Gameplay as On-Line Mediation Search

Justus Robertson and **R. Michael Young**
Liquid Narrative Group
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
jjrobert@ncsu.edu, young@csc.ncsu.edu

Abstract

Mediation is a plan-based interactive narrative generation algorithm that creates a cascading policy of plans. This policy represents a branching story and can be used by an execution manager in a game to control the series of events that unfold. With a few modifications we show that mediation's search space can embed all possible traversals through a game world. This shift allows gameplay to be modeled as an on-line mediation search where the game's interface is a representation of the underlying graph traversal. In this paper we outline these modifications and present a text-based implementation.

Introduction

One popular way of generating interactive narratives is planning (Porteous, Cavazza, and Charles 2010). Mediation (Riedl, Saretto, and Young 2003) is a plan-based interactive narrative generation algorithm that was first created for the Mimesis system (Young et al. 2004). Mediation has since been modified to use more intelligent search strategies (Riedl et al. 2008), change its story in response to a plan-recognition module (Harris and Young 2009), respond to player actions based on a model of player preference (Ramirez, Bulitko, and Spetch 2013), and modify past events outside the player's knowledge (Robertson and Young 2013). One thing these systems share is their underlying representation of branching story as a *mediation tree*.

A mediation tree (Riedl and Young 2006) (Figure 1) is a cascading policy for controlling interaction in a game world. The nodes of a mediation tree are narrative plans and its edges are *exceptional actions*. An exceptional action is any action a player can take during gameplay that breaks the execution flow of the current plan. When this happens control transitions down the edge that corresponds to the exceptional action and is given to the plan at the tail of the edge. This new plan is generated by a re-planning process called *accommodation* to incorporate the player's action and still reach the interactive story author's desired goal state.

In this paper we transition away from the mediation tree representation to that of a game tree. This shift in representation allows all possible sequences of actions to be directly encoded in mediation's search space.

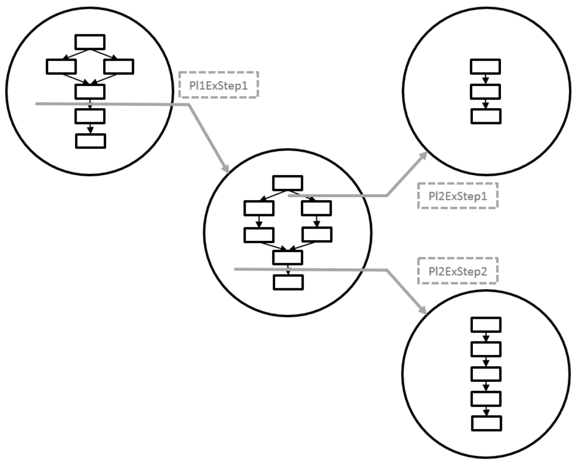


Figure 1: A mediation tree.

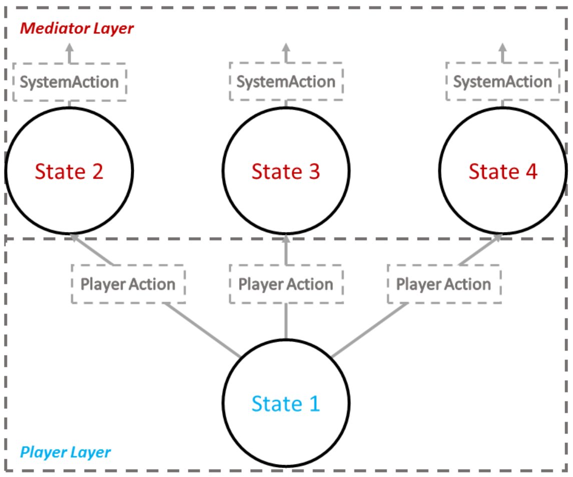
Narrative Trajectories

Riedl and Bulitko (2013) visualize experience management as leading a player through one of a set of narrative trajectories that an intelligent author deems favorable. The automated experience manager guides the player by allowing her to take action and responds with non-player character (NPC) actions and direct world modifications that lead her down a favorable path. This concept of desired narrative trajectories through a space of states influenced by player actions forms the basis of our representation.

