Toward Autonomous Crowd-Powered Creation of Interactive Narratives

Boyang Li, Stephen Lee-Urban, and Mark O. Riedl
School of Interactive Computing, Georgia Institute of Technology
{boyangli, lee-urban, riedl}@gatech.edu

Abstract

Interactive narrative is a form of storytelling that adapts to actions performed by users who assume the roles of story characters. To date, interactive narratives are built by hand. In this paper, we introduce SCHEHERAZADE, an intelligent system that automatically creates an interactive narrative about any topic from crowdsourced narratives. Our system leverages the experience and creativity of humans by crowdsourcing a corpus of linear narrative examples. It then constructs an executable plot graph, which is a knowledge structure that defines the legal space of an interactive narrative, by learning the plot events, execution precedence, and event separations. We demonstrate the system can successfully construct an interactive narrative based on noisy human input.

Introduction

Interactive Narrative (IN) is a form of storytelling in which users affect a dramatic storyline through actions by assuming the role of characters in a virtual world. The simplest INs are Choose-Your-Own-Adventure books and hypertexts in which each plot point has branching options. More complex systems use artificial intelligence (AI) to determine available options for the user.

A common knowledge representation employed by AI-based IN is the *plot graph*, which models the author-intended logical flow of events in a virtual world as a set of precedence constraints between plot events (Weyhrauch 1997). A plot graph defines the space of legal story progression and ultimately determines possible events at any given point in time. For example, a plot event of the player opening a vault must be preceded by plot events of the player finding the vault and acquiring the vault's key (Nelson & Mateas 2005).

This paper asks the question: *Can an intelligent computational system create an interactive narrative?* To date, plot graphs have been created by human authors, who impart their knowledge of how the virtual world should work. An intelligent system capable of generating a plot graph would greatly alleviate the need for such manual work, and allow interactive narratives to be created on demand. Ideally, a user could declare that he or she wants to play an interactive narrative about any desired topic, such as robbing a bank or going on a date, and have it ready to be played within a short time period. To meet such demands, an AI system must be able to generate a plot graph about any topic in a timely and economical manner. We envision a number of practical applications: facilitating amateur game creation, rapid acquisition and rapid generation of training practice environments.

One of the critical research challenges for the automatic generation of interactive narratives is how to obtain domain knowledge about the topic for which to construct an interactive narrative. This is complicated by the fact that we cannot assume a pre-existing ontology of plot events. This is especially true if plot events can be unique to a particular interactive narrative. Our solution is to delegate this complex knowledge authoring task to a large number of anonymous workers via Web services, i.e. we *crowdsource* (Quinn & Bederson 2011).

We do not assume crowd workers possess expertise in computer science, modeling, or graph structures. Instead, crowd workers are asked to provide linear archetypical narratives in natural language for the given topic; this is a natural mode of communication for humans. This yields a highly specialized corpus of example narratives from which a general model of the topic can be learned. This corpus is constructed in a way that reduces the reliance on natural language processing (NLP), which is still unreliable for complex inputs. Crowdsourcing provides access to human memories and creativity; a surrogate for the lifetime of experiences held by a human author.

To our best knowledge, this is the first attempt to automatically generate an executable plot graph for the purpose of creating an interactive narrative. We describe the technical approach, provide a real-world example, and discuss future directions.

Background and Related Work

A plot graph is usually defined as a set of plot events with precedence constraints, with which the author of an IN system indicates a logical flow of events (Weyhrauch 1997). For any given plot event, precedence constraints indicate which other events must occur first. For example, finding the vault and the key must precede opening a vault. Additionally, there may be OR-relations between a plot event's parents, indicating that a plot event can be reached by a variety of distinct means (Nelson & Mateas 2005).

A plot graph constrains the sequence of events that can be performed to be consistent with the precedence constraints. It defines a space of legal sequences of plot events. It does not, however, distinguish whether one sequence is better than another. This responsibility is handled by a *Drama Manager* (DM), an autonomous, omniscient, non-embodied agent that attempts to maximize a set of author-provided heuristic functions to improve user experiences. Search-based Drama Management (e.g. Weyhrauch 1997; Nelson & Mateas 2005) uses adversarial search to select DM actions—causers, deniers, and hints—that increase the likelihood that the player will follow a trajectory that scores well. Declarative Optimization-based Drama Management (e.g. Nelson et al. 2006) uses reinforcement learning to select DM-actions. We leave drama management for an automatically generated plot graph for future work.

