LogiEx: Integrating Formal Logic and LLMs for Explainable Transit Planning

Ziyan An*, Xia Wang*, Hendrik Baier†, Zirong Chen*, Abhishek Dubey*
Taylor T. Johnson*, Jonathan Sprinkle*, Ayan Mukhopadhyay‡, Meiyi Ma*
*Department of Computer Science, Vanderbilt University, Nashville, TN, USA
†Department of Industrial Engineering & Innovation Sciences, Eindhoven University of Technology, Eindhoven, The Netherlands
‡Department of Computer Science, William & Mary, Williamsburg, VA, USA

***Abstract*—Human-centered cyber-physical systems (CPS), such as intelligent transportation services, warehouse robotics operated by human supervisors, and healthcare infrastructures involving clinicians and medical staff, increasingly rely on Artificial Intelligence (AI)-driven sequential decision-making under uncertainty. However, the lack of transparent reasoning in these systems limits trust, verifiability, and human oversight. This challenge is particularly acute for planning algorithms like Monte Carlo Tree Search (MCTS), whose stochastic search processes are opaque to engineers and operators. To address this gap, we introduce LogiEx, a logic-integrated framework that combines large language models (LLMs) with formal methods to generate trustworthy explanations for planning behavior. LogiEx transforms free-form user queries into logical statements with templated variables, then verifies whether evidence extracted from the decision process aligns with both the environment state and the constraints of the stochastic planning model. This enables grounded explanations across a wide range of user questions—from factual retrieval to comparative reasoning. LogiEx also supports Human-Guided Search (HuGS), allowing users to pose conditional "what-if" queries that trigger new, scenario-specific searches, ensuring that humans are not passive observers but active participants who can steer and refine the planning process. We evaluate LogiEx through both quantitative assessments and user studies, finding that it consistently outperforms baselines, achieving up to 7.9× higher semantic similarity (BERTScore) and 1.6× higher factual consistency (FactCC) compared to baseline LLMs, and is the most preferred form of explanation among CPS practitioners.**[1]

***Index Terms*—Explainable AI, Monte Carlo Tree Search, Large Language Models, Formal Logic, Intelligent Transportation.**

I. Introduction

Human-centered cyber-physical systems (CPS) [1, 2] increasingly integrate learning-enabled components to support real-time decision-making and control in dynamic physical environments such as warehouse automation, intelligent transportation, and assistive mobility. These systems tightly couple data-driven Artificial Intelligence (AI) with physical actuation, requiring not only high performance but also transparent and interpretable behavior to maintain safety and effective human oversight [3, 4, 5, 6, 7, 8]. Despite recent progress in AI, practitioners in CPS remain cautious about large-scale deployment due to the limited transparency and interpretability of the underlying algorithms [9]. Among these algorithms, Monte Carlo Tree Search (MCTS) [10] has become a powerful planner for sequential decision-making under uncertainty, but its stochastic search trees make it difficult to trace or verify internal reasoning [11]. This lack of explainability is especially problematic in complex CPS domains, where expert knowledge is critical for understanding edge cases and ensuring safe decision-making. Interpreting the outputs of planners like MCTS is particularly challenging due to the complexity of their sampling-based search trees [11]. This leads to an urgent need for an explainable approach that offers both rigorous and human-understandable explanations while incorporating domain expert input into the planning process.

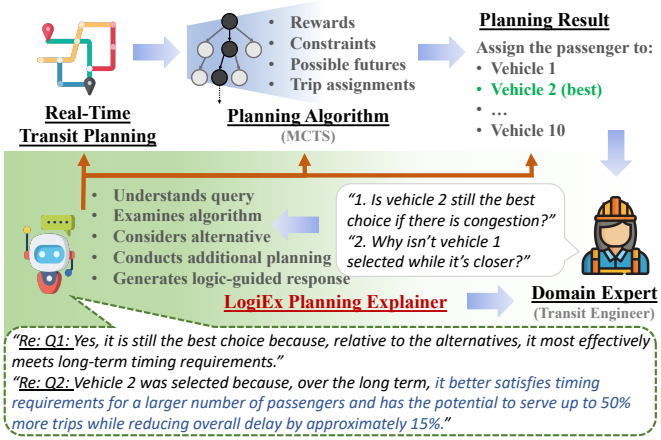

Fig. 1: LogiEx explains sequential planning by combining domain knowledge, search process, and logical reasoning.

Targeting this challenge, we develop LogiEx, a logic-enabled large language models (LLMs) framework that integrates knowledge and logical reasoning into natural language explanations [12, 13], creating a robust yet expressive explainable AI (XAI) system for explaining planning algorithms like MCTS. Leveraging paratransit routing as a testbed, LogiEx (Figure 1) generates explanations for queries related to the planning algorithm, accommodating users with varying technical backgrounds in AI.

[1]Corresponding author email: ziyan.an@vanderbilt.edu, meiyi.ma@vanderbilt.edu. The code is available at https://github.com/AICPS-Lab/LogiEx.git.

LogiEx is, to the best of our knowledge, the first approach to enable state-of-the-art LLMs to correctly understand the intricacies of complex search trees, which was challenging due to the extensive information involved in the planning task, the sampled futures, and the highly specific scenario context, as suggested by our preliminary studies. While standard LLMs operate as black boxes and offer limited transparency, making them ill-suited for high-stakes CPS applications [14, 15, 16], LogiEx is designed to be interpretable by construction. It grounds its responses in formal logical reasoning, the structure of the planner's search process, and a domain knowledge base, which allows it to generate detailed, factually grounded explanations that meet the rigor expected by domain experts. LogiEx offers broad flexibility in handling queries by converting natural language inquiries submitted via a chat interface into parameterized logic expressions using variables. It then evaluates the search tree based on the criteria specified by these logic expressions, and the results are presented in the final explanation, once again expressed in natural language. The LogiEx explainer enables an unlimited number of follow-up queries, facilitating an interactive, back-and-forth communication with the user.

LogiEx explains planning algorithms by addressing three categories of queries. The first category involves queries answerable from the existing search tree, the second category enables human-guided search (HuGS) [17], and the third category is based on background knowledge. LogiEx enables HuGS by transforming questions into expressions that correspond to user-specified requirements about possible future scenarios. Routing decisions, by default, are made using a regular demand model generated from real-world historical data. However, a user may ask questions such as "Which vehicle will pick up the passenger if there is traffic congestion ahead?" and "What alternative vehicle can be used if the current vehicle breaks down?". In such cases, the existing search tree is insufficient to provide answers, and LogiEx engages MCTS for additional searches to generate the necessary results. For queries answerable with background knowledge, LogiEx retrieves domain knowledge from a knowledge base using the retrieval-augmented generation (RAG) technique [18], ensuring the answers are comprehensive yet relevant to the user's inquiry.

To evaluate LogiEx, we explore four research questions through both quantitative evaluation and user study. The first part of the quantitative evaluation assesses the accuracy of state-of-the-art LLMs in converting user queries to the correct logic using different prompting techniques, which the second part examines the factual consistency of the final explanations generated by the system. Our user study evaluates users' experiences with our system. It compares the user experience of our conversation-based explanation framework against two baselines, one of them being state-of-the-art in our application setting, evaluating them on a 5-point Likert scale in terms of understandability, satisfaction, completeness, and reliability.

We summarize our contributions as follows: **(1)** We present LogiEx, the first logic-enabled XAI framework based on LLMs, designed to explain sequential planning from three distinct perspectives: post-hoc, HuGS, and knowledge-based; **(2)** We leverage a diverse set of parameterized logic templates by matching queries to predefined logic statements and dynamically populating the templates; **(3)** We enable HuGS-based explanations by allowing user-specified search conditions through natural language queries; **(4)** To handle inquiries about the general planning scenario and the MCTS decision process, we identified frequently-asked queries through a prior user study and constructed a knowledge base to facilitate RAG; and **(5)** LogiEx significantly outperforms baseline LLMs, achieving up to 1.7× higher factual consistency (FactCC) and up to 7.9× higher semantic similarity (BERTScore) compared to the basic GPT-4 and Llama 3.1 models, respectively.