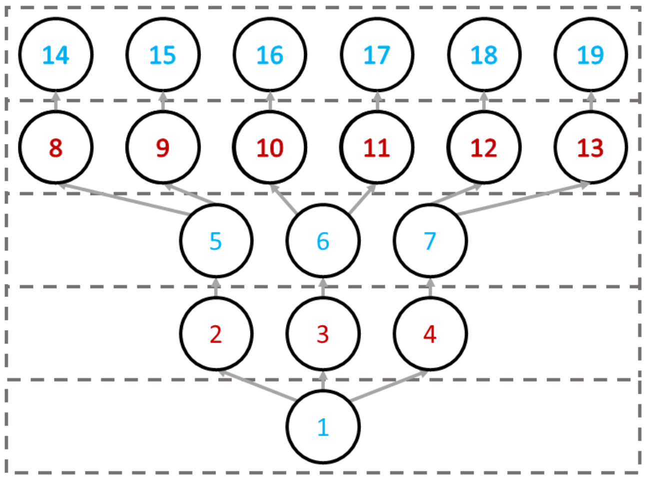
Game Tree Representation

We recast mediation's search space as a game tree (Figure 2) with two participants: the player and the mediator. The nodes of the game tree are world states and its edges are enabled actions. The game tree alternates between layers of nodes that belong to the player and layers that belong to the mediator. The outgoing edges of a player-layer node correspond to the set of player actions enabled by the current node's world state. Each of these edges lead to a node that corresponds to the world state after the action is performed. Each of these resulting nodes belong to a mediator-layer.

A mediator-layer node has a single outgoing edge that is the system response to the player's last action. This system



(a) Two layers of a game tree.



(b) Game tree space.

Figure 2: Game tree visualizations.

response can include an action for each NPC to take and direct alterations of the world state outside of NPC actions. Each of these edges lead to a new player-layer node.

It is the experience manager's job to ensure that the progression of events in this space result in the best possible story given the actions the player chooses to take.

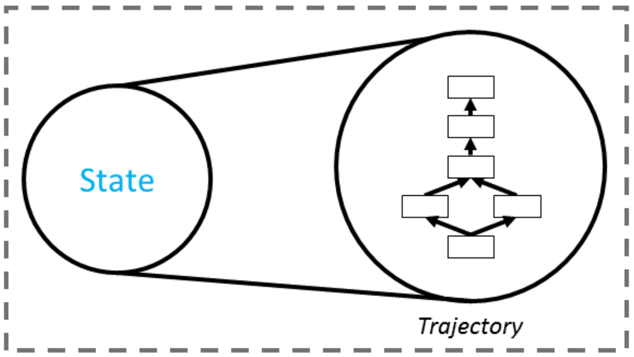


Figure 3: Player-layer node and narrative trajectory.

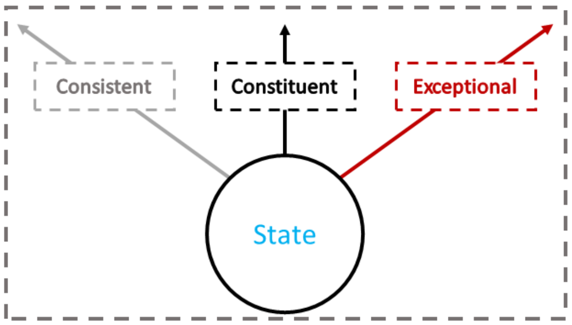


Figure 4: Player-layer action classifications.

System Response Oracle

Game tree space is built from actions available to the player, NPC responses, and modifications of the game world. We imagine that for every series of actions the player could take there is an optimal series of system responses. Together the set of all possible player action sequences and their optimal responses form the optimal branching story structure. When building the game tree, mediation needs an oracle that prescribes the best possible system response. This oracle is currently implemented with narrative plans.

