Finding Schrödinger's Gun

Justus Robertson and **R. Michael Young**
Liquid Narrative Group
Department of Computer Science
North Carolina State University
Raleigh, NC 27695

Abstract

Interactive narratives are branching stories with events that change based on feedback from participants. One method of generating interactive narratives is a plan-based process called *mediation*. A sub-process within mediation called *accommodation* creates new story content when a participant deviates from the main storyline. We show that a model of character knowledge allows accommodation to find a novel class of branching stories previously inaccessible by the algorithm.

Introduction

Interactive narratives are available in many mediums such as tabletop role playing games, gamebooks, interactive fiction, and video games. One drawback of authoring interactive narratives is that a branching storyline requires a large amount of story content. If branching narrative is viewed as a *story graph* whose vertices are story content and edges are user actions, unique user choices are edges that transition to unique vertices. A branching narrative that offers a large number of unique choices will have an equally large number of vertices which creates a high *authorial burden* or combinatorial explosion of story content (Bruckman 1990). Luckily, AI algorithms can automate the creation of branching story graphs.

Planning is one system capable of generating interactive narratives (Porteous, Cavazza, and Charles 2010). Recent research has modeled interesting properties of narrative within the context of planning such as character intention (Riedl and Young 2010) and conflict (Ware and Young 2011). These linear planners can be paired with an algorithm called *mediation* (Riedl, Saretto, and Young 2003) to manage human interaction as a character within the story. Mediation accomplishes this by building a *mediation tree* data structure. The mediation tree is a representation equivalent to story graphs (Riedl and Young 2006).

However, there is a class of interactive stories that mediation cannot currently generate. This class corresponds to the phenomenon popularly termed *Schrödinger's Gun*. In this paper we incorporate a model of character knowledge into mediation to generate this novel class of interactive stories.

Related Work

Mediation is a plan-based interactive narrative generation algorithm first created for the Mimesis system (Young et al. 2004). Mediation generates a branching story, represented by a mediation tree, by applying *accommodation* and *intervention* to a linear story plan. Accommodation is the process of expanding a mediation tree while intervention prunes away possible branches. There have been several modifications of mediation (Riedl et al. 2008; Harris and Young 2009; Ramirez, Bulitko, and Spetch 2013) but unlike previous work we expand the number of branches mediation can create while building its tree. We accomplish this by identifying a new type of interactive story during accommodation.

Our process is a form of *alibi generation* (Sunshine-Hill and Badler 2010; Li et al. 2014), a method that dynamically creates backstories for non-player characters (NPCs). Sunshine-Hill and Badler use alibi generation to reduce the cost of simulating large crowds by transitioning NPC behavior from a random distribution to an intelligent pattern when observed by a player. Li et al. show that socially believable alibis can be learned from exemplar graphs obtained through crowdsourcing. In this paper we present a mediation system with a model of character knowledge that can replan past events during accommodation, rewriting NPC alibis in response to player actions so they further an author's goals while building a branching story.

Mediation Algorithm

In this paper we present descriptions and definitions that span three layered algorithms: planning, mediation, and Schrödinger accommodation. This section briefly describes the planning and mediation machinery on which our work is built as a foundation for later sections.

Planning

A planner is an automated reasoner that devises an ordered set of actions which accomplish a desired goal. The process begins with a planning problem which consists of an initial and goal state description and a set of operator schemata that define how the world can be transformed. Planners conduct a search to produce a plan, a solution to the planning problem that defines a series of actions which transform the planning problem's initial state into its goal state.

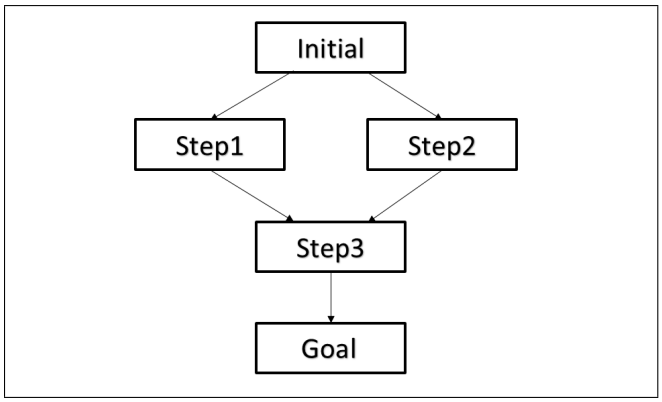

Figure 1: POCL plan representation of five steps, a partial ordering between Step1 and Step2, and five causal links.

