Generating Story Analogues

Mark O. Riedl
School of Interactive Computing
Georgia Institute of Technology
Atlanta, Georgia, USA
riedl@cc.gatech.edu

Carlos León
Depto. de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid
Madrid, Spain
cleon@fdi.ucm.es

Abstract

In this paper, we describe a computational system that generates story analogues based on previous stories. Unlike many previous works on story generation that attempt to produce a story artifact from a set of high-level specifications, we describe an approach that generates relatively novel stories by transforming existing stories to new contexts. We describe an algorithm for using analogical reasoning to find similarities between story contexts in order to map events from an existing story to a novel context.

Introduction

The interactive entertainment and entertainment computing research areas are interesting because they blend computational sciences, engineering, and creative endeavors. Recently, there has been a growing interest in *computational creativity*, the notion that computational systems can be developed that bear abilities resembling human creativity. We will not delve deeply into what it means to be creative except to say that a creative system – human or computer – takes some set of input parameters and constraints and produces an *artifact* that is *relatively* novel, valuable, and unexpected (Boden 2004).

In this paper, we describe a computational system that generates story analogues based on previous stories. Unlike many previous works on story generation that attempt to produce a story artifact from a high-level set of input specifications (c.f., (Meehan 1976; Lebowitz 1987; Riedl and Young 2004)), we describe an approach that generates relatively novel stories by transforming existing stories to new contexts. For example, given a story set in the American Old West, we might be used to produce an analogous story set in a Science Fiction future. By "analogous" we mean possessing deep structural similarities. In that sense, our work is more closely aligned with case-based reasoning approaches to story generation (c.f., (Turner 1992; Pérez y Pérez and Sharples 2001; Gervás et al. 2005; Swartjes, Vromen, and Bloom 2007)). Case-based reasoning (Kolodner 1993a) and analogical reasoning (Falkenhainer, Forbus, and Gentner 1989; Larkey and Love 2003) have been posited to be related to creativity (Boden 2004; Kolodner 1993b).

Computational creativity is relevant to the interactive entertainment community because there is value in computational entertainment products being able to generate content on-demand. On-demand content is the idea that increased value is derived from a product if it is individualized or permuted to take into account more personalized or immediate context. For example, we can begin to consider novel applications such as automated quest generation for role-playing games such that the player can specify criteria for interesting experiences (e.g., "I like to slay dragons") and receive tailored gameplay content. Another application is a "storytelling companion" that can assist a user in the creation of novel content. In general we speculate, that there is an advantage to being able to produce novel narrative and gameplay content to increase engagement and individualization of gameplay experience. We do not explore specific applications further. Instead, we introduce a technological solution to generating novel story and quest content from existing story content, which can be considered a step toward opening up computational creativity to entertainment computing applications.

Specifically, we describe a process that generates stories by taking existing stories, or fragments of stories, and adapting them to novel contexts. Our goal is to strike a balance between novelty and familiarity. In brief, stories should be novel, but not so novel that they are unrecognizable (Giora 2003). We use analogical reasoning to find similarities between the contexts – domains – in which different stories are set in order to produce a transformation function that transforms a story set in one domain into a skeleton of a story set in another domain. Planning is used to fill in any gaps.

Related Work

Story generation has been the emphasis of numerous research projects. Our work and most related work are primarily concerned with plot – what stories are about – as opposed to discourse – how to tell a story. Story generation systems include story world character simulations (Meehan 1976), planning (Lebowitz 1987; Riedl and Young 2004), and case-based reasoning (Turner 1992; Gervás et al. 2005; Pérez y Pérez and Sharples 2001; Riedl and Sugandh 2008). The work described in this paper looks at story generation

from the perspective of adapting existing stories into novel contexts either as a stand-alone generation process or as an input into a case-based plot planner such as (Riedl and Sugandh 2008).

