

Proactive Mediation in Plan-Based Narrative Environments

Justin Harris and R. Michael Young

North Carolina State University
Department of Computer Science
jtharris@ncsu.edu, young@csc.ncsu.edu

Abstract. In interactive plan-based narrative environments, user's actions must be monitored to ensure that conditions necessary for the execution of narrative plans are not compromised. In the Mimesis system, management of user actions is performed on a reactionary basis by a process called mediation. In this paper, we describe an extension to this approach, proactive mediation, which calculates responses to user input in an anticipatory manner. A proactive mediation module accepts as input a plan describing the actions being performed by the user (generated by a plan recognition system) and identifies portions of that plan that jeopardize the causal structure of the overall narrative. Once these portions are identified, proactive mediation generates modifications to the narrative plan structure that avoid the unwanted interaction between user and story. This extension to the original mediation algorithm provides more responses to a user's actions and generates responses that are tailored to the user's actions.

1 Introduction

Recently, a number of interactive applications, including computer games, training simulations, and intelligent tutoring software involve a human user interacting with one or more embedded agents acting in a virtual environment. These applications often require the agents, in concert with the user, to perform coordinated sequences of novel actions structured as an unfolding story or narrative. One approach used to address the coordination of the actions within these story-based systems is the use of a centralized planning system, in which a single planner defines the actions of all agents in a narrative plan [1–3].

If the user in such plan-based systems is allowed a significant amount of autonomy, careful attention must be paid to guarantee that she does not alter the environment such that those actions specified by the planning system cannot be performed. A previously defined process called reactive mediation [4] addresses this issue by pre-determining responses to destructive user behavior. One noteworthy limitation of reactive mediation is that user behavior is examined on a per action basis. That is, mediation responses are taken only at the point where the harmful action is performed. While preserving the validity of the plan's causal structure, this approach fails to take into account the larger

context of the user’s actions. Often, a user performs a *sequence* of actions leading to some desired result, in which one or more of those actions may be harmful to the global plan.

In this paper, an extension to reactive mediation, a process called *proactive mediation*, is described. Rather than examining single user actions, the proactive mediation module examines a proposed plan (provided by an external plan recognition component) that the user is performing in the context of a larger story. Having knowledge about hypothetical future actions that the user may execute allows the proactive mediation module to generate a wider variety of responses to potential harmful user activity, as well as to shape those responses to better integrate with the overall course of the narrative.

2 Background

Generating responses to unanticipated change in an environment has been addressed by a number of research efforts. Firby’s Reactive Action Packages define various action sequences that a robot can perform for a given task in case of failure [5]. Gordon and Iuppa [6] introduce storyline adaptation strategies which define the ways that a story can change in response to unanticipated user action at choice points in a story. Steve, an animated pedagogical agent, monitors user activity and appropriately responds if a user interrupts the current task being demonstrated [7]. While these approaches deal with the generation of responses to unexpected behavior as it arises, none of these systems exploit expectations about likely future events to alter the unfolding action.

In contrast, work by Magerko and Laird [8] incorporates hypothesized future user behavior in the Interactive Drama Architecture (IDA) system. IDA uses a rule-based user model to predict world state changes between predefined plot points in a narrative. Their model is used to determine if a user’s expected actions are likely to satisfy the preconditions of any plot points and to adapt their execution environment accordingly to further advance the story. While their approach detects and reacts to anticipated inconsistencies in the story, the system responds to expected user actions only at the end of the simulation created by their rule-base. In contrast, the process we define below uses an explicit plan representation to describe hypothesized user behavior. This representation not only allows for planning responses to user actions, but also identifies specific harmful actions and the conditions they require for execution. The resulting system can preemptively alter the world state and the actions the system will execute in order to prevent the user from performing some harmful action.

The following section contains a brief overview of Mimesis, the system in which our approach is implemented. For a more detailed description, see [3].

2.1 Mimesis Architecture

The Mimesis system architecture is a distributed, service-oriented approach to the generation and execution of interactive narratives within virtual environ-

ments. Components of the system communicate via XML across the internet to reason about high-level narrative structure.

