

FloCA: Towards Faithful and Logically Consistent Flowchart Reasoning

Jinzi Zou¹ Bolin Wang² Shuo Zhang¹ Nuo Xu¹ Junzhou Zhao^{1*}

¹MoE KLINNS Lab, Xi'an Jiaotong University, China

²China Mobile Group Shaanxi Co., Ltd.

{jinzizou, zs412082986}@stu.xjtu.edu.cn

nxu@sei.xjtu.edu.cn, junzhou.zhao@xjtu.edu.cn

wangbolin2@sn.chinamobile.com

Abstract

Flowchart-oriented dialogue (FOD) systems aim to guide users through multi-turn decision-making or operational procedures by following a domain-specific flowchart to achieve a task goal. In this work, we formalize flowchart reasoning in FOD as grounding user input to flowchart nodes at each dialogue turn while ensuring node transition is consistent with the correct flowchart path. Despite recent advances of LLMs in task-oriented dialogue systems, adapting them to FOD still faces two limitations: (1) LLMs lack an explicit mechanism to represent and reason over flowchart topology, and (2) they are prone to hallucinations, leading to unfaithful flowchart reasoning. To address these limitations, we propose FloCA, a zero-shot flowchart-oriented conversational agent. FloCA uses an LLM for intent understanding and response generation, while delegating flowchart reasoning to an external tool that performs topology-constrained graph execution, ensuring faithful and logically consistent node transitions across dialogue turns. We further introduce an evaluation framework with an LLM-based user simulator and five new metrics covering reasoning accuracy and interaction efficiency. Extensive experiments on FLODIAL and PFDial datasets highlight the bottlenecks of existing LLM/VLM-based methods and demonstrate the superiority of FloCA. The code and dataset are publicly available at <https://github.com/Jinzi-Zou/FloCA-flowchart-reasoning>.

1 Introduction

Flowcharts are widely used to guide users through decision-making and procedural steps, especially in domains such as troubleshooting (Raghu et al., 2021, 2022; Yamanaka et al., 2025), medical diagnosis (Li et al., 2023; Kim et al., 2024; Xu et al., 2025c), and legal reasoning (McLachlan and Webley, 2021; Onami et al., 2025). Unlike textual

*Corresponding author.

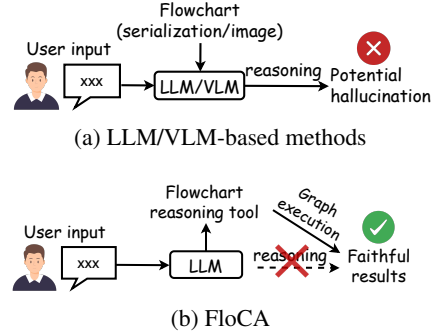


Figure 1: Comparison of workflows for the existing methods and FloCA. (a) relies on the intrinsic reasoning ability of LLMs or VLMs, which suffers from hallucination, and (b) leverages a flowchart reasoning tool to preserve topological integrity and ensure faithful and logically consistent reasoning.

guides, flowcharts often require users' strict adherence to their prescribed path to reach the final goal. For example, in troubleshooting, each node in the flowchart corresponds to a specific diagnostic question that leads to different branches. Making an incorrect judgment at any decision node can lead to an incorrect diagnosis. However, real-world flowcharts are often quite complex and domain-specific, making it difficult for non-expert users to follow them correctly without external guidance.

Raghu et al. (2021) propose FLONET, an end-to-end flowchart-based dialogue system for troubleshooting. However, FLONET lacks mechanisms to ensure the model follows the correct decision path and thus may result in unfaithful reasoning. This highlights the need for a task definition that explicitly requires faithful and logically consistent reasoning. To fill this gap, we formally define a new task of **Flowchart-Oriented Dialogue (FOD)**, specifying its core objectives, and introduce both a novel method and an evaluation framework tailored to this task.

Despite recent advances in using Large Language Models (LLMs) for task-oriented dialogue (TOD) (Li et al., 2024; Xu et al., 2025b; Baidya

et al., 2025; Acikgoz et al., 2025), applying them to FOD remains challenging, primarily due to two core difficulties. **First**, LLMs lack an explicit mechanism to represent and reason over complex flowchart topology. **Second**, LLMs are prone to struggle with complex reasoning, leading to incorrect and logically inconsistent node transitions in multi-turn dialogues.

Prior work has attempted to mitigate these issues through three main strategies: (1) *Retrieval-Augmented Generation (RAG)-based methods* (Raghu et al., 2022; Soman et al., 2025), which encode path attributes as embeddings for retrieval; (2) *Visual Language Models (VLMs)-based methods* (Singh et al., 2024; Zhang et al., 2024; He et al., 2025; Suri et al., 2025), which use VLMs to reason directly over flowchart images; (3) *Graph serialization methods* (Ye et al., 2024; Xu et al., 2025a; Yin et al., 2025), which linearize flowcharts into textual sequences to input LLMs. Among them, RAG-based methods are sensitive to retrieval accuracy; VLM-based methods may fail to ground decision logic to specific visual elements; graph serialization methods flatten the topology, forcing LLMs to reason over a lossy representation. Other efforts to enhance LLM reasoning through prompting (Liu et al., 2025; Yao et al., 2023; Besta et al., 2024), fine-tuning (Wan et al., 2024; Morishita et al., 2024; Bao et al., 2024; Zhang et al., 2025), or symbolic solvers (Pan et al., 2023; Ryu et al., 2025; Callewaert et al., 2025) either remain probabilistic, require costly data, or cannot handle interactive, multi-turn tasks. Moreover, a common thread persists, i.e., relying on the intrinsic capabilities of LLMs or VLMs does not eliminate the underlying risk of hallucination in node transitions.

To overcome the above limitations, we propose **FloCA**, a zero-shot autonomous **Flowchart-oriented Conversational Agent**. FloCA leverages an LLM for user intent understanding, semantic matching, and response generation, while delegating flowchart reasoning to a flowchart reasoning tool. This tool performs topology-constrained graph execution, ensuring faithful and logically consistent node transitions throughout the dialogue (Fig. 1). To evaluate FloCA in realistic scenarios, we also introduce a novel evaluation framework equipped with an LLM-powered user simulator and five new metrics that assess both reasoning accuracy and interaction efficiency. Extensive experiments on the FLODIAL and PFDial datasets show that FloCA significantly outperforms existing

LLM-based and VLM-based methods, establishing a strong baseline for future FOD research. Our contributions are summarized below:

- To the best of our knowledge, we are the first to formally define the task of **Flowchart-Oriented Dialogue**, where the main objective is to guide the user to make decisions and conduct operations step-by-step along the correct path within flowcharts (§3).
- We design a new flowchart-oriented conversational agent **FloCA**, which is equipped with a faithful reasoning tool to ensure logically consistent flowchart reasoning (§4).
- We propose a new evaluation framework for the FOD task, bridging the gap left by traditional TOD evaluation settings that are incompatible with the FOD task (§5).
- We conduct extensive experiments on FLODIAL and PFDial datasets. Empirical results show that FloCA achieves the highest task success rate among all LLM-based and VLM-based baselines, establishing it as a strong baseline for future research. (§6).

2 Related Work

LLMs for Flowchart Reasoning. Recent research about flowchart reasoning can be broadly divided into three categories. (1) *Visual question answering*. These methods input flowchart images into VLMs to perform logical reasoning tasks. Several benchmarks such as FlowchartQA (Tannert et al., 2023), FlowCE (Zhang et al., 2024), FlowLearn (Pan et al., 2024), SCI-CQA (Shen et al., 2024) and FlowVQA (Singh et al., 2024), evaluate VLMs’ capabilities in visual logic and spatial reasoning. TEXTFLOW (Ye et al., 2025) converts flowchart images into text via VLMs, then performs reasoning in the language space. FlowPathAgent (Suri et al., 2025) further enhances interpretability by mapping attribution paths from the answer back to relevant flowchart regions. (2) *Flowchart recognition and code generation*. These methods focus on recognizing flowchart images from user prompts and translating them into executable code in various programming languages (Herrera-Camara and Hammond, 2017; Liu et al., 2022; Shukla et al., 2023; Mentari et al., 2024; He et al., 2025). (3) *Decision-making*. Flowcharts have been leveraged to guide LLMs in multi-step decision-making

across diverse domains, including clinical diagnosis (Li et al., 2023), reasoning enhancement via visual context (Wang et al., 2024), extraction of decision flows from dialogues (Yamanaka et al., 2025), and code repair for nullability errors (Karimipour et al., 2025). Despite recent advances, existing approaches primarily treat flowcharts as enhanced external knowledge or prompting context for LLMs, which still rely on probabilistic generation and remain vulnerable to hallucination. In contrast, our work treats the flowchart as a structured reasoning tool that explicitly constrains the agent’s reasoning trajectory along the flowchart path, effectively mitigating hallucination.

Autonomous LLM-based Conversational Agents. Autonomous LLM-based agents decompose complex tasks into manageable subtasks by planning, making decisions, and invoking external tools or APIs when needed. Recent works such as ReAct (Yao et al., 2022), Reflexion (Shinn et al., 2023), Toolformer (Schick et al., 2023), AutoGPT (Richards, 2023), BabyAGI (Nakajima, 2023), AgentLM (Zeng et al., 2024), Microsoft’s Jarvis (HuggingGPT) (Shen et al., 2023), and AgentGPT (ReworkdAI, 2025), explore various architectures for enabling such capabilities. More recently, conversational agents that integrate dialogue-based planning, goal tracking, and adaptive decision-making have shown promising results in domains such as healthcare (Yang et al., 2024; Abbasian et al., 2025), medical diagnosis (Tu et al., 2025; Chen et al., 2025), recommendation (Huang et al., 2025), and TOD systems (Xu et al., 2024, 2025b; Baidya et al., 2025). However, existing work has largely overlooked the flowchart-oriented dialogue task. To fill this gap, we propose a faithful conversational agent that supports topology-constrained flowchart reasoning, offering a new conversational agent design paradigm for scenarios that demand strict adherence to procedural logic.