Narrative plans are given to mediation from some linear story generation process and are used as desired narrative trajectories to guide search through the game tree.

Trajectories as System Responses

Every player-layer node in the game tree is associated with a desired *narrative trajectory* (Figure 3). A narrative trajectory is a series of events that mediation desires to take place in the story world from the current state. Every narrative trajectory corresponds to the most desired traversal of the game tree from the current node. Trajectories are supplied by a linear narrative generation module, which is currently a planner. For every player-layer node in the tree mediation classifies the enabled player actions according to how they interact with the current trajectory. Player actions are classified as *consistent*, *constituent*, or *exceptional* (Figure 4).

Together, action classification and narrative trajectories satisfy mediation's system response oracle and allow it to generate mediation-layers in response to player actions.

Consistent Actions

Consistent actions are player actions that are not in the narrative trajectory but do not prevent it from executing. The system responds to consistent actions by first updating the world state to reflect the player's action. It then examines the narrative trajectory to determine if any NPC actions can

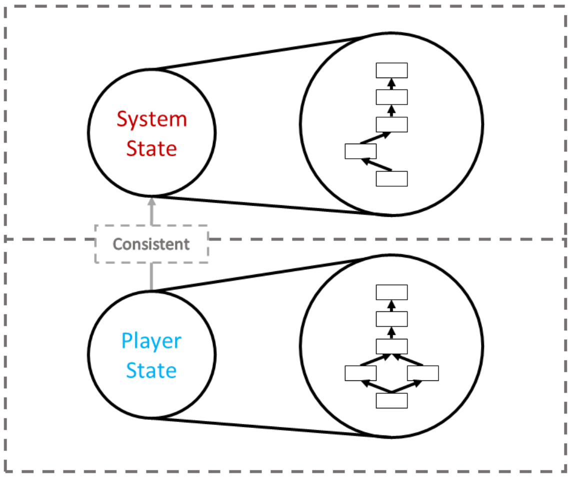


Figure 5: Consistent action update.

be taken at the current point in the plan. If so it allows each NPC character to take one available action from the plan, updates the world state for each action taken, and returns control to the player by creating the next player-layer node.

In Figure 5 the step removed from the story trajectory in the mediation-layer is performed by an NPC. No player action is removed from the trajectory. This new plan becomes the trajectory at the next player-layer node.

Constituent Actions

Constituent actions are player actions that are prescribed by the narrative trajectory and desired by mediation. The constituent action update is very similar to the process performed for consistent actions. The difference between the two is that for a constituent action the player has performed an action included in the trajectory plan. So before the system checks for possible NPC actions it removes the player's performed action from the trajectory.

In Figure 6 both a player-performed and NPC-performed action are removed from the narrative trajectory during the mediator-layer update.

Exceptional Actions

Exceptional actions are player actions that break the execution flow of the narrative trajectory by reversing world conditions needed for future actions to execute. Exceptional actions are handled with accommodation, the process of generating a new narrative trajectory. This new trajectory should be the optimal series of events from the current state given the player's action.

The system is allowed to update its trajectory and take NPC actions in the same layer. Figure 7 shows a new trajectory created by a mediator-layer after an exceptional action is taken.

Game Trees vs. Mediation Trees

Viewing mediation as a game tree search process is similar to viewing it as a mediation tree search process, but the two differ in several important ways. First, game trees model consistent, constituent, and exceptional player actions in their search space, whereas a mediation tree only explicitly models constituent and exceptional actions. Secondly, game trees model how the narrative trajectory changes in response to consistent, constituent, and exceptional player actions, whereas a mediation tree only models updates after exceptional actions. Finally, because of the first two differences the game tree models every possible series of player actions, system actions, and world states that could occur in a game world, whereas a mediation tree models only desired trajectories through an unspecified set of possible states.

Explicitly modeling consistent actions, trajectory updates, and world states makes game tree space larger than mediation tree space, but it affords interesting new possibilities. One such possibility is viewing gameplay as an on-line search through game tree space carried out as a discourse between the player and mediator.

