HypoAgent: An Agentic Framework for Interactive Abductive Hypothesis Generation over Knowledge Graphs

Anonymous ACL submission

Abstract

Abductive reasoning over knowledge graphs aims to generate logical hypotheses that explain observed entities or facts. Existing controllable hypothesis generation methods allow users to guide this process with explicit conditions, but they remain limited in interactive settings: they struggle to ground evolving natural-language intents across multi-turn dialogues and provide little fine-grained diagnosis when generated hypotheses fail. To address these limitations, we propose **HypoAgent**, an Agentic framework for interactive abductive Hypothesis Generation over knowledge graphs. HypoAgent integrates three agents: an Intent Recognition Agent that grounds user utterances and dialogue history into executable KG conditions, a Hypothesis Generation Agent that performs controllable hypothesis generation according to the extracted user intention, and a Root Cause Analysis Agent that diagnoses unreliable hypothesis fragments and leverages KG neighborhood probing to identify supported refinements. Experiments on commonsense and biomedical domain-specific knowledge graphs demonstrate that HypoAgent achieves state-of-the-art semantic similarity under single-turn, multi-turn, and unconditional settings.

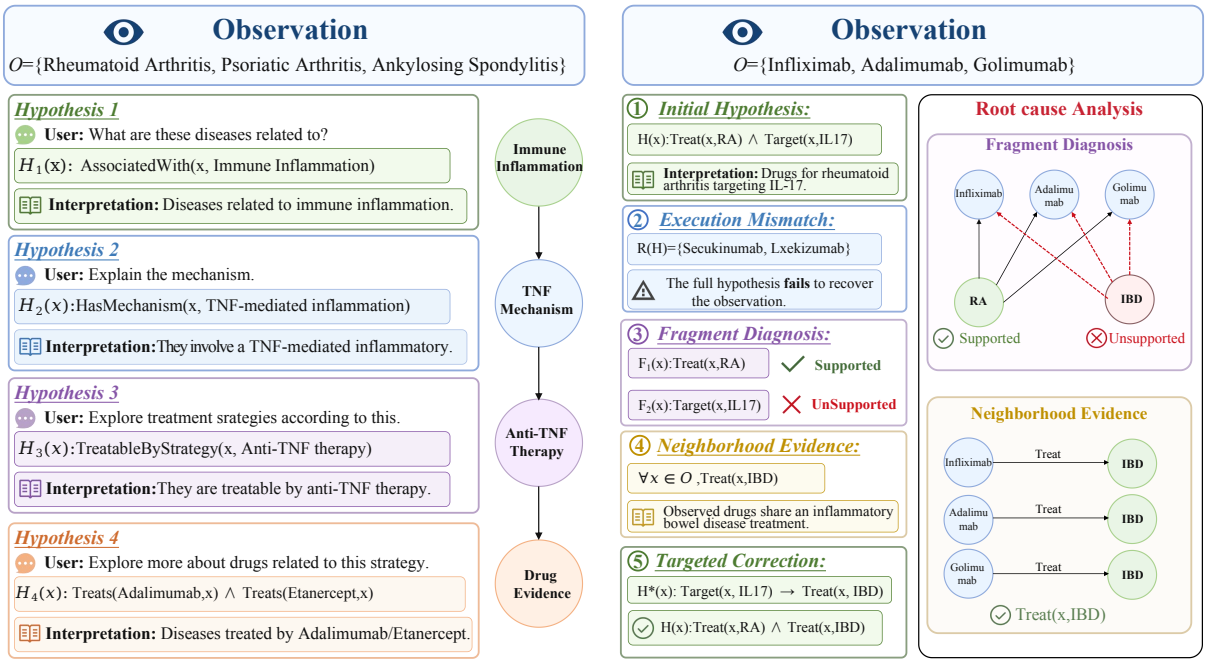
1 Introduction

Abductive reasoning (Paul, 1993) is one of the three classical modes of reasoning, together with deductive (Johnson-Laird, 1999) and inductive reasoning (Hayes et al., 2010). It seeks the most plausible explanation for observed phenomena by reasoning from evidence to explanatory hypotheses. It is widely used in clinical diagnosis (Pukancová and 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 and 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 by representing domain knowledge as interconnected entities and relations. Over a KG, abductive reasoning (Bai et al., 2024) aims to recover a first-order logic hypothesis whose answer set explains a given set of observed entities. For example, given several diseases, a system may generate hypotheses about their shared mechanisms, treatments, or susceptible populations. Such logical hypotheses are interpretable because their answer sets and supporting paths can be inspected over the KG. However, abductive reasoning over KG is challenging, as 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 through controllable hypothesis generation, where users guide generation with explicit conditions such as target entities, relations or logical patterns to obtain hypotheses that match their interests and intents.

However, existing abductive reasoning methods (Bai et al., 2024; Gao et al., 2025, 2026b) face two key challenges when used in interactive hypothesis generation.

First, they lack effective mechanisms for grounding evolving and potentially ambiguous natural-language intents in multi-turn interactions. In practice, users often begin with a broad exploration of the observations and gradually refine their needs through follow-up feedback. As shown in Fig. 1a, given the biomedical observation {rheumatoid arthritis, psoriatic arthritis, ankylosing spondylitis}, a user may first ask what these diseases are related to, then request a mechanistic explanation, and later shift toward treatment strategies or drugs. Although these turns can be mapped to executable KG conditions, the intended control conditions are not static: they change dynamically with the dialogue context, and previous interaction history can provide essential clues for interpreting the current utterance.



(a) Evolving user intent.

(b) Fine-grained failure diagnosis.

Figure 1: Two challenges in interactive abductive reasoning over knowledge graphs.

Therefore, grounding user intent requires jointly considering both the current utterance and the accumulated dialogue history. However, existing single-turn controllable generators typically require users to explicitly specify control signals. While they can handle fixed conditions such as target relations, entities, or logical patterns, they cannot interpret dynamic and context-dependent natural-language intents, such as asking to "explore more" about a previously discussed mechanism.