There are other approaches to Drama Management that use partial order plans instead of plot graphs (Magerko 2005; Young et al. 2004; Riedl et al. 2008). While these approaches generate the narratives users experience dynamically, they require a hand-authored set of domain operators, which similarly defines a space of possible story plans except with events selected and ordered on the fly.

To date, the creation of interactive narratives and their plot graphs or domain models is a task for human designers. Giannatos et al. (2011) describe a technique by which an intelligent system can suggest new plot events and new precedence constraints that prune undesirable narrative sequences, although it cannot indicate the semantic interpretation of these new plot events. Other work (Nelson & Mateas 2008; Treanor et al. 2012) focuses on semi-automated generation of simple arcade games but focus on game mechanics.

Mining web content is an emerging technique in interactive entertainment. Crowdsourcing is used in *The Restaurant Game* (Orkin & Roy 2009) to teach NPCs to emulate restaurant-going behavior via player action traces. The Restaurant Game has an underlying domain model comprised of the actions and animations that avatars can perform. In comparison, we learn a list of primitive events and their precedence from scratch. *SayAnything* (Swanson and Gordon 2008) co-creates stories with human assistance by mining events from Weblogs and thus does not require a fixed domain model. Human players provide every other sentence, which helps to retain story coherence.

Outside of games and storytelling, Chambers and Jurafsky (2009) describe a technique for learning script-like knowledge from news corpora but do not specify an application. By crowdsourcing, our system obtains a specialized corpus with easy-to-process linguistic structure that helps overcome many of the challenges of NLP.

The SCHEHERAZADE System

The SCHEHERAZADE system (distinct from the story annotation system by Elson and McKeown (2010)) attempts to create an interactive narrative from a simple user-provided topic, such as bank robbery (used throughout the paper). The system uses Amazon Mechanical Turk (AMT) to rapidly acquire a number of linear narrative examples about typical ways in which the topic might occur. This is equivalent to outsourcing a set of highly relevant experiences from which to learn a generalized model—the plot graph—of those experiences. Our system can generate an interactive narrative about any topic for which a crowd of average people can generally agree on the main events that should occur, although not necessarily on the specific sequence of events.

Our plot graph representation differs slightly from that of previous work. In our system, a plot graph is a tuple $G = \langle E, P, M, O \rangle$ where E is the set of plot events, $P \subseteq \{(x, y) | x, y \in E\}$ is a set of ordered pair of events that describe precedence constraints, $M \subseteq \{(x, y) | x, y \in E\}$ is a set of unordered mutual exclusion relations, and $O \subseteq E$ is the set of optional events. Whereas some prior plot graph representations use OR-relations between precedence constraints to indicate multiple possible ways of activating a plot event, we note which plot events can never co-occur in the same narrative experience. This performs the same function as OR-relations, but is more general.

The following sections describe our technique for automatically learning a plot graph from a human crowd and using the plot graph during interactive execution.

Plot Graph Learning

To learn a plot graph for a given topic we use a four-stage process. The first three stages are described in Li et al. (2012) and summarized below. The fourth stage is introduced as a new contribution. By identifying mutual exclusion relations between events and identifying events that are optional, we make interactive execution possible.

The process begins with a user request for an interactive narrative on a particular topic. The system then generates an automated query to AMT to solicit *typical* narratives of

the given topic, provided in natural language. To simplify the complexity of natural language processing, crowd workers are asked to segment their narratives such that each sentence contains one event. Crowd workers are instructed to use one verb per sentence, avoid complex linguistic structures such as conditionals, avoid compound sentences, and avoid pronouns.

Second, the system analyzes the simplified natural language narrative examples to discover the fundamental plot points on which people agree. Specifically, sentences from different narrative examples that are semantically similar are clustered together to create a plot event. Because of the simplified language used in narrative examples, the system can use simple semantic similarity and clustering algorithms to discover plot events with relatively high accuracy. For example, we identify plot events for the bank robbery scenario with 80.4% purity, a standard measure of cluster homogeneity. We can also use a second round of crowdsourcing to ask crowd workers to improve the accuracy of NLP / clustering.

