Narrative Inspiration: Using Case Based Problem Solving to Support Emergent Story Generation

Ivo Swartjes, Joost Vromen and Niels Bloom
Human Media Interaction
University of Twente
PO Box 217, 7500 AE Enschede, The Netherlands
{swartjes,vromen,bloom}@cs.utwente.nl

Abstract

We consider a system that can generate stories as a creative system. One approach to building such a system is to simulate and narrate the behaviour of believable characters in a virtual story world. A risk of this approach is that no interesting story emerges. In order to make the behaviour of the characters more interesting from a story perspective, we propose a system that can use example story pieces, written from a plot perspective by a human author, to inspire decisions for characters in an emerging story.

On a more philosophical note, we discuss the story generation process in the light of a characterization of creative systems and show that considering an automated story generator as a creative system can help to reveal implicit design choices.

Keywords: Story Generation, Computational Creativity, Case Based Reasoning, Fabula.

1 Introduction

The Virtual Storyteller is an experiment in the creation of stories through simulation of a dramatic story world inhabited by virtual characters (Theune et al., 2004). We separate the content of a story from its presentation (e.g., in the form of natural language or animation). The focus of this paper will be on the generation of story *content*, meaning we focus on a way to generate an interesting sequence of events.

The characters in the Virtual Storyteller are modelled to be believable, which means amongst other things that they are pursuing their own goals, have an emotional model and a simple form of personality[1]. Believable behaviour does not necessarily lead to a coherent and interesting plot, though. This was the big lesson learned from one of the first systems to use character models to generate stories, namely TALE-SPIN (Meehan, 1981). TALE-SPIN was able to tell simple Aesop-like stories about animal characters that were trying to fulfil their basic needs. The stories generated by TALE-SPIN were certainly believable but often uninteresting. There is no guarantee that the coincidental interaction between the characters results in a dramatically interesting and coherent whole. Subsequent approaches to story generation have therefore tried to focus more on plot development. In these approaches, the succession of a sequence of dramatic events is modelled, and characters are placed in function of these events (*right...we need a robbery...which characters can we use for that?*) which in turn makes it quite difficult to have the characters appear believable.

A contemporary example of character-driven narrative is the FearNot! system, which simulates affectively driven characters that exhibit and respond to bullying behaviour (Aylett et al., 2005). The goal of the system is to educate children – who play the role of invisible friend of the main character that is being bullied – how to deal with such behaviour. A typical plot-driven approach is the Fabulist system (Riedl and Young, 2005), which is able to plan a plot that fulfils certain dramatic goals, whilst trying to relate the plan steps to character intentions and personality.

We are investigating character-driven (emergent) story generation as in the FearNot! project. The main difference is that in the approach we pursue, characters have a double role: they are believable inhabitants of a virtual story world as well as improvisational actors that help to create an entertaining experience. With the latter, we hope to make up for the potential lack of story development when using only believable characters. We also intend to use a plot agent that influences the course of the story to increase the chance of interesting plot developments.

In the course of an emergent narrative, there are many decisions to make for both the character agents (in terms of which goals to pursue, how to respond emotionally to new information) and the plot agent (in terms of how to affect the emerging story). Character agents are implemented using an affective architecture with the aim of making believable decisions. We hope to augment the decision space with options that are also interesting from a story development perspective. We want to allow human

[1]An extensive discussion of what it means for virtual characters to be believable can be found in (Loyall, 1997)

authors to be able to write example story pieces and allow a Case Based Reasoning (CBR) system to use these pieces as example solutions that inspire some of these decisions.

After discussing some work related to the use of CBR in story generation in section 2, we will describe a creative problem solver based on CBR and explain how it can inspire decision making for the character and plot agents in section 3. Sections 4 and 5 will provide a bit more in-depth description of the proposed system in terms of cases and creativity heuristics, respectively. The implementation of the creative problem solver is discussed in section 6. Section 7 will position the design of a story generator within the context of the formal framework for categorizing creative systems proposed by Wiggins (2001). We argue that such a positioning is useful to make more informed choices in the design of creative story generation systems.