POCL Planning

Partial Order Causal Link (POCL) planning is the specific type of planner on which our system is built. POCL plans are comprised of partially ordered steps, which in our case represent actions taken by characters in a story. Actions are defined by a set of general action templates, called operators.

Definition 1 (**Operator**). A planning *operator* is a template for actions that can be added to a plan. Planning operators are represented $<\rho,\epsilon,n>$where ρ, the set of *preconditions*, is a set of predicates that must hold true for the action to be performed, ϵ, the set of *effects*, is a set of predicates that become true after the action, and n is the operator name.

When an operator is added to a plan it is called a step.

Definition 2 (**Step**). A *step* is an instantiation of an operator in a plan. Steps are represented by the triple $<\rho,\epsilon,i>$, where ρ and ϵ are preconditions and effects from the corresponding operator, and i is some unique identifier.

The POCL planning process begins with a plan that consists of two steps. One specifies the initial state of the world and the other specifies a state the system wishes to achieve.

Definition 3 (**Planning Problem**). A *planning problem* describes an initial and desired state and a set of operators. A problem is represented $<\Lambda,I,\Gamma>$, where Λ is the set of operators, I is a set of effects that represent the initial state, and Γ is a set of preconditions that represent a goal state.

The planning process is driven by unification.

Definition 4 (**Unification**). Given two propositions, p and q, *unification* returns a substitution for free variables in p and q, if one exists, such that p and q are logically equivalent.

Steps are added to and arranged in a POCL plan using causal links, ordering constraints, and binding constraints.

Definition 5 (**Causal Link**). A *causal link* tracks how preconditions are fulfilled and is represented $<s_i,e,r,s_j>$, or $s_i \xrightarrow{e,r} s_j$, where r is a precondition of s_j, e is an effect of s_i, and e unifies with r. s_i *establishes* precondition r for s_j.

Definition 6 (**Ordering Constraint**). An *ordering constraint* is a temporal relationship among steps. An ordering constraint states that one step, s_i, within a plan's set of steps, S, is to be ordered before a second step s_j, or $\{s_i < s_j | s_i, s_j \in S\}$.

Definition 7 (**Binding Constraint**). A *binding constraint* is a pair of free variables or constants, (u, v), that must co-designate or a pair, $\neg(u, v)$, that must not co-designate.

Together, a set of steps, binding constraints, ordering constraints, and causal links represent a POCL plan.

Definition 8 (**Plan**). A *plan* is a solution to some planning problem, represented $<S,B,O,L>$, where S is a set of steps, B is a set of binding constraints on free variables in S, O is a set of ordering constraints, and L is a set of causal links.

Mediation builds a tree of plans to control interaction.

Mediation

Mediation is (in our case) an offline process that takes a narrative plan as input and generates a contingency for every action a user could take that breaks plan execution at runtime. An offline planner is one that does not run concurrently with plan execution, does not have access to the current state of the world, and must plan for possible contingencies before they arise. Mediation plans for user activity offline by identifying and responding to possible exceptional user actions.

Definition 9 (**Exceptional Action**). An *exceptional action* is any possible user-performed step s_u instantiated from Λ somewhere in the span $\{s_i < s_u < s_j\}$ of a causal link l, $s_i \xrightarrow{e,r} s_j$, where s_u has some effect q such that q unifies with $\neg r$.

Exceptional actions are steps that can be performed by a user at runtime that reverse some world condition needed for the plan to execute. At plantime, mediation identifies every exceptional action the user could perform and devises a contingency using intervention or accommodation. This paper extends the accommodation process.