Third, we identify the precedence constraints between plot events. Crowd workers produce noisy and sometimes erroneous answers such as omitting steps, requiring resilience against noise. For all pairs of plot events we select between the two orderings $e_1 \rightarrow e_2$ or $e_2 \rightarrow e_1$ based on statistical frequency (or neither if both orderings are equally likely or there is not enough data to make a conclusion). Precedence relations with statistical frequency greater than a threshold T_p are recognized. Relations that fall slightly below the threshold may be recognized if adding the relation to the graph reduces the error metric, computed as the difference between the distance between two events on the graph and their average distance in the input data set. This effectively lowers the threshold locally and provides a flexible mechanism that caters to noisy input narratives.

Fourth, we go beyond prior techniques to identify mutual exclusion links and optional events. Mutual exclusion links indicate that two events cannot co-occur in a single narrative experience. We measure the *mutual information* between events to determine whether they might be mutually exclusive. The mutual information of two random variables describes their interdependence, or the extent that one can predict the other. Suppose E_i is a random variable representing whether event e_i exists in an example narrative. The mutual information between two events is:

$$\mathrm{MI}(E_1, E_2) = \sum_{E_1 \in [0,1]} \sum_{E_2 \in [0,1]} C\,(E_1, E_2)$$

where

$$\mathrm{C}(E_1, E_2) = p(E_1, E_2) \log\left(\frac{p(E_1, E_2)}{p(E_1)p(E_2)}\right)$$

and $p(\cdot)$ denotes a probability distribution or a joint distribution. Thus, $p(E_1{=}1)$ is the probability that event e_1 happens in a narrative, $p(E_1{=}0)$ is the probability that e_1 doesn't happen in any story, $p(E_1{=}1,\ E_2{=}1)$ is the probability that e_1 and e_2 happen in the same narrative, etc.

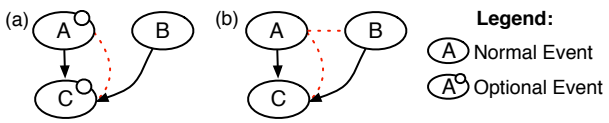

Figure 1. (a) Events A and C must be optional. (b) A and C are not optional due to the mutual exclusion between A and B.

Events e_1, and e_2 are both deemed mutually exclusive when $\mathrm{MI}(E_1, E_2)$ is sufficiently high, indicating that there is a non-random relationship between the two variables, and when $C(E_1 = 1, E_2 = 0) + C(E_1 = 0, E_2 = 1) > 0$, indicating that the presence of one event predicts the absence of the other. We use a threshold T_m on $\mathrm{MI}(E_1, E_2)$ to indicate the degree of non-randomness required to create a mutual exclusion link between events e_1 and e_2.

For the purpose of creating executable experiences, we generally assume that precedence constraints are *causal* in nature. That is, if e_1 precedes e_2, e_2 can only execute after e_1 is executed. However, if a mutual exclusion relation also exists between e_1 and e_2, an contradiction is created: e_2 cannot execute before e_1 (due to precedence) nor after e_1 (due to mutual exclusion). This suggests that the causal assumption does not apply and we should interpret both events as being optional, thus allowing one or both to be skipped. Specifically, two events e_1 and e_2 are considered optional if (a) e_1 and e_2 share a mutual exclusion link and (b) e_1 is a predecessor of e_2 according the set of precedence constraints, and (c) e_1 is not mutually exclusive with another event e_3 that is also a predecessor of e_2. This case is shown in Figure 1(a). However, another predecessor e_3, being mutually exclusive to e_1, indicates the existence of a valid path to e_2 even if e_1 is executed. In this case, we do not recognize these events to be optional, as shown in Figure 1(b). The recognition of optional events can be considered as local relaxation of the causality assumption.

Figure 2 shows a learned plot graph for a bank robbery situation. The plot graph was generated from the procedure described above, although we manually corrected the semantic clusters to simulate the iterative process of crowdsourcing corrections, which is not yet implemented.

Interactive Execution

In this section, we describe how SCHEHERAZADE creates an interactive experience from the plot graph representation. The plot graph ensures that players always experience valid stories on the topic, regardless of their choices in the interactive narrative. Once a plot graph has been constructed, the remaining task is to determine what options—the set of events that can happen next—are available to the