2 CBR in the Context of Story Generation

Case Based Reasoning (CBR) is a reasoning process that finds its origins in the psychological model of episodic memory. In CBR, knowledge is captured in the form of a set of cases that describe a specific problem, and a specific solution to that problem. To reason about solutions for problem situations, the collection of cases is explored in order to find similar situations. The solutions for those similar problem situations can be adapted to form a solution to the problem that needs to be solved.

CBR has been applied in the context of storytelling or storytelling-like systems with satisfying results (Mueller, 1987; Turner, 1994; Fairclough, 2004; Gervás et al., 2004). With the exception of Mueller (1987), who uses CBR in a system that generates daydreams, the use of CBR in these systems has been inspired by or directly based on the work of Vladimir Propp (Propp, 1968). Propp analysed Russian folk tales, and discovered a big structural similarity between them. He identified a set of character roles (e.g., the hero, the villain) and a set of character functions (e.g., departure, interdiction and marriage). Each of the folk tales he investigated contains a subset of the character functions, and always in a certain order. This makes Propp's analysis pleasant to formalize and use in story generators, but its very order constraints make it difficult to use in the emergent story generation process we are exploring, where – theoretically – anything can happen that does not conform with this order.

We therefore use CBR to generate high-level character behaviour based on story-specific input rather than Proppian plot variations. We want to provide a human author with the possibility to write story content that has the flexibility to be recombined and reformed to expand the space of situations in which it can be used. Our problem solver, discussed in section 3, has been influenced most strongly by the MINSTREL system (Turner, 1994). Turner considers creativity to be an extension of problem solving, the result of cognitive processes that bring together pieces of old knowledge in new ways. MINSTREL has demonstrated the possibilities of case based problem solving to model the creative process of generating simple stories in the King Arthur domain. MINSTREL takes storytelling goals as problems to solve, and retrieves cases that form a solution to these problems. Usually, CBR retrieves solutions by comparing the problem to solve with similar problems in the case base. MINSTREL adds creativity to this problem solving by actively transforming the problem descriptions into similar descriptions, and subsequently adapting the found cases to fit the original problem. MINSTREL uses a collection of Transform-Recall-Adapt methods (TRAMs) for this process.

For example, if MINSTREL attempts to create a story in which a knight commits suicide, a problem description could express the following problem: *A knight does something that results in the knight's death.* A TRAM can transform this problem description into: *A knight does something that results in someone's death.* Based on this problem description, the following case can be retrieved:

> *A knight fights a troll with his sword, killing the troll and being injured in the process.*

This retrieved case can then be adapted to form a solution to the original problem description:

> *A knight fights and kills himself with his sword.*

A combination of such problem solving steps, guided by author-level goals and themes, leads to the structural composition of simple stories.

MINSTREL was implemented using a custom frame-based language called Rhapsody, implemented in Lisp. Currently, attempts are being undertaken to re-implement MINSTREL using the contemporary knowledge formalism OWL-DL, a dialect of the well known ontology language OWL resembling Description Logic (Peinado and Gervás, 2006). Similarly, our problem solver uses knowledge from a story world ontology specified in OWL-DL for its transformations. The problem solving cycle has big similarities to MINSTREL's model of creativity but is only loosely based on its specific details. As we will see in section 3, we intend to use case based problem solving for a different purpose than the structural composition of a story from cases, as done in MINSTREL.

3 Using Case Based Problem Solving for Character and Plot Decisions

Our problem solver uses CBR for two reasons. First, because the knowledge needed for a certain problem domain is covered by a set of example cases instead of an extensive set of 'first principles'[2], CBR can decrease the amount of knowledge needed and reduce or eliminate the need to model the causal interactions of this knowledge in the form of a reasoning system (Cunningham, 1998).

Second, we believe that cases form an intuitive way for a human author to write story content. As we will discuss in section 7, the designer of a story generator is also responsible for the quality of its generated stories. It therefore makes sense to make the input knowledge as

[2]In the case of our decision making, these are principles like 'a person can fight a dragon when he is near one, has a weapon and is not too afraid' and 'if you are hungry, and you know where food is, then you should eat the food'.

accessible to authors as possible and we believe that using examples (rather than worrying about the 'first principles' rules that underly them) satisfies that concern.