II. Background on Smart Transit

A. Markov Decision Process

We use a paratransit planning scenario formulated as a Markov Decision Process (MDP) as the testbed. Paratransit services provide a socially beneficial, needs-based form of transportation, typically operated by public transit agencies to serve individuals with disabilities who are unable to use fixed-route buses. The MDP is defined using the tuple $\langle S, A, \rightarrow, \mathsf{rew}, \gamma \rangle$, where: S is the set of states, A is the set of actions, $\rightarrow$ is the transition between states, $\mathsf{rew}(s, a)$ is the reward for action a in s, and γ is the discount factor.

State Space. The state in this MDP represents the status of all trip requests and vehicles. It is defined as $\langle t, \theta, r, V \rangle$, where t represents the current time of the state, θ stores the current route plan (a list of locations) for all vehicles, r denotes a current incoming request, and V represents the vehicle fleet. C_i and O_i are components of V that denote the capacity and occupancy of vehicle i. Each state in the MDP includes a new request r, defined by the tuple $\langle t_r, t_p, t_d, l_p, l_d, u \rangle$, where t_r denotes the time the request was made, t_p denotes the specified pick-up time, and t_d denotes the passenger's requested drop-off time. Additionally, l_p refers to the pick-up location, l_d to the drop-off location, and u indicates the status of the trip request: waiting, assigned, in-transit, or dropped-off.

State Transition. The state transition $s \rightarrow s'$ happens whenever a new request r arrives, which is driven by a simulated demand model for paratransit trip requests. We leverage the simulated demand model from a real-world paratransit dataset [19]. As a result, there is no analytical form of the state transition.

Action Space. The action space is defined as the finite set of actions $A = \{a_1, \cdots, a_{|V|}\}$, where a_i is the action of assigning the passenger of the current incoming request to vehicle i, and $|V|$ denotes the total number of vehicles. The maximum number of actions in a given state corresponds to the number of available vehicles. Assigning a trip request to a vehicle results in a new route plan, where the new pick-up and drop-off locations specified in the trip request are inserted into vehicle i's route plan, θ_i.

To determine the insertion points, let $\theta_i = [l_1, \ldots, l_n]$ represent the current route plan, and let l_p and l_d denote the

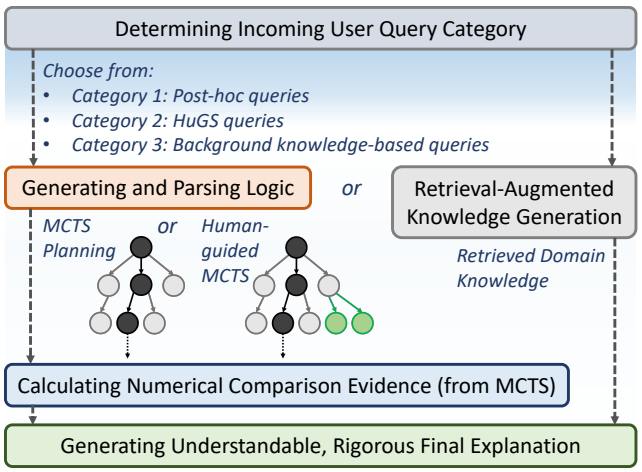

Fig. 2: LogiEx key components include determining query categories (Sec. III-A), generating and parsing explanation logic (Sec. III-B), performing human-guided search (Sec. III-C), calculating numerical comparison evidence (Sec. III-D), retrieval-augmented knowledge generation (Sec. III-E), and generating final explanations (Sec. III-F).

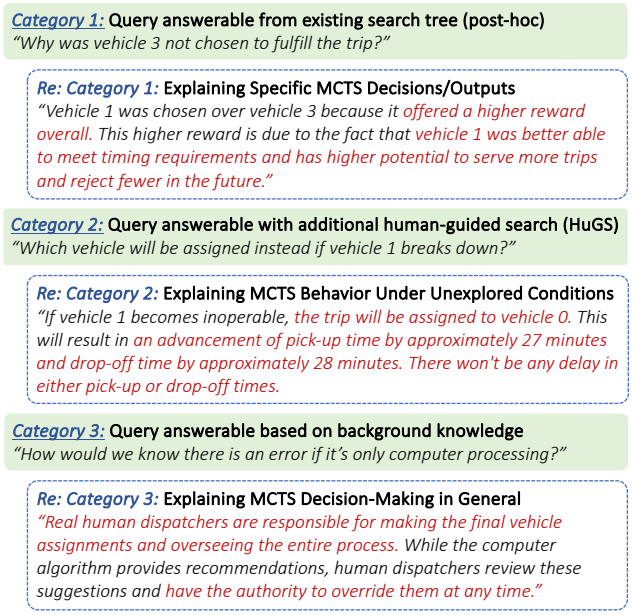

Fig. 3: Query categories in LogiEx: query answerable from existing search tree; query answerable with additional HuGS; query answerable based on domain knowledge.

pick-up and drop-off locations to be inserted. The function $d(l_i, l_j)$ calculates the travel time between locations l_i and l_j. For each new location, the insertion point is chosen to minimize the increase in travel time:

$$j^* = \arg \min_{j \in \{1,2,\ldots,n-1\}} \big(d(l_s, l_j) + d(l_s, l_{j+1}) - d(l_j, l_{j+1})\big),$$

where l_s is the new location to be inserted (l_p or l_d). The pick-up location l_p is always scheduled before the drop-off location l_d in the updated route plan.

B. Monte Carlo Tree Search

MCTS is initiated whenever a request is made by a passenger. The process of assigning a passenger request is referred to as a "decision epoch" [20]. The desired behavior of the paratransit system is that, over time, more passengers are accommodated, with their pick-up and drop-off times as close to their specified times as possible. Additionally, there is a physical constraint on vehicle capacity.

Reward Function. The MCTS reward combines two components: trip fulfillment and timing accuracy. The trip fulfillment component measures the proportion of fulfilled trips (in-transit or completed), while the timing accuracy component penalizes deviations between requested and actual pick-up/drop-off times. These components are weighted and combined to optimize both passenger satisfaction and operational efficiency. Formally, let R denote the set of trip requests, and $|R|$ be the total number of requests. The trip fulfillment ratio is defined as:

$$w_f = \frac{1}{|R|} \sum_{i=1}^{|R|} \mathbb{1}\big(u_i = \text{“in-transit” or “dropped-off”}\big),$$

where $\mathbb{1}$ is the indicator function. For timing accuracy, let t^i_{ap} and t^i_{ad} be the actual pick-up and drop-off times, and t^i_p and t^i_d be the corresponding requested times. For each request i,

$$\text{Timing}_i = \begin{cases} t^i_{ap} - t^i_p, & \text{if } u_i = \text{“in-transit”}, \\ (t^i_d - t^i_{ad}) + (t^i_{ap} - t^i_p), & \text{if } u_i = \text{“dropped-off”}. \end{cases}$$

The timing component across all trips is: $w_t = \sum_{i=1}^{|R|} \text{Timing}_i$. Let a and b denote the weights for the fulfillment and timing components, respectively. The final reward is given by:

$$\text{Reward} = a \cdot w_f + b \cdot w_t.$$

This formulation allows MCTS to explore future vehicle–request assignments that balance service efficiency and passenger satisfaction, while enabling our later formal logic analysis to evaluate and explain the resulting plans.

