Dialogue planning model for simulation puzzles

Donghao Wang[1]

School of Computing, Queen's University, Kingston, Canada

Abstract

Generating goal-oriented dialogue agents has been a popular market idea in the chatbot industry. Aside from the growing demand for dialogue agents for business processing and customer service, there is potential for increasing demand for entertainment-purpose chatbots.

For this short-term project, I propose an automated planning model for dialogue agents used for simulation puzzles. A simulation puzzle is a one-to-one game that involves one player describing the start and ending of an event, and the other player constructs the full scope of this event by asking a series of questions. The proposed planning model is a declarative representation of the simulation puzzle dialogue agent written in Planning Domain Definition Languages. The approach described in this paper focuses on the modeling process from the model domains to the generated plans by instances.

1. Introduction

Generating useful dialogue agents has been a challenge in the academic field and a rapidly growing demand for the industry. In particular, a variety of artificial intelligence (AI) techniques can be applied to this area. In this paper, the simulation puzzle dialogue agent is constructed by automated planning models using Planning Domain Definition Language(PDDL) and Relational Dynamic Influence Diagram Language(RDDL).

Automated planning is a subfield of artificial intelligence (AI) that involves developing algorithms and software systems that can generate plans to achieve a specific goal or set of goals. The goal stands for completing a task or solving a problem, and the plan is a set of actions that are designed to achieve that goal. In this context, the goal for the dialogue agent is to draw the full scope of the story using the provided start and end.

The first challenge for this project is to find out the most suitable state and action representation for the simulation puzzle problem. Automated planning involves several steps, including modeling the problem, generating potential solutions, and selecting the optimal solution. The process often involves creating a formal representation of the problem and generating plans using a specific heuristic. The initial idea is to represent the state of the story through five elements: when(time), who(character), conflict (why), where, and how(the emotional states of the characters). The action of the dialogue agent is to ask the user for information related to the five elements of the story and generate the story after accumulating the five elements of the story.

The second challenge for the project is whether the state transition is going to be probabilistic. The simulation puzzle is a fully observable, non-deterministic (FOND) problem, but the state transition could be probabilistic, which might upgrade to the MDP (markovian decision process) problem. To deal with such variance, the FOND model is written in PDDL, and the MDP model is written in RDDL. RDDL deals with the model better in terms of a probabilistic situation.

One important aspect of the languages used for writing the model planner(pddl and rddl) is that pddl and rddl cannot simulate the natural language processing for reading user input on the command line. The approach described in this report only describes the heuristic of the dialogue agent.

2. Background

2.1. FOND and FOP Planning

For dialogue planning models, the problem is assumed to be Fully Observable Non-deterministic(FOND) or Fully Observable Probabilistic (FOP). The fully observable setting stands for the outcome of each action in the state space is fully observed by the agent, and the non-deterministic/probabilistic part stands for whether the state transition is random or defined by a probability.

Definition 1(Planning problem) A planning problem <F,I,A,G> consists of four components. F:Fluents defines the state of the problem using boolean representations(true or false); the initial state $I \subseteq F$ is where the agent starts execution in the problem; A: Action describes what the agent can d; the goal $G \subseteq F$ specified to be achieved when the problem is solved. A complete state (or just state) is a subset of the fluents F that are presumed to be true (all others presumed to be false);

CISC-813: Automated Planning – Final Project

a partial state is similarly defined but without any presumption about the truth of fluents outside of the set. [1]

Definition 2(Actions) An action $a \in A$ in the planning problem consists of 2 components: The precondition and the effect. The precondition should be the set of fluents $f \in F$ that has to be held true for the action to be executed, and the effect is the set of fluents $f \in F$ defined to be true or false after the action has been executed.

In the FOND dialogue planning problem, the effect of an action could be defined either as a nested set of conjunction or disjunction of fluents or negated fluents. For FOP problem, the definition of the action effect is the same but the outcome is determined by a defined probability distribution.

