IPEXCO: A Platform for Iterative Planning with Interactive Goal-Conflict Explanations

Rebecca Eifler[1], Guilhem Fouilhé[2]

[1]Laboratory for Analysis and Architecture of Systems (LAAS-CNRS), 7 Avenue du Colonel Roche, 31400 Toulouse, France
[2]Institut de Recherche en Informatique de Toulouse (IRIT), 118 Route de Narbonne, 31062 Toulouse, France

Abstract

When automating plan generation for a real-world sequential decision problem, the objective is often not to replace the human planner, but rather to facilitate an iterative reasoning and elicitation process. In this process, the human's role is to guide the planner according to their preferences and expertise. In this context, explanations that address users' questions are crucial to improve their understanding of potential solutions and increase their trust in the system. We present a platform that implements this iterative planning approach and provides explanations to user questions based on conflicting goals and preferences. The platform supports both a classical template-based interface and a multi-agent Large Language Model (LLM) architecture that enables interactive explanations tailored to the user and context. The integration of online user studies allows for the evaluation of the effectiveness of the explanations and the impact of the communication interface. The platform code is available at https://github.com/r-eifler/IPEXCO-frontend.

Keywords

Planning, Explanations, minimal unsolvable sets (MUS), LLM, conversational

1. Introduction

Real-world problems are often oversubscribed, meaning that due to constraints such as limited resources or time, not all goals and user preferences can be satisfied. An iterative planning process, as outlined in [1], enables users to explore the plan space and to refine their preferences to identify a satisfactory trade-off. In such an interactive planning setup, explanations that help the user to understand the dependencies between the goals and their preferences are crucial. We present IPEXCO, an online platform that facilitates iterative planning with interactive explanations. The platform has two main objectives. First, it provides a user-friendly interface for iterative planning, supported by goal conflict explanations as introduced in [2] and more recently extended in [3]. Explanations based on minimal unsolvable sets (MUS) or minimal conflicts are also relevant in constraint programming [4, 5]. They provide contrastive explanations [6] to user questions such as "Why does the plan not satisfy property p?" by offering insights into the properties that cannot longer be satisfied in the alternative proposed in the question. These explanations are accessible either via a template-based interface or via an interactive natural language interface leveraging the recent advances in large language models, while still relying on specialized algorithms to compute the explanations. To facilitate the evaluation of the explanations, our platform supports online user studies, in a controlled environment, allowing participants to utilize their own strategies during the planning process and the exploitation of the explanations.

In the following we provide an overview of the main platform features, the iterative planning process, goal conflict explanations and the user study support. Then we present the interactive explanation interface based on a multi-agent LLM architecture. Finally, we give an overview of the modular platform architecture.

Late-breaking work, Demos and Doctoral Consortium, colocated with The 3d World Conference on eXplainable Artificial Intelligence: July 09–11, 2025, Istanbul, Turkey

*Corresponding author.
rebecca.eifler@laas.fr (R. Eifler); guilhem.fouilhe-lafforgue@irit.fr (G. Fouilhé)
0000-0001-8275-7813 (R. Eifler); 0009-0002-7108-9930 (G. Fouilhé)

2. Iterative Planning Process

The platform implements an iterative planning process. In each iteration step, the user is presented a sample plan π_i (if one exists). Based on π_i, the user must then decide which goals and preferences the sample plan π_{i+1} in the next iteration should satisfy. If the goals and preferences enforced by the user are unsolvable, no sample plan can be computed. To restore solvability, the user must then decide which goals and preferences to forego. During the exploration of the plan space new preferences may emerge, and goals which are to restrictive might be dropped. Our platform facilitates this iterative process by offering interfaces to access the individual iteration steps, along with the preferences and goals satisfied in each step. Additionally, it allows users to define temporal goals, reflecting their preferences. The planning task is given by a PDDL domain and problem file and the goals are defined during the planning process via different interface options, which are outlined next.

Goal Creation At the beginning of the planning process, the user defines all known goals and potential preferences. As the planning process progresses, the initial goals and preferences are refined based on the analysis of sample plans. Goals are defined by LTL_f [7] formulas with literals based on the predicates and objects of the planning task. To facilitate the goal creation process, the requirement to write LTL_f is bypassed, as it is not suitable for laypeople. Two options are supported.

