothesis.

To generate structured proposals efficiently, the Proposal Agent invokes a lightweight Transformer-based conditional generator p_θ instead of asking an LLM to write the logical form. Given $\mathcal{O}$ and $\mathbf{c}_t$, the generator decodes a hypothesis H_t autoregressively:

$$p_\theta(H_t \mid \mathcal{O}, \mathbf{c}_t) = \prod_{\ell=1}^{L} p_\theta(h_t^\ell \mid h_t^{<\ell}, \mathcal{O}, \mathbf{c}_t), \tag{2}$$

where h_t^ℓ denotes the ℓ-th token of H_t. The logical hypothesis is then verbalized into natural language for interaction.

We train p_θ in two stages. In the first stage, it is trained without conditions on sampled observation–hypothesis pairs to learn to generate well-formed hypotheses that explain $\mathcal{O}$. In the second stage, it is trained on observation–condition–hypothesis triples to learn to satisfy the specified semantic and structural constraints while still explaining the observations.

At inference time, the Proposal Agent receives $\mathbf{c}_t$ (from the user, from a previous refinement round, or as an empty set) and calls p_θ to obtain H_t. The hypothesis is executed on the training graph to obtain $\mathcal{A}_{\mathcal{G}_{\text{train}}}(H_t)$, and its quality is measured by $\text{sim}\big(\mathcal{A}_{\mathcal{G}_{\text{train}}}(H_t), \mathcal{O}\big)$, instantiated as Jaccard similarity. To avoid information leakage, all refinement signals are computed on $\mathcal{G}_{\text{train}}$ rather than $\mathcal{G}_{\text{test}}$. If the score exceeds a threshold, H_t is returned; otherwise, the system proceeds to root-cause analysis.

4.3 Root-Cause Analysis

Root-cause analysis uses execution feedback to investigate why H_t fails to recover the observations. It examines the failure from two complementary angles: the internal structure of H_t and external evidence in the KG neighborhood of $\mathcal{O}$.

Hypothesis Fragment Diagnosis. We first decompose H_t into smaller executable fragments $\{H_t^{(1)}, \ldots, H_t^{(m)}\}$. Each fragment is a meaningful component of the formula, such as a single relational atom, an intermediate projection chain, or a partial conjunction. The diagnostic tool executes these sub-queries on $\mathcal{G}_{\text{train}}$ and reports their answer counts and overlap with the observations. For a fragment with answer set $A_t^{(j)}$, the overlap count is

$$o_t^{(j)} = \left| A_t^{(j)} \cap \mathcal{O} \right|. \tag{3}$$

The Refiner prioritizes low-overlap fragments for inspection and treats high-overlap fragments as candidate building blocks. Since a fragment's role depends on its logical context (e.g., a branch with low standalone Jaccard can still be necessary for a correct intersection), these statistics guide revision, while complete hypotheses are always scored by training-graph Jaccard (Appendix B.3).

KG Neighborhood Probing. In parallel, we probe the local neighborhood of the observed entities, where useful semantic anchors often appear. The analysis searches for relations, entities, and short relational paths that frequently connect to the observations, and evaluates how well the corresponding candidate fragments recover $\mathcal{O}$. High-scoring candidates can reveal relations or entities that are missing from the current hypothesis.

The diagnostic signal ϕ_t aggregates both views: fragment-level coverage statistics, branches to inspect, and candidate neighborhood anchors supported by $\mathcal{G}_{\text{train}}$.

4.4 Hypothesis Refiner Agent

The Hypothesis Refiner Agent decides how H_t should be revised. It receives the diagnostic signal ϕ_t, the observation set $\mathcal{O}$, the user utterance u_t, and the history $\mathcal{D}_{<t}$ of previously generated hypotheses together with their Jaccard scores. The key design is to use conditions as a structured repair interface: the diagnosis changes the generator's inputs, so the LLM need not express every repair as a logical query. We therefore ask the Refiner both to revise the hypothesis directly and

to propose updated conditions that steer subsequent generation toward a better match with the observations. For example, on top of the user-intent conditions, it may add anchors suggested by the fragment-level coverage in ϕ_t, or frequent relations and entities from the neighborhood of the observed entities. If u_t refers to a previous result (e.g., "make it simpler"), it is interpreted with respect to $\mathcal{D}_{\leq t}$ rather than in isolation. Formally, this process is written as

$$(\tilde{H}_{t+1},\, \mathbf{c}_{t+1}) = \mathrm{Refiner}(\mathbf{c}_t,\, \phi_t,\, \mathcal{D}_{\leq t},\, u_t), \tag{4}$$

where $\tilde{H}_{t+1}$ is the hypothesis revised by the Refiner and $\mathbf{c}_{t+1}$ is the updated condition set passed back to the Proposal Agent. The Proposal Agent then uses p_θ to generate candidates under both the unchanged and the updated conditions, complementing the Refiner's direct revision. Each candidate is executed on $\mathcal{G}_{\mathrm{train}}$ and ranked first by adherence to the user conditions and then by Jaccard similarity with $\mathcal{O}$. The system keeps the best hypothesis across rounds and returns it once the similarity threshold is reached or the refinement budget is exhausted. Implementation details are given in Appendix B.4.

5 Experiment

We evaluate HypoAgent along five research questions: single-turn controllable generation (RQ1), history-aware multi-turn refinement (RQ2), unconditional self-improvement (RQ3), component contributions (RQ4), and inference efficiency (RQ5).

5.1 Experimental Setup

Datasets. We evaluate HypoAgent on three KG benchmarks: BioKG (Walsh et al., 2020), PharmKG8k (Zheng et al., 2021), and DBpedia50 (Auer et al., 2007). Following prior work (Bai et al., 2024; Gao et al., 2025), we partition KG edges in an 8:1:1 ratio and construct cumulative training, validation, and test execution graphs (Appendix A). We sample hypothesis–observation pairs from 13 logical patterns and retain only test queries whose complete answer sets contain at most 32 entities, so no answer set is truncated. The unconditional setting uses these sampled pairs directly. For the single-turn and multi-turn dialogue settings, we further simulate user requests from the sampled cases: single-turn requests verbalize sampled control conditions, while multi-turn requests are built from cases that share the same observations but differ in hypotheses, reflecting progressive shifts in user intent. Details are provided in Appendix A.

Evaluation Metrics. We evaluate generated hypotheses in terms of semantic similarity and condition adherence. For semantic similarity, we execute the selected hypothesis on $\mathcal{G}_{\mathrm{test}}$ and compare its answer set with the observation set $\mathcal{O}$ using the Jaccard similarity, Dice coefficient, and Overlap coefficient. All inference-time diagnosis, candidate validation, selection, and stopping use only $\mathcal{G}_{\mathrm{train}}$. For condition adherence, Accuracy is the mean binary satisfaction of the dataset's gold conditions, which are independent of the conditions parsed by the model.

Implementation Details. Following Gao et al. (2025), the hypothesis generator is a 6-layer Transformer. For the LLM backend of the agents, we instantiate HypoAgent with DeepSeek-V4-Flash, Qwen3-235B (Yang et al., 2025), and GPT-5.4-mini, denoted as $\text{HypoAgent}_{\mathrm{DS}}$, $\text{HypoAgent}_{\mathrm{QWEN}}$, and $\text{HypoAgent}_{\mathrm{GPT}}$, respectively. The maximum number of refinement rounds is set to three. All experiments are conducted on a single NVIDIA A6000 48GB GPU. Further details on training and agent construction are reported in Appendix B.

