The Affective Storyteller: How Emotion Influences Narrative Generation

F.C.A. Kaptein

Faculty EEMCS

TU Delft Delft University of Technology

Challenge the future

The Affective Storyteller: How Emotion Influences Narrative Generation

by

F.C.A. Kaptein

in partial fulfillment of the requirements for the degree of

Master of Science
in Computer Science

at the Delft University of Technology,
to be defended publicly on 20 August, 2015 at 15:30.

Supervisor:	Dr. ir. J. Broekens	
Thesis committee:	Dr. ir. K. Hindriks,	TU Delft
	Dr. ir. M. Theune,	University of Twente
	B. Kybartas,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft Delft University of Technology

PREFACE

This thesis is the result of a Master of Science research project at the University of Technology at Delft, The Netherlands. This project was carried out at the faculty of Electrical Engineering Mathematics and Computer Science. It was carried out for a master of Computer Science with the track Media and Knowledge engineering and with the specialization Interactive Intelligence.

I wish to thank dr. ir. Joost Broekens for supervising this project and for helping me find this interesting and challenging subject, I would further like to thank my family and friends for their support and for listening to my endless stories about storytelling and their advice which was always helpful.

This work is to the best of my knowledge original, except where acknowledgments and references are made to previous work. Neither this, nor any substantially similar dissertation has been or is being submitted for any other degree, diploma or any other qualification at any other university

F.C.A. Kaptein
Delft, August 2015

ABSTRACT

In this research it is investigated how character emotions can be used to aid the narrative generation process. To this end the Affective Storyteller is developed, a narrative generation framework where character emotions are closely interconnected with the narrative generation process. For story generation in the Affective Storyteller, it is first required to define a story-domain. A story-domain exists out of actors, events and locations. The Affective Storyteller can then generate many different stories given such a domain. A story is formed by a sequence of events. As the amount of possible events in such a sequence grows, the computation time for generating all possible story sequences in the story-domain grows exponentially with it. Dependent on the preferences of the audience such a set of stories typically contains both good, desirable stories and less or unsatisfying stories.

Two main challenges in narrative generation are addressed. Since stories are valued differently dependent on the audiences' preferences, it is difficult to generate stories that fit all these individual needs. When we would be able to steer the type of stories shown at runtime, then this would enable coping with this wide range of preferences. The first challenge we address is therefore *customization*. We aim to show the audience stories that adhere to certain abstract structures. These structures can be build (customized) by the person generating the stories. Another problem in computer-based storytelling is the high cost in computation time and space for longer stories. When presenting stories to users at runtime, it is not desirable if the calculation takes hours. We improve the *efficiency* of the narrative generation by forcing the storyteller to take the requested customizations into account during the generation of the stories. When the storyteller considers storylines that will not adhere to the customization, then the calculation of those storylines can be cut of before completing them.

The Affective Storyteller simulates character emotions using GAMYGDALA, an emotion framework based on the OCC model. The OCC model generates emotions based on how well the characters perform in regards to their goals. These emotions are updated after every story-event, giving us an emotional flow corresponding to the story. We believe this emotional flow to be a powerful abstract representation of the stories. The user can then define a filter that corresponds to these emotional flows. The Affective Storyteller analyzes the emotional patterns and rejects stories that do not comply with a chosen filter.

This research shows different methods to analyze these patterns and filter the proper stories given a set of stories and affective data. Further, some experimental evidence supporting the feasibility of this approach, is presented. We tested the impact of affective filtering of stories on readers' affective perception of these stories, and found a significant effect on the perception of positively versus negatively filtered stories, supporting the validity claim of the filtering method. We further tested the impact of using an affective filter during the generation and reduced the calculation time exponentially in an example domain and filtering heuristic.

Continuing this research would mainly mean to investigate how the filtering can be optimized for use during generation and to what extend it can be used to find stories that follow affective patterns that are known to produce certain qualities in stories. For example excitement or conflict. The overall conclusion for this research is that character emotions are a promising tool for plot management of stories.

CONTENTS

1

INTRODUCTION

Storytelling is a vital part of our everyday lives and has been for as long as humans exist. Stories about Greek mythology, a story of a child coming home from school, or my dog explaining to me that he found something remarkably interesting in the garden. These are all examples of stories, some longer, others shorter, told in different languages but all with one global thing in common. Stories are in essence meant to *share information*. It does not matter whether this is something practical like teaching mathematics or the deeper meanings in stories like Alice in Wonderland or the Lord of the Rings. In this chapter we will first talk about the value of storytelling, explaining why it has always played such an important role throughout history , then we will give some background information on the subject. Finally we will introduce our contributions and give an overview of the remainder of the paper.

1.1. THE VALUE OF STORYTELLING

Why would we want to tell stories in the first place? This is perhaps one of the most important questions to ask ourselves when doing research in narrative generation. Before aiming to answer this scientifically, let's tell a story as illustration:

> *Once upon a time there was a women named Truth. Truth wanted to share her wisdom for she knew it would make the world a better place. So she went into town and to the town square. She gathered the people around and spoke to them but no one would listen. The minute she started talking, people ran away and shouted in terror. People went into their houses and shut the doors and windows. In a matter of minutes there were only very few people left in the streets.*
>
> *What could she do? If people wouldn't even listen to her then how could she deliver her message? In absolute despair Truth did the last thing she could think of to make the people pay attention to her. She took off her clothes and tried again but the result was worse. The people ran away from her. They went into their houses and shut the windows.*
>
> *All alone she cried and was just about to leave town when something remarkable happened. She heard people leaving their houses again. She heard laughter and joy. People were talking to each other and all despair had left their faces. Truth looked around and then she saw a woman walking towards her. The townsmen were following this woman with happiness in their faces.*
>
> *The woman noticed Truth and walked towards her. 'Why are you crying?' She asked. Truth was very upset and cried: 'Everyone was scared of me! No one wanted to listen to my words.' The woman looked sympathetic. 'That is because no one likes the truth. Especially the naked truth.' The woman smiled and gave Truth her beautiful sparkling cloak. 'Try again' she said. Carefully Truth tried talking to the townsmen again and like a miracle, this time people listened. They smiled and nodded by her words. No one ran away. For you see, the woman that gave truth her cloak was Story and when cloaked by story even truth is liked by everyone.*
>
> *- Jewish Story*

This story playfully introduces the importance of storytelling. The story is not so much better than the truth in a message, it's just that the truth needs some form of cloaking to be accepted by people and story is an ideal

way of doing so. Research shows that story can bring people into an altered state of consciousness that makes them lose their awareness of their surroundings [1], which means it can be used to gain and hold someone's attention. It is a powerful tool to influence others and to deliver your message [2–4]. This influence makes storytelling a powerful tool, useful in other fields than only entertainment, such as education, health care, advertising, training and argumentation [5].

1.2. THE ART OF STORYTELLING

Stories have a beginning, a middle and an ending. Stories contain characters who perform different actions that shape the story and are part of some plot. A more formal definition is given here:

Definition 1.1. A story is the presentation of a series of logically and chronologically related events that are caused or experienced by actors [6].

This definition tells us that a story consists of building blocks, events, that have some form of relation between them. In these events there are actors causing or experiencing them (often both). Often these actors will be human-like characters, but this is not always the case. An earthquake could for example be seen as an action performed by the 'actor': *mother nature*. Finally the definition claims a story consists of some presentation. The series of related events is the content of the story. This however holds no information about how this content should be *communicated* to the audience. That is what the term presentation is directed at.

For narratologists such a differentiation, between the generation of the story events and the manner in which these events are presented to the user, is often used [6]. This difference can be defined as fabula and presentation. *Fabula* is the chronological order of all events that occur in the story. It is the content, possibly including background information, of the story. The fabula can be presented in different ways. Examples are: speech, writing, drawing or even (more abstractly) music. These different elements refer to *presentation* and can even be combined (movies combine speech, video and music). A difficult question in storytelling is how events should be ordered and shaped to generate stories that the audience receives as *good*. There is no consensus on how to measure the value of a story [7]. There is however work on how certain story qualities can be controlled. Examples of story qualities are: *suspense*, *humor*, *conflict*, *memorability* and *realism*. We now shortly discuss some of these story qualities.

First, to add *suspense* to a story the amount of paths a hero can go through needs to diminish in the eyes of the audience [8]. That means that two different heroes that need to the do exactly the same to solve their problems might not cause the same perceived suspense by the audience. The hero that is perceived to have lost more solutions to his problems will cause a significantly higher feeling of suspense. Second, the audience will *remember* a passage of the story better when the event has a high amount of causal relations towards other events [9]. A causal relation between event A and event B means that if event A didn't happen, event B would not have happened. These two events will then be deemed more important and are remembered better by the audience. Third, to make a story plausible and interesting, the characters in it need to have plausible reactions respecting their personality traits to the events occurring, characters should have distinctive traits to make them individuals, and they should have emotions [10]. Finally, the writer Jan Veldman states that a storyline needs to have conflict, have friction between the characters and have this well spread throughout the story [11]. So characters need to occasionally cause events that have a negative impact on other characters' goals.

As mentioned above, there is no consensus on how to measure the value of a story [7]. This means we do not know *when* and *how* we should add these story qualities to our own stories. At least not before gaining background information of the audience. There however does seem to be a high demand for control in ones storytelling. If the storyteller is capable of increasing or decreasing on these story values then he would be able to target a much wider range of audiences. In other words there is need for customization of the narrative generation process for different audiences.

1.3. NARRATIVE GENERATION

Algorithms can be used to automatically generate stories, i.e., narrative generation (for a review see [7]). A straightforward method of computerized storytelling is to script one or more possible stories at design time. This is however very limited in the sense that it provides the system with little or no possibilities to adapt to the users' personality. Possibly a more powerful method is to generate them dynamically, based on some story domain. A story can then be seen as a planning problem in where some initial state can be transformed

into a goal state through the execution of different events. In this way computers are capable of generating multiple storylines each in a single story domain. This enables several interesting possibilities, amongst which customizing a story to fit a particular user [12]. In education, for example, generated narrative could aid in delivering user-specific information using user-preferred content. This type of story generation however limits the storyteller in making longer stories due to higher costs in computation time and space. In this research we aim to increase the customization possibilities of computerized storytelling and cope with the computational limitations of planning based storytelling.

1.4. CONTRIBUTION

Computer models can be used to simulate the emotions of virtual characters. This can be seen in storytellers like the virtual storyteller [13] and FearNot! [14]. Emotions are important to have believable coherent characters [10]. In this research it is intended to add an emotion framework, capable of calculating these character emotions, to the fabula planning in order to achieve higher quality permutations of the story. Stories developed in this way can not only narrate how the story develops, but also how these developments affected the characters.

The OCC model [15] is an emotion model that determines affective responses to events based on how they affect the agents' goals. When examining some of the story qualities mentioned in section 1.2, then we envision a link between them and how characters perform on their goals. According to [8] *suspence* in a story is increased when the audience believes that the hero becomes less likely to achieve his goals. As a second example, we may reason that when a character is responsible for diminishing this likelihood of goal achievement, then this results in some form of *conflict* between the responsible character and the character owning the goal.

Do to this relation between character goals and the discussed story qualities, we believe that character goals are a good heuristic for the customization of stories. Since character emotions are directly related to their goals in the OCC model, we propose to use this affective information to improve on the before mentioned aim of customization. We further propose to use these heuristics during the fabula planning in order to decrease the computation time and space. We will *not* discuss the exact relations between the individual story qualities and the customizations, meaning we will not discuss the precise emotions a character should display to add, for example, suspense. Exact implementations of such transitions remain future work.

In section 2 we will discuss the related work in narrative generation and using emotions in the generation process. In section 3 we will discuss the challenges of customization and efficiency more thoroughly. Sections 4 and 5 discuss our method for addressing these challenges. Section 4 talks about the Affective Storyteller framework and how the affective data is incorporated in the storytelling process. Section 5 illustrates methods to use this affective data to customize the stories. In section 6 we show some tests regarding our method and finally in section 7 we discuss the performed work and possible continuations for future work.

2

Related work

2.1. Approaches to Narrative Generation

The first storytelling system documented [16] dates back to 1973 (for a brief history of storytellers see (Gervás 2009)). This storyteller was limited in the diversity of the stories. It was a state machine that told a murder story where the sequence of the scenes was hard-coded. The events in the different scenes were chosen by chance. Later storytelling systems found ways to procedurally choose the different scenes and order of scenes. Characters where defined to be agents with beliefs and actions. With a Least commitment, partial order planner [17] the actions of these agents can be ordered to show all possible stories in the given domain. The domain is written by a human author who now defines a story-world with its characters. Then the possible actions of the characters and the consequences of these actions have to be defined. Given some finishing state, the planning algorithm finds all possible sequences of events.

This way of story-planning however does not yet measure the quality of these different story-lines. Amongst others, concerning believability, characters need to be perceived to be intentional agents or readers might dissociate with the fabula [10]. Modern storytellers have different tactics to cope with this problem. The Virtual Storyteller adds goals to its characters and forces the characters to perform actions according to these goals automatically shaping the story [18]. They added another intentional agent controlling the plot development in order to have well-structured story-lines. The storyteller developed by Riedl and Young [12] has an algorithm that plans the events of the story together with the intentions of the characters. They generate a plan for the narrative in where the actions of the agents are justified in terms of their goals and which works towards some plot. In this way the different story-lines the planner produces are all story-lines with intentional characters.

The plot was however limited because it was not possible for the characters to fail their individual goals. A character will only perform actions that at some point in the story fulfil its goals. This limits the possible story-lines. As mentioned in [19] this however does not likely provide very interesting plots. It even removes the story-lines where characters work against each other, have inner conflicts or simply make a mistake. As a next step CPOCL [20] continues on the IPOCL planning algorithm by adding conflict possibilities to intent driven planning. With this expanded algorithm it is possible to have characters that intend to achieve some goal but fail somewhere along the way making stories with conflict possible.