Accommodation Given a narrative plan P=$<S,B,O,L>$, an exceptional action s_u, and the causal link l, $s_i \xrightarrow{e,r} s_j$, threatened by s_u, accommodation produces a new plan P' that incorporates s_u and is a solution to the planning problem that produced P. The process begins by removing s_j and all steps causally dependent on s_j from S.

Definition 10 (**Causal Dependence**). Given two steps in a plan, s_i and s_j, s_j is *causally dependent* on s_i if there is a causal link that establishes a precondition of s_j with an effect of s_i. This definition is recursive, all steps s_k that are causally dependent on s_j are also causally dependent on s_i.

The process removes the broken causal link $s_i \longrightarrow s_j$ from L, adds the exceptional step s_u to S, and adds the ordering $\{s_i < s_u\}$ to O. Accommodation sends the modified story to its planner which returns a new plan P' if one exists. This new plan is a revised story that incorporates the user's exceptional action and accomplishes the planning problem's goal state. All new steps added to P' by the planner are ordered after the exceptional action s_u which serves as the current time index. Once generated, P' is added as an accommodation to the policy table for P under exceptional action s_u.

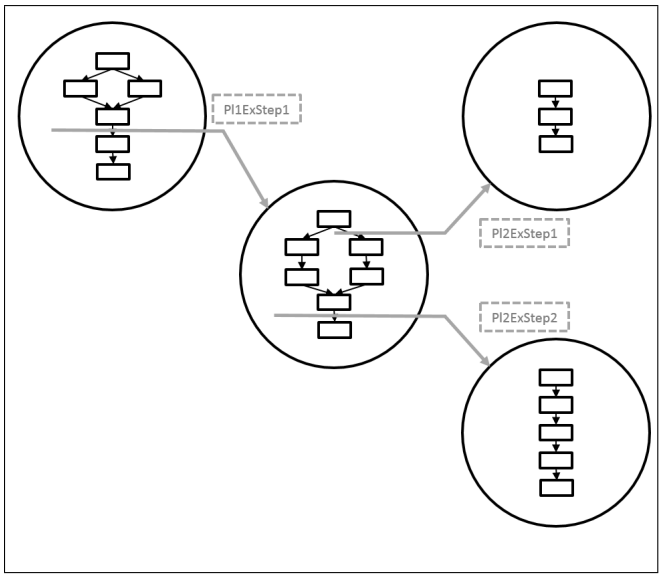

Figure 2: A mediation tree.

Mediation is recursively invocated to find all exceptional actions in P' that happen after s_u, or for all causal links $s_i \longrightarrow s_j$ where $\{s_j < s_u\} \notin O$. When this recursive process is complete, either an accommodation plan or failure mode intervention is assigned to every possible exceptional action in P as well as every nested plan generated by accommodation. The graph of a cascading mediation policy whose vertices are narrative plans and edges are exceptional actions is called a *mediation tree*. An example mediation tree is pictured in Figure 2. Mediation trees are at least as powerful a representation as the acyclic branching story graphs used in many interactive narrative systems (Riedl and Young 2006).

However, accommodation is not guaranteed to find a new plan P' that incorporates s_u. When this is the case it prevents the mediation tree from expanding. One way to allow mediation to accommodate more actions is to broaden its search for valid plans. In the next section we present a new type of interactive story and show how it can be identified during accommodation with a model of character knowledge.

Schrödinger's Gun

This system widens the search space of accommodation by identifying a new class of plans. These plans alter plot events ordered before a user's exceptional action and correspond to the narrative trope *Schrödinger's Gun*, the idea that a fictional universe is constrained only by what is revealed to the audience. The trope is named after *Schrödinger's Cat*, a thought experiment that interprets unobserved aspects of the world as existing in multiple simultaneous states, and *Chekhov's Gun*, a dramatic principle that requires every element introduced in a narrative to serve some role in the story. This trope allows important story events, the Chekhov's Guns, to exist and not exist until observed, like Schrödinger's Cat. A variety of human-authored interactive stories take advantage of this trope to decide unseen plot events after receiving feedback from participants.