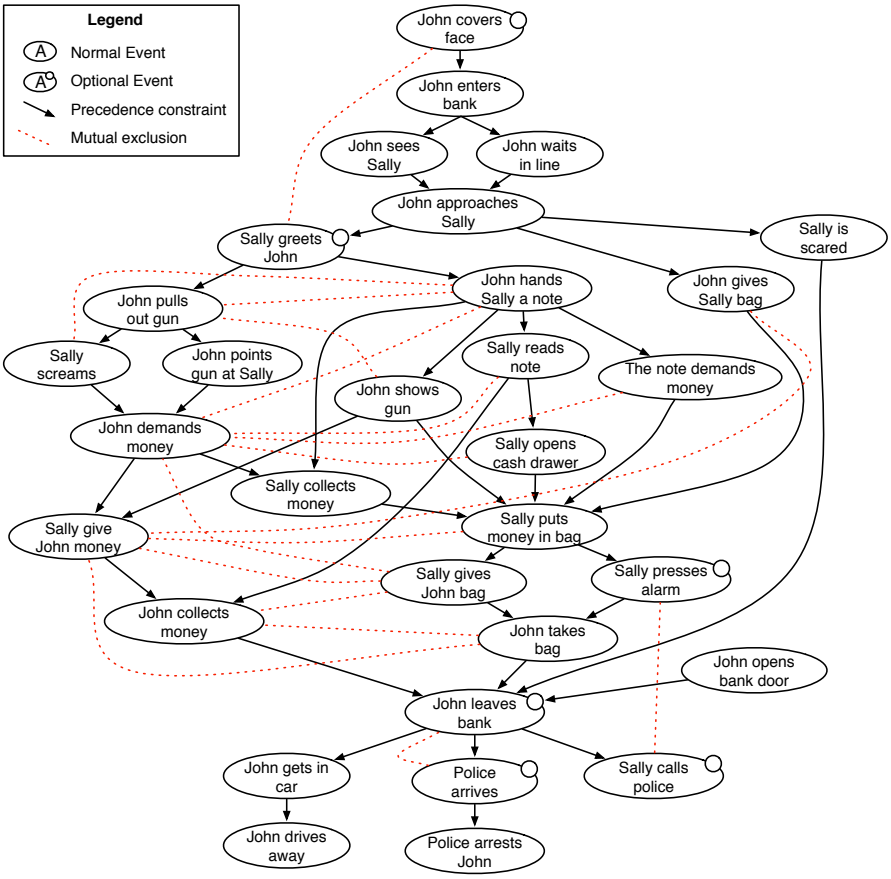

Figure 2. A plot graph for the bank robbery situation

player at any given time point. Figure 3 shows the SCHEHERAZADE game loop.

First, the system determines which plot events are executable. At each step in the game, we compute the set of executable events. A plot event is executable when all if its direct, non-optional predecessors have been executed, except those parents excluded by mutual exclusion relations, as shown in Figure 4. An executable event can belong to the player ("John") or a non-player character (NPC) (e.g., "Sally", or the police). Either the player or an NPC makes a decision to execute an event. NPCs are addressed further in the next section.

Once a plot event from the list of executable events is executed and is added to the history, SCHEHERAZADE performs graph maintenance to keep track of mutual exclusion relations and optional events. Events that are mutually exclusive to the newly executed event are removed from the graph. A recursive process also removes any event completely dependent on deleted events, i.e. an event is removed if all its parents have also been removed. The exclusion process continues until no such events exist in the graph. To avoid losing structural information, direct parents of removed events are linked to direct successors of removed events with precedence constraints.

To simplify record keeping, we also remove optional events that have not been executed but their successors have. These optional events have been skipped and cannot be executed without violating precedence relations. The algorithms for selecting executable events and graph maintenance after event execution are given in Figure 4.

Interactive Event Selection

After candidate events for execution are found, someone—the player or an NPC—must choose from the candidates. A simple syntactic analysis is used to find the subject of the sentences underlying each event, which determines the actor of the event. When every candidate event can be performed by the player, the system presents all options to the player and waits for a response. When every candidate event can be performed by NPCs, the system randomly picks an event to execute. In future work, a Drama Manager would inform the selection of NPC events to better manage the player's experience.

Executable events may contain both player options and NPC options. When this occurs, SCHEHERAZADE waits a predetermined period of time for the player to make a selection. By the end of the period, if the player hasn't

```
Procedure GAMELOOP (plot-graph)
  history = ∅
  For each optional event e in plot-graph
    For each p in direct predecessors of e
      For each s in direct successors of e
        Insert a link from p to s into plot-graph
  Do
    options = EXECUTABLEEVENTS(plot-graph, history)
    executed = WAITFOREXECUTION(options)
    history = history + executed
    UPDATEGRAPH(plot-graph, history)
  While not ISTERMINAL(executed)
End Procedure
```

Figure 3. The interactive narrative execution loop.

acted, the system randomly selects an NPC option. This introduces a competition mechanism, which may add a little fun to the game. For example, when playing as a bank robber, the player may drive away before the police arrive, or the police may arrive just in time to arrest the player. More sophisticated techniques for resolving the race condition will be explored in future work.