2.2. Emotions in Synthetic Storytelling

Emotions play an important role in synthetic storytelling. There are two main approaches to incorporate emotions in storytelling [21]. In the first approach user emotion is measured and used to steer the story in desirable directions. Many have tried to adjust the story flow based on user emotions, usually in order to induce certain emotions in the user [22–26]. Typically, such storytellers have a pre-defined directed graph of story-segments. They then steer the story through this graph based on emotional information gotten by analyzing the user. For example, in [24] emotionally annotated pre-authored text and video is used to elicit emotional changes in the user. The annotation is calculated based on the average emotional response over all previous users. Also, the emotional responses of the current user are tracked. If the user reacts in the expected way, the typical next emotion-dependent story segment is given, otherwise another segment is chosen based on reasoning over the dissimilarities. Similarly, in [25] a story is divided in story-segments, each segment

being annotated with the expected change in user emotion. Based on an Emotional Path Graph, defining how the user emotion should develop, and a user emotion tracking engine the story-segment that best fits the emotional path is chosen. In [26] a path is planned through the directed graph of story segments based on the characters' feelings. These feelings are influenced by the emotional behavior of the user which is measured through speech recognition.

The second approach aims to model the emotions of the virtual characters in a story. It is reasoned that in order to make a satisfying story, emotions of characters need to be taken into account [7]. The success of a story is dependent on the understandability and believability of its characters [27]. Clearly, to make a character believable it needs some sort of emotional interpretation of the world [10]. Writers of literature thrive on imagined emotions of their characters. Characters correctly or incorrectly analyze a situation and have a valenced (positive or negative) reaction towards it. This reaction then changes the behavior and/or judgment of the characters. To computationally simulate emotions in artificial characters, cognitive appraisal theory has been used as a psychological basis in many computational models of emotion [14, 28–31], usually based on the OCC model [15]. The OCC model predicts the emotion for an agent based on the agents' social relations and the effects of a given event. Many of these computational models automatically generate emotions given the defined beliefs, goals and social relations between the characters. The different events in the story should update the beliefs of the characters, and the model typically calculates what effects these updated beliefs will have on the characters' goals and given these effects emotions such as hope and fear are generated for the character. In MEXICA [32] the user defines the possible story actions and in addition defines a set of previous stories. MEXICA builds a knowledge base based on the previous stories and uses this to determine the subset of applicable story actions. Of this subset one action is chosen randomly. The storyteller MEXICA was pioneer because it used emotional relations between characters to choose the possible story actions [33]. A phenomenon that became more apparent in later storytellers. Prom Week [34, 35] continues on the work on Façade[36]. It concerns an interactive social puzzle game where 18 characters have different personalities and relations amongst each other. The user plays as one of the characters and interacts with the other characters. The users' choices are limited to a number of possible interactions with every other character. With over 3500 social rules the storyteller is capable of generating on the fly narratives that are dependent on the users' choices in the game. FeatNot! [14, 37] is a framework that uses emotional virtual agents to train children to cope with bullying. The agents choose their goals based on the intensity of the emotions they feel in combination with the importance of their goals. The work in [38] tries to identify how affective characters in storytelling applications should be configured but they focus on single characters instead of drama management with author-defined goals for the plot. This character centered approach is also seen in the way the virtual storyteller uses emotional modeling [39, 40]. Emotions do not directly guide the storyline but on a lower level characters can get emotions that influence the importance of certain goals. Anger could for example cause a character to be more eager to fight. The characters can feel hope-fear, joy-distress or pride-shame directed towards themselves and they can feel admiration-reproach, hope-fear, and love-hate towards other actors [41].

2.2.1. THE AFFECTIVE STORYTELLER

Numerous storytellers have considered using emotions to improve the narrative generation process. Some aim to use the users' emotions to manage the plot tension, others add emotional calculations to their characters. This later approach is similar to what the Affective Storyteller does. Characters have emotional responses to the events and this influences the eventually generated narratives. What makes the Affective Storyteller unique is how deeply these character emotions are related to the plot development of the story. Instead of using predefined plot structures a story should follow, the Affective Storyteller allows to define the desired plot development at runtime by using the character emotions as heuristics. Meaning, the user can define emotional flows the characters should follow and the storyteller forces the story generation to reject stories that do not comply with this emotional flow for the character.

When generating a story then scripted stories with little diversion are faster to generate. A planning mechanism increases the replay value and makes it possible to customize the presented stories to the user [12]. This second approach is however less efficient which makes it harder to generate longer stories. The Affective Storyteller uses heuristics obtained from data about the character emotions, to prevent the calculation time and space to grow exponentially when the amount of possible events in the story domain increases.

3

CHALLENGES IN NARRATIVE GENERATION

Narrative generation is a vast field with many challenges. Some were already discussed in section 1.2. How can we for example manage challenges like: *suspense*, *conflict*, or story *realism*? When we look at these storytelling concepts we can see that they are often in some sense linked to how characters are affected by the stories' events. Suspense, for example, is linked to the characters' diminishing likelihood that he achieves his goals and conflict is linked to characters causing events that are detrimental to other characters' goals. If we would have some abstract representation of stories, based on how these characters perform in regard to their goals, then we would also be one step closer in addressing these challenges.

For this research we propose to use character emotions as basis for these abstract representations. The OCC model [15] is an emotion model that determines affective responses to events based on how they affect the agents' goals. It becomes too much for this thesis to address all the before-mentioned challenges individually. Instead we aim to use this emotional data to generate abstract data representations of the stories and use filtering techniques to filter stories that do not adhere to certain predefined structures. It is not the goal of this research to find the exact structures that correspond to concepts like suspense or conflict, but rather to enable generating abstract structures, based on affective data, which stories should follow. When we have this type of control over the story generation process, then we can use this not only to gain control over the *customization* of the stories, but also to improve the *efficiency* of the generation process. We will now expand further on these concepts.

3.1. EFFICIENCY

When generating a narrative with a planning approach as explained in section 1.3 then it quickly becomes a costly operation in terms of computation time. A story is a highly complex and difficult intellectual product, the creative procedure of creating human literature may very well be beyond the computational possibilities of current computers [7, 42]. It requires reasoning about character goals, feelings about time and space, knowledge of natural language and even on how all this needs to be combined to invoke certain feelings in the audience [7].

A story-domain consists of actors and events. A computer based storyteller can generate many different stories given such a domain. A story is formed by a sequence of events. As the amount of possible events in such a sequence grows, the computation time for generating all possible story sequences in the story-domain grows exponentially with it. For efficiency, when the user of a storytelling program requires a story, then it is not desirable if that calculation takes hours. When the number of possible events is too large then this reduces the ability to generate long narratives [12]. Heuristics are desirable that can reject story paths before completely calculating them, which would save a significant amount of time in longer stories.

3.2. CUSTOMIZATION

In order to cope with the different preferences between audiences, we want to be able to adjust the stories presented by the storyteller. We want to steer the story generation process to only present stories that follow certain structures. So the challenge is how to operationalize concepts of storytelling, like *humor* or *realism*, into formal models usable in the generation process [43]. The abstract representation as mentioned above, can help to achieve this. Some audiences may find concepts like humor and joy more important in stories,

others may prefer drama and realism. According to [44], 'A text is considered appropriate when its content and style satisfy the constraints of the task, in particular the characteristics of the audience and the purpose of the text.' Narrative generation can be customized both from a readers' as well as from an authors' perspective. It may for example be that the author intends to write an exciting story, which is not necessarily the desire of the audience. When a story domain provides thousands of fabula, all with slightly different characteristics, then powerful algorithms to filter stories based on certain heuristics can help finding the desired stories. This should then enhance the reading experience, and/or better target a particular audience.

3.3. USING AFFECTIVE PATTERNS

A stories' fabula consists of a sequence of events. The Affective Storyteller uses a panning based construction of such a fabula. The story is viewed as a planning problem in where the initial state can be transformed into a goal state through the execution of different events. Many of these planed event sequences will however prove to result in undesirable stories. If the user could define constraints on how the storyline should develop, then it would be possible to break the calculation of some event sequences before completely finishing the calculation. Emotions of the characters could aid in this. For example, when the user wants a 'damsel in distress' type of story then he wants the damsel to first experience *distress* and later in the fabula this should become *joy* and *gratitude*. The storyteller can prune story-lines that do not follow the emotion path [distress → Joy/ grateful] for said damsel.

Customizing a story can be achieved in a similar manner. If the reader prefers stories with a happy ending, then the emotion path of the hero should end with joy. Having these personalized stories should already increase the reading experience of that single user, but it might even be possible to go one step further using the gathered data of many users of storytelling software (e.g., when embedded in a published game). This data driven approach can be used to learn what emotional paths are perceived as enjoyable by which audiences, teaching designers (and software) to make successful stories for a particular target audience.

To find the stories with these affective patterns we need affective filtering techniques to analyze them. In section 4 we show how the Affective storyteller enables affective filtering. For this the storyteller creates some communication between the storyteller and the emotional framework GAMYGDALA. GAMYGDALA calculates proper character emotions for the event sequences, the Affective Storyteller then stores the affective information received from GAMYGDALA in proper ways for the filtering techniques. In section 5 we will discuss different filtering techniques over this data with their pros and cons. In section 6 we will discuss our findings and results while testing one of these techniques on their potential for efficiency and customization.

4

THE AFFECTIVE STORYTELLER

In this research we aim to address the problems named in section 3 by using characters' simulated emotions. For this we needed a framework that was capable of generating a narrative with the proper affective information. Generating a narrative starts with having a domain which consists of characters and possible events out of which fabulas can be generated. In the Affective Storyteller events are actions performed by actors. Often these actors are the story characters, but they could also be abstract actors like 'mother nature' causing an earthquake. Every possible event sequence is calculated. Characters perform with different success rates regarding their GOALS in the different sequences, providing a large set of fabulas with different affective patterns. Finally these event sequences need to be transformed into a human readable text.

We have developed the *Affective Storyteller*. This is a domain independent system that goes through these standard steps of narrative generation, but in addition uses emotion generation for the story characters based on cognitive appraisal, both in the fabula planning and the story-text conversion. The Affective storyteller aims to show how emotion can be used to improve narrative generation on the previously mentioned challenges. It uses GAMYGDALA [28] to generate synthetic emotions for the characters in the story. We used GAMYGDALA for the following reasons. GAMYGDALA is a light-weight appraisal engine. It has been developed to be Game-AI independent. One of the intended applications is the story-oriented platform of role-playing-games [28]. Further, GAMYGDALA can be used as long as the using application has events that are identifiable by the user, and it is possible to define actors with goals that relate to these events. As generated narrative in modern storytellers is based on actors that live through events and have goals, and as we wanted to be free with respect to the use of our story planner, we opted for GAMYGDALA. This means that we had to link GAMYGDALA and the Affective storyteller which posed some difficulties and forced us to make necessary decisions in design. In this chapter we will discuss the Affective storyteller based on the three main steps of story generation (Story domain → Fabula → Presentation) and how GAMYGDALA is incorporated in those processes.

The architecture of the narrative generation process in the Affective storyteller and how this is linked to the emotion engine GAMYGDALA can be seen in figure 4.1. On the left side of the dotted line this image shows the Affective Storyteller framework. On the right side of this line is the GAMYGDALA emotion framework [28], which is responsible for calculating the emotional states for the characters. The image on the left side shows how a narrative is generated. It is the presentation of some fabula which can be generated out of a story-domain. In this section we will explain how the Affective Storyteller performs these steps and how it is linked to the GAMYGDALA framework. A complete overview of the functionality of the Affective storyteller can be found in the frameworks' manual at appendix A.

4.1. GAMYGDALA EMOTION ENGINE

To understand how the connection between the GAMYGDALA framework [28] and the Affective storyteller can be defined, it is first required to understand how GAMYGDALA beliefs and goals work. GAMYGDALA is a psychologically grounded emotion engine based on the OCC model [15]. It enables to easily add emotions to synthetic characters. In GAMYGDALA you can define different characters each with their own set of goals. Every goal gets a utility value; a value indicating how important this goal is for the character, and a likelihood that indicates how probable it is, according to this character, that he will achieve this goal. The changes

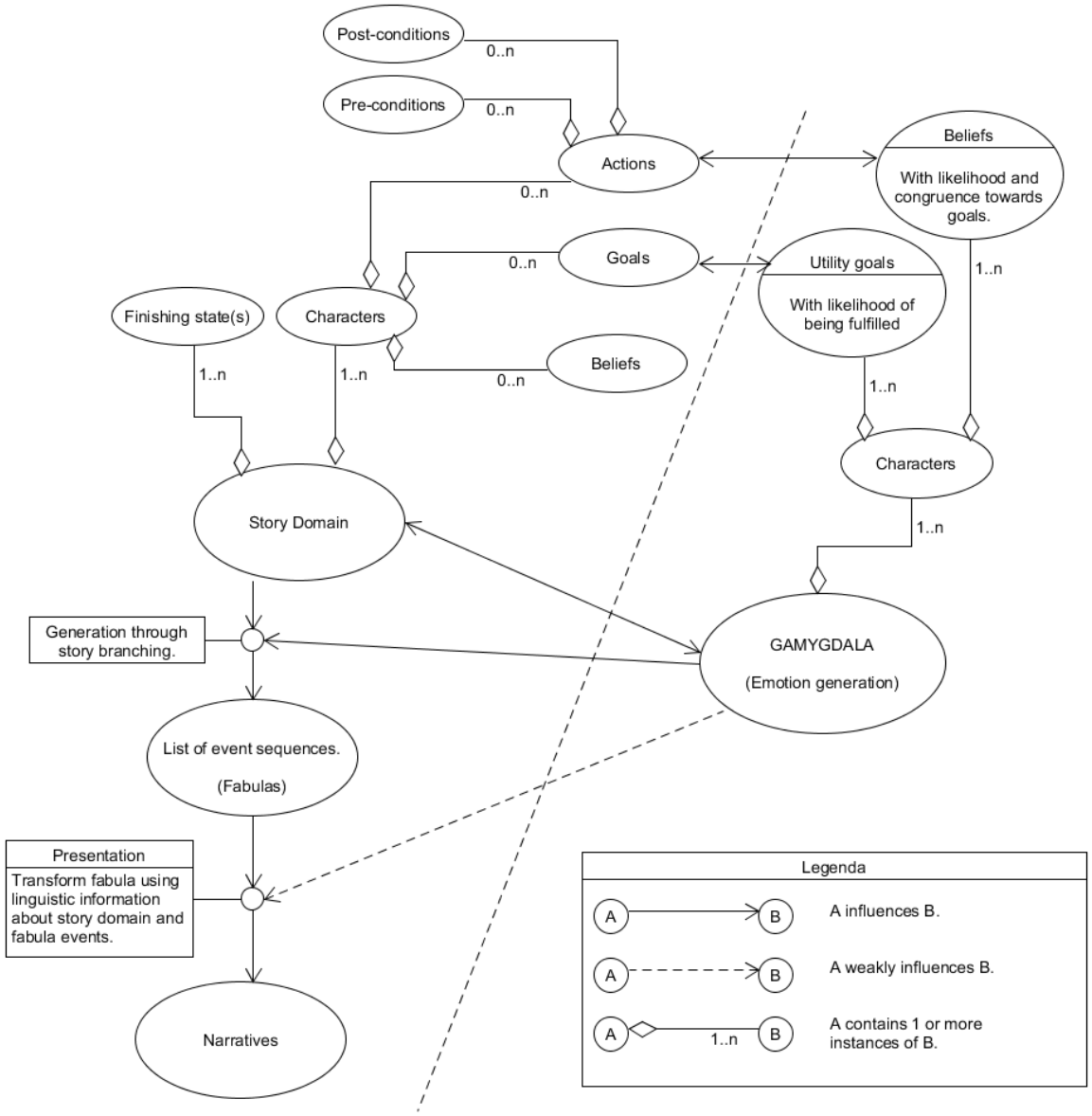

Figure 4.1: This image shows the architecture of the narrative generation process in the Affective storyteller and how this is linked to the emotion engine GAMYGDALA.

in likelihood trigger emotions in the characters. When it for example becomes more likely that a character achieves some goal, then he will experience *hope*.