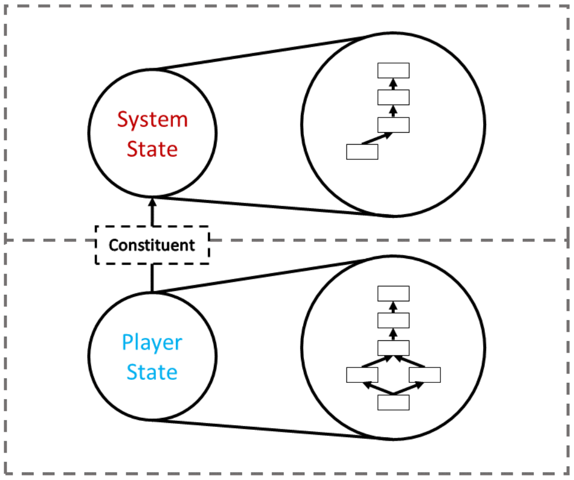


Figure 6: Constituent action update.

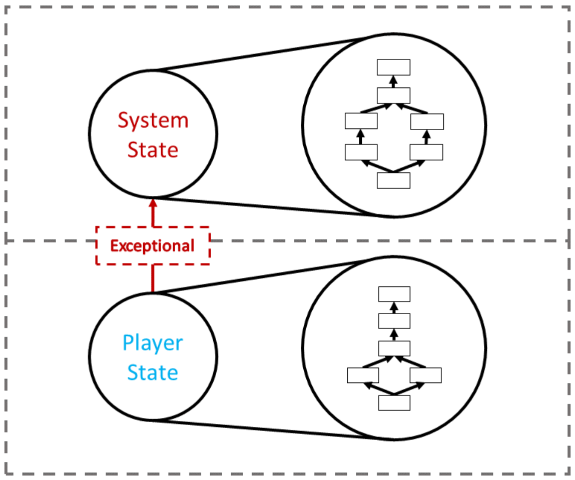


Figure 7: Exceptional action update.

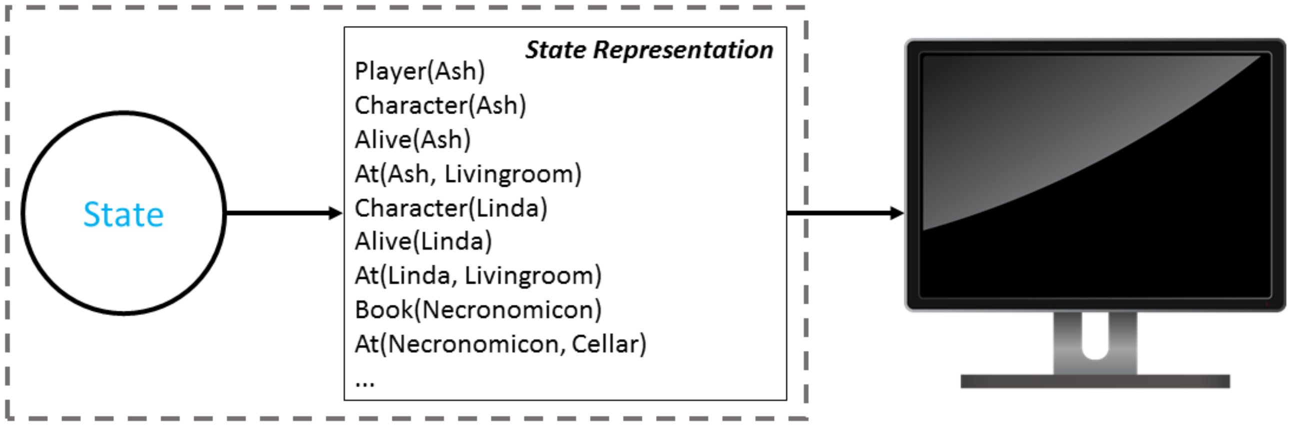


Figure 8: Graph to visualization pipeline.

