Conversational Goal-Conflict Explanations in Planning via Multi-Agent LLMs

Submission #4858

Abstract

When automating plan generation for a real-world sequential decision problem, the goal is often not to replace the human planner, but to facilitate an iterative reasoning and elicitation process, where the human's role is to guide the planner according to their preferences and expertise. In this context, explanations that respond to users' questions are crucial to improve their understanding of potential solutions and increase their trust in the system. To enable natural interaction with such a system, we present a multi-agent Large Language Model (LLM) architecture that is agnostic to the explanation framework and enables user- and context-dependent interactive explanations. We also describe an instantiation of this framework for goal-conflict explanations, which we use to conduct a pilot case study comparing the LLM-powered interaction with a more conventional template-based explanation interface.

1 Introduction

One can think of planning as the task of finding a plan that satisfies a set of properties, but also as the iterative process that starts before the goals, objectives and preferences are fully defined [Smith, 2012], and ends when a plan is found that is satisfactory for all parties involved. From this perspective, explanations serve the purpose of accelerating the convergence of preferences elicitation by humans. Interactive planning has become a relatively well-accepted paradigm, yet the crucial role of interactive *explanations* in that process has been less widely investigated. Explanations are by nature interactive, as explanation is something that one person (the explainer) does for the sake of another (the explainee). The request for an explanation arises because the explainee has a conceptual problem [Bromberger, 1962; Achinstein, 1980]; there is something they do not understand. The explainer's task is to resolve the conceptual problem. However, fixing a conceptual problem may not be a one shot deal; it may take several interactions to arrive at an explanation that is satisfactory for the explainee – hence the inherent interactivity of explanations. Furthermore, previous work in psychology and human-machine interaction [Liao *et al.*, 2020; Dazeley *et al.*, 2021; Lakkaraju *et al.*, 2022; Zhang *et al.*, 2024] argue that interactive explanations are more interpretable and effective than static ones [Miller, 2019]. Other methods were proposed for interactive explanations [Nguyen *et al.*, 2023; Shen *et al.*, 2023; Slack *et al.*, 2023; State *et al.*, 2023; Feldhus *et al.*, 2023]; they use a similar high level architecture as Figure 1, but none of them are designed for planning systems.

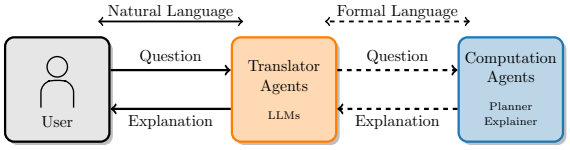

Figure 1: Architecture of our approach to interactive planning with explanations. Translator agents based on LLMs translate the user input into the formal language required by the computation agents.

If explanations are ideally interactive, then to provide appropriate explanations, we need something with conversational capacities, something that is able to respond to an initial request for an explanation, but also to follow-up questions and remarks. LLMs fill this need; they are conversationally fluent and have impressive flexibility in translating natural language to formal languages, which a symbolic planner even when coupled with a template interface cannot do. However, LLMs cannot do everything. Empirical evidence demonstrates that LLMs cannot reliably perform complex planning tasks independently [Valmeekam *et al.*, 2023] and there are theoretical reasons to suspect models using current transformer architectures never will [Peng *et al.*, 2024]. Nevertheless, they can serve valuable supporting roles [Kambhampati *et al.*, 2024]. Our work extends this view as depicted in Figure 1, showing how LLMs can be effectively integrated into the planning process – specifically for explanations and natural language interaction – while delegating the core computational planning tasks to specialized algorithms designed for that purpose.

To enable natural interactions between users and planning systems, we need to accommodate different ways in which a user formulates their question. To this end, we leverage LLMs in two distinct roles: as a classifier of the question type and as a translator of the question topic. Explanations should resolve a conceptual problem of the explainee, which is typically linguistically introduced by certain types of questions. These types determine the form of the explanation; answers to *why* questions differ from answers to a *what-if* question; e.g., *"Why can't I go to my fitness course?"* versus *"What*

if I go grocery shopping before the fitness course?". However, the type of question by itself does not suffice to provide an explanation; we also need to specify the topic that the explanation must address. For example, a question like *"Why can't I go to my fitness course?"* has a different answer from *"Why can't I visit my friend?"*. The explanation framework responds to the question-type and the question topic, translated into a formal specification, with a formal explanation. To deliver this explanation to the user, we use an LLM as an explanation translator. It converts this formal explanation into a natural language response that can serve as the basis of an extended dialogue in which the original explanation is refined to meet the needs of the user.

The paper is organized as follows. Section 2 introduces the necessary background. Section 3 presents our general framework for iterative planning with explanations, and details where translation between natural and formal language is required. Section 4 provides an implementation of explanations that addresses common user questions in this framework, whereas Section 5 describes our implementation of the LLM-based translators. Finally, Section 6 compares the LLM-based with a template-based version of the framework in a pilot user study.

2 Background

2.1 Large Language Models

Large Language Models (LLMs) are neural networks based on the transformer architecture [Vaswani *et al.*, 2017] that have been pre-trained on vast amounts of text data. These models process text as sequences of tokens and use self-attention mechanisms to capture relationships between different parts of the input. Through their pre-training, LLMs have acquired capabilities in various natural language tasks, including text generation, translation, summarization and understanding complex instructions. Such models are typically referred to as foundation models [Touvron *et al.*, 2023; Bommasani *et al.*, 2021] when it is possible to access their parameters to specialize them to a specific purpose.

LLMs can be adapted in two main ways: fine-tuning, which involves additional training on specific tasks, or in-context learning, which uses carefully crafted prompts to guide the model's behavior [Brown *et al.*, 2020]. Here we use the latter approach, which has been widely adopted since it allows users to adapt LLM behavior without parameter modifications, often requiring fewer than 10 examples to achieve strong performance across many tasks.

Multi-Agent LLM Approaches

Multi-agent LLM approaches involve multiple large language models working collaboratively to solve complex tasks [Guo *et al.*, 2024]. These systems typically leverage individual LLM strengths by assigning specific roles to each agent, allowing for specialization and improved performance through in-context learning tailored to specific functions.