Two of these components are central to the discussion here: a story planning system based on the Longbow planning system [9] and the component that serves as the virtual world interface between a user and the rest of Mimesis. This component, called the MWorld, contains logic for translating declarative descriptions of the narrative produced by the planning system into function calls that execute directly in the user's virtual environment.

The planning component of Mimesis is used to generate the story structure executed by the characters within the story world. Before an interactive session begins, the planning system builds a story plan which represents the actions of all the agents in the story world, including those of the user. Longbow plan structures are similar to those used in partial-order, causal link and HTN-style planning systems [10, 11]. The plans contain annotations that explicitly mark the temporal relationships between all actions in the story plan, defining a partial order indicating the steps' order of execution. Other annotations, called causal links, mark all causal relationships between the actions in the plan as well. A causal link connects plan steps s_1 and s_2 via condition e , written $s_1 \rightarrow^e s_2$ when s_1 establishes the condition e needed by subsequent action s_2 to execute.

Once the planning system has generated a plan for a story, a scheduler component called the execution manager translates the plan structure into a directed acyclic graph (DAG) representing the temporal dependencies between the steps in the plan. As steps in the DAG are ready to execute, the manager sends commands to the MWorld invoking the corresponding function calls. The MWorld provides updates to the execution manager regarding the status of each function's execution; as each function call completes, the manager updates the temporal dependencies within the execution DAG and sends commands to execute the next set of plan actions.

2.2 Reactive Mediation

As described above, Mimesis drives the action within its story world based on the structure of a plan produced by a narrative planner. Plan execution is complicated, however, because users are relatively unconstrained with respect to the actions that they can perform in the world as the plan is being executed. The plans used by Mimesis are dependent upon user actions, both because some user actions are required for the plans to progress and because the consequences of user actions may inadvertently interfere with the world state on which the plan structure depends. As users issue commands for their characters to perform actions within the story world, these actions must be checked against the narrative plan to determine how they fit with the plan's structure. This process, called *reactive mediation*, is described in detail in [4]. We provide a summary of the process below as background for the extension to mediation that is the main contribution of this paper.

Characterizing User Actions. As the user performs an action, Mimesis must characterize the act with respect to the story’s requirements; actions that the story is depending upon must be identified in order for the story to progress, while actions that interfere with the story’s structure must be identified so that the damage that they might cause to the narrative can be avoided or minimized. By comparing each action executed by the user to the structure of the story world plan, Mimesis automatically characterizes user actions into one of three categories: constituent, consistent, and exceptional.

A *constituent* action is one that maps directly to a step in the story world plan. The action’s type, arguments and temporal and causal structure all match the corresponding constraints on a step specified for user execution.

A *consistent* action is one that is not constituent and whose effects do not alter the virtual world in a way that interferes with the successful execution of the story world plan. Specifically, an action a is consistent just when it is not constituent and, for each of its effects e , there is no causal link in the story world plan that spans the point in time where a is being executed and that is labeled $\neg e$. In practice, most user actions fall into this category.

An *exceptional* action is neither constituent nor consistent, that is, at least one of the effects of the action threatens a causal link in the narrative plan. Formally, action a with effect $\neg e$ threatens causal link $s_1 \rightarrow^e s_2$ when a is performed after s_1 and before s_2 . Exceptional actions, if allowed to execute, break the causal dependencies on which a story plan is based, making the plan impossible to execute.

Responding to Exceptional Actions. When exceptional actions are initiated by the user, their execution changes the state of the story world in such a way that the story plan is no longer executable. In order to prevent this consequence, the Mimesis execution manager monitors each command sent by the user to the virtual world, characterizing it immediately as consistent, constituent or exceptional. When an exception is detected, the system determines an appropriate response before the user’s command is queued for execution. The Mimesis execution manager responds to each exception either by preventing the exception’s threatening effect to be established or by adjusting the narrative such that the action’s performance poses no threat. These outcomes are achieved by *accommodation* or *intervention*, described briefly below.