5.2 RQ1: Single-turn Controllable Hypothesis Generation

We compare HypoAgent with the following baselines in the single-turn setting. *Symbolic Search* performs structure-based search on the training KG. Since it cannot interpret natural-language conditions, we use it only to find the hypothesis with the highest semantic similarity and report its condition accuracy as "–". *ReAct* is a free-form ReAct-style LLM agent (Yao et al., 2022) equipped with the same KG tools, namely *Hypothesis Fragment Diagnosis* and *KG Neighborhood Probing*, but without the structured refinement loop of HypoAgent. *CtrlHGen* (Gao et al., 2025) is evaluated with a single generation and with k independent generations (*best-of-k*), where k equals the maximum number of refinement rounds of HypoAgent. The best-of-k variant measures the gain

Table 1: Single-turn controllable hypothesis generation results. All values are means; across-case standard deviations are in Appendix D. (**Bold**: best; Underline: runner-up.)

Model	BioKG				PharmKG8k				DBpedia50			
	Jaccard	Dice	Overlap	Accuracy	Jaccard	Dice	Overlap	Accuracy	Jaccard	Dice	Overlap	Accuracy
Symbolic Search	0.420	0.480	0.623	–	0.495	0.583	0.771	–	0.460	0.485	0.539	–
ReAct	0.224	0.247	0.347	0.275	0.216	0.248	0.343	0.315	0.409	0.422	0.446	0.440
CtrlHGen	0.718	0.761	0.858	0.909	0.633	0.702	0.812	0.925	0.781	0.829	0.896	0.730
CtrlHGen (best-of-k)	0.758	0.803	0.895	0.904	0.704	0.775	0.882	0.926	0.810	0.852	0.912	0.710
ReAct + CtrlHGen	0.718	0.768	0.893	0.802	0.595	0.659	0.773	0.753	0.839	0.873	0.934	0.676
HypoAgent$_{\text{DS}}$	**0.885**	**0.917**	**0.971**	0.941	**0.826**	**0.858**	**0.947**	0.935	**0.945**	**0.961**	**0.984**	**0.746**
HypoAgent$_{\text{QWEN}}$	0.859	0.890	0.951	0.940	0.817	0.856	0.926	**0.944**	0.897	0.929	0.979	0.731
HypoAgent$_{\text{GPT}}$	0.860	0.893	0.953	**0.945**	0.812	0.855	0.938	0.943	0.915	0.946	0.982	0.736

Table 2: Multi-turn hypothesis generation results. All values are means; across-case standard deviations are in Appendix D. (**Bold**: best; Underline: runner-up.)

Method	BioKG				PharmKG8k				DBpedia50			
	Jaccard	Dice	Overlap	Accuracy	Jaccard	Dice	Overlap	Accuracy	Jaccard	Dice	Overlap	Accuracy
ReAct	0.317	0.329	0.383	0.052	0.279	0.294	0.326	0.062	0.386	0.391	0.399	0.098
CtrlHGen (best-of-k)	0.724	0.752	0.821	0.758	0.815	0.856	0.931	0.766	0.753	0.799	0.860	0.664
ReAct + CtrlHGen	0.611	0.624	0.662	0.816	0.659	0.678	0.731	0.773	0.707	0.720	0.744	**0.751**
HypoAgent (w/o history)	0.892	0.910	0.956	**0.847**	0.919	0.940	0.980	**0.807**	0.895	0.921	0.957	0.704
HypoAgent (w/ history)	**0.910**	**0.926**	**0.959**	0.837	**0.923**	**0.945**	**0.983**	0.805	**0.910**	**0.934**	**0.970**	0.741

obtainable from repeated sampling alone, without diagnosis or refinement; note that k matches the number of rounds rather than the total number of candidates. Finally, *ReAct + CtrlHGen* gives the same ReAct agent access to the CtrlHGen generator as an additional tool.

Table 1 reports the results, with standard deviations in Appendix Table 5. First, the ReAct agent with KG tools performs poorly on structured hypothesis generation and even falls behind symbolic search. This is consistent with prior findings (Gao et al., 2025; 2026b) that even advanced general-purpose LLMs struggle to understand and construct highly structured logical hypotheses. Second, *ReAct + CtrlHGen* is more competitive on DBpedia50, where general world knowledge is more readily available, but remains weaker than repeated CtrlHGen generation on the biomedical BioKG and PharmKG8k datasets. A likely reason is that general-purpose LLMs are more familiar with the entities and relations in DBpedia50 than with those in biomedical graphs. Third, HypoAgent achieves the best semantic similarity on all datasets and metrics, substantially outperforming the baselines, while remaining competitive in condition adherence. These gains hold across all three LLM backends, indicating that they do not depend on a particular LLM. The comparison with CtrlHGen (best-of-k) is particularly informative: although both methods produce multiple candidates, HypoAgent achieves consistently higher similarity, showing that its gains come from diagnosis-guided refinement rather than repeated sampling alone. HypoAgent also exhibits markedly lower across-case variability (Appendix Table 5), suggesting that refinement mainly rescues cases in which the initial generation fails, rather than marginally improving cases that are already well explained.

5.3 RQ2: Multi-turn Goal-oriented Hypothesis Refinement

Our multi-turn benchmark consists of scripted three-turn dialogues, where each turn targets a different reference hypothesis for the same observation set. Relative requests such as "make the logic simpler" can only be interpreted with conversational context. Since the dialogues are scripted, later requests do not adapt to each system's previous outputs. We compare baselines that differ in how they use this history. *ReAct* is equipped with the same KG tools as in the single-turn setting. *CtrlHGen* receives conditions converted from the current user request by an LLM, and can be viewed as a variant of the Hypothesis Proposal Agent. *ReAct + CtrlHGen* uses CtrlHGen as a hypothesis generation tool; since it observes tool feedback, it also has access to previously generated hypotheses.

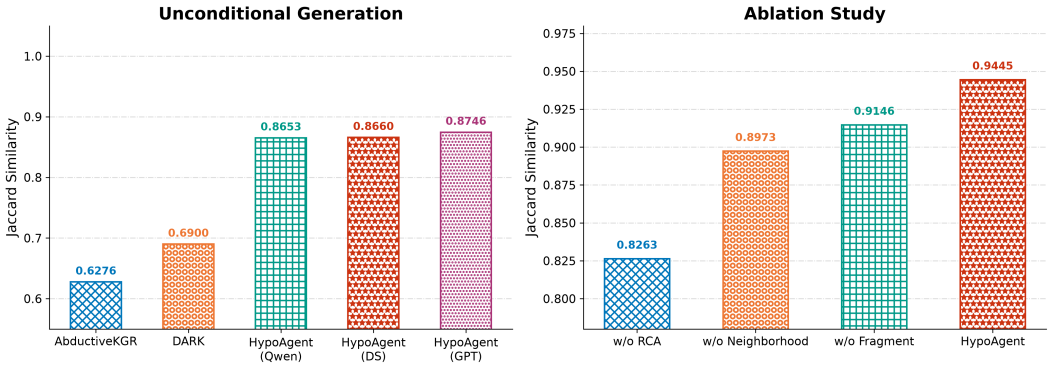


Figure 3: Jaccard similarity on DBpedia50 under the unconditional setting (left) and the single-turn ablation (right). Full ablation numbers are in Table 4.