Definition 3(Contigent Plan) C.Muise et.al.[1] defines the solution to a FOND planning problem as a Noded graph <N,ϵ,n_0>. N corresponds to the action of the agent in the form of nodes in the graph, ϵ are the edges corresponding to the possible outcomes of each action associated with the nodes. n_0 is the start node equivalent to the initial state I, where the agent starts executing the actions.

The contingent plan is a formalism for connecting the generated plan and the execution in terms of a decision tree map. In this project, the contingent plan will be shown in the form of a decision tree graph on the following pages.

2.2. Epistemic Planning

Epistemic planning is a relatively more complex concept than FOND and FOP planning. The aim of epistemic planning is to generate plans that are robust to changes in the agents' knowledge and beliefs, and that achieve the desired goals despite uncertainty. Epistemic planning involves modeling and reasoning about epistemic states, which represent the agents' knowledge and beliefs at different points in time, and epistemic actions, which change the agents' knowledge and beliefs.
In general, the planning involves multiple agents, and each agent has their own Belief state. The actions defined may change the Belief states of the agent, and the agent's belief state may affect their decision on what action to choose.
Definition 4(Epistemic task)[2] An epistemic planning task on (P, A) is (A, s_0, ϕ_g) where A is a finite set of epistemic actions on (P, A), s_0 is an epistemic state on (P, A), and $\phi_g \in L_K C(P, A)$. We say that (A, s_0, ϕ_g) is an epistemic planning task for agent $i \in A$ if s_0 and all $a \in A$ are local for i. A planning task (A, s_0, ϕ_g) is called global if s_0 is global. Given any planning task $\pi = (A, s_0, \phi_g)$, the associated local planning task of agent i, denoted π_i , is $s(a^i | a \in A, s_0^i, \phi_g)$. In the FOND dialogue planning problem, the effect of an action could be defined either as a nested set of conjunction or disjunction of fluents or negated fluents. For FOP problem, the definition of the action effect is the same but the outcome is determined by a defined probability distribution.

2.3. Simulation puzzles

The term simulation puzzle refers to a conversational game based on a defined constraint of topics. The game involves two participants. The first participant(story maker) will think of a story/event and provide the start and result/end of the story. The second participant (questioner) needs to ask a series of questions to the first participant about the details of the story, and then make the guess. The questions have to and can only be answered with yes or no.

In this project, the agent is playing the part of the questioner and the user(human participant) is playing the part of the storyteller.

3. Approach

3.1. Planning models for simulation puzzles

A dialogue agent or chatbot is an automatic response system that responds to the user input with appropriate output. The state transition between actions and states would be needed to be considered for specific cases of Planning domains.

In this case, the initial action representation is a single action that asks questions about a single fluent missing in the common knowledge base. The effect would be changing the state of the fluent(true or false) in the state space.

The state representation, as explained before, is the five elements of a story: when(time), who(character), conflict (why), where, and how(the emotional states of the characters). For each simulation puzzle problem, the ultimate goal is the same: accumulate the five missing elements and construct the story.
The action-state transition between the actions and states is represented in the diagram shown below:

The Current State of the domain is a collection of fluents.

Figure 1: State-action transition

The ask action will move the state of the domain to the next state. If the agent receives a response giving valid information, which means a new fluent is added to the domain. If the user response didn't give valid information, the agent will not be able to update the fluent base thus it remains in the current state, The ask action can be repeated to get to the goal state.

The problem setting could have extended the problem to three approaches: FOP, FOND, and Epistemic. For this case, the epistemic planning case is no longer needed since the problem only involves a single agent that receives information from the outside world. Although the user can be considered as another agent that has his own beliefs, the user's beliefs actually represent the common knowledge in the world. The agent's knowledge base is constantly being updated and the user response doesn't involve deceptive or ambiguous responses. thus there is no need for the dialogue agent to "reason" facts based on its belief.