Gameplay as Game Tree Search

Since a fully expanded game tree contains all possible series of events that could play out in the game world it can be used to drive game experiences. Mediation begins with two inputs: a domain file that specifies actions available to agents and a problem file that describes the initial and goal states of the game world. If given an interface that maps domain actions and world state literals to textual, 2D, or 3D representations, mediation can drive interaction as an on-line search through its game tree.

Every game tree node has an associated world state that consists of logical predicates. Using a mapping function the system can generate a graphical representation of the world state for the player. At every state the player can choose from one of their enabled actions. Once it receives player feedback mediation moves along the edge associated with the player's action in its game tree, makes its mediator-layer update, conveys the NPC actions to the player, and presents the world state associated with the new player-layer node to the player through its visualization system.

With this method the underlying mechanics of the game world are simulated by mediation and depend on the input domain and problem files. These mechanics are then conveyed to the player through a general mapping and display interface that can be used with any domain and problem. So a new experiences can be created with only a new domain, problem, and mapping.

The General Mediation Engine

The General Mediation Engine (GME) is an implementation of a game tree mediator. One aim of GME is to divorce the linear story generation process from the branching story generation process. Because of this GME is not tied to any one story generation component but supports any that are driven by PDDL (McDermott et al. 1998). GME takes as input a domain file, a problem file, and a story oracle. With these it is able to build and search the game tree space that corresponds to the three inputs.

Domain

The domain file consists of PDDL representations of actions that can be performed in the environment. Each action has a list of predicates that must be true in a world state in order for the action to be enabled and a list of predicates that the action makes true once it's executed.

Problem

The problem file consists of PDDL representations of the initial world state and the goal state. It specifies all objects that exist in the world and defines the start and end-points of the narrative trajectory the oracle forms.

Story Oracle

The story oracle is any process that takes as input a PDDL domain and problem file and returns a series of actions that transforms the initial world state specified in the problem file into its goal state. The story oracle exists outside the GME system and is communicated with through PDDL files. As a proof of concept we use the Fast Downward (Helmert 2006) planner as the current story oracle.

