Generating Plots for a Given Query Using a Case-Base of Narrative Schemas

Pablo Gervás[1], Raquel Hervás[2], and Carlos León[2]

[1] Instituto de Tecnología del Conocimiento, Universidad Complutense de Madrid
Ciudad Universitaria, 28040 Madrid, Spain
pgervas@ucm.es
[2] Facultad de Informática, Universidad Complutense de Madrid
Ciudad Universitaria, 28040 Madrid, Spain
raquelhb@fdi.ucm.es,cleon@ucm.es
WWW home page: http://nil.fdi.ucm.es/

Abstract. Computational generation of literary artifacts very often resorts to template-like schemas that can be instantiated into complex structures. With this view in mind, the present paper presents a case-based reasoning solution that builds a plot line to match a given query, expressed in terms of a sequence of abstraction of plot-bearing elements of a story, by retrieving and adapting templates for narrative schemas from a case-base. The abstractions of plot-bearing elements of a story are defined in terms of Propp's character functions. The case-base of narrative schemas is built based on a review of a number of existing attempts to provide an elementary set of patterns for basic plots. A selection of these patterns, reformulated in terms of Propp's character functions, is used as case-base. The paper explores a solution for automatic generation of stories based on this formulation of the narrative schemas.

Keywords: computational creativity, narrative, narrative schemas, transformational case adaptation, compositional case adaptation

1 Introduction

Humans that write stories reuse material from stories they know. This may include characters, settings, scenes, or lines of dialogue. Of these, the most important is the reuse of story structure. In order to capture computationally this type of reuse of experience, an abstract representation of story structure is needed. The present paper describes a case-based solution for story generation that relies on Vladimir Propp's Morphology of the Folk Tale [13]. A case-base of narrative schemas described using this representation [5] is used to provide plot lines to match a query, and the plot lines are then fleshed out into full stories by instantiating the abstract plot line with specific story actions [4].

2 Previous Work

To support the approach followed in this paper, four areas of previous work need to be considered: case-based approaches to story generation, Propp's formalism

for analysing stories, the Propper system for story generation, and existing approaches to case adaptation.

2.1 Case-Based Approaches to Story Generation

Roger Schank stated that the way in which memory works is not only based on processes that manipulate mental data, but instead as continuous recalling and adapting process of previous stories that define our world [18, 17]. Turner's MINSTREL exemplified this approach by generating short stories about King Arthur and his Knights of the Round Table [19]. MINSTREL handled episodic memories in two different ways: either by instantiating a matching schema in the story from a basic query, or by performing a basic adaptation on the query, querying the episodic memory with it and returning an adaptation and modification of the query. Knowledge intensive case-based reasoning approaches [3, 10, 11] use Semantic Web technologies for knowledge representation and simple combinatorial algorithms for generating the structure of new plots by reusing fragments of structure of previous stories, inspired in the morphology of Russian folk-tales studied by Vladimir Propp [13]. Relying on more shallow representations, [14] and [16] introduce a story planning algorithm inspired by case-based reasoning that incorporates *vignettes* – pre-existing short narrative segments – into the story being generated. Other approaches to story generation based on case bases of previous schemas include efforts towards incorporating analogy-based reasoning to knowledge acquisition [9, 15]. These systems are usually focused on the retrieval, adaptation and evaluation of old schemas to new domains. In general, all these approaches rely on inter-domain analogies and generate new instances of old narrative schemas. Reuse of previous stories is also applied in [12], where case-like structures known as Story Contexts are mined from a set of previous stories and used to inform the selection of the next action to add to a story in an incremental generation process.

2.2 Proppian Morphology of a Story

At the start of the 20th century, Vladimir Propp [13] identified a set of regularities in a subset of the corpus of Russian folk tales collected by Afanasiev [1]. These regularities he formulated in terms of *character functions*, understood as acts of the character, defined from the point of view of their significance for the course of the action. According to Propp, for the given set of tales, the number of such functions was limited, the sequence of functions was always identical, and all these fairy tales could be considered instances of a single structure. The set of character functions identified by Propp includes a number of elements that account for a journey (`departure`, `return`), a number of elements that detail the involvement of the villain and the struggle between hero and villain (`villainy`, `struggle`, `victory`, `pursuit`, `rescue_from_pursuit`), a

number of elements that describe the acquisition of a magical agent by the hero (`test_by_donor`, `hero_reaction`, `acquisition_magical_agent`).[3]

2.3 The Propper System

The Propper system developed by Gervás [4] constitutes a computational implementation of a story generator initially based on Propp's description of how his morphology might be used to generate stories.