The cases in our problem solver therefore take on the form of example pieces of story. We believe that such example story pieces can form a good knowledge source in the decision making of character agents in a storytelling domain, most importantly because they describe character behaviour in a narrative context, transcending individual decision making. Take for example a case expressing the following example story piece: *Frustrated by her singing, John insults his little sister, making her cry*. Not only does this case offer a character a believable coping behaviour for being frustrated at one's sister, it is also an *interesting* decision because it affords an interesting story situation (in this case, a decision that affects important other characters). When the cases have been constructed to express believable character behaviour as well as interesting narrative situations, using these knowledge sources results in behaviour that is in theory both believable and interesting.

The character agents of the Virtual Storyteller make decisions based on appraisal and deliberation processes (which goals to take on, which actions to pursue, how to interpret perceptions and how to respond to them emotionally). As an alternative to using these processes, such decisions can be contracted out to the creative problem solver by translating them into problem descriptions and asking the problem solver to find solutions for them. We aim for an integration of this decision making within the processes that our character agents already run, similar to the work of Moraes and Costa (2004) which shows how to make such an integration of 'improvised' decision making within a BDI architecture.

The cases can also be used by the plot agent to make decisions about influencing the emerging story. Such decisions involve introducing dramatic events, adding new knowledge about the story world, or suggesting actions and goals to the characters. Because the plot agent uses much of the same episodic knowledge as the characters, it can make reasonable assumptions about the character's reaction to these events. In-depth discussion of the integration of the decision making process with the Virtual Storyteller architecture falls outside the scope of this paper, which focuses rather on the decision making process itself.

Figure 1 shows the decision making process we propose, including the CBR problem solving cycle we intend to incorporate in the Virtual Storyteller. A user – be it one of the character agents or the plot agent – needs to make a decision which it translates into a problem for the creative problem solver. The problem solving cycle starts, in which copies of the problem are transformed into similar problems. This is a recursive process; transformed problems can again be fed into the problem solving cycle. Cases that match the transformed problem specification are retrieved, and adapted so that they form a solution to the original problem. This will be explained in more detail in sections 4, 5 and 6.

The choice which of the creative solutions to use as decisions for the users of the creative problem solver (i.e., the character and plot agents) should be informed by con-

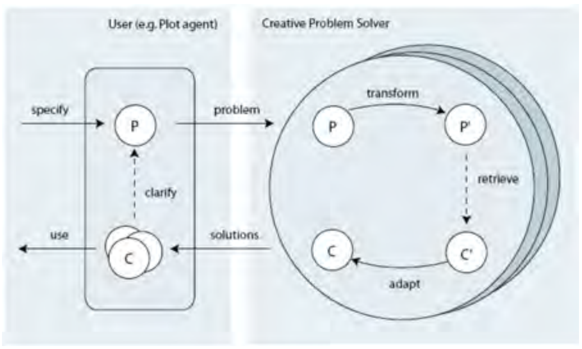

Figure 1: The process of decision making using the creative problem solver

straints given by the users. Inherent to unconstrained creativity is that creativity errors can occur. A child having a limited concept of objects and holes could have a creative idea to fit a square peg through a round hole. The idea is understandable, but reality proves that the solution does not work. By constraining the valid solutions, we can decrease the number of creativity errors. These constraints could be given by a domain-specific model of the 'impossibilities' of the domain or by the state of the story world at a particular moment in time. If a solution meets the constraints, it can be used as a decision. If not, the problem can be *clarified* by extending it with extra information in the form of new constraints. For instance, it could be that a solution for the knight to use a sword to kill himself does not work, because the story world contains no sword. The problem description should at that point be clarified: 'find solutions that do not contain a sword'.

4 Knowledge Representation

A case in the context of our system expresses an example story piece in a formalized language. As discussed before, this piece should express both believable behaviour of the character(s) partaking in it, and the narrative context of the event sequence itself. We will first discuss a knowledge representation used to express such story pieces, and then give formal definitions of case and problem representations.

4.1 Fabula Representation

In the ability to interpret a sequence of events as a story, causality between events plays a major role (Trabasso and Nickels, 1992). Indeed, the story generation system MAKEBELIEVE (Liu and Singh, 2002) is able to create simple stories using only facts about causality between situations as represented in a large common sense knowledge base. From the viewpoint of narratology, a distinction is often made between the *fabula* of the story, a series of causally and chronologically related events that are caused or experienced by characters in a story world, and the *sjuzet*, a dramatic and subjectified abstraction of the fabula.