3 Flowchart-oriented Dialogues

We formalize the flowchart-oriented dialogue (FOD) task as follows. A flowchart is represented as a graph $\mathcal{F} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, where $\mathcal{V} = \{v_1, \dots, v_{|\mathcal{V}|}\}$ is a set of nodes, $\mathcal{E} = \{e_1, \dots, e_{|\mathcal{E}|}\}$ is a set of edges, and $\mathcal{X} = \{x_v \mid v \in \mathcal{V}\} \cup \{x_e \mid e \in \mathcal{E}\}$ denotes the textual attributes attached to nodes and edges. Each edge $e = (v_i, v_j) \in \mathcal{E}$ specifies a transition from a source node v_i to a target

node v_j , and the associated edge attribute encodes the condition required to move from v_i to v_j . This transition corresponds to a decision-making step or the execution of specific actions and procedures required to progress along the flowchart. For example, in the troubleshooting flowchart shown in Fig. 2, each node represents a diagnostic question, such as “Fuse wire blown?” and “Short after blown wire?”, associated with some text describing the trouble in detail. An edge connecting two nodes represents a possible transition during troubleshooting, e.g., if the user reports “Fuse wire has blown”, then the agent may recommend the user to check whether there is a short circuit.

A flowchart-oriented dialogue $\mathcal{D}$ consists of T turns of alternating user and agent utterances, denoted by $\mathcal{D} = \{(c_1^u, c_1^a), \dots, (c_T^u, c_T^a)\}$, where (c_t^u, c_t^a) denotes the user and agent utterances at the t -th turn. We assume the agent acts after observing the user utterance c_t^u . Given a flowchart $\mathcal{F}$ and the dialogue history up to the t -th turn, $\mathcal{D}_t = \{(c_1^u, c_1^a), \dots, (c_{t-1}^u, c_{t-1}^a), c_t^u\}$, where $1 \leq t \leq T$, the agent pursues the following two objectives.

- **Flowchart reasoning.** The agent should first ground the current dialogue context to a node $\hat{v}_t \in \mathcal{V}$ in the flowchart. The inferred node sequence across turns $(\hat{v}_1, \hat{v}_2, \dots, \hat{v}_t)$ must remain consistent with the flowchart topology, i.e., either $(\hat{v}_k, \hat{v}_{k+1}) \in \mathcal{E}$ or $\hat{v}_k = \hat{v}_{k+1}$ for all $k = 1, \dots, t-1$. For example, if $\hat{v}_k$ is node “Open circuit on run”, then the $\hat{v}_{k+1}$ can be “Fuse wire blown check” while “Short after blown wire” cannot.
- **Response generation.** Conditioned on $\mathcal{D}_t$ and the grounded node $\hat{v}_t$, the agent generates a context-aware utterance $\hat{c}_t^a$ based on the attributes $x_{\hat{v}_t}$. For example, if the grounded node is “Fuse wire blown?”, then the agent should follow the flowchart and generate a response about checking whether the fuse wire has blown.

4 FloCA: A Flowchart Reasoning Agent

We propose FloCA, a zero-shot LLM-based autonomous conversational agent for the FOD task. Fig. 2 shows the workflow of FloCA with a troubleshooting example. FloCA consists of an instruction-following LLM and a faithful flowchart reasoning tool. The key idea of FloCA is to delegate topology-constrained flowchart reasoning to the tool, while the LLM handles user intent understanding, semantic matching, and response generation. During interaction, FloCA addresses three

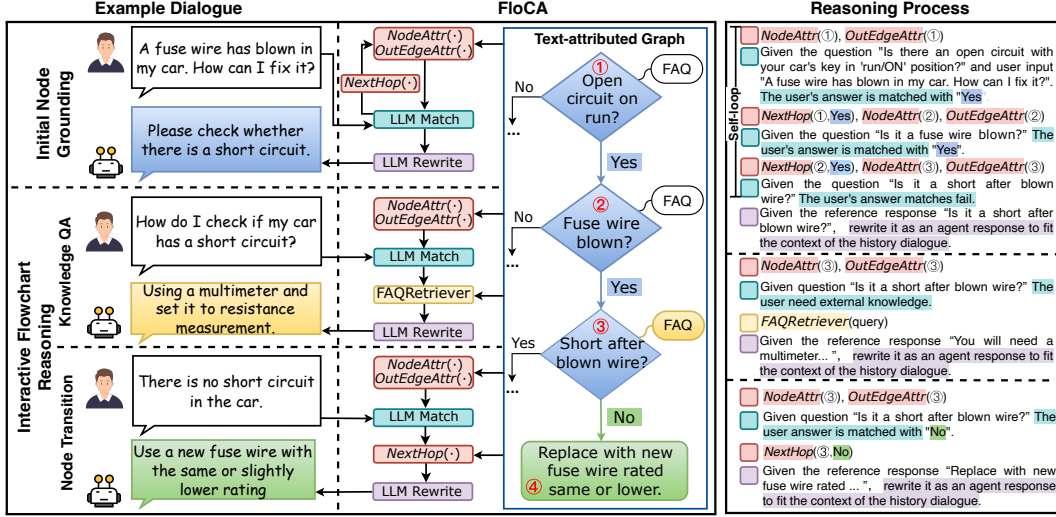


Figure 2: An overview of FloCA. FloCA consists of two core components: an instruction-following LLM and a faithful flowchart reasoning tool. The left figure shows an example dialogue in troubleshooting, where each agent output corresponds to the result of the flowchart reasoning for a specific flowchart node or an answer from FAQ database. The right figure depicts the entire multi-turn reasoning process, with colors representing the graph functions and LLM reasoning processes, showing the input and output at each step and how reasoning is carried out.

key challenges: initial node grounding, interactive flowchart reasoning, and domain knowledge QA. We first introduce the flowchart reasoning tool and then describe how FloCA addresses each challenge.

4.1 Flowchart Reasoning Tool

To enable faithful flowchart reasoning and ensure correct node grounding at each dialogue turn, we introduce a flowchart reasoning tool that guarantees deterministic transitions according to the flowchart topology. To preserve both the topological structure and textual information of the flowchart, we model the tool as a text-attributed graph with four key graph functions:

- `NodeAttr(NodeID)`: Returns the textual attribute of a given node.
- `OutEdgeAttr(NodeID)`: Returns a list of textual attributes for all outgoing edges of the given node.
- `NextHop(NodeID, EdgeAttr)`: Given a source node and the textual attribute of one outgoing edge, returns the connected target node ID.
- `TerminalCheck(NodeID)`: Returns 1 if the node is terminal and 0 otherwise.

FloCA uses the LLM to interpret user utterances, analyze intent, and determine which transition condition is supported by the user’s input, while delegating node transitions to the reasoning tool. Given the currently grounded node, FloCA uses the LLM

to rewrite the node attribute into a natural language response, which may either present a decision question or explain a required operation. We next describe how FloCA combines the LLM and the reasoning tool to handle three key dialogue challenges.

4.2 Initial Node Grounding

At the beginning of a dialogue, the user’s first utterance often conveys their request or current situation. In some cases, this utterance already provides information that implicitly resolves early decision questions, allowing the dialogue to be grounded at an intermediate node rather than always starting from the root. For example, as shown in Fig. 2, if a user reports that a fuse wire has already blown, the agent can begin reasoning from the node ③ rather than node ①. To avoid redundant interaction turns, we introduce a self-loop mechanism for initial node grounding. Internally, FloCA starts the self-loop from the root and calls the `NodeAttr` and `OutEdgeAttr` functions to retrieve the attributes of the current node and its outgoing edges. For a decision node, the LLM matches the user’s utterance against the transition conditions of the outgoing edges. For an operation node, the LLM checks whether the user has already completed the required action. If a valid transition is identified, FloCA calls `NextHop` to move to the next node and iteratively continues the grounding process. Otherwise, FloCA remains at the current node and uses the LLM to rewrite the node attribute into a context-

tually appropriate response.

This mechanism allows FloCA to make effective use of the information in the user’s initial input, thereby reducing unnecessary interactions while ensuring accurate initial grounding.

4.3 Interactive Flowchart Reasoning

After the initial node is grounded, FloCA continues the dialogue by recursively applying the same graph execution logic at each subsequent turn. Given the currently grounded node, FloCA calls `NodeAttr` and `OutEdgeAttr` to retrieve the attributes of the node and its outgoing edges and uses the LLM to determine how the dialogue should proceed. This interaction leads to three possible outcomes. (1) *First*, if the user’s utterance supports a valid outgoing transition, FloCA calls `NextHop` to move to the next node and continues the dialogue from there. (2) *Second*, if no valid transition is supported, FloCA remains at the current node and further clarifies the current decision question or required operation. (3) *Third*, the user may be unable to answer the current decision question because resolving it requires domain-specific knowledge beyond the user’s expertise. For example, as shown in Fig. 2, if the agent asks the user to check for a short circuit in a car, the user may not know how to perform the check and instead ask a question such as “How do I check whether my car has a short circuit?” In such cases, FloCA uses the LLM to identify the user’s utterance as a domain-specific knowledge query, suspends node transition, and calls a retriever to fetch the relevant answer from an external FAQ database. During this process, the currently grounded node remains unchanged until the user provides the information required to determine a valid next-hop transition. FloCA then resumes flowchart reasoning from that node.

The above interaction continues until `TerminalCheck` returns 1, indicating that a terminal node has been reached.

5 Evaluation Framework for FOD Task

Prior works (Raghu et al., 2021, 2022) primarily evaluate flowchart TOD systems by comparing semantic similarity between generated responses and annotated dialogues. However, these methods overlook the agent’s ability to follow the correct flowchart path. As we explained in §3, faithful and logically consistent flowchart reasoning is critical in addressing the FOD task. To overcome these lim-

itations, we propose a new evaluation framework that incorporates an instruction-following LLM-based user simulator to support an interactive reasoning environment. Meanwhile, we also introduce five novel metrics to measure both reasoning accuracy and interactive efficiency.

