Want To Choose Your Own Adventure? Then First Make a Plan.

Nisha Simon[†,*], Christian Muise[†],
[†] Queen's University, Canada

Abstract

Our research demonstrates that Large Language Models (LLMs) can successfully be used for automated story generation provided that they are first given a valid input plan that has been generated by an automated planner. A well-known issue with LLMs is that while they can generate coherent outputs over short spans of text, they lose coherence when they are asked to produce longer text narratives. In this study, we combine the fields of Automated Planning and Text Generation to show how text-based interactive '*Choose Your Own Adventure*' (CYOA) stories can be created using LLMs in conjunction with Fully Observable Non-Deterministic (FOND) based Automated Planning.

Keywords: Large Language Models, Automated Storytelling, Automated Text Generation, Automated Planning, Interactive Narratives.

1. Introduction

'*Choose-Your-Own-Adventure*' (CYOA) stories are interactive narratives where the reader assumes a fictional persona and then chooses various paths in order to reach the ending of a story [1]. If the reader makes the right choices, they are rewarded with a 'happy' ending, such as winning a mountain of treasure; if not, they could end up severely disappointed, such as finding themselves trapped in a large dark pit. '*Choose-Your-Own-Adventure*' stories can, therefore, be thought of as a representation of an agent proceeding through a non-deterministic environment in order to achieve a certain goal. In our study, the agent is the story itself, and the non-determinism is driven by the reader. The non-deterministic nature means that an agent's actions may have an impact on the world that is not known until the time when the action is executed. The '*Fully Observable*' designation means that all parts of the environment are known by the agent (i.e. no parts of the environment are hidden from the agent), and only the final impact of each action is uncertain. Thus, CYOA stories can be created using LLMs in conjunction with Fully Observable Non-Deterministic (FOND) based Automated Planning.

In our running example, the hero of the tale is a brave adventurer named Jack, who is navigating the intricacies of an ancient castle. The user takes on the persona of Jack and makes decisions about what action to take at each stage of the story. As shown in Figure 1, if the user chooses a '*good*' path, Jack escapes from the castle and is rewarded with treasure and glory. If the user makes the '*wrong*' choices, Jack could meet with an untimely demise. Note that the '*good*' and '*wrong*' choices are based only on the perspective of the main character. The valid plan that the automated planner generates is simply concerned with reaching the goal state from the starting state, within the confines of the given environment, and therefore it makes no distinction as to whether the '*game over*' state is reached in a '*positive*' or '*negative*' way for the main character.

Our main contributions are that we demonstrate how Fully Observable Non-Deterministic (FOND) Automated Planning can be used to drive LLMs to generate text that can be used in CYOA stories.

Background: Automated Planning problems are represented using the Planning Domain Definition Language (PDDL) [2]. Planning problems use two files written in PDDL format: the *Domain* file and the *Problem* file. The Domain file contains the *requirements, types,*

[*] nisha.simon@queensu.ca

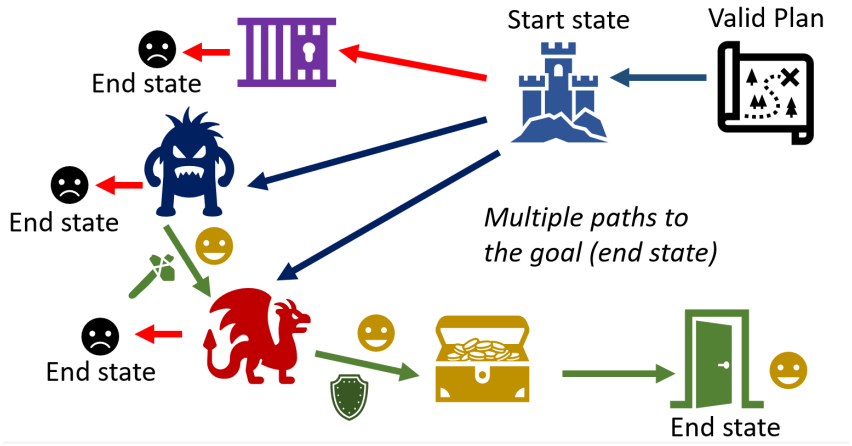

Figure 1. Example of various possible paths through a Choose Your Own Adventure (CYOA) story

predicates and *actions*, while the Problem file holds the *objects*, the *initial state* and the *goal*. A particular domain could have multiple problems associated with it. The domain and problem files are fed into an automated planner, and the planner then produces a *plan* (typically represented by a sequence of actions or steps) that lead to the goal state.