Story Examples

Drood is a musical based on the unfinished Charles Dickins novel *The Mystery of Edwin Drood* whose plot focuses on the murder of the title character. Since the novel is unfinished the musical gives the audience an opportunity to vote on who kills Drood before the play begins. All performances of the musical proceed uniformly until the final scene where the murderer is revealed. At this point the full plot changes based on what character was voted by the audience to be the killer. This choice retroactively affects the events of the story but no choice conflicts with what is shown to the audience up to the finale. The film *Clue*, based on the board game of the same name, has a similar branching reveal scene as *Drood*. Three endings to the movie were filmed with different murderers and explanations of the story's events.

The *Choose Your Own Adventure* series of books makes use of this principle to provide varied plotlines. The reader is often presented with a decision under incomplete knowledge of the problem they face and the reality of the situation changes after the decision. For example, in *Space and Beyond* the reader must choose to investigate a mysterious illness by following diplomats from another planet or researching pollution levels. If the reader chooses to investigate the diplomats, she discovers that the illness originated on another planet. If the reader researches pollution levels, she finds that the illness is a byproduct of pollution.

Many video games take advantage of this property of fictional worlds to guide players into making decisions that the author desires. The game *inFAMOUS* asks its player to save one of two towers, one of which contains the protagonist's loved one. No matter what tower the player chooses to save, the story is written so that it was the wrong choice. The *Hitchhiker's Guide to the Galaxy* text adventure has an infamous puzzle that relies on Schrödinger's Gun. In the game, the player is forced to locate ten tools and is told that one will be important at the end of the game. If the player does not collect all ten tools the story is rewritten such that the tool the player needs will always be one she left behind.

Algorithm

The contributions of this system are twofold. First, we add a general mechanism to model how story characters observe and know their world in the mediation process. This general mechanism is a collection of axioms that can be hand-tailored for specific story domains. Second, we incorporate reasoning about the newly added character knowledge information into the mediation process at both the policy generation and planning layers. The policy generator uses this information to remove unobserved steps from an initial plan and the planner identifies new story plans that do not contradict the user's experience. As a result of these modifications the algorithm's search space is enlarged to include plans with revised events ordered before exceptional actions. These new plans take advantage of the user's incomplete information of the narrative environment to rewrite plot events that have already taken place without contradicting the player's experience.

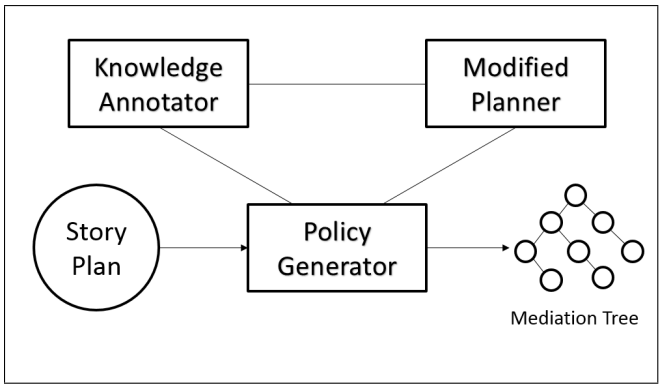

Figure 3: System overview.

Knowledge Microtheory

This system needs a model of knowledge that determines whether a character observes plot events (plan steps) and whether a character observes how plot events change the story world (literals established by step effects). Creating an accurate model of how agents observe and know aspects of their world is a complex problem (Petrick and Bacchus 2004) and may change based on each story domain. Instead of incorporating a highly-structured model of character knowledge into our system, we introduce a general framework that can be modified to fit particular domains. This general framework is called a microtheory.

A *microtheory* (Guha and Lenat 1993) is a group of statements about some topic that may make special assumptions or simplifications about the world. Microtheories are intended to provide a scaffold for building extensible databases that describe a particular domain. Instead of expecting its statements to always hold, a microtheory instead expects that its axioms hold only in a certain *context* (McCarthy 1993). Microtheories are useful because they allow systems to be built on a working theory of some domain that may later be substituted for or integrated with a more fine-grain solution. For this system we present a base model of character knowledge as a microtheory that may be substituted for or integrated with domain-specific theories for each story world.