5.1 User Simulator

Inspired by prior work (Xu et al., 2024), we construct the user simulator based on an instruction-following LLM and design instruction prompts. For each annotated dialogue sample, we extract user-known facts that are established at the start of the conversation from the dialogue, including observable situations and completed operations, and use them as scenario context in the prompt. These facts specify only the user’s observable situation. To further promote natural user language and realistic questioning behavior, we also include an annotated reference dialogue aligned with the scenario context. The full prompt template and the process for extracting user-known facts are provided in Appendix A.4.

5.2 Flowchart Reasoning Metrics

We evaluate the agent’s ability in flowchart reasoning from three key aspects, i.e., *initial node reasoning*, *terminal node reasoning*, and *path coverage*, which together assess both the accuracy and success of the agent’s flowchart reasoning. Let N denote the total number of test dialogue samples. For the i -th sample, let $\mathcal{P}_i = (\hat{v}_{i,1}, \dots, \hat{v}_{i,\hat{T}_i})$ and $\mathcal{G}_i = (v_{i,1}, \dots, v_{i,T_i})$ be the predicted and ground-truth node sequences, respectively, where $\hat{T}_i$ and T_i denote the number of interaction turns in the predicted and ground-truth dialogue.

Initialization Metrics measure how accurate the agent can identify the initial node in the flowchart since each dialogue may not always start from the root node. Specifically, we introduce initial node grounding accuracy.

• **Initial Node Grounding Accuracy (INGA)** measures whether the agent identifies the correct initial node given the user’s first utterance, i.e.,

$$\text{INGA} \triangleq \frac{1}{N} \sum_{i=1}^N \mathbf{1}(\hat{v}_{i,1} = v_{i,1}), \quad (1)$$

where $\mathbf{1}(\cdot)$ is the indicator function. Higher INGA suggests more reliable flowchart reasoning initialization. This metric is crucial for decision-making

flowcharts, since an incorrect initial node prediction can route the dialogue to a wrong branch and lead to task failure.

Task Success Metrics assess the accuracy of the predicted node sequences against the ground truth. Specifically, we introduce two task success metrics.

• **Terminal Node Grounding Accuracy (TNGA)** measures whether the agent reaches the correct terminal node, i.e.,

$$\text{TNGA} \triangleq \frac{1}{N} \sum_{i=1}^N \mathbf{1}(\hat{v}_{i, \hat{T}_i} = v_{i, T_i}). \quad (2)$$

A terminal node represents either the final solution or the last required step of an operational procedure. Higher TNGA indicates a higher likelihood of successful task completion.

• **Path Coverage Accuracy (PCA)** measures whether the predicted node sequence covers all ground truth nodes in order. Note that only predicting the correct terminal node is insufficient to reflect task success. For example, in a passport application process, which involves legally mandated steps, PCA is essential because skipping an intermediate requirement, such as submitting a photo or providing a signature, invalidates the entire application. Formally, we define PCA by

$$\text{PCA} \triangleq \frac{1}{N} \sum_{i=1}^N \mathbf{1}(\mathcal{G}_i \sqsubseteq \mathcal{P}_i), \quad (3)$$

where $\mathcal{G}_i \sqsubseteq \mathcal{P}_i$ indicates that $\mathcal{G}_i$ is a subsequence of $\mathcal{P}_i$. Higher PCA demonstrating better logical consistency in flowchart reasoning.

Efficiency Metrics measures the flowchart interaction efficiency. Specifically, we introduce two efficiency metrics.

• **Node Stay Redundancy (NSR)** measures how often the agent stays at the same flowchart node due to repeated clarifications within one-hop transition. We define NSR by

$$\text{NSR} \triangleq \frac{1}{N} \sum_{i=1}^N \frac{\hat{T}_i - K_i}{\hat{T}_i}, \quad (4)$$

where K_i is the number of predicted node transitions in the i -th sample, and $\hat{T}_i - K_i$ counts the redundant turns (assuming each transition ideally takes one turn). If $K_i = 0$ (no transition occurs), the term equals 1. Lower NSR indicates fewer redundant turns and more efficient interaction.

• **Timeout Rate (TR)** measures the proportion of test samples that fail to reach the terminal node within a turn budget T_τ . We define TR by

$$\text{TR} \triangleq \frac{1}{N} \sum_{i=1}^N \mathbf{1}(\hat{T}_i > T_\tau). \quad (5)$$

Lower TR indicates that the agent can more reliably advance the dialogue to a terminal node within the turn budget.

6 Experiments

6.1 Settings

Datasets. We use two public datasets, i.e., FLODIAL (Raghu et al., 2021) and PFDial (Zhang et al., 2025). FLODIAL contains 12 troubleshooting flowcharts and 2,738 in-domain dialogues split into 1,798/456/484 for training/validation/testing. PFDial contains 440 training flowcharts, and 80 flowcharts each for in-domain and out-of-domain testing in PlantUML format, from which we construct 513 in-domain and 416 out-of-domain synthetic dialogues based on the extracted flowchart paths. Detailed dataset statistics and construction processes are provided in Appendix A.1.

Base Models. We implement FloCA based on a diverse set of language models, including GPT-4o, GPT-5, Claude Opus 4.1, DeepSeek-R1, Qwen3, Llama-3.3-70B, Llama-3.1-8B-Instruct, and Qwen2.5-7B-Instruct.

Baselines. On FLODIAL, we compare FloCA with representative RAG-based, graph-serialization-based, RAG-enhanced graph-serialization-based, and VLM-based methods. Specifically, these include SA-FloNet (Raghu et al., 2022), graph serialization methods that directly provide serialized flowcharts to the model, RAG-enhanced graph serialization methods built on the retriever in SA-FloNet, and VLM-based methods that take flowchart images as input, including Qwen-VL-Max, GPT-4o, GPT-5, and Claude Opus 4.1. On PFDial, following its instruction-tuning setting (Zhang et al., 2025), we fine-tune Llama-3.1-8B and Qwen2.5-7B as baselines.

Evaluation Tasks. We use the metrics in §5 to evaluate the performance of FOD tasks in two settings, i.e., (1) in-domain, where the test flowcharts are seen during training; (2) out-of-domain, where the test flowcharts are unseen during training.

Implementation Details. For fair comparison, we use DeepSeek-R1 to construct the user simulator

Method		Initialization Metrics			Task Success Metrics		Efficiency Metrics	
		INGA $\uparrow$			TNGA $\uparrow$	PCA $\uparrow$	TR $\downarrow$	NSR $\downarrow$
		Root-init $\uparrow$	Middle-init $\uparrow$	Overall $\uparrow$				
SA-FloNet (Raghu et al., 2022)		93.71	41.79	67.75	1.65	0.20	0.00	23.24
Graph Serializa- tion Methods	Llama-3.3-70B	100.00	0.00	50.00	0.20	0.00	86.12	69.54
	Claude Opus 4.1	98.57	23.13	60.85	83.43	80.12	0.00	0.38
	GPT-4o	98.28	7.46	52.87	78.26	73.49	0.00	0.85
	GPT-5	94.85	<u>67.91</u>	81.38	87.16	79.29	0.62	1.15
	Qwen3	78.00	2.23	40.11	67.70	54.24	0.41	4.34
	DeepSeek-R1	89.71	15.67	52.69	76.81	66.66	0.00	1.39
RAG- enhanced Graph Serializa- tion Methods	Llama-3.3-70B	86.00	18.65	52.32	0.00	0.00	82.81	55.11
	Claude Opus 4.1	<u>98.85</u>	22.38	60.62	87.16	6.62	0.00	42.25
	GPT-4o	<u>98.85</u>	3.73	51.29	80.53	75.98	0.00	0.72
	GPT-5	94.85	<u>67.91</u>	81.38	87.16	79.29	0.62	1.15
	Qwen3	88.57	9.70	49.13	71.84	51.12	4.14	5.94
	DeepSeek-R1	93.71	14.92	54.31	76.81	66.66	1.03	1.10
Visual Lan- guage Models	Claude Opus 4.1	85.71	12.68	49.20	16.35	3.72	<u>0.20</u>	1.80
	Qwen-VL-Max	71.14	5.97	38.55	36.23	12.62	4.34	5.18
	GPT-4o	59.71	50.00	54.85	33.12	12.83	0.62	2.60
	GPT-5	82.57	73.13	<u>77.85</u>	50.10	29.39	9.10	8.45
FloCA	Llama-3.3-70B	97.42	26.11	61.77	83.85	83.01	0.00	0.86
	Claude Opus 4.1	97.42	24.62	61.02	91.30	<u>90.46</u>	0.00	<u>0.23</u>
	GPT-4o	<u>98.85</u>	25.37	62.11	<u>91.71</u>	90.88	0.00	0.15
	GPT-5	98.28	24.62	61.45	91.92	90.88	0.00	0.60
	Qwen3	97.42	22.38	59.90	<u>91.71</u>	90.88	0.00	0.64
	DeepSeek-R1	100.00	20.89	60.44	90.26	89.84	0.00	0.45

Table 1: *In-domain* FOD results on FLODIAL. All values are multiplied by 100 for better readability. “Root-init” and “Middle-init” represent the subsets where the ground-truth initial node is the root node (350 samples) and a non-root node (134 samples) in a flowchart, respectively, and “Overall” is computed on all test samples. “Task Success Metrics” serve as the primary indicators of overall model performance. The **bold font** highlights the best results, and the underline highlights the second-best results.

across all evaluations and set the dialogue turn budget T_τ to twice the number of ground-truth turns for each test sample. We train SA-FloNet and fine-tune baselines on PFDial using the official hyperparameter settings from the original papers. For FLODIAL, all methods use the same SA-FloNet retriever to handle domain-knowledge QA so that performance differences mainly reflect flowchart reasoning ability rather than retrieval variation.

6.2 Results on FLODIAL Dataset

Table 1 reports the results of the FOD task on FLODIAL. We present the main observations below.

