Lessons Learned From a Rational Reconstruction of Minstrel

Author(s) Name(s)

Affiliation(s) Go Here in 9 Point Times Roman
Author(s) Address(es) Go(es) Here in 9 Point Times Roman
Author(s) Address(es) Go(es) Here in 9 Point Times Roman
Author(s) E-mail(s) Go(es) Here in 9 Point Times Roman

Abstract

Scott Turner's 1993 Minstrel system was a high water mark in story generation, harnessing the concept of imaginative recall to generate creative stories. Using case-based reasoning and an author level planning system, Minstrel models human creative processes. However, the algorithmic and representational commitments made in Minstrel were never subject to principled and quantitative analysis. By rationally reconstructing Minstrel, we are able to investigate Turner's computational model of creativity and learn new lessons about his architecture. We find that Minstrel's original performance was tied to a well-groomed case library, but by modifying several components of the algorithm we can create a more general version which can construct stories using a sparser and less structured case library. Through a rational reconstruction of Minstrel, we both learn new architectural and algorithmic lessons about Minstrel's computational model of creativity as well as make his architecture available to the contemporary research community for further experimentation.

Introduction

Current work in artificial intelligence draws from key ideas dating back to the 1960s. Many seminal papers in AI presented systems that used novel algorithms to model human cognitive functions or achieve groundbreaking results. Modern processing power and experimental techniques could be used to gain important insights into such foundational systems, but many of these systems are not accessible to the AI community. Whether because their source code is not available or simply because they run on architectures that are now rare, some early AI systems are now inaccessible.

Some AI techniques have been extensively studied such as Neural Networks and Graphical Models. These algorithms have been the subject of ongoing study, and their details and variants are well-understood. In less well-explored areas of AI, however, some systems have never been thoroughly tested, and the architectures that they proposed have not been studied. The technique of rational reconstruction addresses these issues.

By rebuilding systems from their technical descriptions, rational reconstruction seeks to gain new insight into existing architectures and learn lessons applicable to modern research. Rebuilding systems reveals key qualities of their architectures, and can differentiate important architectural choices from trivial ones. Our work involves the rational reconstruction of Scott Turner's 1993 Minstrel system (Turner 1993, Turner 1994). Turner's Minstrel modeled human creative processes to generate stories using a combination of simple planning and a novel case-based reasoning system. Minstrel models creativity in storytelling as a process of imaginative recall: the modification of memories of previous events to invent novel scenarios. Although Turner ran a variety of experiments to demonstrate its effectiveness, he did not perform large-scale quantitative analysis of his system. We are interested in imaginative recall as a model of creativity, including what kind of memory base must be present to support it and whether it can be applied to generating large batches of stories.

Our reconstruction highlights several interesting design choices in the Minstrel architecture and provides quantitative results to demonstrate how the architecture performs. We show that Minstrel's story library was critical to its success, and provide an alternate implementation which does not rely so heavily on this library. Our implementation is also tuned to provide sustainable variety over the course of many stories, instead of generating only a few very creative stories before exhausting its library. Finally, we have made Minstrel available (downloadable from http://minstrel.soe.ucsc.edu) as a Scala library so that other interested researchers can use it in new projects or perform their own experiments on it.

Related Work

Story and Planning and Case-Based Reasoning

Minstrel and its Remix are similar to a number of other systems for story generation. Some of the earliest work in this area was done by systems that relied primarily on planning algorithms such as Universe (Lebowitz 1977) and Talespin (Meehan 1977). Later storytelling systems such as Fairclough's multiplayer story generator and Gervas et al.'s plot generator have demonstrated the effectiveness of CBR techniques applied to the problem of story generation (Fairclough and Cunningham 2003, Gervas et al. 2005). One of the difficulties in applying Case-Based Reasoning (CBR) to stories is coming up with a searchable and compositional representation of story events. Fairclough and Gervas et al. drew extensively from Vladimir Propp's formal theories about the substance of Russian folktales (Propp 1968) to come up with representations of stories that were amenable to being searched as a case base. Turner was aware of Propp's analysis, but developed his own method of representing stories involving a graph of frames describing story events.

Minstrel is related to all of these systems since it uses both planning and CBR. A modified form of Case-Based Planning which involves incremental transformation and adaption steps is employed for fine grained generation while planning is used to simulate higher level authorial intention and additions that an author might use while creating a story.