Second, they lack fine-grained diagnosis of abductive reasoning failures. In scientific applications, a failed hypothesis may still contain useful fragments (Okes, 2019). As shown in Fig. 1b, given the observed drugs {Infliximab, Adalimumab, Golimumab}, an abductive system may generate a hypothesis stating that these entities are drugs that treat rheumatoid arthritis and target IL-17. Executing this hypothesis may instead retrieve {Secukinumab, Ixekizumab}, indicating that the full hypothesis does not recover the original observation. However, this mismatch alone does not reveal the root cause of the failure. A fragment-level analysis can show that the treatment relation to rheumatoid arthritis is supported, while the IL-17 target fragment is unsupported for the observed drugs. Further neighborhood evidence may indicate that the observed drugs instead share an inflammatory bowel disease treatment relation. Thus, the useful correction is not to discard the entire hypothesis, but to replace the faulty fragment with a KG-supported relation, yielding a refined hypothesis involving treatment of both rheumatoid arthritis and inflammatory bowel disease. Existing methods (Bai et al., 2024; Gao et al., 2026b) usually evaluate hypotheses only through final answer-set similarity, so they cannot identify which logical fragments are correct, which fragment causes the error, or how to repair the hypothesis in a targeted way.

Our approach. To address these challenges, we propose **HypoAgent**, an **Agent**ic framework for interactive and explainable abductive **Hypo**thesis Generation over knowledge graphs. HypoAgent consists of three collaborative agents. The Intent Recognition Agent grounds each user utterance, together with the dialogue history, into executable KG conditions, enabling the system to capture evolving intents from ambiguous natural-language feedback. The Hypothesis Generation Agent invokes a controllable hypothesis generator based on the grounded conditions and maintains turn-level memory, including hypotheses, logical forms, interpretations, and evaluation results. The Root Cause Analysis Agent diagnoses failed hypotheses by decomposing them into executable fragments, checking their KG support, and probing observed-entity

neighborhoods for missing relation–entity anchors. It further proposes targeted condition corrections, allowing the system to repair unreliable fragments rather than regenerate hypotheses from scratch.

Our main contributions are summarized as follows:

- We identify two key challenges faced by existing abductive KG reasoning methods in interactive multi-turn settings: grounding evolving natural-language user intents across dialogue turns, and performing fine-grained root cause analysis for failed or partially correct hypotheses.
- We propose HypoAgent, an agentic framework that integrates intent recognition, history-aware controllable hypothesis generation, and KG-grounded root cause analysis. This design enables users to interactively refine abductive hypotheses through natural-language feedback.
- We conduct experiments on DBpedia50, PharmKG, and BioKG, covering both commonsense and biomedical domain-specific knowledge graphs. Results show that HypoAgent achieves state-of-the-art performance in semantic similarity under unconditional, single-turn, and multi-turn settings.

2 Related Work

Abductive Reasoning. Abductive reasoning aims to infer 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) introduced abductive reasoning as a commonsense reasoning task, where 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 techniques (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 and 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 aims to generate logical hypotheses that explain a set of observed entities or facts. AbductiveKGR (Bai et al., 2024) first introduced this problem by formulating KG abduction as Transformer-based hypothesis generation, where a model produces candidate logical explanations for observations. More recently, CtrlHGen (Gao et al., 2025) introduced controllable abductive hypothesis generation by conditioning the generator on user-specified control signals, enabling generated hypotheses to better satisfy desired semantic or structural constraints. DARK (Gao et al., 2026b) further unified deductive and abductive reasoning in knowledge graphs by capturing their bidirectional relationship: it treats hypotheses and conclusions as two sides of a sequence and uses masked diffusion to model reasoning in both directions. Together, these efforts have significantly advanced neural graph database reasoning (Bai et al., 2025; Li et al., 2026; Xie et al., 2026; Gao et al., 2026a) by improving the generation, control, and better understanding user query intentions.

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. Following the open-world assumption (Drummond and Shearer, 2006), unobserved triples are treated as unknown rather than false. 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 simplicity, 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 assigned to the target variable X for which $H(X)$ holds true on $\mathcal{G}$.

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 in KG aims to infer a plausible first-order logic hypothesis H that explains the observations. The quality of H is measured by the similarity between its answer set $\mathcal{A}_{\mathcal{G}}(H)$ and the observed entity set $\mathcal{O}$. That is to say, the objective is to find an optimal hypothesis H^* that maximizes the similarity between its answer set and the observed entities:

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

where the similarity function $\operatorname{sim}(\cdot, \cdot)$ can be instantiated by set-based metrics such as Jaccard similarity, Dice coefficient, and Overlap coefficient. In controllable abductive hypothesis generation (Gao et al., 2025), the model is additionally provided with a user-specified control condition $\mathbf{c}$. The goal is to generate a hypothesis H that not only explains the observed entities $\mathcal{O}$, but also satisfies the control condition $\mathbf{c}$. In this paper, we extend this setting from single-turn control to multi-turn interactive hypothesis generation. At turn t, the control condition $\mathbf{c}_t$ is grounded not only in the current user utterance u_t, but also in the interaction history $\mathcal{D}_{<t}$.

4 Method

4.1 Overview

We propose **HypoAgent**, a multi-agent framework for interactive abductive reasoning over knowledge graphs, as illustrated in Fig. 2. Given a set of observed entities $\mathcal{O}$ and a user utterance u_t at turn t, the system aims to generate a first-order logic hypothesis H_t that explains $\mathcal{O}$ while satisfying the user's control intention. At each turn, **HypoAgent** operates through three sequential components. First, the Intent Recognition Agent (in Section 4.2) interprets the user's natural-language utterance in the context of the interaction history and converts it into structured control conditions. Second, the Hypothesis Generation Agent (in Section 4.3) invokes a trained lightweight hypothesis generator to produce hypotheses under the recognized conditions. If the generated hypothesis is unsatisfactory, the Root Cause Analysis Agent (in Section 4.4) diagnoses the failure from both structural and semantic perspectives, refines the control conditions, and feeds them back into the next reasoning iteration.