It relies on the following specific representations for the concepts involved:

- a *character function*, a label for a particular type of acts involving certain named roles for the characters in the story, defined from the point of view of their significance for the course of the action
- a sequence of character functions chosen as backbone for a given story
- possible instantiations of a character function in terms of specific *story actions*, involving a number of *predicates* describing events with the use of *variables* that represent the set of characters involved in the action

Based on these representations the Propper system defines a procedure that first chooses a sequence of character functions to act as abstract narrative structure to drive the process, and then progressively selects instantiations of these character functions in terms of story actions to produce a conceptual representation – in terms of an ordered sequence of predicates – of a valid story. This conceptual representation is a *fabula*, a sequence of states that contain a chain of story actions – which are instances of those character functions. A story action involves a set of preconditions – predicates that must be present in the context for continuity to exist –, and a set of postconditions – predicates that will be used to extend the context if the action is added to it. Each story action is linked to its context of occurrence by having its preconditions satisfied by the preceding state.

2.4 Case Adaptation

Probably one of the most difficult processes in the CBR cycle is the reuse or adaptation stage. After retrieving the most similar case (or cases) from the case base, the solution from the retrieved case must be used to create a new solution for the problem at hand.

Wilke and Bergman [20] established a classification of CBR adaptation into three different methods: null adaptation, transformational adaptation and generative adaptation. The simplest kind of adaptation is *null adaptation*, where the solution of the retrieved case is used without any modification. As simple as this adaptation method is, it can obtain very good results for simple problems. *Transformational adaptation* consists on the transformation of the solution of

[3] For reasons of space, only a number of character functions relevant to the examples given in the paper are described. Readers can check the referenced sources for more detail.

the retrieved case into the solution required for the query. In order to do that, the retrieved solution may be reorganized and modified by deleting or adding new elements. Finally, *generative adaptation* consists on generating the new solution from scratch, but reusing the process used to obtain the solution from the retrieved case.

These three adaptation methods are formalized by considering that only one case is retrieved and adapted. However, some problems may be better solved by reusing information from more than one case. This is what Wilke and Bergman called *compositional adaptation*, where the new solution is obtained by adapting the solutions of multiple cases. This multiple case adaptation can be done using transformational or generative methods, but the main idea is that the solution for the case at hand can be better obtained by taking into account more than one case from the case base.

There are many examples of compositional adaption in recent CBR works. Arshadi and Badie [2] apply this adaptation in a tutoring library system. In this kind of application it is probable that many cases can be similar to the user request at the same time, so it is important to take all of them into account when generating the solution for a given query. Hervás and Gervás [6] also use multiple cases for text generation based on templates. When the information that must appear in a sentence is not covered by the template of the retrieved case, a new retrieval process is triggered in order to find more cases which templates can cover the information in the query. Ontañón and Plaza [8] present the concept of amalgam as a formal operation over terms in a generalization space. Although amalgams are not proposed as an adaptation method by themselves, the notion of amalgam is related to merging operations that can be used in compositional adaptation to combine two or more cases. Müller and Bergmann [7] use a compositional adaptation approach for cooking recipes represented as cooking workflows. During the adaptation stage, missing parts of retrieved cooking workflows are covered using information from other cases.

3 Case-Based Construction of Plot Lines for Stories

The present paper describes a case-based approach to the construction of plot lines for stories – described as sequences of character functions – which can then be fleshed out into stories.

3.1 Case-Based Construction of Plot Lines

The system operates from a query provided by the user. This query is expressed as a sequence of character functions that the user would like to see included in the desired plot line.

The system compares the given query with the set of plot lines represented in its case base.

The Case-Base The case base of schemas used for this paper is built from the narrative schemas reviewed in [5]. These correspond to a set of sequences of character functions – in Propp's sense of plot relevant abstractions of the activity of characters – that correspond to a number of theoretical characterizations of possible plots for stories, also referred as *narrative schemas*. The case-based reasoning approach will therefore operate over sequences of character functions, and it will return a sequence of character functions that best matches the given query.