The Virtual Storyteller produces fabula which is subsequently fed to processes that select and narrate a sjuzet.

We capture the fabula of our simulated story world in the form of a knowledge representation that is discussed in Swartjes and Theune (2006). This representation captures the temporal-causal course of events in the story world. The representation is given by a quadruple $< E, T, C, D >$ where E is a set of fabula elements, T is a set of temporal annotations to these fabula elements, C is a set of causal relationships between fabula elements, and D is a set of descriptive contexts that are linked to the fabula elements and describe their contents. E is divided into six categories: Event, Perception, Internal Element, Goal, Action and Outcome. C is divided into four categories: physical causality, psychological causality, motivation and enablement. Elements of D are subgraphs that can contain fabula as well; this allows for embedded expressions like:

> "The bank owner believes *that it is the bank robber's goal to be rich, and that that goal motivates an action for the robber to rob the bank*; this belief psychologically causes the bank owner to be scared."

The fabula representation is based on a cognitive model for the comprehension of simple stories (Trabasso and Nickels, 1992). This model describes the causal connections that children ascribe to a series of events in a picture story in their attempt to understand the story that the picture sequence conveys. The Virtual Storyteller generates similar causal connections; a fabula is the result of logging the causality between dramatic events that happen in the story world, the resulting beliefs and emotions of the character agents, their goals, attempts to reach these goals in the form of planned actions, and the outcomes of these goals once a goal is achieved or abandoned.

4.2 Problem and Case Representation

For a smooth integration of the problem solver with the Virtual Storyteller, the fabula representation is used to express both the cases in the case base of the creative problem solver, and the problems that should be solved. The problems can be constructed by the agents that use the problem solver; the cases are in principle constructed by a human author.

A problem P is a tuple $< Pat, Con >$ where Pat is the pattern space defining knowledge that must occur in the solution, and Con defines the constraint space in the form of a pattern that should *not* occur in the solution. Both Pat and Con are expressed in terms of fabula but may contain uninstantiated elements.

A case C, also expressed in terms of fabula, has the following requirements:

- C demonstrates a *narrative concept*. A narrative concept could for instance be 'hiding from a threat' or 'flying over an area to search for something'. The expression of a narrative concept is an implicit description of an example problem and a solution to it. The problem is for instance the threat, and the solution is to hide. Of course, there can be many cases demonstrating the same narrative concept.
- C is *context complete* with regard to its narrative concept. Cunningham (1998) states that the case representation must capture the predictive features of a problem. Applied to storytelling problems, we define a context complete case as a case that contains all the elements that are necessary for the case to be viewed as 'believable' by the author of the case, regarding the narrative concept it is supposed to express, and contains nothing more than that.

5 Creativity Heuristics

Unlike in standard CBR, MINSTREL uses creativity heuristics that actively transform problems to find cases instead of finding and adapting cases based on similarity metrics. The reasons for this given by Turner are that truly creative solutions are not found if the problem stays intact, and that adaptation of retrieved cases to fit the problem specification is very difficult. By using transformations and a simpler retrieval mechanism, the adaptation of retrieved cases can be done by reverse application of the transformations.

To guide the transformation of the problem space, we use creativity heuristics similar to Turner's TRAMs. The heuristics are domain-specific and provide a way to transform a problem P into a similar problem P'. When P' enables the system to retrieve a case C, the used heuristic defines a way to apply a reverse transformation to C to create C' which forms a solution to P. When searching for a solution, the creative problem solver can use any applicable creativity heuristic to transform the problem. Problems can undergo a series of these transformations in succession, although too many transformations will make the problem too dissimilar from the original problem and cause found solutions to be unsuitable to solve the original problem. Therefore, the number of successive transformations are limited.

Our creativity heuristics implement the following steps:

Match determines if the heuristic is applicable to the current problem;

Transform defines how the problem space is transformed;

Retrieve finds the cases that match the transformed problem;

Adapt defines how a retrieved case is adapted to the original problem by applying the transformation in reverse.