We define the microtheory by calling it *KnowledgeMt*. To specify that some axiom A belongs to *KnowledgeMt*,

$$ist(\textit{KnowledgeMt}, A) \tag{1}$$

where *ist* stands for 'is true in', *KnowledgeMt* is the context where the axiom holds, and A is the axiom. The driving force behind the base microtheory is that a character observes everything at their current location.

$$\begin{aligned} ist(\textit{KnowledgeMt}, \forall xyz\textit{Character}(x) \wedge \textit{At}(x,z) \\ \wedge \textit{At}(y,z) \rightarrow \textit{Observes}(x,y)) \end{aligned} \tag{2}$$

The characters in this theory can perform actions. If one character observes another character as they perform an action, the first character also observes the action.

$$\begin{aligned} ist(\textit{KnowledgeMt}, \forall xyz\textit{Character}(x) \wedge \textit{Actor}(y,z) \\ \wedge \textit{Action}(z) \wedge \textit{Observes}(x,y) \rightarrow \textit{Observes}(x,z)) \end{aligned} \tag{3}$$

Finally, things in the world may have properties or characteristics. If a character observes a thing in the world, the character also observes its properties.

$$\begin{aligned} ist(\textit{KnowledgeMt}, \forall xyz\textit{Character}(x) \wedge \textit{Property}(y,z) \\ \wedge \textit{Observes}(x,z) \rightarrow \textit{Observes}(x,y)) \end{aligned} \tag{4}$$

This microtheory is used during the knowledge annotation process to determine what parts of a plan are observed by the user's character. These axioms are procedural and produce supported implications when applied to a fact database.

Knowledge Annotation

Given a plan P=<S,B,O,L>, a microtheory of knowledge *Mt*, and the user's character u, the annotation algorithm decides what steps and step effects in P that u could possibly observe under *Mt*. The system returns two sets: S_o, the set of steps in S observed by u, and E_o, the set of relevant step effects observed by u. The system assumes actions will be executed in a linear order as the plan controls interaction with the user. However, POCL plan steps are partially ordered. In order to reason about all actualizations of P the system must enumerate its total orderings.

State Trees A *total ordering* of a partial order plan is any valid linear arrangement of the steps in S according to the ordering constraints in O. The set of all total orderings of P is equivalent to the set of valid topological sorts of the directed acyclic graph whose vertices correspond to the steps in S and whose edges correspond to the orderings in O.

Given T, the set of all total orderings t of P, the system builds a state tree. A *state tree* is a tree whose vertices are world states and edges are plan steps.

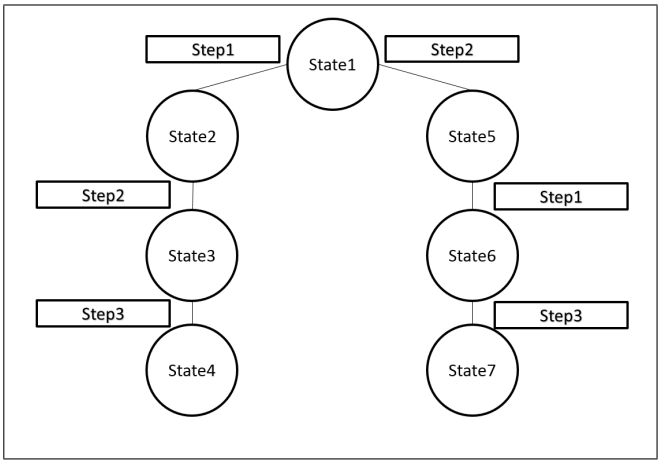

Figure 4: A state tree that corresponds to the Figure 1 plan.