Merging Plot Lines When dealing with plot lines in terms of sequences of character functions it is often necessary to merge two plot lines to obtain a third plot line. Because plot lines are sequentially ordered, and specific elements in the plot may have dependencies with other elements, the relative order in which they appear in the sequence is very relevant. For the purposes of the present paper, this is done as follows:

- the query is traversed sequentially
- each character function in the query is checked against the next character function in the case
- if they match the character function is added to a *matching* subsequence
- if they do not, the character function from the query is added to a *wanted* subsequence, and the next character function from the query is checked against the character function in the case
- if the end is reached for the query the rest of the case is added as an *added* subsequence
- if the end is reached for the case the rest of the query is added as a *wanted* subsequence

The merge is constructed by concatenating into a single sequence the subsequences of character functions that are generated during the merge in this fashion. This has the advantage of interleaving the character functions from the original query with the contributions from the various cases involved while always respecting the order in which these character functions appeared in the query.

Similarity We consider a similarity function for plot lines based on identifying the relative mutual coverage between query and case. The set of subsequences of the query that appear as subsequences of the case in the corresponding order is referred to as the *match*. The *remainder* is the set of subsequences of the query that are not covered by the case. The *addition* is the set of subsequences of the case that did not appear in the query.

The similarity employed in the current version of the system is calculated as an average between the percentage of the query covered by the case – the ratio between the size of the match and the size of the query – and the percentage of the case that is involved in the match – the ratio between the size of the match and the size of the case. This is intended to capture the suitability of the

case both in terms of maximum coverage of the query and in terms of minimum addition of character functions beyond the query.

To compute these values the query is merged with the case as described above. The match is then reckoned to be the set of *matching* subsequences. The remainder is then reckoned to be the set of *wanted* subsequences. The addition is then reckoned to be the set of *added* subsequences.

Retrieval and Adaptation If there is a case whose plot line matches the query, that case is returned as solution.

Otherwise, the cases are ranked based on their similarity with the query. The set of character functions that appears in the overall set of subsequences resulting from this process constitutes a possible solution to the problem posed by the query, as it would constitute a combination of the query and the case.

If the retrieved case does not cover all the character functions in the query, further retrieval processes will be required. This corresponds to solving the given query with a complex story that combines more than one plot line. To achieve this, an additional retrieval process is set in motion using the remainder of the first retrieval process as a query to the second one.

For each additional case retrieved, the resulting solution is merged with the result of prior stages using the same procedure as for merging a query and a case. These ensures that relative order of appearance of related character functions within each narrative substructure that has been reused is respected in the final solution.

The retrieval and adaptation process can be iterated until the remainder of the query is empty. The merge obtained at this point is the final solution. This sequence of character functions is the solution found by the system as plot outline for a story to match the given query.

An Example of Plot Line Construction For a query `villainy departure villain_punished return`, the most similar case retrieved is:[4]

`villainy` *`hero_pursued rescue_from_pursuit struggle victory`* `villain_punished`

The merge of the query and case, with the different subsequences marked[5] is:

`villainy` `DEPARTURE` *`hero_pursued rescue_from_pursuit struggle victory`* `villain_punished` `RETURN`

Within the resulting merge, the elements not matched by the retrieved case (the remainder: `departure return`) appear in the same relative position with respect to the other elements of the query as they did in the original sequence of the query.

[4] Elements in the case that match the query are shown in plain text, and elements that do not are shown in italic.

[5] `Matched` elements are shown in plain text, `WANTED` elements in small caps, and *`added`* elements in italic.

To cover this remainder, a second case-based reasoning process is set in motion, with the remainder as a query. For this second process, the query would then be DEPARTURE RETURN. The most similar case retrieved is:[6]

departure *difficult_task task_resolved* hero_pursued rescue_from_pursuit struggle victory *test_by_donor hero_reaction acquisition_magical_agent* return

The merge of this additional case with the result of the prior CBR process, with the different subsequences marked as above is

villainy departure *difficult_task task_resolved* hero_pursued *rescue_from_pursuit struggle victory* villain_punished *test_by_donor hero_reaction acquisition_magical_agent* return

This implies that the remainder is now empty.

3.2 Fleshing out the Plot Line for the Story

Because character functions are abstractions of plot relevant activities by the characters, the draft plot line obtained as a result of the retrieval and adaptation stage needs to be fleshed out before it can be considered a story.

