Vignette-Based Story Planning: Creativity Through Exploration and Retrieval

Mark O. Riedl

School of Interactive Computing
Georgia Institute of Techology
85 Fifth Street NW, Atlanta, Georgia 30308, USA
riedl@cc.gatech.edu

Abstract. Storytelling is a pervasive part of our daily lives and culture that is being applied to computational systems for entertainment, education, and training. Because of the prevalence of story in non-interactive media such as books and movies, as well as interactive media such as computer games, automated generation of narrative content is an essential component of making computers creative. We describe artificial intelligence planning as a model of narrative creation and then detail how the retrieval and reuse of vignettes can facilitate the creation of narratives within the planning framework. Vignettes are fragments of story that exemplify certain familiar narrative situations. We show how this new approach to story generation is capable of exploring a larger space of creative solutions and can create more valuable stories.

Keywords. Story generation; exploratory creativity; planning; case-based reasoning

1 Introduction

Storytelling is a pervasive part of our daily lives and culture. Storytelling is particularly prominent in entertainment, where stories can be viewed as artifacts to be consumed by an audience. Story also plays a role in education and training, where stories and scenarios can be used to illustrate and guide. The production of these artifacts – stories and scenarios – is a primary activity in the entertainment industry and also a significant bottleneck in the educational and training industries. In an "on-demand" society, waiting for periodic updates to serial narratives – weekly television series, movie series, and novels – is not considered ideal. Likewise, players of computer games that rely on stories and quests can complete quests faster than design teams can create new quests. How do we handle the situation in which content consumption is scaled up to the point where content consumption outpaces content production? One way to overcome the bottleneck of content production is to instill in a computer system the creative ability to generate new content.

Because of the prevalence of story in non-interactive media such as books and movies, as well as interactive media such as computer games, we concern ourselves with the automated generation of narrative content. The issue is whether an automated story generation system can be considered creative enough or skilled enough to be trusted to produce content – stories – that will be experienced by users. More generally, the output of a creative system, such as an automated story generation system, must be *novel*, *surprising*, and *valuable* [1]. Whether an artifact is valuable is subjective. For the purposes of this paper, we will consider the minimal requirements for a story artifact to be considered valuable if it (a) meets the intended purpose of its creation and (b) is sufficiently mimetic – appearing to resemble reality, but in a way that it is more aesthetically pleasing than reality. In brief, stories should be novel, but not so novel that they are unrecognizable [2].

Two well-recognized approaches to story generation are planning and "knowledge-intensive" [3] techniques such as case retrieval. In this paper we present a step towards automated story generation that combines planning and the retrieval of narrative fragment – which we call *vignettes* – that are known to be "good" examples of mimetic situations.

2 Related Work

Boden [1] distinguishes between creativity as exploration and creativity as transformation. Likewise, narrative generation systems can be categorized roughly with regard to whether they treat the problem of creating narrative content as a problem of search or the problem of adapting existing knowledge and cases to new contexts.

Search based narrative generation approaches include Tale-Spin [4], which uses a simulation-like approach, modeling the goals of story world characters and applying inference to determine what characters should do. Dehn [5] argues that a story generation system should satisfy the goals of the human user. That is, what outcome does the user want to see? The Universe system [6] uses means-ends planning to generate an episode of a story that achieves a user's desired outcome for the episode. More recent work on narrative generation attempts to balance between character goals and human user goals [7; 8]. Further work on story planning addresses expanding the space of stories that could be searched [9].

Treating the problem of narrative generation as adapting existing knowledge has lead to variety of approaches that use case-based reasoning and/or analogy. Minstrel [10] implements a model of cognitive creativity based on routines for transforming old stories into new stories in new domains. ProtoPropp [3] uses case-based reasoning to generate novel folk tales from an ontological case base of existing Proppian stories. Mexica [11] uses elements of both previous story retrieval and means-ends planning.

Case-based reasoning (c.f. [12]) has been found to be related to creativity [1; 13]. The story planning algorithm that we describe in Section 3 is reminiscent of a certain class of case-base reasoners called transformational multi-reuse planners. Transformational multi-reuse planners are planners that attempt to achieve given goals by identifying and retrieving solutions to similar, previously solved problems. A multi-reuse planner may attempt to combine several cases in order to solve the given problem. Our story planning algorithm is similar in many ways to [14] and [15]. We compare and contrast our algorithm to these at the end of Section 3.2.