Rational Reconstruction

Rational reconstruction (RR) is the process of reconstructing existing systems with the goal of understanding their operation as well as the original design decisions involved, and how those decisions affected the final system. Rational reconstruction efforts offer a chance to critically reexamine older research, as well as to explore fruitful designs using modern tools. There are a number of books which discuss RR as a useful tool such as Partridge's books on the fundamentals of Artificial Intelligence (Partridge 1990, Partridge 1991). Additionally, a number of papers discuss successful rational reconstruction projects such as Langley's rebuilding of the GRIDS system (Langley and Stromsten 1990), Ritchie's reconstruction of Proteus (Ritchie 1984), McDonald's reconstruction of Genaro (McDonald 1999), and even Peinado and Gervas' reconstruction of Minstrel's ontology using the OWL Web Ontology Language (Peinado and Gervas 2006). These works demonstrate rational reconstruction's potential for producing knowledge about existing systems and informing new theories.

Computational Creativity

Turner's work on Minstrel built on an existing literature in computational creativity, including the AM (Lenat 1976) and Daydreamer (Meuller and Zernick 1984) systems. Since his work, Margaret Boden (1996, 2004) and Graeme Ritchie (2001, 2007) have proposed models for creativity and methods for measuring it. Boden focuses on the distinctions between transformational, exploratory, and combinational creativity (Minstrel focuses on the latter). Boden also describes the difference between P-Creativity (creativity relative to a creator's knowledge, which Minstrel exhibits) and H-Creativity (creativity relative to a culture). Ritchie builds on Boden's ideas and focuses on novelty and quality of artifacts as the most important measurable manifestations of creativity. Although we don't focus on analyzing Minstrel Remixed in terms of its creativity in this paper, these developments in creativity theory have informed our reconstruction. In particular, there is a close relationship between the TRAM system in Minstrel and both the variety and quality of results.

Architecture

Our system, called Minstrel Remixed (MR), is a rational reconstruction of Scott Turner's 1993 Minstrel system. Working from Turner's dissertation describing the system, we have re-implemented the core algorithms and supporting structures in Scala. MR includes the original components of Minstrel as well as a few upgrades of our own devising. Additionally, MR is easily configurable, and it includes a configuration designed to imitate the original Minstrel as closely as possible.

The Minstrel Architecture

Minstrel is a complex architecture including many subsystems and components in this paper we focus on three subsystems which are relevant to how imaginative recall interacts with the story library: TRAMs, ALPs, and boredom. TRAM stands for Transform Recall Adapt Method and 'the TRAM system' refers to a collection of TRAMs and accompanying support mechanisms that are used to generate the fine details of a story. ALP stands for Author Level Plan and these are used to direct the broader themes of a story as well as being used to enforce a variety of consistency constraints. Together the planning of the ALP system and the modified case-based reasoning of the TRAM system allow for complete stories to be pieced together. The third relevant system, boredom, helps to ensure that stories that are generated differ both from the stories in the library and from subsequently produced stories.

TRAMs

TRAMs (Transform Recall Adapt Methods) support a modified form of case based reasoning. In Minstrel, many small fragments of stories are created over the course of creating a whole story and this is done by the TRAMs. The TRAM system is called with a search query which is generally a partially defined story fragment such as "someone does something to a dragon which kills it." The system has a number of TRAMs which can be recursively

applied in any order to a given query to find a match: each TRAM applies changes (Transforms) the query (usually making it more general) until a match is found (Recall), and then each TRAM Adapts the result, so that at the end of the process the modified result applies to the original query. So the TRAMs both control the search direction and handle the adaptation step in Minstrel's case-based reasoning subsystem.

The first step for each query is to attempt to recall fragments out of the story library which match without any transformation. Failing this, each TRAM has a transformation that it applies to the query before recursing. Eventually a result is found, and each TRAM in the recursive stack applies its own adaptation code, transforming the result into a form applicable to the original query. The benefit of this method is that it can use precise transformations and adaptions in sequence to effectively find cases that are quite different from the original query. Of course, the TRAMs are not perfect, and the more TRAMs used, the higher the risk of a result which is incoherent with the rest of the story. Thus the TRAM system relies on a story library which contains examples that are fairly close to the queries it attempts to answer.

In Figure 1, an example query is shown in which Bob needs to do something to a dragon but no stories are available in the library which can be used. As a result, a TRAM converts the dragon to a generic monster and this new query retrieves a fragment from the case library about Bob fighting a troll with a magic sword. This result is finally adapted back to a fragment in which Bob fights the dragon with a magic sword, satisfying all of the original requirements.