Table 3: DBpedia50 single-turn inference usage, averaged per case, with DeepSeek-V4-Flash. Budgets are method-specific; counting details are in Appendix B.2.

Method	Budget	Remote LLM	Generations	KG exec.	Cost (CNY)	Jaccard
ReAct	10 steps	9.52	-	4.93	0.1618	0.409
ReAct + CtrlHGen	10 steps	9.33	1.32	4.53	0.1440	0.839
HypoAgent	3 rounds	**4.27**	1.76	**2.92**	**0.014**	**0.945**

We further compare two variants of HypoAgent with DeepSeek-V4-Flash, which differ in whether previously generated hypotheses are provided as references for refinement.

Table 2 reports the results, with standard deviations in Appendix Table 6. In semantic similarity, ReAct agents, even with CtrlHGen as a generation tool, underperform CtrlHGen fed directly with the converted conditions, possibly because errors in earlier hypotheses carry over to later turns. ReAct + CtrlHGen is more competitive on the commonsense DBpedia50 graph than on the biomedical BioKG and PharmKG8k graphs, and even achieves the highest condition accuracy on DBpedia50. HypoAgent remains the strongest method overall: both variants outperform CtrlHGen and ReAct + CtrlHGen in semantic similarity on all three datasets. Its margin over CtrlHGen (best-of-k) widens relative to the single-turn setting on BioKG and DBpedia50, and history-aware refinement also yields higher condition accuracy than CtrlHGen on all three datasets, indicating that tracking earlier hypotheses helps the system follow evolving requests. Providing previously generated hypotheses as references further improves semantic similarity on every dataset, e.g., raising BioKG Jaccard from 0.892 to 0.910, indicating that earlier hypotheses help guide subsequent refinement. The effect on condition adherence is mixed: accuracy decreases slightly on BioKG (0.847 to 0.837) and PharmKG8k (0.807 to 0.805) but increases on DBpedia50 (0.704 to 0.741).

5.4 RQ3: Unconditional Self-improving Hypothesis Generation

We evaluate unconditional hypothesis refinement on DBpedia50, where no user-specified conditions are given. Starting from an initial hypothesis, HypoAgent analyzes its execution results and the observed entities to induce conditions for subsequent generation. We compare HypoAgent with AbductiveKGR (Bai et al., 2024) and DARK (Gao et al., 2026b), representative autoregressive and diffusion-based abductive hypothesis generation methods, respectively.

The left panel of Fig. 3 shows the Jaccard similarity on DBpedia50. HypoAgent reaches 0.8653–0.8746 Jaccard across backends, compared with 0.6276 for AbductiveKGR and 0.6900 for DARK, showing that the framework remains effective without explicit user conditions. The agent thus derives useful guidance from its own proposals and from graph evidence. Condition induction extends the refinement framework beyond user-directed generation, allowing the system to identify promising anchors and explore alternative hypotheses on its own.

5.5 RQ4: Ablation Study

The right panel of Fig. 3 shows the DBpedia50 single-turn ablation, and Table 4 reports Jaccard similarity on all three datasets. Removing the entire RCA loop causes the largest drop, from 0.885 to 0.739 on BioKG, from 0.826 to 0.644 on PharmKG8k, and from 0.945 to 0.826 on DBpedia50, confirming the benefit of execution-guided iterative refinement over one-shot conditional generation. Removing neighborhood probing hurts more than removing fragment diagnosis on PharmKG8k and DBpedia50, while the two ablations perform nearly identically on BioKG. The full model performs best on every dataset, reflecting the complementary roles of the two components: fragment diagnosis guides revision of the current hypothesis, while neighborhood probing supplies alternative relations and anchors.

To examine the two revision routes, we remove each candidate category from the recorded candidate pools and reselect the best remaining hypothesis (Appendix E); regeneration under unchanged conditions remains available in both settings. Removing condition-update candidates reduces Jaccard by 0.128–0.148, compared with 0.032–0.091 for direct rewriting. The larger drop shows that condition updates contribute beyond repeated generation under the same constraints, while the smaller but consistent drop for direct rewriting shows the value of keeping both routes. At the sample level, removing condition-update candidates lowers test Jaccard for 44.2%, 53.0%, and 33.6% of cases on BioKG, PharmKG8k, and DBpedia50, respectively, compared with 14.0%, 15.6%, and 24.0% when direct rewriting is removed. Both routes thus provide useful alternatives for a substantial fraction of cases.

5.6 RQ5: Inference Efficiency

Table 3 compares $\text{HypoAgent}_{\text{DS}}$ with the two ReAct agents on DBpedia50 under their respective inference budgets, using DeepSeek-V4-Flash as the LLM backend for all methods. Compared with ReAct + CtrlHGen, $\text{HypoAgent}_{\text{DS}}$ achieves higher Jaccard similarity with $2.2\times$ fewer remote LLM calls and about $10.3\times$ lower cost: it raises Jaccard from 0.839 to 0.945 while reducing the average number of remote calls from 9.33 to 4.27 and the cost from 0.1440 to 0.0140 CNY per case. It also requires fewer KG tool executions, since diagnosis is invoked only when the current hypothesis falls below the threshold. Hence, giving a free-form agent access to the same generator does not by itself achieve the quality–cost trade-off of structured refinement; combining targeted condition updates with specialized generation yields stronger hypotheses at lower remote inference cost. Detailed accounting is provided in Appendix B.2.

5.7 Case Study

Appendix C presents three examples of how HypoAgent turns diagnosis into revision: targeted anchor updates on PharmKG8k, history-guided structural changes on BioKG, and condition induction without user guidance on DBpedia50. In the PharmKG8k example, adding the neighborhood-derived anchor DB11914 to the conditions yields a two-branch hypothesis whose Jaccard rises from 0.231 to 1.000, whereas directly combining this anchor with an anchor retained from the initial hypothesis scores only 0.083. The example highlights the role of the generator beyond retrieval: a promising anchor must be composed with a suitable partner, and condition updates let the generator search for this composition instead of requiring the LLM to specify it.

6 Conclusion

We proposed HypoAgent, an agentic framework for iterative abductive hypothesis generation over knowledge graphs. Its central idea is to use conditions as an interface for revision: an LLM translates execution feedback and neighborhood evidence into updated conditions, and a specialized generator turns this guidance into structured hypotheses. Direct rewriting complements condition-based generation, and previous hypotheses inform subsequent refinement. Experiments show improved semantic similarity in single-turn and multi-turn generation on BioKG, PharmKG8k, and DBpedia50, as well as in unconditional generation on DBpedia50. On DBpedia50, HypoAgent also achieves higher quality than ReAct-style agents with fewer remote LLM calls and lower cost. Together, these results demonstrate the value of coupling agentic revision planning with specialized hypothesis generation to move beyond one-shot abduction.

REFERENCES

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *international semantic web conference*, pp. 722–735. Springer, 2007.