When an exceptional action is accommodated, it is allowed to execute, and the remaining plan is restructured so that no causal links are threatened. This restructuring can often be slight, such as selecting a different character to perform a task. However, in certain cases, the revised narrative plan may be substantially different from the original, requiring significant computation on the part of the planner. Further discussion of this re-planning process is beyond the scope of this paper.

When an exceptional action is handled by intervention, an alternative action is executed in its place. This alternative action, instantiated from a set of pre-

defined failure modes, is similar in appearance and function to the exceptional action, but has a different set of preconditions and effects.

Policy Tables. The process of revising plans and finding suitable failure modes is complex, and cannot reliably execute in an acceptable amount of time if performed when the exception occurs. In order to provide satisfactory response time when an exception does occur, accommodations and interventions are generated in advance, and held in the *mediation policy table*.

After generating a narrative plan but prior to its execution, Mimesis examines the plan's causal structure and identifies which user actions can cause exceptions at every point in the plan where a user may act. For every possible exception, a queue of appropriate responses (interventions and accommodations) is computed. This action/response queue pair is inserted as an entry into a mediation policy table, along with the interval in the narrative plan during which the action can be performed. The queue of responses is sorted by a heuristic function which determines a qualitative measure of the responses' effectiveness.

3 Extending Mediation

One significant limitation of reactive mediation is that it responds to user activity at the last possible moment. While this approach localizes the point in a story where a user's agency may need to be restricted, intervention and accommodation at the point of an exception can be problematic. Consider a scenario in which the user executes a long series of actions that clearly lead to an exceptional action, for instance, besieging the castle of a story's central character, capturing him and then attempting to kill him. If the system allows the user to spend the time and resources to capture the nobleman, but then intervenes repeatedly as the user swings his sword, the user will not only be frustrated with his apparent inability to hit his target but also with the failure of the system to have guided the story more effectively. The user has put in significant effort in pursuit of a particular course of action, and yet the system has done nothing to deter the user until the action sequence's very end.

This problem is addressed by proactive mediation, which preemptively restructures the narrative plan to better account for anticipated user activity. Rather than monitoring the user's activity only as it occurs, a proactive mediator also examines the user's anticipated plan of action provided by an external plan recognition component (e.g., [12, 13]). Steps in the user's anticipated plan¹ are characterized as constituent, consistent, or exceptional, just as individual actions are categorized under reactive mediation. Proactive mediation extends the

¹ Here, it is appropriate to make the distinction between a step and an action. A step refers to a data structure which describes an event in the virtual world. A plan contains a set of steps, whose referent events, at the time of planning, have not yet occurred. An action refers to an event which is occurring at the present time, regardless of whether or not it is described in a plan.

notions of intervention and accommodation to avoid threats from exceptional steps that have not yet occurred. While the basic objectives of reactive and proactive mediation are the same, a proactive mediator's knowledge of expected future steps is utilized to allow a wider range of responses to user activity.

3.1 Proactive Mediation Input

As described above, Mimesis uses a plan representation to describe story events in a virtual environment. Proactive mediation uses the same plan representation to describe likely future event sequences that the user may perform. This plan is supplied by a plan recognition component which, upon recognizing a user's plan, submits it to the proactive mediator. A plan is defined formally below.

Plan: A plan is a tuple $\langle S, B, O, L \rangle$ where S is the set of steps, B is the set of binding constraints on the steps in S , O is the set of ordering constraints between steps in S , and L is the set of causal links between steps in S . The proactive mediator takes as input a *narrative* plan, a *recognized* plan and an *operator library*, defined formally below:

Narrative plan: A narrative plan is a plan $N : \langle S_N, B_N, O_N, L_N \rangle$ generated by the Mimesis system which describes all of the steps to be performed by the characters in a story, including those of the user. We say that a step s is a narrative step just when $s \in S_N$.

Recognized plan: A recognized plan is a plan $R : \langle S_R, B_R, O_R, L_R \rangle$ that is proposed by a plan recognition system. This plan hypothesizes the sequence of steps that the user intends to perform, and that the user expects to occur. We say that a step s is a recognized step just when $s \in S_R$.