The changes in goal likelihoods are calculated through events. Whenever an event takes place, this should be communicated to GAMYGDALA. An event influences the goal likelihoods. GAMYGDALA then calculates the proper emotions following from this event. GAMYGDALA contains no reasoning concerning *how* an event should update the likelihoods of character goals. It requires this information from an external source. It defines the concept of *congruence* for their events. The congruence of an event is a value indicating in what extent an event opposes or contributes to the achievement of a goal. This value can be between -1 and 1. -1 indicates that an event completely blocks the goal, 1 indicates that it completely achieves the goal. An event has a congruence value towards every relevant goal. Furthermore, an event has a responsible agent. An agent that causes the event. For a responsible agent of the event, a congruence value towards some goal, and an owner of the goal, GAMYGADALA calculates the proper emotions for the responsible agent and the owner of the goal. The responsible agent and the owner of the goal can be the same agent. In that case GAMYGDALA

refers to internal emotions, otherwise GAMYGDALA refers to social emotions. A complete overview of the 16 emotions from the OCC-model that GAMYGDALA can generate can be seen in table 4.1. The table further shows the exact trigger functions. The first 8 emotions are internal, the 8 emotions after that are social. For social emotions the relation between the responsible agent and the owner of the goal is relevant. Causing an undesirable event for a liked agent triggers another emotion than causing such an event for a disliked agent. Finally, the emotions in GAMYGDALA are triggered with some intensity. The intensity of an emotion is calculated with $Intensity = |U(g)| * |\Delta L(g)|$. U(g) is the utility value of a goal. $|\Delta L(g)|$ is the delta likelihood; the change in likelihood before and after the event. This means that emotions will have a higher intensity if the agents finds the goal to be more important or if the influence of the event has a higher impact on the likelihood of the goal.

Emotion name	Trigger	Description
Joy	$(U(g) > 0 \wedge L(g) == 1) \vee$ $(U(g) < 0 \wedge L(g) == 0)$	A desirable goal succeeds or an undesirable goal fails.
Distress	$(U(g) < 0 \wedge L(g) == 1) \vee$ $(U(g) > 0 \wedge L(g) == 0)$	An undesirable goal succeeds or a desirable goal fails.
Hope	$(U(g) > 0 \wedge L(g) < 1 \wedge \Delta L(g) > 0) \vee$ $(U(g) < 0 \wedge L(g) > 0 \wedge \Delta L(g) < 0)$	A desirable uncertain goal increases in likelihood of success or an undesirable, uncertain goal decreases in likelihood of success.
Fear	$(U(g) > 0 \wedge L(g) > 0 \wedge \Delta L(g) < 0) \vee$ $(U(g) < 0 \wedge L(g) < 1 \wedge \Delta L(g) > 0)$	An undesirable, uncertain goal increases in likelihood of success or a desirable, uncertain goal decreases in likelihood of success.
Satisfaction	$U(g) > 0 \wedge L(g) == 1 \wedge \lvert\Delta L(g)\rvert < 0.5$	A desirable goal was expected to succeed and it actually does succeed.
Fears confirmed	$U(g) < 0 \wedge L(g) == 1 \wedge \lvert\Delta L(g)\rvert < 0.5$	An undesirable goal was expected to succeed and it actually does succeed.
Relief	$U(g) < 0 \wedge L(g) == 0 \wedge \lvert\Delta L(g)\rvert > 0.5$	An undesirable goal was expected to succeed and it fails.
Disappointment	$U(g) > 0 \wedge L(g) == 0 \wedge \lvert\Delta L(g)\rvert > 0.5$	A desirable goal was expected to succeed and it fails.
Anger	$des(e,g,self) < 0 \wedge Agent(b) = !self$	An undesirable event is caused by another agent.
Gratitude	$des(e,g,self) > 0 \wedge Agent(b) = !self$	A desirable event is caused by another agent.
Guilt	$des(e,g,other) < 0 \wedge$ $Responsible_Agent(e) == self \wedge$ $like(self,other) > 0$	The agent causes an undesirable event for a liked agent.
Gratification	$des(e,g,other) > 0 \wedge$ $Responsible_Agent(b) == self \wedge$ $like(self,other) > 0$	The agent causes a desirable event for a liked agent.
Happy for	$des(e,g,other) > 0 \wedge$ $like(self,other) > 0$	A desirable event happens to a liked agent.
Pity	$des(e,g,other) < 0 \wedge$ $like(self,other) > 0$	An undesirable event happens to a liked agent.
Gloating	$des(e,g,other) < 0 \wedge$ $like(self,other) < 0$	An undesirable event happens to a disliked agent.
Resentment	$des(e,g,other) > 0 \wedge$ $like(self,other) < 0$	A desirable event happens to a disliked agent.

Table 4.1: In this table the different emotions generated by GAMYGDALA are described including the exact state that an agent should be in to experience them. U(g) stands for the utility of a goal. L(g) for the goal likelihood. $|\Delta L(g)|$ is the delta likelihood; the change in likelihood before and after the event. The reader does not need to understand more about des(b, g, ?agent) than: 'the event (e) influences goal (g) owned by agent (self or other) in a desirable (value above 0) or undesirable (value below 0) way. Responsible_Agent(e) is the agent causing event 'e' and finally like(?agent, ?agent) shows a relation between two agents.

An example of an event causing emotions according table 4.1 could be a dog barking to his owner. The dog may have the goal to 'go for a walk in the park'. Barking increases the chance of achieving this goal. The agent causing the event and having the goal are the same, therefore this is an internal emotion. The goal has

a utility above 0. (A utility below zero would mean the 'goal' is something to prevent.) The likelihood of the goal increases, meaning the dog will feel *hope*. The owner however had the goal to 'stay inside'. He finds the couch comfortable and he doesn't like the weather. The dog barking, however, means he may have to give up on these goals in order to prevent the dog from urinating inside the house. This means the responsible agent (the dog) causes an event (barking) that decreases the likelihood of the owner maintaining his goals. This will cause the owner to become *angry*. Finally, if the dog likes his owner, and if the dog is aware of his owners' goals then this causes him to feel *Pity* and *Guilt*. We invite the reader to inspect what emotion would be triggered if the dog disliked his owner.

4.2. STORY DOMAIN

The first step in story generation is defining some story domain. A domain consists of characters, events and some final state(s) for all characters that defines when the story is completed. The Affective storyteller can handle any domain that can be defined with these building blocks, but provides little guidance in how such a domain should be built. The manual in appendix A talks more about how domains can be constructed with help of the story editor. This step, however, is still mostly the task of some human author in the Affective Storyteller. The human author needs to define agents that behave like characters. For this the agents are defined with statements concerning the world, which they believe to be true. These are referred to as 'beliefs'. The agent further has goals which are statements that the agent wants to be true. Finally the agent needs to have a set of actions it can perform. An action has pre-conditions which are beliefs that need to be true before the action can execute and post-conditions which are adjustments of beliefs for one or more agents in the domain. All these statements are strict. Beliefs and action conditions can either be true or false. Goals can be achieved or not. When looking at figure 4.1 we can see that actions and goals are also linked to the GAMYGDALA emotion framework.

Actions are linked to GAMYGDALA beliefs with a likelihood and congruence values towards goals and goals are linked to GAMYGDALA goals with likelihood of being/ becoming true. There are some design choices here in how these frameworks ought to be connected. In the paper of GAMYGDALA [28] beliefs are referring to events that influence the characters' goals. It is therefore a natural choice to link them to the character actions. This is however not a one-to-one connection since the belief still has some likelihood and some congruence values. Furthermore a character action in the Affective storyteller is defined for one character but can influence the goals of many characters. The GAMYGDALA event therefore should be linked to all characters and have congruence values towards the goals of all characters. GAMYGDALA defines a congruence to be a value between -1 and 1, where -1 means the event completely blocks the goal and 1 means the event fulfils the goal. Before explaining how the Affective storyteller defines the proper congruence for events we discuss how story domain goals are linked to the GAMYGDALA goals.
This may become a bit clearer with the previous example of the dog barking. When the agent 'dog' performs the action 'bark' in the story domain, then GAMYGDALA is informed by letting all agents believe 'dog barked'. Looking at figure 4.1 we see that the GAMYGDALA belief has a likelihood and congruence values. In the Affective Storyteller we are always certain an event happened, therefore all characters believe this event happened with likelihood 1. This event influences different goals. In the example it influenced the dogs' goal of going for a walk through the park, and it influenced the owners' goal of 'staying inside'. The Affective Storyteller calculates in what extent the goal likelihoods are influenced. This was informed to GAMYGDALA with the before mentioned congruence values. But, how should we define how likely it now is that the dog achieves 'go for a walk in the park', or the owner achieves 'stay inside'?

Calculating GAMYGDALA goal likelihoods Character goals in the Affective Storyteller story domain and character goals in GAMYGDALA both have a utility that defines their importance for the character. However, character goals in the story domain, as mentioned above, are simply achieved or not. Character goals in GAMYGDALA have a likelihood of being achieved between 0 and 1 and it is this likelihood and the changes in it that determine the character emotions. Linking GAMYGDALA's definition for goals to the strict character goals in the story domain by setting the value in GAMYGDALA to be 0 for not achieved and 1 for achieved would be too simple and causes GAMYGDALA to be incapable of finding some emotions. *Hope*, for example, is only triggered when a goal becomes more likely, but is not yet certain. This means that there is need for some artificial intelligence that computes proper likelihoods for character goals in GAMYGDALA to be achieved. These likelihoods do not have to be exact. It is not needed to know the true likelihood of for exam-

ple 'going for a walk in the park'. It is needed to have a likelihood that gives a proper estimate of the likelihood given the characters current perception of the world state.

A possible tactic could be to calculate the amount of steps required to achieve the characters' goal state and the amount of intermediate actions other characters could perform that would block the goal. This however assumes that the character can actually oversee all this information and that every event has an equal chance of becoming reality. Another tactic is to require the story domains' human author to define another layer of information for the story. A layer of sub-goals that are required to become true for the complete goal to become true and optionally a congruence of these goals towards their 'parent'. The main disadvantage of this approach is that it requires yet another design step that the human author needs to perform. The main advantage is that it yields no assumptions and therefore maintains the domain in-dependency of the Affective storyteller. Since the Affective Storyteller is designed to be domain independent, we have chosen for this second approach.

With a mechanism that calculates proper goal likelihoods for GAMYGDALA, it is only needed to communicate this to GAMYGDALA. The GAMYGDALA way of doing this is through the congruence values of the event. GAMYGDALA can then calculate emotional data for all characters, for every event.

4.3. GENERATING THE FABULA

With the emotional model for the story characters in place, it is now possible to perform actions and retrieve proper affective data. The Affective storyteller calculates possible event sequences with a forward chaining planning mechanism. Given all the possible actions (for every character) in a story-domain the storyteller sees if it would be possible to perform the action. Then for every possible performed action, the narrative generation algorithm again checks what follow-up actions would be possible. The storyteller continues this process until one of the finishing states in reached. In this way a tree of stories is generated.

After every performed action it can be inspected what the character emotions are. The Affective storyteller informs GAMYGDALA about the performed event and what congruence values this event has towards the character goals. GAMYGDALA then calculates and returns emotions and emotion intensities to the Affective storyteller. Finally, the Affective storyteller stores this emotional data by saving the current intensities, intensity deltas and total intensities after every step in the event sequence. Additionally the Affective storyteller also saves a debug string representation of the story. More about this can be seen in appendix A. For now it is enough to know that these different data representations are needed by different filtering techniques which will be explained in chapter 5.

Another aspect of the emotional data generated by GAMYGDALA is the decay speed. Over time the emotion intensities of the characters will decay. The Affective storyteller allows the human author to define the desired decay speed of these intensities, which would have an effect on the textual presentation of the stories and the affective filters which are yet to be explained. A decay of 1 would mean that all emotion intensities are reduced to 0 after every story event, decay 0 means that emotion intensities are not decayed at all.