Figure 1. TRAM system in action

TRAMs operate inside of a backtracking search tree, allowing for many different sets of TRAMs to be tried in order to eventually find a match for a given query. Figure 2 below illustrates a TRAM search tree with 3 useable TRAMs. The query passed in is the top node in which a knight fights something. Three TRAMs are possible children of this node and in the diagram, the middle child has been picked, transforming the query to a person fighting something. At this point Minstrel might recall, a peasant fighting a troll. When adapted back, a story about a knight fighting a troll makes sense. If the recall were to fail however (if there are no stories in the library about people fighting) there are 2 children, relaxing the constraint that there is a person involved (which would likely result in adequate results) or releasing the constraint of a fight being in the matching story. Now Minstrel might match a story about a princess eating a berry, yielding a much stranger story after adaptation: a knight fighting a berry.

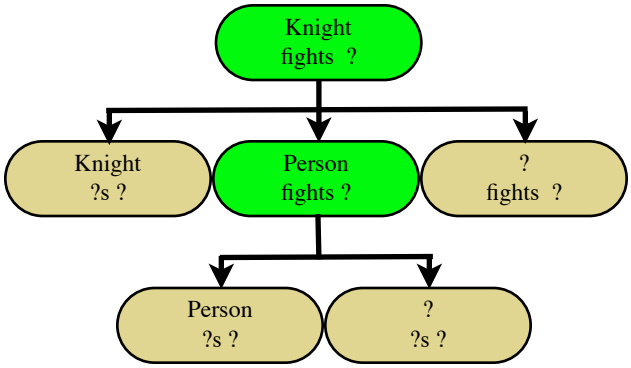

Figure 2. TRAM search space

Figure 2 includes two first level children which are instances of the same TRAM: Generalize Constraint (which turns a constraint into a '?'). The original Minstrel only considers each TRAM once, choosing a target for its modification. Minstrel makes these decisions randomly, which can cause it to overlook possible results.

We made two significant changes to TRAM search in Minstrel Remixed to support deterministic searches while maintaining the core functionality. First, we split each node of the search tree into fragments which include targeting information as well as a given TRAM. This allows the system to explore all of the possible applications of each TRAM, but also greatly expands the search space. To compensate for this expansion, we added a strict limit to the number of nodes the system was allowed to explore. In addition to Turner's limit of a search depth of 3 TRAMs, we forced the system to stop searching for a result after exploring 250 nodes in the TRAM search tree. We also modified the TRAM selection algorithm to treat variants of a single TRAM with different targeting information as a single TRAM for the purposes of selection during search, to ensure that our addition of variants didn't bias the search process.

Our second modification was the addition of a weighted random selection method to control the search. In our modified version, each TRAM is assigned a weight, corresponding roughly to how much it modifies the original query. TRAMs are then selected randomly with probability inversely proportional to their weights, so that TRAMs which have more drastic effects are chosen less frequently. The overall effect of this modification is to keep

stories from being too strange even when multiple TRAMs are required to get a result by searching more intensely in the part of the search space closest to the original query (in terms of the strength rather than the number of TRAMs). The cost is that the most creative solutions are more rare, because dramatic modifications to a query are one (risky) way to achieve creative results. To compensate for this, we generate more stories: instead of a system that can generate five quite creative stories before beginning to generate incoherent stories, we want to produce a system that can generate hundreds of different stories, most of which will contain some creativity and some of which will be highly creative.

Author-Level Planning

In the author-level planning system, Minstrel pursues author-level goals (ALGs) by retrieving and executing author-level plans (ALPs) that serve specific functions in the service of generating a story. High level goals consider the story as a whole, and represent tasks such as deciding on a theme for the story or checking the story for opportunities to insert foreshadowing. At the same time, lower level goals concern things including filling out the details of a particular state or act within the story, or checking that a particular story node is consistent and properly motivated. Some of the ALPs encode micro-theories about storytelling (theories of consistent motivation, for example) that help Minstrel produce consistent output. Other ALPs rely on the story library to act as a model of a well-formed story, using the TRAM system to fill in pieces of the story under construction with appropriate material.

It is worth noting that Minstrel does not simply generate free-form story structures. Instead, it relies on a "planning advice theme" (PAT, or more generally a story template) to give structure to its stories. These story templates contain rough specifications for important parts of the plot. Minstrel generates stories by selecting an appropriate PAT and then filling in the details of the PAT, adding extra scenes to the story as necessary along the way.