Operator library: An operator library is a collection of operators characterizing the actions available for the given story world domain, instantiated as steps. An operator is a tuple $\langle P, E \rangle$ where P is a set of preconditions that must hold true before the step is executed, and E is a set of effects that are true after the step is executed. Each step in either the narrative plan or the recognized plan can be a system step (any action executed by system resources, characters, etc) or a user step (one initiated by the user and performed by the user's character) and is identified by a unique ID. The set of system steps is denoted S_{SYS} , where $S_{SYS} \in (S_N \cup S_R)$. The set of user steps is denoted S_{USER} , where $S_{USER} \in (S_N \cup S_R)$.

3.2 Generating the Mediated Plan and Identifying Steps

The first step in the proactive mediation process is the creation of a working plan that encapsulates the actions of both input plans. This new plan, called the *mediated plan*, is formed by merging the narrative and recognized plans in the following manner. To simplify the current discussion, a step in the narrative plan and a step in the recognized plan whose IDs are identical are assumed to refer to the same event. The mediated plan is thus a tuple $M = \langle S_M, B_M, O_M, L_M \rangle$ where $S_M = S_R \cup S_N$, $B_M = B_R \cup B_N$, $O_M = O_R \cup O_N$ and $L_M = L_R \cup L_N$.

Once the mediated plan is created, the steps in the recognized plan are then categorized as *inclusive* or *exclusive*. Inclusive steps occur in both plans (i.e., N and R), while exclusive steps only occur in the recognized plan. Inclusive steps may be either user steps or system steps, while exclusive steps are assumed to be only performed by the user. That is, it is assumed that the user will not plan for system steps to occur which are not actually part of the narrative plan.

A step s is an inclusive step just when $s \in S_R \cap S_N$. The set of inclusive steps is denoted S_{IN} . A step s is an exclusive step just when $s \in S_R$ and $s \notin S_N$. The set of exclusive steps is denoted S_{EXL} .

User steps can be constituent, consistent, or exceptional, just as user actions. The definitions of these steps are similar to their action counterparts, with the understanding that the steps refer to future events. The one deviation is the definition of an exceptional step, which can be described as a potential exceptional action given the ordering constraints of the mediated plan. All inclusive steps that are performed by the user are identified as constituent, and all exclusive steps are identified as either consistent or exceptional.

Constituent Step: A step s is constituent just when $s \in S_{IN} \cap S_{USER}$.

Exceptional Step: A step s with effect $\neg e$ is exceptional just when **a)** $s \in S_{USER}$, **b)** $\exists s_1 \rightarrow^e s_2 \in L_N$, and **c)** s is not required to come before s_1 or after s_2 , based on the transitive closure of the ordering constraints within O_M . The set of exceptional steps is denoted S_{EXP} .

Consistent Step: A step s is consistent just when $s \in S_{EXL} - S_{EXP}$.

3.3 Handling Exceptional Steps

Once the steps in the mediated plan have been characterized, the mediator then determines how to respond to each exceptional step. We say that an exceptional step is avoided when the mediator alters the narrative plan in a manner that deals with the harmful effects of the exceptional step. For each exceptional step $s_x \in S_M$ with effect $\neg e$ that threatens some causal link $s_1 \rightarrow^e s_2 \in L_M$, s_x can be avoided by:

- *Proactive Intervention:* Stopping the user from performing step s_x in the mediated plan.
- *Proactive Accommodation:* Eliminating the need for the causal link $s_1 \rightarrow^e s_2$ in the mediated plan.
- *Proactive Reordering:* Enforcing orderings such that s_x cannot occur between s_1 and s_2 .

Proactive Intervention. The purpose of intervention is to prevent the exception's threatening condition from being established. Reactive intervention achieves this by replacing the execution of the exceptional action with the execution one of its failure modes that does not have the threatening condition as an effect. This solution is also possible under proactive intervention. However, since an exception considered by the proactive mediator has not yet occurred,

additional action can be taken by the system to make one or more preconditions of the exception false, thus making the action itself un-executable. This can be achieved by executing a system step which makes a condition false, or by removing a step that causally leads to the exception.