While a single LLM could theoretically be fine-tuned for complex planning tasks, this faces key challenges: insufficient task-specific training data and limited generalization beyond the training distribution. Even approaches using complex in-context learning like chain-of-thought prompting can be unreliable for multi-task scenarios. To address these limitations, we propose decomposing complex tasks into specialized subtasks handled by different LLM agents working in concert. This architecture leverages LLMs' natural strengths in language understanding and translation while mitigating their weaknesses in complex reasoning tasks.

2.2 Planning Formalism

A lifted planning task is a tuple $\tau = \langle \mathcal{P}, \mathcal{O}, \mathcal{A}, I, G \rangle$ where $\mathcal{P}$ is a finite set of first-order predicates and $\mathcal{O}$ a set of objects. P is a ground predicate or **atom** if all variables have been replaced by objects. The **goals** G is a set of atoms. A **state** is a set of atoms; atoms not in the set are assumed to be false in the state. I is the **initial state**. Each action schema $A \in \mathcal{A}$ has a list X_A of parameter variables and a **precondition** pre_A, an **add list** add_A and a **delete list** del_A which are sets of predicates from $\mathcal{P}$ where all variables are replaced by an element in $X_A \cup \mathcal{O}$. We obtain a (ground) action a from an action schema A by replacing all variables X_A in pre_A, add_A, and del_A with an object from $\mathcal{O}$. A action a is **applicable** in a state s if $pre_a \subseteq s$ and $\mathcal{A}(s)$ denotes the set of all applicable actions in s. Applying action a in state s, results in the state $s[\![a]\!] = (s \setminus del_a) \cup add_a$. The state resulting from an iteratively applicable sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$ is denoted by $s[\![\pi]\!]$. A **plan** is an action sequence π such that $G \subseteq I[\![\pi]\!]$. A task τ is **solvable** if a plan exists. By $\Pi(\tau)$ we denote the set of all plans for task τ and by $\tau(G')$ we denote the task $\tau' = \langle \mathcal{P}, \mathcal{O}, \mathcal{A}, I, G' \rangle$

In the following we consider a setting similar to over-subscription planning [Smith, 2004; Domshlak and Mirkis, 2015], where not all the goals can be satisfied, due for example to insufficient resources. But instead of finding a subset of goals that maximize a utility, we are interested in conflicts between a set of reference goals G^{ref}. For a task τ these conflicts and possible resolutions are given by **minimal unsolvable subsets** (MUS) [Eifler *et al.*, 2020a] and **minimal correction sets** (MCS) respectively. A set of goals $C \subseteq G^{\text{ref}}$ is a MUS if $\tau(C)$ is unsolvable but for all $G \subset C$, $\tau(G)$ is solvable. A set of goals $R \subseteq G^{\text{ref}}$ is a MCS if $\tau(G^{\text{ref}} \setminus R)$ is solvable but for all $G \subset R$, $\tau(G^{\text{ref}} \setminus G)$ is unsolvable. By $\mathcal{G}^{\text{MUS}}(\tau, G^{\text{ref}})$ and $\mathcal{G}^{\text{MCS}}(\tau, G^{\text{ref}})$ we denote the set of all MUS and MCS for task τ with respect to the reference goals G^{ref}. Both sets can be exponentially large. The following relation holds between MUS and MCS over the same set of goals G: $\mathcal{G}^{\text{MCS}}(\tau, G) = \mathit{HIT}(\mathcal{G}^{\text{MUS}}(\tau, G))$, where $\mathit{HIT}(\mathcal{S})$ is the set of all minimal hitting sets of the sets in $\mathcal{S}$. For algorithms to compute $\mathcal{G}^{\text{MUS}}(\tau, G)$ we refer to [Eifler *et al.*, 2020a; Eifler *et al.*, 2020b].

Temporal Goals

By using a compilation approach [Edelkamp, 2006; Baier and McIlraith, 2006; De Giacomo *et al.*, 2014] for temporal properties defined in finite linear temporal logic (LTL_f), it is possible to reason about conflicts not only over goal facts, but also over temporal properties of plans [Eifler *et al.*, 2020b]. In the following we assume that all goals are defined in LTL_f over atoms. For the syntax and semantic of LTL_f we refer to [De Giacomo *et al.*, 2014]. With $\mathcal{L}(\tau)$ we denote the set of all well-formed LTL_f formulas of task τ.

LTL_f formulas are interpreted over a finite sequence of states σ. An action sequence π satisfies LTL_f formula ϕ in state s, iff ϕ holds of the state sequence $\sigma(\pi, s)$ that results from executing π in s, i.e. iff $\sigma(\pi, s) \models \phi$. A task $\tau(G)$ with temporal goals G is solvable if there exists an action sequence π such that for all $\phi \in G : \sigma(\pi, I) \models \phi$.

3 Iterative Planning with Explanations

Iterative planning is based on the idea formalized by [Smith, 2012] that it is often not purposeful to compute just one plan. Given conflicting goals and users who have not yet fully formed their preferences, an iterative exploration of possible plans is more suitable. In this context explanations are crucial, especially those that help the user to understand the dependencies between the goals and their preferences. We define a generic framework for iterative planning with explanations, focussing on the points of interaction with users.

The iterative process of determining a final plan is divided into individual steps. Each step δ_i represents one snapshot of the user's exploration of the plan space, defined by a set of reference goals G_i^{ref}, a set of enforced goals $G_i^{\text{enf}} \subseteq G_i^{\text{ref}}$, and a sample plan π_i satisfying G_i^{enf}:

Definition 3.1 (Iteration Step)**.** Given a planning task τ, an **iteration step** is a tuple $\delta = \langle G^{\text{ref}}, G^{\text{enf}}, \pi \rangle$, where G^{ref} is a set of (temporal) goals for τ, $G^{\text{enf}} \subseteq G^{\text{ref}}$, and $\pi \in \Pi(\tau(G^{\text{enf}}))$ if $\tau(G^{\text{enf}})$ is solvable and $\pi = \epsilon$ otherwise.

Based on the sample plan π_i, the user defines the set of reference goals G_{i+1}^{ref} for the next step. They represent the goals and preferences that the user is interested in. In addition, a subset of goals $G_{i+1}^{\text{enf}} \subseteq G_{i+1}^{\text{ref}}$ is selected which must be satisfied by the sample plan π_{i+1} in the next iteration.