4.2 Intent Recognition Agent

The Intent Recognition Agent (IRA) converts a free-form user utterance into structured control conditions that can be consumed by the hypothesis generator. Different from single-turn instruction parsing, the IRA is history-aware: it uses both the current utterance and previous interaction states to resolve vague, implicit, or relative user intentions. Specifically, at turn t, the IRA receives the observed entity set $\mathcal{O}$, the current user utterance u_t, and the interaction history $\mathcal{D}_{<t}$, where $\mathcal{D}_{<t}$ stores previously generated hypotheses, their natural-language verbalizations, recognized conditions, and execution scores. This allows the IRA to interpret utterances such as "make it simpler", "keep the entity but change the relation", or "I want to explore more about it" according to the previous hypotheses rather than treating them as isolated commands. Formally, the IRA performs the following transformation:

$$\mathrm{IRA}(\mathcal{O}, u_t, \mathcal{D}_{<t}) \rightarrow \mathbf{c}_t, \tag{2}$$

where $\mathbf{c}_t = \{(k_i, v_i)\}$ is a structured condition set. Following Gao et al. (2025), we decompose user control intentions into five condition types from both semantic and structural perspectives. Specifically, each key k_i belongs to one of five condition types: *relation*, *entity*, *relationnumber*, *entitynumber*, and *pattern*. Among them, *relation* and *entity* capture semantic constraints by specifying the desired relation and entity anchor, while *relationnumber*, *entitynumber*, and *pattern* capture structural constraints by specifying the number of relations, the number of entity anchors, and the specific logical pattern of the target hypothesis. For explicit instructions, the IRA directly extracts the corresponding values from the utterance and grounds relation/entity names to the KG vocabulary. For vague or comparative instructions, the IRA infers the intended conditions from the interaction history. For example, if the user says "make it simpler", the IRA may reduce the number of relations or entity anchors relative to the previous dialogue; if the user says "use another relation", it will preserve the structural conditions while replacing the semantic relation constraint.

4.3 Hypothesis Generation Agent

The Hypothesis Generation Agent (HGA) generates executable first-order logic hypotheses under the control conditions $\mathbf{c}_t$ recognized by the IRA.

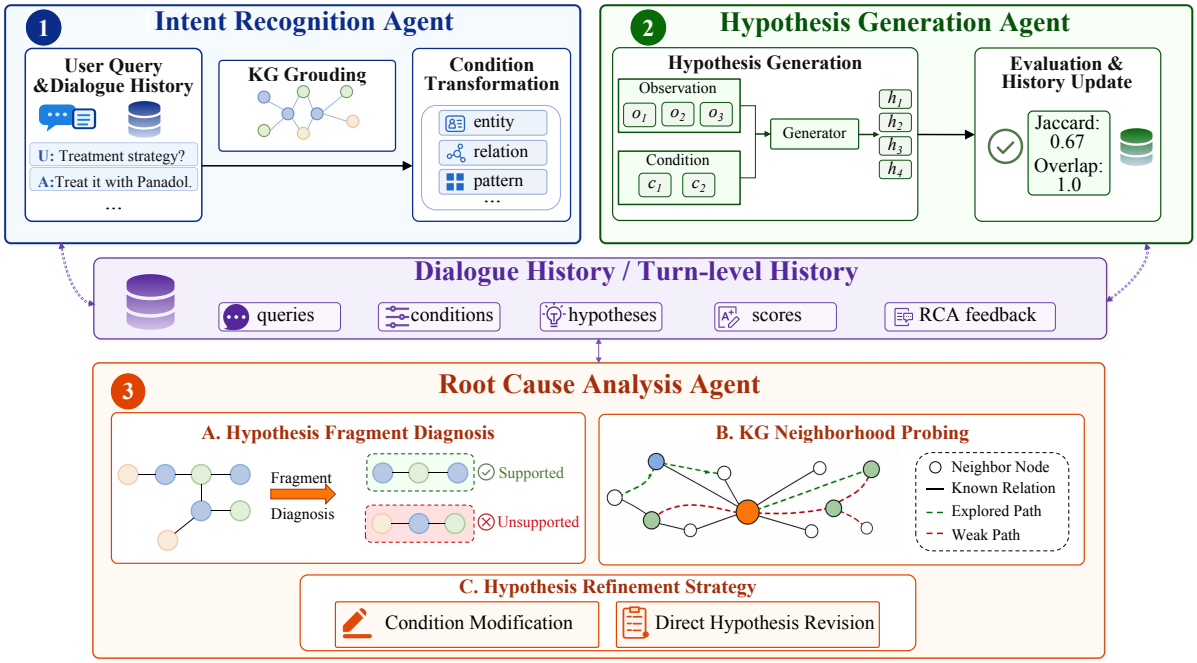


Figure 2: Overview of HypoAgent, which consists of an Intent Recognition Agent, a Hypothesis Generation Agent, and a Root Cause Analysis Agent for interactive abductive hypothesis generation over knowledge graphs.

To enable efficient and controllable generation, the HGA invokes a lightweight conditional generator p_θ. Specifically, given the observed entity set $\mathcal{O}$ and the recognized condition set $\mathbf{c}_t$, the generator produces a first-order logic hypothesis H_t. This process can be formulated as an autoregressive generation process:

$$p_\theta(H_t \mid \mathcal{O}, \mathbf{c}_t) = \prod_{\ell=1}^{L} p_\theta(h_t^\ell \mid h_t^{<\ell}, \mathcal{O}, \mathbf{c}_t), \quad (3)$$

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

We train the generator p_θ in two stages. In the first stage, we train it without conditions to learn general hypothesis generation ability from sampled observation–hypothesis pairs. In the second stage, we introduce multiple control conditions and further train the generator on observation–condition–hypothesis triples, enabling it to generate hypotheses that both explain the observed entities and satisfy the specified semantic and structural constraints.

During inference, the HGA receives $\mathbf{c}_t$ from the IRA and calls the trained generator p_θ to produce a hypothesis H_t. The generated hypothesis is executed on the KG to obtain its answer set $\mathcal{A}_\mathcal{G}(H_t)$. The quality of H_t is evaluated by comparing $\mathcal{A}_\mathcal{G}(H_t)$ with the observation set $\mathcal{O}$ using set-based metrics such as Jaccard similarity, Dice coefficient, and overlap coefficient. If the generated hypothesis reaches a satisfactory score, it is direclty returned to the user. Otherwise, the system invokes the Root Cause Analysis Agent for further diagnosis and refinement.

4.4 Root Cause Analysis Agent

The Root Cause Analysis Agent (RCAA) analyzes why a generated hypothesis fails and provides targeted refinement signals for the next reasoning iteration. Instead of simply requesting another sample from the generator, the RCAA examines the failure from two complementary perspectives: the internal structure of the generated hypothesis and the external evidence provided by the KG neighborhood of the observations. Specifically, it performs Hypothesis Fragment Diagnosis to identify reliable and unreliable parts of the current hypothesis, and Knowledge Neighborhood Probing to discover additional semantic evidence around the observed entities. Together, these two analyses support more informed refinement of the control conditions.