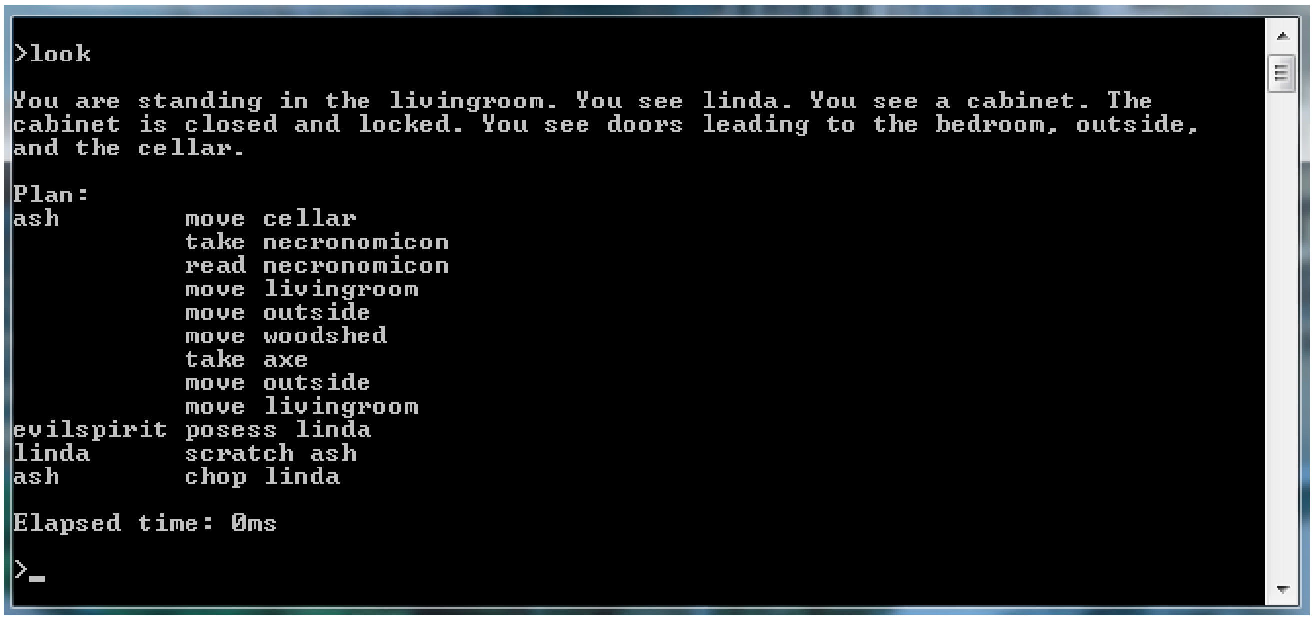
Interface

GME is currently outfitted with a text-based interface that formats and displays the current world state, allows the player to issue commands that correspond to their enabled PDDL actions, and notifies the player of observed actions taken by NPC characters.

Example

As one possible example domain and problem we have modeled *The Evil Dead*. The domain has 12 actions with at most 4 parameters. The problem file contains 13 world objects, an initial state of 35 predicates, and a goal of 3 predicates not including the closed world assumption. A subset of the predicates in the initial state is shown in Figure 8.

Figure 9 shows GME's current text-based interface. The system accepts commands from the player and navigates the underlying game tree according to input. In Figure 9 the



```
>look

You are standing in the livingroom. You see linda. You see a cabinet. The
cabinet is closed and locked. You see doors leading to the bedroom, outside,
and the cellar.

Plan:
ash        move cellar
           take necronomicon
           read necronomicon
           move livingroom
           move outside
           move woodshed
           take axe
           move outside
           move livingroom
evilspirit posess linda
linda      scratch ash
ash        chop linda

Elapsed time: 0ms

>_
```

Figure 9: Initial state of Evil Dead domain.

player has issued a command to look around them. The interface holds observation axioms that can compute the subset of any state that the player can observe. The system runs the current state predicates through these observation axioms, filters away unobservable aspects of the world, formats the remaining axioms into a natural language sentence, and displays the result.

The system is in debug mode so it also displays the current narrative trajectory and the time elapsed between the player issuing a command and the system displaying the results. The trajectory from the initial state is a 12 step plan where Ash unleashes an evil spirit by reading the Necronomicon, the spirit turns Linda into a zombie, Linda injures Ash, and Ash finishes Linda with an axe.

As pictured in Figure 10 the dialog between the mediator and the player is a traversal through the mediator's game tree representation that plays out through the text interface.

Figure 11 shows the system response to the exceptional player action of Ash moving to the bedroom instead of the cellar. The system updates its world state based on the player's action, filters the new state predicates through its observation axioms, formats the remaining predicates, and displays them in natural language. Since the player took an exceptional action the system is forced to create a new narrative trajectory. The trajectory from the new player-layer node is an 11 step plan where Linda reads the Necronomicon, is turned into a zombie, Ash takes the key he finds in the bedroom, unlocks the gun cabinet, and uses the Boomstick instead of the axe. The first action of this plan, where Linda moves to the cellar from the living room, has already been taken during the mediator-layer update.

On an Intel Core i7 3.5GHz system with 8GB RAM it takes about 250ms to query Fast Downward and create a game tree node. The elapsed time here between command and display is 2ms because the system expands and caches the graph frontier in downtime between player actions.

Limitations and Future Work

There are two types of advances that must be made to the system. First, more work must be done to determine how properties of the underlying game tree structure relate to player feelings like enjoyment or agency. Second, the system must be engineered to produce interesting game tree structures and present them to players in a satisfying manner.