Proactive intervention prevents the user from performing some exceptional step s_x in question. This can be done by stopping the execution of s_x itself, or any step that *contributes* to s_x .

A step s_1 contributes to s_x just when $\exists s_1 \rightarrow^e s_x \in L_M$ or some other step s_2 contributes to s_x and $\exists s_1 \rightarrow^e s_2 \in L_M$. The execution of each contributory or exceptional step s in the mediated plan can be avoided via intervention by one of the following measures:

Substitution: Prevent the exceptional step from establishing the threatening condition by replacing s with one of its failure modes s^* . If s is contributory, s^* must be selected so that at least one of the conditions established by s and used in a contributory causal link is not asserted by s^* . If s is exceptional, the effects of the failure mode must not threaten any causal link in the narrative plan. If the substituted failure mode has any preconditions which are not in the original step, those preconditions are considered open, and appropriate flaws are added to the plan. Fixing these flaws requires additional plan construction in order to make the resulting plan complete.

Aversion: Prevent the execution of s by making one or more preconditions of s false at the point immediately prior to its execution. This is achieved by inserting a system step s_i , called an inversion step, into the plan. Step s_i has an effect $\neg f$, where f is a precondition of s . Additional ordering constraints are added such that s_i must come before s , and s_i must come after all steps which establish f , including the original source of the causal link. If the resulting plan's ordering constraints are inconsistent, aversion using s_i cannot be performed.

Although the inversion step s_i is ordered before s , there is no guarantee that the condition $\neg f$ will be established before s is executed by the user. A race condition exists between the system's execution of s_i and the user's execution of s ; if s appears early in the recognized plan, or if a long sequence of steps is required to establish $\neg f$, then s could be executed first. This race condition is avoided by the system "instantaneously" executing the aversion steps, unconstrained by rules in the virtual world such as animation times or physics. For instance, the system can programmatically shut and lock a door without requiring that a character perform the act. This approach will work, however, only if a) the user does not already know the status of any aversion effects, and b) the user cannot directly observe the state changing from these actions.

Disablement: Remove a contributory inclusive step s from the mediated plan. This prevents s from establishing causal links which contribute to an exceptional step. Once s has been removed (along with all causal links, variable bindings and step orderings relating to s), conditions in the narrative plan that were satisfied by s are no longer satisfied, and some re-planning will be required to reestablish those conditions.

Proactive Accommodation. Proactive accommodation allows the user to perform the exceptional step s_x ; in response, the system re-plans the narrative to reestablish any causal links threatened by s_x 's effects. In this regard, there is no difference between proactive accommodation and reactive accommodation, and the basic mechanism does not differ between the two. The fundamental difference between proactive accommodation and reactive accommodation is that proactive accommodation can alter the narrative steps which occur prior to s_1 . By modifying steps prior to s_1 , proactive accommodation can eliminate extraneous steps from being executed to establish the original causal link, resulting in a narrative plan that is potentially more coherent. Further, plans generated with proactive accommodation can use effects of exclusive steps in the recognized plan to satisfy any open preconditions, potentially giving the user a stronger sense of participation in the story line.

Proactive Reordering. Proactive Reordering introduces additional orderings to the narrative plan that make it temporally impossible for s_x to be executed between s_1 and s_2 . Conceptually, there are two ways to enforce this: add the ordering $s_x < s_1$, or add the ordering $s_2 < s_x$. Both of these options effectively reorder the steps to prevent s_x from being executed between s_1 and s_2 , but neither option is particularly viable.

If the ordering $s_x < s_1$ is introduced into the mediated plan, Mimesis would wait for the user to perform s_x before executing s_1 . This rigid requirement can easily stall the entire narrative if the plan recognition component proposed an inaccurate plan, or if the user simply changed her mind about performing s_x .

Enforcing the ordering $s_2 < s_x$ can be problematic as well, for the simple reason that s_x is to be performed by the user. Stopping the user from executing s_x if attempted before s_2 is already accomplished by reactive intervention.