Hypothesis Fragment Diagnosis. Given an unsatisfactory hypothesis H_t, the RCAA decomposes it into smaller executable fragments. Each fragment corresponds to a meaningful component of

the original first-order formula, such as a single relational atom, an intermediate projection chain, or a partial conjunction of constraints. The RCAA executes each fragment on the KG and evaluates the similarity between its answer set and the observed entity set $\mathcal{O}$. Fragments with high similarity are treated as reliable components that should be preserved, whereas fragments with low similarity are regarded as potentially irrelevant or erroneous components that should be revised or removed. In this way, fragment-level diagnosis distinguishes hypothesis components that are likely to explain the observations from those that may introduce irrelevant or misleading constraints.

KG Neighborhood Probing. In parallel, the RCAA probes the local KG neighborhoods around the observed entities. The intuition is that useful semantic anchors often appear in the shared or nearby neighborhoods of the observations. The agent searches for candidate relations, entities, and short relational paths that commonly connect to the observed entities, and evaluates how well the corresponding candidate hypotheses recover $\mathcal{O}$. High-scoring neighborhood evidence suggests promising relation or entity conditions that may not have appeared in the original user utterance or the generated hypothesis.

Hypothesis Refinement Strategy. Based on Hypothesis Fragment Diagnosis and KG Neighborhood Probing, the RCAA refines hypotheses in two ways. First, it repairs the current hypothesis by preserving reliable fragments and replacing weak components with plausible relations, entities, or paths discovered from the KG neighborhood. Second, when direct repair is insufficient, it synthesizes refined control conditions that guide the generator toward a more promising hypothesis space while preserving the user's intent. For example, a strong semantic anchor found during neighborhood probing can be incorporated into the repaired hypothesis or added as a condition for the next generation step. The refined conditions are then fed back with the updated interaction history, forming an iterative reasoning loop. Thus, HypoAgent progressively improves hypotheses through user control, conditional generation, and root-cause-driven refinement.

5 Experiment

We conduct experiments to evaluate HypoAgent under single-turn, multi-turn, and unconditional abductive hypothesis generation settings. Our evaluation is designed to answer the following research questions:

- **RQ1: Single-turn controllable hypothesis generation.** Does HypoAgent improve hypothesis quality through root cause analysis when explicit generation conditions are provided?
- **RQ2: Multi-turn goal-oriented refinement.** Can HypoAgent achieve user goals by interpreting feedback and refining hypotheses across multiple dialogue turns?
- **RQ3: Unconditional self-improvement.** Can HypoAgent infer useful conditions and improve hypothesis quality when users cannot provide explicit requirements?

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), each KG is split into training, validation, and test graphs with an 8:1:1 ratio. We then sample hypothesis–observation pairs from 13 logical query patterns on these splits for training and evaluation, with at most 32 observed entities per case. The unconditional setting directly uses these sampled pairs. For single-turn and multi-turn dialogue settings, we further simulate user queries from the sampled cases: single-turn queries verbalize sampled control conditions, while multi-turn queries are built from cases with the same observations but different hypotheses to reflect progressive intent shifts. Qwen2.5-7B (Hui et al., 2024) is then used for converting structured conditions or inferred intents into natural-language user queries. Details of data construction are provided in Appendix A.

Evaluation Metrics. We evaluate generated hypotheses from two perspectives: semantic similarity and condition adherence. For semantic similarity, we compare the answer set of each generated hypothesis with the ground-truth observation set using Jaccard similarity, Dice coefficient, and overlap coefficient. For condition adherence, we report accuracy, which measures whether the generated hypothesis satisfies the specified control condition.

Baselines. For single-turn controllable hypothesis generation, we compare HypoAgent with CtrlHGen (Gao et al., 2025), the controllable hypothesis

Table 1: Single-turn controllable hypothesis generation results. (**Bold**: best; Underline: runner-up).

Model	BioKG				PharmKG8k				DBpedia50			
	Jaccard	Dice	Overlap	Acc	Jaccard	Dice	Overlap	Acc	Jaccard	Dice	Overlap	Acc
CtrlHGen	$71.8_{\pm 0.37}$	$76.1_{\pm 0.35}$	$85.8_{\pm 0.32}$	$\mathbf{90.9}_{\pm 0.27}$	$63.3_{\pm 0.36}$	$70.2_{\pm 0.33}$	$81.2_{\pm 0.31}$	$\mathbf{92.5}_{\pm 0.26}$	$78.1_{\pm 0.31}$	$82.9_{\pm 0.28}$	$89.6_{\pm 0.26}$	$\underline{73.0}_{\pm 0.44}$
$\text{HypoAgent}_{\text{DS}}$	$\mathbf{90.4}_{\pm 0.22}$	$\mathbf{92.9}_{\pm 0.18}$	$\mathbf{97.3}_{\pm 0.14}$	$\underline{90.1}_{\pm 0.30}$	$\mathbf{82.4}_{\pm 0.26}$	$\mathbf{87.6}_{\pm 0.20}$	$\underline{94.0}_{\pm 0.15}$	$\underline{92.3}_{\pm 0.27}$	$\mathbf{94.0}_{\pm 0.16}$	$\mathbf{96.0}_{\pm 0.11}$	$\mathbf{99.6}_{\pm 0.04}$	$\mathbf{74.3}_{\pm 0.44}$
$\text{HypoAgent}_{\text{QWEN}}$	$87.8_{\pm 0.24}$	$91.0_{\pm 0.20}$	$96.4_{\pm 0.14}$	$89.9_{\pm 0.30}$	$\underline{82.2}_{\pm 0.25}$	$\underline{87.5}_{\pm 0.20}$	$93.8_{\pm 0.15}$	$87.6_{\pm 0.33}$	$90.0_{\pm 0.20}$	$93.1_{\pm 0.15}$	$\underline{98.0}_{\pm 0.11}$	$69.1_{\pm 0.46}$
$\text{HypoAgent}_{\text{GPT}}$	$\underline{88.0}_{\pm 0.24}$	$\underline{91.1}_{\pm 0.20}$	$\underline{96.6}_{\pm 0.15}$	$89.9_{\pm 0.30}$	$81.9_{\pm 0.25}$	$87.4_{\pm 0.20}$	$\mathbf{94.1}_{\pm 0.15}$	$86.8_{\pm 0.34}$	$\underline{90.8}_{\pm 0.19}$	$\underline{93.7}_{\pm 0.15}$	$\underline{98.0}_{\pm 0.12}$	$70.3_{\pm 0.46}$

Table 2: Multi-turn hypothesis generation results.(**Bold**: best; Underline: runner-up).