The reference goals must be defined in a language the planner understands; here we are using LTL_f. However, this language is not suitable for a lay person as an input language. Thus, we require a goal translator.

Definition 3.2 (Goal Translator)**.** Given a task τ with well-formed temporal goals $\mathcal{L}(\tau)$, a **goal translator** is a function $T_G : \mathit{NL} \mapsto \mathcal{L}(\tau) \cup \epsilon$, that maps a natural language input to a goal $\phi \in \mathcal{L}(\tau)$ and to ϵ if the natural language description does not describe a goal represented by any formula in $\mathcal{L}(\tau)$.

A more advanced goal translator could include a feedback loop with the user to recover from misunderstandings or alert the user that a very similar goal is already considered. For such features, the translator needs to depend on the *interaction context* reflecting the previous interaction with the translator:

Definition 3.3 (Interaction Context)**.** The **interaction context** for translator T is a sequence $\mathcal{C}(T) = c_0 c_1 \cdots c_n$ of interactions $c = \langle \delta, \mathit{IN}, T(\mathit{IN}) \rangle$, where each interaction c is associated with an iteration step δ and contains the translator input *IN* and the translation result $T(\mathit{IN})$.

Explanations can be provided, to facilitate the decision of which goals to enforce in the next iteration. Those explanations can address different objectives such as clarifying the model [Sreedharan *et al.*, 2021], identifying the trade-offs between plan quality measures [Krarup *et al.*, 2024], or providing a better understanding of the dependencies between the goals [Eifler *et al.*, 2020a]. Explanations are provided as answers to specific user questions. Depending on whether the enforced goals G_i^{enf} are satisfiable and therefore a sample plan π_i exists or whether the enforced goals are unsolvable, the user's questions will vary. In the former case, the question may relate to how the sample plan solves the task whereas in the latter, the user is more interested in why G_i^{enf} cannot be satisfied. Formally we define a question as follows.

Definition 3.4 (Question)**.** Given a task τ and a set of question types M_Q, a **question** is a tuple $Q = \langle \mu_Q, \mathit{args} \rangle$ where $\mu_Q \in M_Q$ is the question type and $\mathit{args} \subseteq \wp(\mathcal{L}(\tau))$ (the powerset of $\mathcal{L}(\tau)$) are the question arguments.

The question types M_Q depend on the explanation framework. In Section 4 we introduce the types our explanation framework supports. An example is $\mu_Q = \texttt{S-why-not}$, i. e. "Why is g not satisfied by plan π?", which requires one argument. Roughly, each question word in natural language, e. g. *who, what, why, how* maps to a different type of question.

This is the second point of interaction. Again, we require a translator to allow the user to ask their question in natural language, which is then translated into a formal question.

Definition 3.5 (Question Translator)**.** Given a task τ, and a set of question types M_Q, a **question translator** is a function $T_Q : \mathit{NL} \mapsto (M_Q \times \wp(\mathcal{L}(\tau))) \cup \epsilon$, that maps a natural language expression to a question type and its arguments and to ϵ if no matching question type exists.

Question translation can be divided into two steps. First the question type is identified. Then, the question topic, i. e. the arguments, are translated. This last step can be performed by the goal translator, as the translation tasks are the same.

A context-dependent question translator with access to previously asked questions enables follow-up questions with implicit references. For example, for the questions "Why can I not visit Alice?" and "Can you enforce it?', the visited location or even the entire question argument depend on the context.

The explanation framework computes a set of formal explanations $\mathcal{E}(Q)$ based on the translated question $Q = \langle \mu_Q, \mathit{args} \rangle$ and all additional required data, for example the planning task τ and the current iteration step δ_i. We do not place any restrictions on the exact format or language of an explanation $E \in \mathcal{E}(Q)$, and will simply refer to the language as $\mathcal{L}_\mathcal{E}$. However, we assume that each explanation $E \in \mathcal{E}(Q)$ is sufficient in itself to answer the question. $\mathcal{E}(Q)$ is regarded as a selection of possible explanations.

These explanations must be communicated to the user, and thus we again need a translator function.

Definition 3.6 (Explanation Translator)**.** An **explanation translator** is a function $T_\mathcal{E} : \wp(\mathcal{L}(\mathcal{E})) \mapsto \mathit{NL}$, that maps a set of formal explanations to a natural language explanation.

This is a simple version of an explanation translator that does not offer a user or context-dependent translation. User-dependent translation is important in order to customize the vocabulary to the user and to provide an answer with the expected level of detail. Incorporating the interaction context can have several benefits, from the possibility to ask follow-up questions, which naturally increases interactivity, to the

inclusion of previous interactions into the selection and summarization of the explanations.

Selecting Explanations The explanation framework provides a selection of explanations, each of which answers the question Q on its own. However, some causes or properties may not be as relevant as others or may not fit into the interaction context. In line with the insight from social science [Miller, 2019] that humans select small relevant explanations given the context, it can therefore be advantageous to use only a subset of explanations.

Often the size of the explanations is chosen as selection criteria [Chakraborti *et al.*, 2017]. [Junker, 2004] selects *preferred* explanations based on an upstream or interleaved process to collect a preference ranking over the goals.

Summarizing Explanations It may be preferable to convey multiple explanations $\hat{\mathcal{E}} \subseteq \mathcal{E}(Q)$, in cases where the effect has multiple sufficient causes neither of which is necessary. But since explanations may be at different levels and expressing their logical relation results in an overly lengthy response, it is often simpler to provide a summary of the explanations $\hat{\mathcal{E}}$. In line with how humans give explanations [Miller, 2019], such a summary should convey only relevant information as effectively as possible in the given context. As for the explanation selection, the user questions can provide an indication of the level of detail expected. Also, the interaction context can be used to enable or facilitate the summarization by referring to previously addressed questions or explanations.

Selecting the correct abstraction level for the explanations [Sreedharan *et al.*, 2019], or only communicating via a predefined vocabulary [Vasileiou and Yeoh, 2023] also leads to a summarization in the sense of leaving out details that are not required or known.