III. LogiEx: A Logic-Guided LLMs Framework for Explainability

Overview. LogiEx integrates LLMs with formal logic to explain the decision process of sequential planners such as MCTS. As shown in Figure 2, it follows a multi-stage pipeline that transforms user queries into logical statements, verifies them against the search tree or knowledge base, and produces natural-language explanations. The process begins with query classification, which categorizes each question into (1) post-hoc queries answerable from the existing MCTS tree, (2) HuGS queries that trigger new MCTS runs under user-specified conditions, or (3) background-knowledge queries supported by RAG. Next, the system generates and parses logic formulas corresponding to the classified query type, extracting relevant evidence from the MCTS process. For HuGS

queries, LogiEx modifies the planning context and re-executes MCTS before collecting evidence. Finally, LogiEx synthesizes explanations by leveraging verified logical evidence or retrieved knowledge and LLMs, yielding concise, grounded responses that clarify why specific planning decisions were made or how they would change under alternative scenarios.

A. Determining Query Categories

LogiEx generates clear and correct explanations by leveraging domain knowledge, search algorithm, and logical reasoning. We start by defining three coarse categories of queries, as shown in Figure 3. The first two categories focus on explaining specific MCTS decisions or outputs, requiring precise, factually consistent information from the search tree. The last category focuses on the decision-making process in general.

Post-hoc Queries. Post-hoc queries seek explanations for the returned plan after the algorithm has completed its execution. We address these queries by extracting and interpreting information from the existing search tree. This approach does not require modifying the algorithm or the plan but instead focuses on interpreting the internal planning process.

HuGS Queries. While post-hoc queries are limited to the states explored in the original search, LogiEx supports additional *HuGS queries* that simulate alternative conditions. These queries allow users to specify hypothetical changes (e.g., forcing certain decisions, altering environmental variables, or fixing future actions) to trigger a new MCTS search under the modified assumptions. This enables forward-looking, conditional reasoning about "what-if" scenarios that were not explored in the original search due to resource limitations or search heuristics.

Background Knowledge-based Queries. Lastly, background knowledge-based queries target broader questions about the general behavior of the MCTS algorithm, such as its exploration strategy or the frequency of certain decisions across states. LogiEx answers these queries by retrieving factual information from a structured knowledge base rather than explaining specific search outcomes.

Query Classification Pipeline. As shown in the topmost component of Figure 4, the process begins when the user submits a query. A Query Classification LLM component then interprets the query and attempts to classify its intent into one of three coarse query categories. Since user queries in LogiEx are unconstrained in content and phrasing (though limited to one question at a time), LogiEx also performs fine-grained classification to better align each query with its underlying intention. This finer classification applies to the first two categories, enabling more strategic and targeted responses. For Category 1 queries that are answerable using the existing MCTS search tree, we define 26 specific types, each corresponding to a distinct aspect of the search process. For Category 2 queries that require additional HuGS, we define 5 types that allow users to modify the search conditions based on hypothetical scenarios. In contrast, Category 3 queries, which rely on background knowledge, are not broken into specific types, as a single query may map to multiple pieces of knowledge, and vice versa.

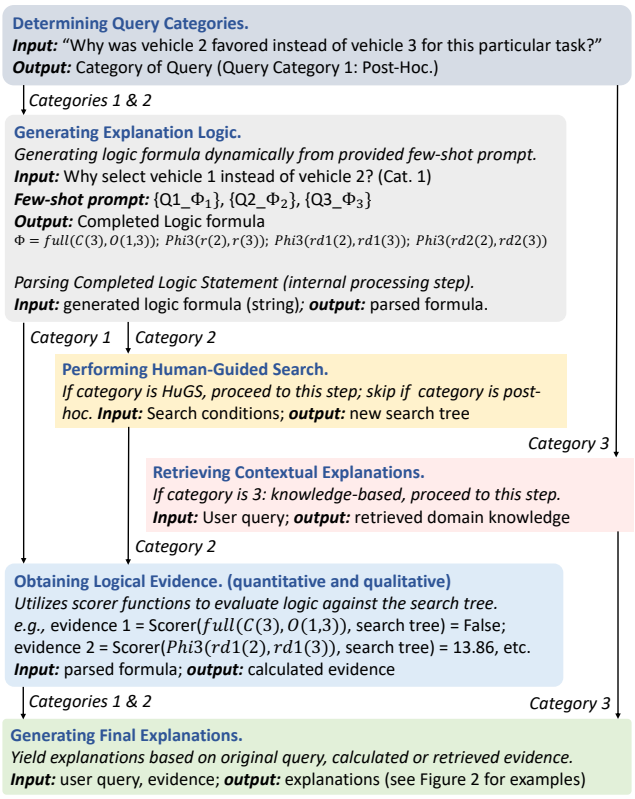

Fig. 4: Step-by-step outline for explanation generation.

B. Generating Explanation Logic

Once a query has been classified into a fine-grained type, it is matched with a corresponding logic formula or a HuGS operation. Although users may express their queries in varied forms, such as asking why a vehicle was not assigned or whether a passenger will arrive early or late, the underlying reasoning often maps to the same logical structure, differing only in variable values. To accommodate this variability in phrasing and content, logic is generated dynamically based on the query. Each fine-grained query type is associated with a tailored few-shot prompt that includes representative example queries and their corresponding logical formulations. After a new query is assigned a type, the associated prompt guides the logic generation LLM component in constructing an appropriate logical statement for the query.

Evidence Complexity. The level of detail and complexity required to generate an explanation varies across query types. Some queries can be answered by examining a single node in the MCTS search tree, while others require summarizing an entire subtree or comparing outcomes across different branches. We categorize evidence complexity into three types based on the retrieval method used within the tree: *(1) Base-level* complexity involves extracting information directly from a single MCTS state. For example, retrieving the requested pick-up time for a specific passenger. *(2) Second-level* derived

TABLE I: Examples of user queries, variables, and logic templates by evidence complexity.

Query Example	Variable(s)	Logic Template
Q1: What is the scheduled drop-off time for the passenger at state s?	$t_d(s)$	$t_d(s)$
Q2: How long might the drop-off delay be at state s?	$\mathsf{vio}_d(s)$	$\mathsf{vio}_d(s) = \mathsf{eta}(s) - t_d(s)$
Q3: What is the expected rate of delay for dropping off the passenger when traveling in vehicle 1?	$\mathsf{pct}_d(s)$	$\mathsf{pctd}(t_d(1), \mathsf{eta}(1))$
Q4: Does state s_1 result in less delay than state s_2?	$\phi_1(s_1, s_2)$	$\phi_1(s_1, s_2) = AG\big(\mathsf{vio}_d(s_1) < \mathsf{vio}_d(s_2)\big)$
Q5: Does vehicle 1 offer a route with fewer stops than vehicle 2?	$\phi_2(s_1, s_2)$	$\phi_2(s_1, s_2) = AG\big(s_d(s_1) < s_d(s_2)\big)$

TABLE II: LogiEx parser variables and logic examples aligned with the three-level hierarchy $\mathcal{X} \to \mathcal{V} \to \Phi$.