The project uses two approaches to model this problem: FOP, FOND. At the start, the FOP model is implemented using the RDDL planner. However, there are certain limits in RDDL planner in reflecting the strong cyclic plan generated for the task. In later steps, the FOND model is implemented using the PDDL planner. The following section will describe each of the approaches in terms of the problem setting and actual implementation.
The initial state of the problem should have one or two fluents given, to reflect that the user provides the start and end of the story at the beginning of the puzzle. The goal state of the problem should have all 5 fluents: when, who, where, how, and conflict.

3.2. FOP planning

Different from pddl, Rddl defines the actions and states in the form of fluents. This model, it involves a single object story-component, one state fluent complete() and one action fluent question(). The state fluent complete() states if the corresponding story-component has been completed. The action fluent question() fluent turns true if the story component has been asked.
In the simulation puzzle involving a human and an AI agent, for each question, the AI agent asks, there is a probability to get a yes or no response that depends on the states that turn true. The probability to complete each component is set to 50% at each problem set and in the problem definition the variable is called PROB-COMPLETE(storycomponent), this probability can be adjusted. The formula for completing a story component is listed below:
$P(storycomponent) + (1 - P(storycomponent)) * \Sigma_{k=storycomponent|complete(k)}/(1 + \Sigma_{k=storycomponent})$
The formula states that the more story components are acquired, there is higher probability to get the correct information for the missing story components.
The state fluents and action fluents are associated with a reward function that is defined by the REWARD-COMPLETE-COMP and COST-QUSESTION variables, by adjusting the reward during the experiment, the model would show different behaviors.
The RDDL model implemented is a rather trivial model for representing the state of the problem. One reason that the project didn't push the RDDL model further is that the current RDDL planner doesn't draw a clear state transition diagram for a specific task. Another reason is that for the simulation puzzle, whether the response reflects the correct information is dependent on the question the AI agent asks, which cannot be correctly formulated by defining a constant probability.

3.3. FOND planning

The FOND planning model is implemented in PDDL. PDDL planning models involve two major components: predicates and actions. The predicates are factual shared in the common knowledge base, and actions are used for changing the status of the predicates. The goal of the model is fixed to acquire the five elements of the story.
The initial implementation of FOND planning models involves two predicates: (maketrue ?x), (uncertain ?x). The variables in the predicates are all objects, referring to the 5 elements of the story. (maketrue ?x) represents that factuals about story element x are acquired by the agent, (uncertain ?x) states that the facts about story element x are still not clear. The model has two actions: (ask ?x) and (askagain ?x). (ask ?x) is invoked when there is no factual acquired for the story element x and (askagain ?x) is invoked when the story component ?x is uncertain. Each action uses an oneof statement to reflect the non-deterministic nature of this model: both of the actions has an effect to either return (maketrue ?x) or (uncertain ?x). The major difference is that the (askagain ?x) will only be invoked when there is an (uncertain ?x)

present in the common knowledge base.
The initial model is a fairly trivial implementation with a fairly obvious solution. In the simulation puzzles, the communication between user and AI would be more complex, thus in further implementation, more things is added to the model.
The further improvement of the model adds two more elements into the predicate, (overlap ?x ?y). (overlap ?x ?y) states whether there is overlapped information between two story components, To reflect the changes in the two new predicates, a new action (deduct_overlap ?x ?y) is introduced. The action (deduct_overlap ?x ?y) has a precondition that requires element ?x to be true and element ?y not in the knowledge base, the effect of the action could lead to either (uncertain ?y) or (maketrue ?y).
The improvement reflects the self-reasoning of the AI agent towards the goal according to the global knowledge base knowledge. However, in the actual experimentation, the models would still produce a strong cyclic plan that only involves the use of ask and askagain action, which is not the normal case for an AI agent that is capable of self-reasoning.
The final model fixes this issue by adding a variable (total-cost) and adding another two predicate (conflict ?x ?y) and (done) with two actions (conflict_resolve ?x ?y) and (solve).
The function (total-cost) stands for the total-cost of the action and each action has different action costs.The evaluation metric of the planner is to reach the goal with minimum total-cost. As the ask-and-ask-again actions are assigned with higher costs, the planner would tend to choose actions with lower costs.
For the new predicates and actions, (conflict ?x ?y) shows that there is a conflict between two factual, and (done) refers to the planner having accumulated all 5 story components without conflicts. The action (conflict_resolve ?x ?y) removes the conflict and makes one of the story components in the conflict uncertain. The action (solve) represents that the AI agent uses the acquired 5 story components to produce a valid story, which has an effect to produce (done).