We represent a classical planning problem P as a tuple denoted by $\langle F, A, I, G \rangle$. **F** is the *set of fluents* or items that can be either TRUE or FALSE in the domain. **A** is the *set of actions* or what the agent is allowed to do in the given environment. **I** is the initial state. **G** is the goal the agent is trying to achieve or the *set of fluents* that must be TRUE at the end of the planning process. An action **a** in the set of actions **A** has three characteristics: PRE(a): the preconditions of action **a** or the set of fluents that must hold to execute action **a**, DEL(a): the set of fluents that are removed from the current state when action **a** is executed, or the '*delete effects*' of action **a**, and ADD(a): the set of fluents that are added to the current state when action **a** is executed, or the '*add effects*' of action **a**. If PRE(a) $\subseteq$ s, the agent can take action **a**. We *progress* from a state ***s*** to state s' using action **a** by removing every fluent that **a** deletes, and then adding every fluent that **a** adds. That is to say *Progress* (s,a) = (s $\setminus$ DEL(a)) $\cup$ ADD(a). The goal is achieved when G $\subseteq$ s. In FOND planning, we extend classical planning to allow actions to have more than one outcome, thus potentially leading to more than one successor state at execution time. The keyword '*oneof*' indicates multiple possible effects. By including non-determinism, the plan now takes the form of a decision tree instead of a sequence of actions [2].

2. Related Work

The issue of LLMs losing coherence over longer text generations has been problematic even for the largest and most powerful LLMs [3]. Yang, Klein, Peng, and Tian state that '*It is nontrivial to maintain overarching coherence or even basic relevance to an initial premise or plan.*'. GPT-based systems like '*Stories by AI*' can create simple narratives. However, the authors conclude that "*if you don't steer the story, the AI tends to meander around in circles*" and "*without steering, the AI suggestions tend to become circuitous and repetitive*" which means that there always needs to be a human in the loop [4].

Neurosymbolic methods have previously been used to provide a logical reasoning system to guide LLMs to overcome this issue [5]. Directed graphs and branching story trees have been used to direct players through interactive text-based games, as demonstrated by Yu and

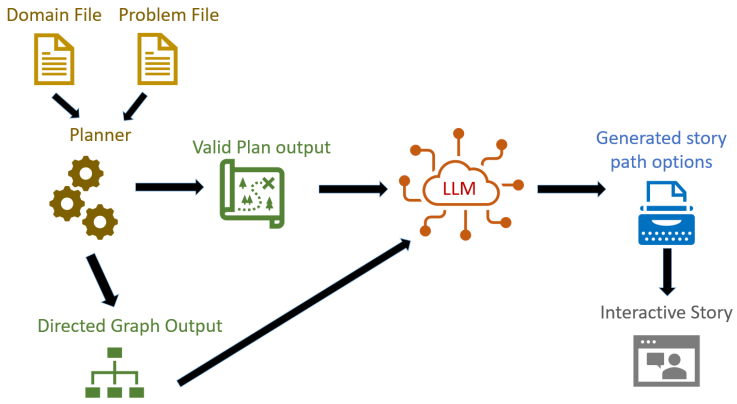

Figure 2. High-level System Architecture Diagram. A Domain file and one (or more) Problem file(s) in PDDL are fed into an automated planner in order to produce a valid plan. The outputs of the plan are in the form of text files as well as a directed graph. The outputs of the plan are then fed into an LLM to generate natural language storylines, which are then used to guide a player through an interactive text-based adventure.

Riedl [6]. Kelly, Calderwood, Wardrip-Fruin, and Mateas developed a system to create plans from Natural Language text but conceded that the complexity of the PDDL in terms of the number of predicates and parameters used, as well as the complexity of the generated stories, is low [7]. Their system also only creates a valid plan 34% of the time. Indeed as noted by Valmeekam, Olmo, Sreedharan, and Kambhampati, LLMs '*even in simple common-sense planning domains, LLMs seem to display subpar performance*' [8]. Planning and long-range reasoning, therefore, are not the LLM's forte [9]. Clark and Smith et al. have created a system that provides the user with pairwise suggestions for story writing, but their work is designed more for the evaluation of the actual model [10].

Simon and Muise have previously used classical planning with LLMs for story generation in a FOD (Fully Observable Deterministic) format [11]. We extend this approach by using FOND planning to generate the storylines. While FOD planning provides a set story with no choices for the reader to make, FOND stories allow the reader to create their own custom adventure each time.