3 A Computational Approach to Generating Stories

We view story generation as a problem-solving activity where the problem is to create an artifact – a narrative – that achieves particular desired effects on an audience. We favor a general approach where we model the story generation process as planning (c.f. [6], [7], [8], [9]). Conceptually a planner can be thought as an approach to problem solving in which *critics* [16] are applied to incomplete solutions until a complete solution is found. Each critic inspects a plan for a different type of flaw. We conceive of a critic as containing two parts: a flaw recognizer and a flaw repairer. The flaw recognizer component of each critic is invoked after any new potential solution plan is generated. If a critic recognizes a flaw in the plan, it annotates the plan to indicate that the plan cannot be considered a valid solution because of a flaw. The planner chooses an incomplete plan to work on and chooses a flaw to work on. The appropriate critic's flaw repairer routine is invoked, which proposes zero or more new plans in which the flaw is repaired (and often introducing new flaws). For example, one type of critic recognizes and repairs *open condition flaws*. An open condition flaw exists when an action (or the goal state) in a plan has a precondition that is not established by a preceding action (or the initial state). The critic can repair this flaw by applying one of the following repair strategies:

(i) Selecting an existing action in the plan that has an effect that unifies with the precondition in question.
(ii) Selecting and instantiating an operator from the domain operator library that has an effect that unifies with the precondition in question.

The planner uses special annotations called *causal links* to indicate when open conditions are satisfied. A causal link establishes the causal relationship between two actions in the case that the effects of the former action establish a condition in the world necessary for the latter action to execute successfully. The set of critics used in conventional planners such as [17] assure plan that plans are sound, meaning that they are guaranteed to execute successfully in the absence of unanticipated changes in the world. However, stories are much more than just ways of achieving an intended outcome in the most efficient manner. Stories should meet the expectations of the audience. This may mean putting in details that are aesthetically pleasing even if they are not strictly necessary.

When humans write stories, they call on their lifetime of experiences as a member of culture and society. A computer system that generates stories does not have access to this wealth of information. As a way of mitigating this handicap, a computer system can be provided with a wealth of knowledge in the form of traces of previous problem-solving activities or libraries of previous solutions (e.g. stories). One "knowledge intensive" approach [3] is to use a form of case-based reasoning. Our story planning algorithm achieves longer and more mimetic narrative sequences by accessing a library of vignettes – fragments of stories that capture some particular context. We do not presume to know how these vignettes were created, only that we have the solutions and that they have favorable mimetic qualities.

```
Vignette:
Steps: 1: Start-Battle (?c1 ?c2 ?place)
       2: Wound (?c1 ?c2)
       3: Wound (?c1 ?c2)
       4: Mortally-Wound (?c2 ?c1)
       5: Die (?c1)
       6: End-Battle (?c1, ?c2)
Constraints: (character ?c1)
             (character ?c2)
             (stronger ?c2 ?c1)
Ordering: 1→2, 2→3, 3→4, 4→5, 4→6
Causation: 1→(battling ?c1 ?c2)→2
           1→(battling ?c1 ?c2)→3
           1→(battling ?c1 ?c2)→4
           1→(battling ?c1 ?c2)→6
           4→(mortally-wounded ?c1)→5
Variable-constraints: ?c1 ≠ ?c2
Effects: (battling ?c1 ?c2)
         (not (battling ?c1 ?c2))
         (wounded ?c2)
         (mortally-wounded ?c1)
         (not (alive ?c1))
```

Fig. 1. An example vignette data structure describing a battle in which a weaker character (*?c1*) takes on a stronger character (*?c2*) and dies.

3.1 Vignettes

We use the term *vignette* to refer to a fragment of a story that represents a "good" example of a situation and/or context that commonly occurs in stories [18]. For example, a library of vignettes would contain one or more specific instances of bank robberies, betrayals, cons, combat situations, etc. It is important to note that the library contains specific examples of these situations instead of general templates. The implication of the existence of this library is that a story generator does not need to "reinvent the wheel" and thus does not need the specialized knowledge required to be able to create specialized narrative situations. Vignettes are fragments of story structure. Unlike a script, a vignette can be thought of as a fragment of a narrative – a partially ordered set of events that are perceived to be essential in presenting a narrative situation. How does one know what actions should be included in the vignette and which can be left out? We use the minimal vignette rubric: a *minimal vignette* is one in which removing any one action from the vignette causes it to no longer be considered a good example of the situation and/or context it was meant to represent.