3.4. Approach conclusion

Overall, the project chose the FOND approach to finalize as the final model of the project. PDDL plays a better part in simulating the simulation puzzle in non-deterministic state-transitions without specifying the probability. Both of the approaches are evaluated through the prpviz(pddl) and viz(rddl) commands, which is illustrated in the evaluation section.

4. Evaluation

4.1. FOP:RDDL

As the initial model is only a prototype model with the most simplified actions and state representation, the testing instance given is only a general case .rddl file containing the initial state as start and end given.
The instance.rddl file contains the non-fluent setting for the rddl model. In the actual implementation, the conflict story component is replaced by emotion1 and emotion2 component, to introduce more diversity in the plan generation. THe probabiity for completing the emotion1 and why story component is set to 0.3 and 0.2 respectively to reflect the hardness of extracting certain story components.
The RDDL planner doesn't have a goal state defined in the instance.rddl file, instead, the planner searches for a heuristic that reaches the maximum reward. Thus, the penlaty given for each question asked and the reward given for each component completed matters in terms of the length of actions.
For the default setting the longest series of actions can go over to 16 among 30 trials for rddl planning, as shown in Fig2. The reward setting is 100 for completing a story component and -10 for asking a question.
From fig2, we can see that the emotion1 and why story component is completed the last among all the story components. Since these two components have lower chances of returning positive results, it is a common choice for planners to do so. The shortest action length is 7 actions in total, which also follows this pattern.
An experiment is done to verify the effect of reward setting on the performance of models. The penalty of giving a question is fixed at 10, but respectively,the instance files contain different reward setting for completing a story component as 10, 50, and 100. For each instance setting, the same domain is used for generating the model. Each domain-instance pair is used for generating 30 plans.

Reward-complete	10	50	100
Action-length	5-7	5-12	6-16

As the table shows, the larger the difference between the reward and penalty, the larger the range of action length. In this case of the model, it is better to set the reward for completing a component the same as the penalty for asking a question.
For execution speed, the plan executes within a range of 5 second for each plan.

4.2. FOND:PDDL

As described in the approach, the PDDL model has iterated for three times to reach the final model. The