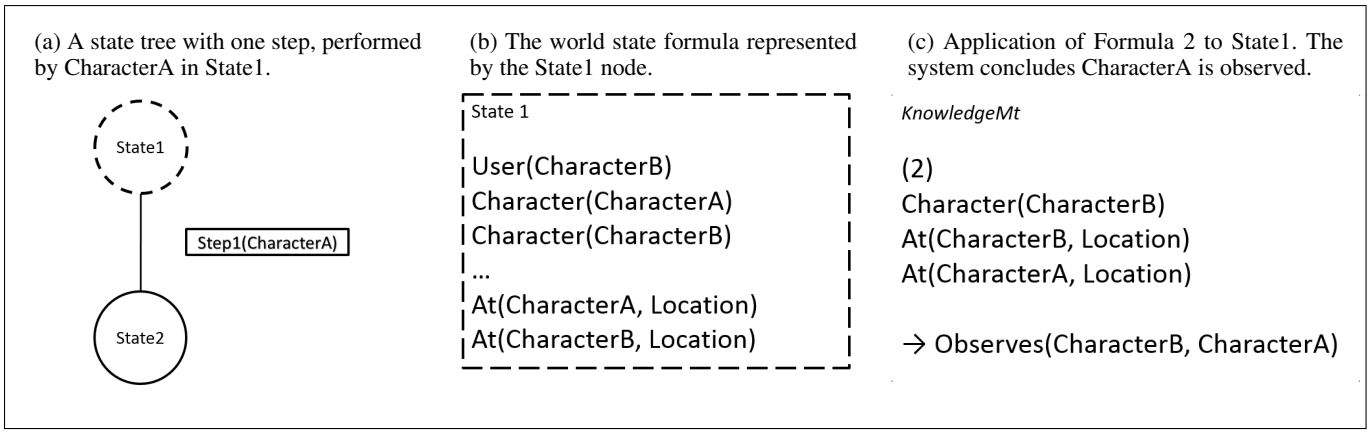

Figure 5: Microtheory application.

Definition 11 (World State). A *world state* is a conjunction of ground literals that specify what is true in a world. In POCL plans the world's initial state is obtained from the effects of the first plan step.

To generate a state tree, the system takes the conjunction of effects from P's initial step and sets it as the root node r. For every unique step s_1 that is ordered after the initial step s_0 in some $t \in T$, the system creates a new branch from r. For each branch created by step s_1 from r, the corresponding child node is generated by applying the effects of s_1 to the conjunction contained in r. The resulting formula will be the state of the world after s_1 is executed in all total orderings where s_1 is performed after the initial step s_0. This process is repeated until all possible states are enumerated. An example state tree is given in Figure 4.

Observing Actions The system uses the state tree to determine what story events are observed by the user's character. For every step $s \in S$ the system determines whether s is observed by u in any total ordering t. This is accomplished with a search through the state tree paired with the observation axioms contained in Mt. The system begins at r and performs a depth first search for an edge that corresponds to s. Actions represented by edges are performed in the preceding state, modeled as the vertex v at the tail of the edge.

To determine whether u observes s the system applies *Mt* to the state representation contained in v to create facts about what characters observe. The system adds predicates *Action*(s) and *Actor*(c,s) to describe the action and its performer to the state description. s is observed by u if *Observes*(u,s) is produced by applying *Mt* to v. If s is observed, the system adds *Observes*(u,s) to S_o. Figure 5 is an example of applying $KnowledgeMt$ to a state description.

Observing Effects A user may observe ways in which the world changes due to an action without observing the action itself. In order to preserve changes of unseen actions the system must track what effects of unobserved steps a user can observe and at what points in the plan they are observed. This system reasons about time in terms of steps, so steps serve as indeces as it tracks this class of effects.

For every unobserved step s_u such that *Observes*(u,s_u) is not in the user's fact database and for every effect e in the set of effects ϵ of s_u, search the state tree for states in which *Observes*(u,e). For each state s that contains *Observes*(u,e), let s_i be the plan step that corresponds to the outgoing edge of s. For every unique s_i, add *Observes*(u,e,s_u,s_i) to the database of facts the user has observed. Once finished, the algorithm has collected all steps and observed effects of unobserved steps that the user can witness at run-time.

Policy Generation

The policy generation algorithm is run at plan-time to build a mediation tree. It begins with an initial plan P and the exceptional step s_e that triggered the parent node (null if the parent is the root). If there can be instantiated and established any new exceptional step s_u that breaks a causal link l ordered after s_e in P the algorithm must prepare a new node of the mediation tree by generating a plan using accommodation.