4 Goal Conflict Explanations

In the following, we extend the explanation framework by [Eifler *et al.*, 2020a; Eifler *et al.*, 2020b] to address a larger number of question types, allowing a richer user interaction. The questions types have been collected during an interview conducted at a manufacturing company[1] with the end users of a future explainable planning system. We call this extension *Explanation Framework with Conflicts and Corrections* (EF_{CC}), in line with the fact that all answers are based on either the minimal conflicts (MUS) or minimal corrections (MCS) of the reference goals G^{ref} and the arguments of the question. In the following definitions we assume the user asks questions only about goals already included in G^{ref}. If this is not the case then they can be simply added to G^{ref} before calling the MUS or MCS computation. We list all supported question types and the corresponding formal explanations. To clarify the meaning of the question and answer we include one possible natural language version of both. As an input we require the task τ, the iteration step $\delta = \langle G^{\text{ref}}, G^{\text{enf}}, \pi \rangle$ and the question $Q = \langle \mu_Q, \mathit{args} \rangle$. The produced explanations $\mathcal{E}(Q)$ are subsets of G^{ref}, which means the formal language of an explanation is $\mathcal{L}_{\mathcal{E}} = \wp(\mathcal{L}(\tau))$.

[1]The company is not named to preserve submission anonymity.

If $\tau(G^{\text{enf}})$ is unsolvable we support the following two question types:

`US-why`: "Why is the task unsolvable?":

$$\mathcal{E}(\langle \texttt{US-why}, \emptyset \rangle) = \mathcal{G}^{\text{MUS}}(\tau, G^{\text{enf}})$$

- "The task is unsolvable because it is not possible to satisfy any of the conflicts in $\mathcal{E}(\langle \texttt{US-why}, \emptyset \rangle)$."

`US-how`: "How can I make the task solvable?"

$$\mathcal{E}(\langle \texttt{US-how}, \emptyset \rangle) = \mathcal{G}^{\text{MCS}}(\tau, G^{\text{enf}})$$

- "To make the task solvable you have to forego one of the goal sets in $\mathcal{E}(\langle \texttt{US-how}, \emptyset \rangle)$."

If $\tau(G^{\text{enf}})$ is solvable, we support question types referring to goals not satisfied by the sample plan π. By $G^{\text{true}}(\pi) = \{\phi \in G^{\text{ref}} \mid \sigma(\pi, I) \models \phi\}$ we denote the goals satisfied by π, and by $G^{\text{false}}(\pi) = G^{\text{ref}} \setminus G^{\text{true}}(\pi)$ the goals not satisfied by π. For all the following question types, the arguments must not be satisfied by the current plan, i.e. $\mathit{args} \subseteq G^{\text{false}}(\pi)$.

Answers to question types $\{\texttt{S-why-not}, \texttt{S-what-if}, \texttt{S-can}\}$ are based on the same information, but are phrased differently.

$$\mathcal{E}(\langle \mu_Q, \mathit{args} \rangle) = \min_{\subseteq}(\{C \setminus \mathit{args} \mid C \in \mathcal{G}^{\text{MUS}}(\tau, G^{\text{ref}}), C \subseteq G^{\text{true}}(\pi) \cup \mathit{args}\})$$

where $\min_{\subseteq}(S)$ filters the sets in S that are subset-minimal.

`S-why-not`: "Why are *args* not satisfied?"

- $\mathcal{E}(\langle \texttt{S-why-not}, \mathit{args} \rangle) = \emptyset$: "*args* can be satisfied without foregoing any of the already satisfied goals."
- $\emptyset \in \mathcal{E}(\langle \texttt{S-why-not}, \mathit{args} \rangle)$: "The goals in *args* cannot be satisfied together."
- otherwise: "There is a conflict between *args* and all the goal subsets in $\mathcal{E}(\langle \texttt{S-why-not}, \mathit{args} \rangle)$."

`S-what-if`: "What happens if we enforce *args*?"

- $\mathcal{E}(\langle \texttt{S-what-if}, \mathit{args} \rangle) = \emptyset$: "*args* can be satisfied without foregoing any goal satisfied by the plan."
- $\emptyset \in \mathcal{E}(\langle \texttt{S-what-if}, \mathit{args} \rangle)$: "Then the problem would be unsolvable."
- otherwise: "You could no longer satisfy any of the goal sets in $\mathcal{E}(\langle \texttt{S-what-if}, \mathit{args} \rangle)$."

In the yes/no question type `S-can`, the intention that none of the currently satisfied goals should be given up is implicit. This interpretation is based on the sample responses of the human planners who did not consider the option of foregoing any satisfied goals in $G^{\text{true}}(\pi)$.

`S-can`: "Can *args* be satisfied?"

- $\mathcal{E}(\langle \texttt{S-can}, \mathit{args} \rangle) = \emptyset$: "*args* can be satisfied."
- otherwise: "It is not possible."

`S-how`: "How can *args* be satisfied?"

$$\mathcal{E}(\langle \texttt{S-how}, \mathit{args} \rangle) = \min_{\subseteq}(\{C \setminus G^{\text{false}}(\pi) \mid C \in \mathcal{G}^{\text{MCS}}(\tau, G^{\text{ref}}), C \cap \mathit{args} = \emptyset\})$$