Jiaxin Bai, Yicheng Wang, Tianshi Zheng, Yue Guo, Xin Liu, and Yangqiu Song. Advancing abductive reasoning in knowledge graphs through complex logical hypothesis generation. In *ACL (1)*, pp. 1312–1329. Association for Computational Linguistics, 2024.

Jiaxin Bai, Zihao Wang, Yukun Zhou, Hang Yin, Weizhi Fei, Qi Hu, Zheye Deng, Jiayang Cheng, Tianshi Zheng, Hong Ting Tsang, et al. Top ten challenges towards agentic neural graph databases. *arXiv preprint arXiv:2501.14224*, 2025.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen tau Yih, and Yejin Choi. Abductive commonsense reasoning, 2020. URL https://arxiv.org/abs/1908.05739.

Giacomo Camposampiero, Michael Hersche, Aleksandar Terzic, Roger Wattenhofer, Abu Sebastian, and Abbas Rahimi. Towards learning abductive reasoning using VSA distributed representations. In *NeSy (1)*, volume 14979 of *Lecture Notes in Computer Science*, pp. 370–385. Springer, 2024.

Chunkit Chan, Xin Liu, Tsz Ho Chan, Jiayang Cheng, Yangqiu Song, Ginny Wong, and Simon See. Self-consistent narrative prompts on abductive natural language inference, 2023. URL https://arxiv.org/abs/2309.08303.

Maksym Del and Mark Fishel. True detective: A deep abductive reasoning benchmark undoable for gpt-3 and challenging for gpt-4, 2023. URL https://arxiv.org/abs/2212.10114.

Nick Drummond and Rob Shearer. The open world assumption. In *eSI workshop: the closed world of databases meets the open world of the semantic web*, volume 15, pp. 1, 2006.

Eamon Duede and James Evans. The social abduction of science. *arXiv preprint arXiv:2111.13251*, 2021.

Paul Engelschalt, Maxime Röske, Johanna Penzlin, Dirk Krüger, and Annette Upmeier zu Belzen. Abductive reasoning in modeling biological phenomena as complex systems. In *Frontiers in Education*, volume 8, pp. 1170967. Frontiers Media SA, 2023.

Yisen Gao, Jiaxin Bai, Tianshi Zheng, Qingyun Sun, Ziwei Zhang, Xingcheng Fu, Jianxin Li, and Yangqiu Song. Controllable logical hypothesis generation for abductive reasoning in knowledge graphs. *arXiv preprint arXiv:2505.20948*, 2025.

Yisen Gao, Jiaxin Bai, Haoyu Huang, Zhongwei Xie, Yufei Li, Hong Ting Tsang, Sirui Han, and Yangqiu Song. Kgpfn: Unlocking the potential of knowledge graph foundation model via in-context learning. *arXiv preprint arXiv:2605.14907*, 2026a.

Yisen Gao, Jiaxin Bai, Yi Huang, Xingcheng Fu, Qingyun Sun, and Yangqiu Song. Unifying deductive and abductive reasoning in knowledge graphs with masked diffusion model. In *Proceedings of the ACM Web Conference 2026*, pp. 3600–3611, 2026b.

Brett K Hayes, Evan Heit, and Haruka Swendsen. Inductive reasoning. *Wiley interdisciplinary reviews: Cognitive science*, 1(2):278–292, 2010.

Wen-Chao Hu, Wang-Zhou Dai, Yuan Jiang, and Zhi-Hua Zhou. Efficient rectification of neuro-symbolic reasoning inconsistencies by abductive reflection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 17333–17341, 2025.

Philip N Johnson-Laird. Deductive reasoning. *Annual review of psychology*, 50(1):109–135, 1999.

Emīls Kadiķis, Vaibhav Srivastav, and Roman Klinger. Embarrassingly simple performance prediction for abductive natural language inference, 2022. URL https://arxiv.org/abs/2202.10408.

Yufei Li, Yisen Gao, Jiaxin Bai, Jiaxuan Xiong, Haoyu Huang, Zhongwei Xie, Hong Ting Tsang, and Yangqiu Song. Towards neural graph data management. *arXiv preprint arXiv:2603.05529*, 2026.

Emmy Liu, Graham Neubig, and Jacob Andreas. An incomplete loop: Instruction inference, instruction following, and in-context learning in language models, 2024. URL https://arxiv.org/abs/2404.03028.

Carlo Martini. Abductive reasoning in clinical diagnostics. In *Handbook of abductive cognition*, pp. 467–479. Springer, 2023.

Gabriele Paul. Approaches to abductive reasoning: an overview. *Artificial intelligence review*, 7(2): 109–152, 1993.

Júlia Pukancová and Martin Homola. Abductive reasoning with description logics: Use case in medical diagnosis. In *Description Logics*, volume 1350 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning, 2021. URL https://arxiv.org/abs/2010.05906.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language, 2021. URL https://arxiv.org/abs/2012.13048.

Paul Thagard. Can chatgpt make explanatory inferences? benchmarks for abductive reasoning, 2024. URL https://arxiv.org/abs/2404.18982.

Annette Upmeier zu Belzen, Paul Engelschalt, and Dirk Krüger. Modeling as scientific reasoning—the role of abductive reasoning for modeling competence. *Education Sciences*, 11(9):495, 2021.

Jay Wm Wackerly. Abductive reasoning in organic chemistry. *Journal of Chemical Education*, 98 (9):2746–2750, 2021.

Brian Walsh, Sameh K Mohamed, and Vít Nováček. Biokg: A knowledge graph for relational learning on biological data. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 3173–3180, 2020.

Zhongwei Xie, Jiaxin Bai, Shujie Liu, Haoyu Huang, Yufei Li, Yisen Gao, Hong Ting Tsang, and Yangqiu Song. Ngdb-zoo: Towards efficient and scalable neural graph databases training. *arXiv preprint arXiv:2602.21597*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *CoRR*, abs/2210.03629, 2022. doi: 10.48550/ARXIV.2210.03629. URL https://doi.org/10.48550/arXiv.2210.03629.

Wenting Zhao, Justin T Chiu, Jena D. Hwang, Faeze Brahman, Jack Hessel, Sanjiban Choudhury, Yejin Choi, Xiang Lorraine Li, and Alane Suhr. Uncommonsense reasoning: Abductive reasoning about uncommon situations, 2024. URL https://arxiv.org/abs/2311.08469.

Shuangjia Zheng, Jiahua Rao, Ying Song, Jixian Zhang, Xianglu Xiao, Evandro Fei Fang, Yuedong Yang, and Zhangming Niu. Pharmkg: a dedicated knowledge graph benchmark for bomedical data mining. *Briefings in bioinformatics*, 22(4):bbaa344, 2021.

Tianshi Zheng, Jiayang Cheng, Chunyang Li, Haochen Shi, Zihao Wang, Jiaxin Bai, Yangqiu Song, Ginny Y. Wong, and Simon See. Logidynamics: Unraveling the dynamics of logical inference in large language model reasoning, 2025. URL https://arxiv.org/abs/2502.11176.

Tianyang Zhong, Yaonai Wei, Li Yang, Zihao Wu, Zhengliang Liu, Xiaozheng Wei, Wenjun Li, Junjie Yao, Chong Ma, Xiang Li, Dajiang Zhu, Xi Jiang, Junwei Han, Dinggang Shen, Tianming Liu, and Tuo Zhang. Chatabl: Abductive learning via natural language interaction with chatgpt, 2023. URL https://arxiv.org/abs/2304.11107.