The system removes all steps causally dependent on l as well as all unobserved steps ordered before s_u. For every step s that is ordered before s_u, $(s < s_u) \in O$, the system checks the fact database for $\neg$*Observes*(u,s). If so, s is removed from S and for every effect e of s and every remaining step s_i in S the system checks for *Observes*(u,e,s,s_i). If so, a new precondition e is added to s_i's list of preconditions ρ. This new precondition preserves the observed effect of the recently removed step at this point in the plan. Finally, the the prepared partial plan is sent to a modified POCL planner.

If a valid plan P' is found it is associated with a new node in the mediation graph reachable from the parent node containing P by way of an edge associated with s_u. The algorithm is recursively invoked to find all children of P'.

Replanning

This system modifies the planner's operator selection phase to ensure added steps are consistent with user observations. A new step s is consistent only if it is ordered after the user's exceptional step s_u, $\{s_u < s\} \in O$, or if $\neg$*Observes*(u,s). Any unobserved step s added to the plan that is possibly ordered before s_u must have consistent effects. An effect e is consistent if for all steps s_i possibly ordered before s_u either e satisfies a precondition of s_i or $\neg$*Observes*(u,e,s,s_i). If a

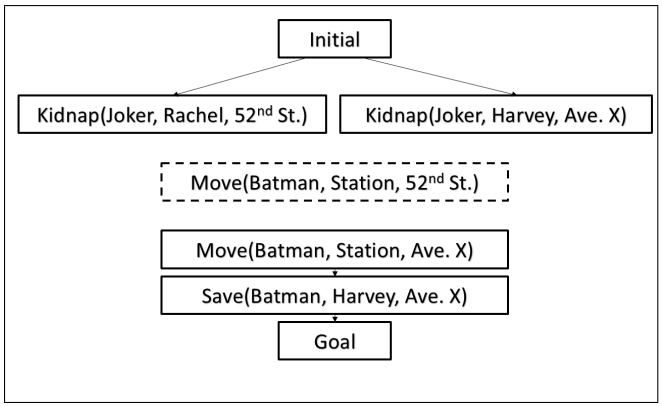

Figure 6: An exceptional action in the Batman domain.

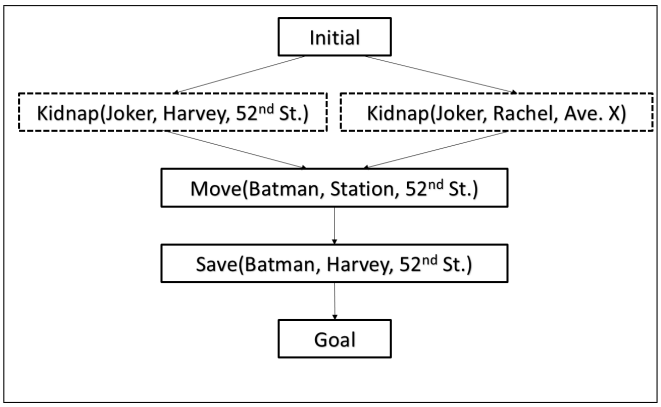

Figure 7: The accommodated plan.

consistent step has one or more inconsistent effects, the step becomes inconsistent. Only consistent steps may be added.

If successful, the planner returns a story that accomplishes the author's objectives, incorporates the user's exceptional action, may include modifications to story events that occurred prior to the inciting incident, and remains consistent with everything the user has observed in the story world. This plan will assume control of execution if the user takes action s_u during P at run time.

Example

To illustrate the system's mechanics consider a scene modified from the third act of *The Dark Knight* (Nolan, 2008).

Linear Plan

The Joker has kidnapped two important characters: Rachel and Harvey. The Joker tells Batman the two kidnapping locations but does not specify what character is where. Batman has time to save one of them. In the film, Batman travels to Ave. X where he saves Harvey at the expense of Rachel.