Unlike the system described in this paper, the majority of work on story generation approaches the problem of generating a story from building blocks smaller than complete stories, such as planning operators or cases/vignettes. Case-based reasoning has consequently been a popular method for generating stories. Minstrel (Turner 1992) is a problem-solving approach to story generation where, given a problem, transform-recall-adapt methods (TRAMS) are invoked to search for additional knowledge, which may be parts of previous stories, to apply to the problem. MEXICA (Pérez y Pérez and Sharples 2001) uses phases of engagement and reflection to pick next actions based on similarity with previous stories and repair inconsistencies, respectively. Gervás and colleagues (Gervás et al. 2005) describe an ontological approach where story "functions" – patterns of plot development originally devised by Propp (Propp 1968) – are used as the building blocks. VBPOCL (Riedl and Sugandh 2008) blends partial-order planning and transformational multi-reuse planning to solve the problem of generating a sequence of plot events.

Analogical reasoning has been applied to stories; the "Karla the Hawk" and "Zerdia" stories are often used to illustrate an analogical reasoning system's ability to find deep structural similarities in first-order logical representations of narratives (c.f., (Falkenhainer, Forbus, and Gentner 1989) and (Larkey and Love 2003)). However, the problem we are addressing here is not whether we can detect analogical stories, but whether we can *produce* an analogue to a given story. Other work on analogies in stories attempts to identify analogical character roles in different stories (Hervás et al. 2006). We see identification of analogous characters as an important part of analogical transformation, but also look for analogies in the domain actions.

We use the Connectionist Analogy Builder (CAB) (Larkey and Love 2003) to reason about analogies, as part of our larger system. CAB is an implementation of a cognitive model of analogy that finds correspondences between subcomponents of two representations. Given two directed graph representations, CAB detects correspondences among sub-structures, producing a mapping between corresponding elements through a constraint satisfaction approach to comparison. Briefly, CAB iteratively constructs hypotheses about graph node correspondences based on whether they play compatible roles in the two representations. The role of a node is determined relative to other nodes. Hypothetical mappings are proposed and each mapping can "vote" on whether to strengthen or weaken the weight of mappings of nearby nodes. Poor correspondences are punished and their weights drift toward 0 while good correspondences are rewarded and their weights drift towards 1. Further details on CAB processing is beyond the scope of this paper.

Analogical Transformation of Stories

Between-domain analogy occurs between concepts that embrace dissimilar knowledge and require deep structural similarities. This is opposed to *within-domain* analogy, in which concepts share the same – or very similar – knowledge and can be performed using surface similarities. In this work, we are concerned with between-domain analogies about stories. A domain is both a story world description and a set of rules for actions and events that are legal to occur in that story world.

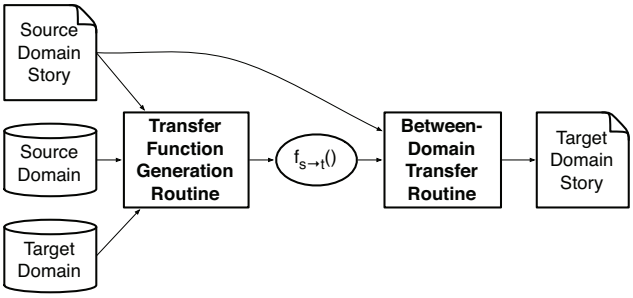

Figure 1: The analogical story transformation process.

In this section we describe how to use analogical reasoning to engage in between-domain transfer. That is, to transform a source story that exists in a source domain to a target story that exists in a target domain. Analogy-finding algorithms such as the Structure-Mapping Engine (SME) (Falkenhainer, Forbus, and Gentner 1989) and CAB have been demonstrated to be able to find analogies in stories when they exist. However, our problem is different: we have a story in a source domain, but no story in the target domain; we are solving the problem of transforming a story from a source domain into a new story in the target domain.

The transformation process is visualized in Figure 1. The inputs into the process are the source story p_s represented as a plan (see below), the source domain D_s, and the target domain D_t. The first stage is to produce a transfer function, $f_{s \mapsto t}$, that maps concepts in D_s into concepts in D_t. Specifically, $f_{s \mapsto t}$ maps actions and ground symbols in D_s to actions and symbols in D_t. The second stage is to use the transfer function to map the concepts in our source narrative plan p_s into a new data structure that is a narrative plan in the target domain. Note that in either the first two stages failure to find and apply mappings can leave gaps in the target story plan p_t. For example, an action in p_s may not have an analogue in D_t, resulting in the case where the solution story p_t has a missing action. A partial order planner, operating in the target domain, fills in the gaps.

