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

Guilhem Fouilhé[1], **Rebecca Eifler**[2], **Sylvie Thiébaux**[2,4], **Nicholas Asher**[1,3]

[1]IRIT [2]LAAS-CNRS [3]CNRS, Toulouse, France
[4]Australian National University, Canberra, Australia
guilhem.fouilhe-lafforgue@irit.fr, rebecca.eifler@laas.fr, sylvie.thiebaux@laas.fr, nicholas.asher@irit.fr

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 the tedious work. In an iterative process, the human's role is to guide the planner according to their preferences and expert experience. Explanations that respond to users' questions are crucial to increase trust in the system and improve understanding of the sample solutions. To enable natural interaction with such a system, we present an explanation framework agnostic architecture for interactive natural language explanations that enables user and context dependent interactions. We propose conversational interfaces based on Large Language Models (LLMs) and instantiate the explanation framework with goal-conflict explanations. As a basis for future evaluation, we provide a tool for domain experts that implements our interactive natural language explanation architecture.

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 satisfactory plan from the point of view of all involved agents is found. From this perspective, explanations serve the purpose of accelerating the convergence of preferences elicitation by human agents. Even though interactive planning has now become a relatively well-accepted paradigm, interactive explanations are a somewhat lost second cousin. Nevertheless, 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 remedy or remove 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.

If explanations are ideally interactive, then to provide appropriate explanations for a planning system, 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. LLMs have conversational capacities that will enable truly interactive explanations that go along with interactive planning. 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 (see Section 3). Nevertheless, they can serve valuable supporting roles (Kambhampati et al. 2024). Our work extends this view showing how LLMs can be effectively integrated into

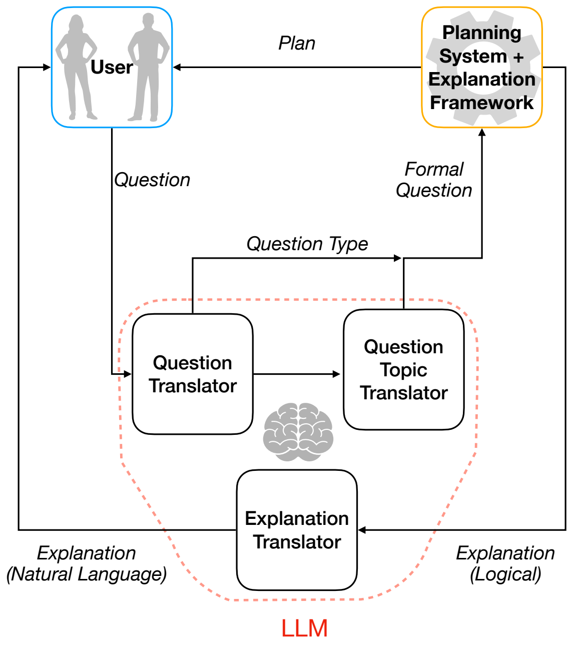


Figure 1: Architecture of our approach to interactive explanations with LLMs. This figure describes a single question-answer interaction in a single planning iteration step.

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 fully leverage these conversational capabilities of LLMs in planning systems, we need to understand the different modes of interaction they can support. We distinguish between two types of interactivity: *across-step interactivity* to refer to the inherent interactivity of explanations in iterative planning systems (see Section 4), and *intra-step interactivity* to refer to the interactivity of explanations in a single planning iteration step. LLMs, with their conversational capacities, enable *intra-step interactivity* as well as naturally improving over the course of an iterative planning session, enhancing *across-step interactivity*.

Our main contributions are the following:

- we formalize a general framework for iterative planning with explanations (Section 4),
- we instantiate this framework with goal-conflict based explanations with LLM interfaces (Section 6),
- we present an evaluation tool of this framework that will be used during a follow-up user study (Section 7).

2 Our Vision

To enable natural interactions between users and planning systems, we need to accommodate different ways in which a user might formulate their question in natural language or express the problem that moved them to ask for an explanation. An LLM excels at handling such varied natural language inputs. On the other hand, given that LLMs cannot reliably plan, we need to access a formal planner that requires a logically explicit input in a fixed format, which we cannot expect a lay person to provide. To accommodate both of these desiderata, we need a method for translating between varied natural language inputs and formal queries to the planning system. Our proposal is to introduce a new framework for interactive planning with explanations: a multi-agent architecture that uses LLMs as specialized translators. These translators mediate between natural language conversations and the formal representations required by the planning system. The use of prompt-based techniques makes it possible to adapt to new domains with minimal effort, and our framework is agnostic with respect to particular planning and explanation frameworks. We illustrate our approach in the following sections with oversubscription planning with a logistics use case and simple deductively valid explanations.

Our iterative explanation framework (Figure 1), described in detail in Section 4, leverages these LLMs in three distinct roles: first as a translator of questions that the user may have for the system; second as a translator of questions' topic for the explanation framework; third as an explanation translator. As we mentioned in the introduction section, a pragmatically suitable explanation should resolve a conceptual problem of the explainee, which is typically linguistically introduced by certain kinds of questions. These questions will determine the form of the explanation; answers to *why* questions differ from answers to a *what-if* question; e.g., *"Why can't I deliver package P_1?"* versus *"What if the truck has 20 liters more fuel?"*. 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 deliver package P_1?"* has a different answer from *"Why can't I visit the coffee shop?"*. The question and question-topic translators translate a natural language question into a formal specification to which the planner can respond with a formal explanation. The explanation translator takes the formal explanation of the planner back into what linguists call a natural language dialogue move or sequence of moves in the conversation with the user. We say *sequence* of dialogue moves, because in an interactive system the explanation may extend over several clauses, even over several different interactions with the user.

Having flexible translators on top of a formal explanation framework not only allows a more natural interface for an interaction that could still exist differently, e.g. using a template-based translation. It is also motivated by new features like explanation selection or summarization, which has been shown to be central in humans explanations (Miller 2019).