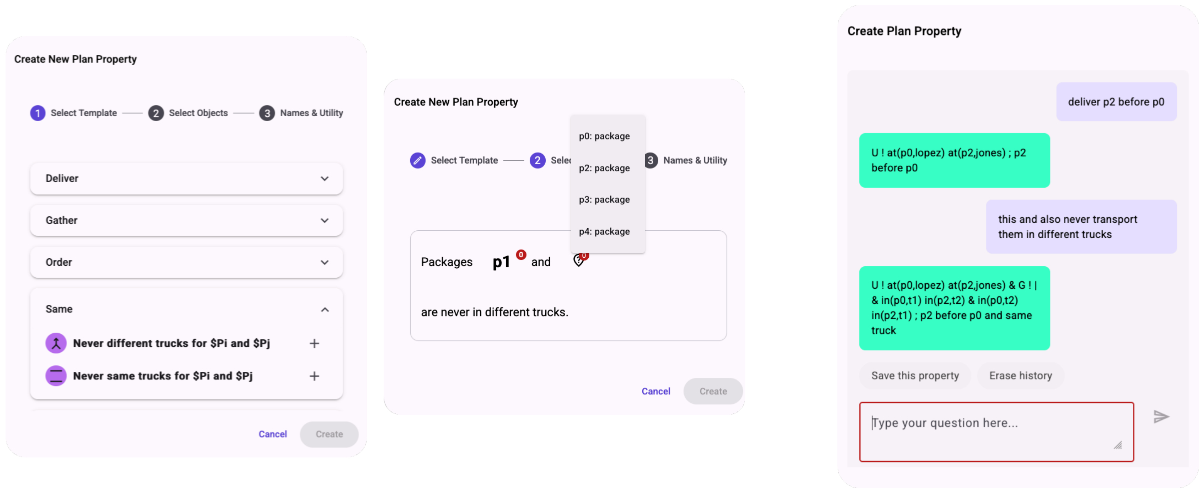

Figure 1: Screenshot of the goal creation using templates (left, middle) and LLM-based goal translator (right).

First, it is possible to define templates for each domain to cover commonly used goals and temporal preferences. Such a template maps a natural language description, e. g. “Load package `$P_i` before package `$P_j` into truck `$T`” to an LTL_f formula: $\neg in(P_j, T)$ U $in(P_i, T)$. To restrict the objects offered for selection, the allowed types (e.g. `$P_i` must be of type `package`) and facts that must/must not be satisfied in the initial state (e.g. $\neg in(P_j, T)$) can be specified. This helps to ensure that only well-formed goals can be instantiated. In Figure 1 on the left a list of possible templates for a logistics domain are shown. The first step in creating a goal is choosing one of those templates. In the second step the objects are selected.

The second option is to use a LLM-based goal translator, that translates a natural language description of the goal into an LTL_f formula. An example is given on the right in Figure 1. The LLM infers the delivery location from the planning task. The translated LTL_f formulas shown in this figure are intended to be shown exclusively to expert users. An alternative approach to verify the translation involves the use of a *reverse translation* of the users’ question, which is then converted back into natural language without ambiguities. Several tools such as `NL2LTL` [8], `Lang2LTL` [9], and `nl2spec` [10] use LLMs and prompting to translate temporal goals into LTL or LTL_f. We opted for a simple base implementation, similar to the *End-to-End Approach* described by [9].

Iterative Planning Process During an iterative planning process the user explores the plan space by iteratively enforcing goals and inspecting the resulting sample plans. The interface is designed to provide users with the information they need to make informed decisions about which goals to enforce. To make this decision, users need an overview of the existing iteration steps and access to more detailed information for each step. At the same time, users should be able to select the goals to be enforced in the next iteration step. Our tool implements this process as follows.