Narrative Representation

We represent narratives as partially-ordered plans because of similarities between the knowledge captured in a plan data structure and the type of knowledge present in narrative at the plot level. Narratives at the plot level and plans both contain knowledge about action (or events that effect change to a world), temporality, and causality (Young 1999). More specifically, we represent narratives as a partial ordering of instantiated operators, each representing an action that a story world character takes, or an event that changes the story world but is otherwise unintentional.

A domain $D = \langle S, \Lambda \rangle$ is a tuple such that S is a description of the state of a world and Λ is the set of all possible operations that can change the world. We assume that the operators in Λ are *ground operators*, meaning they do not contain variables. However, our techniques would also work if operators did have variables – ground operators can be generated from operators with variable parameters by creating instances for all possible legal bindings of ground operators to variables.

A narrative plan $p = \langle I, A, O \rangle$ is a tuple such that I is a description of the initial state, A is a set of ground operators – called actions – and O is a set of temporal ordering constraints of the form $a_1 < a_2$ where $a_1, a_2 \in A$ and a_1 necessarily precedes a_2 in the story. A narrative plan is said to be of domain $D = \langle S, \Lambda \rangle$ if $a_i \in \Lambda$ for all $a_i \in A$ and $I \subseteq S$.

We define a plan operator as a tuple $\langle h, P, E \rangle$, where h is the head of the action, defining its name and its arguments, P is the set of ground preconditions, a set of propositions that define the previous state needed for the action in the story to be performed, and E is the set of ground effects, that is, the set of propositions that are made true when the action is performed (e.g., propositions added to the world state and negated propositions deleted from the world state under the closed world assumption).

Pre-Processing

In our approach, we use the Connectionist Analogy Builder (Larkey and Love 2003) as a component in our first stage. CAB works with graphs, and thus we need to translate STRIPS-like operators to graphs. To translate an operator into a graph we create a root node with the head of the action, and children nodes for the arguments of that action. We add as children to the root node a *precondition* node and an *effect* node, whose children are the propositions of the sets P and E, respectively. The graph is completed by adding selected propositions from the domain state information. The *precondition* and *effect* nodes provide structure to the graphs and also provide consistent terms that can be leveraged by CAB that help avoid very unlikely matches.

State information provides context about the ground symbols that is essential for finding correct analogies between operators. Context is important because many operators have very similar structures (e.g., propositions that become negated) and CAB would be unable to find the difference between any two operators with surface-level structural similarity without additional contextual information. Our process choses context information to be included in the graph by picking world state propositions that relate the ground symbols referenced by the operator. An example operator is in Figure 2 and resultant graph is in Figure 3. Gray nodes represent ground symbols. The white nodes at the bottom of the figure represent state information. Some nodes and links are omitted for space. Notably, special nodes (not shown) capture cardinality of relations. For example, the ordering of parameters in *stronger(king2, princess)* is semantically important.

```
Action:
   Head:  Capture (king2, princess)
   Precondition:  not(captured(king2)),
                  not(captured(princess)),
                  stronger(king2, princess)
   Effect:  captured(princess),
            captured-by(princess, king2)
Context knowledge:
   stronger(king1, king2), enemies(king1, king2),
   daughter(princess, king1), has(king2, spear1)
```

Figure 2: The operator, Capture(king2, princess) in STRIPS representation with ground symbols.

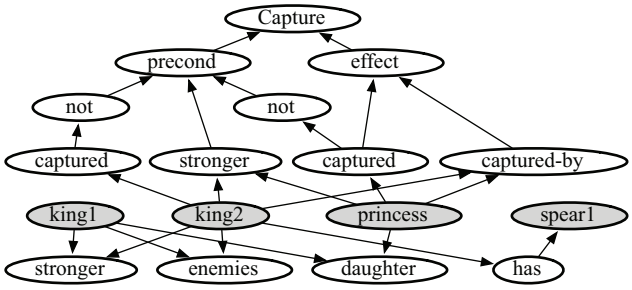

Figure 3: The operator, Capture(king2, princess), in graphical form. For clarity, some links and nodes are left out.