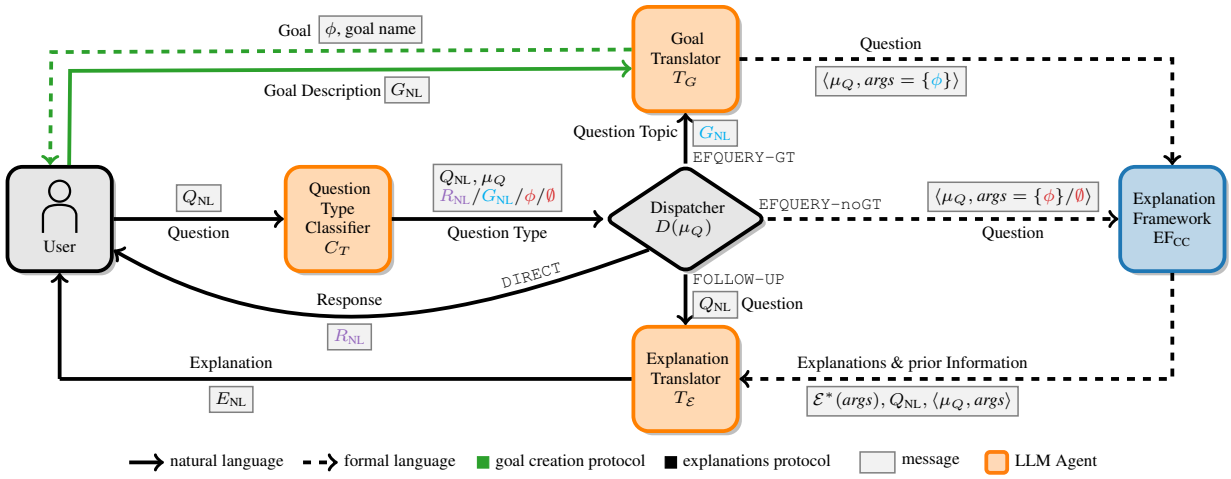

Figure 2: Communication protocol between the user, the translators and the explanation framework. In addition to the information provided in the messages, each agent has access to the planning task and the iteration step. For a goal translation (green, top left) the user directly communicates with T_G. If the user asks a question (black) the dispatcher chooses one of four routing options (DIRECT, FOLLOW-UP, EFQUERY-noGT and EFQUERY-GT) depending on the question type identified by C_T.

- $\emptyset \in \mathcal{E}(\langle \texttt{S-how}, args \rangle)$: "*args* can be satisfied without foregoing any goals satisfied by the plan.
- $\mathcal{E}(\langle \texttt{S-how}, args \rangle) = \emptyset$: "It is not possible."
- otherwise: "You have to forego one of the goal sets in $\mathcal{E}(\langle \texttt{S-how}, args \rangle)$."

The number of MUS and MCS can be exponential in the number of goals, but for all question types M_Q one goal subset $E \in \mathcal{E}(Q)$ is sufficient to answer the question. However, some conflicts may be more relevant, or it may be easier to forego some corrections. Therefore, a selection of $\hat{\mathcal{E}} \subseteq \mathcal{E}(Q)$ and a summarization of the explanations $\hat{\mathcal{E}}$ is desirable.

5 Multi-Agent LLMs

Next, we describe the implementation of our interactive framework using LLM based translators to communicate with EF_{CC}. Each translator, as well as the planner and EF_{CC} (which we consider black boxes that generate plans and explanations when given a correct input message), are *agents* that communicate with each other. We first present the communication protocol between these agents, and then discuss the specifics of each translator agent.

5.1 Communication Protocol

Figure 2 shows the communication protocol between the user, translators and the explanation framework. To translate a natural language goal description G_{NL} into an LTL_f expression ϕ, the user directly communicates with the goal translator T_G.

When a user asks a question Q_{NL}, different routes are possible. The *Question Translator* T_Q is composed of two agents: the *Question Type Classifier* C_T and the *Goal Translator* T_G. A *Dispatcher* routes the response of C_T depending on the identified question type μ_Q.

$\mu_Q =$ DIRECT: If C_T is capable of answering the question, then the answer is given directly to the user. This includes for example questions like "Which questions can I ask?" but also incomprehensible messages or unsupported questions. More examples are given in the technical appendix.

$\mu_Q =$ FOLLOW-UP: If Q_{NL} is identified as a follow-up question, not requiring new input from EF_{CC}, then Q_{NL} is directly sent to the *Explanation Translator* $T_\mathcal{E}$.

$\mu_Q \in M_Q$: If Q_{NL} is classified as one of the question types in M_Q then the question topic is forwarded to T_G and the combined result is sent to the explanation framework (EFQUERY-GT). If the question type does not require an argument or when the question topic is about a goal already used in a previous iteration step (i.e, in $G^{\text{ref}}_{all} = \bigcup G^{\text{ref}}_i$) T_G is bypassed and the (already known) LTL_f formula corresponding to the goal is used (EFQUERY-noGT). The resulting formal explanation is then translated by $T_\mathcal{E}$.

5.2 Translator Implementations

All agents are instantiated from the same base model *GPT-4o-mini*. Their contexts, the input and output histories, are maintained separately from each other. Each LLM agent receives input in the form of a structured string consisting of the prompts described in the technical appendix and the input components listed in Figure 2.

Next we address the tasks of the individual agents and the specific details of their implementation.

Question Type Classifier The *Question Type Classifier* C_T is the first step in question processing. The tasks of this agent extend beyond the identification of the question type μ_Q. In case of DIRECT it directly generates the response R_{NL}. For EFQUERY-GT and EFQUERY-noGT, it extracts a goal description G_{NL} and identifies if G_{NL} matches with a goal description $G'_{NL} \in G^{\text{ref}}_{all}$. It can also identify FOLLOW-UP questions to route them directly to $T_\mathcal{E}$.

Goal Translator The goal translator T_G produces both the LTL_f formula and a natural language description of the goal (used in UI and communication between LLM agents).

The task of translating natural language to LTL or LTL_f has been explored in different fields with different approaches [Brunello *et al.*, 2019]. More recently, approaches based on LLMs that use prompting have been used to implement tools such as `NL2LTL` [Fuggitti and Chakraborti, 2023] a template-based classifier, `Lang2LTL` [Liu *et al.*, 2022] which works without templates and only provides the available literals to the LLM, and `nl2spec` [Cosler *et al.*, 2023] which addresses sub-formulas iteratively to counteract ambiguities.

Our evaluation focuses on the question classification and the explanation translation. Thus, we opted for simple base implementation, similar to what [Liu *et al.*, 2022] call *End-to-End Approach*. The LLM is provided with the predicates and objects that can compose the literals and a few samples.

Explanation Translator An advantage of an LLM-based translator is its ability to provide user-dependent translation, select explanations, and summarize them. Instead of just providing explanations, we provide both conflicts and corrections $\mathcal{E}^*(Q)$. This is motivated by the anticipation of follow-up questions (e.g. a "how" question after a "why" question). It additionally receives the user's question Q_{NL}, and the formal translation Q. μ_Q indicates how to interpret $\mathcal{E}^*(Q)$, while Q_{NL} helps to generate a natural response. We expect the explanation translator to leverage prompt examples to understand how to select or summarize explanations. It is thus a domain-specific design choice to provide examples of the expected selection/summarization strategy. With `FOLLOW-UP` questions, users can actively request a summary or selection. We leave the evaluation of this feature and the extraction of user preferences from selection requests as future work.