Zhi-Hua Zhou. Abductive learning: towards bridging machine learning and logical reasoning. *Sci. China Inf. Sci.*, 62(7):76101:1–76101:3, 2019.

A Data Construction

KG edges are partitioned into disjoint training, validation, and test subsets in an 8:1:1 ratio. Execution graphs are cumulative: $\mathcal{G}_{\text{train}}$ contains training edges, $\mathcal{G}_{\text{valid}}$ adds validation edges, and $\mathcal{G}_{\text{test}}$ contains all three subsets.

Following the sampling protocol of CtrlHGen (Gao et al., 2025), a query is admitted to the test set only if its full test-graph answer set contains at most 32 entities; answers are neither truncated nor subsampled. Test cases are constructed so that the target hypothesis has additional answers on the cumulative test graph that are absent from its answer sets on the training and validation graphs. The test graph is used only for data construction and final evaluation, never for candidate selection or refinement at inference time.

Evaluation target. For a sampled reference query H_{ref}, the observations are its complete test-graph answer set, $\mathcal{O} = \mathcal{A}_{\mathcal{G}_{\text{test}}}(H_{\text{ref}})$, with $|\mathcal{O}| \leq 32$. Final semantic evaluation compares $\mathcal{A}_{\mathcal{G}_{\text{test}}}(\hat{H})$ with $\mathcal{O}$, so the reference query attains a Jaccard of 1. Any alternative explanation with the same answer set is equally rewarded, and syntactic recovery of H_{ref} is not required. Since $\mathcal{O}$ contains answers that the reference query cannot retrieve on the earlier graph splits, the training-graph feedback used during inference differs from the final test-graph score.

Condition labels. Reported Accuracy is computed against the dataset's gold conditions, independently of the model's condition parsing. Conditions parsed during inference guide proposal generation and candidate ranking but are not used as evaluation labels, so parsing errors are penalized rather than hidden.

Hypothesis sampling. Given a knowledge graph G and a predefined logical pattern P, the algorithm samples a random node v as the target answer and recursively instantiates a hypothesis whose answer set contains v and whose structure matches P. At each step of the recursion, the current logical operator determines how the hypothesis is expanded. For projection, the algorithm samples an incoming edge (u, r, v) of v and recursively constructs the preceding sub-hypothesis from u. For intersection, all sub-hypotheses are instantiated with the same target node v, since v must satisfy every branch. For union, one branch is instantiated to derive v and the remaining branches are instantiated from randomly sampled nodes, since satisfying any single branch suffices for v to be in the union.

We use only the original directed edges of the knowledge graph and do not add inverse edges. The 13 predefined logical patterns used for hypothesis sampling are shown on the right. Following Gao et al. (2025), five types of conditions are extracted from each query during data processing: `entitynumber`, the number of entities in the query; `relationnumber`, the number of relations in the query; `entity`, a specific entity in the query; `relation`, a specific relation in the query; and `pattern`, the corresponding e/p query pattern.

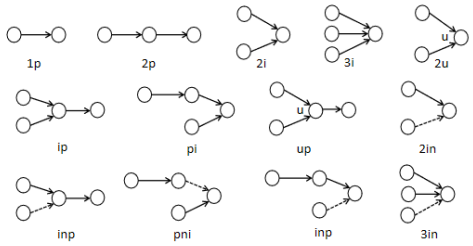


Figure 4: 13 logical patterns.

Multi-turn dialogues. For the multi-turn setting, we construct three-turn dialogues for each dataset. We first group queries by answer set and remove duplicate queries within each group. We then organize three queries from each group into a dialogue, with one query per turn.

The follow-up request at each turn follows one of three intention modes: `specific-condition`, `scope-narrowing`, and `scope-expansion`. The first turn always uses `specific-condition`, since there is no preceding query to compare against. From the second turn on, we compare the numbers of entities and relations in the current query with those in the next target query and sample a valid intention mode. We say that query A is at least as complex as query B if A has at least as many entities and at least as many relations as B, or if the total number of entities and relations in A is at least that of B. If the current query is at least as complex as the target query, `scope-expansion` is disabled; if the target query is at least as complex as the current query, `scope-narrowing` is disabled. Under `scope-narrowing`, the follow-up asks to reduce the logical complexity, e.g., "This is too complex. I want to make the logic simpler"; under `scope-expansion`, it asks to broaden the hypothesis, e.g., "I want to know more. Let's explore more." We randomly select 500 dialogues from each dataset as test samples. These dialogues are scripted from reference hypotheses rather than collected from real users. The labels `scope-narrowing` and `scope-expansion` describe changes in structural complexity and do not imply subset or superset relations between answer sets. Since a relative request may admit several plausible responses, the history comparison measures whether access to prior hypotheses helps the system reach the scripted target, rather than general dialogue understanding.

B Experiment Details

B.1 Lightweight Hypothesis Generation Model Training Details

The lightweight hypothesis generator is a 6-layer Transformer trained with the AdamW optimizer. The learning rate is set to 1×10^{-4}, and the weight decay is set to 1×10^{-5}. For DBpedia50, we train the model for 200 epochs in the unconditional setting and 100 epochs in the conditional setting. For BioKG, we train for 40 epochs in the unconditional setting and 15 epochs in the conditional setting. For PharmKG8k, we train for 100 epochs in the unconditional setting and 160 epochs in the conditional setting.

B.2 Inference Budget Accounting

All counts in Table 3 are averages per evaluated case. Each remote LLM request counts as one API call, including every request made within an RCA invocation. Generations count invocations of the hypothesis generator. KG executions count tool-level operations: a `graph_validation` call counts once even if it executes multiple sub-queries, and a neighborhood search, including its internal candidate scoring, counts as one separate operation; primitive graph queries and individual candidate evaluations inside a tool are not counted. The table reports usage under each method's configured budget rather than under matched total computation. Likewise, best-of-k sets k to the maximum number of refinement rounds and does not match the total number of candidates evaluated by HypoAgent.

B.3 Fragment Diagnosis Semantics

Fragment diagnosis reports answer counts and overlap counts, whereas final candidate validation uses Jaccard similarity. The Refiner prompts prioritize low-overlap branches for inspection as a search heuristic. How these statistics should be read depends on the logical context: a conjunctive branch may return many extra entities that another branch filters out, a disjunctive branch need not cover all observations by itself, and an intermediate projection may return entities that play a different semantic role from the final target. Fragment statistics therefore suggest where to revise, while every proposed repair is evaluated as a complete hypothesis on the training graph.

Table 4: Single-turn ablation study.

Ablation	BioKG	PharmKG8k	DBpedia50
Hypoagent	0.8846	0.8264	0.9445
w/o fragment	0.8757	0.8128	0.9146
w/o neighborhood	0.8754	0.7943	0.8973
w/o RCA	0.7390	0.6439	0.8263

B.4 Inference Conditions and Candidate Roles

Within each dialogue turn, let $\mathbf{c}^{\text{user}}$ denote the conditions extracted from the current request (resolved with conversational context), and let $\mathbf{c}_t$ denote the proposal conditions at refinement round t. The former defines the condition-adherence criterion for candidate selection, while the latter may include additional anchors suggested by the analysis. Both differ from the gold conditions used to compute Accuracy.