While the translation features of the proposed framework within each step of the iterative planning process are our main topic here, in-context learning capabilities of LLMs should improve interactions between users and the formal planner over the planning iterations by providing a much more natural conversational interface.

3 Background

3.1 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 prompting, which uses carefully crafted input text to guide the model's behavior. The latter approach, called in-context learning (Brown et al. 2020), 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.

While LLMs excel at natural language understanding and generation, they face limitations in tasks requiring precise logical reasoning (Bhar and Asher 2024; Asher et al. 2023; Mirzadeh et al. 2024; Kambhampati 2024). In addition, using communication complexity (Kushilevitz 1997), (Sanford, Hsu, and Telgarsky 2024; Peng, Narayanan, and Papadimitriou 2024) analyze the computational powers of the attention calculation and show that there are severe limitations in their ability to compose functions. This is directly relevant for deterministic planning, according to which actions are understood as functions that take an input state and output a transformed one. Such results make LLMs better suited for tasks involving natural language understanding, generation and translation rather than direct problem-solving in domains like automated planning.

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.

3.2 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. 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 $\pi = \langle a_1, \dots, a_n \rangle$ is denoted by $s[\![\pi]\!]$. A **plan** is an action sequence π where $G \subseteq I[\![\pi]\!]$. A task τ is **solvable** if a plan exists. By $\Pi(\tau)$ we denote the set of all plans of 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 $R \subseteq G$ 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,b).

Temporal Goals By using a compilation approach (Edelkamp 2006; Baier and McIlraith 2006; De Giacomo, De Masellis, and Montali 2014) for temporal properties defined in finite linear temporal logic (LTL$_\text{f}$), it is possible to not only reason about the conflicts of goals facts but over properties of plans (Eifler et al. 2020b). In the following we assume that all goals are defined in LTL$_\text{f}$ over atoms. LTL$_\text{f}$ formulas are interpreted over a finite sequence of states (trace) $\sigma = s_0 s_1 \cdots s_n$. We denote $s_i \cdots s_n$ by σ_i. Given a trace σ and a LTL$_\text{f}$ formula ϕ we say $\sigma \models \phi$ iff:

- $\sigma_i \models \texttt{final}$ iff $i = n$
- $\sigma_i \models \texttt{true}$ and $\sigma_i \not\models \texttt{false}$
- $\sigma_i \models \varphi$ where φ is an atom iff $\varphi \in s_i$
- $\sigma_i \models \neg\varphi$ iff $\sigma_i \not\models \varphi$
- $\sigma_i \models \phi \wedge \psi$ iff $\sigma_i \models \phi$ and $\sigma_i \models \psi$
- $\sigma_i \models \bigcirc\phi$ iff $i < n$ and $\sigma_{i+1} \models \phi$
- $\sigma_i \models \phi \,\mathrm{U}\, \psi$ iff $\exists j : i \leq j \leq n : \sigma_i \models \psi$ and $\forall k : i \leq k < j : \sigma_k \models \phi$

In addition, the following standard temporal operators are used: eventually $\Diamond\phi := \mathit{true} \,\mathrm{U}\, \phi$ and always $\Box\phi := \neg\Diamond\neg\phi$. We refer to the set of all well-formed LTL$_\text{f}$ formulas of task τ as $\mathcal{L}(\tau)$. An action sequence π satisfies LTL$_\text{f}$ formula ϕ in state s, iff ϕ holds in 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 $\phi \in G : \sigma(\pi, I) \models \phi$.

4 Iterative Planning with Explanations

Iterative planning is based on the idea formalized by (Smith 2004) that in many real-world scenarios it is not purposeful to generate just one plan. Given conflicting objectives and goals and users who have not yet fully formed their preferences, an iterative exploration of possible plans is more suitable. In this framework explanations are crucial, especially those that help the user to understand the dependencies between the objectives, goals and preferences. Next we define a generic framework for iterative planning with explanations, focusing on the interaction points with the users. In the next section we will provide a concrete implementation of the explanatory framework.

In iterative planning, the process for determining a final plan is divided into individual iteration 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^{enf}, and a sample plan π_i satisfying G_i^{enf}:

Definition 4.1 (Iteration Step)**.** Given a planning task τ, an **iteration step** is a tuple $\delta = \langle G^{\mathrm{ref}}, G^{\mathrm{enf}}, \pi \rangle$, where G^{ref} is a set of (temporal) goals for τ, $G^{\mathrm{enf}} \subseteq G^{\mathrm{ref}}$, and $\pi \in \Pi(\tau(G^{\mathrm{enf}}))$ if $\tau(G^{\mathrm{enf}})$ is solvable and $\pi = \epsilon$ otherwise.

As depicted on the left in Figure 2, based on the sample plan π_i, the user defines the set of reference goals G_{i+1}^{ref} considered in the next step. These reference goals represent the objects, goals and preferences that the user is currently interested in. Finally, the user must select a subset of goals $G_{i+1}^{\mathrm{enf}} \subseteq G_{i+1}^{\mathrm{ref}}$ that should be satisfied by the sample plan π_{i+1} in the next iteration.

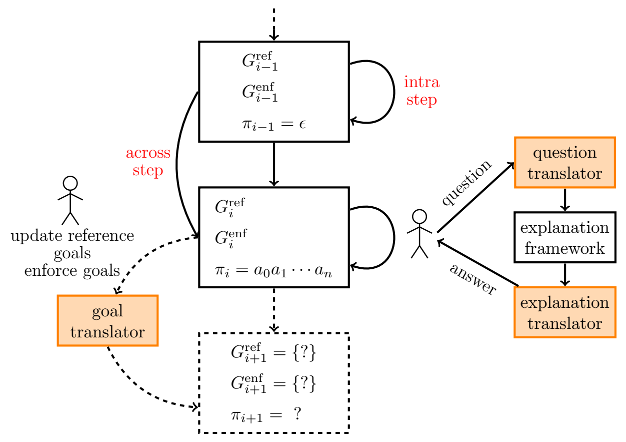


Figure 2: *Intra-step* and *across-steps* user interaction via translators.