Turner (1994) has implemented and evaluated a number of creativity heuristics in MINSTREL. Most of them can be divided into a number of functional categories: relaxation, generalization, substitution of a similar subpart and planning knowledge. We will show how we have adapted two of MINSTREL's heuristics to the problem solving domain of the Virtual Storyteller: Generalize Actor and Switch Intention. We will discuss these heuristics below.

5.1 Generalize Actor

Some of MINSTREL's most useful heuristics involve generalization: moving from a specific problem to a more general problem. Elements that can be generalized in our fabula representation are for instance objects in the story world, character roles, actions and their actors, represented in terms of domain specific OWL ontologies. Generalization heuristics assume that a problem definition will remain valid when one such element is replaced by a generalization of that element. An example is shown for generalization of the actor of an action (generalizing other elements proceeds in a similar fashion):

Match: The problem should contain at least one action individual with an `agens` relation to an actor individual. The `agens` property of an action refers to the character performing the action.

Transform: Select one such actor individual at random, and replace its type by a generalized type based on its ontology hierarchy. Only generalize a type one step up in the hierarchy.

Retrieve: For each of the cases in the case base, check if part of the case description unifies with the pattern space Pat of the transformed problem description, and the case description does not contain knowledge specified in the constraint space. Select the cases for which this holds.

Adapt: Identify the actor individuals from the retrieved case whose type matches the generalized type created by the Transform step. Replace all such actor individuals and their type with the actor individual and type from the original problem.

Consider a fragment of a problem description expressing "A princess runs away from a dragon." Examples of actor and action generalizations of this fragment (generalized element in italics):

1. "A princess runs away from *a monster*," leading to the retrieval of cases about princesses running away from orcs and trolls;
2. "*A woman* runs away from a dragon," leading to the retrieval of cases about queens, shepherdesses or little girls running away from dragons.
3. "A princess *moves* away from a dragon," leading to the retrieval of cases about princesses walking, sailing or swimming away from dragons;

5.2 Switch Intention

Switch Intention is an example of a heuristic that substitutes a subpart of a problem for a subpart that is similar in meaning. Switch Intention is based on the idea that if something happens unintentionally, one can try to make this happen intentionally. Actions can cause events that the agent did not intend or expect, or failed to take into account. Cases describing this can be transformed into cases where these events *are* intended:

Match: The problem should contain at least one goal - uninstantiated action - positive outcome combination. In other words, the problem is about "what action brings the goal to a successful outcome?"

Transform: Using the goal found in the match step, find an event that achieves that goal[3]. Construct a new problem containing an uninstantiated action that unintentionally causes the found event. The new problem is about "what action can cause an (accidental) event that brings the goal to a successful outcome?"

Retrieve: Similar to the Retrieve step of the Generalize Actor heuristic.

Adapt: Replace the original uninstantiated action with the action found in the retrieved case, and add the fact that the original goal (from the match step) motivates this action.

This heuristic could be used in a context like the following. A little princess wants to go play outside but needs to ask the king for permission. When she finds him, he is sound asleep in his chair. What can the princess do to wake up the king? The Switch Intention heuristic is based on the premise that instead of executing an action to wake up the king in some way, maybe she can cause an event that wakes him up. If the case base contains a case where a burglar wants to close the door and therefore slams it shut, accidentally waking up the house owner, this case can be transformed using the Switch Intention heuristic so that the princess will slam the door shut in order to wake up the king.

6 Implementation

A prototype implementation has been developed in Java. The ontolologies of the fabula and the objects in the story world are expressed in OWL-DL. At the moment, these are simple ontologies that do not use the full extent of OWL-DL's expressive power. The fabula representation used to describe the cases and problems are expressed as named RDF graphs to express modality (Carroll et al., 2005).