ID	Base-Level $\mathcal{X}$	Notation
1.1	Scheduled pick-up time at state s	$t_p(s)$
1.2	Scheduled drop-off time at state s	$t_d(s)$
1.3	Number of MCTS simulations from state s	$N(s)$
1.4	Maximum capacity of the vehicle in s	$C(s)$
1.5	Current occupancy of the vehicle in s	$O(s)$
1.6	Stops before pick-up in s	$s_p(s)$
1.7	Stops before drop-off in s	$s_d(s)$
1.8	Total / decomposed reward in s	$r(s),\ r_d^1(s),\ r_d^2(s)$
1.9	Estimated time of arrival in s	$\mathsf{eta}(s)$

ID	Second-Level $\mathcal{V}$	Formula / Notation
2.1	Degree of delay	$\mathsf{vio}_d(s) \coloneqq \mathsf{eta}(s) - t_d(s)$
2.2	Degree of early arrival	$\mathsf{vio}_a(s) \coloneqq t_p(s) - \mathsf{eta}(s)$
2.3	Probability of delay	$\mathsf{pct}_d(s)$
2.4	Probability of early arrival	$\mathsf{pct}_a(s)$
2.5	Capacity violation	$\mathsf{vio}_c(s) \coloneqq O(s) - C(s)$

ID	Logic Comparison Φ	Logic Template
3.1	Compare delay degrees between states	$\phi_1(s_1, s_2)$: $AG(\mathsf{vio}_d(s_1) < \mathsf{vio}_d(s_2))$
3.2	Compare delay probabilities between states	$\phi_2(s_1, s_2)$: $AG(\mathsf{pct}_d(s_1) < \mathsf{pct}_d(s_2))$
3.3	Compare rewards between states	$\phi_3(s_1, s_2)$: $AG(r(s_2) < r(s_1))$
3.4	Compare number of stops between states	$\phi_4(s_1, s_2)$: $AG(s_d(s_1) < s_d(s_2))$

complexity requires examining multiple states across different depths or branches to evaluate possible future outcomes from a given state. An example includes estimating how much a pick-up time would be violated if a particular vehicle were assigned. *(3) Logic comparison* complexity builds on second-level reasoning by comparing outcomes under two distinct conditions. For instance, determining whether one vehicle performs better than another in adhering to timing constraints.

Variable Hierarchy. The variables used for explanation are organized into a three-level hierarchy, where each level builds on the previous one (Table II). At the base are first-level variables $\mathcal{X}$, extracted directly from a state s in the MCTS tree. Each state encodes a request $r(s)$ and a vehicle $v(s)$. Second-level variables $\mathcal{V}$ are derived by applying scorer functions f_j to the base-level variables at state s, producing metrics such as drop-off delay or the probability of a timing violation. For example, $\mathsf{vio}_d(s) = \mathsf{eta}(s) - t_d(s)$ captures the estimated delay relative to the scheduled drop-off time. Finally, third-level logic expressions Φ are defined by applying comparison functions g_k over second-level metrics from multiple states. For instance, $\phi_1(s_1, s_2) = AG(\mathsf{vio}_d(s_1) < \mathsf{vio}_d(s_2))$ checks whether all future paths from state s_1 consistently result in lower timing violations compared to those from s_2. This hierarchy of $\mathcal{X} \to \mathcal{V} \to \Phi$ allows LogiEx to support explanations ranging from factual retrieval to complex comparative reasoning over branching futures.

C. Performing Human-Guided Search

HuGS queries cannot be answered from the existing search tree, as they involve modifications to the planning conditions. Thus, LogiEx re-engages MCTS to generate a new search tree under the updated user-specified constraints before producing an explanation. In addition to the logic templates in Table II, these queries are paired with search-condition variables that guide HuGS execution, such as $\mathsf{breakdown(v)}$ and $\mathsf{congestion(r)}$, where v denotes a vehicle and r a passenger request. Representative examples include queries such as "What are the potential consequences of placing the passenger in this alternative vehicle?", "What occurs when traffic becomes congested?", "What happens if the assigned vehicle breaks down?", "What should we do if this trip includes two more passengers?", and "What is the reassignment plan for passengers currently on vehicle 2 if it is inoperable?". After executing the new MCTS search, LogiEx applies the Scorer to the resulting tree using the same logical evaluation and aggregation process described earlier. LogiEx currently supports five HuGS query types in the paratransit testbed, each corresponding to one of the conditions above.

D. Obtaining Logic-Guided Evidence

To obtain quantitative and qualitative logical evidence guided by the generated explanation logic, we define scorer functions (e.g., f and g) that take the MCTS tree comprising states and actions as input, and return numerical or Boolean values based on specified evaluation criteria.

Scoring Overview. The LogiEx Scorer serves as the analytical component of our framework, translating high-level natural language queries into formal evaluation procedures over the

MCTS tree. Once the LLM generates an explanation logic corresponding to a user query, the scorer functions interpret this logic in a structured, measurable form. These functions operate at two complementary levels: base-level evidence, which links logic variables to concrete states, and second-level evidence, which aggregates values across relevant states to yield interpretable quantitative summaries.

Base-level logical evidence is obtained by identifying the target state corresponding to the variable through tree traversal. Thus, LogiEx implements a breadth-first tree traversal to locate the relevant state. More advanced methods can be substituted for efficiency. Second-level evidence is computed by averaging quantitative results across all states in the search tree whose states match the queried condition (e.g. formulas provided in the right-most column of Table II). For example, if one asks about the delay for request r_1 when assigned to vehicle v_1, we aggregate the delay data from all states in the tree where request r_1 is assigned to vehicle v_1, then average the results to provide the final output.

Logic Comparison Layer. While the first two levels quantify local or aggregated outcomes, users often seek comparative or causal insights, such as contrasting delays, rewards, or constraint violations across alternative branches of the search tree. To support such reasoning, LogiEx employs Computation Tree Logic (CTL) [21] to formally specify which portions of the sampled MCTS tree are relevant to the query. The tree is modeled as $\tau = \langle S, \rightarrow, s^0 \rangle$, where S is the set of states, $\rightarrow \subseteq S \times S$ denotes parent–child transitions, and s^0 is the root.

LogiEx uses standard CTL syntax (e.g., $AX\,\phi$, $EX\,\phi$, $AF\,\phi$, $AG\,\phi$) to describe temporal conditions over branching executions. Primitive propositions p encode state-level properties such as ON_TIME ($t_{est} \leq t_{requested}$) or LATE ($t_{est} > t_{requested}$). Temporal operators then determine how these properties propagate along the tree: for example, $AG\,p$ requires that p hold across all descendants, while $EF\,p$ asserts that some descendant satisfies p. Applying CTL model checking on a subtree (τ, s^i) therefore yields a precisely defined subset of states $\{\, s \in S^i \mid (\tau, s) \models \phi \,\}$ that satisfy the queried logical condition. CTL is used strictly for this selection task, which identifies which states matter for a comparison.

Transition to Quantitative Evidence. Once the relevant state sets have been selected, LogiEx transitions from Boolean logic to numerical evaluation. A real-valued function $\mu_\phi(s)$ assigns a quantitative score to each state s satisfying ϕ (e.g., a delay magnitude or reward value). These scores form the basis for the weighted aggregation and comparison procedures introduced next. In this way, CTL provides the structural filter, and the numerical layer provides the substantive values used to answer comparative user queries.

Aggregation Comparisons. Given the quantitatively scored states selected by CTL, LogiEx aggregates these values to produce evidence suitable for comparison. Let $\mu_\phi : S \rightarrow \mathbb{R}$ be the real-valued evaluation function associated with formula ϕ, assigning a numerical score to each state s (e.g., a delay magnitude or reward measure). For consistency with earlier notation, we denote the estimated event time as $t_{est} = \mathsf{eta}(s)$ and the requested event time as $t_{requested} = t_d(s)$, yielding the delay quantity $\mathsf{vio}_d(s) = t_{est} - t_{requested}$.

Algorithm 1 LogiEx Variable Scorer

Input: Base variables $\mathcal{X}$, Derived variables $\mathcal{V}$, Logic formulas Φ, MCTS tree τ, scorers f, g
Output: Evidence table evi
Initialize $evi \leftarrow \{\}$
`/* Base-level evidence */`
for each $X \in \mathcal{X}$ **do**
  $evi[X] \leftarrow f(X)$
end for
`/* Second-level evidence */`
for each $V \in \mathcal{V}$ **do**
  for each relevant state s in τ **do**
    $evi[V, s] \leftarrow g(V, s, evi)$
  end for
end for
`/* Logic comparison evidence */`
for each $\phi \in \Phi$ **do**
  for each anchor state s^* in τ **do**
    Evaluate $(\tau, s^*) \models \phi$ via CTL model checking
    Aggregate satisfaction using $\mathrm{op}_\phi \in \{avg, pct\}$
    Store result in $evi[\phi, s^*]$
  end for
end for
Return evi

Aggregation is performed over the subtree rooted at state s^i. Let S^i be the set of all states reachable from s^i, and let $S_\phi = \{\, s \in S^i \mid (\tau, s) \models \phi \,\}$ denote the subset selected by CTL. Each state s is weighted by its visit count $w(s)$ from MCTS. LogiEx supports two operators for aggregation: visit-weighted average (avg) and visit-weighted percentage (pct). For $\mathrm{op} = \mathsf{avg}$, the aggregate is defined as $\mathrm{Aggregate}_\phi(s^i) = \sum_{s \in S_\phi} w(s)\, \mu_\phi(s) \,/\, \sum_{s \in S_\phi} w(s)$. For $\mathrm{op} = \mathsf{pct}$, it is defined as $\mathrm{Aggregate}_\phi(s^i) = \sum_{s \in S^i} w(s)\, \mathbb{1}[(\tau, s) \models \phi] \,/\, \sum_{s \in S^i} w(s)$, where $\mathbb{1}[\cdot]$ is the indicator function.