Model	Setting	BioKG				PharmKG8k				DBpedia50			
		Jaccard	Dice	Overlap	Acc	Jaccard	Dice	Overlap	Acc	Jaccard	Dice	Overlap	Acc
$\text{HypoAgent}_{\text{DS}}$	w/o RCA	$72.4_{\pm 0.40}$	$75.2_{\pm 0.39}$	$82.1_{\pm 0.37}$	$\underline{75.8}_{\pm 0.43}$	$81.5_{\pm 0.29}$	$85.6_{\pm 0.26}$	$93.1_{\pm 0.23}$	$\mathbf{66.6}_{\pm 0.42}$	$75.3_{\pm 0.34}$	$79.9_{\pm 0.32}$	$86.0_{\pm 0.32}$	$\underline{66.4}_{\pm 0.47}$
	w/ RCA	$\mathbf{93.6}_{\pm 0.19}$	$\mathbf{95.1}_{\pm 0.17}$	$\mathbf{98.8}_{\pm 0.10}$	$\mathbf{77.5}_{\pm 0.42}$	$93.6_{\pm 0.14}$	$96.2_{\pm 0.11}$	$98.8_{\pm 0.09}$	$64.9_{\pm 0.43}$	$87.4_{\pm 0.25}$	$90.4_{\pm 0.22}$	$94.1_{\pm 0.20}$	$\mathbf{67.4}_{\pm 0.47}$
$\text{HypoAgent}_{\text{QWEN}}$	w/o RCA	$72.7_{\pm 0.39}$	$75.9_{\pm 0.37}$	$83.9_{\pm 0.35}$	$73.4_{\pm 0.44}$	$84.0_{\pm 0.27}$	$87.9_{\pm 0.24}$	$94.8_{\pm 0.20}$	$\underline{65.1}_{\pm 0.48}$	$79.1_{\pm 0.32}$	$83.3_{\pm 0.29}$	$88.8_{\pm 0.28}$	$63.3_{\pm 0.48}$
	w/ RCA	$90.9_{\pm 0.24}$	$92.5_{\pm 0.23}$	$96.7_{\pm 0.17}$	$71.7_{\pm 0.45}$	$\mathbf{94.2}_{\pm 0.13}$	$\underline{96.3}_{\pm 0.10}$	$\underline{99.0}_{\pm 0.08}$	$64.5_{\pm 0.48}$	$\underline{89.1}_{\pm 0.22}$	$\underline{92.2}_{\pm 0.18}$	$\underline{96.5}_{\pm 0.15}$	$64.0_{\pm 0.48}$
$\text{HypoAgent}_{\text{GPT}}$	w/o RCA	$74.8_{\pm 0.38}$	$77.8_{\pm 0.36}$	$85.4_{\pm 0.34}$	$71.3_{\pm 0.45}$	$84.9_{\pm 0.26}$	$88.6_{\pm 0.23}$	$95.2_{\pm 0.20}$	$63.8_{\pm 0.48}$	$74.5_{\pm 0.33}$	$79.7_{\pm 0.31}$	$87.0_{\pm 0.31}$	$61.6_{\pm 0.49}$
	w/ RCA	$\underline{93.0}_{\pm 0.20}$	$\underline{94.5}_{\pm 0.18}$	$\underline{97.1}_{\pm 0.15}$	$71.4_{\pm 0.45}$	$\mathbf{94.2}_{\pm 0.12}$	$\mathbf{96.5}_{\pm 0.10}$	$\mathbf{99.5}_{\pm 0.04}$	$64.0_{\pm 0.48}$	$\mathbf{90.7}_{\pm 0.20}$	$\mathbf{93.5}_{\pm 0.16}$	$\mathbf{97.1}_{\pm 0.13}$	$62.1_{\pm 0.48}$

generation backbone. For unconditional hypothesis generation, we compare with two representative abductive hypothesis generation methods, AbductiveKGR (Bai et al., 2024) and DARK (Gao et al., 2026b), which implement generation with autoregressive and diffusion-based architectures respectively.

Implementation Details. For the small hypothesis generation model, we follow the previous setting (Gao et al., 2025) and adopt a 6-layer Transformer architecture. 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 $\text{HypoAgent}_{\text{DS}}$, $\text{HypoAgent}_{\text{QWEN}}$, and $\text{HypoAgent}_{\text{GPT}}$, respectively. All experiments are conducted on an NVIDIA A6000 48GB GPU. More implementation details including training and agent construction are reported in Appendix B.

5.2 RQ1: Single-turn Controllable Hypothesis Generation

Table 1 reports the results of single-turn controllable hypothesis generation. HypoAgent consistently improves semantic similarity over CtrlHGen across all three datasets, with all variants achieving overlap scores above 0.9. For example, on PharmKG8k, the best HypoAgent variant improves Jaccard similarity from 63.3 to 82.4. These gains show that root-cause-driven refinement helps generate hypotheses that better explain the observations. Although HypoAgent slightly lowers condition-following accuracy on BioKG and PharmKG, this reflects a trade-off where semantic refinement may improve explanatory quality while occasionally relaxing strict condition adherence. Among LLM backends, $\text{HypoAgent}_{\text{DS}}$ performs best overall, with only small gaps among variants.

5.3 RQ2: Multi-turn Goal-oriented Hypothesis Refinement

Since user intents are more implicit and context-dependent in the multi-turn setting, we use DeepSeek-v4-Flash to judge whether generated hypotheses satisfy the intended conditions based on the full dialogue history. We have reported the details of the judgement prompt in Appendix B. Each case contains three turns, where the user progressively refines the target hypothesis. Table 2 reports the results.

Compared with single-turn evaluation, multi-turn refinement shows lower condition-adherence accuracy, reflecting the difficulty of resolving contextual intent shifts across turns. Nevertheless, RCA consistently improves semantic similarity across datasets and LLM backends, outperforming w/o RCA on Jaccard similarity, Dice coefficient, and overlap coefficient. This shows that RCA remains effective in multi-turn refinement by identifying unreliable fragments and guiding the generator toward hypotheses that better explain the observations. Among the backends, $\text{HypoAgent}_{\text{GPT}}$ achieves slightly better semantic similarity overall, while the gaps among variants remain small.