An RDF graph is a set of triples $< S, P, O >$ expressing subject, predicate and object of a fact, respectively. A named RDF graph is an RDF graph with an identifier that can be referred to. Each triple that is part of a named RDF graph can be described as a quadruple (called *quad*) $< G, S, P, O >$, with G being the graph identifier. This contextualizes statements and allows for expressing information *about* triples by referring to their graph identifier. So not only can we express that a certain goal to attain some state motivates an action to walk somewhere:

```
(maingraph
     goal.2 motivates action.9)
(maingraph
     goal.2 rdf:type AttainGoal)
(maingraph
     action.9 rdf:type Walk)
```

[3]This follows from the effects of the events if the events are represented as planning operators.

```
...
```

but we can also express what the goal *means* (for instance, the goal is that the princess has a friend):

```
(maingraph
     goal.2 hasContents graph.5)
(graph.5
     princess.1 hasFriend friend.5)
...
```

The prototype implementation is based on Jena[4] and uses a process of querying (using the SPARQL query language[5]) and transformation (using low-level quad replacement) of these sets of quads. The match step queries the pattern space of the problem description, and one result of this query is chosen at random. This result is a set of variable bindings; the quads that were used to bind these variables are mapped to their transformed counterparts (e.g., by looking up the superclass of an individual's type). The mapping is remembered for the adaptation step. In this step, the original quads are used to replace quads from the retrieved case by quads with their original values filled in.

We have implemented the problem solving cycle using standard breadth-first search, assuming that solutions that are not found with a limited number of successive transformations, will be too far off to form a good solution to the problem being solved. Preliminary results indicate that even with two generalization heuristics and a limit on the number of transformations, the search space quickly becomes quite big, due to the fact that there are many ways in which one heuristic may match the problem description. Even a simple problem description may contain quite a number of objects and actions that can be generalized. For application in the Virtual Storyteller in reasonable processing time, we might have to make concessions in the number of calls to the creative problem solver and the creative possibilities of it. Another option to limit the processing time is to pre-process transformations of some expected problem descriptions offline.

7 Story Generators as Creative Systems

Wiggins (2001) discusses a formal framework to define and categorize creative systems. Based on the work of Boden (1990), Wiggins discerns the ingredients of an exploratory creative system (i.e., a system that selects and values partial or complete concepts that are found by traversing a conceptual space) and considers transformational creativity (i.e., creativity that changes the rules which define this conceptual space) as exploratory creativity at the meta-level. Placing systems that can automatically generate stories within this framework explicates design choices that are sometimes made implicitly. In the discussion of automated story generation as a creative process, the following terms are relevant:

- $\mathcal{C}$: the conceptual space, which in the case of a story generator can be interpreted as the set of "well-formed stories" for a given domain.
- $\mathcal{R}$: the constraints that define $\mathcal{C}$, which can be interpreted as the rules that determine whether a potential story is well-formed.
- $\mathcal{T}$: the rules that specify how to traverse the conceptual space, which can be seen as the story generation algorithm.
- $\mathcal{E}$: the constraints that evaluate $\mathcal{C}$, which can be seen as the rules that determine the quality of the story.

Riedl and Young (2005) discuss story planning in the context of exploratory creativity. They claim that the evaluation criteria $\mathcal{E}$ are not generally known or knowable in the domain of storytelling. The rules that constitute $\mathcal{E}$ (i.e., define the quality of a story) seem indeed difficult to formalize. This is why a creative story generation system must somehow be set up in such a way that it does not rely on evaluation by $\mathcal{E}$. The implication is that the system must deliver good stories without having knowledge about *why* they are good. The input of the system should already be fertile with narrative potential and the developer of the system therefore also becomes, in a sense, responsible for the quality of the produced stories. Taking this position emphasizes the role of authoring: an effective story generation process should be transparent enough for the developer to have an understanding of the relationship between certain input and their effect on the generated stories.

However, what *is* possible to some extent is to determine $\mathcal{R}$, i.e., the "well-formedness" of the generated stories. This at least allows a story generator to explore a space of possible stories. Story generation systems often use formalizations of $\mathcal{R}$ based on findings in narratology (e.g., Propp) or story understanding (e.g., story grammars as used by Lang (1999)) . The Fabulist system (Riedl and Young, 2005) is based on two criteria: character believability and plot coherence. The first criterium (say, $\mathcal{R}_1$) is formalized by requiring that every action in the story is intended by a character; the second criterium (say, $\mathcal{R}_2$) is formalized by requiring that every action has a direct or indirect causal relation to the outcome of the story. Such a formalized $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ enables the traversal of the conceptual space by means of a story planner $\mathcal{T}$. The fulfilment of $\mathcal{R}_2$ is a direct result of using a Partial Order, Causal Link planner (POCL) which starts its planning process from the outcome of the story and plans its way back satisfying causal requirements. Such a planner would never incorporate plan steps that have no causal relationship with the outcome.