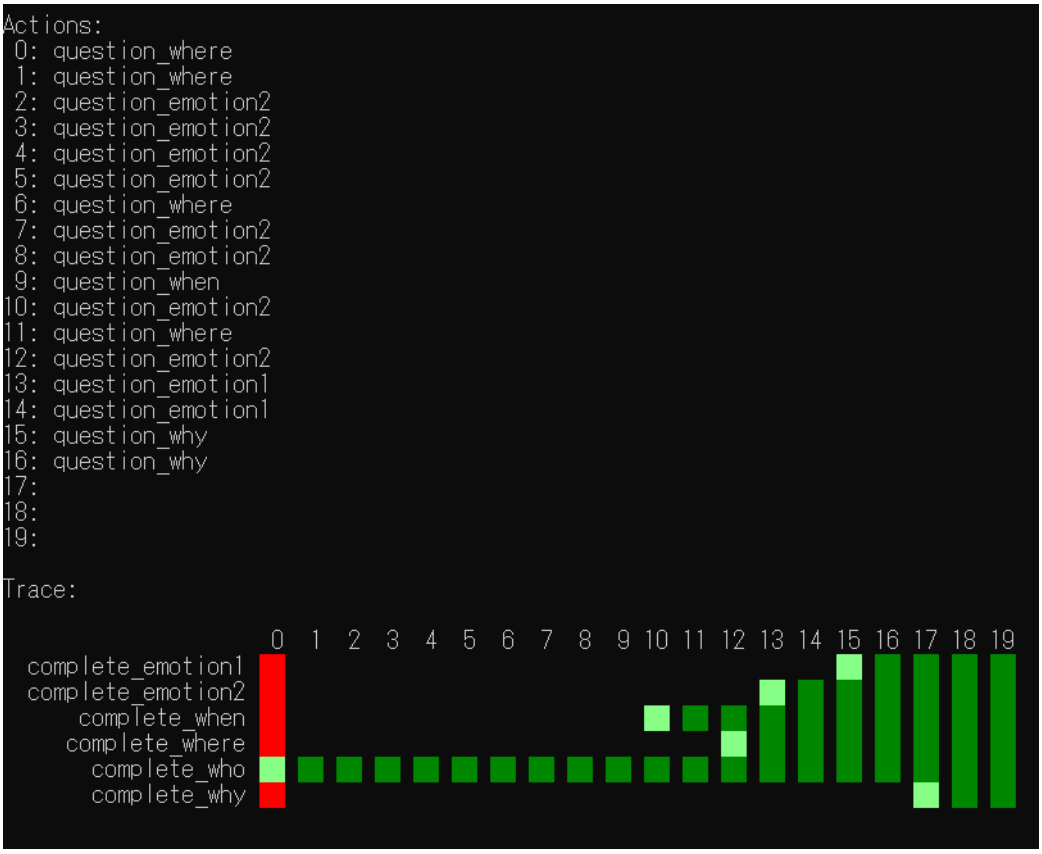

Figure 2: Longest action from 30 trials FOP planning RDDL

domain file defines the 5 story components: who, when(time),where,why and conflict.
The initial model is a trivial model consists of only the ask and askagain action, the result of the execution is illustradted by the image below. The domain files gives the maketrue(who) statement.
Fig3 reveals a fairly simple strong cyclic plan that asks for the story component one by one. For a real-world scenario, the communication between the user and the agent is based on this plan but the agent here lacks ability to self-reason the factuals.
The intermediate model adds an overlap statement to allow the agent to deduct missing information from the given information. In this case, the test problem adds the (overlap conflict why) (overlap why conflict) (overlap who why) fluents in the initial setting, and produce the plan as shown in Fig4. Finally, after adding more actions and statements, the planner generates a plan according to the final model in the approach. The final plan shows a rather complicated circle for reasoning about the facts given and acquiring necessary information. The model generates a fairly realistic plan for the AI agent on a real-world basis proving that the agent in the plan is capable of reasoning the facts in the world and acquiring necessary information related to the story content.
The average time for generating a plan for the final model in the PDDL planner is 9 seconds.

4.3. Evaluation conclusion

The plan generated by the final model illustrates the decision process of an agent that is capable of reasoning facts and acquiring facts. The average running time to generate a plan for a scenario with limited given information is within 10 seconds and the level of state transition is within 20 levels, it can be concluded that the model provides a concrete plan for simulation puzzle solvers.