3. Methodology

The overall architecture of our system is shown in Figure 2. The plan representations, i.e. the Domain and Problem files, were manually created by the authors. However, it is also possible to build domains from previously created stories. The Domain and Problem files for the initial story were manually modelled and written in PDDL and then fed in as inputs to an automated FOND planner (in this case, an extension to the *PRP* planner) [12]. The PRP planner extension is an off-the-shelf, state-of-the-art planner that was used as a component of our system. The particular planner is not a contribution of our paper since other FOND planners could also be used instead of PRP. A small snippet of sample PDDL code from the Domain file can be seen in Figure 3. The FOND planner produces output in the form of multiple text files and a directed graph such as the one in Figure 4 that shows the various paths that lead from the starting state to the end (goal) state, or the '*game over*' state.

After computing the plan, we feed the plan outputs to the LLM and then obtain the corresponding next Natural Language line, one at a time, as follows: First, we include any background or real-world contextual information as the initial or '*hidden*' (i.e. not always

```
(:action pick_weapon
  :parameters(?l1 - location
              ?b1 - hero)
  :precondition (and (at ?b1 ?l1)
    (not(have_weapon))
    (not(is_dead ?b1)))
  :effect
    (and (at ?b1 ?l1)(have_weapon)
    (labeled-oneof get_weapon
    (outcome shield
      (and
       (have_shield)(picked_weapon
         shield)
       (not (have_mace))
       (not (have_axe))
       )
    )
    (outcome mace
      ...
))
```

Figure 3. PDDL code snippet from the Domain file.

directly related to the specific actions in the story) input prompts (also called *story-agnostic* prompts) to the LLM. The manually generated story-agnostic prompts provide background or common-sense information that guides the rest of the output and can be common to multiple stories. The story-agnostic prompts are also useful for style purposes, such as how the story is written. The LLM requires at least two (or more) initial patterns of '*action*' and '*story*' prompts in order to recognize and generate the text. The '*pattern*' is the combination of the story-agnostic prompts, as well as the initial inputs. Second, we took one line at a time, in sequence, of the solved plan. Third, we appended this line to the hidden prompt for the LLM input. Fourth, we let the LLM generate one line of output (ignoring any further lines that were generated). Fifth, taking this one line of generated output, we added it to the end of the existing input prompt and used the result as the new input prompt.

The planner outputs are consolidated into a single text file and then combined with hidden prompts to be fed into an LLM (specifically *gpt2-x1* [13]) via the *Hugging Face API* [1] to generate user options for the interactive story. The hyperparameters used were as follows: Only one return sequence was requested at a time for each input line. The maximum number of new tokens was set to 25 to allow for fairly descriptive storylines while still staying faithful to the inputs. The temperature of the sampling operation, which can range from 0 to 100, was set to 0.7, where 100 is close to uniform probability and 0 means take the highest score. Sampling was turned on, and Top-k was set so that only the top 10 tokens returned were used to generate the new text. The specific LLM that is used is not the main point of the study, as the Hugging Face API provides access to a large variety of LLMs, and the LLM can be considered merely a 'black box' component of the overall system that can be switched out at will. We do not directly use fine-tuning of the LLM. Instead, we allow the LLM to predict the outputs based on input prompts that are given to

[1] https://huggingface.co/models

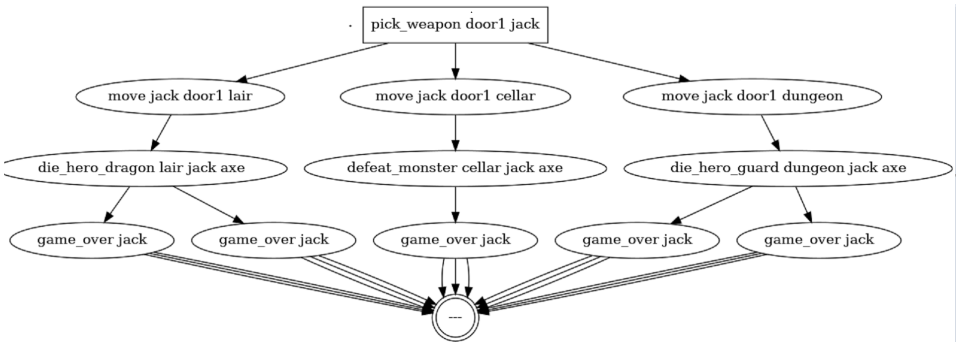

Figure 4. Example of a directed graph that is produced by the automated planner as part of the output

it at run-time. The output of the overall task is therefore the complete story that is created based on the choices made by the user.