The reference goals must be defined in a language the system understands, here we are using LTL$_\mathrm{f}$. However, this language is not suitable for a lay person as an input language. Thus, we require a goal translator.

Definition 4.2 (Goal Translator)**.** Given a task τ with well-formed temporal goals $\mathcal{L}(\tau)$, a **goal translator** is a function $T_G : NL \mapsto \mathcal{L}(\tau) \cup \epsilon$, 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 $\mathcal{L}(\tau)$.

The task of translating natural language to LTL or LTL$_\mathrm{f}$ has been explored in different fields with different approaches (Brunello, Montanari, and Reynolds 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.

A more advanced goal translator could also include a feedback loop with the user to recover from misunderstandings or to notify the user that a very similar property is already part of the reference goals. For such features the translator $T_G^{\mathcal{C}}$ needs to depend on the *interaction context* $\mathcal{C}$ reflecting the previous interaction with the translator:

Definition 4.3 (Interaction Context)**.** The **interaction context** for translator T is a sequence $\mathcal{C}(T) = \iota_0 \iota_1 \cdots \iota_n$ of interactions $\iota = \langle \delta, \mathit{INP}, T(\mathit{INP}) \rangle$, where each interaction ι is associated with an iteration step δ and contains the translator input *INP* and the translation result $T(\mathit{INP})$.

To facilitate the decision of which goals to enforce in the next iteration, explanations can be provided. Those explanations can address different objectives such as clarifications about the model (Sreedharan, Chakraborti, and Kambhampati 2021), identifying the trade-offs between plan quality measures (Krarup et al. 2024) or a better understanding of the dependencies between the goals (Eifler et al. 2020a). Our implementation of such a framework is described in the next section. Explanations are provided as answers to specific user questions. Note that 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} is not satisfiable. Formally we define a question as follows.

Definition 4.4 (Question)**.** Given a task τ and a set of question types M_Q, a **question** is a tuple $\kappa = \langle \mu_Q, \mathit{args} \rangle$ where $\mu_Q \in M_Q$ is the question type and $\mathit{args} \subseteq \mathcal{P}(\mathcal{L}(\tau))$ are the question arguments.

The question types M_Q depend on the explanation framework used. In the Section 5 we introduce the question types our explanation framework supports. Until then an example is the question type `S-why-not`, i. e. "Why is g not satisfied by plan π", which requires one argument g signifying some goal. Roughly, each question word in natural language, e. g. *who, what, why, how* maps to a different type of question, but other types of questions can be formed from these elements and boolean operators – e. g. *"What if p?"*, *"Why p and $\neg$q?"*.

This is the second interaction point with the user and the framework. Thus, we again require a translator to allow the users to ask their question in natural language, which is then translated to a question format which can be processed by the explanation framework:

Definition 4.5 (Question Translator)**.** Given a task τ, and a set of question types M_Q, a **question translator** is a function $T_Q : NL \mapsto (M_Q \times \mathcal{P}(\mathcal{L}(\tau))) \cup \epsilon$, that maps a natural language expression to a question type and its parameters and to ϵ if no matching question type exists.

A context-dependent question translator $T_Q^{\mathcal{C}}$ with access to the previously asked questions enables follow-up questions with implicit references. For example, for the questions "Why don't you deliver package P_0?" or "Can you enforce it?', the delivery location or even the entire question argument depend on the context.

The explanation framework computes a set of formal explanations $\mathcal{E}(\kappa)$ based on the translated question $\kappa = \langle \mu_Q, \mathit{args} \rangle$ and all additional required data, for example the planning task τ and the current iteration step δ_i.

For now, we do not place any restrictions on the exact format or language of an explanation $E \in \mathcal{E}(\kappa)$, and will simply refer to the language as $\mathcal{L}_X$. However, we assume that each explanation $E \in \mathcal{E}(\kappa)$ is sufficient in itself to answer the question. Thus, $\mathcal{E}(\kappa)$ can be regarded as a selection of possible explanations.

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

Definition 4.6 (Explanation Translator)**.** An **explanation translator** is a function $T_{\mathcal{E}} : \mathcal{P}(\mathcal{L}_X) \mapsto NL$, 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. However, the explanation translator is crucial for the usefulness of the explanation and the acceptance and trust of the user. User-dependent translation is important 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 the interactivity, to the inclusion of previous interactions into the selection and summarization of the explanations. Thus, in the following we discuss possible extensions of a context-dependent explanation translator $T^{\mathcal{C}}_{\mathcal{E}}$.

Explanation Vocabulary A desirable feature is to provide natural language explanations that correspond to the vocabulary used by the user. One user might describe the planning problem, goals and frame their questions differently from another user. Vasileiou and Yeoh (2023) achieve such personalized explanations in terms of abstraction level and vocabulary by relying on a predefined user vocabulary. While we cannot address this issue in its full generality, a context dependent translator can take into account the natural language questions and descriptions of the references goals in addition to the previous queries. This provides a reference to the user's vocabulary for the given domain. Since the reference goals G^{ref}_i reflect the properties of the plans in which the user is currently interested, it makes sense to provide explanations based on these properties. This also leads to explanations at an appropriate level of abstraction for the user.

Selecting Explanations The explanation framework can provide a selection of explanations, each of which answers the question κ on its own. However, some causes or properties may not be as relevant as others or may not fit into the interaction context because, for example, they relate to a completely different topic than what was previously discussed. It can therefore be advantageous to use the explanations $\hat{\mathcal{E}} \subseteq \mathcal{E}(\kappa)$ that best match the user's expectations and the interaction flow. These selection criteria are in line with the insight from social science (Miller 2019) that humans select small relevant explanations given the context.