Once a goal is selected and a set of plans to achieve that goal is found, plans are tried one by one until one succeeds. If all available plans fail, the system re-enqueues the current goal with half of its original priority and drops it entirely if this would put it below a minimum priority threshold. This convention allows goals to interact: if one goal cannot be solved initially, other goals are attempted in the hopes that they will alter the story configuration and make the initial goal solvable. Once the goal queue is empty, story generation is finished.

In the original Minstrel, the goal queue contained low-priority goals (which would be executed last) for generating prose from the story graph that Minstrel had produced. Because Minstrel Remixed does not yet have a natural language generation component, our system's output is currently raw story graphs. Although these aren't accessible to a wide audience, we are able to interpret them and judge the variety and consistency of the stories that they represent in order to learn about Turner's algorithm.

Boredom

Boredom is the final relevant subsystem to our current discussion. It is implemented as a table of query/solution signatures coupled with a boredom value. Every time a query is given to the TRAM system and a solution returned, the boredom value of that signature is incremented. Minstrel won't allow high boredom solutions to be presented for their matching queries so as the boredom value for pairs rises, new solutions must be found for given queries. This method is a way of keeping the results varied; without it Minstrel would sometimes generate duplicate stories (of course, the random nature of the TRAM searches also contributes to variation whenever TRAMs are used to generate results).

Classic Minstrel has a boredom threshold of two and increments boredom values of signatures by one every time a query/result pair is used. In section 15.7.2 of his dissertation, Turner evaluates boredom by generating six stories, showing how boredom drives the use of new material, but also that by the sixth story, Minstrel has exhausted its story library (Turner 1993). As a result, Minstrel Remixed contains an upgraded boredom system in which each call to the TRAM system fractionally decrements the values of all signatures in the boredom table. Functionally this means that signatures refresh over time, allowing them to be reused in subsequent stories. Using this system, instead of quickly becoming bored of everything, Minstrel establishes a cyclical pattern of boredom. This forces it to generate a variety of stories, just like the original boredom heuristic, and the random nature of the TRAM system means that the set of boring results doesn't have any predictable pattern. The effect is similar to generating each new story using a random subset of the story library, always avoiding recent results. This randomization produces much more sustainable variety than Turner's original boredom mechanism: it distributes the creative potential of the system given the story library more evenly over a large number of stories.

Experiments

Although Minstrel's original configuration works well for building stories, it does this by virtue of access to a well-tailored story library: in section 15.10 of his dissertation, Turner reports that 88% of searches in his system were able to recall episodes directly from the story library without using imaginative recall (Turner 1993). The original system was also designed to generate only a few stories at a time, especially given the aggressive boredom mechanism (discard a result forever if it's been used twice). These facts raise questions about imaginative recall as a model of creativity: can imaginative recall function effectively in an environment where the memory is less dependable, and can imaginative recall be tuned to focus

on producing variety with occasional creativity, rather than making everything creative but quickly running out of episodes to recall? In our reconstruction, we focused on changes that allow the system to work with a sparser library and to produce substantially more stories by distributing its creativity more evenly.

One reason that Turner focused on generating only a small number of stories may have been the limited computing power available to him in 1993. Given a modern computer (2.4 GHz Intel i7 with 8 GB of RAM), we can generate stories quite quickly (56 seconds per story), and thus it is reasonable to expect a system to generate tens or hundreds of stories at a time. Turner's original system took an average of 1,400 seconds to construct a story, so such mass production was not quite as feasible. His approach also corresponds to a different application of creativity: if a person were to produce one creative story twenty times in a row, she would produce a different result than if you ask her for twenty creative stories. Turner's original model is closer to the first case, while our changes have made the system operate a bit more like the second case (although explicit planning for distributing creative results across multiple stories remains a direction for future work).

To test our modifications, we set up a version of our system that duplicated the performance of the original as closely as possible, and compared it to our modified version. Due to slight differences in our architecture and information lacking in Turner's dissertation (such as the exact contents of the story library) we were not able to perfectly duplicate the original. However, we did recreate the original boredom system and TRAM search method, which are the only components varied between our two experimental conditions.