Context-Dependent Translators
A translator solely based on the inputs specified in Figure 2, can only provide context-independent translations and is not capable of addressing follow-up question. To enable context-dependent translators we leverage the context memory of LLMs: they retain their context across interactions, so they can build on and reference previous interactions. This inherently provides all the benefits listed in Section 3. While LLM contexts can theoretically retain every previous iterations with a user, we implemented separate contexts for each iteration step. This allows to maintain clarity by avoiding mixing information from different iteration steps. Examples of context-dependant interactions, selections or summarization of explanations are given in the technical appendix.

6 Evaluation

We developed a web-based platform extending [Eifler *et al.*, 2022] with our LLM-based framework and an improved UI. The source code will be published upon acceptance. We then used this tool to conduct a pilot user study aiming at evaluating the effectiveness of some of our LLM agents and the impact of our LLM-based explanation interface on users' ability to solve a planning task. The material presented in this section is complemented in appendix.

6.1 User Study Design

Similarly to [Eifler *et al.*, 2022], we used a parent's afternoon planning task as the scenario. The participants were tasked with the planning of afternoon activities for a family, not all of which can be carried out due to time constraints. As a proxy for user preference, each goal has an associated utility. The objective of the participants was to select a subset of the goals to maximize the total utility of the plan. The goal utility is unknown to all other agents (planner, explainer and LLMs). Consequently, these agents are capable of providing information regarding the solvability of a set of goals, and the conflicts and corrections, but are unable to provide direct assistance in maximizing utility.

The study was divided into three parts. First, participants were introduced to the tool through an introduction task (6 goals). Second, a more complex task (15 goals with 35 conflicts and 19 corrections) was used to compare the effect of the different interfaces used by G^{TPL} and G^{LLM} on maximizing the utility within 20 min. Finally, participants completed a post-experiment questionnaire.

In order to evaluate the effectiveness of LLM agents in an interactive explanation framework, we divided the 70 participants into two groups. Our control group G^{TPL} included 34 participants utilizing a **template-based** interface, which allowed them to select from the predefined set of questions introduced in Section 4 and receive answers based on simple natural language templates. The second group G^{LLM} included 36 participants given the **LLM-based** interface, with which they asked freely formulated natural language questions that were processed by LLM-based translators as described in Section 5. We did not include a group without explanations as an earlier study [Eifler *et al.*, 2022] had already examined that baseline and demonstrated that our control condition was more effective.

6.2 Evaluation of LLM Agents

We first evaluated the accuracy of the question type classifier C_T on the complete set of 134 questions collected from G^{LLM}. The classification of both the question type and the question argument was correct in 89.47% of cases. All observed failures were related to routing decisions, where the question translator T_Q unnecessarily routed questions to EF_{CC} by predicting $\mu_Q \in M_Q$ instead of responding directly (`DIRECT`) or forwarding them to the explanation translator $T_\mathcal{E}$ (`FOLLOW-UP`). Importantly, these classification errors resulted in suboptimal routing – with no delay since the explanations were already computed – but $T_\mathcal{E}$ was still able to provide appropriate responses to the users' questions.

To ensure a controlled user study environment with reasonable latency for explanations, we decided to provide a fixed set of reference goals. This results in all users having the same optimization task and allows to precompute all explanations. This means that users did not create new goals, which leaves the evaluation of the goal translation for future work. As for $T_\mathcal{E}$, the main challenge is to measure the correctness of a summarized explanation which is out of scope of this paper.

6.3 User Groups Comparison

We compared the two groups using the questionnaire and a number of metrics, including the time spent on the task, the maximum utility reached, and the number of questions asked.

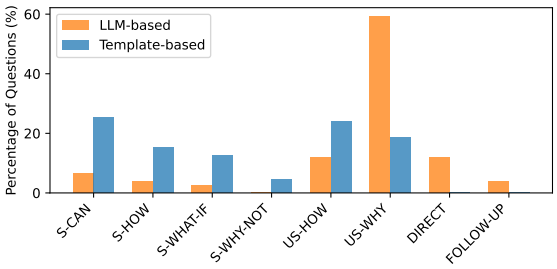

Figure 3: Comparison of question types used by $\mathrm{G}^{\mathrm{LLM}}$ and $\mathrm{G}^{\mathrm{TPL}}$.

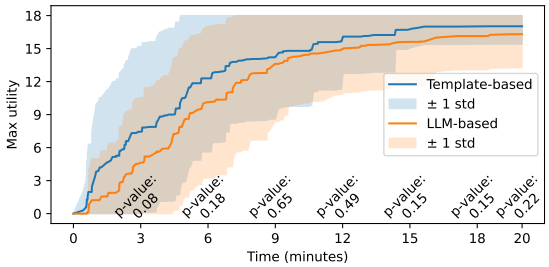

Figure 4: Comparison of maximum utility achieved over time between $\mathrm{G}^{\mathrm{LLM}}$ and $\mathrm{G}^{\mathrm{TPL}}$. Group means are computed at each time step.

Users took on average 12.0 mins $\pm$ 7.0 in $\mathrm{G}^{\mathrm{LLM}}$ and 12.3 mins $\pm$ 6.3 in $\mathrm{G}^{\mathrm{TPL}}$ to complete the evaluation task. 22/36 (61.1%) users in $\mathrm{G}^{\mathrm{LLM}}$ and 23/34 (67.6%) in $\mathrm{G}^{\mathrm{TPL}}$ reached the maximum utility of 18.

$\mathrm{G}^{\mathrm{TPL}}$ asked on average significantly more questions (10.5 $\pm$ 13.3) vs. (2.5 $\pm$ 2.6) for $\mathrm{G}^{\mathrm{LLM}}$ ($p < 0.001$, t-test). $\mathrm{G}^{\mathrm{TPL}}$ also asked more diverse questions, as shown in Figure 3. Although $\mathrm{G}^{\mathrm{LLM}}$ had complete freedom in the questions they could ask, they mainly asked `US-why` questions. The analysis of participants' feedback on their strategies also shows more diversity in the $\mathrm{G}^{\mathrm{TPL}}$ group. These points suggest that the $\mathrm{G}^{\mathrm{LLM}}$ users did not take full advantage of the conversational capabilities of the LLM interface.