O1: FloCA achieves the highest overall task success. Although VLMs and graph-serialization methods obtain higher INGA, they perform much worse on TNGA and PCA. This is because taking the entire flowchart as input may help global semantic matching for initial node grounding, but subsequent reasoning still depends on probabilistic generation, requiring the model to implicitly maintain flowchart topology in language space. This often leads to semantically plausible but topologically invalid transitions, reducing task success. In contrast, FloCA uses a self-loop mechanism to ground the initial node more conservatively. When evidence is insufficient, it may remain at a node preceding

the ground-truth initial node, which can slightly lower INGA but avoids skipping required steps and thus improves PCA. Moreover, FloCA performs reasoning through topology-constrained graph execution, explicitly preserving flowchart topology and yielding the most reliable end-to-end success.

O2: Decoupling complex reasoning from LLMs reduces repeated clarification. FloCA achieves $TR = 0$ and the lowest NSR, indicating that it completes reasoning within the dialogue budget without repetitive clarification. Most baselines show much higher NSR, and RAG-based enhancement can even worsen repetition for strong LLMs such as Claude Opus 4.1, suggesting that misaligned retrieval may interfere with reasoning. FloCA avoids this by decoupling semantic understanding from flowchart reasoning, where LLM handles intent understanding, edge matching, and response generation, while the reasoning tool performs topology-constrained transitions. This separation improves robustness and supports stable, logically consistent progression across dialogue turns.

Flowchart Reasoning Accuracy around Domain Knowledge QA. We evaluate whether an agent can correctly resume flowchart reasoning after domain knowledge QA interruptions. Since domain knowledge QA in FOD is intended to resolve user

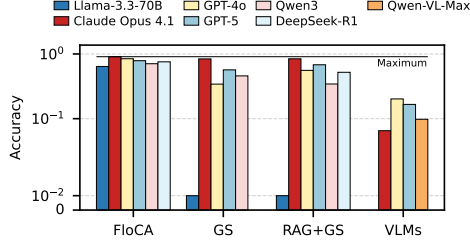


Figure 3: Flowchart reasoning accuracy around domain knowledge QA on FLODIAL. “GS”, “RAG+GS” and “VLMs” denote baselines of graph serialization methods, RAG-enhanced graph serialization methods, and visual language models.

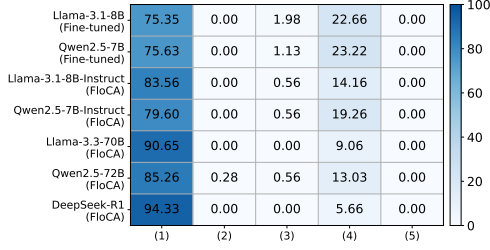


Figure 4: Distribution of path-coverage relations on PFDial in the *in-domain* setting. Each value denotes the percentage of test samples belonging to the corresponding category.

confusion and support dialogue progression, the key ability is whether the agent can return to the pre-QA flowchart state and make the correct next-hop transition after a single-turn or multi-turn QA exchange. As shown in Fig. 3, FloCA consistently achieves the highest accuracy across all compared baselines. For LLM-based baselines, the current node state is only implicitly represented in the dialogue history, so intermediate QA exchanges can interfere with the model’s ability to recover the pre-QA flowchart state, leading to incorrect next-hop transitions. In contrast, FloCA performs flowchart reasoning through explicit graph execution, which preserves node state independently of the dialogue context. This allows FloCA to handle out-of-flowchart knowledge questions while accurately resuming the interrupted reasoning process.

6.3 Results on PFDial Dataset

Table 2 compares FloCA with baselines fine-tuned on PlantUML flowcharts in the in-domain setting. Here we mainly answer the following question: **Does SFT on serialized flowcharts enable reliable multi-turn flowchart reasoning in LLMs?** The results suggest that SFT alone is insufficient for reliable multi-turn flowchart reasoning. Although the fine-tuned baselines achieve TNGA comparable to FloCA, their PCA remains consistently lower than FloCA under similar model sizes. This gap

Method	Reasoning Metrics			Efficiency Metrics	
	INGA $\uparrow$	TNGA $\uparrow$	PCA $\uparrow$	TR $\downarrow$	NSR $\downarrow$
<i>Fine-tuned on PFDial</i>					
Llama-3.1-8B	100.00	92.63	75.35	0.00	<u>0.77</u>
Qwen2.5-7B	100.00	95.18	75.63	0.00	0.80
<i>FloCA</i>					
Llama-3.1-8B-Instruct	100.00	92.91	83.56	0.00	10.66
Qwen2.5-7B-Instruct	100.00	94.05	79.60	0.00	8.33
Llama-3.3-70B	100.00	<u>97.73</u>	<u>90.65</u>	0.00	2.71
Qwen2.5-72B	100.00	94.33	85.54	0.00	2.71
DeepSeek-R1	100.00	98.30	94.33	0.00	0.60

Table 2: *In-domain* FOD results on PFDial. All values are multiplied by 100 for better readability. The **bold font** highlights the best results, and the underline highlights the second-best results.

Group	Method	TNGA (%)	PCA (%)
1	w/o self-loop mechanism	84.67	83.02
2	w/o edge matching	91.30	90.26
3	w/o FAQ interruption recovery	90.68	89.64
–	FloCA	91.92	90.88

Table 3: Ablation study of FloCA on FLODIAL using GPT-5.

indicates that SFT on serialized flowcharts may improve local semantic matching between user utterances and flowchart content, but it still fails to reliably maintain topology-consistent node transitions across dialogue turns. The out-of-domain results in Appendix A.5 further show that SFT generalizes poorly to unseen domains, with performance dropping sharply compared with the in-domain setting.

To further analyze the failure patterns, we categorize each test sample into one of five path-coverage relations between the ground-truth path and the predicted path: (1) exact match, (2) ground-truth contained in prediction, (3) prediction contained in ground-truth, (4) partial overlap without containment, and (5) disjoint paths. As shown in Fig. 4, most samples fall into group (1) or (4), indicating that many failures arise from deviating to an incorrect branch during intermediate turns. This further supports that serialized-flowchart SFT is better at learning local alignment than globally consistent multi-turn flowchart reasoning.

6.4 Ablation Study

We evaluate the efficacy of three key components in FloCA: the self-loop mechanism, edge matching for interactive flowchart reasoning, and explicit node-state preservation during FAQ interruptions. Table 3 reports the results. In ablation 1, the LLM directly predicts the initial node from the serialized flowchart. In ablation 2, the LLM predicts the next node transition based on the current node and its one-hop serialized subgraph. In ablation 3, the LLM recovers the current

Method		Failure Rate (%)
SA-FloNet (Raghu et al., 2022)		12.21
Graph Serialization Methods	Llama-3.3-70B	2.89
	Claude Opus 4.1	2.86
	GPT-4o	0.82
	GPT-5	5.17
	Qwen3	8.48
	DeepSeek-R1	1.65
RAG- enhanced Graph Serialization Methods	Llama-3.3-70B	3.51
	Claude Opus 4.1	2.71
	GPT-4o	0.72
	GPT-5	5.17
	Qwen3	4.34
	DeepSeek-R1	1.65
Visual Language Models	Claude Opus 4.1	24.22
	Qwen-VL-Max	13.87
	GPT-4o	14.28
	GPT-5	6.00
FloCA	Llama-3.3-70B	4.96
	Claude Opus 4.1	2.07
	GPT-4o	0.62
	GPT-5	1.24
	Qwen3	2.69
	DeepSeek-R1	1.24

Table 4: Task failure rates attributable to incorrect initial node grounding on FLODIAL.

Method		Failure Rate (%)
FloCA	Llama-3.3-70B	16.14
	Claude Opus 4.1	8.69
	GPT-4o	8.28
	GPT-5	8.07
	Qwen3	8.28
	DeepSeek-R1	9.52

Table 5: Task failure rates attributable to LLM edge matching errors on FLODIAL.

node state from the dialogue history and serialized flowchart. From the results, all removals degrade TNGA and PCA, with self-loop mechanism removal causing the largest drop. Instead of directly predicting the initial node from the entire serialized flowchart, it iteratively grounds from the root through topology-constrained execution, reducing ambiguity among semantically similar nodes and preventing topology-inconsistent transitions.

6.5 Flowchart Reasoning Failure Analysis

FloCA guarantees structural faithfulness by constraining node transition to the topology. However, semantic reasoning still depends on LLM-based edge matching. We analyze task failures from two perspectives: the downstream impact of incorrect initial node grounding and the remaining edge-matching errors in FloCA.

Incorrect Initial Node Grounding. Incorrect initial grounding may place the dialogue on an incorrect reasoning trajectory. We regard a task as failed when incorrect initial grounding results in completely disjoint predicted and ground-truth paths. As shown in Table 4, FloCA with GPT-4o achieves

Data Source	SE $\uparrow$	CE $\uparrow$	MSTTR $\uparrow$	MTLD $\uparrow$	HDD $\uparrow$
FLODIAL					
Annotated Dialogues	3.61	0.13	0.91	37.09	14.48
User Simulator	3.79	0.16	0.92	43.57	16.46
PFDial					
Annotated Dialogues	2.35	0.03	0.97	6.45	5.46
User Simulator	2.62	0.04	0.98	11.33	7.00

Table 6: Language diversity comparison of user utterances in the *in-domain* setting. Metrics are computed per dialogue turn and averaged over all test samples.

the lowest failure rate. Instead of inferring the initial node directly from the global flowchart, FloCA restricts each semantic decision to the outgoing edge conditions of the current node, reducing the reasoning burden on the LLM and yielding more reliable initialization.

Edge-Matching Failures. In this analysis, a task is considered failed when the predicted and ground-truth paths either partially overlap without containment or are completely disjoint. As shown in Table 5, the failure rate decreases from 16.14% with Llama-3.3-70B to 8.07% with GPT-5. This indicates that semantic edge matching remains dependent on the language understanding capability of the backbone LLM. The results reveal that even a strong LLM remains sensitive to ambiguous user input and superficial lexicality in binary decision grounding. A detailed error taxonomy with representative examples is provided in §A.2.

6.6 User Simulator Evaluation

We first verify that our user simulator does not introduce factual violations using GPT-5 as an LLM judge. Details are provided in Appendix A.4. We then evaluate the linguistic diversity of generated user utterances. As shown in Table 6, compared to static annotated dialogues, the simulator achieves higher scores across all metrics (Terragni et al., 2023), indicating richer surface realizations while preserving the underlying semantics.