There is a case, however, in which system steps can be reordered such that s_x can only be executed after s_2 . If $\exists s_c \in S_{IN} \cap S_{SYS}$ and s_c is ordered before s_x , adding the ordering $s_2 < s_c$, implicitly ensures that s_x cannot occur before s_2 (assuming that the ordering is consistent with the mediated plan).

3.4 Re-planning

A number of mediation strategies described above involve removing elements of the mediated plan and filling in the resulting gaps with alternative plan structure. The re-planning process used to fill in the missing plan structure is similar to the plan generation process we use, with one notable difference: no ordering link $s_n < s_e$ can be added to the plan, where s_n is a narrative step and s_e is an exclusive user performed step. This restriction is in place because the system has no direct control over when s_e will be performed, since its execution is left up to the user. As a result, s_e 's execution cannot be guaranteed to follow that ordering. Ordering links and causal links are, however, allowed from exclusive steps to narrative steps, which can act to further involve the user in the narrative by effectively making exclusive actions inclusive.

3.5 Mediation Algorithm

A single mediated plan can potentially contain multiple exceptions, which must all be avoided before the plan can become the active storyline and begin execution. The mediation algorithm used to avoid all of the exceptions is similar in many respects to our planning algorithm, which can be characterized as a *refinement search* through a plan space graph [14]. A node in the graph represents a partial plan, and a node’s children are refinements of a specific flaw in the parent. Similarly, the mediation algorithm is a refinement search through a mediation space graph, where a node represents a plan and its corresponding policy table, and a node’s children are responses to a specific exception. Search through this mediation space is guided by an author-defined heuristic, which is used to qualitatively evaluate each plan/policy table pair. This heuristic could be derived from the same planning heuristic used to generate the original story plan, so that proactive responses that coincide with the author’s intended story are favored.

While expanding the mediation space tree, choosing an exception to avoid is not completely arbitrary. Avoiding an earlier exception can (in the case of intervention) implicitly avoid any exceptions that are causally dependent on the former, so only candidate exceptions that have no contributory exceptions are chosen.

4 Example

The following is an example of a mediated plan just after the narrative plan and recognized plan have been merged and the steps identified. The original narrative plan describes the events in a single chapter of a larger story, in which a secret agent played by the user is attempting to acquire secret documents by posing as an employee of a corporation. The chapter ends with the user being caught in her boss’s office looking for the documents. The scenario begins with the user being given a master key to the building by a collaborator on the inside (*spy*), and then walking to her boss’s office and using the master key to enter. At the same time, the boss arrives in his car, takes the elevator to the seventh floor, and then enters his office. This narrative corresponds to steps 0-8 in the merged plan depicted in Figure 1.

The clever user, knowing that the boss rides the elevator every morning, decides to sabotage the elevator so that she will not be caught. This threatens the narrative plan, as the goal of this portion of the story is for the user to be caught. Specifically, *Move* (Step 9) is identified as exceptional because one of its effects, $\neg At(user, hallway)$, threatens the causal link between the *InitialState* (Step 0) and *Move* (Step 4). Similarly, *Sabotage* (Step 11) is identified as exceptional because one of its effects, $\neg Working(elevator)$, threatens the causal link between the *InitialState* (Step 0) and *RideElevator* (Step 3).

The first exceptional step, *Move* (Step 9), must be allowed to execute, because no failure modes have been defined for *Move*, and no inversion steps exists