Generating the Transfer Function

With the operators represented as graphs, the transformation algorithm, given a source story plan $p_s = \langle I, A, O \rangle$ and a target domain $D_t = \langle S, \Lambda_t \rangle$, computes a concept map of the form $map = \{a_{s_1} \Leftrightarrow a_{t_1}\}...\{a_{s_n} \Leftrightarrow a_{t_n}\}$ as follows. The algorithm iterates over the set of actions in the source story p_s according to the temporal ordering of actions in the plan. First, it computes the world state for the source domain $state_s$ and target domain $state_t$. During the first iteration of the process, $state_s$ is the initial state of p_s and $state_t$ is the initial state of D_t. Then, for the current action a_{s_i} from the source story, we invoke the `find-best` sub-procedure, which computes the best action from Λ_t (call it a_{t_i}) that corresponds to a_{s_i} given $state_s$ and $state_t$ as contextual cues. If `find-best` returns a value we create an entry in our transformation mapping. The algorithm then updates $state_s$ and $state_t$ by applying the effects of a_{s_i} and a_{t_i}, respectively. The update is important because the actions in the source story progressively modify the world state and the domain initial state becomes less relevant. Without progressively updating the world state the algorithm becomes more prone to errors. The process repeats until every action in the source story has a mapping to an operator in the target domain, or to *nil*. The transformation process can fail to find an analogues for actions, however, leaving gaps in the target domain story plan. We describe how to handle missing actions in a later section.

The `find-best` routine is responsible for finding the best operator in the target domain for an operator in the source domain. It is important to note that we have not de-

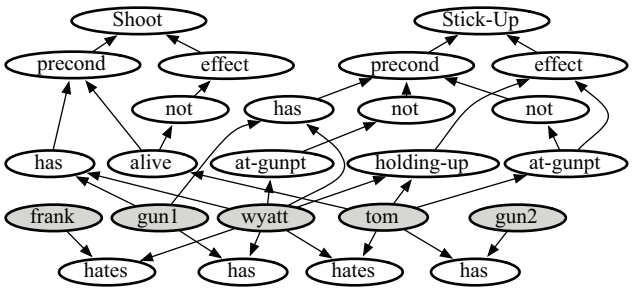

Figure 4: Two target domain operators, Shoot(Wyatt, Tom, gun1) and Stick-Up(Wyatt, Tom, gun1), merged into a graph. For clarity, some links and nodes are left out.

veloped a test for optimality, and thus there is not an exact way for finding the best mapping. However, we refer to the "best" or "optimal" operator when, intuitively, that operator would be chosen by a human. We implement `find-best` as a single-elimination competition of target domain operators. The routine compares a source action with a randomly chosen pair of target actions. Paired target domain actions are merged into the same directed graph by linking operator graph nodes to a shared set of ground symbol nodes. See Figure 4 for an example of a pair of conflated target domain actions. The conflation of target domain actions takes advantage of the way CAB uses connectionist operations to find structural commonalities between graphs. This approach forces CAB to choose which of the two head nodes in the target domain graph makes the best correspondence to head node nodes of the graph of the single source action. The loser is discarded while the winner is matched against another randomly chosen target domain action. This repeats until only one target domain action remains.

The purpose of conflating target operators is to force CAB to choose whether to map the head of the source operator to a head of one of the two target operators. We believe this is more robust than attempting to identify a metric of "goodness" of analogy; the notion of "goodness" is hard to computationally formalize. However, under certain conditions, CAB will not return a mapping of head nodes. This occurs when structural similarity between the source graph and any portion of the target graph does not meet a minimum threshold. If there is a good analogue in the target domain, the tournament routine will find it. If there is no good analogue, meaning the source story references an action that has no identifiable equivalent in the target domain, `find-best` will return no solution.

The single-elimination competition is equally effective as algorithms that compare a source operator to all pairs of target domain operators while only requiring a linear number of comparisons with respect to the length of the story. This is important because CAB is NP-complete; our process does not significantly affect the computational complexity of the entire transformation algorithm.