For conditional refinement, the Refiner prompt produces three candidate routes: regenerating with unchanged conditions, regenerating with updated conditions, and directly rewriting the hypothesis. Complete candidates are evaluated on the training graph and ranked first by adherence to $\mathbf{c}^{\text{user}}$ and then by Jaccard. The best candidate is retained across rounds, and the final output is this best-so-far hypothesis, which is not necessarily the last round's proposal. The unchanged-condition route serves as a fallback when an updated condition is unhelpful.

For unconditional generation, there is no $\mathbf{c}^{\text{user}}$. Instead, Prompt 3 induces structural, semantic, and hybrid hints from the initial query, and the resulting proposals are compared with the original unconditional output. These induced hints steer generation but are not enforced during selection, where candidates are ranked by training-graph Jaccard alone. An improved candidate may therefore use a different anchor from the one in the induced hint.

We use at most three refinement rounds and a training-graph Jaccard threshold of 0.95; RCA is triggered whenever the score of the current hypothesis falls below this threshold.

B.5 Agent Details

We report the prompts of the main LLM-based components of HypoAgent. Fig. 5 shows the condition parsing prompt used by the Hypothesis Proposal Agent to convert user utterances into structured conditions. Fig. 6 shows the Hypothesis Refiner Agent prompt for diagnosing and refining imperfect hypotheses. Fig. 7 shows the prompt for inducing conditions from unconditional hypotheses in the unconditional setting.

B.6 Ablation Study

Table 4 reports the detailed single-turn ablation results.

C Case Study

C.1 Single Turn

Figure 8 illustrates how RCA and the Hypothesis Refiner Agent improve hypothesis generation in a single-turn PharmKG8k case. Although the initial hypothesis satisfies the parsed structural constraints (two entities and two relations), its anchor entities only partially explain the observed drugs, resulting in low semantic similarity. Condition satisfaction alone is thus insufficient: the model must also identify entities that are relevant to the observation set.

The Refiner revises the hypothesis by diagnosing weak anchors and exploring the KG neighborhood. In this example, regenerating under the original condition does not improve the result. In contrast, the updated condition (Candidate 2) introduces DB11914, which neighborhood search identifies as a more informative entity, in place of the weaker anchor of the initial hypothesis. This targeted update yields a hypothesis that perfectly matches the observations. Interestingly, directly combining

Prompt 1: Hypothesis Proposal Agent

Task: You are the hypothesis proposal agent in knowledge graph abductive reasoning. Your task is to convert the user's conversation into the format suitable for the hypothesis generator, so as to use it to propose hypothesis later. Parse the following question into a JSON array of condition dicts.

Valid condition types:

- `relation`: value = ONE relation NAME, e.g. `"GG"`
- `entity`: value = ONE entity NAME, e.g. `"chrnb3"`
- `relationnumber`: integer count of relations, e.g. `"3"`
- `entitynumber`: integer count of entities, e.g. `"2"`
- `pattern`: structural pattern using only `i/u/n/p/e` tokens, e.g. `"i p e p e"`

Rules:

- NEVER use `unconditional` — always infer at least one concrete condition.
- AT MOST ONE `relation` condition and AT MOST ONE `entity` condition.
- `relationnumber`/`entitynumber` take integer values, NOT names.
- `pattern` must use only `i/u/n/p/e` tokens.
- Never output empty string as value for `relation` or `entity`.

Examples:

```
"I want a hypothesis with relation GG"
→ [{"type":"relation","value":"GG"}]
"I want pattern i p e p e with 2 entities"
→ [{"type":"pattern","value":"i p e p e"},
  {"type":"entitynumber","value":"2"}]
```

Question: *[user followup question]*

Output: Return ONLY the JSON array, nothing else.

Figure 5: Hypothesis Proposal agent prompt used to convert natural language user requests into structured condition lists.

the best neighborhood candidate with the stronger original anchor performs poorly, indicating that effective refinement requires not only retrieving promising entities but also composing them with suitable partners. Overall, this case shows that RCA turns error diagnosis into actionable condition updates, enabling the generator to revise problematic fragments rather than merely regenerating from the original instruction.

C.2 MULTI-TURN

Fig. 9 presents a multi-turn case study on BioKG. The observations are Reactome pathway or reaction entities, as indicated by their `R-HSA` identifiers. This case shows how HypoAgent progressively adapts the generated hypothesis to evolving user intents across dialogue turns.

In Turn 1, the user requests a hypothesis with three entities and the relation *MEMBER_OF_COMPLEX*. The Proposal Agent parses these constraints, and the system generates a relatively complex hypothesis involving negation and a protein–protein interaction path. Its description names only two anchors rather than the three requested, so the entity-number condition may not be fully satisfied. Although this hypothesis already achieves high semantic similarity, its complexity conflicts with the user's subsequent preference for simplicity. In Turn 2, the user states that the hypothesis is too complex and asks for simpler logic. The system interprets this relative request as reducing the number of entities while keeping the core relation *MEMBER_OF_COMPLEX*, producing a simpler hypothesis with higher semantic similarity. This is one reasonable interpretation of a simplification request among several possible ones.

Prompt 2: Hypothesis Refiner Agent

Task: You are an analysis agent for KG abductive reasoning. Given observed entities O, find a logical hypothesis H such that executing H on the KG returns exactly O. H is one of 13 patterns (`1p/2p/2i/3i/ip/pi/2u/up/2in/3in/inp/pni/pin`). Analyze why the current hypothesis is imperfect, then propose 3 candidates to refine the hypothesis.

Input:

- **Observations** (entity names): *[answer_nl]*
- **User's original condition** (must be respected): *[original_followup]*
- **Generation history**: *[round, condition, jaccard per round]*
- **Current hypothesis**: NL form, raw action string, pattern, Jaccard score
- **ID↔Name lookup**: relation and entity id-to-name maps

Step 1 — Sub-logic Decomposition (Hypothesis Fragment Diagnosis) (`graph_validation`)
Call `graph_validation(query_tokens, label_answers, split='train')`. Each sub-query result contains `answer_count`, `overlap_count`, `relation_to_label`. Identify the weakest branch (lowest `overlap_count`) to fix first. Tool budget: at most 3 calls.

Step 2 — Neighborhood Search (`incoming_edge_intersection`)
Call `incoming_edge_intersection(answer_entity_ids, split='train', top_k=10)`. Returns `flat_candidates` (1-hop entity-relation pairs with Jaccard) and `two_hop_candidates` (2-hop paths). For `2i/3i/ip/pi` patterns, also call `intersection_candidates(flat_candidates_json, mode)`.

Step 3 — Produce 3 Candidates
Candidate 1 (keep): original condition unchanged, `hypothesis_raw=null`.
Candidate 2 (update): new condition extending original with analysis findings, `hypothesis_raw=null`.
Candidate 3 (generate): directly compose a flat action string `hypothesis_raw` from tool results.
Flat format: space-separated tokens, no parentheses. Relations: negative integers (`-8`). Entities: positive integers (`1312`). E.g. `2i`: `"i -8 1312 -20 1303"`.