Computationally, vignettes are stored as plan fragments. As a plan fragment, it is possible that some actions do not have to have all of its preconditions satisfied. This is a way of saying that it is not important how the situation is established or even why, but once the conditions are established certain things should happen. Vignette plan fragments do not reference specific characters, objects, or entities so that a planner can fit the vignette into new story contexts by making appropriate assignments. To ensure illegal or non-sense assignments are not made, co-designation and non-co-designation variable constraints are maintained. Fig. 1 shows an example vignette capturing a very simple combat between two characters where one character (represented by the variable *?c2*) is stronger than the other (represented by the variable *?c1*). The weaker character wounds the stronger character twice before the stronger character delivers a mortally wounding blow. Finally, the mortally wounded character dies of its wounds. This vignette could be used in any plan in which a character must become wounded, mortally wounded, or dead.

3.2 Planning with Vignettes

The Vignette-Based Partial Order Causal Link (VB-POCL) planner is a modification of standard partial order planners to take advantage of the existence of a knowledge base of vignettes. The VB-POCL planning algorithm is similar to other multi-reuse case-based planners such as [14] and [15] in that it modifies the open condition critic by adding a third strategy for repairing open condition flaws:

(iii) Retrieve and reuse a case that has an action with an effect that unifies with the precondition in question.

Given an action in the plan that has an unsatisfied precondition VB-POCL non-deterministically chooses one of the three above strategies. Strategies (i) and (ii) are performed in the standard way (c.f., [17]). If strategy (iii) is selected, VB-POCL's open condition critic retrieves a vignette that has an action with an effect that will satisfy the unsatisfied precondition. The retrieval process is one of identifying a vignette in the knowledge base that has an action that has an effect that unifies with the open precondition that the open precondition critic is trying to satisfy. But the critic does not splice the vignette into the flawed plan. Instead, another critic recognizes that the plan if flawed because the vignette has not been fitted into the plan and annotates the plan with a new type of flaw called a *fit flaw*. A fit flaw is repaired only when all the actions in the retrieved vignette have been instantiated in the plan. Repairing a fit flaw is a process of selecting an action from the retrieved vignette and adding it to the new plan (or selecting an existing action in the plan that is identical to the selected action to avoid unnecessary action repetition) with all relevant causal links, temporal links, and variable bindings.

It may take several invocations of the fitting critic to completely repair a fit flaw. This may seem more inefficient than just adding all vignette actions to the plan at once. There are three advantages to iterative fitting. First, it is easier to recognize and avoid action repetition. Second, it allows other critics to observe the plan at intermediate stages of fitting in case there are interesting synergies between critics. For example, fitting may lead to the creation of new open condition flaws that in turn are repaired through conventional planning (strategies i and ii) or by retrieving new vignettes (strategy iii). Third, problems in the fitting process can be identified sooner in case the strategy must be abandoned. One of the interesting properties of VB-POCL – shared with other case-based planning algorithms – is that it can operate when there are no applicable vignettes available; the algorithm can fall back on conventional planning. If applicable vignettes are available, plan-space search control heuristics are required to prevent a potential explosion of open conditions.

VB-POCL is a variation on one type of case-based reasoning called *transformational multi-reuse planning* (c.f. [14], [15]). Transformational multi-reuse planners attempt to reuse components of solutions to similar problems to solve new problems. VB-POCL is a variation on transformational multi-reuse planning; vignettes are not solutions to previous problems and VB-POCL does not attempt to learn to solve problems from past examples. That is, VB-POCL does not *retain* its solutions because they are neither vetted nor minimal, as vignettes are required to be. VB-POCL relies on certain assumptions that make it different from other case-based reasoning techniques. First VB-POCL assumes that vignettes are minimal. Because of this, VB-POCL doesn't stop fitting a vignette until all actions in the vignette are present in the new plan, even if some actions do not serve a causal purpose. This is in contrast to other case-based reasoning techniques that discard actions that are strictly unnecessary from the perspective of achieving a goal state. Second, VB-POCL assumes that vignettes in the library are in the domain of the story being generated. The implication of this assumption is that the planner does not need to deliberate about the cost tradeoff between using standard planning versus retrieval and reuse (which is otherwise very high). VB-POCL non-deterministically chooses between flaw repair strategies (i and ii) and (iii), meaning that it applies all strategies to each and every flaw by branching the search space and exploring each branch in turn. This is not practical if vignettes require extensive modifications for reuse. To support this property of VB-POCL, we use an offline algorithm based on analogical reasoning to pre-process the vignette knowledge base and transform vignettes from their native story world domains to the domain of the new story to be generated [18]. The vignette transformation process is overviewed in Section 3.4.