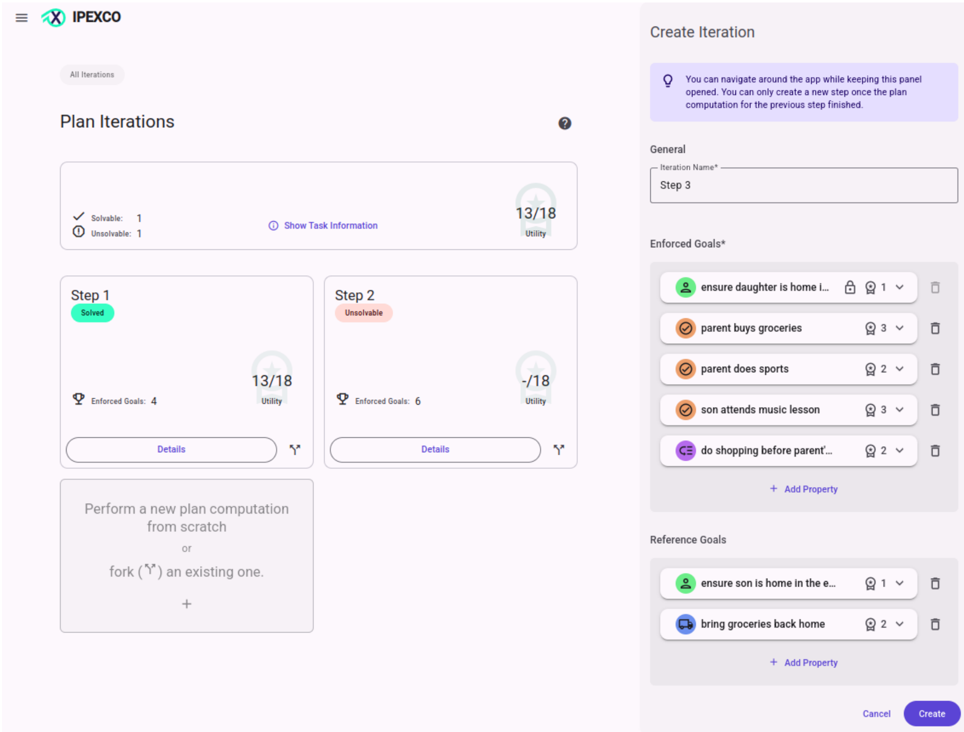

Figure 2: Screenshot of the iteration steps overview with the interface to create new steps on the right.

The overview of the iteration steps is shown on the left side of Figure 2. This view allows to quickly identify which steps were unsolvable and solvable along with the utility they achieved. The details view of an iteration step offers, in addition to the information provided in the overview, the following details. For a solvable iteration step (a plan satisfying the enforced goals exists), the additional satisfied reference goals, the unsatisfied reference goals and the enforced goals are listed. The sample plan satisfying the enforced goals, is accessible and displayed in a separate view. By default, the plan is displayed as a list of actions; alternatively, it is possible to implement a domain-dependent plan visualization. For an unsolvable iteration step (no plan satisfying the enforced goals exists), the enforced goals are listed.

The interface to create the next iteration step is located in a side panel, as shown on the right in Figure 2. To create a new step, a user must select at least one enforced goal. These goals can be either existing or newly created. Additionally, users can also select or create new reference goals. Reference goals represent objectives and preferences the user is interested in, but that need not be satisfied by the next sample plan. The reference goals are considered in the explanations offering insights on how to satisfy them in subsequent iteration steps.

3. Goal Conflict Explanations

Goal-conflict explanations in planning were introduced in [2] and more recently extended in [3]. They are based on *minimal unsolvable subsets* (MUS) and *minimal correction sets* (MCS), a concept also used in constraint programming [4, 5]. A MUS G is a set of goals that is not solvable, i. e. there exists no plan that satisfies all goals in G, but all proper subsets of G are solvable. This means a MUS is a minimal conflict. For example in a logistics problem the set $\{P_1, P_2, P_3\}$, where P_i indicates that package i is delivered, is a MUS if it is not possible to deliver all three package due to for example limited fuel, but any combination of two packages can be delivered. Given a set of goals G_{unsat} that is unsolvable, an MCS is a subset $G \subseteq G_{unsat}$ such that $G_{unsat} \setminus G$ is solvable but for each proper subset of G this is not the case. This means an MCS is a minimal set of goals one has to forego to restore solvability. If for example the packages $\{P_1, P_2, P_3, P_4\}$ can not be delivered, then the set $\{P_1, P_2\}$ is an MCS if $\{P_3, P_4\}$ can be delivered but additionally delivering either P_1, or P_2 is not possible. Note that the conflicts we analyze, such as those related to the delivery of packages or more generally to any temporal properties of a set of plans, are not caused by inconsistencies in the planning model [11]. Conflicts may arise due to various factors, including resource or time constraints, or conflicting objectives, such as minimizing the overall driving distance while ensuring even utilization of the individual trucks.