Our approach to story generation can be viewed as a process of *exploratory* creativity. We adopt criteria similar to Riedl and Young (2005) for the well-formedness of a story, and traverse the conceptual space by using autonomous characters that are designed to meet $\mathcal{R}_1$, i.e., to be believable. We hope to meet $\mathcal{R}_2$ by equating the outcome of the story with the outcome of a particular important goal of one of the characters. Our approach does not ensure that $\mathcal{R}_2$ is met, since it is easy to imagine other characters doing things that have no relevance whatsoever to the outcome of the chosen goal. But if a certain story does not meet $\mathcal{R}_2$ (i.e., the story contains parts that have no causal relation to the outcome), certainly we can

[4]http://jena.sourceforge.net

[5]See http://www.w3.org/TR/rdf-sparql-query/

find a subset within the event sequence that *does* meet $\mathcal{R}_2$ (namely, only those parts that *are* causally related to the outcome of a certain goal), as long as we make this causality explicit. For a more detailed explanation of these considerations, see (Swartjes and Theune, 2006).

7.1 The Creative Problem Solver as Creative System

Our creative problem solver is a creative system in its own right, performing a process of *transformational* creativity. In this case, we have different interpretations of Wiggins' terms:

- $\mathcal{C}$: the conceptual space, being the set of solutions in terms of narrative cases.
- $\mathcal{R}$: the constraints that define which solutions are in $\mathcal{C}$, which can be interpreted as the problem description.
- $\mathcal{T}$: the rules that specify how to traverse the conceptual space, which in this case is simply the retrieval query.
- $\mathcal{E}$: the constraints that evaluate $\mathcal{C}$, which can be seen as the rules that determine the quality of the found solutions.

A problem description $\mathcal{R}$ defines what a concept case c should look like (thus being a constraint on the conceptual space $\mathcal{C}$). We then transform this problem description into a different one $\mathcal{R}'$ on the assumption that it is a similar problem description. Effectively we have transformed the rule that determines what concept cases are in $\mathcal{C}$. This makes the process one of transformational creativity. And this process of transformational creativity is indeed an exploratory creativity process at the meta-level if we consider the problem space $\mathcal{R}$ to be the conceptual space of this meta-creative system: we use creativity heuristics – the rules of the transformational creativity – to explore the set of possible rule sets. We enter difficult terrain here as neither the rules that select appropriate problem descriptions nor the evaluation criteria for these problem descriptions being valued are easy to formalize. We make a (possibly invalid) assumption that if a problem description $\mathcal{R}$ is appropriate, then the transformed problem description $\mathcal{R}'$ is also appropriate, by carefully defining the creativity heuristics. This explicit design effort is in line with Wiggins' view on transformational creativity, where the designer needs to be aware of the rules being applied rather than rely on serendipity (Wiggins, 2001).

8 Conclusion

The Virtual Storyteller generates stories based on two criteria for their well-formedness: they must portray believable character behaviour and contain coherent plots. Believable character behaviour is achieved by simulating a story world in which autonomous character agents try to achieve their goals and respond emotionally to their environment. A coherent plot is formed by a subset of the fabula generated by the simulation that contains a coherent causal chain (e.g., a chain of events that have contributed to the outcome of an important goal of one of the characters).

We extend the design of the Virtual Storyteller with a creative problem solver that allows for decision making that falls outside the range of decisions that would have been made using only autonomous character agents. The problem solver uses example story pieces that can provide solutions to decision problems that occur in the course of the simulation.

Adding the creative problem solver to the Virtual Storyteller addresses two issues. First of all, a known risk of using autonomous character behaviour to generate stories is that no interesting story emerges. We think that example story pieces, written from a story perspective, allow for more interesting stories to happen, since they define character behaviour in a story context that transcends the decision space of the characters' individual behaviour. Second, the designer of the story generation system is in a sense responsible for the production of good stories, since it is difficult to formalize rules that assess the quality of a story. This stresses the importance of authoring: the architectures and input knowledge of the story generator should afford the designer-as-author to express his narrative intent. We believe that example story pieces form an intuitive way to write such input knowledge.