4.4. PRESENTATION OF THE FABULA

When the fabulas have been generated, the next step is to transform them into story-texts. For this transition the user can choose whether he finds it desirable to let the Affective Storyteller generate the story-text fully automatically or to maintain more control over this process. The story-text is generated automatically by using three main building blocks: the goals of the agents, the main affective responses of the characters during the story, and finally the story events being the actions performed by the different agents. The first building block, the character goals, can be used to automatically generate an introduction to the story. Let us give an example of such a generated introduction. Consider a story domain with two agents. A 'owner' and a 'dog'. The owner has one goal with a high utility: 'Clean house'. He has two less important goals: 'Inside' and 'Relaxed'. The dog has one important goal: 'Needs to urinate'. He has one less important goal: 'Play outside'. The importance is defined with the utility values the author has given the goals. A utility above 0.7 is important. This value is chosen arbitrary and may need to change in other domains. The text is built in the following structure:

> *For every agent do: ?Agent_A found it important to ?Goal_A, ?Goal_B, .., and ?Goal_C. He further liked to ?Goal_D, ?Goal_E, .., and ?Goal_F. –current achievement of goals– . That made him feel –emotional state–.*

In the example domain this becomes:

> *The owner found it important to have a clean house. He further liked to be warm and comfortable inside the house and to be relaxed. He had achieved all of his goals. That made him feel happy and satisfied. The dog found it important to urinate. He further liked to play outside. He was yet to achieve his goals, which made him feel distressed.*

In this example it can be seen that what is presented in the intro are (a) for each actor the goals it finds important, (b) the extent to which these goals are met or not, and (c) the initial affective response towards this situation. Every goal and action have a story-text connected to them. So the goal 'Inside' that the owner had provides the story-text: 'be warm and comfortable inside the house'. The Affective Storyteller helps the user to define these story-texts (see Appendix: A, figure A.3).

After the introduction the agents start performing actions. After every action the characters experience some emotional changes as simulated by GAMYGDALA. If the user chooses to let the Affective Storyteller perform this behavior then after every action the most salient affective changes (highest intensity change) in the characters are verbalized. For example, when the owner takes the dog for a walk in the park, it is only mentioned that the dog is happy and grateful. The fact that the dog still needs to urinate and that he is still distressed about this will not be mentioned. This is an intuitive and informed choice, but obviously further research is needed on how to verbalize emotional states of characters in stories.

The final and most important building block for the story-text transition process is the story-text connected to the different events in the story. (So the event 'dog barked' can have the story-text: 'The dog barked loudly. ') This text can again be generated automatically. This however only rarely gives desired texts, therefore the Affective Storyteller also provides the possibility to define logically constructed event texts. Meaning the story-text connected to an event can be different dependent on the outcome of some logic statements concerning character believes and emotions, the storyteller constructs the proper event-text. The structure of such an event-text can be seen below:

> The logic statement was: ?[<?agent_name»{ *some logic statement of beliefs and emotions* }]
> (true.)
> (false.) I'm sure of it!

The string ?[<Wolf>{ at #Grandma's house && (Ate #Grandma || !Hungry)} && <Red>{at #Grandma's house && €Distress3}] is true when the Wolf and Red are both at grandma's house, the wolf has either eaten grandma or is not hungry and little red riding hood is distressed about something. So when this statement is *true* when the action the story-text belongs to is performed, then the Affective Storyteller would print: 'This logic statement was true. I'm sure of it!'. Logic statements can be nested and can contain logic constructors like ||, &&, ! and brackets. A logic statement is built with agent beliefs and emotions.

With this technique it is possible to maintain a lot of control over the produced story-text allowing the Affective Storyteller to write more interesting stories. An example of a story produced from the little red riding hood domain, using this technique, can be seen in Appendix D. Important to understand for this technique is that it is not intended to try to have a different story-event text for every possible event sequence. Even though this is theoretically possible, defining the story-text for over 10.000 possible cases would be far from desirable. Rather than doing this it is simply intended to write logic sentences with minor differences, dependent on what may have preceded the event.

5

AFFECTIVE STORY CUSTOMIZATION

In this section we discuss different methods that filter and sort stories based on affective patterns.

5.1. SORTING STORIES WITH EMOTION VALUING

The narrative generation problem in the Affective storyteller starts with the user defining a domain or choosing one from the example library. The Affective storyteller then calculates all possible event sequences that lead to the defined finishing state of the story. These event sequences are all somewhat different in storyline. In particular, the emotions generated by the characters are different in every event sequence. The problem we address here is how to find the subset of event sequences that fit the individual user preferences.

In every event sequence the emotions for the different characters are measured and logged. GAMYGDALA calculates different emotions and emotion intensities. These emotion measurements can then be used to sort the stories. The user is presented an emotion panel 5.1 that shows all *occurring* emotions in the characters of the story. When a user for example increases the importance for ´Joy´, then the Affective storyteller will search for stories in where the character experiences more joy. Currently, a user can choose these values for the first half and the second half separately.

GAMYGDALA generates 16 different emotions. The Affective storyteller calculates the total sum for each of the emotion intensities in both the first and second half of the story being a total of 32 measurements for each event sequence. The user chooses for every emotion, for both the first and second half of the story, how important this occurrence is to him on a scale from 0 to 1. This again results in a total of 32 numbers. We use the dot product of these two vectors as a value function for the different event sequences:

$$Value(Seq) = \sum_{i=1}^{32} \left\{ EI_i * EC_i \right\}, \tag{5.1}$$

where EI is the emotion intensity, and EC is the emotion congruency as defined by the user. A problem with this function is that some emotions may have such strong intensities that they push away the contribution of other emotions in the outcome of equation 5.1. A way of preventing this is to dampen the individual emotion intensity measurements to a value between 0 and 1:

$$Value(Seq) = \sum_{i=1}^{32} \left\{ \left(\frac{EI_i * gain}{EI_i * gain + 1} \right) * EC_i \right\}. \tag{5.2}$$

where, Gain is a constant set by the user defining how aggressive the intensity dampening is. Additionally. users can choose which function fits them best.

The above discussed algorithm is a straightforward method for affective customization of a story. This method was however lacking functionality. By bluntly dividing the story in two halves, the user can only define emotional flows that follow this time-frame. More enhanced heuristics are however required, allowing the user to define more subtle emotional structures for the story. We implemented two more advanced methods for affective customization and will discuss these in the remainder of this section.

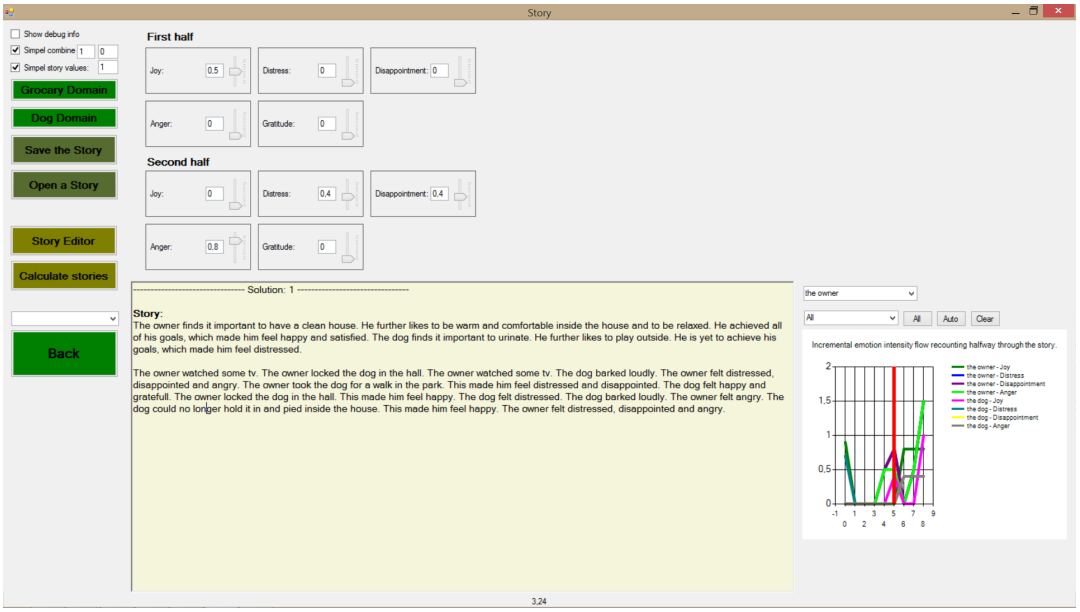

Figure 5.1: Screen shot from the Affective Storyteller tool. Top showing the sorting panel in which the user can select which emotions are relevant for which part of the story (only showing actually occurring emotions). Right showing the emotion intensity in the story for the first and second part. Bottom showing the currently selected story.

5.2. EMOTION HEURISTICS

In this first advanced method we mainly aim to solve the time-frame problem as mentioned before. In this method the user defines multiple emotion meta-data blocks that we will refer to as heuristics. Every heuristic mentions an emotion that should be experienced by one or more actors with a certain intensity and at a certain relative moment in the story. The emotion script has an XML-like format as in [45] An example of such a heuristic can be seen below:

Emotion ID = start_story_hopeForAnyone
{
Category = Hope
Intensity_min = 0.3
Intensity_max = 1
ExperiencedBy = Any
Order = 0
Freedom = 3
}

The user can define as many heuristics as desired. We will explain the different elements in these heuristics one at a time. The first four elements between the curly brackets define some emotional state that should be experienced by one or more agents. The elements 'order' and 'freedom' define a relative position in story time stating when in the story this emotional state should be apparent. For a complete explanation of all elements in these heuristics we refer to Appendix A, section A.8.1. Here we will only explain the main elements.

Category & intensity
The category should be filled in with one of the 16 emotions produced by GAMYGDALA (see Appendix B). This is the emotion that the user desires to be present in the story. The minimum and maximum intensities define in what amount the emotion is allowed to be experienced. Emotion intensities are always between 0 and 1.

ExperiencedBy
The agent or agents that should experience the emotion. This field can be filled in with some agent name (for example: 'the owner') or it can contain one of the quantifiers 'Any' or 'All'. When it contains 'Any' as in the

example, then the algorithm will aim to have one of the agents feel this emotion. 'All' will force the algorithm to have all agents experience the emotion.

Order
Order defines a relative position in the story. When there are many different heuristics defined then the algorithm sorts these based on their order number. It will then first aim to accomplish every heuristic of the lowest order number before it continues to the next number. The example below shows three metadata-blocks.

Emotion ID = start_story_hopeForAnyone
{
Category = Hope
Intensity_min = 0.3
Intensity_max = 1
ExperiencedBy = Any
Order = 0
Freedom = 3
}
Emotion ID = 2_end_story_resentmentForAnyone
{
Category = Resentment
Intensity_Min = 0.1
ExperiencedBy = Any
Order = 1
Freedom = 3
}
Emotion ID = 3_start_story_angerForAll
{
Category = Anger
Intensity_Min = 0.2
Intensity_Max = 1
ExperiencedBy = All
Order = 0
Freedom = 5
}

In this example the algorithm would first aim to fulfill the emotions *1_start_story_hopeForAnyone* and *3_start_story_angerForAll*. It starts performing the events in the fabula, one at a time. When the order 0 emotions are fulfilled the algorithm will no longer look at them and start trying to fulfill all emotions with order 1. In this case that is only the emotion *2_end_story_resentmentForAnyone*. It continues performing the events in the fabula until either the story is over or the next order number emotions are accomplished. When the story is over and there are still heuristics (metadata blocks) left to fulfill, then the algorithm returns a fail. The story does not fulfill the defined heuristics. If all the heuristics *are* fulfilled, then the story can be shown to the user.
When the order increases it is no longer needed to have the emotions with a lower temporal order (in the example hope and anger) present in the characters. The algorithm just continues performing fabula events until it either succeeds or fails.

Freedom
Freedom allows the user to define how many events are allowed to happen before a heuristic should be fulfilled. This means that in the above example the heuristic *1_start_story_hopeForAnyone* should be fulfilled after at most 3 events and stay fulfilled until all order 0 heuristics are fulfilled.

With these emotion heuristics the user has extensive possibilities in defining emotional flows for different stories. Functionally there is however still a drawback since it is not possible to define statements like agent A should feel either Joy OR Hope at the beginning of the story. These disjunctive statements could still be achieved by adding to this emotional script language but this could become a bit tedious and make the language even more difficult to read.

5.3. EMOREGEX

This last and most powerful method of story filtering is based on regular expressions. It searches a textual presentation of the story and looks if a user defined pattern can be found (to be exact it searches the story debug string, which is explained in Appendix A, section A.6). To make this possible we have developed an extension of the regular expression language itself. In this extension language a user can talk about events, agents and emotions in a more natural manner and the compiler will transform the EmoRegEx string into a normal regular expression. An example of such an EmoRegEx string can be seen below:

```
${0,3}<the owner>€Joy#{ 2*numOfAgents }$*<the dog>€Anger4
```

The compiler transforms this string into the following regular expression:

```
\$0[^\$]*(\$[^\$]*) {0,3}the\ owner\ felt[^\$€]* Joy:\ (1|(0,[4-9]))
[^\$]*(\$[^\$]*)* the\ dog\ felt[^\$€]* Anger:\ (1|(0,[4-9]))[^\$]*
```

Without any prior knowledge about regular expressions and/or emotional regular expressions, it can already be seen that the first expression looks friendlier and easier in use. In this example the string should be read as: '*after 0 to 3 actions the owner felt a little joy, then after any amount of actions the dog felt a little anger*'. Table 5.1 explains the above EmoRegEx string step by step. Some parts are part of the basic language and some parts come from the extension language. The power of EmoRegEx is that it is possible to use almost any part of the normal regular expressions. It only adds to those by giving a special meaning to some statements that talk purely about events, agents and emotions in a story.