Often simply the size of the explanations is chosen as selection criteria (Chakraborti et al. 2017). The most relevant explanations are referred to as *preferred* explanations by (Junker 2004). Users are asked in an upstream or interleaved process to provide some sort of preference ranking over the properties that will later on be used in selecting or computing explanations. This approach is suitable if the preferences are already known to the user and do not change drastically during the iterative planning process. It would however be better if the explanation translator could extract the users' preferences from the interaction context, to automatically select the best suited explanations.

Summarizing Explanations Instead of selecting a small explanation, it may be preferable to convey multiple explanations offered by the explanation framework. However, this approach can result in an overly lengthy explanation if the natural language translation is too literal and repetitive. A summary of the explanations is therefore desirable in order to convey a common reason more compactly.

Selecting the correct abstraction level for the explanations (Sreedharan et al. 2019) also leads to a summarization in a sense of leaving out not required or known details. The aim of summarizing explanations is in line with how humans provide explanations (Miller 2019): conveying only relevant information as effectively as possible in the given context.

In the following we will consider two options for summarization: lossless and lossy summarization. With lossless summarization, all information is retained, and only an attempt is made to transmit it efficiently. With lossy summarization, information is discarded to result in smaller explanations. This can lead to incomplete or even incorrect explanations that no longer answer the user's question. Our aim is therefore to only discard information that does not affect the correctness.

As for the explanation selection, the user questions can provide an indication of the level of detail expected of the explanation. Generally, a very detailed question calls for a detailed answer. Of course, there are exceptions to this rule, e.g. if there is a very simple reason that the user was not aware of, then the translation should not overcomplicate it. Also, the interaction context can be used to enable or facilitate the summarization by referring to previously addressed questions or explanations.

Although both ultimately lead to a smaller natural language explanation, there is a clear distinction between selection and summarization. Selection attempts to decide which explanations are relevant for the user, possibly based on previous interactions, while summarization is about efficiently conveying a given number of explanations.

5 Goal Conflict Explanations

In the following, we extend the explanation framework by (Eifler et al. 2020a,b) to address a larger number of question types, allowing a richer user interaction. The questions have been collected during a user study conducted at Airbus with the end users of explainable planning systems (Morandini et al. 2024). All answers of the framework EF_{CC} 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 question and answer. As an input we require the task τ, the iteration step $\delta = \langle G^{\text{ref}}, G^{\text{enf}}, \pi \rangle$ and the question $\kappa = \langle \mu_Q, \mathit{args} \rangle$ with question type μ_Q and arguments *args*. The produced explanations $\mathcal{E}(\kappa)$ are all subsets of G^{ref}, which means the formal language of an explanation is $\mathcal{L}_X = \mathcal{P}(\mathcal{L}(\tau))$.

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, $\mathit{args} \subseteq G^{\text{false}}(\pi)$.

The answers to the question type $\mu_Q \in \{\texttt{S-why-not}, \texttt{S-what-if}, \texttt{S-can}\}$ are based on the same information, however the answers are phrased differently.

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

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

- $\mathcal{E}(\langle \texttt{S-why-not}e, \mathit{args} \rangle) = \emptyset$: "*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?"

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

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

Both the number of MUS and MCS can be exponential in the number of goals, but for all question types $\mu_Q \in \{\texttt{US-why}, \texttt{US-how}, \texttt{S-why-not}, \texttt{S-what-if}, \texttt{S-can}\} = M_Q$ one goal subset $E \in \mathcal{E}(\langle \mu_Q, \mathit{args} \rangle)$ 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 $\hat{\mathcal{E}} \subseteq \mathcal{E}(\langle \mu_Q, \mathit{args} \rangle)$ and a summarization of the explanations $\hat{\mathcal{E}}$ is desirable.

6 Implementing Translators with LLMs

In this section, we describe our implementation of the interactive explanation framework of Figure 1 using goal-conflicts explanations (Section 5).

Each translator, as well as the planner and explainer services (which we now consider black-boxes that generate plans and explanations when provided a correct request) are *agents* that communicate with each other to solve the complex task of exploring the plan-space to find a satisfying plan with the help of a human agent via a chat interface.

6.1 Design Choices

The goal, question and explanation translator are implemented using LLMs. Several design choices are possible such as using a single LLM to act as all three translators or several LLMs, namely one for each translator. In this work we explore the second option where each task is handled by a different model, but still instantiated from the same base model (we use `gpt-4o-2024-11-20`, OpenAI's current flagship model). This allows us to completely separate the instructions for each LLM-based agent and manage their contexts as described at the end of this section.

We use OpenAI Assistants[1] to implement the translators. This API allows us to easily manage agents and to keep track of the context of the conversation of each translator separately. Relevant context that comes from other agents (translators, the planning system and explanation framework) is provided to each translator, as described in Subsection 6.3.

6.2 Communication Protocol

When users write a message in the chat interface (Figure 5), it is passed sequentially to the other agents in the order described in Figure 1. An exception is made when the question topic is about a goal already used in a previous planning

[1]https://platform.openai.com/docs/assistants/overview

iteration step (i.e, in $G^{\text{ref}}_{all} = \bigcup G^{\text{ref}}_i$): in this case, the goal translator is bypassed and the (already known) LTL$_\text{f}$ formula corresponding to the goal is retrieved deterministically.

Each LLM agent receives inputs as a structured string containing distinct sections corresponding to different input components (see Table 1 and prompts in Appendix A). The output is also a string and can be structured in different parts: the question translator T_Q classifies the question type μ_Q, and also outputs a goal description *args*, and a binary indicator of whether the goal has been used before. The goal translator T_G produces both the LTL$_\text{f}$ formula and a concise natural language description of the goal for UI reference. To ensure seamless interaction between other tool components and translators, these outputs must follow a specific format. While robust methods exist for enforcing output structures (Liu et al. 2024), we opted for a straightforward approach given our consistent and simple output requirements: outputs fields are delimited by semicolons (e.g., `"S-why-not ; C never on table ; ALREADY-USED"` for the question translator). While this simple delimiter-based approach served our immediate needs, developing a more robust solution remains as future work.