Method

Our tests involved generating 75 stories and measuring the performance of the TRAM system. Because the TRAM system is the main mechanism for creativity, the number of TRAMs used and the number of TRAMs tried reveal information about the performance of the system. For our reconstruction of Turner's original system, we generated 75 stories using random TRAM selection and no boredom reduction. We then restarted the system (to reset boredom) and asked it to generate 75 new stories, this time with weighted TRAM selection and boredom reduction enabled. For each batch of stories, we recorded every call to the TRAM system, including whether it succeeded, how many TRAMs it tried while searching for a result, and how many TRAMs were used to find the result if it succeeded (note that the number of TRAMs used is never more than three (the depth limit) but that the number of TRAMs tried may be much larger). For both experiments, we used Turner's original depth limit of three TRAMs, as well as our exploration limit of 250 TRAMs. We used a single story template (PAT) for all 150 stories, which helps exercise the boredom mechanism (Turner's original boredom experiment also used a single PAT). Our story template consists of one character trying to kill another character but accidentally dying, after which a third character avenges the first character.

Results

Given our recreation of the original system, we found that for successful queries that involved TRAMs (as opposed to queries that found a result directly) an average of 2.38 TRAMs were used. In contrast, our modified system used an average of 1.39 TRAMs when it used TRAMs to find a result.

In the recreated version, out of 490 total TRAM attempts, 59% of searches found a result directly, 22.2% used one or more TRAMs, and 18.7% failed (as a result of running into the exploration limit). In our modified version, which made 503 total TRAM attempts, 72.2% of searches found direct results, 24.5% used one or more TRAMs, and 3.4% failed. For searches that succeeded (including searches that succeeded directly), the recreated system tried an average of 13.8 TRAMs, while our version tried 7.7 TRAMs. When only searches that required TRAMs to succeed are considered, the recreated system tried 141.3 TRAMs on average, while ours tried 56.8. These results are summarized in table 1.

	Turner's Version	Recreated Version
Total Searches	490	503
Direct Matches	289 (59%)	363 (72.2%)
Indirect Matches	109 (22.2%)	123 (24.5%)
Failures	92 (18.7%)	17 (3.4%)
Avg. TRAMs tried	57.9	15.8
Avg. TRAMs tried (all matches)	13.8	7.7
Avg. TRAMs tried (indirect matches)	141.3	56.8
Avg. TRAMs used (all matches)	0.65	0.35
Avg. TRAMs used (indirect matches)	2.38	1.39

Table 1: Experimental results from the generation of 25 stories.

Most of the results are driven by our boredom changes: with fewer results excluded via boredom, it's easier to find valid results. This means that fewer TRAMs are needed, fewer TRAMs need be attempted, and fewer queries fail. Overall, this leads to the conclusion that Turner's original design was tuned to generate only a small number of stories, which makes sense given the experiments that he chose to conduct. It's also apparent that our story library is not as well tuned as Turner's was: he reported that 88% of searches found results directly, whereas only about 59% of ours do. Although spending time tuning our library might increase the number of direct results, it's not a good way to

leverage the architecture. Effectively, tuning the story library represents increasing the quality of stories by hand-authoring more content, and this tuning must be done separately for each domain that applies the architecture.

Instead, we have made the core algorithm more robust: our modifications reduce the percentage of failed TRAM searches from 18.7% to just 3.4%, even given the less dependable story library (relative to Turner's). Of course, a single failed TRAM search does not mean that the relevant story failed to generate: the search will be tried again later by Minstrel's author-level planning system, and might succeed, or information for that node might be filled in by other searches. TRAM failures do represent lost opportunities to find creative results, of course, and they could cause story generation to fail entirely if they occur too frequently.

Results that are found by applying more TRAMs are generally more creative (but also more likely to be incoherent). This is because the changes to the query made by each TRAM decrease the amount of context used in the search. Turner reported that after just five stories, the boredom assessment was causing TRAMs to find questionable results. The modified TRAM and boredom system, by performing fewer TRAM applications on average, reins in this creativity, meaning that most of our results are quite mundane. When it uses TRAMs to find a result, our system only explores 56.8 TRAM combinations on average, and the results that it finds use an average of 1.4 TRAM applications. In contrast, the reconstructed system explores an average of 141.3 TRAM combinations and uses an average of 2.38 TRAMs. Compared to our version, Turner's version requires more effort to find results, and when it finds them, they are less similar to the original query. These numbers only include searches that used TRAMs successfully, so they are independent from the difference in failure rates. Of course, when generating a large batch of stories, creative results will be mixed in either due to luck or due to our modified boredom mechanism. Rather than quickly exhaust Minstrel's creativity, we have chosen to mete it out more sparingly across many stories, so that our system can effectively generate a wide variety of simple variations from a template, with some especially creative variations mixed in.