5.4. DISCUSSION

The in this chapter discussed methods show different approaches to find fabulas that uphold certain structures. In particular the EmoRegEx language provides a powerful mechanism to customize stories. One main advantage of EmoRegEx over Emotion Heuristics are that EmoRegEx allows repetitive structures like experiencing anger every three actions. In Emotion Heuristics the amount of (temporal) order numbers are defined by the user and can not be relative to the amount of events in a story. So practically speaking, given a set of fabulas with unknown length, it is not possible to ensure having anger every three actions with only defining the Emotion Heuristics explained in section 5.2. A second main advantage of EmoRegEx over Emotion Heuristics is the possibility to define disjunctive statements in EmoRegEx. Emotion Heuristics can only construct conjunctive statements. So having a character experience joy OR Hope is not possible.

In section 1.2 we mentioned some rules for storytelling. 1, a story yields more suspense when the hero appears to lose possibilities to achieve his goal and 2, a story needs conflict, friction between the characters, to be proper. When we use these storytelling rules as inspiration then we could come up with the following EmoRegEx strings.

An EmoRegEx statement that ensures fear in the hero:

```
(${3,}<the hero>€Fear1){#{lengthOfStory \3}}
```

An EmoRegEx statement that ensures a proper amount of anger in the characters.

```
(${3,}<any>€Anger1){#{lengthOfStory \3}}
```

Proving the validity of these statements for now remains future work, but the ease of creation and simplicity of the statements already shows the potential strength of EmoRegEx filtering. These same statements are significantly more difficult to define using Emotional heuristics and even impossible when using basic sorting (section 5.1). In the next section we will use EmoRegEx filtering and validate its potential in coping with the efficiency and customization challenges in story generation (section 3).

$	This sign overrides the original meaning in RegEx. Originaly it meant 'start the search at the beginning of the string'. In EmoRegEx the search *always* starts at the beginning of the string, making this redundant. In EmoRegEx it means **'perform one action´**.
{0,3}	This is basic regular expression syntax with 'm' and 'n' being integer values. It means: **'Repeat the previous statement at least m, but no more than n times'**.
${0,3}	The above two rules explain that this should be read as: **'perform 0 to 3 actions'**.
<?agent_name>	This means that **the following emotional statements are going to apply to *?agent_name*.** *?agent_name* can be initialized with the name of an agent that is a part of the story or with one of the quantifiers 'any' or 'all'.
€?emotion	An emotional statement, asking the algorithm to **look for emotion *?emotion* in the current agent**. *?emotion* can be one of the 16 GAMYGDALA generated emotions (see appendix B).
€?emotionX	The **X** is an integer and talks about **the minimum intensity the agent should experience for this emotion**. Emotions can have an intensity from 1 to 10 (0 means the emotion is not present). It is also possible to define a maximum intensity for emotions. For this the syntax is ^X. Meaning that for example Anger^1 means that the character is not allowed to experience any anger.
#{ 2 * numOfAgents}	The text between #{ } is treated as a mathematical formula. This is calculated before transforming the EmoRegEx string into the corresponding RegEx string. There are different variables possible. For now the variables numOfAgents and lengthOfStory can be used. In future updates of EmoRegEx this can be expanded.
${0,3}<the owner> €Joy#{ 2*numOfAgents }	After 0 to 3 story-events the owner should feel Joy with a minimum intensity of 2 x the total number of storyAgents (= 4).
$*	The * is basic RegEx syntax. It means: '**Repeat the previous statement any number of times**'. Any number can also be zero.
${0,3}<the owner> €Joy4$*<the dog> €Anger4	After 0 to 3 story-events the owner should feel Joy with a minimum intensity of 4, then after any number of actions the dog should feel Anger with a minimum intensity of 4.

Table 5.1: EmoRegEx Syntax

6

VALIDATION

6.1. EFFICIENCY

The fabula generation method as explained in section 5.1 consists of two main steps. First all possible story-paths are found, for this a breadth-first search through the tree of events is used. Then they are filtered based on the similarity of their emotion pattern to the user preference. There is however a performance problem in narrative generation as explained in section 3. Here we propose to use emotional structure in stories to prune certain parts of the search space that appear to be less interesting. We show that a proof of concept heuristic based on conflict between the characters enhances efficiency. We reason that conflict between two characters means they perform actions that are undesirable for the other. GAMYGDALA generates anger for these type of actions, which means that stories with insufficient anger should be pruned.

	Initial domain	1 Idle action	2 Idle action	3 Idle action
Prune on anger	2,27 sec	5,31 sec	8,57 sec	12,25 sec
Breadthfirst without emotions	0,04 sec	0,27 sec	7,26 sec	26 min; 31 sec

Table 6.1: Time of calculation for the narrative generation. Pruning on anger uses character emotion to reduce the search space. Breadth-first searches the entire tree but saves time by skipping emotion calculations.

In table 6.1 the calculation time of the pruning algorithm is compared with that of the complete breadth first search. The algorithms are tested on an example domain where idle actions are incrementally added. An idle action is designed to change nothing in the belief bases of the agents except the fact that it occurred. The fabula planning of the Affective Storyteller can then perform the action at any point in time, but only once in a story. This greatly increases the total amount of stories that can be generated in a story domain.

Initially the breadth first search is quicker because it does not have the overhead of calculating the emotions of the characters. However, because the number of possible paths grows exponentially and this pruning heuristic decreases the exponent, the computation time of the emotionally-pruned algorithm quickly outperforms the classic breadth-first algorithm.

6.2. USER STUDY ON AFFECTIVE FILTERING

In section 5.4 the strength of the different filtering techniques is discussed. Especially the EmoRegEx technique is powerful. In section 6.1 we saw how a simple filter, forcing a high amount of conflict in stories, can already drastically improve the calculation time on larger story domains. In this section we test the customization capabilities of EmoRegEx filtering. For this we have performed a field study that tests the distinguishability of two filters. We wanted the test subjects to read joyful stories and sad stories and constructed corresponding emotional regular expressions. We developed a story domain inspired by Little Red Riding Hood. The used filters can be seen below.

The first filter states the wolf may not have experienced any distress and grandma must have experienced joy. The second filter forces the wolf to have experienced distress and prevents grandma from having experienced joy. These filters only check the emotions at the end of the story. Since we simulated the character emotions

Joy: ${#{lengthOfStory}}<Wolf>€Distressˆ1<Grandma>€Joy1

Distress: ${#{lengthOfStory}}<Wolf>€Distress1<Grandma>€Joyˆ1

without decay, these filters test if there has been any joy or distress at the end of the story. We generated 10.000 stories from the Little Red Riding Hood domain and filtered these using the above EmoRegEx strings. This gave two subsets of approximately 1.000 stories. With random draws from these subsets, the users were presented with stories and asked if they found the story joyful, or sad.

As mentioned in section 4.4 the Affective storyteller is capable of generating complicated presentations of the fabulas. The filtering mechanism however only filters stories based on the fabula information and not on these final presentations. In order to test these filters it is therefore desirable to have an minimum of influence from the presentation and purely test the impact the actual fabula has on the audience. The Affective Storyteller was therefore tuned to only present the test subjects abstract, objective sentences verbalizing the different events that form the fabula. For example: 'The wolf knocked on grandma's door' or 'It became night'. In Appendix D an example of a fabula and complete script can be found.

One example story for the first filter can be seen below:

Grandma went hunting

It became night

Grandma went back home

It became day

The wolf knocked on grandma's door. Thinking it was Red, grandma let the wolf in

Red travelled the road, to go and visit her grandma

The wolf drank tea at grandma's house

The end.

One example for the sad filter is:

It became night

Red travelled the road, to go and visit her grandma

It became day

Grandma went hunting

Grandma saw some wolf cubs trapped in a river current. The mother wolf was nearby, but did not see the danger for her cubs.

Grandma killed the wolf cubs. The mother wolf was nearby and saw what happened.

The wolf ate grandma in forest

The wolf went home

The test subjects were all presented with four stories stories for each filter, making a total of eight stories. For every story they were asked to validate it on a 7-point Likert scale where 1 meant a very sad (distressed) story, 4 meant a neutral story, and 7 meant a joyful story. To amend for bias from order-, learning- and fatigue effect from previous measurements, the order of the conditions was randomized among the participants. Figure 6.1 shows boxplots of the data retrieved from this test.

We used a repeated measures 2x4 ANOVA with the two affect filters as one factor and the numbered measure (for a certain filter) as another factor. The subjects for this test had to have a good understanding of the English language. The chosen subjects are approximately between the ages of 20 and 60. For 23 test subjects, this test showed a significant difference on affect between the two filters Joy (M=5.22, SD=0.22) and Distress (M=2.21, SD=0.20), Wilks' Lambda = 0.21, $F(1,22) = 85.05$, $p < 0.001$. The effect of measures and the interaction effect were not significant.

The ANOVA test showed that there was a significant difference in perceived affective content of the stories for the different filters. The means of the two filters (joy, M = 5.22; distress, M = 2.21) also show that the

average rating of affective content corresponds with the used filter. This implies that the EmoRegEx filtering technique has successfully found stories that were perceived as being joyful or sad, fulfilling our original intention with the filters.

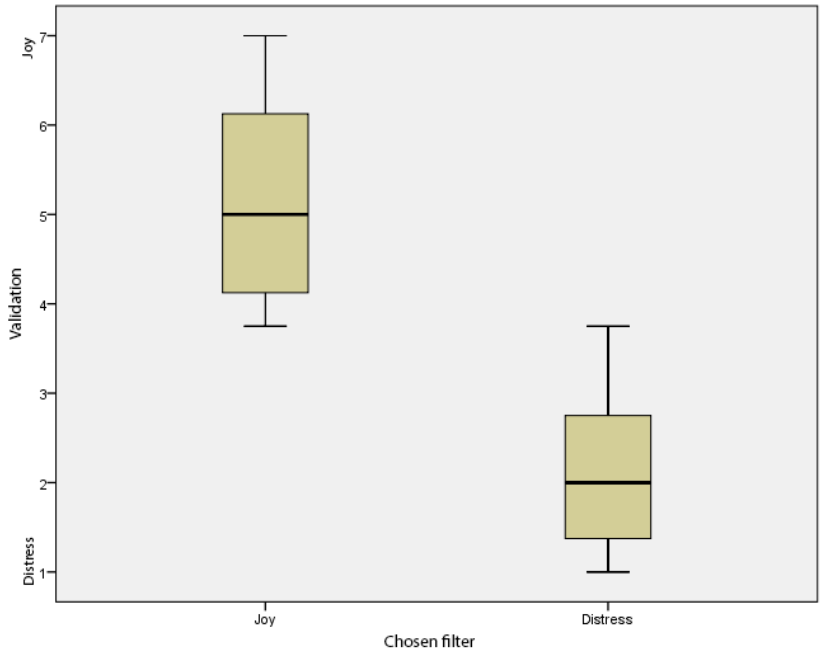

Figure 6.1: A boxplot showing the distribution of the data for the two different filters.

7

DISCUSSION & FUTURE WORK

Two challenges in narrative generation were introduced. *Customization* to fit individual user needs and *efficiency* to cope with exponential growth in computation time as the size of the story domain increases. We have proposed a framework that combines affective modeling of the individual characters with the generation of story-lines in order to get closer to addressing these challenges. We have shown that these stories can be personalized with emotion patterns allowing the user to request different type of stories for a given domain.

For example, the pattern (${3,}<any>€Anger1){#{lengthOfStory \3}} means that throughout the story characters should experience anger towards one another. This emotional filtering allows the user to customize stories. We have tested the validity of this filtering method and found that the method strongly influences the perceived affective content of the stories. Having found this, the next step is to find filters that improve stories on some of the, in this paper mentioned, quality measures. Making good stories is particularly difficult because good is a subjective measure. As mentioned, there is no consensus on how to measure the quality of a story [7]. It is however very possible that certain emotion flows will almost never result in good stories and vice versa. Jan Veldman states [11] that there is no proper story without conflict between the characters. He however does not define what 'conflict between characters' exactly means. One possible definition is that characters are responsible for events that are detrimental to another characters' goals. A more strict definition could be that characters have opposing goals. In the first definition we can simply look for emotions like anger and guilt. The second definition is more difficult. An action should simultaneously inflict hope or joy in one character and fear or distress in the other character, but even then it is not certain that the goals are opposing since the emotions could be inflicted due to different effects of the action. An interesting future project is to investigate what emotional patterns are linked to this notion of conflict and under which prerequisites.

Another example where story structure influences user emotions can be found in the work of Gerrig and Bernardo [8]. They have shown that by diminishing the certainty or number of paths through the hero's problem space the readers feel more suspense in a story. It is important to mention that this concerns paths known to the audience. When thinking back of the difference between fabula and presentation as explained in section 1.2, this means that diminishing the possible paths in the ordered sequence of events (fabula) will only increase the experienced suspense when it is also mentioned in the presentation of the story. As GAMYGDALA is based on the OCC model [15, 28] a decrease versus an increase in the certainty or number of paths towards the characters' goals is connected to the amount of fear versus hope of the character. It may be too simple to state that stories with fearful characters are therefore stories with more suspense but it does suggest some relation between the emotions of the characters and the experienced suspense of the audience.

A good and interesting topic for future work would be to test certain affective filters and find if they indeed improve stories on some quality measures. We proposed two EmoRegEx filters in section 5.4. These filters could be a good start where the questions to the readers could concern the perceived amount of conflict in the stories and the perceived amount of suspense in stories. Finding filters that improve stories in these and other directions would be a powerful tool in plot management.

Future work can further consist of improving the story-pruning method. Not all EmoRegEx filters can be used for filtering during generation, and there might be desire for some mechanism that does not completely

reject stories that do not comply with the presented filter(s). A possibility would be to have a weighted list of EmoRegEx filters during generation. Every branch in the generation tree then scores points based on the weight of the filters and whether the branch (so far) fulfills the filters. At every computation step the most promising branch is expanded.

This is of course only one of the many possible ways of doing this. The future research should therefore inspect multiple methods and choose the most powerful mechanism.

Although there is much more work to be done on the exploration of emotion patterns in stories, affective customization of stories, presentation of the story and especially the relation of these aspects to perceived quality measures of stories, we conclude based on our current work that emotion simulation is a promising direction to address planning complexity and customization.