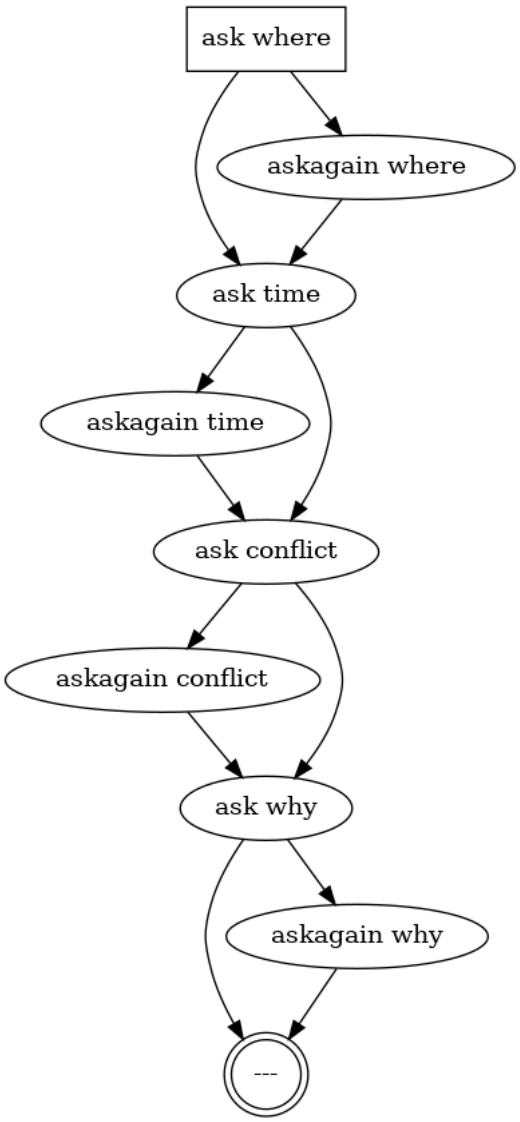

Figure 3: plan for the initial model

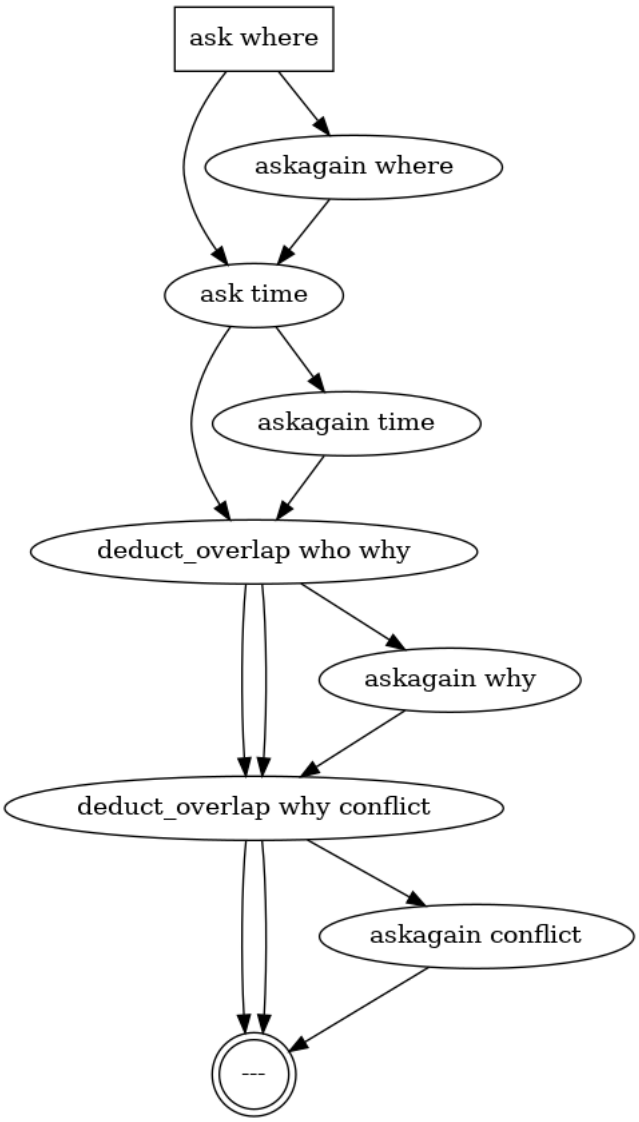

Figure 4: Generated plan for the intermediate model

5. Related Work

5.1. Planning for Goal-Oriented Dialogue Systems[1]

In this paper, C.Muise et.al. proposes a novel approach for developing dialogue systems that can satisfy the goals for business communication such as customer services. The proposed approach is based on automated planning techniques using PDDL, which enable the system to reason about a user's goal and generate a sequence of actions that will lead to the desired outcome.