3.3 Example

To illustrate the VB-POCL planning algorithm, we provide an example of how the planner could use the vignette shown in Fig. 1. Suppose we wanted a story set in J.R.R. Tolkein's Middle Earth. The story world is in the state in which one character, called *Enemy*, has in his possession a Silmiril – a precious magical stone. The outcome, provided by the human user, is that another character, called *Hero*, gains possession of the Silmiril. The planner starts by non-deterministically choosing to satisfy the goal by having Hero take the Silmiril from Enemy. This requires that the Enemy not be alive. The planner could use the vignette from Fig. 1 here by retrieving it and binding Enemy to *?c1*. Note that this strategy will eventually fail because it would require a character stronger than Enemy. Instead the planner solves the problem chooses to instantiate an action, *Die-of-Infection*, that causes Enemy to not be alive. This requires that Enemy be superficially wounded. Here VB-POCL retrieves the vignette from Fig. 1 because it has an action that can have the effect (once variables are bound) of causing Enemy to become wounded.

Each vignette action is spliced into the new story plan one at a time. The temporal and causal relationships between vignette actions are maintained in the new plan. However, any time a new action is added to the plan, causal threats may arise in which the new action potentially undoes an existing causal relationship. These causal threats are handled by imposing additional temporal constraints on actions in the plan (or if the threat cannot be resolved, the planner backtracks) [17]. For example, when *Die(Hero)* is spliced into the story plan, it must be temporally ordered after *Take* to

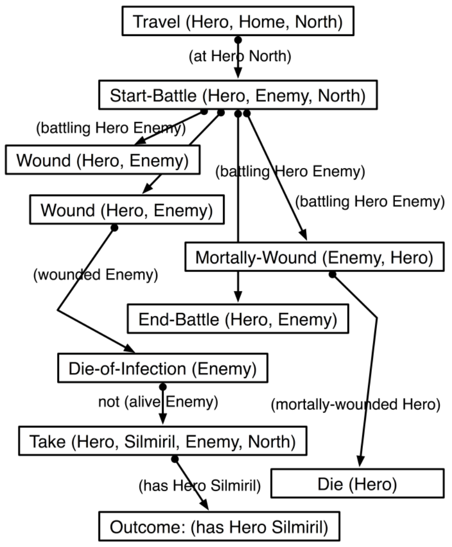

Fig. 2. An example story plan generated by VB-POCL. Boxes represent actions or events and arrows represent causal relationships between those actions.

avoid inconsistencies; a dead characters cannot perform actions. The order that actions are chosen from the vignette and spliced into the plan does not matter.

When a vignette is retrieved, one action is chosen non-deterministically to be the *satisfier action*. This is the action that will be used to satisfy the original open condition flaw that triggered the case retrieval in the first place. In the example, the satisfier action is one of the *Wound* actions because it has the effect of causing the Enemy to become wounded. When the satisfier action is spliced into the plan, the planner takes the extra step of causally linking the satisfier to the action with the original unsatisfied precondition – in this case, Enemy needing to be superficially wounded in order to die of an infection – that triggered the vignette's retrieval in the first place. Thus the open condition flaw is finally repaired.

The vignette is fairly self-contained, but the vignette action, *Start-Battle* does require that the planner establish that both Hero and Enemy are at the same place, which in this case the North. This precondition is satisfied in the normal way, by instantiating an action in which Hero travels to the North (Enemy is already there). The final story plan is shown in Fig. 2. Boxes are actions and arrows represent causal links. A causal link indicates how an effect of one step establishes a world state condition necessary for a precondition of latter steps to be met. For clarity, only some preconditions and causal links on each action are shown.