Conclusion

Rational reconstruction aims to rebuild existing systems from their technical specifications in order to learn more about the motivations behind architectural decisions as well as figure out which parts of a system are essential to its operation. In our rational reconstruction of Minstrel, we have also tried to create a system that makes the Minstrel architecture available in a modern format and which adapts the architecture to exploit the increases in computing power since Minstrel's first release. Instead of focusing on generating a few very creative stories, our system focuses on longer-term generation of story variants, some of which we expect to be creative. Our system is also less reliant on a finely-tuned story library than Turner's. It is able to function with only a 3% search failure rate even with only a 72% direct match rate against our story library. Our reconstruction of Turner's system, in contrast, matches against the story library only 59% of the time and fails 22% of the time. By modifying the boredom heuristic we are able to achieve dramatically increased success rates, and by also modifying the TRAM search method, we have created a system tuned for producing large batches of results that contain some creativity.

As we reconstructed Minstrel, it seemed at first to be extremely specific. Because Turner's original story library isn't available (it isn't fully described in his dissertation) we constructed our own story library, but quickly found that it didn't match our templates closely enough. If extensive library tuning were required for Minstrel to function, it would not be a model of strong creativity, because its creative power would depend on a high rate of unimaginative recall rather than extensively using imaginative recall. Although Turner claims that his system demonstrates that creative recall is not necessarily common when making up stories, it is more useful to have a system which can operate using more or less imagination depending on the situation: such a system can then be compared to human performance in a variety of situations. Thus instead of tuning our library to produce results, we focused on modifications to the original that could allow it to work with a sparser library, while still retaining creative results. As our experiment demonstrates, we have created a version of the Minstrel architecture that works well for generating large batches of stories, even with a less tailored library.

Through our efforts, we not only gained knowledge about the original system, but also created a new version which is available in a modern format. This is an encouraging result because it indicates that rational reconstruction of older systems is a productive way to gain new knowledge about AI architectures and to better understand existing computational models of human cognition. Using our reconstruction, it is now feasible to concretely investigate imaginative recall as a model of human creativity by comparing it directly to human results.

References

Boden, M. ed. 1996. *Dimensions of Creativity*. Boston, MA.: MIT Press

Boden, M. 2004. *The Creative Mind: Myths and Mechanisms*.: London, UK.: Routledge

Fairclough, C., and Cunningham, P. 2003. A multiplayer case based story engine. In *4th International Conference on Intelligent Games and Simulation*, 41–46.

Gervas, P., Diaz-Agudo, B., and Hervas, R. 2005. Story plot generation based on cbr. *Journal of Knowledge Based Systems*, 235-242

Langley, P., and Stromsten, S. 2000. Learning Context-Free Grammars with a Simplicity Bias. In *Proceedings of the 11th European Conference On Machine Learning*, 220-228

Lebowitz, M. 1985. Story-telling as planning and learning. *Poetics*, 14(6).

Lenat, D. 1976. AM: An Artificial Intelligence Approach to Discovery. Ph.D. diss., Dept. of Computer Science, Stanford University, Stanford, CA.

McDonald, D. 1999. A Rational Reconstruction of Genaro. In *Proceedings of the RAGS Workshop*.

Meehan, J. 1977. Tale-spin, an interactive program that writes stories. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*.

Mueller, E. and Zernick, U. 1984. GATE Reference Manual. Technical Report, UCLA-AI-84-4, Computer Science Department, Univesity of California, Los Angeles CA.

Partridge, D. 1991. *A New Guide To Artificial Intelligence*. Norwood, NJ.: Ablex Publishing

Partridge, D., Yorick, W. eds. 1990. *The Foundations of Artificial Intelligence: A Sourcebook*. New York, NY.: Cambridge University Press.

Peinado, F., Gervas P. 2006. Minstrel reloaded: from the magic of lisp to the formal semantics of OWL. In *Technologies for Interactive Digital Storytelling and Entertainment*, 93–97.

Propp, V. 1968. Morphology of the Folktale, trans. Laurence Scott. Austin, TX.:University of Texas Press.

Ritchie, G. 1984. A Rational Reconstruction of the Proteus Sentence Planner. In *Proceedings of the 10th international conference on Computational linguistics*.

Ritchie, G. 2001. Assessing Creativity. In *Proceedings of the AISB '01 Symposium on AI and Creativity in Arts and Science*, 3-11.

Ritchie, G. 2007. Some Empirical Criteria for Attributing Creativity to a Computer Program. *Minds and Machines*. 17(1). 67-99.

Turner, S. 1993. MINSTREL: a computer model of creativity and storytelling.

Turner, S. 1994. The creative process: a computer model of storytelling and creativity.