The iteration step and the planning task are automatically provided to the translators, so that the translators have access to the necessary information to perform their tasks. For example, the question translator T_Q needs to know if the current task is solvable, as well as the lists of goals currently enforced (G^{enf}) or not ($G^{\text{ref}} \setminus G^{\text{enf}}$), and for the latter, whether they are *satisfied* (in G^{true}) or *unsatisfied* (in G^{false}) in the current planning iteration.

Table 1: Input, Provider, and Output structure of each translator. τ is the planning task $\langle \mathcal{P}, \mathcal{O}, \mathcal{A}, I, G \rangle$, δ is the current iteration step $\langle G^{\text{ref}}, G^{\text{enf}}, \pi \rangle$, G^{ref}_{all} is the set of reference goals $\bigcup G^{\text{ref}}_i$ across all iteration steps $\Delta = \bigcup \delta_i$.

Component	Input	Provider	Output
Goal Translator (T_G)	NL Goal	User/T_Q	
	$\mathcal{P}$	τ	ϕ
	$\mathcal{O}$	τ	Goal Name
	G^{ref}_{all}	δ	
Question Translator (T_Q)	NL Question	User	
	G^{enf}	δ	
	$G^{\text{true}}(\pi)$	δ	μ_Q
	$G^{\text{false}}(\pi)$	δ	*args*
	G^{ref}_{all}	Δ	Used
	Solvability	δ	
Explanation Translator ($T_\mathcal{E}$)	NL Question	User	
	μ_Q	T_Q	
	args	T_Q	
	$\mathcal{E}(\langle \mu_Q, args \rangle)$	EF$_\text{CC}$	
	$\mathcal{P}$	τ	NL Expl.
	$\mathcal{O}$	τ	
	G^{enf}	δ	
	$G^{\text{true}}(\pi)$	δ	
	$G^{\text{false}}(\pi)$	δ	
	G^{ref}_{all}	Δ	

6.3 Context Management

By using the inputs and outputs specification from Table 1, we can develop a context-independent version of our implementation where translators have sufficient information to provide their translations without having access to previous user interactions. Here, we describe a context-dependent version that leverages context memory of LLMs to improve intra-step interactivity, motivated in Section 4.

The translators maintain their context across iteration steps, enabling explanations to build upon previous interactions. Translators only have access to their own previous inputs and outputs, but are provided directly in their input with other translators' relevant inputs and outputs (for example, the exact natural language question asked by the user is relevant to the explanation translator). Thus, our implementation supports across-step interactivity.

6.4 Goal Translation

Translating natural language to LTL$_\text{f}$ formulas is a non-trivial task with many possible ambiguities. We implemented a similar translator as what Liu et al. (2022) call *End-to-End Approach*. We do not restrict formulas to match a set of predefined templates like Fuggitti and Chakraborti (2023) do. This has the advantage of potentially being able to translate more complex goals and is easier to implement but can be unreliable. Future work could include a more sophisticated approach to goal translation.

6.5 Explanation Selection and Summarization

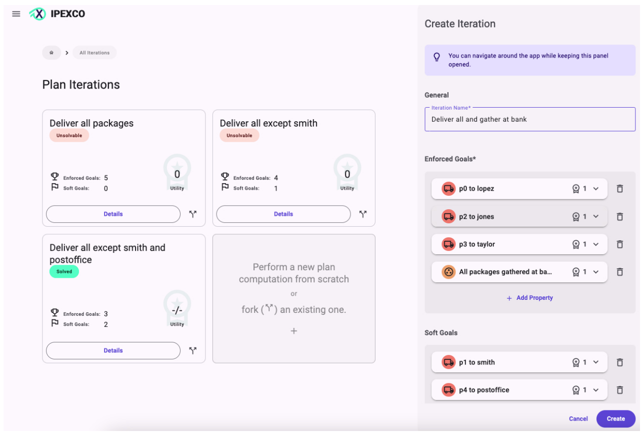
We expect the explanation translator $T^{\mathcal{C}}_{\mathcal{E}}$ to leverage prompt examples to understand how it is supposed to select and/or summarize explanations. It is thus a domain-specific design choice to provide examples of the expected selection/-summarization strategy. For the current implementation, we stick to simple examples that do not require summarization, but the example shown in the prompt Appendix A.4 shows how the translator can perform selection. Future work includes ensuring users can actively request summarization or selection in follow-up requests.

7 Evaluation Tool

We develop a web-based platform (built on top of previous work (Eifler et al. 2022) with an improved UI and the addition of LLM features) implementing our framework. The planning service is based on Fast Downward (Helmert 2006) and the explanations service computes the MUS and MCS using the algorithms introduced in (Eifler et al. 2020b) also implemented in Fast Downward. We will release our code publicly upon acceptance. We demonstrate the features using a reasonably sized instance of a transportation domain.

The tool provides an overview of the iteration steps (left Figure 3) and a detailed view for each step with access to the sample plan and the explanations (Figure 5). During the inspection of this information users can create the next iteration step in a sidebar by choosing enforced goals G^{enf} and soft goals $G^{\text{ref}} \setminus G^{\text{enf}}$ (right Figure 3).