Game Tree Structure

Game trees are built from player choices, NPC actions, and direct manipulations of the game world. There are several methods for selectively constructing game trees from these

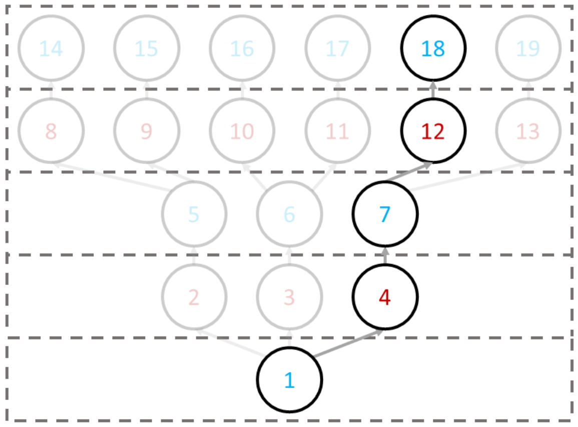
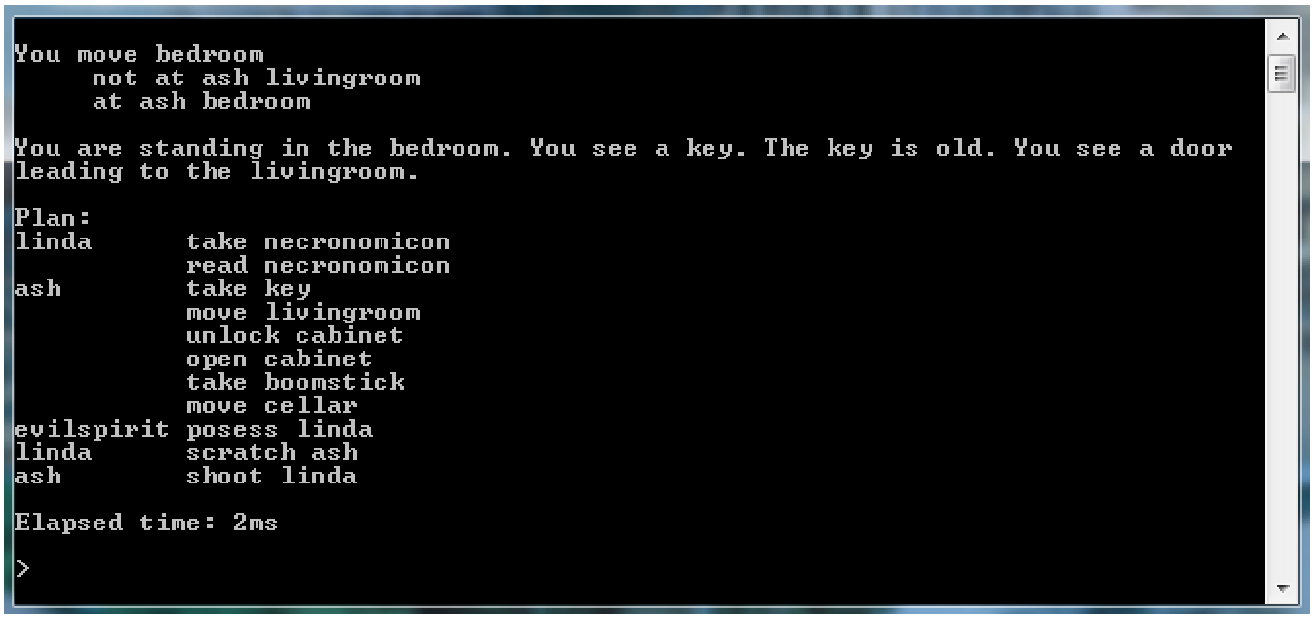


Figure 10: Player traversal through game tree.



```
You move bedroom
     not at ash livingroom
     at ash bedroom

You are standing in the bedroom. You see a key. The key is old. You see a door
leading to the livingroom.

Plan:
linda      take necronomicon
           read necronomicon
ash        take key
           move livingroom
           unlock cabinet
           open cabinet
           take boomstick
           move cellar
evilspirit posess linda
linda      scratch ash
ash        shoot linda

Elapsed time: 2ms

>
```

Figure 11: State after exceptional move action is taken.

elements that are not utilized by the current system. These methods can be divided into more intelligently selecting user choices and selecting system responses:

User Choices The current system assumes that whatever action in the current domain file that is enabled in the current state should be available for the user to perform. It may be possible to intelligently limit or expand the action templates available to the system at any state in order to create a better game tree. For example, Yu and Riedl (2013) selectively present choice options to players based on learned preferences to influence player choices.