Based on MUS and MCS questions about unsolvable as well as solvable iteration steps can be addressed. In an unsolvable step, the MUS of the enforced goals provide insights into *why* the step is unsolvable, while the MCS offer guidance on *how* to restore solvability. In a solvable step questions such as *"Why is goal g not satisfied?"*, *"What happens if I enforce goal g?"* or *"How can I satisfy goal g?"* can be addressed. These questions refer to goals that are not satisfied by the sample plan, but the user is considering enforcing them in the next iteration step and wants to identify the implications of doing so. This approach offers contrastive explanations [6] that provide insights into the goals that cannot longer be satisfied in the alternative proposed in the question. For a more detailed description we refer to [3].

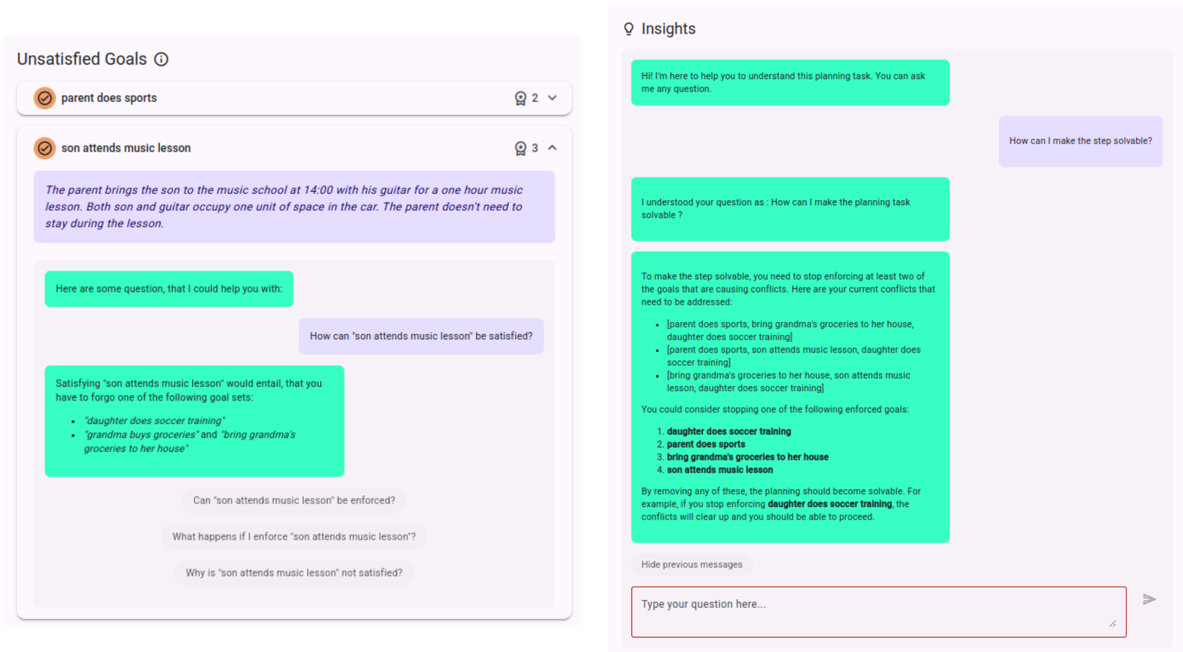

Figure 3: Screenshot of the template-based explanations for a solvable step (left) and of the LLM-based explanations for an unsolvable step (right).

Explanations are provided in the details view of the iteration steps. The access to the explanations and their presentation differs depending on the chosen interface. Currently, two options are supported. In the template based explanation interface the user is restricted to a predefined list of questions and the answer lists all explanations. The questions and explanations are displayed in a chat interface, as shown on the left in Figure 3. For an unsolvable step the questions refer to the entire selection of enforced goals. Therefore, the explanation interface is displayed above the list of enforced goals. In a solvable step the questions refer to individual unsatisfied reference goals. Therefore, for each unsatisfied goal there is a dedicated explanation interface accessible by clicking on the goal (see left Figure 3).

The LLM-Chat based explanation interface is displayed in the top of the iteration step details view for both solvable and unsolvable steps (see right Figure 3). Users can submit questions and receive answers via a chat interface. The interface employs a hybrid approach. The LLM-agents function as a bridge, translating the user's natural language into the formal language of the planner, which serves as the reasoner and computes the MUS and MCS. More examples and details can be found in the next section.