Goals can be created using predefined templates or a chat interface with $T^{\mathcal{C}}_{G}$, as illustrated in Figure 4. The translated

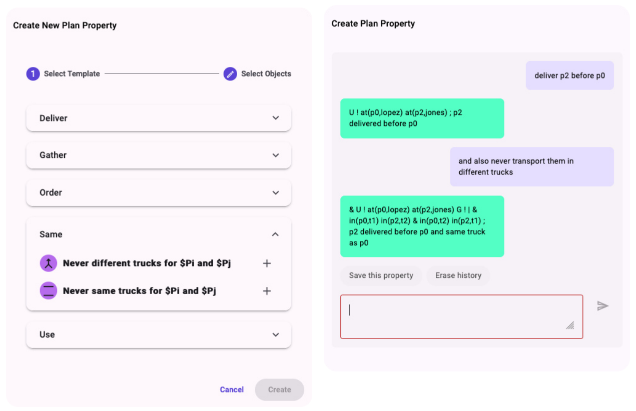


You can navigate around the app while keeping this panel opened.

Deliver all and gather at bank

Figure 3: Screenshot of the project view where users can navigate between planning iteration steps and create new ones using the panel on the right side.

LTL_f formulas shown in this figure are intended to be shown exclusively to expert users. The example illustrated a multi-step interaction with $T_G^{\mathcal{C}}$ to refine the goal which is only possible with a context-dependent translator.



deliver p2 before p0

U ! at(p0,lopez) at(p2,jones) ; p2 delivered before p0

and also never transport them in different trucks

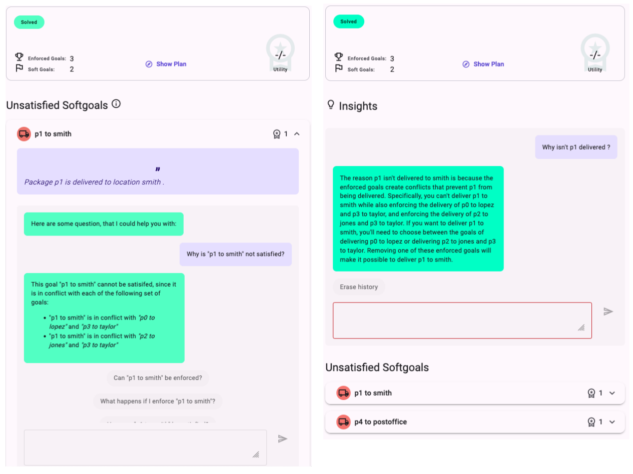
& U ! at(p0,lopez) at(p2,jones) G ! | & in(p0,t1) in(p2,t2) & in(p0,t2) in(p2,t1) ; p2 delivered before p0 and same truck as p0

Figure 4: Screenshot of the goal creation using templates baseline (left) and LLM-based goal translator (right).

Figure 5 compares the template and the LLM based generation of natural language explanations. When using the templates the user is restricted to the questions in the bottom of the chat window and the answer simply lists the conflicts. In the LLM-based chat, users can freely formulate questions. The response goes beyond listing conflicts; it attempts to answer the user's question by providing coherent text that can perform selection or summarization, refer to previous interactions, and adapt to the user's vocabulary.

8 Conclusion and Future Work

In this paper we presented a high-level architecture that provides a framework for producing interactive explanations in the context of iterative planning. We motivated our framework by noting that interactive explanations require a fluid



"
Package p1 is delivered to location smith .

Here are some question, that I could help you with:

Why is "p1 to smith" not satisfied?

This goal "p1 to smith" cannot be satisifed, since it is in conflict with each of the following set of goals:

- "p1 to smith" is in conflict with *"p0 to lopez"* and *"p3 to taylor"*
- "p1 to smith" is in conflict with *"p2 to jones"* and *"p3 to taylor"*

Can "p1 to smith" be enforced?

What happens if I enforce "p1 to smith"?

Why isn't p1 delivered ?

The reason p1 isn't delivered to smith is because the enforced goals create conflicts that prevent p1 from being delivered. Specifically, you can't deliver p1 to smith while also enforcing the delivery of p0 to lopez and p3 to taylor, and enforcing the delivery of p2 to jones and p3 to taylor. If you want to deliver p1 to smith, you'll need to choose between the goals of delivering p0 to lopez or delivering p2 to jones and p3 to taylor. Removing one of these enforced goals will make it possible to deliver p1 to smith.

Figure 5: Screenshot of the explanation chat for a solvable iteration step using templates-based baseline (left) and LLM-based multi-agent interface.

and flexible natural language interface, which is something that LLMs can effectively provide. On the other hand, given that prior research has shown that LLMs on their own are not ideal planners, we need a formal planning system that can guarantee the correctness of the explanations the system proposes. Our framework thus uses a multi agent LLM, in which agents handle translations from a user's natural language requests to formal queries to system and back from formal explanations to an explanation in natural language. We illustrated our framework with goal-conflict explanations in planning and we provided an implementation designed to be accessible to non-experts, which will serve as a basis for evaluating our proposed framework.

Future work includes developing more robust and flexible communication protocols between agents, supporting feedback and clarification at each step of the process, and mitigating LLMs potential limitations like hallucinations.

We are also investigating the extension of this approach to other explanations frameworks. This includes MUS and MUC based approaches used in CP (Povéda et al. 2024; Gamba, Bogaerts, and Guns 2023) but also other approaches used in planning (Krarup et al. 2021; Sreedharan, Chakraborti, and Kambhampati 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.

References

Achinstein, P. 1980. *The Nature of Explanation*. Oxford University Press.

Asher, N.; Bhar, S.; Chaturvedi, A.; Hunter, J.; and Paul, S. 2023. Limits for learning with language models. In *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (* SEM 2023)*, 236–248.

Baier, J. A.; and McIlraith, S. A. 2006. Planning with first-order temporally extended goals using heuristic search. In *Proc. AAAI*, 788–795.

Bhar, S.; and Asher, N. 2024. Strong hallucinations from negation and how to fix them. In *Findings of the Association for Computational Linguistics ACL 2024*, 12670–12687.

Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R.; Arora, S.; von Arx, S.; Bernstein, M. S.; Bohg, J.; Bosselut, A.; Brunskill, E.; et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Bromberger, S. 1962. An Approach to Explanation. In Butler, R., ed., *Analytical Philsophy*, 72–105. Oxford University Press.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 1877–1901.