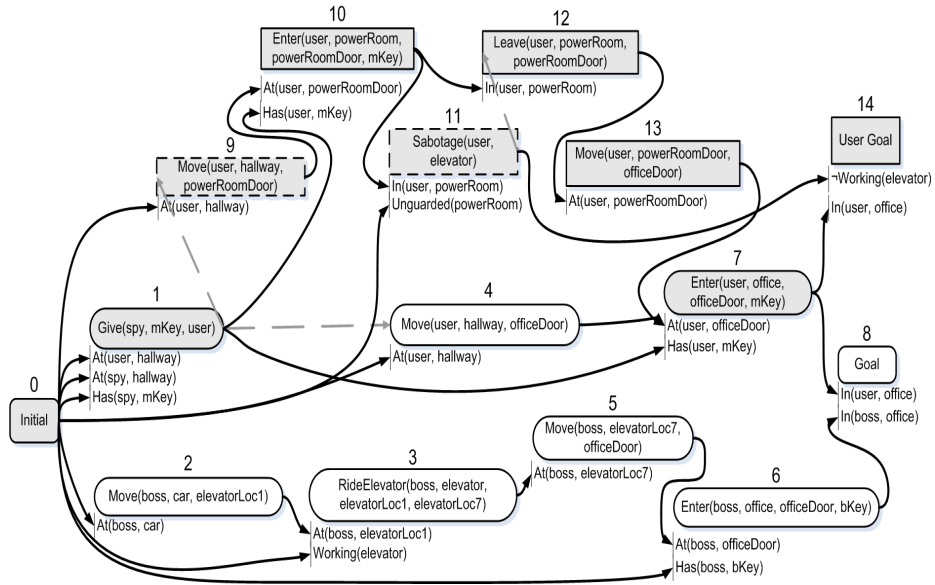


Fig. 1. An example of a merged plan, with all steps identified: the rounded nodes indicate narrative steps, the gray nodes indicate recognized steps, the box nodes indicate exclusive steps, and the dashed nodes indicate exceptional steps. Solid arrows represent causal links, dashed arrows represent ordering constraints.

which can move the user to a different location (she must do this on her own). The only viable option is accommodation, which in this case can remove *Move* (Step 4) and all associated causal links and ordering constraints. This does not introduce any new flaws to plan because *Move* (Step 13) also establishes the condition $At(user, officeDoor)$ which was originally established by *Move* (Step 4).

Substitution can be used to stop the user from sabotaging the elevator, by replacing *Sabotage* (Step 11) with a failure mode, or by replacing any exclusive contributory step (Steps 9 or 10) with a failure mode. Performing a substitution on the *Sabotage* step is handled by reactive mediation (since that single action is exceptional), and in this world no failure modes have been defined for the *Move* operator. The *Enter* operator, however, does have a failure mode defined, which is called *JammedDoor*. This failure mode results in the door not opening and the user not being in the power room.

Aversion can also be used to mediate this plan by inserting a *Move* step followed by an inversion step: *StandWatch*, which moves the seventh floor's security guard to watch over the power room. This has an effect of $\neg Unguarded(powerRoom)$, which prevents the user's character from sabotaging the elevator. Note that the animations for walking the security guard to the power room do not need to play out if the user is not in the area (the action

is not observable). The system can simply make the effects of the step true, in effect "warping" the guard to his new post.

The third form of intervention, disablement, can be applied to *Give* (Step 1), because it is the only inclusive step that contributes to *Sabotage* (Step 11). After removing the step from the mediated plan, two preconditions are left open (in Steps 10 and 7). Some additional planning is required to reestablish $Has(user, ?key-7)$ with the additional constraint that $Has(user, ?key-10)$ is not reestablished. In this particular world, the spy has a second key which only opens the boss's door, so one alternate plan replaces *Give* (Step 1) with a different *Give*, in which this second key is given to the user.

Accommodation allows the user to sabotage the elevator, planning around the $Working(elevator)$ causal link between the *InitialState* (Step 0) and *RideElevator* (Step 3). Alternate plans can include a repairman fixing the elevator, or the boss taking the stairs instead.

The *Sabotage* step can also be allowed if it occurs after the boss is on the seventh floor. This can be achieved by adding the ordering constraint $3 < 1$ to the plan, which waits for the boss to be on the seventh floor before the spy gives the user the master key.

5 Conclusions

In many interactive environments, a human user is permitted to manipulate the environment in a variety of ways. In a plan-based narrative environment, this manipulation may disrupt the actions of other agents or even the actions that the system intends the user to perform. Proactive mediation expands upon reactive mediation to generate a variety of responses to a user's proposed sequence of actions in the environment. Having a hypothesis about future user actions allows proactive mediation to generate a broader range of responses to user actions that can be temporally distributed over the course of a plan.