A

AFFECTIVE STORYTELLER; MANUAL

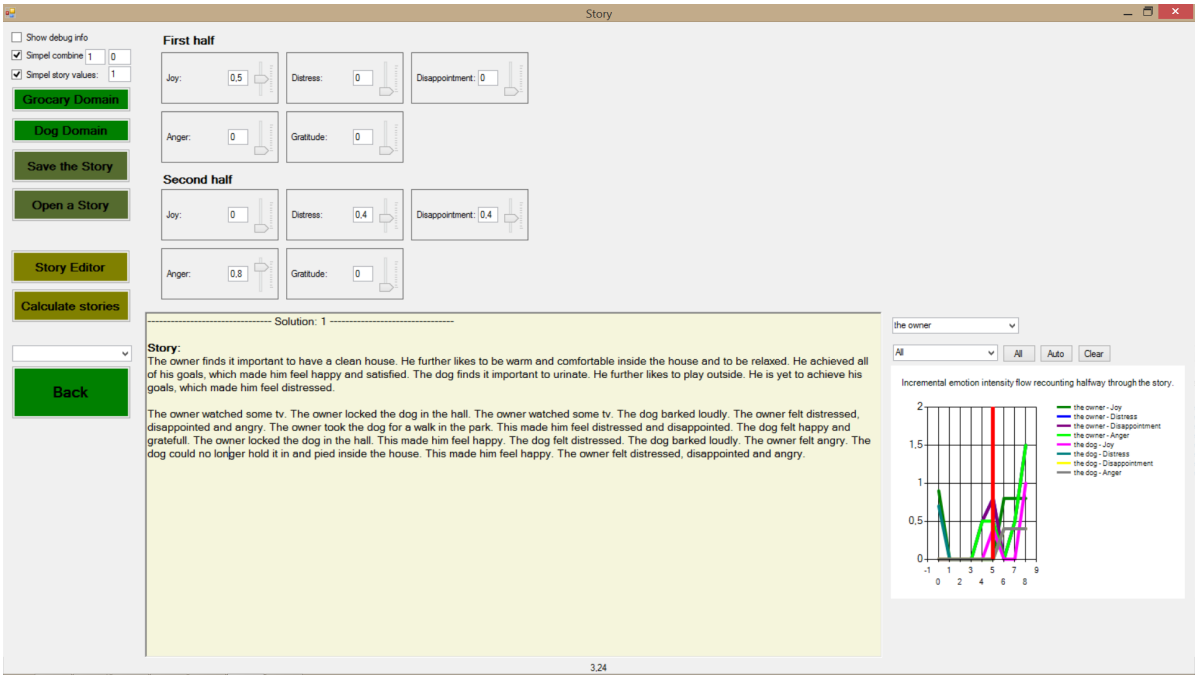

Figure A.1: Screen shot from the Affective Storyteller tool. Top showing the sorting panel in which the user can select which emotions are relevant for which part of the story (only showing actually occurring emotions). Right showing the emotion intensity in the story for the first and second part. Bottom showing the currently selected story.

A.1. INTRODUCTION

In this manual, the Affective Storyteller framework is explained. First the different steps of narrative generation are briefly discussed. When the reader is familiar with these concepts, the functionality offered by the Affective Storyteller is discussed, based on these steps in narrative generation. This manual can be used to gain a better understanding of the framework used in the report and as a reference for future research based on the Affective Storyteller.

The narrative generation in the Affective Storyteller mainly exists out of three parts. First it is needed to define a domain. The characters, locations and possible events that can populate the eventual story. Dependent on how the events are defined, they can happen on different moments in the story, e.g. multiple event sequences are possible. These different event sequences all represent some possible fabula that could be realized in this story-world. These event sequences are however not yet the story that the narrative generator is looking for. Not everything that is in a fabula necessarily needs to be present in the eventual story. Some

subset can be made leaving out information that is either redundant or implicit. This subset can be referred to as the sjuzet [6]. Finally these sequences need to be presented to the audience in some way which is referred to as the presentation of the story. This presentation can be anything that delivers the message, ranging from movies and textual presentations to even abstract representations like music. The Affective Storyteller, however, is limited to textual presentation only.

These before mentioned steps are quite standard and used in present day storytellers [12, 40]. The Affective Storyteller discriminates itself from these storytellers by using emotion as a key component in all of these processes. It allows users to define emotional heuristics for the story and then use these heuristics in the fabula planning.

A.2. A STORY DOMAIN; THE STORY EDITOR

The first step in story generation is defining the characters, locations and events out of which the fabula will consist. This step is still mostly the task of some human author in the affective storyteller. The human author needs to define agents that behave like characters. This means every agent needs to have statements concerning the world of which it thinks they are true. These are referred to as 'beliefs'. The agent further has goals which are beliefs that the agent wants to make or keep true. Finally the agent needs to have a set of actions it can perform. An action has pre-conditions which are beliefs that need to be true before the action can execute and post-conditions which are adjustments of beliefs for one or more agents in the domain. Figure A.2 shows the story editor which can be used to generate these type of domains.

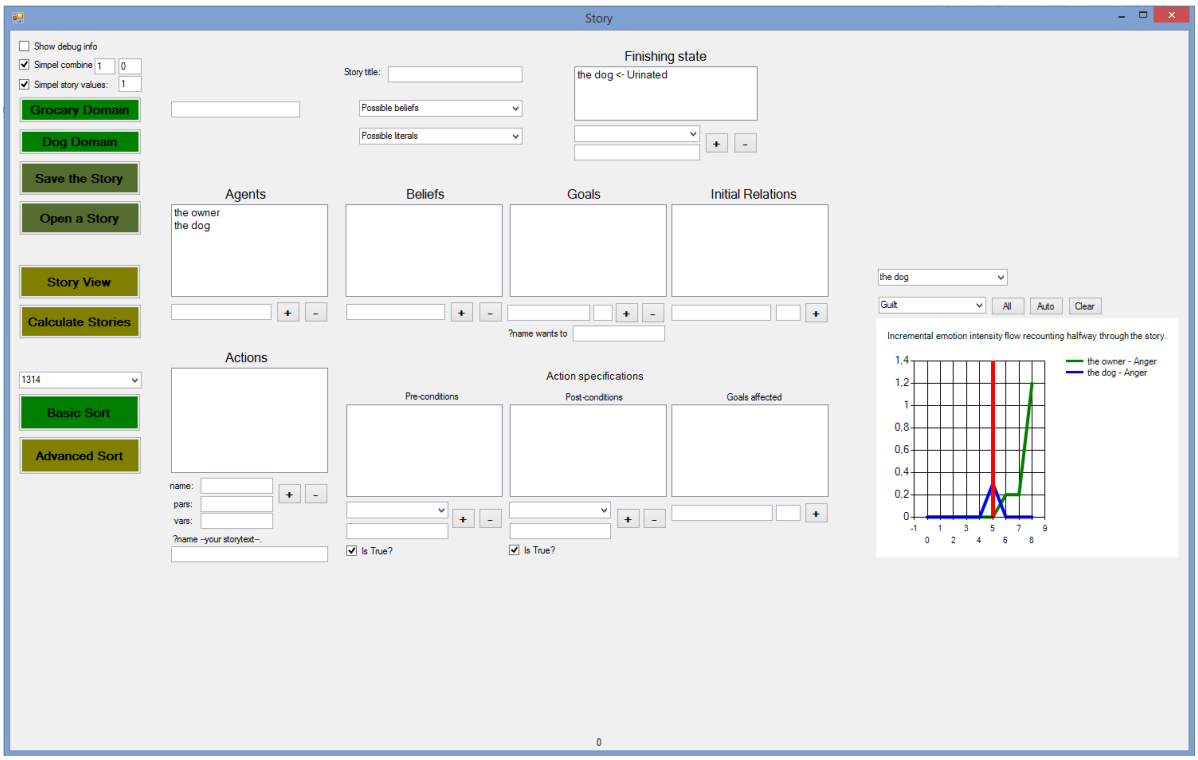

Figure A.2: The story editor. A tool to build and edit stories.

After pressing the button 'Story Editor' in the main screen, figure A.2 becomes visible. The story editor is capable of making any domain of the before mentioned type, but it is still work in progress. It lacks a good interface and is therefore somewhat tedious in use. Figure A.3 shows the syntactic structure for adding beliefs, goals and actions. First an agent should be added by defining a name and pressing the add button underneath as in A.3a.

Beliefs

This agent can have some initial beliefs prior to the story. Beliefs exist out of two elements. First the name of the belief (in the example at figure A.3bthis is 'Location'), optionally followed by 1 or more variables. In this

interface a variable is added using the '#' sign. The example in A.3bshows one variable, namely: 'living room'. It is then added using the plus button.

Goals

Goals have additional information that needs to be presented to the storyteller. A goal has a utility showing its importance, this value needs to be between [0, 1) and is used to later calculate emotion intensities. Important goals will inflict higher emotional intensities. A goal can also be given a story text. This will help the storyteller to generate better textual presentations of the generated fabula. When adding a story text the user should make a valid sentence with the following structure: '?name wants to -your storytext-. In example figure A.3cthis becomes: The owner wants to be warm and comfortable inside the house.

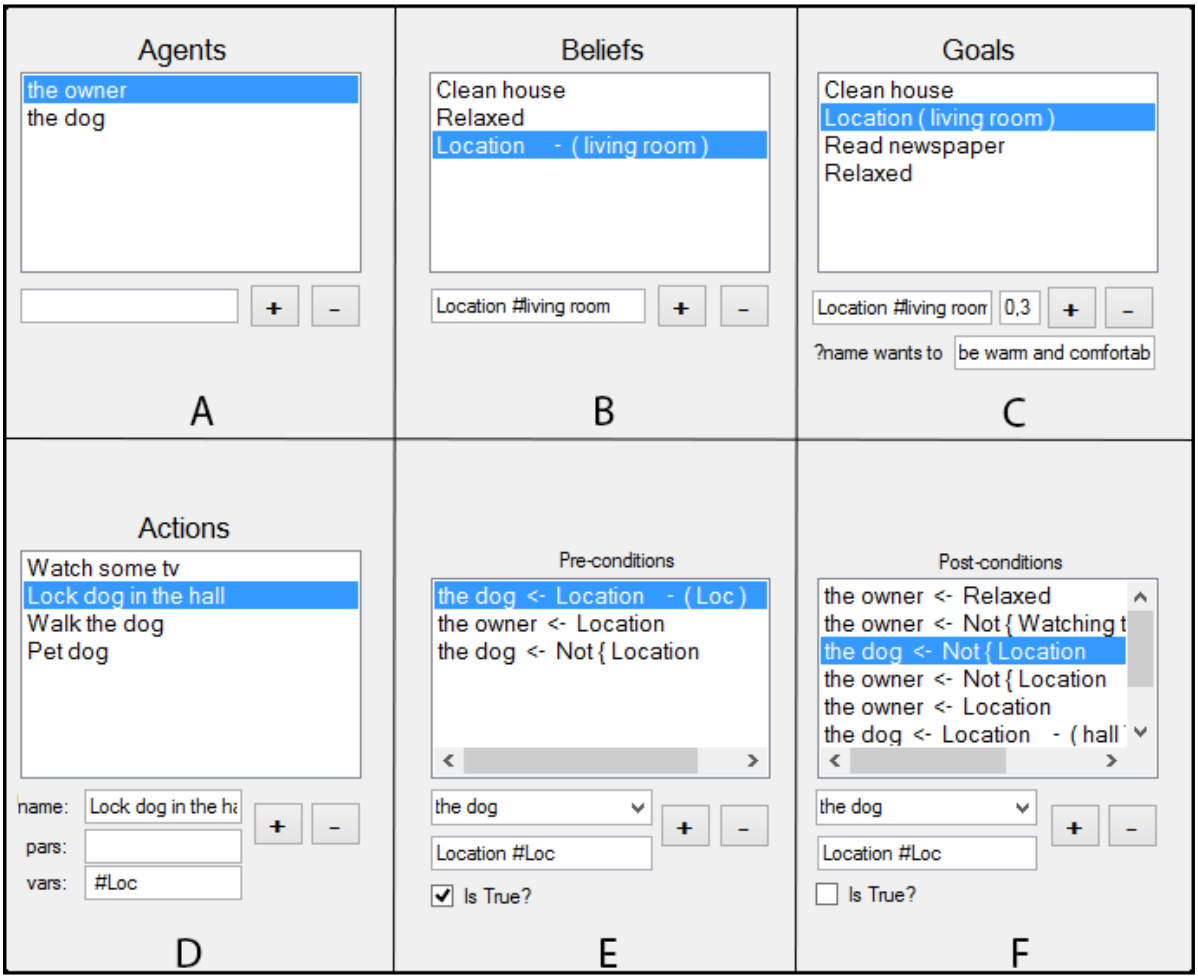

Figure A.3: The story editor. A tool to build and edit stories.

Actions

The agent is capable of performing actions that change the state of the environment. When defining an action in the Affective Storyteller a name needs to be defined. An action optionally has some parameters, variables and a story text for improved textual presentation of the story. Parameters are elements that the storyteller can fill in to create different actions. An example would be the action goto(X), where X is a parameter and can be filled in making different actions like goto(store) or goto(home). This action can then have some variable Y. This variable is filled in automatically by the actions' preconditions. Goto(X) could for example have as precondition Location(Y) and as postconditions Not Location(Y) and Location(X). Choosing the parameter X = 'store' and automatically finding Y = 'home', this means the agent now no longer believes it is home but instead believes it is at the store.

Figure A.3eand A.3fshow that pre- and postconditions are not necessarily referring to the agent itself. The selected action 'lock dog in the hall', is altering the belief base of the dog so that he now believes his location is the hall.

Deleting these different elements can easily be done by selecting the statement to be deleted and then pressing the button showing the minus sign. Future versions of the storyteller need easy editing of the different elements and more intuitive use of variables.

After defining the domain the affective storyteller links GAMYGDALA to the agents, so that it can calculate the different emotions after every event. With the domain defined and initialized the storyteller can start calculating the different possible event sequences.