7 Conclusion

We formally define the FOD task and propose FloCA, a zero-shot conversational agent that achieves faithful flowchart reasoning via a topology-constrained external tool. We further introduce a novel evaluation framework for FOD. Extensive experiments demonstrate that FloCA achieves robust and logically consistent reasoning, outperforming existing LLM/VLM-based methods.

Limitations

In this work, we propose a zero-shot conversational agent (FloCA) to address the FOD task. We analyze the limitations for future improvement: (1) *Backward transition and state correction*. FloCA currently assumes only forward progression along the flowchart path. However, real-world scenarios often require backtracking. For example, in troubleshooting, a user may report an incorrect earlier judgment, e.g., “I thought the fuse was blown, but it actually isn’t”. Handling such cases requires the agent to support backward transitions, moving from the current node to a previous node or a different branch. (2) *Complex flowchart modeling*. Our framework models flowcharts as text-attributed graphs, which assumes a single, fully-connected graph structure. However, real-world flowcharts often contain more complex elements, such as swimlanes, disconnected subgraphs, off-page connectors (linking to other flowcharts or pages). These irregularities pose challenges for integrating disparate process information into a unified graph representation. (3) *Lack of large-scale and real-world datasets*. While FloCA demonstrates strong performance on existing datasets such as FLODIAL and PFDial, these datasets are limited in scale and domain coverage. The scarcity of large-scale, real-world flowchart-oriented dialogue datasets hinders a more comprehensive evaluation of FOD.

Despite these limitations, FloCA still achieves optimal performance over existing LLM/VLM-based methods, which can be a strong baseline in future research. In future work, we will extend FloCA to more challenging FOD settings, including backtracking for revisiting earlier decisions and handling irregular flowcharts.

Acknowledgments

This work was supported in part by Xi’an Jiaotong University-China Mobile Communications Group Co.,Ltd. Joint Institute and National Natural Science Foundation of China (62272372).

References

Mahyar Abbasian, Iman Azimi, Amir M Rahmani, and Ramesh Jain. 2025. Conversational health agents: a personalized large language model-powered agent framework. *JAMIA open*, 8(4):ooaf067.

Emre Can Acikgoz, Cheng Qian, Hongru Wang, Vardhan Dongre, Xiusi Chen, Heng Ji, Dilek Hakkani-Tür,

and Gokhan Tur. 2025. A desideratum for conversational agents: Capabilities, challenges, and future directions. *arXiv:2504.16939*.

Avinash Baidya, Kamalika Das, and Xiang Gao. 2025. The behavior gap: Evaluating zero-shot llm agents in complex task-oriented dialogs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 23455–23472.

Qiming Bao, Alex Yuxuan Peng, Zhenyun Deng, Wanjun Zhong, Gael Gendron, Timothy Pistotti, Neşet Tan, Nathan Young, Yang Chen, Yonghua Zhu, et al. 2024. Abstract meaning representation-based logic-driven data augmentation for logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5914–5934.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*.

Benjamin Callewaert, Simon Vandeveld, and Joost Vennekens. 2025. Verus-llm: a versatile framework for combining llms with symbolic reasoning. *arXiv preprint arXiv:2501.14540*.

Xi Chen, Huahui Yi, Mingke You, et al. 2025. Enhancing diagnostic capability with multi-agents conversational large language models. *NPJ digital medicine*, 8(1):159.

Mengliang He, Jiayi Zeng, Yankai Jiang, Wei Zhang, Zeming Liu, Xiaoming Shi, and Aimin Zhou. 2025. Flow2code: Evaluating large language models for flowchart-based code generation capability. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8124–8146.

Jorge-Ivan Herrera-Camara and Tracy Hammond. 2017. Flow2code: from hand-drawn flowcharts to code execution. In *Proceedings of the Symposium on Sketch-Based Interfaces and Modeling*, pages 1–13.

Xu Huang, Jianxun Lian, Yuxuan Lei, et al. 2025. Recommender ai agent: Integrating large language models for interactive recommendations. *ACM Transactions on Information Systems*, 43(4):1–33.

Nima Karimipour, Michael Pradel, Martin Kellogg, and Manu Sridharan. 2025. Llm-based repair of static nullability errors. *arXiv preprint arXiv:2507.20674*.

Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik S Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae W Park. 2024. Mdagents: An adaptive collaboration of llms for medical decision-making. *Advances in Neural Information Processing Systems*, 37:79410–79452.

- Binbin Li, Tianxin Meng, Xiaoming Shi, Jie Zhai, and Tong Ruan. 2023. Meddm: Llm-executable clinical guidance tree for clinical decision-making. *arXiv preprint arXiv:2312.02441*.
- Zekun Li, Zhiyu Chen, Mike Ross, Patrick Huber, Seungwhan Moon, Zhaojiang Lin, Xin Luna Dong, Adithya Sagar, Xifeng Yan, and Paul A Crook. 2024. Large language models as zero-shot dialogue state tracker through function calling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8688–8704.
- Tongxuan Liu, Wenjiang Xu, Weizhe Huang, Yuting Zeng, Jiaxing Wang, Xingyu Wang, Hailong Yang, and Jing Li. 2025. Logic-of-thought: Injecting logic into contexts for full reasoning in large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 10168–10185.
- Zejie Liu, Xiaoyu Hu, Deyu Zhou, Lin Li, Xu Zhang, and Yanzheng Xiang. 2022. Code generation from flowcharts with texts: A benchmark dataset and an approach. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6069–6077.
- Scott McLachlan and Lisa C Webley. 2021. Visualisation of law and legal process: An opportunity missed. *Information Visualization*, 20(2-3):192–204.
- Mustika Mentari, Pramana Yoga Saputra, Yan Watequlis Syaifudin, Imam Fahrur Rozi, and Naufal Nafidiin. 2024. Automatic java code generation system from flowchart. In *Proceedings of the 5th Annual Advanced Technology, Applied Science, and Engineering Conference (ATASEC) 2023*, page 102. Springer Nature.
- Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. 2024. Enhancing reasoning capabilities of llms via principled synthetic logic corpus. *Advances in Neural Information Processing Systems*, 37:73572–73604.
- Yohei Nakajima. 2023. Babyagi. <https://github.com/yoheinakajima/babyagi>.
- Eri Onami, Taiki Miyaniishi, Koki Maeda, and Shuhei Kurita. 2025. Legalviz: Legal text visualization by text to diagram generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6657–6676.
- Huitong Pan, Qi Zhang, Cornelia Caragea, Edward Dragut, and Longin Jan Latecki. 2024. Flowlearn: Evaluating large vision-language models on flowchart understanding. In *ECAI 2024*, pages 73–80. IOS Press.
- Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824.
- Dinesh Raghu, Shantanu Agarwal, Sachindra Joshi, et al. 2021. End-to-end learning of flowchart grounded task-oriented dialogs. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 4348–4366.
- Dinesh Raghu, Suraj Joshi, Sachindra Joshi, et al. 2022. Structural constraints and natural language inference for end-to-end flowchart grounded dialog response generation. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 10763–10774.
- ReworkdAI. 2025. AgentGPT. <https://github.com/reworkd/AgentGPT>.
- Toran Bruce Richards. 2023. Auto-GPT. <https://github.com/Significant-Gravitas/Auto-GPT>.
- Hyun Ryu, Gyeongman Kim, Hyemin S. Lee, and Eunho Yang. 2025. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. In *The Thirteenth International Conference on Learning Representations*.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551.
- Lingdong Shen, Kun Ding, Gaofeng Meng, Shiming Xiang, et al. 2024. Rethinking comprehensive benchmark for chart understanding: A perspective from scientific literature. *arXiv preprint arXiv:2412.12150*.
- Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugging-gpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.
- Shreya Shukla, Prajwal Gatti, Yogesh Kumar, Vikash Yadav, and Anand Mishra. 2023. Towards making flowchart images machine interpretable. In *International Conference on Document Analysis and Recognition*, pages 505–521. Springer.
- Shubhankar Singh, Purvi Chaurasia, Yerram Varun, Pranshu Pandya, Vatsal Gupta, Vivek Gupta, and Dan Roth. 2024. Flowvqa: Mapping multimodal logic in visual question answering with flowcharts.