4. Interactive Explanations with LLMs

As described in Section 3, IPEXCO supports an LLM-powered explanation interface introduced in [3]. This interface offers several key advantages for users interacting with the planning system. First, it allows users to formulate their questions in natural language, with an LLM translating these questions into formal queries for the explanation service. Secondly, the LLM converts the technical explanations generated by the explanation service into natural language responses tailored to the user's original question. This not only creates a more natural conversation flow and adapts to the user's vocabulary, but also enables more complex interactions, including requests for summarized or partial explanations as needed. Finally, using LLMs can facilitate the overall interaction by asking users to clarify their requests when ambiguities arise.

Similar approaches involving LLM to enrich explainer system outputs have been proposed in explainable machine learning [12, 13, 14, 15] and in scheduling [16]. The LLM interface implemented in IPEXCO shares similarities with these works while being the first such application to planning systems. The technical implementation of this LLM-powered explanation system is based on a multi-agent approach where specialized LLM agents essentially perform translation tasks between natural language and formal explanation queries.

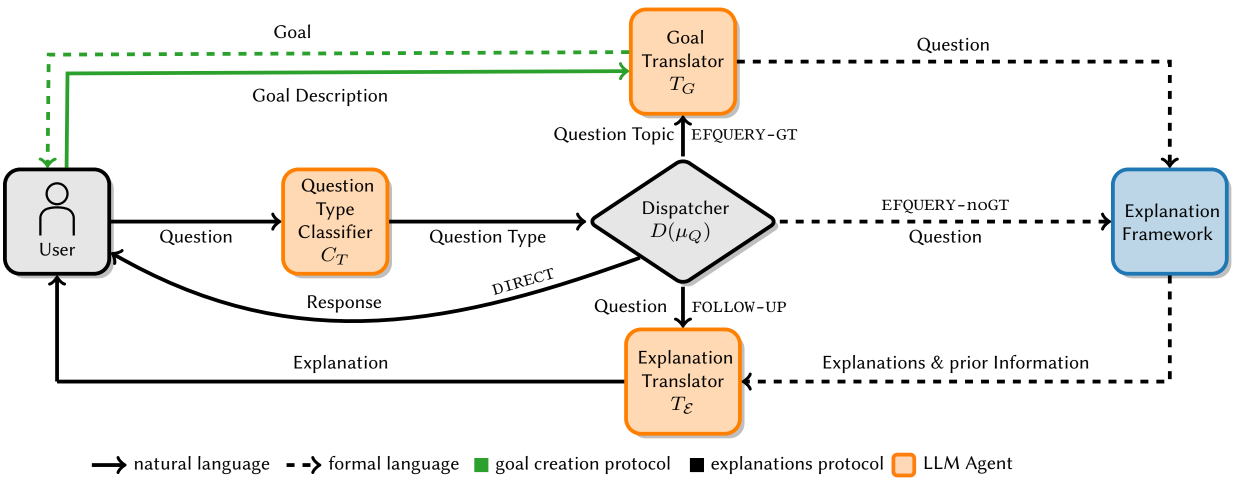

Figure 4: Communication protocol between the user, the translators and the explanation service. In addition to the information provided in the messages, each agent has access to the planning task and the iteration step. For a goal translation (green, top left) the user directly communicates with the goal translator. 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 the question type classifier.

Multi-Agent LLM Architecture Our system (Figure 4) consists of three main components: **Question Type Classifier**, which analyzes the user's question to determine its type and how it should be processed (it can directly answer simple questions like "What can I ask", identify follow-up questions that reference previous explanations, or route supported questions to the explanation service while translating the question into a formal request); **Goal Translator**, which converts natural language goals into LTL_f formulas that the planning system understands and creates clear natural language descriptions of these goals; and **Explanation Translator**, which converts technical explanations from the formal system into natural language responses tailored to the user's original question. The system maintains context across interactions, allowing for natural conversation flow with follow-up questions and references to previous explanations. For example, a user might ask “Why can’t I achieve goal X?” followed by “How can I fix that?” without needing to restate the original goal. Beyond allowing a natural language interaction, this system also allows more complex interactions with explanations including selection and summarization, which can be actively requested by users.

Example interaction We demonstrate the capabilities of the LLM-chat based explanation interface through an example interaction in an unsolvable iteration step. In the UI, these examples would be formatted in Markdown for enhanced readability.