A.2.1. TESTING YOUR DOMAIN; STORY VIEW

Defining a qualitatively good domain is often a far from trivial task. The story author should test if the actions perform as desired in term of belief-base updating and character emotions. To do this, we found that it is desirable to have a method that allows to manually perform actions and test them step by step. For this purpose, the story view panel is developed. In this panel the user can manually select some action, perform it, and see what changes emotionally and in the belief-bases of the characters.

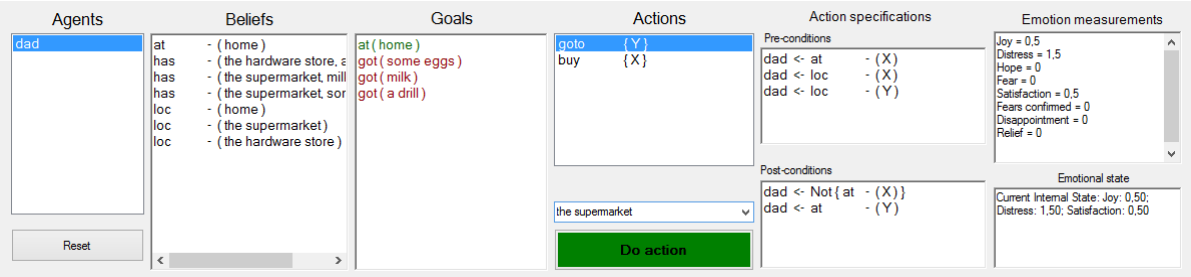

Figure A.4: The story view panel allows the user to manually walk through a story.

The user can simply select an action, like goto(Y) in figure A.4, and choose the desired initialization of parameters (in this example Y = 'the supermarket') and press perform action to see the belief-base and emotional changes in the characters.

A.3. GENERATING THE FABULA; CALCULATE STORIES

There are multiple fabula possible given a story-domain. By pressing the button 'Calculate stories' in the left menu panel (see figure A.1), these story-paths are found and listed. This is done with a breath first search through the tree of events. Different paths in the tree are searched until the finishing state is found. The affective storyteller however aims to find *all* possible paths rather than the quickest path towards the finishing state. This means that when a certain intermediate state is found that was already found before, that state can still be accepted. A state will however not be accepted if it was already found earlier on the same path. The example in figure A.5 should help to clarify this.

In the left sequence the agent goes home twice, without having changed anything and is therefore not accepted in the algorithm to prevent infinite looping. Should the agent however have bought something in the hardware-store, then there is a difference between the two times he went home, making the sequence valid. In the second sequence the agent results in the exact same state as in the previous one, but this time the sequence is accepted since there are no duplicate states inside the sequence itself.

This restriction in the fabula generation is a design choice to prevent infinite looping. The downside however is that it limits the possible fabula that can be found. A possible story could for example be about 'Hans the working guy'. *Every day Hans goes to work and buys coffee until one day the machine was broken. He fell asleep while working heavy machinery and accidentally tore down the apartment, costing the company millions of dollars.* This is a fine story but the fabula repeats Hans going to work and drinking coffee multiple times, which would normally be rejected by the algorithm.

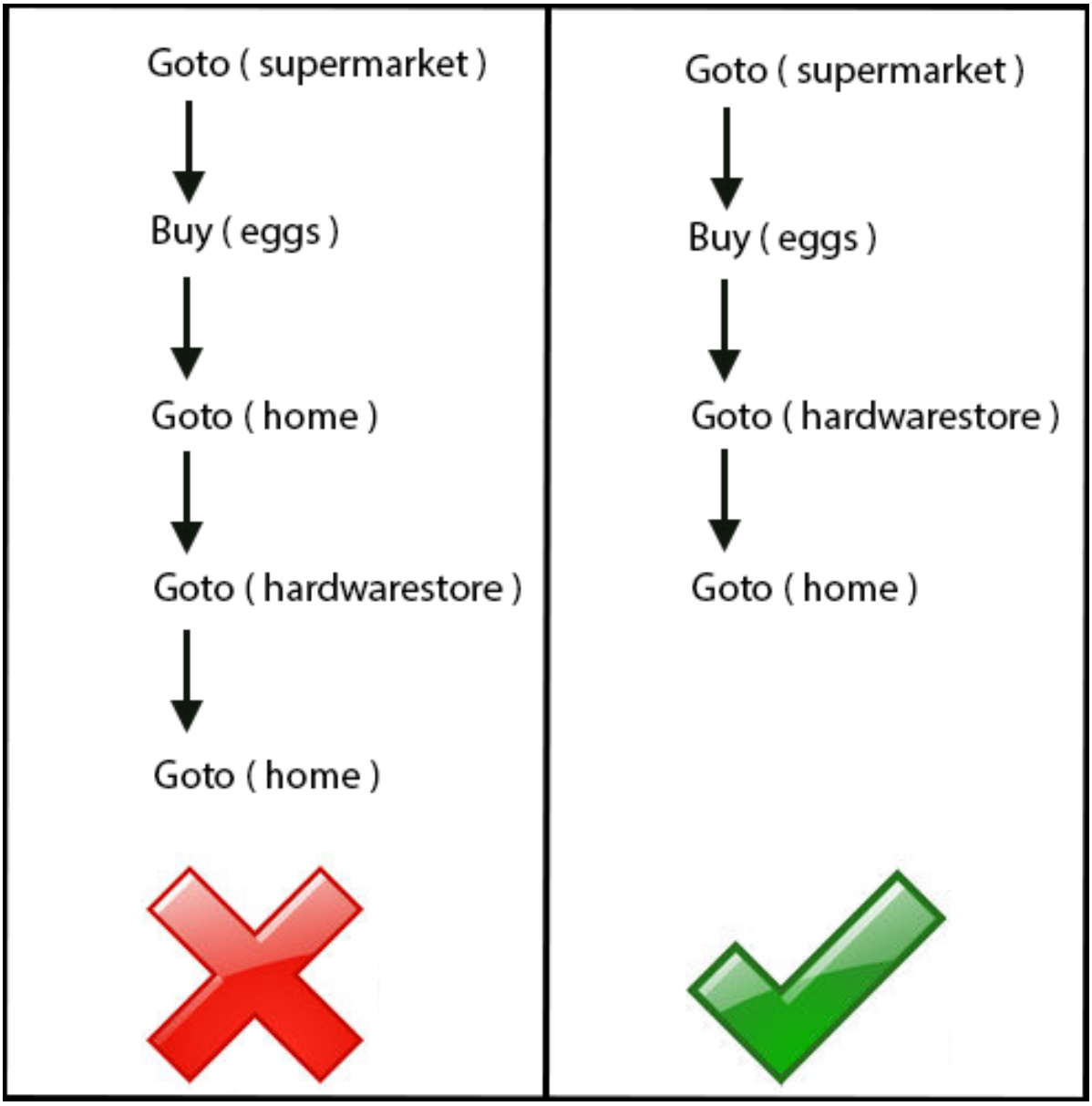

Figure A.5: A fabula can have some looping, but there should always be some, however minor, change in one of the agents' belief states. The left example above has exact duplicate state and is therefore rejected by the algorithm.

A.3.1. EMOTIONS IN THE FABULA

In every event sequence, e.g. every possible fabula, the emotions for the different characters are measured and logged. This is done through the emotional framework Gamygdala [28]. Gamygdala calculates emotions and emotion intensities based on the OCC model [15]. These emotion measurements will later be used to sort the stories and are used in the story presentation.

A.4. FROM FABULA TO STORY-TEXT; THE STORY-TEXT FIELD

When the fabulas have been generated, the next step is to transform them into story-texts. For this transition the user can choose whether he finds it desirable to let the Affective Storyteller generate the story-text fully automatically or to maintain more control over this process. The story-text is generated automatically by using three main building blocks: the goals of the agents, the main affective responses of the characters during the story, and finally the story events being the actions performed by the different agents. The first building block, the character goals, can be used to automatically generate an introduction to the story. Let us give an example of such a generated introduction. Consider a story domain with two agents. A 'owner' and

a ‘dog’. The owner has one goal with a high utility: ‘Clean house’. He has two less important goals: ‘Inside’ and ‘Relaxed’. The dog has one important goal: ‘Needs to urinate’. He has one less important goal: ‘Play outside’. The importance is defined with the utility values the author has given the goals. A utility above 0.7 is important. This value is chosen arbitrary and may need to change in other domains. The text is built in the following structure:

> *For every agent do: ?Agent_A found it important to ?Goal_A, ?Goal_B, .., and ?Goal_C. He further liked to ?Goal_D, ?Goal_E, .., and ?Goal_F. –current achievement of goals– . That made him feel –emotional state–.*

In the example domain this becomes:

> *The owner found it important to have a clean house. He further liked to be warm and comfortable inside the house and to be relaxed. He had achieved all of his goals. That made him feel happy and satisfied. The dog found it important to urinate. He further liked to play outside. He was yet to achieve his goals, which made him feel distressed.*

In this example it can be seen that what is presented in the intro are (a) for each actor the goals it finds important, (b) the extent to which these goals are met or not, and (c) the initial affective response towards this situation. Every goal and action have a story-text connected to them. So the goal ’Inside’ that the owner had provides the story-text: ’be warm and comfortable inside the house’. The Affective Storyteller helps the user to define these story-texts (see Appendix: A, figure A.3).

After the introduction the agents start performing actions. After every action the characters experience some emotional changes as simulated by GAMYGDALA. If the user chooses to let the Affective Storyteller perform this behavior then after every action the most salient affective changes (highest intensity change) in the characters are verbalized. For example, when the owner takes the dog for a walk in the park, it is only mentioned that the dog is happy and grateful. The fact that the dog still needs to urinate and that he is still distressed about this will not be mentioned. This is an intuitive and informed choice, but obviously further research is needed on how to verbalize emotional states of characters in stories.

The final and most important building block for the story-text transition process is the story-text connected to the different events in the story. (So the event ‘dog barked’ can have the story-text: ‘The dog barked loudly. ’) This text can again be generated automatically. This however only rarely gives desired texts, therefore the Affective Storyteller also provides the possibility to define logically constructed event texts. Meaning the story-text connected to an event can be different dependent on the outcome of some logic statements concerning character believes and emotions, the storyteller constructs the proper event-text. The structure of such an event-text can be seen below:

> The logic statement was: ?[<?agent_name»{ *some logic statement of beliefs and emotions* }]
> (true.)
> (false.) I’m sure of it!

The string ?[<Wolf>{ at #Grandma’s house && (Ate #Grandma || !Hungry)} && <Red>{at #Grandma’s house && €Distress3}] is true when the Wolf and Red are both at grandma’s house, the wolf has either eaten grandma or is not hungry and little red riding hood is distressed about something. So when this statement is *true* when the action the story-text belongs to is performed, then the Affective Storyteller would print: ‘This logic statement was true. I’m sure of it!’. Logic statements can be nested and can contain logic constructors like ||, &&, ! and brackets. A logic statement is built with agent beliefs and emotions.

With this technique it is possible to maintain a lot of control over the produced story-text allowing the Affective Storyteller to write more interesting stories. An example of a story produced from the little red riding hood domain, using this technique, can be seen in Appendix D. Important to understand for this technique is that it is not intended to try to have a different story-event text for every possible event sequence. Even though this is theoretically possible, defining the story-text for over 10.000 possible cases would be far from desirable. Rather than doing this it is simply intended to write logic sentences with minor differences, dependent on what may have preceded the event.

Story-event texts also need a defining symbol at the start of the sentence. This symbol represents the event location. When generating the story-text the Affective Storyteller needs a mechanism to determine

when to segment the text into paragraphs. The chosen mechanism for this is to write a paragraph when the event location is different from the previous event. When for example Little Red Riding Hood travels the road and then meets the wolf, then no new paragraph would be desired. If however red travels the road and then the wolf knocks on grandma's door then this would read strangely without starting a new paragraph. When automatically generating the story-event text a default event location is defined.

A.5. BASIC EMOTIONAL SORTING; STORY-SORTING PANEL

The emotions measured by GAMYGDALA can be used to sort the stories. The Affective storyteller allows two different methods of emotional sorting. A basic method that compares some predefined desired emotional measurement to the actual measurements in the different fabula and a more complex method using emotional scripting. In this section the basic sorting is explained.

GAMYGDALA generates 16 different emotions. The events that occur in the story cause the characters to experience these different GAMYGDALA simulated emotions with a certain intensity. Every time an emotion is experienced its intensity is added to a cumulative sum showing the total emotional intensity. When recounting halfway through the story, these values show the total experienced emotion intensities in the first and second half of the story. In basic emotional sorting, the Affective storyteller calculates the total sum for each of the emotion intensities in both the first and second half of the story being a total of 32 measurements for every event sequence. The panel in figure A.6 however only shows 12 emotions, making 24 panels. This is because the story-sorting panel only loads emotions that actually occur in the corresponding half of the story.

Figure A.6: The basic sorting panel.

The user can define emotion flows that the story should follow. The user chooses for each emotion, for both the first and second half of the story, how important this occurrence is on a scale from 0 to 1. 0 meaning this occurrence holds no importance and 1 that it is very important. This again results in a total of (max) 32 numbers. When we look at these two measurements as vectors then we can use the dot product as a value function for the different event sequences.

$$Value(Seq) = \sum_{i=1}^{32} \left\{ EI_i * EC_i \right\}, \tag{A.1}$$

EI stands for the emotion intensity, EC is the emotion congruency as defined by the user. A problem with this function is that some emotions may have such strong intensities that they shove away the other emotions in the outcome of equation A.1. A way of preventing this is to cast the individual emotion intensity measurements to a value between 0 and 1.

$$Value(Seq) = \sum_{i=1}^{32} \left\{ \left(\frac{EI_i * gain}{EI_i * gain + 1} \right) * EC_i \right\}. \tag{A.2}$$

Gain is a constant that defines how fast the emotion intensity values increase from 0 to 1. Whether function A.2 is better could be questioned since it may very well be intended by the user that stories with a lot of joy would shove away stories with a little joy and some hope.