- In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1330–1350.
- Sumit Soman, HG Ranjani, Sujoy Roychowdhury, Venkata Dharma Surya Narayana Sastry, Akshat Jain, Pranav Gangrade, and Ayaaz Khan. 2025. A graph-based approach for multi-modal question answering from flowcharts in telecom documents. *arXiv preprint arXiv:2507.22938*.
- Manan Suri, Puneet Mathur, Nedim Lipka, Franck Deroncourt, Ryan A Rossi, Vivek Gupta, and Dinesh Manocha. 2025. Follow the flow: Fine-grained flowchart attribution with neurosymbolic agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 22496–22519.
- Simon Tannert, Marcelo G Feighelstein, Jasmina Bogojeska, Joseph Shtok, Assaf Arbelle, Peter WJ Staar, Anika Schumann, Jonas Kuhn, and Leonid Karlinsky. 2023. Flowchartqa: The first large-scale benchmark for reasoning over flowcharts. In *Proceedings of the 1st Workshop on Linguistic Insights from and for Multimodal Language Processing*, pages 34–46.
- Silvia Terragni, Modestas Filipavicius, Nghia Khau, Bruna Guedes, André Manso, and Roland Mathis. 2023. In-context learning user simulators for task-oriented dialog systems. *arXiv preprint arXiv:2306.00774*.
- Tao Tu, Mike Schaekermann, Anil Palepu, Khaled Saab, Jan Freyberg, Ryutaro Tanno, Amy Wang, Brenna Li, Mohamed Amin, Yong Cheng, et al. 2025. Towards conversational diagnostic artificial intelligence. *Nature*, 642(8067):442–450.
- Yuxuan Wan, Wenxuan Wang, Yiliu Yang, Youliang Yuan, Jen-tse Huang, Pinjia He, Wenxiang Jiao, and Michael Lyu. 2024. Logicasker: Evaluating and improving the logical reasoning ability of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2124–2155.
- Xingjin Wang, Jiahao Zhao, Jiahui Shi, Linjing Li, and Daniel Zeng. 2024. A novel visual-enhanced dual stream long-term decision framework for large language model agents. In *International Conference on Neural Information Processing*, pages 332–346. Springer.
- Hao Xu, Xiangru Jian, Xinjian Zhao, Wei Pang, Chao Zhang, Suyuchen Wang, Qixin Zhang, Zhengyuan Dong, Joao Monteiro, Bang Liu, et al. 2025a. Graphomni: A comprehensive and extendable benchmark framework for large language models on graph-theoretic tasks. *arXiv preprint arXiv:2504.12764*.
- Heng-Da Xu, Xian-Ling Mao, Fanshu Sun, Tian-Yi Che, Chun Xu, and Heyan Huang. 2025b. Agenttod: A task-oriented dialogue agent with a flexible and adaptive api calling paradigm. *ACM Transactions on Information Systems*, 43(5):1–32.
- Heng-Da Xu, Xian-Ling Mao, Puhai Yang, Fanshu Sun, and He-Yan Huang. 2024. Rethinking task-oriented dialogue systems: From complex modularity to zero-shot autonomous agent. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 2748–2763.
- Xingru Xu, Michel Dumontier, and Chang Sun. 2025c. Using clinical guidelines, domain ontology, and llms for personalized leukemia treatment recommendations. In *CEUR Workshop Proceedings*, volume 4001. CEUR-WS.
- Yuuki Yamanaka, Hiroshi Takahashi, and Tomoya Yamashita. 2025. Flowchart-based decision making with large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14836–14842.
- Ziqi Yang, Xuhai Xu, Bingsheng Yao, Ethan Rogers, Shao Zhang, Stephen Intille, Nawar Shara, Guodong Gordon Gao, and Dakuo Wang. 2024. Talk2care: An llm-based voice assistant for communication between healthcare providers and older adults. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.
- Junyi Ye, Ankan Dash, Wenpeng Yin, and Guiling Wang. 2025. Beyond end-to-end vlms: Leveraging intermediate text representations for superior flowchart understanding. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3534–3548.
- Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2024. Language is all a graph needs. In *Findings of the association for computational linguistics: EACL 2024*, pages 1955–1973.
- Haoteng Yin, Jinha Kim, Prashant Mathur, Krishanu Sarker, and Vidit Bansal. 2025. How to talk to language models: Serialization strategies for structured entity matching. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7836–7850.
- Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024. Agenttuning: Enabling generalized agent abilities for llms. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077.

Enming Zhang, Ruobing Yao, Huanyong Liu, Junhui Yu, and Jiale Wang. 2024. First multi-dimensional evaluation of flowchart comprehension for multimodal large language models. *arXiv preprint arXiv:2406.10057*.

Ming Zhang, Yuhui Wang, Yujiong Shen, Tingyi Yang, Changhao Jiang, Yilong Wu, Shihan Dou, Qin hao Chen, Zhiheng Xi, Zhihao Zhang, et al. 2025. Pfdial: A structured dialogue instruction fine-tuning method based on uml flowcharts. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2626–2649.

A Appendix

A.1 Datasets

We evaluate the performance of FloCA on FLODIAL and PFDial. Table 7 shows a statistic summary of the datasets.

FLODIAL is a flowchart-oriented dialogue dataset containing 12 troubleshooting flowcharts and 2,738 dialogue samples collected via Amazon Mechanical Turk. The flowcharts cover the decision-making process for diagnosing technical issues in cars and laptops. Each dialogue corresponds to a specific decision path in the flowchart and is annotated with ground-truth node IDs for both agent and user turns. Since FloCA and all LLM/VLM-based baselines are evaluated in a zero-shot setting and require no training, we report results on the in-domain setting only on FLODIAL. The experimental results show that even when compared against SA-FloNet, which trained on the in-domain split, FloCA consistently achieves better performance across all metrics.

PFDial is a Chinese flowchart-based dialogue instruction dataset. Flowcharts in PFDial are represented in PlantUML. The advantage of PlantUML is that it employs a code-like structured representation utilizing syntax features such as indentation, branching, and loops, making flowchart descriptions both intuitive to read and convenient for program processing. However, PFDial is originally constructed as an instruction corpus, and dialogues are decomposed into atomic units formatted as five-tuples (i.e., flowchart description, current node, user input, next node, and agent output), rather than providing complete multi-turn dialogues aligned to the corresponding flowchart path. To enable evaluation on our proposed flowchart-oriented dialogue (FOD) task, we first extract all paths from the <start> node to the <end> node based on each PlantUML flowchart by strictly following

PlantUML parsing rules. We then use DeepSeek-V3.2 to synthesize a complete dialogue for each extracted path, conditioning on the attributes of all nodes and edges along the path. The synthesized dialogues follow the same format as FLODIAL. Each dialogue turn is annotated with the corresponding ground-truth node ID, providing accessibility for evaluation and further research on FOD task. All synthesized dialogues are released in our open-source repository.

Table 8 shows the result of *out-of-domain* FOD results on PFDial. Compared to *in-domain*, the limitations of SFT-based learning become pronounced under the *out-of-domain* setting. For Llama-3.1-8B, all reasoning metrics collapse, accompanied by a high failure rate, which is 48.31%. We define “Failure” as cases where the model outputs an empty response or produces a next-state text that cannot be matched to any node in the flowchart. The high failure rate shows that when both the flowchart content and user expressions shift in *out-of-domain*, the SFT model often fails to align an unseen user utterance with the specific node texts of an unseen PlantUML flowchart and thus generates a non-existent next-state that cannot be grounded to any node in the flowchart.

In contrast, FloCA remains stable across domains with zero Failure for all backbones. This robustness stems from it decoupling complex flowchart reasoning to an external tool that executes the graph and enforces topology-valid transitions, making the process less dependent on fitting a particular training distribution. These results suggest that robust performance on FOD relies more on topology-constrained execution than on fitting a training distribution via SFT on serialized flowcharts. Although FloCA performs lower on the out-of-domain than in-domain, this drop is because there are 1,961 decision branches in *out-of-domain* flowcharts, while only 870 in *in-domain*. The resulting decision space is more than twice as large, increasing ambiguity at each transition.

To analyze error patterns, we categorize each test sample into one of five path-coverage relations between the ground-truth path and the predicted path: (1) exact match, (2) ground-truth contained in prediction, (3) prediction contained in ground-truth, (4) partial overlap without containment, and (5) disjoint paths. Figure 5 shows the path-coverage relations on *out-of-domain* settings. The error pattern is identical with *in-domain* settings, as there are proportional samples that fall into group (4).

Datasets	Domain	Train Dialogs	Val Dialogs	Test Dialogs		Train flowcharts	Val flowcharts	Test flowcharts
				Root-init	Middle-init			
FLODIAL	In-domain	1,798	456	Root-init 350	Middle-init 134	12	12	12
	Out-of-domain	1,786	454	root node 408	middle node 90	8	2	2
PFDIAL	In-domain	12,705	–	Root-init 513	Middle-init 0	440	–	80
	Out-of-domain			Root-init 416	Middle-init 0			80

Table 7: A statistic summary of the datasets.

Method	Reasoning Metrics			Efficiency Metrics		Failure↓
	INGA↑	TNGA↑	PCA↑	TR↓	NSR↓	
Fine-tuned on PFDial						
Llama-3.1-8B	8.17	2.88	0.00	53.12	26.52	48.31
Qwen2.5-7B	<u>99.27</u>	66.34	37.50	13.22	3.66	<u>0.72</u>
FloCA						
Llama-3.1-8B-Instruct	100.00	70.19	29.32	0.00	4.86	0.00
Qwen2.5-7B-Instruct	100.00	65.14	39.66	0.00	2.53	0.00
Llama-3.3-70B	100.00	92.06	72.35	0.00	2.75	0.00
Qwen2.5-72B	100.00	82.69	62.01	0.00	<u>1.21</u>	0.00
DeepSeek-R1	100.00	97.83	83.41	0.00	1.02	0.00

Table 8: *Out-of-domain* FOD results on PFDial. The **bold font** highlights the best results, and the underline highlights the second-best results.

This indicates that many dialogues deviate from the ground-truth path at intermediate nodes. Therefore, improving overall flowchart reasoning success relies on increasing the correctness of every decision branch selection.

A.2 LLM edge matching error taxonomy

To analyze the specific error patterns in edge matching, we manually categorize LLM edge matching failures into three representative types, as shown in Table 9. The results reveal that even a strong LLM remains sensitive to ambiguous user input and superficial lexicality in binary decision grounding.

A.3 Additional Results on User Simulator

Table 10 shows the language diversity evaluation of the user simulator on PFDial in *an out-of-domain* setting. Metrics include Shannon Entropy (SE), conditional bigram entropy (CE), Mean Segmental Type–Token Ratio (MSTTR), Measure of Textual Lexical Diversity (MTLD), and Hyper-geometric Distribution (HDD). The results show that our proposed user simulator consistently achieves higher diversity metrics than the static annotated dialogues across all experimented datasets.

A.4 The Prompt for User Simulator

We first use GPT-5 to extract user-known facts from each annotated dialogue sample. The extraction prompt constrains the LLM to retain only infor-

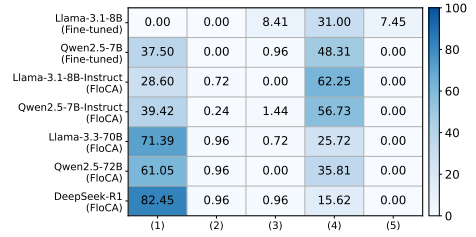


Figure 5: Distribution of path-coverage relations on the PFDial in the *out-of-domain* setting.

mation that would reasonably be available to the user, including the user’s goal, observable situation, and previously completed actions or checks. To simulate scenarios where the dialogue starts from an intermediate flowchart node rather than the root, we augment such samples with extra user-known information. Specifically, for each sample whose annotated initial node is not the root, we extract the textual attributes along the flowchart path from the root node to the annotated initial node and provide them in the prompt as “Pre-established actions”. This information represents actions or checks that the user would already have completed or known. The prompt of extraction is presented in Figure 6.