Output: `final_answer(candidates)` — a list of 3 dicts with keys `analysis`, `new_condition`, `hypothesis_raw`.

Figure 6: Hypothesis Refiner agent prompt used in the hypothesis refinement loop.

In Turn 3, the system interprets "know more and explore more" as a request for broader coverage rather than further simplification. It accordingly changes the pattern to a two-way union and generates a disjunctive hypothesis over two *MEMBER_OF_COMPLEX* anchors. This final hypothesis perfectly matches the observation set, showing that HypoAgent can adjust both the structural pattern and the anchor entities during multi-turn interaction. Overall, this case shows how the query structure evolves across successive requests, and also illustrates that answer-set quality and condition adherence are distinct criteria.

C.3 UNCONDITIONAL

Fig. 10 shows an unconditional generation case on DBpedia50. The observations are football players, most of whom appear in professional football transfer records. Without any user-provided condition, the initial hypothesis links the observations to the entity *List_of_Iranian_football_transfers_summer_2012* through the relation *currentMember*, but covers only a small portion of the observation set.

Prompt 3: Unconditional Condition Generation Prompt

Task: An unconditional hypothesis was generated. Analyze it and generate structural and semantic conditions to improve hypothesis quality in subsequent conditional generation.

Input:

- **Observations** (entity names): *[answer_nl]*
- **Unconditional hypothesis**: NL form, raw action string, Jaccard score
- **ID↔Name lookup**: relation id-to-name map

Step 1 — Sub-logic Decomposition((Hypothesis Fragment Diagnosis) (`graph_validation`)
Call `graph_validation` once. Use `overlap_count` per branch to identify the strongest building block (semantic anchor) and the weakest branch (structural redesign target). Count sub-queries to infer `entitynumber` and `relationnumber`.

Step 2 — Neighborhood Search (`incoming_edge_intersection`)
Call once with `top_k=10`. Use top `flat_candidates` to identify the best semantic (relation, entity) pair. Use `two_hop_candidates` to infer a good structural pattern (e.g. `2p` if top two-hop has high Jaccard).

Step 3 — Return Condition Analysis
Return a JSON object with three keys via `final_answer()`:

- **structural**: at least one of `entitynumber` (int 1–3), `relationnumber` (int 1–3), `pattern` (`i/u/n/p/e` tokens)
- **semantic**: at least one of `relation` (name string), `entity` (name string)
- **hybrid**: at least one structural key **and** at least one semantic key

Anchor conditions to the unconditional hypothesis already shown — reuse relations, entities, and structure already present rather than proposing unrelated constraints.

Figure 7: Unconditional condition generation prompt: the agent analyzes an unconditional hypothesis and produces structural, semantic, and hybrid conditions for subsequent conditional generation.

The agent then induces structural, semantic, and hybrid conditions from the initial hypothesis. The structural condition preserves only the logical form, i.e., the numbers of entities and relations and the path pattern. Since it leaves the key semantic anchor unchanged, the generated hypothesis is identical to the original one. Similarly, the hybrid condition still constrains generation to the original structure and relation, and thus fails to escape the initial local solution. In contrast, the semantic condition focuses on the relation *currentMember* and the transfer-list entity, allowing the model to search within a semantically related neighborhood. As a result, the model finds a closely related transfer list, *List_of_Iranian_football_transfers_winter_2011–12*, which better explains the observed players.

Note that the improved candidate replaces the summer-2012 anchor named in the induced hint with the winter-2011–12 list, and thus does not strictly satisfy that hint. This is acceptable in the unconditional setting, where induced hints steer generation but are not enforced during selection. Moreover, a single case does not imply that semantic hints are generally more effective than structural ones.

D Standard Deviation Tables

Table 5 reports the standard deviations for the single-turn results in Table 1, and Table 6 reports those for the multi-turn results in Table 2. These are population standard deviations across evaluated cases (divisor N, i.e., `np.std` with `ddof=0`) rather than variability across repeated runs. All metrics lie in $[0, 1]$. For binary condition adherence, the standard deviation is $\sqrt{p(1-p)}$, where p is the

Case Study: Single-turn Hypothesis Generation with RCA

Observations: DB11988, DB15253, DB12489, DB14724, DB13952, DB13953, DB13954, DB12530, DB13956, DB12089, DB15595, DB14762, DB14919
User Condition: I want a hypothesis that contains 2 entities and has 2 relations.
Parsed Conditions: `entitynumber=2`, `relationnumber=2`

Initial Generation
Hypothesis: $H = V_? : \mathit{DDI}(\mathit{DB00163}, V_?) \ \wedge \ \mathit{DDI}(\mathit{DB11972}, V_?)$
NL: Drugs that have a drug-drug interaction with both DB00163 and DB11972.
Jaccard: 0.231

Hypothesis Refiner Agent — update the condition
Condition: I want a hypothesis that contains 2 entities and has 2 relations and includes entity DB11914.
Analysis: Neighborhood search identifies DB11914 as the top flat candidate (Jaccard=0.097 via DDI). Replacing the weaker anchor DB00163 with DB11914 should improve coverage.
Hypothesis: $H = V_? : \mathit{DDI}(\mathit{DB11914}, V_?) \ \wedge \ \mathit{DDI}(\mathit{DB01590}, V_?)$
NL: Drugs that have a drug-drug interaction with both DB11914 and DB01590.
Jaccard: 1.000

Hypothesis Refiner Agent — directly refine the hypothesis
Analysis: Compose a 2i directly using the best flat candidate DB11914 and the stronger original anchor DB11972, both via DDI.
Hypothesis: $H = V_? : \mathit{DDI}(\mathit{DB11914}, V_?) \ \wedge \ \mathit{DDI}(\mathit{DB11972}, V_?)$
NL: Drugs that have a drug-drug interaction with both DB11914 and DB11972.
Jaccard: 0.083

Figure 8: Single-turn case study with Root Cause Analysis.

Table 5: Standard-deviation values for single-turn controllable hypothesis generation.

Model	BioKG				PharmKG8k				DBpedia50			
	Jaccard	Dice	Overlap	Accuracy	Jaccard	Dice	Overlap	Accuracy	Jaccard	Dice	Overlap	Accuracy
Symbolic Search	0.40	0.41	0.46	–	0.35	0.34	0.35	–	0.46	0.47	0.50	–
ReAct	0.38	0.39	0.46	0.447	0.36	0.37	0.44	0.465	0.47	0.48	0.49	0.496
CtrlHGen	0.37	0.35	0.32	0.288	0.36	0.33	0.31	0.263	0.31	0.28	0.26	0.444
CtrlHGen (best-of-k)	0.34	0.31	0.27	0.295	0.32	0.28	0.23	0.262	0.30	0.27	0.25	0.454
ReAct + CtrlHGen	0.36	0.34	0.28	0.398	0.40	0.38	0.36	0.431	0.29	0.26	0.23	0.468
$\text{HypoAgent}_{\text{DS}}$	0.22	0.18	0.13	0.236	0.25	0.20	0.13	0.247	0.16	0.13	0.10	0.435
$\text{HypoAgent}_{\text{QWEN}}$	0.24	0.20	0.14	0.237	0.25	0.20	0.15	0.230	0.20	0.15	0.11	0.443
$\text{HypoAgent}_{\text{GPT}}$	0.22	0.18	0.13	0.228	0.25	0.20	0.13	0.232	0.16	0.13	0.10	0.441

mean accuracy; since these entries are computed from the rounded means in the main tables, they are approximate.