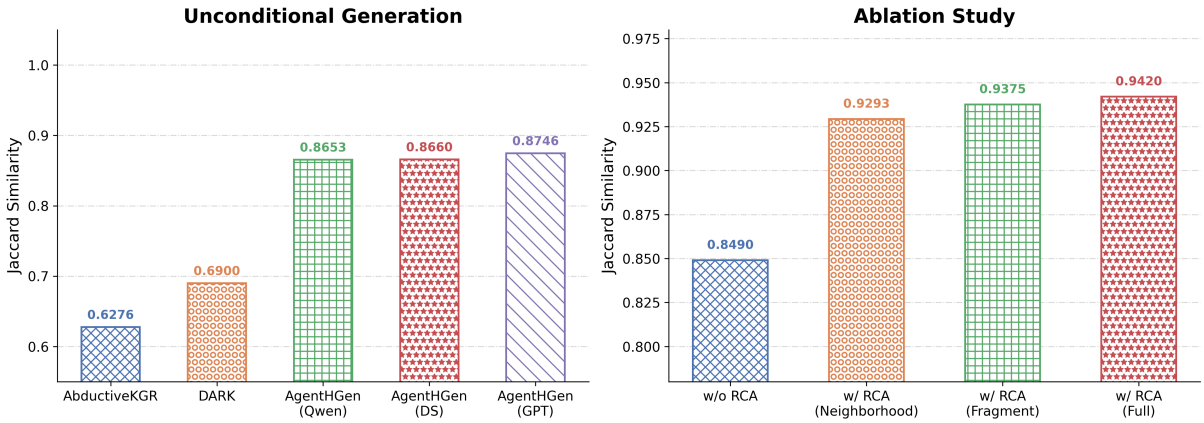


Figure 3: Jaccard similarity comparison under the unconditional setting (left) and RCA ablation study (right).

5.4 RQ3: Unconditional Self-improving Hypothesis Generation

Here, we evaluate unconditional self-improving hypothesis generation on DBpedia50. Different from the controllable settings, no user-specified conditions are provided in this experiment. The model is required to analyze its own generated hypothesis and the observed entities, construct useful refinement signals, and improve the hypothesis without external guidance. We compare HypoAgent with two baseline methods, AbductiveKGR and DARK.

Fig. 3 left presents the Jaccard similarity results on DBpedia50. Overall, HypoAgent substantially outperforms both AbductiveKGR and CtrlHGen, showing that the proposed framework remains effective even without explicit user conditions. This indicates that HypoAgent can successfully derive useful internal conditions from its own analysis and use them to guide subsequent hypothesis refinement. Therefore, the performance gain is not merely brought by external control signals, but also by the model's self-improving ability.

5.5 Ablation Study

We further conduct an ablation study on PharmKG8k using $\text{HypoAgent}_{\text{GPT}}$ to evaluate the Root Cause Analysis Agent and its two components. We compare four variants: w/o RCA, only Knowledge Neighborhood Probing, only Hypothesis Fragment Diagnosis, and the full RCA module.

The results ares shown in Fig. 3 right. Removing RCA leads to a clear performance drop, confirming its importance for hypothesis refinement. Using either Knowledge Neighborhood Probing or Hypothesis Fragment Diagnosis improves over w/o RCA, indicating that both provide useful diagnostic signals. The full RCA module performs best, showing that the two components are complementary. Moreover, removing Hypothesis Fragment Diagnosis causes a larger degradation, suggesting that direct fragment-level diagnosis is especially important for complex hypotheses.

5.6 Case study

We conduct three case studies to qualitatively examine HypoAgent under different settings, with results provided in Appendix C. The first case on PharmKG shows that RCA improves single-turn controlled generation by revising unreliable fragments with neighborhood evidence. The second case on BioKG demonstrates that HypoAgent can track evolving intents in multi-turn interaction and adapt the hypothesis structure accordingly. The third case on DBpedia50 shows that, in the unconditional setting, the model can induce useful conditions from its initial hypothesis to escape weak local solutions. Together, these cases illustrate the synergistic effects of different components in HypoAgent.

6 Conclusion

We presented HypoAgent, a multi-agent framework for interactive abductive hypothesis generation over knowledge graphs. HypoAgent extends controllable hypothesis generation from single-turn control to multi-turn interaction by recognizing user intent, generating condition-aware hypotheses, and refining unsatisfactory results through root cause analysis. Experiments on three KG benchmarks show that HypoAgent consistently improves semantic similarity while maintaining competitive condition adherence.

Limitation

HypoAgent depends on the quality and coverage of the underlying knowledge graph. If the KG is incomplete, noisy, or contains incorrect relations, both hypothesis generation and RCA refinement may be affected by unreliable structural evidence. Moreover, RCA mainly relies on local neighborhood signals, which may be insufficient for sparse graphs or explanations requiring long-range reasoning.

References

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

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

Jiaxin Bai, Zihao Wang, Yukun Zhou, Hang Yin, Weizhi Fei, Qi Hu, Zheye Deng, Jiayang Cheng, Tianshi Zheng, Hong Ting Tsang, and 1 others. 2025. Top ten challenges towards agentic neural graph databases. *arXiv preprint arXiv:2501.14224*.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen tau Yih, and Yejin Choi. 2020. Abductive commonsense reasoning. *Preprint*, arXiv:1908.05739.

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

Chunkit Chan, Xin Liu, Tsz Ho Chan, Jiayang Cheng, Yangqiu Song, Ginny Wong, and Simon See. 2023. Self-consistent narrative prompts on abductive natural language inference. *Preprint*, arXiv:2309.08303.

Maksym Del and Mark Fishel. 2023. True detective: A deep abductive reasoning benchmark undoable for gpt-3 and challenging for gpt-4. *Preprint*, arXiv:2212.10114.

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

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

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

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

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

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

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

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

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

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

Emīls Kadiķis, Vaibhav Srivastav, and Roman Klinger. 2022. Embarrassingly simple performance prediction for abductive natural language inference. *Preprint*, arXiv:2202.10408.

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

Emmy Liu, Graham Neubig, and Jacob Andreas. 2024. An incomplete loop: Instruction inference, instruction following, and in-context learning in language models. *Preprint*, arXiv:2404.03028.

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

Duke Okes. 2019. *Root cause analysis: The core of problem solving and corrective action*. Quality Press.

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

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

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

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *Preprint*, arXiv:2012.13048.

Paul Thagard. 2024. Can chatgpt make explanatory inferences? benchmarks for abductive reasoning. *Preprint*, arXiv:2404.18982.

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

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

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

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Wenting Zhao, Justin T Chiu, Jena D. Hwang, Faeze Brahman, Jack Hessel, Sanjiban Choudhury, Yejin Choi, Xiang Lorraine Li, and Alane Suhr. 2024. Uncommonsense reasoning: Abductive reasoning about uncommon situations. *Preprint*, arXiv:2311.08469.