This involves instantiating the character functions with specific story actions. This can be done following the original procedure for the Propper system [4] for obtaining a fabula from the sequence of character functions corresponding to the resulting plot line. This relies on definitions of the story actions defined in terms of predicates that define an action, with identifiers for the characters as arguments. The definitions of these story actions also contain predicates that define preconditions and effects of the action in question. The instantiation procedure relies on unification of each new story action with the previous context to guarantee continuity and coherence in terms of which characters perform which actions.

Table 1 presents an example of story corresponding to the plot line obtained as a result of the case-based reasoning procedure described in section 3.1.

It is worth noting that although the character functions being instantiated arise from two different original plot lines as provided by the cases, the fleshing out procedure instantiates them with story actions that link up to conform a single coherent story about a hero (character id147) and a villain (character id755). An initial villainy (state 0) forces the hero to set out (state 1), he faces a difficult task (states 2-3), he undergoes several conflicts with the villain (states 4-5 and 6-7). The end of this particular story involves a meeting with a donor that provides a magical agent (states 9-11) and an eventual return of the hero (state 12).

4 Discussion

The approach followed for case adaptation in the described procedure is transformational and compositional. Both the transformation of the retrieved cases to

[6] The use of italics shows the match of the retrieved case with the sequence resulting from the earlier CBR process.

State	Event description
0	character id755 kidnap id755 id756 victim id756 character id756 misbehaved id755
1	seeker id147 character id147 sets_out id147
2	sets id181 id147 id183 character id181 id183 difficult_task id183 involves id183 manufacture
3	character id181 solve id147 id183 before dead_line
4	runs_away id147 pursues id755 id147 hides_in id147 tree tries_to_destroy id755 tree
5	jumps_to_another tree escapes id147
6	weight_contest id147 id755 confrontation id147 id755
7	makes id147 protective_gesture banishes id755
8	pardoned id755
9	shows id388 id389 donor id388 character id388 magical_agent id389 offers_exchange id388 id389 id147 test id388 id147
10	agrees_to_exchange id147 uses id147 id389 id388 deceives id147 id388 positive_result id147
11	helper id53 character id53 meets id755 id53 offers_services id53 id755
12	returns id147

Table 1. An example story corresponding to the plot line shown earlier.

better match the query and the composition of more than one case are covered by the described procedure for merging two sequences of character functions while respecting the relative order of appearance of their elements.

The procedure followed for story construction operates at a higher level of abstraction than [19, 14, 16], and with greater flexibility than [3, 10, 11] – who also use character functions – due to its highly compositional approach to case recombination.

The case-based reasoning procedure described relies on cases to provide a complete backbone for a plot line, reusing the structure of a given plot completely, with no option for leaving out certain parts of it. The procedure for successive retrievals, together with a merging approach that respects the relative order in which character functions occur in the query and interleaves the additions without repetition, allow for more than one such plot backbone to be combined into more complex stories. However, this approach will only succeed as long as there exists some case in the case base with a reasonably similar sequence of character functions. Beyond this, it might be necessary to consider alternative approaches that allow reuse of fragments of cases, to be recombined into longer sequences.

The choice of case base employed here is built from schemas that are intended as complete plots. Alternative formulations of the case base are possible, built from smaller units of plot, such as scenes. These might be represented as subsequences of character functions that occur frequently in different plot lines. A solution along these lines might define the case base in terms of smaller units that would be abstracted during the construction of the case base. This procedure is similar to the one employed in [12], where cases are retrieved to generate the actions of the story one by one (one case per action). An alternative procedure would be to operate over a case base of complete plots but define

a different retrieval algorithm that allows a certain fragmentation of these plots during retrieval.

Two important aspects to consider in creative plot generators are coherence and novelty. By virtue of its process of reusing large segments of existing plots, the described procedure is likely to generate coherent plots, though how coherence is affected by the merging procedure should be addressed in further work. In that sense, the process of instantiation with story actions employed by the Propper system presents an advantage in that it checks the satisfaction of preconditions of each action in its context during construction. With respect to novelty, processes that reuse existing solutions are exposed to the risk of reproducing aspects of prior material. To address this risk, future work should consider establishing limits on the extent of reuse considered. These could take the form of avoiding cases that are perfect matches for a given query, and preferring solutions obtained by combination of more than one case.

5 Conclusions

The case-based reasoning solution described in this paper operates at a sufficiently high level of abstraction to allow the construction of valid plot lines by combination of cases that represent narrative schemas which are merged into a plot line that matches a given query, and which can then be instantiated into a specific coherent story.