There are three actions available in the domain: *Kidnap*, *Move*, and *Save*. The problem's goal is for Rachel to die. The Joker, Harvey, and Rachel begin at the apartment. In order to fulfill the problem's goal Harvey and Rachel must be kidnapped. The only character capable of kidnapping is Joker. Once kidnapped Harvey must be saved. The only character capable of saving is Batman who must first move from his initial location at the Police Station to Harvey's location at Ave. X. Once Batman performs the save action on Harvey Rachel is killed and the planning problem's goal is satisfied.

Accommodation

If given the linear plan mediation identifies a threatened causal link: if the player moves from the Police Station to 52nd St. she breaks the establishing condition of Batman being at the Station. This exceptional action is pictured in Figure 6. Accommodation is called and begins by clearing away the action at the tail of the broken causal link, *Move(Batman, Station, Ave. X)*, and the causally downstream *Save(Batman, Harvey, Ave. X)*. The removal of these steps creates an open precondition of *¬Alive(Rachel)* at the goal step. Accommodation sends this plan fragment to the planner.

Unfortunately, no plan can incorporate *Move(Batman, Station, 52nd St.)* and accomplish *¬Alive(Rachel)* because Batman can only move to one location from the Police Station. Once Batman is at 52nd St. the only action available is *Save(Batman, Rachel, 52nd St.)* which produces the effect *¬Alive(Harvey)* and does not fit the planning problem's goal. Unable to find a solution, mediation must assign a failure mode intervention to the player's choice.

Schrödinger Accommodation

If modified to reason about character knowledge the system finds an accommodative solution to the planning problem. Not only are *Move* and *Save* removed from the plan but the knowledge annotator identifies *Kidnap(Joker, Rachel, 52nd St.)* and *Kidnap(Joker, Harvey, Ave. X)* as unobserved by the player's character and removes them as well. Starting from a plan in which the only action is *Move(Batman, Station, 52nd St.)* the planner constructs a plan where the kidnapped locations of Rachel and Harvey are swapped.

The planner accomplishes *¬Alive(Rachel)* by adding the step *Save(Batman, Harvey, 52nd St.)* before the goal. It then supports this action by adding *Kidnap(Joker, Rachel, Ave. X)* and *Kidnap(Joker, Harvey, 52nd St.)* before the exceptional step and verifying that they are consistent actions. Since the two actions change nothing of the world that the user has observed they are added to the plan. The planner returns this solution as an accommodation for the action.

Implementation

This system is implemented with Longbow (Young and Moore 1994). On an Intel Core i7 3.5GHz system with 8GB RAM the Batman mediation tree is built in 5.54 seconds.

Conclusion

In interactive narratives where the user has incomplete information of the story world unobserved past events can be restructured to further author goals. We show that the branching factor of story graphs produced by interactive narrative systems can be increased to include this class of interactive stories by including a model of character knowledge into the generation process.

References

Bruckman, A. 1990. The Combinatorics of Storytelling: Mystery Train Interactive.

Guha, R. V., and Lenat, D. B. 1993. Cyc: A Midterm Report. *Readings in Knowledge Acquisition and Learning* 839–866.

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.

Li, B.; Thakkar, M.; Wang, Y.; and Riedl, M. O. 2014. Data-Driven Alibi Story Telling for Social Believability. In *Social Believability in Games*.

McCarthy, J. 1993. Notes on Formalizing Context.

Petrick, R., and Bacchus, F. 2004. Extending the Knowledge-Based Approach to Planning with Incomplete Information and Sensing. In *International Conference on Automated Planning and Scheduling*, 613–622.

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. 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., and Young, R. M. 2010. Narrative Planning: Balancing Plot and Character. *Journal of Artificial Intelligence Research* 39(1):217–268.

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.

Sunshine-Hill, B., and Badler, N. I. 2010. Perceptually Realistic Behavior through Alibi Generation. In *Artificial Intelligence and Interactive Digital Entertainment*, 83–88.

Ware, S. G., and Young, R. M. 2011. CPOCL: A Narrative Planner Supporting Conflict. In *Artificial Intelligence and Interactive Digital Entertainment*, 97–102.

Young, R. M., and Moore, J. D. 1994. DPOCL: A Principled Approach to Discourse Planning. In *International Workshop on Natural Language Generation*, 13–20.

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.