Brunello, A.; Montanari, A.; and Reynolds, M. 2019. Synthesis of LTL formulas from natural language texts: State of the art and research directions. In *26th International symposium on temporal representation and reasoning (TIME 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In Sierra, C., ed., *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 156–163. ijcai.org.

Cosler, M.; Hahn, C.; Mendoza, D.; Schmitt, F.; and Trippel, C. 2023. nl2spec: interactively translating unstructured natural language to temporal logics with large language models. In *International Conference on Computer Aided Verification*, 383–396. Springer.

De Giacomo, G.; De Masellis, R.; and Montali, M. 2014. Reasoning on LTL on finite traces: Insensitivity to infiniteness. In *AAAI*.

Domshlak, C.; and Mirkis, V. 2015. Deterministic Oversubscription Planning as Heuristic Search: Abstractions and Reformulations. *JAIR*, 52: 97–169.

Edelkamp, S. 2006. On the Compilation of Plan Constraints and Preferences. In *ICAPS*, 374–377.

Eifler, R.; Brandao, M.; Coles, A.; Frank, J.; and Hoffmann, J. 2022. Evaluating Plan-Property Dependencies: A Web-Based Platform and User Study. In *ICAPS*, 687–691.

Eifler, R.; Cashmore, M.; Hoffmann, J.; Magazzeni, D.; and Steinmetz, M. 2020a. A New Approach to Plan-Space Explanation: Analyzing Plan-Property Dependencies in Oversubscription Planning. In *AAAI*.

Eifler, R.; Steinmetz, M.; Torralba, A.; and Hoffmann, J. 2020b. Plan-Space Explanation via Plan-Property Dependencies: Faster Algorithms & More Powerful Properties. In *IJCAI*, 4091–4097.

Fuggitti, F.; and Chakraborti, T. 2023. NL2LTL – A Python Package for Converting Natural Language (NL) Instructions to Linear Temporal Logic (LTL) Formulas. In *ICAPS*. Best System Demonstration Award Runner-Up.

Gamba, E.; Bogaerts, B.; and Guns, T. 2023. Efficiently Explaining CSPs with Unsatisfiable Subset Optimization. *J. Artif. Intell. Res.*, 78: 709–746.

Guo, T.; Chen, X.; Wang, Y.; Chang, R.; Pei, S.; Chawla, N. V.; Wiest, O.; and Zhang, X. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

Helmert, M. 2006. The Fast Downward Planning System. *JAIR*, 26: 191–246.

Junker, U. 2004. Preferred explanations and relaxations for over-constrained problems. In *AAAI-2004*.

Kambhampati, S. 2024. Can large language models reason and plan? *Annals of the New York Academy of Sciences*, 1534(1): 15–18.

Kambhampati, S.; Valmeekam, K.; Guan, L.; Verma, M.; Stechly, K.; Bhambri, S.; Saldyt, L.; and Murthy, A. 2024. Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*.

Krarup, B.; Coles, A. J.; Long, D.; and Smith, D. E. 2024. Explaining Plan Quality Differences. In Bernardini, S.; and Muise, C., eds., *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2024, Banff, Alberta, Canada, June 1-6, 2024*, 324–332. AAAI Press.

Krarup, B.; Krivic, S.; Magazzeni, D.; Long, D.; Cashmore, M.; and Smith, D. E. 2021. Contrastive Explanations of Plans through Model Restrictions. *J. Artif. Intell. Res.*, 72: 533–612.

Kushilevitz, E. 1997. Communication complexity. In *Advances in Computers*, volume 44, 331–360. Elsevier.

Liu, J. X.; Yang, Z.; Schornstein, B.; Liang, S.; Idrees, I.; Tellex, S.; and Shah, A. 2022. Lang2ltl: Translating natural language commands to temporal specification with large language models. In *Workshop on Language and Robotics at CoRL 2022*.

Liu, M. X.; Liu, F.; Fiannaca, A. J.; Koo, T.; Dixon, L.; Terry, M.; and Cai, C. J. 2024. " We Need Structured Output": Towards User-centered Constraints on Large Language Model Output. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 1–9.

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *AI*, 267: 1–38.

Mirzadeh, I.; Alizadeh, K.; Shahrokhi, H.; Tuzel, O.; Bengio, S.; and Farajtabar, M. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.

Morandini, S.; Fraboni, F.; Hall, M.; Quintana-Amate, S.; and Pietrantoni, L. 2024. Human factors and emerging needs in aerospace manufacturing planning and scheduling. *Cognition, Technology & Work*, 1–19.

Peng, B.; Narayanan, S.; and Papadimitriou, C. 2024. On limitations of the transformer architecture. *arXiv preprint arXiv:2402.08164*.

Povéda, G.; Strahl, A.; Hall, M.; Boumazouza, R.; Quintana-Amate, S.; Alvarez, N.; Bleukx, I.; Tsouros, D.; Verhaeghe, H.; and Guns, T. 2024. Trustworthy and Explainable Decision-Making for Workforce allocation. In *The Seventh Workshop on Progress Towards the Holy Grail (PTHG-24) at CP*.

Sanford, C.; Hsu, D. J.; and Telgarsky, M. 2024. Representational strengths and limitations of transformers. *Advances in Neural Information Processing Systems*, 36.

Smith, D. 2012. Planning as an Iterative Process. In *AAAI*, 2180–2185.

Smith, D. E. 2004. Choosing Objectives in Over-Subscription Planning. In *ICAPS*, 393–401.

Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2021. Foundations of explanations as model reconciliation. *Artif. Intell.*, 301: 103558.

Sreedharan, S.; Srivastava, S.; Smith, D. E.; and Kambhampati, S. 2019. Why Can't You Do That HAL? Explaining Unsolvability of Planning Tasks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 1422–1430.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Valmeekam, K.; Marquez, M.; Sreedharan, S.; and Kambhampati, S. 2023. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36: 75993–76005.