The avg operator yields a weighted quantitative measure (e.g., expected delay), while pct yields the visit-weighted proportion of states satisfying ϕ. This formulation ensures that percentages reflect the underlying distribution of MCTS evidence rather than trivially evaluating to 1, enabling meaningful quantitative comparisons across subtrees.

Scope. It is important to note that CTL formulas in LogiEx are evaluated over the *sampled* MCTS search tree generated under a finite computational budget, rather than over the full state space of the underlying MDP. Consequently, the satisfaction or violation of a formula reflects properties of the explored search structure at explanation time, not formal guarantees about the entire planning domain. This interpretation aligns with the practical goal of LogiEx to provide verifiable, search-grounded evidence for human understanding, while acknowledging that the completeness of logical evaluation is inherently constrained by the exploration depth and budget of the MCTS process. The mechanisms of tree traversal for evidence retrieval, CTL-based model checking for logical validation, and quantitative scoring for aggregation form the

backbone of LogiEx's logic-driven explanation process. This layered approach not only enables the system to translate symbolic logic into interpretable evidence that directly aligns with user queries, but also bridges the gap between formal verification, numerical interpretability, and LLM-based natural language reasoning.

E. Retrieving Contextual Explanations

To deliver domain-informed and contextually grounded explanations, we employ a Retrieval-Augmented Generation (RAG) framework that merges the intrinsic reasoning ability of LLMs with a curated external knowledge base. Providing explanations for MCTS-based paratransit routing is a domain-specific, knowledge-intensive task; hence, the LLM agent must act as a domain expert capable of producing accurate, credible, and verifiable answers.

Knowledge Base Construction. We prepared a lightweight yet comprehensive knowledge base of approximately 3,000 words (about 4,000 tokens), divided into 34 modular chunks to support efficient, low-latency retrieval suitable for deployment in real-time CPS. The knowledge base covers background information on paratransit operations, the MCTS algorithm, reward formulation, constraints, and representative expert-generated question–answer pairs. Each chunk is designed to fit within the context window of modern LLMs while maintaining conceptual completeness.

Indexing and Retrieval. During indexing, all text chunks are encoded into vector embeddings using the *text-embedding-3-small* model, one of OpenAI's latest high-performance embedding models, and stored in a vector database. At query time, the user query is converted into its embedding e_q, and cosine similarity is computed with each information chunk embedding $e_i \in \mathcal{I}$. The top k chunks are ranked by similarity scores, and only those exceeding a predefined relevance threshold ε are retrieved. If no retrieved chunks surpass the threshold, the LLM is instructed to refrain from answering, ensuring reliability. This filtering mechanism prevents hallucinations and maintains factual consistency between retrieved evidence and generated explanations.

Augmentation and Generation. The selected chunks are concatenated with the user query to form an expanded context prompt, which is passed to the LLM for generation. Unlike one-to-one matching systems, our retrieval process allows a query to map to multiple semantically related chunks, enabling broader and more contextual responses. RAG's three-stage process: Retrieval, Augmentation, and Generation [22], therefore supports open-ended, multi-faceted reasoning grounded in domain knowledge, while reducing the information overload typically encountered in direct-context prompting.

F. Generating Final Explanations

As illustrated in Figure 4, once LogiEx obtains the list of logical evidence or the retrieved domain knowledge, the system engages with a Question-Answering LLM to generate the final response. With the obtained evidence or knowledge, we instruct the LLM to generate a user-friendly answer via prompts. For query categories 1 and 2, we provide the LLM with three key pieces of information: the original user query, the evidence variables used, and the result from the scorer function obtained in the previous step. For query category 3, we provide the original user query and the retrieved knowledge. The LLM is then tasked with generating the answer, as demonstrated in the example in Figure 3.

IV. Evaluations

We conduct quantitative evaluations to assess LogiEx's performance and a user study to understand end-users' preferences, guided by the following research questions (RQs): **RQ1:** Are existing LLMs capable of accurately answering different categories of user queries about MCTS and the MDP? **RQ2:** Does LogiEx outperform existing LLMs in generating factually accurate and relevant explanations? **RQ3:** Do the individual components of LogiEx, including logic classification and logic-template generation, contribute effectively to its overall performance? **RQ4:** Do end-users prefer the open QA format of LogiEx over other explanation formats?

A. Quantitative Evaluations

We evaluate three LLMs as backbone models. GPT-4 [23] serves as the primary language model for LogiEx and is integrated into the chatbot interface deployed in our user study. For comparison, we also assess GPT-4o [23], leveraging its built-in code interpreter and file search capabilities as part of a baseline system. Both GPT-4 and GPT-4o are developed by OpenAI and accessed via the official Python API. Additionally, we include Llama 3.1 [24] in our evaluation, using it both as a standalone baseline and as an alternative backbone integrated within the LogiEx framework.

Testing Dataset. We systematically synthesized 620 distinct queries as inputs, designed in consultation with field experts and informed by results from previous user surveys. Each method in our quantitative evaluations was tested with 1,860 queries per task through repeated testing. Each query is fully annotated with category ground truth, logic ground truth, and reference ground truth explanations generated by completing manually defined templates.

Evaluation Method. We compare the explanations generated by LogiEx and baseline LLMs against ground truth narrative paragraphs using two established evaluation metrics: BERTScore [25] and FactCC [26]. BERTScore quantifies the semantic similarity between generated and reference texts by leveraging contextualized embeddings. Higher BERTScore values indicate greater alignment in meaning. FactCC (Salesforce), trained on the CNN/DailyMail dataset, evaluates factual consistency between generated outputs and reference texts. It produces a binary judgment (true/false) for each explanation pair. We report the percentage of outputs classified as "true," with higher scores indicating better factual alignment with the ground truth. Here, @1 and @3 represent the top–1 and best–of–3 generations per query, where the best–of–k selection is based on model confidence scores.

TABLE III: Evaluation of LogiEx vs. baselines across evidence complexities. (BERTScore is abbreviated as **B**; FactCC as **F**.)