User: What are the conflicts?

`LLM: Here are the current conflicts: [Bring friend to sports center, parent does sports, bring grandma’s groceries to her house, parent buys groceries, grandma and friend are not together in car], [Bring friend to sports center, parent does sports, bring groceries back home, bring grandma’s groceries to her house], [Bring grandma’s groceries to her house, do shopping before parent sports] These conflicts indicate issues with achieving your goals based on the currently enforced conditions.`

User: Thanks, is there one goal that is involved in all of these?

`LLM: Yes, the goal "bring grandma’s groceries to her house" appears in all of the listed conflicts. This indicates that it is a common factor contributing to the conflicts preventing successful planning.`

User: So removing it is enough to make the task solvable, right?

`LLM: Yes, removing the goal "bring grandma’s groceries to her house" would be sufficient to resolve the conflicts and make the planning task solvable. By forgoing this goal, you would eliminate all conflicts currently preventing you from achieving your objectives.`

5. User Study Support

In order to evaluate the usefulness of explanations and the impact of different interfaces, user studies are required. To ensure a high-quality evaluation, participants should be given comparable tasks and enough freedom to develop their own strategies and use the explanations according to their preferences.

The platform provides a controlled environment through the use of planning processes with a fixed set of goals. Each goal is assigned a utility that serves as a proxy for an intrinsic preference of the participants. This approach ensures that all users are presented with the same optimization task, which is to maximize the utility of a solvable iteration step. Note that the computation of explanations can require a significant amount of time. The fixed set of goals allows for the pre-computation of explanations, ensuring a responsive experience. Certain features of the iterative planning process may be disabled. Participants who do not have access to sample plans, but only to the information and explanations provided in the iteration step details view, must rely on the explanations. To encourage a high performance, achievable utility for the task and the utility of the best iteration step are displayed.

The platform facilitates online user studies by leveraging the recruitment platform Prolific. User studies are composed of different parts that the user study coordinator can use to set up a user study to their needs. The options are: description steps (general information and instructions), external links (for example for questionnaires), a tool manual (general usage instructions for the tool and the selected interfaces for explanations), planning task information (description of the domain, the specific instance, and list of available goals), and planning tasks (iterative planning process with the objective to maximize the utility). Each part is associated with a minimum or maximum processing time, which restricts the progress of participants. This incentivizes users to allocate a reasonable amount of time to instructions or questionnaires and to limit the processing time of the optimization task. Given the complexity of the tool, it is essential that users receive a training to understand and fully use all its features. For this purpose, it is possible to select a task as an introductory task that contains additional instructions for the user. All actions performed by the user are tracked with timestamps, allowing for the analysis of user strategies and decisions over time. The platform also supports the monitoring of participants during the study, by providing access to metrics such as completion time and achieved utility. For a comprehensive statistical analysis, the user data can be exported as a JSON file.

6. Architecture

The tool is a web platform, i. e. it runs directly in the browser and does not require installation on individual users' machines. This feature is particularly advantageous for conducting user studies.

The system features a modular architecture, in which explanations and plans are provided by individual services. The back-end and the explanation and planning services implement REST APIs, and communication between the back-end and a service is asynchronous. The services themselves maintain a queue for incoming jobs and schedule them based on resource availability. This enables the integration of additional planner and explanation services, and facilitates the expansion of the platform. The supported questions are also modular, so that they can be extended depending on the explanation service selected. The presentation of explanations is not easily generalizable. However, the template-based approach can be easily extended, and the prompts of the LLM-based translators are configurable to accommodate different formal explanation languages.

The included planner service is based on Fast Downward (FD) [17] extended with a compilation approach to support temporal goals [18]. The explanations service computes MUS and MCS using the algorithms introduced in [18] implemented in FD. The LLM implementation, based on prompting, supports the usage of OpenAI models through the Chat Completion API that supports Structured Outputs.This restriction ensures a strict format for all LLM-generated text, which ensures that the user interaction follows the specification in Figure 4. The UI provides interfaces to set up model name (e. g. `gpt-4o-mini`), temperature , maximum completion tokens, prompts and output formats.

7. Conclusion & Future Work

We have introduced IPEXCO, an online platform that facilitates iterative planning with explanations. The platform offers interfaces to goal-conflict explanation, leveraging LLMs to enhance interactivity. The support for user studies enables the evaluation of information and the communication of explanations.