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

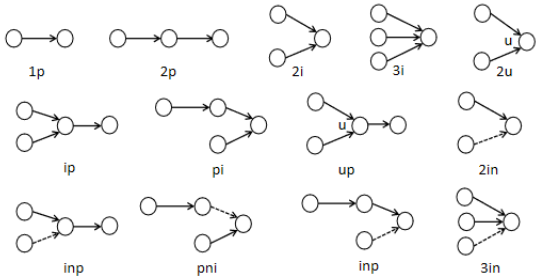


Figure 4: The 13 predefined logical patterns used for hypothesis sampling.

Tianshi Zheng, Jiayang Cheng, Chunyang Li, Haochen Shi, Zihao Wang, Jiaxin Bai, Yangqiu Song, Ginny Y. Wong, and Simon See. 2025. Logidynamics: Unraveling the dynamics of logical inference in large language model reasoning. *Preprint*, arXiv: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. 2023. Chatabl: Abductive learning via natural language interaction with chatgpt. *Preprint*, arXiv:2304.11107.

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

A Data Construction

A.1 Hypothesis-Observation Pair Construction

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. During 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, because every branch must be satisfied by v. For union, the algorithm lets one branch derive v and instantiates the remaining branches from randomly sampled nodes, since satisfying any one branch is sufficient for v to be included in the union result.

Here, for the knowledge graph, we do not augment it with additional inverse edges and only use the original directed edges. The predefined 13 logical patterns used for hypothesis sampling are illustrated in Fig 4.

A.2 Build Single-turn Dataset

We use Qwen2.5-7B-Instruct to generate the single-turn dialogue dataset, with the generation parameters set to `max_new_tokens=80`, `temperature=0.8`, `top_p=0.9`, and `do_sample` enabled. During data processing, five types of conditions are extracted from the `query`: `entitynumber`, which indicates the number of entities contained in the query; `relationnumber`, which indicates the number of relations contained in the query; `entity`, which refers to the specific entity specified in the query; `relation`, which refers to the specific relation specified in the query; and `pattern`, which represents the corresponding e/p query pattern. For each sample, two conditions are randomly selected from these five types and organized into a natural-language follow-up question. The generated sentence follows a unified template:

```
I want a hypothesis that <condition_1>
and <condition_2>.
```

A.3 Build Multi-turn Dataset

For the multi-turn setting, we construct a three-turn dialogue for each dataset. Before building the dialogues, queries with the same answer set are grouped together, and duplicate queries within each group are removed. Then, three queries from each answer group are organized into a multi-turn query list, where each turn corresponds to one query.

For each turn, the follow-up question is generated according to one of three intention modes: `specific-condition`, `scope-narrowing`, and `scope-expansion`. The first turn is fixed to `specific-condition`, since there is no preceding query for comparing complexity changes. Starting from the second turn, we compare the number of entities and relations in the current query with those in the next target query, and sample an intention mode under validity constraints. Specifically, if the current query is already at least as complex as the target query, meaning that its numbers of entities and relations are both greater than or equal to those of the target query, or that the sum of its entities and relations is greater than or equal to that of the target query, then `scope-expansion` is disabled. Conversely, if the current query is no more complex than the target query, meaning that its numbers of entities and relations are both less than or equal to those of the target query, or that the sum of its entities and relations is less than or equal to that of the target query, then `scope-narrowing` is disabled. When all three modes are valid, the program samples `specific-condition`, `scope-expansion`, and `scope-narrowing` with weights of 0.5, 0.25, and 0.25, respectively.

When the intention mode is `specific-condition`, the same five condition types and unified template described in the single-turn setting are reused. Two conditions are randomly selected and combined into one follow-up question. Example outputs include:

```
I want a hypothesis that contains 2
entities and includes the relation "CC".
```

```
I want a hypothesis that follows the
pattern "p i n p e p e" and contains 2
entities.
```

When the intention mode is `scope-narrowing`, the program directly generates:

```
This is too complex. I want to make the
logic simpler.
```

When the intention mode is `scope-expansion`, the program directly generates:

```
I want to know more. Let's explore more.
```

Finally, each follow-up question is written into the corresponding multi-turn dialogue, forming the final multi-turn dialogue dataset. We first sample a large pool of multi-turn dialogues following the above procedure. Specifically, we sample 1,951,477, 259,144, and 31,663 multi-turn dialogues for BioKG, PharmKG8k, and DBpedia50, respectively. From each dataset, we randomly select 500 dialogues as the test samples.

B Experiment Details[1]

B.1 Lightweight Hypothesis Generation Model Training Details

For the lightweight hypothesis generation model, we use a 6-layer Transformer trained with 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.

[1]The code is available at https://anonymous.4open.science/r/hypoagent-5036.

B.2 Agent Details

We report the implementation prompts for the main LLM-based components in HypoAgent. Fig. 5 shows the condition parsing prompt for Intent Recognition Agent to convert user requests into structured conditions. Fig. 6 shows the Root Cause Analysis Agent prompt for diagnosing and refining imperfect hypotheses. Fig. 7 shows the prompt for inducing conditions from unconditional hypotheses in uncondition setting. Fig. 8 shows the LLM-based judge prompt for evaluating condition satisfaction.

For RCA, we set the Jaccard similarity threshold to 0.95 and activate RCA when the score falls below this threshold.

C Case Study

C.1 Single Turn

Figure 9 illustrates how the RCA Agent improves hypothesis generation in a single-turn PharmKG case. Although the initial hypothesis satisfies the parsed structural constraints, i.e., containing two entities and two relations, its selected anchor entities only partially explain the observed drugs, leading to low semantic similarity. This shows that condition satisfaction alone is insufficient: the model must also identify entities that are truly relevant to the observation set.

The RCA Agent refines the hypothesis by diagnosing unreliable anchors and exploring the knowledge neighborhood. Simply regenerating under the original condition fails to improve the result, suggesting that repeating the same coarse constraint cannot effectively correct the underlying entity-selection error. In contrast, Candidate 2 introduces DB11914, which is identified by neighborhood search as a more informative entity, and replaces the weaker anchor in the initial hypothesis. This targeted condition update leads to a hypothesis that perfectly matches the observations. Interestingly, directly combining the best neighborhood candidate with the stronger original anchor does not yield a good result, indicating that effective refinement requires not only retrieving promising entities but also composing them with suitable partners. Overall, this case demonstrates that RCA improves HypoAgent by converting error diagnosis into actionable condition updates, enabling the generator to revise problematic hypothesis fragments rather than merely regenerating hypotheses from the original instruction.