3.4 Vignette Transformation

VB-POCL assumes a library of vignettes that are already in the domain of the story to be generated. A domain is a set of propositions that describe the world, including characters, and a set of operator templates that described what characters can do and ways in which the world can be changed. In the example, the domain describes characters such as Hero and Enemy and operators such as *Travel* and *Wound*. However, we may want the story planner to have access to vignettes from other domains, especially if our new story is set in an unique and specialized story world domain. To transfer vignettes between domains, one must first find analogies between domains. This differs from the problem of finding analogies between stories because there is not a second instance of a story to compare. Instead, we search for analogies between domains and use that information to translate a known vignette from one domain to another. The transfer process is summarized as follows. A *source vignette* is a vignette in an arbitrary domain, called the *source domain*. The *target domain* is the domain of the story to be generated. For each action in the source vignette, the far transfer algorithm searches for an action in the target domain that is most analogical. The search involves a single-elimination tournament where target domain actions compete to be the most analogical according to the Connectionist Analogy Builder (CAB) [19]. The winner of the tournament is the target domain action most analogical to the source domain action. The result is a mapping of source domain actions to target domain actions that can be used to translate a source vignette into a target domain through substitution. Translated vignettes may have gaps where translation is not perfect. This is not a problem because the VB-POCL will recognize this and fill in the gaps via planning. Applying this process to all vignettes in a library results in a new library in which all vignettes are in the proper domain. See [18] for a more detailed description of the algorithm.

4 Discussion

One of the interesting properties of VB-POCL is that vignette retrieval can result in story plans in which there are actions that are not causally relevant to the outcome. Trabasso [20] refers to actions that are causally irrelevant to the outcome as dead-ends. In the example above, the causal chain involving Enemy mortally wounding Hero and then Hero dying appears to be a dead-end because those actions do not contribute to Hero acquiring the Silmiril. Psychological studies indicate that dead-ends are not remembered as well as actions that are causally relevant to the outcome [20], suggesting that dead-ends should be avoided. For example, a battle in which a single wound was inflicted on Enemy would have sufficed, and this is what planners such as [7; 8] and [17] would have settled on. However, human authors regularly include dead-end events in stories suggesting some importance to dead-ends. We hypothesize that there are certain mimetic requirements to be met in any story and that dead-ends can serve this purpose. For example, we assume that a combat scenario in which many blows of varying strengths are exchanged is more interesting than a combat in which a single blow is dealt. Interestingly, what may be a dead-end causal chain to the story planner may not be considered a dead-end by a human reader, and vice versa. That is, the reader may interpret the example story as a *tragedy* and consider the death of Hero as one of *two* primary causal chains, whereas the planner's representation contains only *one* causal chain that leads to the human user's imposed outcome (Hero has the Silmiril). More research needs to be done to create intelligent heuristics to recognize when dead-ends (from the planner's perspective) are favorable, tolerable, or damaging.

Does VB-POCL improve the computational creativity of planning-based story generation approaches? Planning-based story generators treat creativity as exploratory search [9], which is one of two perspectives on creativity offered by Boden [1]. VB-POCL brings exploratory and transformational creativity closer together. Incorporating case-based retrieval into a planning framework suggests that transformation is a special instance of exploration. Conceptually, we can consider plot generation as a search through the space of all possible narratives, where a narrative is a set of temporally arranged events that change the story world (for the purposes of completeness we also consider the empty story). In the space of all possible narratives, narratives that are adjacent differ in just one detail – an extra event or a different temporal ordering of events. One can walk the space of all possible narratives, beginning with the empty narrative by applying the operator, *add-event(e, c)*. This operator moves one from a narrative to an adjacent narrative that differs in that it contains one additional event, *e*. The parameter *c* is a set of constraints that unambiguously positions *e* temporally relative to other events in the narrative. The operator *add-event* is an abstraction of the iterative process used by planners in the search for the first visited structure to which critics do not attribute flaws.

Because of the iterative nature of vignette fitting in VB-POCL, a vignette can be viewed as a strategy for guiding the walking of the space towards a sub-space of stories that are more valuable. Because vignettes are made retrievable by transforming them via analogical reasoning, we hypothesize that transformational creativity may be a special case of exploratory creativity. That is, transformation processes *short-circuit* exploration by using knowledge from seemingly unrelated domains to specify a target or direction for search. In the case of VB-POCL, however, one could claim that some or all of the creativity occurs prior to exploration in the offline process that transformed vignettes into the correct – applicable – domain via analogical reasoning.