When comparing the evolution of the plan utility over time shown in Figure 4, we observe that $\mathrm{G}^{\mathrm{TPL}}$ has a higher mean value than $\mathrm{G}^{\mathrm{LLM}}$ for the whole duration of the task. Nonetheless, the difference is not statistically significant. The utility over iteration steps lead to similar conclusions, and shows a convergence significantly quicker toward the maximum utility for $\mathrm{G}^{\mathrm{TPL}}$. The questionnaire results only shows significant difference on the question *"Did questions help to improve a plan?"* which received higher scores from $\mathrm{G}^{\mathrm{TPL}}$. This corroborates the fact that $\mathrm{G}^{\mathrm{LLM}}$ users mostly used the explanation interface during unsolvable iteration steps.

6.4 Limitations and Discussion

Instance and Task While this user study provides some evidence of better performance of $\mathrm{G}^{\mathrm{TPL}}$ over $\mathrm{G}^{\mathrm{LLM}}$, it is important to note that these results were obtained on a single planning task instance of moderate difficulty. Although this setup was chosen to align with a previous user study in which template-based goal-conflict explanations were found useful [Eifler *et al.*, 2022], it remains to investigate how the proposed architecture would help on a task for which template-based approaches show limitations — particularly problems where summarization is a *necessity*. Perhaps a more extensive training session, highlighting the LLM's capabilities, could have helped users to take fuller advantage of the interface. Another limitation of our study is that it was conducted with lay users. Whether similar results hold for the target users of such tools (domain experts) remains open. One can hypothesize that domain experts might ask more diverse questions than lay persons.

Using LLMs has a (time and effort) cost The free-text feedback obtained through the questionnaire provides insights into why $\mathrm{G}^{\mathrm{LLM}}$ performed slightly worse than $\mathrm{G}^{\mathrm{TPL}}$. Common concerns included system latency (response times averaged $\sim$5 secs, occasionally exceeding 10 secs) and the additional effort of formulating and typing questions. In particular, $\mathrm{G}^{\mathrm{LLM}}$ needed a much deeper understanding of which questions to ask, whereas $\mathrm{G}^{\mathrm{TPL}}$ readily had access to the relevant questions. These factors led $\mathrm{G}^{\mathrm{LLM}}$ users to ask fewer questions and explore less thoroughly the range of possible interactions. Additionally, receiving an unconvincing response early in the task could diminish trust in the system – an effect that particularly impacts $\mathrm{G}^{\mathrm{LLM}}$ users since the explanation interface requires more effort and patience.

7 Conclusion and Future Work

Contributions In this paper we presented a high-level architecture for interactive explanations in the context of iterative planning. We instantiated this framework with goal-conflict explanations and provided an easy-to-use implementation via a web platform. Finally, we conducted a user study to evaluate the effect of the LLM-based explanation interface over a more conventional template-based interface.

Findings We found that there is some potential for LLM-based translators but they did not show objective improvements in helping lay users to solve a moderately difficult planning task. Users of the LLM interface asked fewer and less diverse questions than the users of the template-based interface. While this could be due to the additional effort and patience required to use the LLM interface, it also indicates that users of the template-based interface had a better understanding of the capabilities of the system. The fact that we did not observe this with expert users that tested this tool and successfully utilized the capabilities of the LLM suggests that there is a learning curve that some user study participants did not overcome in time, and that LLM-based interfaces could be helpful in other explainable planning contexts.

Future Work Future work should evaluate LLM-translated explanations on tasks on which template-based translations show limitations. Summarization capabilities of LLMs in this context should be assessed carefully.

We also plan on investigating the extension of this approach to other explanations frameworks. This includes MUS and MCS based approaches used in Constraint Programming [Povéda *et al.*, 2024; Gamba *et al.*, 2023] but also other approaches used in planning [Krarup *et al.*, 2021; Sreedharan *et al.*, 2021]. In addition to the classification of the question type, this adds the challenge of deciding which explanation approach is most suitable to answer the question.

Ethical Statement

We confirm that our user study was conducted under full compliance with the requirements of our local ethics committee [name withheld for anonymity]. All participants were volunteers who gave informed consent, and the study was designed more like a playful, game-like engagement than a constraining activity, ensuring no potential harm or emotional discomfort. Furthermore, no sensitive elements were introduced that could adversely affect the participants' well-being.

While large language models (LLMs) in our framework may occasionally generate misleading or irrelevant statements (i.e., "hallucinations"), the core planning tasks are handled by a dedicated planner, eliminating the risk of producing incorrect plans. At worst, iterative planning might be rendered less effective if LLM responses deviate from the actual planning context. Finally, our emphasis on explainability furthers transparency, aligning our work with the broader objective of making AI-driven systems more interpretable and trustworthy.

References

[Achinstein, 1980] Peter Achinstein. *The Nature of Explanation*. Oxford University Press, 1980.

[Baier and McIlraith, 2006] Jorge A. Baier and Sheila A. McIlraith. Planning with first-order temporally extended goals using heuristic search. In *Proc. AAAI*, 2006.

[Bommasani *et al.*, 2021] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[Bromberger, 1962] Sylvain Bromberger. An approach to explanation. In R. Butler, editor, *Analytical Philsophy*. Oxford University Press, 1962.

[Brown *et al.*, 2020] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Proc. NeurIPS*, 2020.

[Brunello *et al.*, 2019] Andrea Brunello, Angelo Montanari, and Mark Reynolds. Synthesis of ltl formulas from natural language texts: State of the art and research directions. In *Proc. TIME*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[Chakraborti *et al.*, 2017] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *IJCAI*, 2017.

[Cosler *et al.*, 2023] Matthias Cosler, Christopher Hahn, Daniel Mendoza, Frederik Schmitt, and Caroline Trippel. nl2spec: interactively translating unstructured natural language to temporal logics with large language models. In *Proc. CAV*, 2023.

[Dazeley *et al.*, 2021] Richard Dazeley, Peter Vamplew, Cameron Foale, Charlotte Young, Sunil Aryal, and Francisco Cruz. Levels of explainable artificial intelligence for human-aligned conversational explanations. *AIJ*, 2021.

[De Giacomo *et al.*, 2014] Giuseppe De Giacomo, Riccardo De Masellis, and Marco Montali. Reasoning on LTL on finite traces: Insensitivity to infiniteness. In *AAAI*, 2014.