Vasileiou, S. L.; and Yeoh, W. 2023. PLEASE: Generating Personalized Explanations in Human-Aware Planning. In *ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland - Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023)*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, 2411–2418. IOS Press.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 5998–6008.

A LLM Agent Prompts

Each agent has an associated instruction prompt that precedes a few (3-5) domain-specific examples of the task. In this section, we will only show the instruction prompt and one example from a transport domain. The full prompts including all provided examples will be made available in the code repository associated with this paper.

A.1 System Prompt

This prompt is used as a prefix for each LLM agent below.

System Prompt

You are an AI assistant that faithfully answers user questions using the help of multiple modules. Carefully follow the instructions without any deviation and never add any comments of your own.

A.2 Question Translator

Question Translator Prompt

You are an AI assistant that faithfully answers user questions using the help of multiple modules. Carefully follow the instructions without any deviation and never add any comments of your own. You will be given a user question that is semantically equivalent to one of these questions :
US-WHY: ”Why is the task unsolvable?” ;
US-HOW: ”How can I make the task solvable?” ;
S-WHY-NOT: ”Why are Q not satisfied?” ;
S-WHAT-IF: ”What happens if we enforce Q?”;
S-CAN: ”Can Q be satisfied?” ;
S-CAN: ”How can Q be satisfied?”/”How can I improve the plan with respect to o?” ;
Any question you will be given can be mapped to one of these questions. Return the question type and the goal or plan property the user is refering to, without any additional comments of any kind. First, try to see if the question is about a property that is already existing. If it is, return ALREADY-USED as the last part of your answer and use the exact same name. If it is not, invent a name for this property and use the encoding NEVER-USED. If the question is not about a specific property but about the unsolvability of the task, return NOGOAL as the second part of your answer and ALREADY-USED as the last
Examples :
Question: What happens if we enforce using the same truck for p1 and p2?
Enforced Goals: [p0 to bank before p1 to cafe, p0 to cafe]
Satisfied Goals: [p2 to smith]
Unsatisfied Goals: [Never different trucks for p1 and p2]
Existing Plan Properties: [p0 to bank before p1 to cafe, p0 to cafe, p2 to smith, Never different trucks for p1 and p2]
Solvable: True
Return: S-WHAT-IF ; Never different trucks for p1 and p2 ; ALREADY-USED
End of the examples.
[Additional examples omitted for brevity]

A.3 Goal Translator

Goal Translator Prompt

You are an AI assistant that faithfully answers user questions using the help of multiple modules. Carefully follow the instructions without any deviation and never add any comments of your own.You will be given a goal and a list of allowed predicates and objects from a planning problem and domain.
Return the LTLf formula corresponding to the goal, followed by a short summary of the goal in natural language (the fewer words the better) that will be used to refer to it. The LTLf formula must be well-formed and must only contain the predicates and objects given to you. Users might reference previous messages they sent or implicit objects. You should infer that when possible based on the provided information and previous messages. If it is not possible to express the property in LTL using the provided objects and predicates, return UNSUPPORTED ; Unsupported property.
Examples :
Goal: make sure p1 visits the packingstation at some point
Predicates: [(at ?o - object ?l - location), (in ?p - package ?t - truck), (empty ?t - truck), (fuel ?truck - truck ?level - fuellevel)]
Objects: [postoffice, supermarket, taylor, cafe, bank, lopez, jones, smith, packingstation, t1, t2, p0, p1, p2, p3, p4]
Existing Plan Properties: [p0 to cafe, p2 to smith, t1 always above level5]
Return: F at(p1,packingstation) ; p1 visits packingstation
End of the examples.
[Additional examples omitted for brevity]

A.4 Explanation Translator

Explanation Translator Prompt

You are an AI assistant that faithfully answers user questions using the help of multiple modules. Carefully follow the instructions without any deviation and never add any comments of your own.You will be given a question, a list of achieved and unachieved goals, and a conflict list with each conflict presented as a Minimum Unsolvable Goal Subset (MUGS) except if the original question is a How question. In that case, you are provided with list of resolutions, presented using Minimal Goal Correction Sets (MGCS).
All goals in a MUGS cannot be achieved in the same plan, but any strict subset of them can be achieved.
For example the conflict : [a,b,c] means that it is possible to achieve (a and b) or (a and c) or (b and c) in the same plan, but not (a and b and c). If provided with MGCS, each set represent a set of goals it is sufficient to forgo to make the planning task solvable. Choosing any of these and making sure that none are enforced should be sufficient.
Answer the question using the explanation framework to help users to understand what conflict causes their problem and what they can do to solve it. It is crucial to base anwsers on conflicts or resolutions, to ensure providing faithful explanations. Never mention MUGS or MGCS, refer to them as ”conflicts” or ”resolutions” and only use vocabulary users will understand.
Examples :
Question: Why is p2 to jack unsatisfied?
Question Type: S-WHY-NOT
Question Arguments: p2 to jack
MUGS: [[p0 to bank, p3 to lopez], [p1 same truck as p2, p1 to smith],[p1 same truck as p2, p3 to lopez]]
MGCS:
Enforced Goals: [p0 to bank, p1 to smith, p1 same truck as p2]
Satisfied Goals: [p3 to lopez]
Unsatisfied Goals: [p4 to coffee, p2 to jack]
Existing Plan Properties: [p0 to bank, p1 to smith, p2 to jack, p3 to lopez, p4 to coffee, p1 same truck as p2, Always visit packingstation before supermarket, Never different trucks for p1 and p2]
Return: The main conflict is that you can’t have p1 and p2 delivered while enforcing using the same truck for them. If you still choose to deliver p2, you will have to choose between p0 and p3, and choose between delivering p3 or using the same truck for p1 and p2.
End of the examples.
[Additional examples omitted for brevity]