There are, however, additional factors to consider in evaluating our approach. Proactive mediation relies on effective plan recognition and can be sensitive to the frequency of change in input from the plan recognition component. In addition, planning is computationally complex; in cases where many proactive responses require re-planning, there is no guarantee that an appropriate response can be generated within a reasonable amount of time. However, our preliminary consideration indicates that plans containing potential exceptions occur relatively rarely compared to the number of actions a user is, in practice, likely to perform at any given moment within a story. Most user actions are consistent or constituent; it is the effects of exceptions on the coherence of the story rather than their frequency that motivates the need to address them. While computation performed by proactive mediation can be costly in some cases, the approach we outline is readily implemented as an anytime algorithm: should proactive mediation fail to generate a response in time to address an exception, reactive mediation can still be used. Similarly, should reactive mediation fail to

generate an accommodation in time, intervention (which typically amounts to a straightforward look-up) can be invoked.

Finally, restructuring the narrative plan may result in system-controlled agents performing actions that are not clearly motivated. Our current work includes integration of the proactive mediation component with an intent-driven planner [15] that generates plans where agents' actions can be understood in terms of their own beliefs, desires, and intentions.

6 Acknowledgments

This work has been supported by National Science Foundation CAREER award 0092586 and by Microsoft Research's University Grants Program.

References

1. Cavazza, M., Charles, F., Mead, S.: Planning characters' behaviour in interactive storytelling. *The Journal of Visualization and Computer Animation* **13** (2002) 121–131
2. Mateas, M., Stern, A.: Architecture, authorial idioms and early observations of the interactive drama façade. Technical Report CMU-CS-02-198, Carnegie Mellon University (2002)
3. Young, R.M., Riedl, M., Branly, M., Martin, R., Saretto, C.: An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Development* **1** (2004) 52–70
4. Riedl, M., Saretto, C., Young, R.M.: Managing interaction between users and agents in a multi-agent storytelling environment. In: *Proceedings of the Second International Conference on Autonomous Agents and Multiagent Systems*. (2003) 741–748
5. Firby, R.J.: *Adaptive Execution in Complex Dynamic Worlds*. PhD thesis, Yale University (1989)
6. Gordon, A.S., Iuppa, N.V.: Experience management using storyline adaptation strategies. In: *Proceedings of Technologies for Interactive Digital Storytelling and Entertainment Conference*. (2003) 19–30
7. Rickel, J., Johnson, W.: Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence* **13** (1999) 343–382
8. Magerko, B., Laird, J.: Mediating the tension between plot and interaction. In: *AAAI Workshop Series: Challenges in Game AI*. (2004) 108–112
9. Young, R.M., Pollak, M., Moore, J.: Decomposition and causality in partial-order planning. In: *Proceedings of the Second AI Planning and Scheduling Conference*. (1994) 188–193
10. Penberthy, J.S., Weld, D.S.: UCPOP: A sound, complete, partial order planner for ADL. In Nebel, B., Rich, C., Swartout, W., eds.: *KR'92. Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*, San Mateo, California, Morgan Kaufmann (1992) 103–114
11. Sacerdoti, E.: The nonlinear nature of plans. In: *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence*. (1975) 206–214

12. Albrect, D., Zuckerman, I., Nicholson, A.: Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction* **8** (1998) 5–47
13. Fagan, M., Cunningham, P.: Case-based plan recognition in computer games. In: *Proceedings of the 5th International Conference on Case-Based Reasoning*. (2003) 161–170
14. Kambhampati, S., Knoblock, C.A., Yang, Q.: Planning as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning. *Artificial Intelligence* **76** (1995) 167–238
15. Riedl, M., Young, R.M.: An intent-driven planner for multi-agent story generation. In: *Proceedings of the 3rd Autonomous Agents and Multiagent Systems Conference*. (2004) 186 – 193