E Post-hoc Candidate Removal Analysis

To examine the contribution of the Hypothesis Refiner Agent's candidate strategies, we remove one candidate category from the pools recorded during full-system runs and reselect the output. We distinguish **condition updates** (Candidate 2 in Fig. 6) from **direct rewriting** (Candidate 3); regen-

Case Study: Multi-turn Hypothesis Generation

Observations: R-HSA-983412, R-HSA-983413, R-HSA-983414, R-HSA-983416, R-HSA-983418, R-HSA-983419, R-HSA-983420, R-HSA-198896, R-HSA-198897, R-HSA-198904, R-HSA-198906, R-HSA-8951496, R-HSA-1236916, R-HSA-1236922, R-HSA-1236928, R-HSA-167755, R-HSA-1236930, R-HSA-1236931, R-HSA-1236934, R-HSA-983114, R-HSA-983116, R-HSA-983119, R-HSA-983120, R-HSA-983121, R-HSA-199592, R-HSA-983127, R-HSA-983132, R-HSA-983136, R-HSA-8863933

Turn 1
User Condition: I want a hypothesis that contains 3 entities and includes the relation "MEMBER_OF_COMPLEX".
Parsed Conditions: `relation=MEMBER_OF_COMPLEX`, `entitynumber=3`
Query structure: A negated membership branch intersected with a two-relation path. The description below names two anchors, not the three requested.
NL: Entities that are *not* members of the complex containing O95971, but are reachable via PPI from P30511 through MEMBER_OF_COMPLEX.
Jaccard: 0.800

Turn 2
User Condition: This is too complex. I want to make the logic simpler.
Parsed Conditions: `relation=MEMBER_OF_COMPLEX`, `entitynumber=1`
Query structure: A two-relation projection path with one named anchor.
NL: Entities reachable via PPI from P01889 through MEMBER_OF_COMPLEX.
Jaccard: 0.829

Turn 3
User Condition: I want to know more and explore more.
Parsed Conditions: `pattern=2u`
Best Hypothesis: $H = V_? : \mathit{MEMBER_OF_COMPLEX}(\mathit{P01893}, V_?) \vee \mathit{MEMBER_OF_COMPLEX}(\mathit{P10321}, V_?)$
NL: Entities that are members of the complex containing P01893 *or* P10321.
Jaccard: 1.000

Figure 9: Multi-turn case study. Path compositions are described in natural language. A high semantic score does not imply that every requested condition is satisfied.

Table 6: Standard-deviation values for multi-turn hypothesis generation.

Method	BioKG				PharmKG8k				DBpedia50			
	Jaccard	Dice	Overlap	Accuracy	Jaccard	Dice	Overlap	Accuracy	Jaccard	Dice	Overlap	Accuracy
ReAct	0.45	0.45	0.49	0.222	0.42	0.43	0.45	0.241	0.48	0.48	0.49	0.297
CtrlHGen (best-of-k)	0.39	0.39	0.37	0.428	0.29	0.26	0.23	0.423	0.34	0.32	0.32	0.472
ReAct + CtrlHGen	0.47	0.47	0.49	0.387	0.44	0.44	0.43	0.419	0.43	0.48	0.48	0.432
HypoAgent (w/o history)	0.26	0.24	0.20	0.360	0.19	0.18	0.12	0.395	0.23	0.20	0.18	0.456
HypoAgent (w/ history)	0.24	0.22	0.19	0.369	0.18	0.16	0.12	0.396	0.21	0.18	0.15	0.438

eration under unchanged conditions (Candidate 1) remains available in both settings. Reselection follows the original rule, prioritizing the original user conditions and then training-graph Jaccard, and only the selected hypothesis is evaluated on the test graph. Since this analysis is post hoc rather than a rerun of the loop with a strategy disabled, the retained candidates may still depend on the history produced by the full system.

Table 7 reports the average test-graph Jaccard for the full candidate pools and the two removal settings. Removing condition-update candidates causes larger average decreases on all datasets (0.148,

Case Study: Unconditional Generation with Condition Induction

For compact notation, let e_s denote *List_of_Iranian_football_transfers_summer_2012* and e_w denote *List_of_Iranian_football_transfers_winter_2011–12*.

Observations: Mohammad_Mansouri, Giorgi_Krasovski, Yaroslav_Krushelnitskiy, Nenad_Brnović, Mohsen_Hamidi

---

Unconditional Hypothesis
Hypothesis: $H(X) = \textit{currentMember}(e_s, X)$
NL: Entities that are current members of the Iranian football transfers list (summer 2012).
Jaccard: 0.208

Candidate 1 — Unconditional
Conditions: (none)
Hypothesis: $H(X) = \textit{currentMember}(e_s, X)$
NL: Entities that are current members of the Iranian football transfers list (summer 2012).
Jaccard: 0.208

Candidate 2 — Structural
Conditions: `entitynumber=1`, `relationnumber=1`, `pattern=p e`
Hypothesis: $H(X) = \textit{currentMember}(e_s, X)$
NL: Entities that are current members of the Iranian football transfers list (summer 2012).
Jaccard: 0.208

Candidate 3 — Semantic
Conditions: `relation=currentMember`, `entity=List_of_Iranian_football_transfers_summer_2012`
Hypothesis: $H(X) = \textit{currentMember}(e_w, X)$
NL: Entities that are current members of the Iranian football transfers list (winter 2011–12).
Jaccard: 0.833

Candidate 4 — Hybrid
Conditions: `entitynumber=1`, `relation=currentMember`, `pattern=p e`
Hypothesis: $H(X) = \textit{currentMember}(e_s, X)$
NL: Entities that are current members of the Iranian football transfers list (summer 2012).
Jaccard: 0.208

Figure 10: Unconditional generation case study.

0.132, and 0.128) than removing directly rewritten candidates (0.048, 0.032, and 0.091). Condition updates therefore contribute more to average quality, while direct rewriting provides complementary candidates.

Table 8 reports the percentage of evaluated samples whose test-graph Jaccard decreases relative to the full system after reselection. Removing condition-update candidates affects 33.6–53.0% of samples, compared with 14.0–24.0% for direct rewriting. Note that these are degradation rates rather than win rates.

Table 7: Average test-graph Jaccard after post-hoc candidate removal and training-graph-based reselection.

Setting	BioKG	PharmKG8k	DBpedia50
Full system	0.885	0.826	0.945
w/o condition updates	0.737	0.694	0.817
w/o direct rewriting	0.837	0.794	0.854

Table 8: Percentage of evaluated samples with lower test-graph Jaccard after post-hoc candidate removal, relative to the full system.

Removed category	BioKG	PharmKG8k	DBpedia50
Condition updates	44.2%	53.0%	33.6%
Direct rewriting	14.0%	15.6%	24.0%

F AI USE STATEMENT

We used generative AI tools to polish the language of the manuscript, assist with writing, and help implement experimental and analysis code. We have reviewed all AI-assisted text and code, and take full responsibility for the final content of this paper, including its claims, wording, and experimental results.