Moving forward, we aim to incorporate additional explanation methods from planning [19, 20], enhancing their accessibility and facilitating a comparative and evaluative analysis of their applicability, advantages, and disadvantages.

Acknowledgments

This work was funded by the European Union's Horizon Europe Research and Innovation program under the grant agreement TUPLES No 101070149, and was supported by the Artificial and Natural Intelligence Toulouse Institute (ANITI), funded by the French Investing for the Future PIA3 program under the Grant agreement ANR-19-PI3A-000.

Declaration on Generative AI

During the preparation of this work, the author(s) used DeepL and ChatGPT in order to: Grammar and spelling check and improve writing style. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

The presented platform uses ChatGPT to implement the interactive natural language explanations. Examples generated in this work are clearly labelled as such.

References

[1] D. Smith, Planning as an iterative process, in: Proceedings of AAAI Conference on Artificial Intelligence, volume 26, 2012, pp. 2180–2185.

[2] R. Eifler, M. Cashmore, J. Hoffmann, D. Magazzeni, M. Steinmetz, A new approach to plan-space explanation: Analyzing plan-property dependencies in oversubscription planning, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 9818–9826.

[3] G. Fouilhé, R. Eifler, S. Thiebaux, N. Asher, Conversational goal-conflict explanations in planning via multi-agent LLMs, in: AAAI 2025 Workshop LM4Plan, 2025.

[4] A. Belov, I. Lynce, J. Marques-Silva, Towards efficient MUS extraction, AI Commun. 25 (2012) 97–116.

[5] E. Gamba, B. Bogaerts, T. Guns, Efficiently explaining csps with unsatisfiable subset optimization, Journal of Artificial Intelligence Research 78 (2023) 709–746.

[6] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial Intelligence 267 (2019) 1–38.

[7] G. De Giacomo, R. De Masellis, M. Montali, Reasoning on LTL on finite traces: Insensitivity to infiniteness, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 28, 2014.

[8] F. Fuggitti, T. Chakraborti, NL2LTL–a python package for converting natural language (nl) instructions to linear temporal logic (ltl) formulas, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, 2023, pp. 16428–16430.

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

[10] M. Cosler, C. Hahn, D. Mendoza, F. Schmitt, C. Trippel, nl2spec: Interactively translating unstructured natural language to temporal logics with large language models, in: International Conference on Computer Aided Verification, Springer, 2023, pp. 383–396.

[11] G. Vilone, L. Longo, A novel human-centred evaluation approach and an argument-based method for explainable artificial intelligence, in: International Conference on Artificial Intelligence Applications and Innovations, 2022, pp. 447–460.

[12] D. Slack, S. Krishna, H. Lakkaraju, S. Singh, Explaining machine learning models with interactive natural language conversations using TalkToModel, Nature Machine Intelligence 5 (2023) 873–883.

[13] V. B. Nguyen, J. Schlötterer, C. Seifert, From black boxes to conversations: Incorporating XAI in a conversational agent, in: World Conference on Explainable Artificial Intelligence, 2023, pp. 71–96.

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

[15] N. Feldhus, Q. Wang, T. Anikina, S. Chopra, C. Oguz, S. Möller, InterroLang: Exploring NLP models and datasets through dialogue-based explanations, in: Findings of the Association for Computational Linguistics, 2023, pp. 5399–5421.

[16] S. L. Vasileiou, W. Yeoh, Trace-cs: A synergistic approach to explainable course scheduling using llms and logic, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 39, 2025, pp. 29706–29708.

[17] M. Helmert, The fast downward planning system, Journal of Artificial Intelligence Research 26 (2006) 191–246.

[18] R. Eifler, M. Steinmetz, A. Torralba, J. Hoffmann, Plan-space explanation via plan-property dependencies: Faster algorithms & more powerful properties, in: Proceedings of the 29th International Conference on International Joint Conferences on Artificial Intelligence, 2021, pp. 4091–4097.

[19] B. Krarup, S. Krivic, D. Magazzeni, D. Long, M. Cashmore, D. E. Smith, Contrastive explanations of plans through model restrictions, Journal of Artificial Intelligence Research 72 (2021) 533–612.

[20] S. Sreedharan, A. Kulkarni, S. Kambhampati, Explanation as model reconciliation, in: Explainable Human-AI Interaction: A Planning Perspective, Springer, 2022, pp. 59–80.