Example and Discussion

The plot graph shown in Figure 2 is generated from 60 stories crowdsourced from AMT. The workers were asked to provide a bank robbery story involving a robber named John and a bank teller named Sally. Stories that violate our language requirements were manually filtered and rejected. The clustering results have been manually corrected. The threshold $T_p = 0.6$, $T_m = 0.05$, and the minimum cluster size was set to 4. The resultant plot graph contains 31 events, whose names are manually chosen to reflect the cluster's content. Events that have no ancestors and occur in the first half of the graph are automatically selected as possible "start" points for the interactive experience. Thus, "John opens bank door" (Figure 2) is excluded and deleted.

We illustrate the SCHEHERAZADE game loop using the plot graph from Figure 2. Suppose the history at the current time point includes "John enters bank," "John sees Sally," "John waits in line," and "John approaches Sally." At this point, there are five options:

- "John gives Sally bag"
- "Sally is scared"
- "Sally greets John"
- "John pulls out gun"
- "John hands Sally a note"

Because "Sally greets John" is optional, it can be skipped, and thus its two direct successors are also executable.

Suppose the player quickly decides to pull out a gun. This event is mutually exclusive with "John hands Sally a note", which is deleted from the graph along with several of its descendants until "Sally collects money", "Sally puts money in bag", and "Sally gives John money".

```
Procedure EXECUTABLEEVENTS(graph, history)
  executable = events whose every direct predecessor on graph
    is in history OR is optional
  return (executable - history)
End Procedure

Procedure UPDATEGRAPH(graph, history)
  excluded = events recursively excluded by mutual exclusions
  expired = events ruled out by temporal orderings
  For each e in excluded
    For each p in direct predecessors of e on graph
      For each s in direct successors of e on graph
        Insert a link from p to s into graph
  return REMOVEEVENTS(graph, excluded ∪ expired)
End Procedure
```

Figure 4. Finding executable events and updating the graph after event execution.

"Sally greets John" is also deleted because it is optional and one of its successors has been executed. This makes sense: one would not expect to greet a customer once he or she has pulled a gun.

Assuming the next two events executed are John pointing the gun and Sally screaming, the next noteworthy choices are:

- "John gives Sally bag"
- "John demands money"
- "Sally is scared"

If the player demands money, there will be a choice between collecting the money in a bag or getting the money directly, options which are mutually exclusive.

There are at least 149,148 unique linear experiences that can be expressed with the bank robbery plot graph, as determined by a brute-force search. This demonstrates good *authorial leverage* (Chen, Nelson, & Mateas 2009)—the ratio of possible experiences to input authoring effort—that SCHEHERAZADE is capable of, considering that the crowd only had to provide 60 linear examples. Figure 5 shows two of those stories.

Observing the graph, we note that mutual exclusion relations tend to meet our expectations by separating alternatives (use a note vs. directly demand money; use a bag vs. hand over the money) and allowing for different choices to be mixed freely. As a result, portions of different narrative examples can occur together in unique combinations. We tend to agree with all the identified precedence constraints, but feel some may be missing when events do not appear in enough stories; this may be resolved by soliciting more narratives from the crowd.

Since the plot graph is generated from error-prone human examples, flaws can emerge. Mutual exclusions may be overly restrictive, such as between "John demands money" and "Sally gives John bag." We believe most players will not notice the disappearance of apparently legitimate choices. Sometimes precedence constraints ap-

John enters bank.	John covers face.
John sees Sally.	John enters bank.
John waits in line.	John waits in line.
John approaches Sally.	John sees Sally.
Sally greets John.	John approaches Sally.
John hands Sally a note.	John gives Sally bag.
The note demands money.	John pulls out gun.
Sally reads note.	John points gun at Sally.
Sally opens cash drawer.	Sally scared.
Sally collects money.	Sally screams.
John gives Sally bag.	John demands money.
John shows gun.	Sally collects money.
Sally puts money in bag.	Sally puts money in bag.
Sally presses alarm.	Sally presses alarm.
Sally gives John Bag.	John takes the bag.
John takes the bag.	John leaves bank.
Sally is scared.	Police arrests John.
John leaves bank.	
Police arrests John.	