Applying the Transfer Function

Applying the transfer function is straight-forward. The function maps source story actions to target domain actions on a one-to-one basis. Once the transform algorithm is applied, we have the new story in the target domain. Note that the resulting story is not guaranteed to be sound, meaning that should the plan be executed, not all actions are guaranteed to have all of their preconditions satisfied in the world state before execution. The next section describes how we can correct flaws in story plan soundness by filling in gaps left by failures in the analogical transformation.

Correcting Errors with Planning

In the cases where the analogical transformation stage has determined – correctly or incorrectly – that there are actions in the source story that have no analogical equivalents in the target domain, the transform algorithm described above will return a target story with missing actions. Because we have chosen to represent story actions as STRIPS-like operators, we have a wealth of contextual information about actions, in the form of preconditions and effects. This information is exactly the information that partial-order planners (POP) need to repair flaws in plans. A partial-order planner requires every precondition of every action in a plan to be satisfied by the effect of a temporally prior action, or the initial state. *Causal links* are used to record the decision made by the planner about how to satisfy preconditions.

We input our target domain story plan $p_t = \langle I, A, O \rangle$ into a partial-order planner as the starting node in the plan search space. Because our plan representation does not include causal links, the partial-order plan is initialized to consider every precondition on every action in p_t to be a flaw. The planner is also provided the target domain operator library Λ_t. The planner resolves the flaws by first attempting to causally link preconditions on actions to effects of existing actions in the plan. Barring that, the planner then attempts to resolve the flaws by instantiating new actions from the domain operator library. Instantiation of new actions fills gaps in the story plan that resulted from failures in analogical transformation. A heuristic penalizes promiscuous instantiation of new actions. Of course there can be different configurations of causal linkages; the planner backtracks if it makes any mistakes that lead to an impasse. Currently we use a standard off-the-shelf POP planner. However, in future work, we may consider planners specialized to the story generation problem, such as the IPOCL story planner (Riedl and Young 2004).

Example

We show how the story translation works on a very simple story. Consider the following sequence of events set in the domain of J.R.R. Tolkein's *The Silmirillion*:

1. Wander (princess) – Princess wanders away from home
2. Capture (king2, princess) – King2 captures the princess
3. Marry (king2, princess) – King2 marries the princess
4. Have-Child (king2, princess) – A child is born

5. 2-Escape (princess, child, king2) – Princess/child escape
6. Chase (king2, princess) – King2 chases Princess
7. Capture (king1, king2) – King1 captures King2
8. Fealty-Oath (child, king1) – Child oath to King1
9. Attack (king2, child) – King2 attacks child
10. Accidentally-Wound (king2, princess) – King2 accidentally wounds Princess
11. Die-of-Wounds (princess) – Princess dies of her wounds
12. Kill (king1, king2) – King1 kills King2

Part of the initial state information of the source story includes the fact that the princess is the daughter of king1. An example of the `Capture` action with some accompanying initial state information is shown in Figure 3. The target domain is the American Old West. In the target domain, consider three of the many possible operators:

- Shoot (wyatt, tom, gun1) – Wyatt shoots Tom with gun1
- Stick-up (wyatt, tom, gun1) – Wyatt points gun1 at Tom
- Stick-up (tom, wyatt, gun2) – Tom points gun2 at Wyatt
- Go (wyatt, saloon, corral) – Wyatt goes between places

The target domain initial state information includes the following facts: Wyatt has gun1, Tom has gun2, Wyatt hates Tom, and Wyatt hates Frank. This set of operators has been chosen in order to show a simple example, but in a real translation, we would have a larger set of operators that includes more actions and all permutations of character arguments.

The story translation algorithm executes as follows. The first action from the source story, Capture(king2, princess), is selected and combined with source domain initial state information. The system randomly picks two operators from the target domain, Go(wyatt, saloon, corral) and Shoot(wyatt, tom, gun1). CAB prefers the correspondence between Capture and Shoot. However, both Shoot and Go are very poor analogies for Capture. Even though CAB is forced to pick one, when none of the target operators are analogous to the source operator, the choice does not matter.