The paper presents an overview of the goal-oriented dialogue system framework based on automated planning, which consists of several components, including a natural language understanding module, a dialogue manager, a plan generation module, and a natural language generation module. The system's architecture is designed to enable seamless integration of each component, allowing for efficient and effective communication with users. The paper mainly focuses on the planning model section.

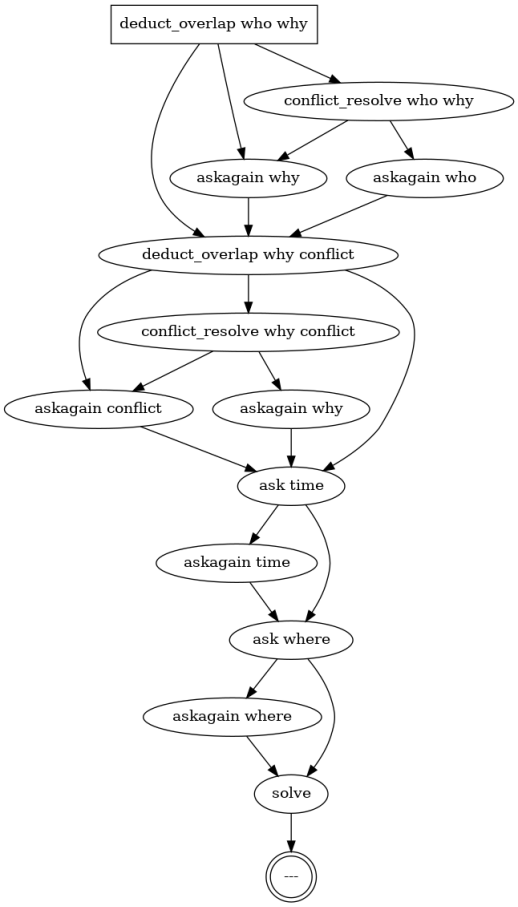

Figure 5: Generated plan for the final model

One of the key advantages of the planning-based approach is its ability to handle complex and dynamic user goals. The system can reason about the user's goal and generate a plan that accounts for different contingencies and potential obstacles. The approach also allows for the incorporation of user preferences and constraints, enabling the system to generate personalized plans that reflect the user's individual needs and preferences.

5.2. A Gentle Introduction to Epistemic Planning: The DEL Approach[2]

Bolander[2] introduces the introduced the DEL approach to epistemic planning, which is a planning framework that takes into account the uncertainty of the world and the knowledge of the agents that operate within it. The paper provides a comprehensive overview of epistemic planning and how it differs from traditional planning.
The DEL language consists of three components: the Dynamic component, which describes how the world changes over time; the Epistemic component, which describes agents' knowledge and beliefs about the world; and the Logic component, which is used to reason about the knowledge and beliefs.
The paper explained the difference between epistemic planning and Classic planning in terms of Epistemic logic. An example of epistemic logic can be represented by an epistemic language that is generated by the BNF below:

$$\varphi ::= \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid C\varphi,$$

Figure 6: Epistemic logic

The character ϕ represents a piece of knowledge in the knowledge base, $K_i\phi$ represents that the knowledge is known to agent K, $C\phi$ represents that the knowledge is common knowledge to all the agents.
The advantage of epistemic planning is that it allows agents to reason about uncertain and dynamic environments, taking into account their own knowledge and beliefs as well as the knowledge and beliefs of other agents in multi-agent environments. It is useful in scenarios where the agents must reason about their environment and their own actions in order to accomplish tasks.

5.3. Report about Akinator[3]

The paper titled "Report about Akinator" provides a comprehensive overview of the popular online game Akinator, which is a program that tries to guess a real or fictional character that a player is thinking of based on a series of questions that can be answered with yes or no.