The user is not privy to the details of the generated plan i.e., the user is unaware of which paths are '*good*' and which paths are '*wrong*' paths within the story. Instead, the user '*plays*' the text-based game by selecting their choices for each stage of the story through an interactive Google Colab notebook whose logic is based on the directed graph and on the LLM outputs, plus some additional manually written text. The user's input is merely an instruction to the system to follow a specific edge on the directed graph. The entire plan representation itself already exists in totality and is not updated based on the user choices.

4. **Results**

We find as a proof of concept that the LLM can successfully generate text story options at each stage of the narrative for the user to progress logically from the starting state to the end or '*game over*' state. For example, from the valid plan file, the action '*die-hero-guard dungeon jack axe*' is correctly translated by the LLM to '*Jack is attacked by a guard at the entrance to the dungeon.*' and '*defeat-monster cellar jack axe*' is translated correctly to '*Jack defeats the monster in the cellar.*', while even the very simple action line, '*game-over jack*', is converted to *'Jack's adventure has ended.*'. Examples of gameplay can be seen in Figures 5 and 6.

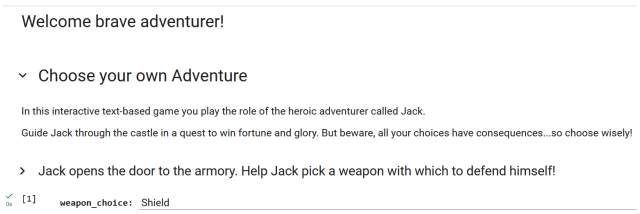

Figure 5. Example sequence to guide a player through an interactive text-based adventure. The main character, Jack, needs to select a weapon in order to proceed and to defend himself from the dangers that lurk in the castle.

5. **Conclusion**

We demonstrate how an LLM can be used to guide the user through a CYOA story while maintaining coherence through a longer narrative. Preliminary results show that the LLM

Figure 6. Example sequence to guide a player through an interactive text-based adventure. Which door should Jack open? The decision is left to the player.

can successfully translate the output of the valid plan into natural language sentences that are then used to provide options to the player of the interactive text-based adventure game. By combining the outputs of the valid plan with the LLM's text generation capability, we provide a logical skeleton of the story, and we can thus maintain coherence over the entire span of the story that is told by the dynamic narrative of the game. Our work is, therefore, a key step in using neuro-symbolic techniques and planning in a novel way for story generation.

References

[1] M. O. Riedl and V. Bulitko. "Interactive narrative: An intelligent systems approach". In: *Ai Magazine* 34.1 (2013), pp. 67–67.

[2] P. Haslum, N. Lipovetzky, D. Magazzeni, and C. Muise. *An Introduction to the Planning Domain Definition Language*. Morgan & Claypool, 2019. ISBN: 9781627058759. URL: http://www.morganclaypoolpublishers.com/catalog_Orig/product_info.php?products_id=1384.

[3] K. Yang, D. Klein, N. Peng, and Y. Tian. "Doc: Improving long story coherence with detailed outline control". In: *arXiv preprint arXiv:2212.10077* (2022).

[4] C. Kallio, S. Zhou, and A. Kurenkov. "Stories by AI". In: *Stories by AI* (2022). URL: https://storiesby.ai/about.

[5] L. Martin. "Neurosymbolic Automated Story Generation". PhD thesis. Georgia Institute of Technology, 2021.

[6] H. Yu and M. O. Riedl. "A sequential recommendation approach for interactive personalized story generation." In: *AAMAS*. Vol. 12. 2012, pp. 71–78.

[7] J. Kelly, A. Calderwood, N. Wardrip-Fruin, and M. Mateas. "There and back again: extracting formal domains for controllable neurosymbolic story authoring". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 19. 1. 2023, pp. 64–74.

[8] K. Valmeekam, A. Olmo, S. Sreedharan, and S. Kambhampati. "Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change)". In: *arXiv preprint arXiv:2206.10498* (2022).

[9] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone. "Llm+ p: Empowering large language models with optimal planning proficiency". In: *arXiv preprint arXiv:2304.11477* (2023).

[10] E. Clark and N. A. Smith. "Choose your own adventure: Paired suggestions in collaborative writing for evaluating story generation models". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2021, pp. 3566–3575.

[11] N. Simon and C. Muise. "TattleTale: Storytelling with Planning and Large Language Models". In: *ICAPS Workshop on Scheduling and Planning Applications*. 2022.

[12] C. Muise, S. McIlraith, and C. Beck. "Improved non-deterministic planning by exploiting state relevance". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 22. 2012, pp. 172–180.

[13] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.