[Domshlak and Mirkis, 2015] Carmel Domshlak and Vitaly Mirkis. Deterministic oversubscription planning as heuristic search: Abstractions and reformulations. *JAIR*, 2015.

[Edelkamp, 2006] Stefan Edelkamp. On the compilation of plan constraints and preferences. In *ICAPS*, 2006.

[Eifler *et al.*, 2020a] Rebecca Eifler, Michael Cashmore, Jörg Hoffmann, Daniele Magazzeni, and Marcel Steinmetz. A new approach to plan-space explanation: Analyzing plan-property dependencies in oversubscription planning. In *AAAI*, 2020.

[Eifler *et al.*, 2020b] Rebecca Eifler, Marcel Steinmetz, Alvaro Torralba, and Jörg Hoffmann. Plan-space explanation via plan-property dependencies: Faster algorithms & more powerful properties. In *IJCAI*, 2020.

[Eifler *et al.*, 2022] Rebecca Eifler, Martim Brandao, Amanda Coles, Jeremy Frank, and Jörg Hoffmann. Evaluating plan-property dependencies: A web-based platform and user study. In *ICAPS*, 2022.

[Feldhus *et al.*, 2023] Nils Feldhus, Qianli Wang, Tatiana Anikina, Sahil Chopra, Cennet Oguz, and Sebastian Möller. Interrolang: Exploring nlp models and datasets through dialogue-based explanations. In *Proc. EMNLP*, 2023.

[Fuggitti and Chakraborti, 2023] Francesco Fuggitti and Tathagata Chakraborti. NL2LTL – a python package for converting natural language (NL) instructions to linear temporal logic (LTL) formulas. In *ICAPS*, 2023.

[Gamba *et al.*, 2023] Emilio Gamba, Bart Bogaerts, and Tias Guns. Efficiently explaining csps with unsatisfiable subset optimization. *JAIR*, 2023.

[Guo *et al.*, 2024] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024.

[Junker, 2004] Ulrich Junker. Preferred explanations and relaxations for over-constrained problems. In *Proc. AAAI*, 2004.

[Kambhampati *et al.*, 2024] Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *ICML*, 2024.

[Krarup *et al.*, 2021] Benjamin Krarup, Senka Krivic, Daniele Magazzeni, Derek Long, Michael Cashmore, and David E. Smith. Contrastive explanations of plans through model restrictions. *JAIR*, 2021.

[Krarup *et al.*, 2024] Benjamin Krarup, Amanda Jane Coles, Derek Long, and David E. Smith. Explaining plan quality differences. In *ICAPS*, 2024.

[Lakkaraju *et al.*, 2022] Himabindu Lakkaraju, Dylan Slack, Yuxin Chen, Chenhao Tan, and Sameer Singh. Rethinking explainability as a dialogue: A practitioner's perspective. In *NeurIPS Workshop on Human Centered AI*, 2022.

[Liao *et al.*, 2020] Q Vera Liao, Daniel Gruen, and Sarah Miller. Questioning the ai: informing design practices for explainable ai user experiences. In *Proc. CHI*, 2020.

[Liu *et al.*, 2022] Jason Xinyu Liu, Ziyi Yang, Benjamin Schornstein, Sam Liang, Ifrah Idrees, Stefanie Tellex, and Ankit Shah. Lang2ltl: Translating natural language commands to temporal specification with large language models. In *Workshop on Language and Robotics at CoRL*, 2022.

[Miller, 2019] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *AI*, 2019.

[Nguyen *et al.*, 2023] Van Bach Nguyen, Jörg Schlötterer, and Christin Seifert. From black boxes to conversations: Incorporating xai in a conversational agent. In *XI*, 2023.

[Peng *et al.*, 2024] Binghui Peng, Srini Narayanan, and Christos Papadimitriou. On limitations of the transformer architecture. *arXiv preprint arXiv:2402.08164*, 2024.

[Povéda *et al.*, 2024] Guillaume Povéda, Andreas Strahl, Mark Hall, Ryma Boumazouza, Santiago Quintana-Amate, Nahum Alvarez, Ignace Bleukx, Dimos Tsouros, Hélène Verhaeghe, and Tias Guns. Trustworthy and explainable decision-making for workforce allocation. In *Workshop on Progress Towards the Holy Grail at CP*, 2024.

[Shen *et al.*, 2023] Hua Shen, Chieh-Yang Huang, Tongshuang Wu, and Ting-Hao Kenneth Huang. Convxai: Delivering heterogeneous AI explanations via conversations to support human-ai scientific writing. *CoRR*, 2023.

[Slack *et al.*, 2023] Dylan Slack, Satyapriya Krishna, Himabindu Lakkaraju, and Sameer Singh. Explaining machine learning models with interactive natural language conversations using talktomodel. *Nature Machine Intelligence*, 2023.

[Smith, 2004] David E. Smith. Choosing objectives in over-subscription planning. In *ICAPS*, 2004.

[Smith, 2012] David Smith. Planning as an iterative process. In *AAAI*, 2012.

[Sreedharan *et al.*, 2019] Sarath Sreedharan, Siddharth Srivastava, David E. Smith, and Subbarao Kambhampati. Why can't you do that hal? explaining unsolvability of planning tasks. In *IJCAI*, 2019.

[Sreedharan *et al.*, 2021] Sarath Sreedharan, Tathagata Chakraborti, and Subbarao Kambhampati. Foundations of explanations as model reconciliation. *AIJ*, 2021.

[State *et al.*, 2023] Laura State, Salvatore Ruggieri, and Franco Turini. Reason to explain: Interactive contrastive explanations (reasonx). In *XAI*, 2023.

[Touvron *et al.*, 2023] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[Valmeekam *et al.*, 2023] Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models-a critical investigation. *NeurIPS*, 2023.

[Vasileiou and Yeoh, 2023] Stylianos Loukas Vasileiou and William Yeoh. PLEASE: generating personalized explanations in human-aware planning. In *Proc. ECAI*, 2023.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[Zhang *et al.*, 2024] Tong Zhang, X Jessie Yang, and Boyang Li. May i ask a follow-up question? understanding the benefits of conversations in neural network explainability. *International Journal of Human–Computer Interaction*, 2024.