We then use DeepSeek-R1 to construct the user simulator in our experiments. The prompt template is presented in Figure 7. For each test dialogue, we initialize the interaction with the ground-truth first user utterance. After the agent generates the first response, the user simulator takes over and dynamically generates user replies for all subsequent turns, conditioned on the dialogue history, while ensuring that the generated utterances remain consistent with the facts specified in the annotated dialogue.

A.5 Additional Results on PFDial Dataset

A.6 The Prompt for LLM Judge

As user responses directly affect the agent’s flow reasoning accuracy, it is crucial to ensure that the user simulator does not introduce factually inconsistent replies that could erroneously mislead the

Error Type	Description	Proportion (%)	Examples
Ambiguous or indirect user response	The user utterance is ambiguous, incomplete, or does not directly answer the agent’s question.	40.0	Agent: “Can you check if a fusible link has blown?”; User: “Yes, I can check that. I’ve done it before on this vehicle.”; Predicted edge: “yes”; Ground truth: “no”
Partial semantic understanding	The user utterance contains sufficient information for correct edge matching, but the LLM fails to correctly interpret semantic cues such as negation, polarity, or logical consistency across multiple sentences.	48.0	Agent: “Can you check if a fusible link has blown?”; User: “There does not appear to be a blown fusible link. The fusible link appears okay.”; Predicted edge: “yes”; Ground truth: “no”
Negative-question polarity error	The LLM incorrectly maps the response polarity because it over-relies on surface negation words (e.g., “not”, “nothing”, and “do not”).	12.0	Agent: “When your Windows boots, is nothing being displayed on the LCD?”; User: “I do not see anything on it.”; Predicted edge: “no”; Ground truth: “yes”

Table 9: Taxonomy of LLM edge matching errors made by FloCA with GPT-5 on FLODIAL. The proportions are computed over all identified edge-matching failures.

Data Source	SE↑	CE↑	MSTTR↑	MTLD↑	HDD↑
	<i>PF Dial</i>				
Annotated Dialogues	2.35	0.03	0.97	6.45	5.46
User Simulator	2.62	0.04	0.98	11.33	7.00

Table 10: Language diversity comparison of user utterances in the *out-of-domain* setting. Metrics are computed per dialogue turn and averaged over all test samples.

agent. To this end, we employ GPT-5 as an LLM judge to verify whether any simulated user utterance in the generated dialogues violates the facts stated in the annotated dialogue. Figure 9 presents the prompt template used by the judge. Specifically, we treat the annotated user utterances as the set of known facts and ask the judge to determine whether each user utterance in the predicted dialogue contradicts any of these facts. For example, if the annotated dialogue states “my battery is short-circuited,” but the user simulator claims “my battery is not short-circuited,” this will be identified as a factual VIOLATION. From the experimental results, the judge detects no fact violations across all experiments, indicating that the user simulator remains faithful to the ground-truth facts throughout the interaction and ensuring a fair comparison among models.

A.7 The Prompt for Interactive Flowchart Reasoning

We use LLM to match each user input to a valid outgoing edge for the next node transition. Figure 8 shows the prompt template for node transition in interactive flowchart reasoning on FLODIAL. Since FLODIAL is built on decision-making flowcharts where each node poses a binary decision question and has two outgoing branches corresponding to “yes” and “no”, the LLM is instructed to

classify the user’s utterance as supporting one of these two conditions given the decision question of the current node. FloCA then performs topology-constrained graph execution and moves to the next-hop node according to the predicted edge condition.

A.8 Case Study

Below we show the annotated dialogue alongside the dialogues generated by FloCA and three baselines: graph serialization, RAG-enhanced graph serialization, and visual language models. All methods use GPT-5 as the backbone. In this comparison, we regard the annotated dialogue as ground truth.

The results indicate that FloCA can perform faithful flow reasoning by executing topologically valid node transitions, thus ultimately reaching the correct terminal node. We observe that the ground-truth dialogue is annotated to start at node 8, whereas FloCA initializes the dialogue state at the root node. This is because FloCA adopts a self-loop mechanism to automatically perform flowchart reasoning starting from the root, but the initial user utterance provides insufficient evidence to determine a reliable next hop. After several interaction turns with the user simulator, FloCA correctly grounds the dialogue to node 8 and subsequently follows the correct decision path to the terminal node. In contrast, the *graph serialization method* fails at initial node grounding and then deviates to a wrong path, resulting in the task failure. Notably, the *RAG-enhanced graph serialization* stays at node 12 and spends 2 dialogue turns to advance the flowchart reasoning. This observation suggests that when the retrieved context is misaligned with the model’s current reasoning, it may introduce additional uncertainty. LLM may be dis-

tracted from making the correct node transition, thereby increasing NSR metric. Besides, the VLM baseline exhibits severe repetitive clarification. It loops at node 12 for eight consecutive dialogue turns and then also transits to a wrong terminal node. This observation is consistent with the NSR results, indicating that vision-based flowchart reasoning is more prone to repeated clarification turns, which increases redundancy and often prevents the dialogue from progressing along the decision path.

The observations in this case study are consistent with the evaluation results. It highlights the limitations of probabilistic generation when performing flowchart reasoning and provides evidence that the explicit topological constraints in FloCA are critical for achieving faithful flowchart reasoning.

Case Study: Ground-truth Dialogue

```
{
  "utterance": [
    {
      "speaker": "user",
      "utterance": "I have a Dell laptop and its fan never runs. It has started to overheat really badly and I am quite worried about it. Can you help me figure this out?",
      "turn_id": 1
    },
    {
      "speaker": "agent",
      "utterance": "Did you put your laptop on a flat and hard surface?",
      "turn_id": 1,
      "pre_doc_id": "chart-8"
    },
    {
      "speaker": "user",
      "utterance": "I have the laptop on a table.",
      "turn_id": 2
    },
    {
      "speaker": "agent",
      "utterance": "Does the overheating cause your laptop to shut down?",
      "turn_id": 2,
      "pre_doc_id": "chart-11"
    },
    {
      "speaker": "user",
      "utterance": "It shuts down when it overheats.",
      "turn_id": 3
    },
    {
      "speaker": "agent",
      "utterance": "Do you have BIOS set to put the laptop into hibernation upon low battery?",
      "turn_id": 3,
      "pre_doc_id": "chart-14"
    },
    {
      "speaker": "user",
      "utterance": "There is no such setting. My laptop continues to run all the time.",
      "turn_id": 4
    },
    {
      "speaker": "agent",
      "utterance": "I recognize the problem with your laptop now. You need to clean all the air vents and replace the active heat sink.",
      "turn_id": 4,
      "pre_doc_id": "chart-26"
    }
  ],
  "flowchart": "laptop_overheating"
}
```

Case Study: Predicted Dialogue Generated by FloCA with GPT-5

```
{
  "utterance": [
    {
      "speaker": "user",
      "utterance": "I have a Dell laptop and it's fan never runs. It has started to overheat really bad and I am quite worried about it. Can you help me figure this out?",
      "turn_id": 1
    },
    {
      "speaker": "agent",
      "utterance": "Do you smell burning or see any smoke coming from the laptop?",
      "turn_id": 1,
      "pre_doc_id": "chart-1"
    }
  ]
}
```

```

},
{
  "speaker": "user",
  "utterance": "No, I don't smell any burning or see smoke coming from the laptop. It's just getting extremely hot to the touch.",
  "turn_id": 2
},
{
  "speaker": "agent",
  "utterance": "Are you using your laptop in a hot environment?",
  "turn_id": 2,
  "pre_doc_id": "chart-2"
},
{
  "speaker": "user",
  "utterance": "No, I'm using it indoors at normal room temperature.",
  "turn_id": 3
},
{
  "speaker": "agent",
  "utterance": "Please tell me if your laptop's battery is overheating?",
  "turn_id": 3,
  "pre_doc_id": "chart-3"
},
{
  "speaker": "user",
  "utterance": "No, the battery itself doesn't seem to be overheating. It's the main body of the laptop, near the vents and the bottom, that gets extremely hot.",
  "turn_id": 4
},
{
  "speaker": "agent",
  "utterance": "Is your laptop's fan running when it overheats?",
  "turn_id": 4,
  "pre_doc_id": "chart-6"
},
{
  "speaker": "user",
  "utterance": "Yes, the fan is running when it overheats. I can definitely hear it.",
  "turn_id": 5
},
{
  "speaker": "agent",
  "utterance": "Are you using your laptop on a hard, flat surface?",
  "turn_id": 5,
  "pre_doc_id": "chart-8"
},
{
  "speaker": "user",
  "utterance": "I have the laptop on a table.",
  "turn_id": 6
},
{
  "speaker": "agent",
  "utterance": "Does your laptop shut down automatically when it overheats?",
  "turn_id": 6,
  "pre_doc_id": "chart-11"
},
{
  "speaker": "user",
  "utterance": "Yes, it does shut down automatically when it overheats, but only when I'm running something demanding like a game or video editing software.",
  "turn_id": 7
},
{
  "speaker": "agent",
  "utterance": "Is your BIOS configured to hibernate the laptop in case of low battery?",
  "turn_id": 7,
  "pre_doc_id": "chart-14"
}

```

```

    },
    {
      "speaker": "user",
      "utterance": "There is no such setting. My laptop continues to run all the time.",
      "turn_id": 8
    },
    {
      "speaker": "agent",
      "utterance": "I understand the issue now. You will need to clean all the air vents on your laptop and replace the active heat sink.",
      "turn_id": 8,
      "pre_doc_id": "chart-26"
    }
  ],
  "flowchart": "laptop_overheating"
}