Method	Base-Level				Second-Level				Logic Comparison				Overall			
	F@1	F@3	B@1	B@3	F@1	F@3	B@1	B@3	F@1	F@3	B@1	B@3	F@1	F@3	B@1	B@3
Llama3.1	36.88%	43.75%	11.88%	18.75%	11.79%	17.14%	02.86%	03.57%	68.75%	77.50%	17.50%	30.00%	25.77%	34.62%	06.15%	12.31%
LogiEx-Llama	91.88%	96.88%	89.38%	98.13%	61.07%	71.79%	92.50%	98.21%	76.25%	96.25%	80.00%	93.75%	67.88%	83.27%	86.54%	97.50%
GPT-4o	67.50%	70.63%	66.25%	73.75%	26.43%	33.21%	33.57%	47.50%	71.25%	45.00%	41.25%	48.75%	42.31%	51.15%	40.00%	55.77%
LogiEx-GPT	96.88%	97.50%	99.38%	100.00%	59.64%	70.71%	86.07%	91.07%	78.75%	86.25%	86.25%	97.50%	72.12%	81.35%	88.46%	94.81%

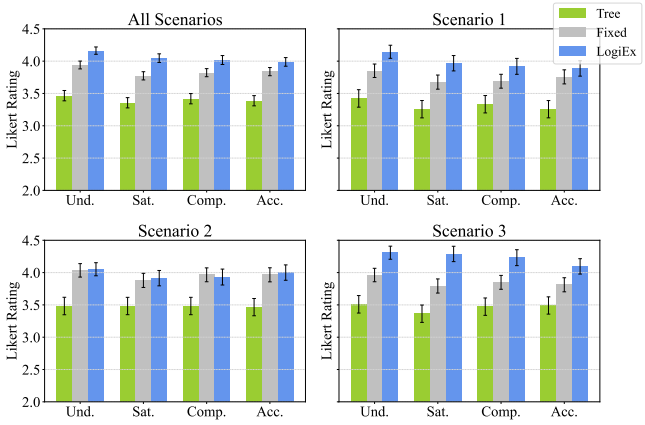

Fig. 5: User Study Results with 95% CI.

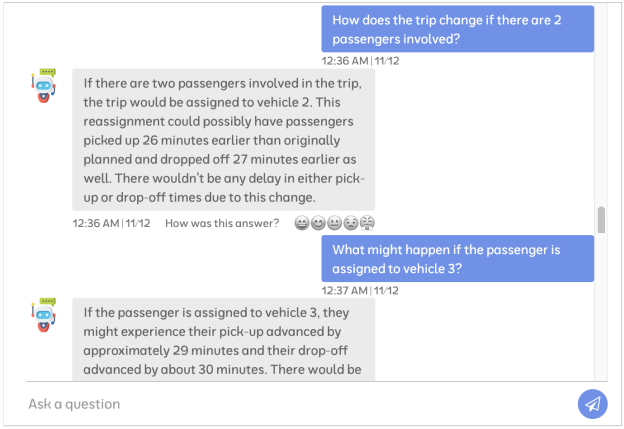

Fig. 6: LogiEx QA Interface in User Study.

TABLE IV: Component-wise Evaluation of LogiEx.

	Query Type Classification		Logic Generation
Complexity	**LogiEx**	**BERT**	**Acc@1**
Base-Level	98.75%	99.38%	100.00%
Second-Level	93.68%	90.00%	99.21%
Logic Comparison	100.00%	98.75%	98.75%
Overall	95.81%	93.55%	99.35%

Baseline LLMs Performance. To investigate *RQ1*, we evaluated the performance of two LLMs when provided with the MCTS tree as context along with prompt instructions, with the task of responding to manually prepared queries. As shown in Table III, the best result achieved was a 51.15% FactCC score for GPT-4o. This outcome was expected, as Llama3.1 lacks the ability to directly access, search, and retrieve relevant information from uploaded documents. Consequently, Llama3.1 frequently refused to answer the questions. Additionally, the highest BERTScore achieved was 55.77% for GPT-4o across the three runs. In summary, both results suggest that basic LLMs struggle to generate relevant and factually accurate explanations directly.

Factual Consistency with LogiEx Integration. To explore *RQ2*, we compared the performance of basic LLMs with LogiEx, integrated using GPT-4 and Llama3.1 models. For queries answered using each type of evidence, LogiEx consistently outperformed the basic LLMs across all categories. Specifically, for the overall FactCC score, we observed a 2.40× improvement with LogiEx using Llama3.1 and a 1.59× improvement with LogiEx using the GPT-4 model. The improvement in BERTScore was even more pronounced, with an overall increase of 7.92× for the Llama3.1 backbone model and 1.70× for the GPT-4 backbone model, respectively. In conclusion, LogiEx significantly outperforms other LLM baselines, providing more factually precise explanations.

Component-wise Evaluation of LogiEx. To explore *RQ3*, we evaluate the performance of two key components of LogiEx: the accuracy of classifying user questions into one of the thirty-one guide queries, and the accuracy of generating the correct evidence statements. For both components, we provide accuracy metrics by evidence type. Queries that can be answered using the knowledge base are excluded from this analysis, as they do not rely on evidence from the MCTS tree, nor do they require classification into guide queries. As shown in Table IV (left), the fine-tuned BERT model [27] performs better for query types requiring base-level evidence, whereas LogiEx outperforms it in other categories as well as overall. As shown in Table IV (right), LogiEx consistently and reliably generates the correct variable and logic statements for extracting evidence from the search tree. The overall accuracy across three runs was 99.35%.

B. User Study

To explore *RQ4*, we conducted an Institutional Review Board (IRB)-approved user study and developed a web interface for LogiEx and deployed it on Google Cloud, where LogiEx interacts with anonymous participants as an online chatbot. We run an MCTS algorithm on the backend to facilitate real-time HuGS explanations. Out of the 89 recruited participants with diverse expertise and technical backgrounds,

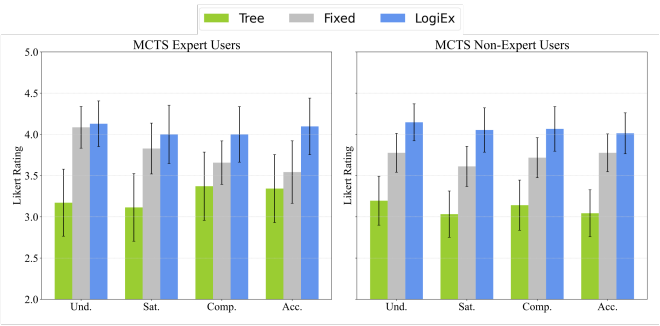

Fig. 7: MCTS Expert vs. MCTS Non-Expert Users.

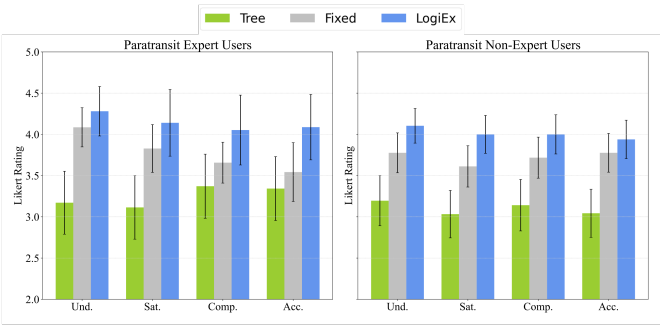

Fig. 8: Paratransit Expert vs. Paratransit Non-Expert Users.

28.09% self-identified as experts in MCTS. Additionally, 31.46% reported as experts in paratransit, including *field operators from two local transit departments* and engineers working on paratransit-related projects. During the survey, participants take on the role of transit dispatchers to simulate a paratransit dispatching scenario and interact with LogiEx through three scenarios with varying complexity. Participants view the status of vehicles, the passenger location, and the vehicle assignment decisions made by the MCTS. Participants are free to ask any questions related to the MDP and MCTS.

Metrics. We use four metrics [9] to evaluate user preferences between explainability methods. Participants are asked to provide rating using a 5-point Likert scale based on the following questions. *Understandability:* I understand this explanation of the planning result. *Satisfaction:* This explanation of the planning result is satisfying. *Completeness:* This explanation of the planning result seems complete. *Assessability:* This explanation helps me to assess the reliability of the planning algorithm.

Baselines. We employ two baselines for MCTS explainability. In the first baseline, users are presented with a map of the scenario and an interactive visualization of the MCTS tree. This tree allows users to explore details such as the reward, state, and actions at each node. The second baseline uses explanations generated from predefined natural language templates with checkboxes [28]. Users are provided with a list of queries and can select the ones they are interested in, displaying the corresponding explanations.