The loser of the first competition, Go, is removed from the list of valid target operators. Next, Capture is compared with the previous winner, Shoot(wyatt, tom, gun1), and Stick-Up(wyatt, tom, gun1). Figure 4 shows the graph of these two target domain operators. This time, CAB prefers the correspondence between Capture and Stick-Up. Note that the mapping is found despite the different number of parameters in Capture and Stick-Up and despite differences in number of preconditions and effects.

The third competition is between Stick-Up(wyatt, tom, gun1) and Stick-Up(tom, wyatt, gun2). The only difference between target operators is the parameter order. We may wonder if it is possible for CAB to discriminate. It turns out that, because of asymmetries in state information – relations between characters and story world objects – combined with the way each operator operates on the story world state is enough to uniquely discriminate between operators.

Further trials are required before the system can finally conclude that Capture(king2, princess) corresponds to Stick-Up(wyatt, tom, gun1). Once the tournament has settled on the best target operator, the effects of the source operator, Capture(king2, princess), are applied to the source story world state and the effects of the target operator, Stick-Up(wyatt, tom, gun1), are applied to the target story world state. This prepares the respective world states for the next tournament, which is a competition to be the best correspondence to Marry(king2, princess) – the next successive operator in the source story.

When the process is complete, the target story analogue is as follows.

1. Go (wyatt, saloon, corral)
2. Stick-Up (wyatt, tom, gun1)
3. Dodge (tom, wyatt)
4. Chase (wyatt, tom, corral, bank)
5. Stick-Up (frank, wyatt, gun3)
6. Shoot (wyatt, tom, gun1)
7. Shoot (frank, wyatt, gun3)

Note that the analogue story is shorter. In some circumstances, the target domain did not have operators that could be matched to source story actions. Gaps occur when best target operator match to a source operator does not achieve a particular correspondence threshold. Consequently, the result may or may not be a sound plan. That is, the result may not possess a logical causal progression. When this occurs, planning is used to fill in the gaps and create a sound plan. For example, the skeleton of a plan created by the analogical transformation does not explain how Frank got to the bank. Planning works backwards from the gap, inserting events that establish any necessary conditions to fill the gap.

Limitations and Future Work

The system described in this paper creates stories that are analogues of previous stories. The advantage of the approach is that source stories can come from heterogeneous sources with non-overlapping event vocabularities. The work described in this paper assumes the existence of libraries of source stories and domain descriptions. While it is undoubtably true that there is a plethora of story material available (e.g., movies, fables, short stories, etc.), they are rarely stored in computational formats that facilitate symbolic reasoning. Independent efforts are underway to create tools and techniques that simplify the creation of story corpora (Elson and McKeown 2007) and to mine stories from the web (Swanson and Gordon 2008). Furthermore, recent interest in *machine reading* (Etzioni, Banko, and Cafarella 2007) has led to techniques for automatically acquiring knowledge from primary sources without significant knowledge-engineering.

One of the limitations of our technique for analogical story transformation is that it requires the source and target domains to be represented at approximately the same level of abstraction. That is, one domain cannot use extremely abstract operators while another domain uses more primitive operators because CAB will not be able to find the structural similarities between operators. Further, the source and target

domains must be fairly analogical to begin with so that correspondences between operators and state propositions can be found. We feel this is a reasonable assumption to make because story worlds mimic the real world to some degree and thus most story world domains, represented at the same level of abstraction, should afford correspondences. One observation we have made is that the less analogical the domains, the more gaps appear in the target story. Further, the more gaps, the less likely CAB will be able to find correspondences between operators after the gaps because the source and target state information become "out of sync." We have not yet determined how to measure the degree of analogical similarity between domains. Our system has not been tested on story world domains of significant complexity, in terms of number of characters or number of distinct propositional statements.