Acknowledgements

This paper has been partially supported by the project WHIM 611560 funded by the European Commission, Framework Program 7, the ICT theme, and the Future Emerging Technologies FET program.

References

1. Alexander Nikolayevich Afanasyev. *Narodnye russkie skazki A. N. Afanaseva [Folk Russian tales of A. N. Afanasev]*, volume 1-3. Moscow, 1855.
2. Niloofar Arshadi and Kambiz Badie. A compositional approach to solution adaptation in case-based reasoning and its application to tutoring library. In *Proceedings of the 8th German Workshop on Case-Based Reasoning, Lammerbuckel*, 2000.
3. P. Gervás, B. Díaz-Agudo, F. Peinado, and R. Hervás. Story Plot Generation Based on CBR. *Knowledge-Based Systems. Special Issue: AI-2004*, 18:235–242, 2005.
4. Pablo Gervás. Computational Drafting of Plot Structures for Russian Folk Tales. *Cognitive Computation*, 07/2015 2015.
5. Pablo Gervás, Carlos León, and Gonzalo Méndez. Schemas for narrative generation mined from existing descriptions of plot. In *Computational Models of Narrative*, Atlanta, Georgia, USA, 05/2015 2015. Scholoss Dagstuhl OpenAccess Series in Informatics (OASIcs), Scholoss Dagstuhl OpenAccess Series in Informatics (OASIcs).

6. Raquel Hervás and Pablo Gervás. Case-based reasoning for knowledge-intensive template selection during text generation. In Thomas R. Roth-Berghofer, Mehmet H. Göker, and H.Altay Güvenir, editors, *Advances in Case-Based Reasoning*, volume 4106 of *Lecture Notes in Computer Science*, pages 151–165. Springer Berlin Heidelberg, 2006.
7. Gilbert Müller and Ralph Bergmann. Compositional Adaptation of Cooking Recipes using Workflow Streams. In *Computer Cooking Contest, Workshop Proceedings ICCBR 2014*, Springer, 2014. The original publication is available at www.springerlink.com.
8. Santiago Ontañón and Enric Plaza. Amalgams: A formal approach for combining multiple case solutions. In Isabelle Bichindaritz and Stefania Montani, editors, *Case-Based Reasoning. Research and Development*, volume 6176 of *Lecture Notes in Computer Science*, pages 257–271. Springer Berlin Heidelberg, 2010.
9. Santiago Ontañón and Jichen Zhu. On the role of domain knowledge in analogy-based story generation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1717–1722. AAAI Press, 2011.
10. F. Peinado and P. Gervás. Evaluation of automatic generation of basic stories. *New Generation Computing*, 24(3):289–302, 2006.
11. Federico Peinado. *Un Armazón para el Desarrollo de Aplicaciones de Narración Automática basado en Componentes Ontológicos Reutilizables*. PhD thesis, Universidad Complutense de Madrid, Madrid, 2008.
12. Rafael Pérez y Pérez and Mike Sharples. MEXICA: A computer model of a cognitive account of creative writing. *Journal of Experimental & Theoretical Artificial Intelligence*, 13(2):119–139, 2001.
13. Vladimir Propp. *Morphology of the Folk Tale*. Akademija, Leningrad, 1928.
14. M. Riedl and Carlos León. Toward vignette-based story generation for drama management systems. In *Workshop on Integrating Technologies for Interactive Stories - 2nd International Conference on Intelligent Technologies for Interactive Entertainment*, 2008.
15. Mark Riedl and Carlos León. Generating story analogues. In *AIIDE*, 2009.
16. Mark O. Riedl and Neha Sugandh. Story planning with vignettes: Toward overcoming the content production bottleneck. In *Interactive Storytelling*, volume 5334 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2008.
17. R. Schank. *Dynamic Memory : A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, 1982.
18. R. Schank and R. Abelson. *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. L. Erlbaum, Hillsdale, NJ, 1977.
19. Scott Turner. *MINSTREL: A Computer Model of Creativity and Storytelling*. PhD thesis, University of California at Los Angeles, Los Angeles, CA, USA, 1992.
20. Wolfgang Wilke and Ralph Bergmann. Techniques and knowledge used for adaptation during case-based problem solving. In *Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial In Telligence and Expert Systems: Tasks and Methods in Applied Artificial Intelligence*, IEA/AIE '98, pages 497–506, London, UK, UK, 1998. Springer-Verlag.