C.2 Multiturn

Fig 10 presents a multi-turn case study on BioKG. The observations are Reactome pathway or biological reaction entities, as indicated by the `R-HSA` identifiers. This case demonstrates how HypoAgent progressively adapts the generated hypothesis according to evolving user intents across multiple dialogue turns.

In Turn 1, the user explicitly requires a hypothesis with three entities and the relation *MEMBER_OF_COMPLEX*. The Intent Recognition Agent correctly parses these constraints, and the system generates a relatively complex hypothesis involving negation and protein-protein interaction composition. Although this hypothesis already achieves high semantic similarity, its logical form is not fully aligned with the user's later preference for simplicity. In Turn 2, the user states that the hypothesis is too complex and asks for simpler logic. The system interprets this implicit instruction as reducing the entity number while preserving the core relation *MEMBER_OF_COMPLEX*, leading to a simpler hypothesis with improved semantic similarity. This shows that the model can use dialogue history to infer non-explicit refinement goals rather than relying only on literal condition extraction.

In Turn 3, the user asks to "know more and explore more", which implies broader coverage instead of further simplification. The system 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 flexibly adjust both the structural pattern and anchor entities during multi-turn interaction. Overall, this case illustrates that the proposed framework can track evolving user intents, transform them into structured conditions, and refine hypotheses from complex composition to simpler logic and finally to broader exploratory coverage.

C.3 Uncondition

Fig. 11 shows an unconditional generation case on DBpedia50. The observations are football player entities, and most of them are related to 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 only

Prompt 1: Condition Parsing

Task: 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: Condition parsing prompt used to convert natural language user requests into structured condition lists.

covers a small portion of the observation set.

The RCA Agent then induces several candidate conditions from the initial hypothesis, including structural, semantic, and hybrid constraints. The structural condition preserves only the logical form, such as the number of entities, the number of relations, and the path pattern. However, this does not change the key semantic anchor, so the generated hypothesis remains the same as the original one. Similarly, the hybrid condition still constrains the generation around the original structure and relation, and therefore 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 identifies a closely related transfer-list entity, *List_of_Iranian_football_transfers_winter_2011–12*, which better explains the observed football players.

This case demonstrates the self-improving ability of HypoAgent under the unconditional setting. Even without explicit user guidance, RCA can analyze the initial hypothesis, induce useful semantic refinement signals, and guide the generator toward a more suitable hypothesis. It also shows that semantic condition induction is particularly important in unconditional generation, since structural constraints alone may preserve the original logical form but cannot correct an inaccurate semantic anchor.

Prompt 2: Root Cause Analysis 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.

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 (`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: Root Cause Analysis (RCA) agent prompt used in the iterative hypothesis refinement loop.

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 (`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.

Prompt 4: LLM-based Condition Satisfaction Judge

Task: You are evaluating whether a knowledge graph hypothesis satisfies a user condition.

Hypothesis: *[hypothesis natural language description]*
Raw action string: *[hypothesis_raw]*

Parsed hypothesis properties:

- Logic pattern: *[pattern, e.g.* `i p e p e`*]*
- Relations (*[relationnumber]*): *[list of relation names]*
- Entities (*[entitynumber]*): *[list of entity names]*

Conversation history (previous conditions, for context only):
[list of prior turn conditions]

Current condition to judge: *[current_condition]*

Does the hypothesis satisfy the current condition? Use the history only as context to interpret what the current condition means (e.g. "make it more complex" means relative to the previous condition).

Output: Reply with ONLY a JSON object: `{"result": true}` or `{"result": false}`, nothing else.

Figure 8: LLM-based condition satisfaction judge prompt. Given a generated hypothesis and the user's current condition (with conversation history as context), the judge outputs a binary verdict on whether the hypothesis satisfies the condition.

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

RCA Agent — Candidate 1 (keep original condition)
Condition: I want a hypothesis that contains 2 entities and has 2 relations.
Hypothesis: $H = V_? : DDI(DB11972, V_?) \ \wedge \ DDI(DB06360, V_?)$
NL: Drugs that have a drug-drug interaction with both DB11972 and DB06360.
Jaccard: 0.083

RCA Agent — Candidate 2 (updated 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: $H = V_? : DDI(DB11914, V_?) \ \wedge \ DDI(DB01590, V_?)$
NL: Drugs that have a drug-drug interaction with both DB11914 and DB01590.
Jaccard: 1.000

RCA Agent — Candidate 3 (LLM directly generated)
Analysis: Compose a 2i directly using the best flat candidate DB11914 and the stronger original anchor DB11972, both via DDI.
Hypothesis: $H = V_? : DDI(DB11914, V_?) \ \wedge \ DDI(DB11972, V_?)$
NL: Drugs that have a drug-drug interaction with both DB11914 and DB11972.
Jaccard: 0.083

Figure 9: Single-turn case study with Root Cause Analysis.

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`
Best Hypothesis: $H = V_? : \neg \mathit{MEMBER_OF_COMPLEX}(\mathit{O95971}, V_?) \wedge \mathit{MEMBER_OF_COMPLEX}(\mathit{PPI}(V_?), \mathit{P30511})$
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`
Best Hypothesis: $H = V_? : \mathit{MEMBER_OF_COMPLEX}(\mathit{PPI}(V_?), \mathit{P01889})$
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 10: Multi-turn case study.

Case Study: Unconditional Generation with Condition Induction

Observations: Mohammad_Mansouri, Giorgi_Krasovski, Yaroslav_Krushelnitskiy, Nenad_Brnović, Mohsen_Hamidi

---

Unconditional Hypothesis
Hypothesis: $H = V_? : \mathit{currentMember}(\mathit{List_of_Iranian_football_transfers_summer_2012}, V_?)$
NL: Entities that are current members of the Iranian football transfers list (summer 2012).
Jaccard: 0.208

Candidate 1 — Unconditional
Conditions: (none)
Hypothesis: $H = V_? : \mathit{currentMember}(\mathit{List_of_Iranian_football_transfers_summer_2012}, V_?)$
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: $H = V_? : \mathit{currentMember}(\mathit{List_of_Iranian_football_transfers_summer_2012}, V_?)$
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: $H = V_? : \mathit{currentMember}(\mathit{List_of_Iranian_football_transfers_winter_2011\text{–}12}, V_?)$
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: $H = V_? : \mathit{currentMember}(\mathit{List_of_Iranian_football_transfers_summer_2012}, V_?)$
NL: Entities that are current members of the Iranian football transfers list (summer 2012).
Jaccard: 0.208

Figure 11: Unconditional generation case study.