Result Analysis. Figure 5 shows the results of the user study by scenario. Notably, LogiEx received the highest overall ratings across all four metrics. The smallest difference between LogiEx and the baselines was observed in Scenario 2, which is the simplest scenario where only four vehicles were present. LogiEx's advantage was most evident in Scenario 3, which involved eight vehicles near the trip origin that significantly increased complexity. A paired t-test revealed statistically significant improvements over two baselines ($p \leq 0.01$) across all metrics. In this case, reading through numerous pre-defined explanations became cumbersome, which showed LogiEx's effectiveness in managing more challenging situations.

User study results separated by participant expertise are provided in Figures 7 and 8, where we observed that LogiEx was preferred by non-MCTS experts and received the highest ratings across all four metrics, with Likert scale scores of 4.13 ± 0.67 and 4.10 ± 0.83 for understandability and accessibility. Similarly, LogiEx received the highest ratings among paratransit experts across all metrics, rated as $1.35\times$ and $1.05\times$ more understandable than both baselines. Additionally, LogiEx was rated as $1.23\times$ and $1.16\times$ more accessible than the two baselines, respectively. This result supports the conclusion that end-users with varying backgrounds prefer the LogiEx explanations over the explanations provided by search tree visualizations or pre-defined questions and answers.

At the end of each scenario, we have included an open-ended feedback section for users to provide their comments. We found that many comments highlighted the understandability and flexibility offered by LogiEx, as listed in Table V.

C. Runtime Efficiency

Lastly, to assess the computational efficiency of LogiEx, we measured the average processing time for evaluating logic formulas across different evidence levels and MCTS tree sizes. Table VI reports the runtime (in seconds) for base-level, second-level, and CTL-based logic evaluations under two sampling configurations (k=10 and k=50). The results show that LogiEx scales efficiently with increasing tree size and formula count. Even for large trees containing over 1,000 nodes, all CTL evaluations completed within 0.45 seconds on an Intel i9-10850K processor, indicating that the logic-scoring component introduces negligible overhead to the overall explanation process.

V. Related Work

A. Explainable Sequential Planning

There has been previous effort within the research community to develop explanations for sequential planning algorithms including heuristic search algorithms and reinforcement learning [29, 30, 31, 32]. Broadly, various approaches have been proposed to explain the behavior of these algorithms, focusing on aspects such as example-based explanations, visualizations of planning results, and highlighting feature importance [33]. For example, Juozapaitis et al. [34] proposes a minimally sufficient method to explain why certain actions are selected while others are avoided, based on the decomposition of

TABLE V: Representative qualitative result of user-perceived clarity and flexibility in explanation

ID	User Feedback
U1	"The chatbot explainer makes it much easier to understand because I can formulate any specific doubts I have and the chatbot can address them directly to help me understand the algorithm and why it chose a specific vehicle over another."
U2	"I can understand the essential logics of this prompt; the system is capable of responding to complex questions as well."
U3	"This version of explanation makes more sense and is much clearer than the previous two. The user has more flexibility and fewer restrictions to ask questions regarding different aspects. The answers are more straightforward, understandable, and comprehensive."
U4	"This explanation helps me understand more background info, which I didn't have or missed in the previous two setups. Although the waiting time is a bit long, overall it reacts within a tolerable time."
U5	"It can perfectly answer the suggested questions and also provide reasonable answers for general questions. It helped me understand the algorithm better. The check-box style questions also helped by providing detailed and specific examples."

TABLE VI: Runtime efficiency comparison by logic type for different tree sizes and counts k.

Tree Size	Base Level		Second Level		Logic Comparison	
	k=10	k=50	k=10	k=50	k=10	k=50
23	0.0011	0.0075	0.0028	0.0134	0.0092	0.0291
42	0.0011	0.0073	0.0067	0.0157	0.0103	0.0252
50	0.0010	0.0062	0.0030	0.0224	0.0101	0.0205
1086	0.0023	0.0051	0.0675	0.2534	0.1389	0.4253
1500	0.0023	0.0060	0.0637	0.2257	0.1584	0.4430

reinforcement learning agents' rewards. On the other hand, in recent work, De et al. [35] argues that non-symbolic AI algorithms (those that do not make decisions based on explicit rules) can be effectively explained through post-hoc analysis, where the outcomes are assessed for compliance against a set of rules. Our work largely aligns with this concept. However, despite being a widely adopted algorithm for sequential planning, there is still limited work focused on explaining the planning process of MCTS. Baier et al. [11, 36] propose foundational approaches to explain MCTS by focusing on possible future scenarios of plans and reducing the complexity of the search tree through summarization. Specifically, they suggest that explanations can be generated post-hoc or through collaborative search with human agents, addressing multiple families of questions frequently asked by users. An et al. [28, 37] extend explainable MCTS into the real-world context of transit planning, where the planning task is inherently complex, and multiple user requirements are specified throughout the process. The authors leverage formal logic to evaluate possible future scenarios explored in the MCTS search tree, offering rigorous yet accessible explanations to non-technical users. Aligned with this line of work, our paper is among the first to integrate formal logic with LLM-based reasoning to explain the full MCTS planning process, enabling both rigor and open-ended, user-driven interpretability.

B. LLMs for Enhanced XAI User-Friendliness

The development of LLMs opens new avenues for enhancing the user-friendliness of xAI methods [38, 39, 12, 40]. For example, Wu et al. [41] advocates for incorporating LLMs into xAI frameworks. The author argues that xAI can contribute to the evolution of LLMs by enhancing their explainability, trustworthiness, and fairness. More importantly, they suggest various high-level strategies for improving the user-friendliness of existing xAI methods through LLMs, particularly by utilizing the capability of LLMs to simplify complex concepts into understandable language. The work proposed in Bills et al. [42] is another example, where they attempt to interpret the behavior of LLMs by employing another LLM to identify patterns of neuron activation. Zytek et al. [39] also supports the use of LLMs to enhance the usability of xAI methods. They outline several future directions, including the design of more effective prompts and the integration of external data. Many existing xAI methods aim to elucidate the inner workings of AI and machine learning techniques for experts or developers. Yet, even those explanations would often demand a certain level of expertise, leaving laypeople behind [43, 44]. With the advent of LLMs, there is potential to bridge this gap [38, 45]. Our work, in contrast, combines the expressive flexibility of LLMs with the formal precision of logic-based validation, ensuring that the generated explanations remain both understandable to non-experts and verifiably correct.

VI. Conclusion

We present LogiEx, a logic-grounded explainability framework for MCTS–based sequential planning in CPS. LogiEx rigorously addresses user queries through three complementary modes: post–hoc explanations, HuGS with user–defined conditions, and RAG based on a curated knowledge base. By integrating LLMs, LogiEx supports open–form queries while ensuring factual consistency through formal logic–guided evidence generation. Empirical evaluations show that LogiEx achieves up to 1.7× higher factual consistency and up to 7.9× higher semantic similarity compared to explanations generated by other LLMs. In user studies, LogiEx explanations were consistently preferred and rated as up to 1.35× more understandable by domain experts.

Acknowledgments

This work was supported in part by the National Science Foundation under grants 2443803, 2427711, 1952011, and 2531369. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] J. Shi, J. Wan, H. Yan, and H. Suo, "A survey of cyber-physical systems," in *2011 international conference on wireless communications and signal processing (WCSP)*. IEEE, 2011, pp. 1–6.

[2] M. Ma, J. Stankovic, E. Bartocci, and L. Feng, "Predictive monitoring with logic-calibrated uncertainty for cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 5s, pp. 1–25, 2021.

[3] Z. Chen, Z. An, J. Reynolds, K. Mullen, S. Maritini, and M. Ma, "Logidebrief: A signal-temporal logic based automated debriefing approach with large language models integration," in *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, J. Kwok, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2025, pp. 9582–9590, aI and Social Good. [Online]. Available: https://doi.org/10.24963/ijcai.2025/1065

[4] J. Marques-Silva and A. Ignatiev, "Delivering trustworthy ai through formal xai," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 11, 2022, pp. 12 342–12 350.