The architecture of Akinator is based on a decision tree structure that contains thousands of nodes, each representing a potential question that can be asked. The system traverses this decision tree based on the player's responses to the questions, narrowing down the set of possible characters until it arrives at a final guess.The paper also explores the different strategies used by the Akinator system to improve the accuracy of its guesses, such as the Support Vector Machine(SVM) applied in the node classification process.

The decision tree method relies on splitting existing data nodes and classifying the nodes based on their labels. One of the most significant advantages of the decision tree is that it is interpretable by showing the decision tree in an interpretable manner. However, a decision

tree can be time-consuming in the training phase and extra effort needs to be done on feature reduction and resampling if facing a complex, unbalanced dataset.

Compare to the planning-based approach by C.Muise et. al.[1], the decision tree model has a longer training period and better interpretability, as well as better performance in locating the target node. For Akinator, the goal is fixed on guessing the character the player is thinking of, thus using the decision tree method can help locate the most possible target node representing the possible characters. For the simulation puzzle, the target is to reconstruct the story based on the provided start and end, which means there might be various versions of the story that is plausible in the content. There are certainly too many features/goals for the decision tree method, thus in this case, the planning approach would perform better in terms of finding the best heuristic.

5.4. Automated Planning of Simple Persuasion Dialogues[4]

E.Black et.al.[4] describe an automated planning approach in producing simple persuasion dialogues. The general approach is to convince the responder to accept the topic of the dialogue by asserting beliefs.

In the planning model explained in this paper, the persuasion situation is described by two participants' dialogues(the persuader and the responder). In each step of the state space, the persuader inserts subsets of its beliefs into the common knowledge base and the responder responds with yes or no to indicate whether it has been persuaded to recognize the current common knowledge base.

In the final discussion, the author points out that the persuader may have some preferences regarding the beliefs it shares with the responder. As the model only considers the success of the dialogue, the situation might be that the persuader gets agreed with the responder's belief. The solution to this problem is to assign values to each belief the persuader shares with the persuaded, thus the optimization metric also takes into account the beliefs the persuader has had to share.

The simulation puzzle planning model uses similar settings to the persuader model. The difference between the two models is that the simulation puzzle model only has one machine agent, and the common base knowledge is entirely up to the questions the machine agent asks.

6. Summary

The project explored the possibility of using FOND and FOP planning to build up a plan for simulation puzzles. The final choice is to use FOND planning in a PDDL planner, due to the PDDL planner's function in generating the final plans and the non-deterministic nature of the simulation puzzles. The final model and the generated plan show that an AI agent following this plan would have the ability to acquire real-world knowledge and reasoning about missing facts.

6.1. Future Work

Due to the limitation of PDDL planning language, the final model is not capable of reflecting the natural language processing part in the actual implementation of a chatbot, which could be considered as a future development of the project.
Except for introducing natural language processing, an interesting evolution of this project would be to have agents have the story and the user guesses. In this scenario, the agent would need to give an appropriate answer to the user's question and give hints to the user in order to let the user figure out the story in a number of steps. This might involve using epistemic planning since it would involve self-reasoning and deceptive information.

References

[1] C. Muise, T. Chakraborti, S. Agarwal, O. Bajgar, A. Chaudhary, L. A. Lastras-Montaño, J. Ondrej, M. Vodolán, C. Wiecha, Planning for goal-oriented dialogue systems, CoRR abs/1910.08137 (2019). URL: http://arxiv.org/abs/1910.08137. arXiv:1910.08137.

[2] T. Bolander, A gentle introduction to epistemic planning: The DEL approach, Electronic Proceedings in Theoretical Computer Science 243 (2017) 1–22. URL: https://doi.org/10.4204%2Feptcs.243.1. doi:10.4204/eptcs.243.1.

[3] N. Karour, Report about Akinator, 2014. URL: https://www.academia.edu/6233778/Report_about_Akinator.

[4] E. Black, A. Coles, S. Bernardini, Automated Planning of Simple Persuasion Dialogues, Lecture Notes in Computer Science (2014). doi:10.1007/978-3-319-09764-0_6.