The creative problem solver performs a process of transformational creativity to extend the range of use of these story pieces, by transforming input problems into problems that are similar in meaning, and adapting found cases to provide a solution to the original problems. This process is guided by explicitly defined creativity heuristics. However, the use of such explicitly defined heuristics does not guarantee finding correct solutions. For instance, the assumption that limiting the number of transformations keeps the solution applicable for the problem at hand, might fail. If the concept of 'dragon' is generalized to 'organism', the system might come up with a creative solution where a knight eats a princess to satisfy his hunger, because it uses a case where a dragon did the same. Evaluating the solutions and tuning the ontologies, heuristics and search control to decrease errors is therefore important. We have designed and partially implemented the problem solver; future work will involve further implementation and evaluation of the problem solver in the context of the Virtual Storyteller.

Acknowledgements

This work has been funded by the Strategic Research Orientation NICE of CTIT, the Netherlands.

References

Aylett, R., Louchart, S., Dias, J., Paiva, A., and Vala, M. (2005). FearNot! - an experiment in emergent narrative. In *Proceedings of the 5th International Workshop on Intelligent Virtual Agents*, pages 305–316.

Boden, M. (1990). *The Creative Mind: Myths and Mechanisms*. Abacus, London.

Carroll, J. J., Bizer, C., Hayes, P., and Stickler, P. (2005). Named graphs. *Journal of Web Semantics*, 3(4).

Cunningham, P. (1998). CBR: strengths and weaknesses. Technical report, University of Dublin, Computer Science Department.

Fairclough, C.R. Cunningham, P. (2004). AI structuralist storytelling in computer games. Technical report, University of Dublin, Computer Science Department.

Gervás, P., Díaz-Agudo, B., Peinado, F., and Hervás, R. (2004). Story plot generation based on CBR. *Knowledge-Based Systems*, 18(4-5):235–242.

Lang, R. R. (1999). A declarative model for simple narratives. In *Proceedings of the AAAI Fall Symposium on Narrative Intelligence*.

Liu, H. and Singh, P. (2002). MAKEBELIEVE: Using commonsense knowledge to generate stories. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI 2002)*.

Loyall, A. B. (1997). *Believable Agents: Building Interactive Personalities*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

Meehan, J. (1981). TALE-SPIN. In Schank, R. and Riesbeck, K., editors, *Inside computer understanding - five programs plus miniatures*, pages 197–226. Lawrence Erlbaum Associates.

Moraes, M. C. and Costa, A. C. d. R. (2004). Imp-BDI: Improvisational BDI architecture. In *Proceedings of the workshop on Architectures and Methodologies for Building Agent-Based Learning Environments*.

Mueller, E. (1987). *Daydreaming and computation: a computer model of everyday creativity, learning, and emotions in the human stream of thought*. PhD thesis, University of California.

Peinado, F. and Gervás, P. (2006). Minstrel reloaded: from the magic of lisp to the formal semantics of OWL. In *Technologies for Interactive Digital Storytelling and Entertainment (TIDSE)*.

Propp, V. (1968). *Morphology of the folktale*. University of Texas Press.

Riedl, M. and Young, M. (2005). Story planning as exploratory creativity: Techniques for expanding the narrative search space. In *Proceedings of the 2005 IJCAI Workshop on Computational Creativity*.

Swartjes, I. and Theune, M. (2006). A Fabula Model for Emergent Narrative. In *Technologies for Interactive Digital Storytelling and Entertainment (TIDSE)*.

Theune, M., Rensen, S., op den Akker, R., Heylen, D., and Nijholt, A. (2004). Emotional characters for automatic plot creation. In *Technologies for Interactive Digital Storytelling and Entertainment (TIDSE)*.

Trabasso, T. and Nickels, M. (1992). The development of goal plans of action in the narration of a picture story. *Discourse Processes*, 15:249–275.

Turner, S. R. (1994). *The creative process: a computer model of storytelling*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Wiggins, G. A. (2001). Towards a more precise characterisation of creativity in AI. In Weber, R. and von Wangenheim, C. G., editors, *Case-Based Reasoning: Papers from the Workshop Programme at ICCBR'01*, pages 113–120, Washington, DC.