[5] M. Saqlain, S. Ali, and J. Lee, "A monte-carlo tree search algorithm for the flexible job-shop scheduling in manufacturing systems," *Flexible Services and Manufacturing Journal*, vol. 35, no. 2, pp. 548–571, 2023.

[6] D. Weng, R. Chen, J. Zhang, J. Bao, Y. Zheng, and Y. Wu, "Pareto-optimal transit route planning with multi-objective monte-carlo tree search," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 1185–1195, 2020.

[7] H. D. Wang, S. Bae, X. Sun, Y. Thatigotla, and M. Ma, "Exact: A meta-learning framework for precise exercise segmentation in physical therapy," in *Proceedings of the ACM/IEEE 16th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2025)*, 2025, pp. 1–11.

[8] B. Luo, A. Pettet, A. Laszka, A. Dubey, and A. Mukhopadhyay, "Scalable decision-making in stochastic environments through learned temporal abstraction," in *The Thirteenth International Conference on Learning Representations*.

[9] R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman, "Metrics for explainable ai: Challenges and prospects," *arXiv preprint arXiv:1812.04608*, 2018.

[10] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[11] H. Baier and M. Kaisers, "Explainable search," in *2020 IJCAI-PRICAI Workshop on Explainable Artificial Intelligence*, 2020, p. 178.

[12] X. He, X. Bresson, T. Laurent, A. Perold, Y. LeCun, and B. Hooi, "Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=RXFVcynVe1

[13] V. Swamy, D. Romano, B. S. Desikan, O.-M. Camburu, and T. Käser, "illuminate: An llm-xai framework leveraging social science explanation theories towards actionable student performance feedback," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 27, 2025, pp. 28 431–28 439.

[14] S. Schwartz, A. Yaeli, and S. Shlomov, "Enhancing trust in llm-based ai automation agents: New considerations and future challenges," in *International Joint Conference on Artificial Intelligence*, 2023.

[15] Y. Liu, Y. Yao, J.-F. Ton, X. Zhang, R. Guo, H. Cheng, Y. Klochkov, M. F. Taufiq, and H. Li, "Trustworthy llms: a survey and guideline for evaluating large language models' alignment," in *Socially Responsible Language Modelling Research*.

[16] M. Yu, F. Meng, X. Zhou, S. Wang, J. Mao, L. Pan, T. Chen, K. Wang, X. Li, Y. Zhang *et al.*, "A survey on trustworthy llm agents: Threats and countermeasures," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 2025, pp. 6216–6226.

[17] G. W. Klau, N. Lesh, J. Marks, and M. Mitzenmacher, "Human-guided search," *Journal of Heuristics*, vol. 16, pp. 289–310, 2010.

[18] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[19] M. Wilbur, S. U. Kadir, Y. Kim, G. Pettet, A. Mukhopadhyay, P. Pugliese, S. Samaranayake, A. Laszka, and A. Dubey, "An online approach to solve the dynamic vehicle routing problem with stochastic trip requests for paratransit services," in *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2022, pp. 147–158.

[20] W. Joe and H. C. Lau, "Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers," in *Proceedings of the international conference on automated planning and scheduling*, vol. 30, 2020, pp. 394–402.

[21] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic," in *Workshop on logic of programs*. Springer, 1981, pp. 52–71.

[22] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.

[23] R. OpenAI, "Gpt-4 technical report. arxiv 2303.08774," *View in Article*, vol. 2, no. 5, 2023.

[24] M. I. Llama, "3.1: Our most capable models to date," 2024.

[25] T. Zhang*, V. Kishore*, F. Wu*, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SkeHuCVFDr

[26] W. Kryściński, B. McCann, C. Xiong, and R. Socher, "Evaluating the factual consistency of abstractive text summarization," in *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, 2020, pp. 9332–9346.

[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[28] Z. An, H. Baier, A. Dubey, A. Mukhopadhyay, and M. Ma, "Enabling mcts explainability for sequential planning through computation tree logic," in *27th European Conference on Artificial Intelligence, ECAI 2024*. IOS Press, 2024, pp. 4068–4075.

[29] K. Boggess, S. Kraus, and L. Feng, "Toward policy explanations for multi-agent reinforcement learning," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

[30] M. Fox, D. Long, and D. Magazzeni, "Explainable planning," in *Proceedings of IJCAI-17 Workshop on Explainable Planning*, 2017.

[31] M. Cashmore, A. Collins, B. Krarup, S. Krivic, D. Magazzeni, and D. Smith, "Towards explainable ai planning as a service," in *2nd ICAPS Workshop on Explainable Planning*, 2019.

[32] T. Chakraborti, S. Sreedharan, Y. Zhang, and S. Kambhampati, "Plan explanations as model reconciliation: moving beyond explanation as soliloquy," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 156–163.

[33] S. Milani, N. Topin, M. Veloso, and F. Fang, "Explainable reinforcement learning: A survey and comparative review," *ACM Computing Surveys*, vol. 56, no. 7, pp. 1–36, 2024.

[34] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez, "Explainable reinforcement learning via reward decomposition," in *IJCAI/ECAI Workshop on explainable artificial intelligence*, 2019.

[35] H. de Bruijn, M. Warnier, and M. Janssen, "The perils and pitfalls of explainable ai: Strategies for explaining algorithmic decision-making," *Government information quarterly*, vol. 39, no. 2, p. 101666, 2022.

[36] H. Baier and M. Kaisers, "Towards explainable mcts," in *2021 AAAI Workshop on Explainable Agency in AI*, 2021, p. 178.

[37] Z. An, X. Wang, H. Baier, Z. Chen, A. Dubey, T. T. Johnson, J. Sprinkle, A. Mukhopadhyay, and M. Ma, "Combining llms with a logic-based framework to explain mcts," in *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, 2025, pp. 2405–2407.

[38] E. Cambria, L. Malandri, F. Mercorio, N. Nobani, and A. Seveso, "Xai meets llms: A survey of the relation between explainable ai and large language models," *arXiv preprint arXiv:2407.15248*, 2024.

[39] A. Zytek, S. Pidò, and K. Veeramachaneni, "Llms for xai: Future directions for explaining explanations," *arXiv preprint arXiv:2405.06064*, 2024.

[40] N. Kroeger, D. Ley, S. Krishna, C. Agarwal, and H. Lakkaraju, "In-context explainers: Harnessing llms for explaining black box models," *arXiv preprint arXiv:2310.05797*, 2023.

[41] X. Wu, H. Zhao, Y. Zhu, Y. Shi, F. Yang, T. Liu, X. Zhai, W. Yao, J. Li, M. Du *et al.*, "Usable xai: 10 strategies towards exploiting explainability in the llm era," *arXiv preprint arXiv:2403.08946*, 2024.

[42] S. Bills, N. Cammarata, D. Mossing, H. Tillman, L. Gao, G. Goh, I. Sutskever, J. Leike, J. Wu, and W. Saunders, "Language models can explain neurons in language models," *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)*, vol. 2, 2023.

[43] A. Das and P. Rad, "Opportunities and challenges in explainable artificial intelligence (xai): A survey," *arXiv preprint arXiv:2006.11371*, 2020.

[44] Y. Rong, T. Leemann, T.-T. Nguyen, L. Fiedler, P. Qian, V. Unhelkar, T. Seidel, G. Kasneci, and E. Kasneci, "Towards human-centered explainable ai: A survey of user studies for model explanations," *IEEE transactions on pattern analysis and machine intelligence*, vol. 46, no. 4, pp. 2104–2122, 2023.

[45] A. Castelnovo, R. Depalmas, F. Mercorio, N. Mombelli, D. Potertì, A. Serino, A. Seveso, S. Sorrentino, and L. Viola, "Augmenting xai with llms: A case study in banking marketing recommendation," in *World Conference on Explainable Artificial Intelligence*. Springer, 2024, pp. 211–229.