These vignette-guided walks also extend beyond the boundaries of the space that can be walked by conventional planning techniques – specifically visiting stories in which there are actions that are not causally necessary. As noted in [9], expanding the space that can be explored provides an opportunity to find more solutions that are valuable. We believe that VB-POCL searches a space of stories that has more valuable and mimetic solutions. This is partially achieved by attempting to retrieve vignettes whenever possible. Future work involves identifying heuristics that can determine when retrieving a particular vignette is more likely to lead to a valuable solution. Future work also involves incorporating additional properties of character and story into the algorithm such as character intentionality [7; 8], character personality [9], and emotion [11].

We find planning to be a valuable model for story generation, in general. One reason for this is that plans are reasonable models of narrative [21]. But also planners walk the space of possible narratives in search of a solution that meets certain qualities. VB-POCL extends the general planning algorithm by retrieving and reusing vignettes. This is a strategy for tapping into the experiences of other presumably expert story authors. Interestingly, VB-POCL can explore a greater space of stories because it can consider story plans that have action that are not causally necessary to reach some given outcome. We believe that some of these stories will be more valuable because of the mimetic qualities of the vignettes and the potential for these stories to possess both global novelty and localized familiarity.

References

1. Boden, M.: *The Creative Mind: Myths and Mechanisms, 2nd Edition*. Routledge, New York (2004).
2. Giora, R.: *On Our Mind: Salience, Context, and Figurative Language*. Oxford University Press (2003).
3. Gervás, P., Díaz-Agudo, B., Peinado, F., Hervás, R. Story Plot Generation Based on CBR. *Journal of Knowledge-Based Systems*, 18(4-5), 235-242 (2006).
4. Meehan, J.: *The Metanovel: Writing Stories by Computer*. Ph.D. Dissertation, Yale University (1976).
5. Dehn, N.: Story Generation after Tale-Spin. In: *7th International Joint Conference on Artificial Intelligence* (1981).
6. Lebowitz, M.: Story-Telling as Planning and Learning. *Poetics*, 14, 483—502 (1985).
7. Riedl, M.O. *Story Generation: Balancing Plot and Character*. Ph.D. Dissertation, North Carolina State University (2004).
8. Riedl, M.O., Young, R.M.: An Intent-Driven Planner for Multi-Agent Story Generation. In: *3rd International Joint Conference on Autonomous Agents and Multi Agent Systems* (2004).
9. Riedl, M.O., Young, R.M.: Story Planning as Exploratory Creativity: Techniques for Expanding the Narrative Search Space. *New Generation Computing*, 24(3), 303—323 (2006).
10. Turner, S.: *MINSTREL: A Computer Model of Creativity and Storytelling*. Ph.D. dissertation, University of California , Los Angeles (1992).
11. Pérez y Pérez, R. and Sharples, M.: Mexica: A Computer Model of a Cognitive Account of Creative Writing. *Journal of Experimental and Theoretical Artificial Intelligence*, 13(2), 119—139 (2001).
12. Kolodner, J.: *Case-Based Reasoning*. Morgan Kaufmann Publishers (1993).
13. Kolodner, J.: Understanding Creativity: A Case-Based Approach. In: *1st European Workshop on Case-Based Reasoning* (1993).
14. Francis, A.G., Ram, A.: A Domain-Independent Algorithm for Multi-Plan Adaptation and Merging in Least-Commitment Planners. In: *AAAI Fall Symposium on Adaptation of Knowledge for Reuse* (1995).
15. Britanik, J., Marefat, M.: CBPOP: A Domain-Independent Multi-Case Reuse Planner. *Computational Intelligence*, 20, (2004).
16. Sacerdoti, E.D.: A Structure for Plans and Behavior. Elsevier, New York (1977).
17. Weld, D.: An Introduction to Least Commitment Planning. *AI Magazine*, 15(4), 27-61 (1994).
18. Riedl, M.O., León, C.: Toward Vignette-Based Story Generation for Drama Management Systems. In: *2nd International Conference on Intelligent Technologies for Interactive Entertainment, Workshop on Integrating Technologies for Interactive Story* (2008).
19. Larkey, L.B. and Love, B.C.: CAB: Connectionist Analogy Builder. *Cognitive Science,* 27, 781—794 (2003).
20. Trabasso, T., van den Broek, P.: Causal Thinking and the Representation of Narrative Events. *Journal of Memory and Language*, 24, 612-630 (1985).
21. Young, R.M.: Notes on the Use of Plan Structures in the Creation of Interactive Plot. In: *AAAI Fall Symposium on Narrative Intelligence* (1999).