```

Case Study: Predicted Dialogue Generated by Graph Serialization Method with GPT-5

```

{
  "utterances": [
    {
      "speaker": "user",
      "utterance": "I have a Dell laptop and it's fan never runs. It has started to overheat really bad and I am quite worried about it. Can you help me figure this out?",
      "turn_id": 1
    },
    {
      "speaker": "agent",
      "utterance": "Have you recently upgraded your laptop's BIOS?",
      "turn_id": 1,
      "pre_doc_id": "chart-7"
    },
    {
      "speaker": "user",
      "utterance": "No, I haven't.",
      "turn_id": 2
    },
    {
      "speaker": "agent",
      "utterance": "Have you changed the laptop's power management settings recently?",
      "turn_id": 2,
      "pre_doc_id": "chart-12"
    },
    {
      "speaker": "user",
      "utterance": "No, I haven't changed any power settings.",
      "turn_id": 3
    },
    {
      "speaker": "agent",
      "utterance": "You're facing overheating due to a hardware failure. Your laptop's fan or temperature sensor is not working properly, and you should replace it.",
      "turn_id": 3,
      "pre_doc_id": "chart-19"
    }
  ],
  "flowchart": "laptop_overheating"
}

```

Case Study: Predicted Dialogue Generated by RAG-enhanced Graph Serialization Method with GPT-5

```

{
  "utterances": [
    {

```

```

    "speaker": "user",
    "utterance": "I have a Dell laptop and it's fan never runs. It has started to overheat really bad and I am quite worried about it. Can you help me figure this out?",
    "turn_id": 1
  },
  {
    "speaker": "agent",
    "utterance": "Have you recently upgraded your laptop's BIOS?",
    "turn_id": 1,
    "pre_doc_id": "chart-7"
  },
  {
    "speaker": "user",
    "utterance": "No, I haven't done that.",
    "turn_id": 2
  },
  {
    "speaker": "agent",
    "utterance": "Have you changed the laptop's power management settings recently?",
    "turn_id": 2,
    "pre_doc_id": "chart-12"
  },
  {
    "speaker": "user",
    "utterance": "No, I haven't done that.",
    "turn_id": 2
  },
  {
    "speaker": "agent",
    "utterance": "Have you changed the laptop's power management settings recently?",
    "turn_id": 2,
    "pre_doc_id": "chart-12"
  },
  {
    "speaker": "user",
    "utterance": "No, I haven't changed any power settings recently.",
    "turn_id": 3
  },
  {
    "speaker": "agent",
    "utterance": "You are experiencing overheating due to a hardware failure. Your laptop's fan or temperature sensor is not working properly and you should replace it.",
    "turn_id": 3,
    "pre_doc_id": "chart-19"
  },
  },
  "flowchart": "laptop_overheating"
}

```

Case Study: Predicted Dialogue Generated by Visual Language Models with GPT-5

```

{
  "utterances": [
    {
      "speaker": "user",
      "utterance": "I have a Dell laptop and it's fan never runs. It has started to overheat really bad and I am quite worried about it. Can you help me figure this out?",
      "turn_id": 1
    },
    {
      "speaker": "agent",
      "utterance": "Have you recently upgraded your laptop's BIOS?",
      "turn_id": 1,
      "pre_doc_id": "chart-7"
    },
  ],
  {

```

```

    "speaker": "user",
    "utterance": "No, I haven't.",
    "turn_id": 2
  },
  {
    "speaker": "agent",
    "utterance": "Have you changed the laptop's power management settings recently?",
    "turn_id": 2,
    "pre_doc_id": "chart-12"
  },
  {
    "speaker": "user",
    "utterance": "No, I haven't changed any power settings recently.",
    "turn_id": 3
  },
  {
    "speaker": "agent",
    "utterance": "Did you change the laptop power management settings recently?",
    "turn_id": 3,
    "pre_doc_id": "chart-12"
  },
  {
    "speaker": "user",
    "utterance": "No, I haven't changed any power settings recently.",
    "turn_id": 4
  },
  {
    "speaker": "agent",
    "utterance": "Have you changed your laptop's power management settings recently?",
    "turn_id": 4,
    "pre_doc_id": "chart-12"
  },
  {
    "speaker": "user",
    "utterance": "No, I haven't changed any power settings recently.",
    "turn_id": 5
  },
  {
    "speaker": "agent",
    "utterance": "Have you changed the laptop power management settings recently?",
    "turn_id": 5,
    "pre_doc_id": "chart-12"
  },
  {
    "speaker": "user",
    "utterance": "No, I haven't made any changes to the power management settings.",
    "turn_id": 6
  },
  {
    "speaker": "agent",
    "utterance": "Have you changed the laptop's power management settings recently?",
    "turn_id": 6,
    "pre_doc_id": "chart-12"
  },
  {
    "speaker": "user",
    "utterance": "No, I haven't made any changes to the power management settings.",
    "turn_id": 7
  },
  {
    "speaker": "agent",
    "utterance": "Have you changed the laptop power management settings recently?",
    "turn_id": 7,
    "pre_doc_id": "chart-12"
  },
  {
    "speaker": "user",
    "utterance": "No, I haven't changed the power management settings.",
    "turn_id": 8
  }

```

```

    },
    {
      "speaker": "agent",
      "utterance": "Did you change the laptop's power management settings recently?",
      "turn_id": 8,
      "pre_doc_id": "chart-12"
    },
    {
      "speaker": "user",
      "utterance": "No, I haven't changed the power management settings.",
      "turn_id": 9
    },
    {
      "speaker": "agent",
      "utterance": "Have you changed the laptop's power management settings recently?",
      "turn_id": 9,
      "pre_doc_id": "chart-12"
    },
    {
      "speaker": "user",
      "utterance": "No, I haven't changed any power management settings.",
      "turn_id": 10
    },
    {
      "speaker": "agent",
      "utterance": "Revert your power management settings to their previous state. This should prevent your laptop from overheating.",
      "turn_id": 10,
      "pre_doc_id": "chart-15"
    }
  ],
  "flowchart": "laptop_overheating"
}

```

System Prompt:	You are an information extraction assistant.
User Prompt:	Extraction goal: Extract only information that the user can reasonably know or observe based on the given annotated dialogue, so that a simulator can consistently play the role of that user in a multi-turn dialogue. Given an annotated dialogue sample, extract the following types of information: 1. User goal: What the user is trying to solve, achieve, or ask about. 2. Observable situation: troubles, conditions, status, or other observable facts described by the user. 3. Pre-established actions or checks: Actions, inspections, or operations that have already been completed before the conversation begins. Important rules: 1. Extract only user-side observable or established facts. 2. Do not infer hidden causes, correct diagnoses, or future outcomes. 3. Do not add facts that are not grounded in the annotated dialogue. 4. Keep the extracted facts concise, neutral, and faithful to the dialogue. 5. Organize the output into the required schema. Output format: <pre>{user_goal: [...], observable_situation: [...], completed_actions_or_checks: [...]}</pre> The given annotated dialogue is below: Annotated dialogue: {dialogue} Pre-established actions: {actions}

Figure 6: The prompt template for user-known fact extraction.

System Prompt:	You are a dialogue simulator acting as a user engaged in troubleshooting with an AI assistant.
User Prompt:	Your task is to generate the user's reply at each turn, based on: 1. User goal: What the user is trying to solve, achieve, or ask about. 2. Observable situation: troubles, conditions, status, or other observable facts described by the user. 3. Pre-established actions or checks: Actions, inspections, or operations that have already been completed before the conversation begins. 4. Reference dialogue: An example dialogue provided for guide language style and questioning behavior. Simulation Rules: 1. Respond only from the user's perspective. 2. Use the provided scenario context (goal, observable situation, and completed actions) when generating replies. 3. If the assistant asks about a condition the user can check or observe, respond based on the known scenario facts. 4. If the assistant asks about something the user does not know, ask for clarification or request guidance naturally. 5. If the question is new (not in the reference dialogue), infer a plausible reply based on the known context and reference dialogue style. 6. Keep the response concise and conversational, as a real user would. <pre>user_goal: {user_goal} observable_situation: {observable_situation} completed_action_or_checks: {completed_actions_or_checks} AI Assistant: {input} User simulator:</pre>

Figure 7: The prompt template for user simulator.

System Prompt: You are an assistant that determines whether the user answered a given yes/no question.

User Prompt: Your job is to analyze the user's input and output one of the following:

- yes – if the user clearly or implicitly answered yes
- no – if the user clearly or implicitly answered no
- trigger – Need to trigger a knowledge base to answer user's question. If the user did NOT answer the question and instead asked a new question or changed the topic.

You MUST output only one word: yes, no, or trigger.

You must carefully analyze what the question is asking; do not rely solely on the presence of positive or negative words in user input.

Do not include any punctuation or extra words.

Given question:
{question}

User input:
{input}

Your response:

Figure 8: The prompt template for interactive flowchart reasoning in FLODIAL.

System Prompt: You are a factual-consistency judge for dialogue generation. Your job is to check whether each USER utterance in a predicted dialogue violates any known facts. Known facts are ONLY the USER utterances from the given ground-truth dialogue. Be conservative: only mark a violation when there is a clear contradiction with known facts. If information is not mentioned in known facts, mark it as "UNKNOWN" rather than "VIOLATION". Return results in the required JSON format only.

User Prompt: Given:

1. ground_truth_dialogue: USER utterances inside it are the ONLY known facts.
2. predicted_dialogue: evaluate EACH USER utterance for factual consistency against known facts.

Labels:

- CONSISTENT: not contradicted by any known fact (may be a paraphrase or compatible detail)
- VIOLATION: clearly contradicts at least one known fact
- UNKNOWN: cannot be verified from known facts (neither supported nor contradicted)

Rules:

- Only judge USER utterances in predicted_dialogue.
- If a predicted user utterance contains multiple claims:
 - If ANY claim is VIOLATION $\Rightarrow$ utterance label is VIOLATION
 - Else if ANY claim is UNKNOWN $\Rightarrow$ utterance label is UNKNOWN
 - Else CONSISTENT

Output (JSON only)

```
{
  "CONSISTENT": ["<pred_turn_id_1>", "<pred_turn_id_2>", "..."],
  "VIOLATION": ["..."],
  "UNKNOWN": ["..."]
}
```

ground_truth_dialogue
{GROUND_TRUTH_DIALOGUE}

predicted_dialogue
{PREDICTED_DIALOGUE}

Figure 9: The prompt of LLM judge for user simulator evaluation.