One interesting conclusion of our work to date is that the story transformation process can work with relatively little extraneous information. However, having more contextual knowledge is almost always advantageous in analogical mapping. Some sources of contextual knowledge we have considered for future work are WordNet or other sources of ontological information. The knowledge provided by such knowledge-bases can potentially increase the processing speed, provide more accurate results, and ease authoring of domain knowledge. In particular, the technique described in (Hervás et al. 2006) finds similarities between characters in different stories based on their role. This work is complimentary to our own and could contribute to our story transformation process by preprocessing correlations between characters in the two domains.

Conclusions

Analogical reasoning has been demonstrated to be effective in discovering whether two stories are analogues. However, in order generate an analogue to an existing story, analogical reasoning cannot be applied directly to the existing story content. Instead, we use analogical reasoning to find mappings between concepts in story world domain knowledge provided about the source story world and the target story world. By finding mappings between entities and operators that comprise disparate story world domains, we can use those mappings to directly translate a story from one domain to another. The work described in this paper is a step towards instilling greater capability for creativity into computational systems. Analogical reasoning is often cited to be an integral component in human creativity (Boden 2004; Kolodner 1993b). Consequently, for this work, we have focused on generating story analogues as a demonstration of computational creativity in the context of novel story content production.

References

Boden, M. 2004. *The Creative Mind: Myths and Mechanisms, 2nd Edition*. Routledge.

Elson, D., and McKeown, K. 2007. A platform for symbolically encoding human narratives. In *Proc. of the AAAI Fall Symposium on Intelligent Narrative Technologies*.

Etzioni, O.; Banko, M.; and Cafarella, M. 2007. Machine reading. In *Proceedings of the AAAI Spring Symposium on Machine Reading*.

Falkenhainer, B.; Forbus, K.; and Gentner, D. 1989. The structure-mapping engine: Algorithms and examples. *Artificial Intelligence* 41:1–63.

Gervás, P.; Díaz-Agudo, B.; Peinado, F.; and Hervás, R. 2005. Story plot generation based on CBR. *Journal of Knowledge-Based Systems* 18(4–5).

Giora, R. 2003. *On Our Mind: Salience, Context, and Figurative Language*. Oxford University Press.

Hervás, R.; Pereira, F.; Gervás, P.; and Cardoso, A. 2006. Cross-domain analogy in automated text generation. In *Proceedings of the 3rd Joint Workshop on Computational Creativity*.

Kolodner, J. 1993a. *Case-Based Reasoning*. Morgan Kaufmann Publishers.

Kolodner, J. 1993b. Understanding creativity: A case-based approach. In *1st European Workshop on Case-Based Reasoning*.

Larkey, L., and Love, B. 2003. CAB: Connectionist analogy builder. *Cognitive Science* 27:781–794.

Lebowitz, M. 1987. Planning stories. In *Proceedings of the 9th Annual Conference of the Cognitive Science Society*.

Meehan, J. 1976. *The Metanovel: Writing Stories by Computer*. Ph.D. Dissertation, Yale University.

Pérez y Pérez, R., and Sharples, M. 2001. MEXICA: A computer model of a cognitive account of creative writing. *Journal of Experimental and Theoretical Artificial Intelligence* 13.

Propp, V. 1968. *Morphology of the Folktale*. Austin, TX: University of Texas Press.

Riedl, M. O., and Sugandh, N. 2008. Story Planning with Vignettes: Toward Overcoming the Content Production Bottleneck. In *Proceedings of the 1st Joint International Conference on Interactive Digital Storytelling*.

Riedl, M. O., and Young, R. M. 2004. An Intent-Driven Planner for Multi-Agent Story Generation. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Swanson, R., and Gordon, A. 2008. Say Anything: A massively collaborative open domain story writing companion. In *Proceedings of the 1st International Conference on Interactive Digital Storytelling*.

Swartjes, I.; Vromen, J.; and Bloom, N. 2007. Narrative inspiration: Using case based problem solving to support emergent story generation. In *Proceedings of the International Joint Workshop on Computational Creativity*.

Turner, S. 1992. *MINSTREL: A Computer Model of Creativity and Storytelling*. Ph.D. Dissertation, University of California , Los Angeles.

Young, R. 1999. Notes on the use of plan structures in the creation of interactive plot. In *Proceedings of the AAAI Fall Symposium on Narrative Intelligence*.