Figure 5. Example Stories

pear to be missing, such as the one between "John shows gun" and "John gives Sally bag" in Figure 5 (left). However, when the options are presented to human players, they might naturally make the more logical choices. Future experimentation will determine the effects of flaws such as these on real human players.

Future Work

Currently SCHEHERAZADE can acquire a plot graph and execute it interactivity. Future work will involve the use of drama management to improve NPC event selection, which is currently shallow. We also note that the current knowledge structures do not support sophisticated textual descriptions of events, as seen in Interactive Fictions. We will continue to leverage the crowdsourcing paradigm to create richer knowledge structures that drive text-to-scene algorithms (Coyne, Bauer, & Rambow 2011).

Conclusions

In this paper, we introduce the problem of automatic creation of interactive narratives. The SCHEHERAZADE system allows a human user to specify any topic that he or she wishes to convert into an interactive experience. Our system overcomes knowledge bootstrapping issues by tapping the experiences and creativity of humans via crowdsourcing services to automatically construct an executable plot graph. It is shown the system can handle noisy input narratives with omitted events and several variations of the same topic. The work presented here is an important first step toward the goal of creating AI systems that minimizes the cost of the authoring for interactive narratives. We envision that reduced authoring cost will one day bring about large-scale applications of AI techniques previously considered intractable to build.

Acknowledgements

We gratefully acknowledge the support of the U.S. Defense Advanced Research Projects Agency (DARPA) for this research.

References

Chambers, N. and Jurafsky, D. 2009. Unsupervised learning of narrative event chains. *Proc. of ACL/HLT 2009.*

Chen, S., Nelson, M., and Mateas, M. 2009. Evaluating the Authorial Leverage of Drama Management. *Proc. of the 5th Conf. on AI and Interactive Digital Entertainment*.

Coyne, B., Bauer, D. and Rambow, O. 2011. VigNet: Grounding language in graphics using frame semantics. *Proc. of the ACL Workshop on Relational Models of Semantics*.

Elson, D.K., McKeown K.R. 2010. Building a Bank of Semantically Encoded Narratives. *Proc. of the 7th Int'l Conf. on Language Resources and Evaluation*, Malta.

Giannatos, S., Nelson, M. Cheong, Y. and Yannakakis, G. 2011. Suggesting new plot elements for an interactive story. *Proc. of the 4th Workshop on Intelligent Narrative Technologies*.

Li, B., Appling, D.S., Lee-Urban, S., and Riedl, M.O. 2012. Learning Sociocultural Knowledge via Crowdsourced Examples. *Proc. of the 4th AAAI Workshop on Human Computation*.

Magerko, B. 2005. Evaluating preemptive story direction in the Interactive Drama Architecture. *Journal of Game Development*.

Nelson, M. and Mateas, M. 2005. Search-based drama management in the interactive fiction Anchorhead. *Proc. of the 1st Conf. on AI and Interactive Digital Entertainment*.

Nelson, M. and Mateas, M. 2008. An Interactive game-design assistant. *Proc. of the 2008 International Conference on Intelligent User Interfaces*.

Nelson, M., Mateas, M. Roberts, D.L. and Isbell, C. 2006. Declarative optimization-based drama management in Interactive Fiction. *IEEE Computer Graphics and Applications*, 26, 30-39.

Orkin J., Roy, D. 2009. Automatic learning and generation of social behavior from collective human gameplay. *Proc. of the 8th Int'l Conf. on Autonomous Agents and Multiagent Systems*.

Quinn, A. and Bederson, B. 2011. Human computation: A survey and taxonomy of a growing field. *Proc. of the 2011 ACM SIGCHI Conf. on Human Factors in Computing Systems*.

Riedl, M.O., Stern, A., Dini, D. and Alderman, J. 2008. Dynamic experience management in virtual worlds for entertainment, education, and training. *International Transactions on System Science and Applications*, 3, 23-42.

Swanson, R. and Gordon, A. 2008. Say Anything: a massively collaborative open domain story writing companion. *Proc. of the 1st Int'l Conf. on Interactive Digital Storytelling*.

Treanor, M., Schweizer, B., Bogost, I., and Mateas, M. 2012. The micro-rhetorics of *Game-O-Matic*. *Proc. of the 2012 Conference on the Foundations of Digital Games*.

Weyhrauch, P. 1997. *Guiding Interactive Fiction*. Ph.D. Dissertation, Carnegie Mellon University.

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, 51-70.