System Responses The system currently responds to exceptional user actions by creating a new narrative trajectory with accommodation. Most other mediation systems employ a second response called *intervention* that prevents new branches from being created after players take exceptional actions. *Proactive intervention* (Harris and Young 2009) allows mediation to intervene before exceptional actions are taken and rearrange the world so they cannot occur. These system responses could be incorporated into GME, allowing it to expand the mediator layer of game trees in new ways.

On-Line Search Strategies

Similar to expanding the game tree's frontier as the player takes action, there may be other on-line game tree search and expansion strategies that allow the system to more quickly and intelligently build its branching story space. For example, proactive intervention strategies were originally designed for a mediation tree and a plan recognition module. In a game tree all possible user actions sequences are modeled in the search space, so a probabilistic player model could inform the mediator as it builds the tree about what route it expects the player to take.

Continuous Events

The system currently assumes that all events are discrete and of the same length. This assumption works well in a text-based game, but may be more challenging in 2D or 3D environments. A more robust visualization layer will be needed for environments that allow continuous mechanics to translate between the game and its underlying discrete representation. For example, a visualization layer may be built to allow the player to move through continuous space but only update the mediation representation when they move between discretized sections, like rooms in a dungeon.

Conclusion

Transitioning from a mediation tree to game tree representation for plan-based experience management allows mediation to be used directly as a game world simulator. Paired with a visualization system that maps states and actions to human-readable output, mediation can create a new branching story experience with each pair of PDDL domain and problem documents. This method opens new possibilities for the branching story generation process.

References

Harris, J., and Young, R. M. 2009. Proactive Mediation in Plan-Based Narrative Environments. *IEEE Transactions on Computational Intelligence and AI in Games* 1(3):233–244.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research* 26:191–246.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. *PDDL - The Planning Domain Definition Language*.

Porteous, J.; Cavazza, M.; and Charles, F. 2010. Applying Planning to Interactive Storytelling: Narrative Control Using

State Constraints. *ACM Transactions on Intelligent Systems and Technology* 1(2):10.

Ramirez, A.; Bulitko, V.; and Spetch, M. 2013. Evaluating Planning-Based Experience Managers for Agency and Fun in Text-Based Interactive Narrative. In *Artificial Intelligence and Interactive Digital Entertainment*, 65–71.

Riedl, M., and Bulitko, V. 2013. Interactive Narrative: An Intelligent Systems Approach. *AI Magazine* 34(1).

Riedl, M. O., and Young, R. M. 2006. From Linear Story Generation to Branching Story Graphs. *Computer Graphics and Applications* 26(3):23–31.

Riedl, M. O.; Stern, A.; Dini, D. M.; and Alderman, J. M. 2008. Dynamic Experience Management in Virtual Worlds for Entertainment, Education, and Training. *International Transactions on Systems Science and Applications* 4(2):23–42.

Riedl, M.; Saretto, C. J.; and Young, R. M. 2003. Managing Interaction Between Users and Agents in a Multi-Agent Storytelling Environment. In *Autonomous Agents and Multiagent Systems*, 741–748.

Robertson, J., and Young, R. M. 2013. Modelling Character Knowledge in Plan-Based Interactive Narrative to Extend Accomodative Mediation. In *Intelligent Narrative Technologies 6*, 93–96.

Young, R. M.; Riedl, M. O.; Branly, M.; Jhala, A.; Martin, R. J.; and Saretto, C. J. 2004. An Architecture for Integrating Plan-Based Behavior Generation with Interactive Game Environments. *Journal of Game Development* 1(1):51–70.

Yu, H., and Riedl, M. O. 2013. Data-Driven Personalized Drama Management. In *Proceedings of the 9th AAAI Conference on Artificial Intelligence for Interactive Digital Entertainment (AIIDE).*