A.6. STORY DEBUG MODE

When analyzing your stories or the stories from the example library it can be desirable to see them in different forms than the plain story-text. In the story-view panel a story can be walked through manually. Besides the story-view panel, there are two different story presentation modes available that show more abstract information about the fabula. The first method is by toggling the checkbox in the upper left corner (figure A.1), activating the story debug mode. The storyteller then shows stories in the following format:

```
printf ( ?agent_name performed action ?action_name ?action variables. )
printf ( action → storytext of emotional consequence )
foreach agent agent in story
    printf ( ?agent_name felt: agent.emotions.ToString() )
end
```

An example output of this would be:

```
The owner pet his dog.
This made him feel resentful. The dog felt happy and grateful.
The owner felt:
Current Internal State: Joy: 0,75; Distress: 0,40; Satisfaction: 0,75; Disappointment: 0,20
Relationships:
    the dog → Anger: 0,20; Resentment: 0,20
The dog felt:
Current Internal State: Joy: 0,55; Distress: 1,10; Hope: 0,24; Satisfaction: 0,15
Relationships:
    the owner → Gratitude: 0,40
```

This output is somewhat difficult to grasp. We will explain it step by step. The first sentence simply states which agent does what. In this case the owner makes a friendly gesture by petting his dog. The second sentence shows the emotional consequence of this action. The dog responds as expected, he is happy and grateful. The owner shows a less conventional emotion, apparently he feels resentment. When seeing such an unexpected emotion the user can try to investigate why the character has it. First it can be good to review what triggers resentment. Appendix B shows a table with al GAMYGDALA emotions and what triggers them. Resentment in this table is defined to be a desirable event happening to a disliked character. So the emotion is triggered if the owner does not like the dog. To see if this is indeed the case more emotional information is required. The rest of the output shows the complete emotional state of all the agents, including social emotions describing their relationship. The emotions the owner feels in regards to the dog can be seen in the emotional state of the owner (after 'The owner felt' and before 'The dog felt'). When looking at his relationship towards the dog it seems one of the previous actions made him angry with the dog. This causes him to dislike the dog, which causes him to feel resentment towards his pet.

When looking at this example output the reader could reason that it is not desirable. An action generating negative emotions towards the character itself should perhaps only be allowed if there are no other actions available since it would otherwise appear the characters are not acting intentionally. The downside of such an approach is that people do not always act optimally, we make mistakes causing us to feel emotions like resentment, guilt and distress. Finding an optimized strategy of using emotions to support character intentionality and story believability at this point still remains to be future work.

A.7. VISUALIZATION; EMOTIONAL GRAPHS

Another method to analyze your stories is by using the emotion graphs panel. This panel allows different initializations showing the emotion flows your characters follow. Figure A.7 shows two initializations of this panel.
The emotion graph panel is directly linked to the basic sorting to enable easy analyzing of the stories. When pressing the button auto in figure A.7, the engine automatically finds all emotions that contribute in the basic sorting. The engine finds all emotions that are set above zero in the basic sorting panel (figure A.6), the emotion flows are then added in the graph as shown in figure A.7. Alternatively emotions can also be added

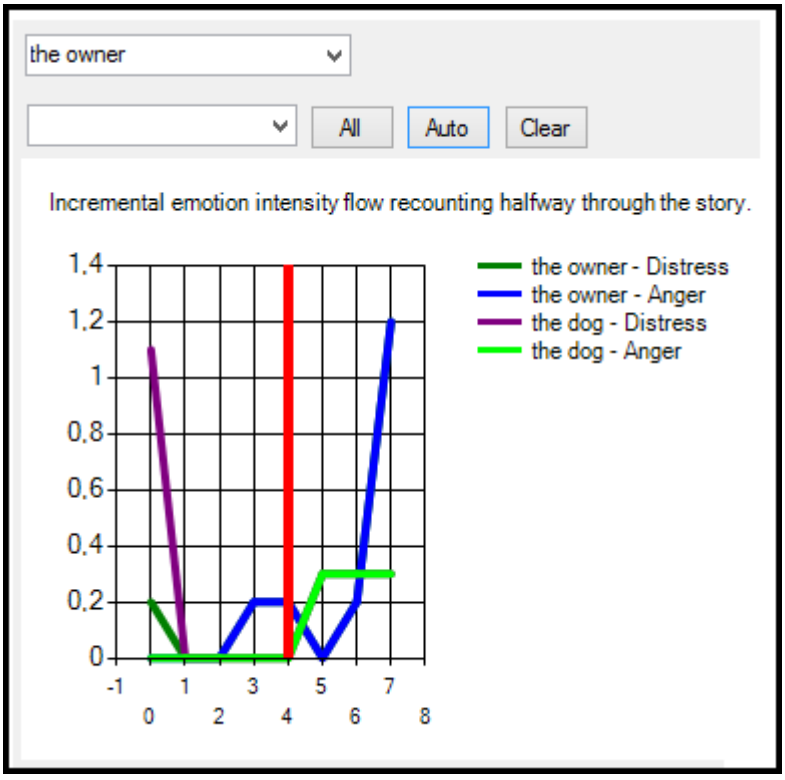

Figure A.7: A visualization of the emotional flows in the story-characters.

manually by selecting and deselecting emotions for every character in the dropdown boxes.

An emotion flow shows the cumulative sum of emotion intensities as explained in section A.5. In this way it can easily be seen how well the best conforming story actually performs given the requested emotion flow.

A.8. ADVANCED STORY SORTING

The advanced sorting panel allows more specific control over the emotional flow in the story than the basic mechanism. There are two methods available for advanced sorting. The emotional heuristics script (EHS) in which the user defines an arbitrary amount of heuristics to which the story should comply and emotional regular expressions (EmoRegEx) that are an extension of the original regular expressions.
By pressing Advanced sorting in the main screen (figure A.1) the screen in figure A.8 should become visible. The advanced sorting panel exists out of a simple textbox with two buttons below. In the textbox the user defines either the emotion heuristics or the EmoRegEx text. By pressing the corresponding button below, the Affective storyteller will compile the text and search the current fabula set to find all stories that fulfill the prerequisites. The advanced sorting methods are perhaps more filters than an actual sorting mechanisms. Stories are sorted into a group of satisfying stories and a group of stories that do not comply with the defined heuristics. The syntax of the two mechanisms is explained below.

A.8.1. EMOTIONAL SCRIPT; HEURISTICS

The emotion script has an XML-like format as in [45] and exists out of multiple emotion meta-data blocks that we will refer to as heuristics. Every heuristic mentions an emotion that should be experienced by one or more actors with a certain intensity and at a certain moment in the story. An example of such a heuristic can be seen below:

Emotion ID = start_story_hopeForAnyone
{
    Category = Hope
    Intensity_min = 0.3

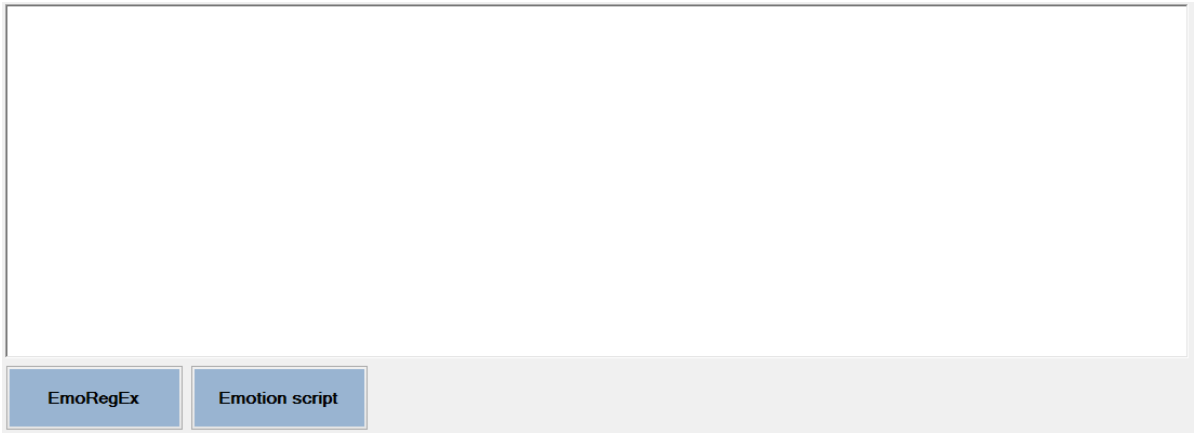

Figure A.8: The advanced sorting panel can read two emotional languages, emotional heuristic scripts (EHS) and emotional regular expressions (EmoRegEx). Given the defined filtering statements in one of these languages the Affective storyteller will sort and present the stories.

Intensity_max = 1
ExperiencedBy = *Any*
Order = 0
Freedom = 3
}

The user can define as many heuristics as desired. We will explain the different elements in these heuristics one at a time. The first four elements between the curly brackets define some emotional state that should be experienced by one or more agents. The elements 'order' and 'freedom' define a relative position in story time stating when in the story this emotional state should be apparent.

Emotion ID
Every heuristic needs an ID to reference to for both the engine and the user himself. This ID can have any value as long as it is distinctive. The ID could simply be named '1', but when the user starts to define many of these meta-data blocks it can become difficult to oversee them all. The name given in the above example tries to explain something about the meaning of the heuristic which makes it possible to see its functionality in one look.

Category
The category should be filled in with one of the 16 emotions produced by GAMYGDALA (see Appendix B). This is the emotion that the user desires to be present in the story. The category needs to be defined for the script to compile.

Intensity_min
The minimum intensity the emotion should have. Emotion intensities are always between 0 and 1. When there is no value specified then the compiler will automatically fill in the value 0.

Intensit_max
The maximum intensity the emotion should have. When there is no value specified then the compiler will automatically fill in the value 1.

ExperiencedBy
The agent or agents that should experience the emotion. This field can be filled in with some agent name (for example: 'the owner') or it can contain one of the quantifiers 'Any' or 'All'. When it contains 'Any' as in the example, then the algorithm will aim to have one of the agents feel this emotion. 'All' will force the algorithm to have all agents experience the emotion.

Order
Order defines a relative position in the story. When there are many different heuristics defined then the algorithm sorts these based on their order number. It will then first aim to accomplish every heuristic of the lowest order number before it continues to the next number. The example below shows three metadata-blocks.

Emotion ID = start_story_hopeForAnyone
{
Category = Hope
Intensity_min = 0.3
Intensity_max = 1
ExperiencedBy = Any
Order = 0
Freedom = 3
}
Emotion ID = 2_end_story_resentmentForAnyone
{
Category = Resentment
Intensity_Min = 0.1
ExperiencedBy = Any
Order = 1
Freedom = 3
}
Emotion ID = 3_start_story_angerForAll
{
Category = Anger
Intensity_Min = 0.2
Intensity_Max = 1
ExperiencedBy = All
Order = 0
Freedom = 5
}

In this example the algorithm would first aim to fulfill the emotions *1_start_story_hopeForAnyone* and *3_start_story_angerForAll*. It starts performing the events in the fabula, one at a time. When the order 0 emotions are fulfilled the algorithm will no longer look at them and start trying to fulfill all emotions with order 1. In this case that is only the emotion *2_end_story_resentmentForAnyone*. It continues performing the events in the fabula until either the story is over or the next order number emotions are accomplished. When the story is over and there are still heuristics (metadata blocks) left to fulfill, then the algorithm returns a fail. The story does not fulfill the defined heuristics. If all the heuristics *are* fulfilled, then the story can be shown to the user.
When the order increases it is no longer needed to have the lower order emotions (in the example hope and anger) present in the characters. The algorithm just continues performing fabula events until it either succeeds or fails.

Freedom
Freedom allows the user to define how much events are allowed to happen before a heuristic should be fulfilled. This means that in the above example the heuristic *1_start_story_hopeForAnyone* should be fulfilled after at most 3 events and stay fulfilled until all order 0 heuristics are fulfilled.
With these emotion heuristics the user has extensive possibilities in defining emotional flows for different stories. Functionally there is however still a drawback since it is not possible to define statements like agent A should feel either Joy OR Hope at the beginning of the story. These disjunctive statements could still be achieved by adding to this emotional script language but this could become a bit tedious and make the language even more difficult to read.

A.8.2. EMOREGEX
This last and perhaps most powerful method of story filtering is based on regular expressions. It searches the story debug string (the same one as explained in section A.6) and looks if a user defined pattern can be found.

To make this possible we have developed an extension of the regular expression language itself. In this extension language a user can talk about events, agents and emotions in a natural manner and the compiler will transform the EmoRegEx string into a normal regular expression. An example of such an EmoRegEx string can be seen below:

```
${0,3}<the owner>€Joy#{ 2 * numOfAgents }$*<the dog>€Anger4
```

The compiler transforms this string into the following regular expression:

```
\$0[^\$]*(\$[^\$]*) {0,3}the\ owner\ felt[^\$€]* Joy:\ (1|(0,[4-9]))
[^\$]*(\$[^\$]*)* the\ dog\ felt[^\$€]* Anger:\ (1|(0,[4-9]))[^\$]*
```

Without any premature knowledge about regular expressions and/or emotional regular expressions, it can already be seen that the first expression looks a lot friendlier and easier in use. In this example the string should be read as: '*after 0 to 3 action the owner felt a little joy, then after any amount of actions the dog felt a little anger*'. Table A.1 explains the above EmoRegEx string step by step. Some parts are part of the basic language and some parts come from the extension language. The power of EmoRegEx is that it is possible to use almost any part of the normal regular expressions. It only adds to those by giving a special meaning to some statements that talk purely about events, agents and emotions in a story.

$	This sign overrides the original meaning in RegEx. Originaly it meant 'start the search at the beginning of the string'. In EmoRegEx the search *always* starts at the beginning of the string, making this redundant. In EmoRegEx it means **'perform one action´**.
{0,3}	This is basic regular expression syntax with 'm' and 'n' being integer values. It means: **'Repeat the previous statement at least m, but no more than n times'**.
${0,3}	The above two rules explain that this should be read as: **'perform 0 to 3 actions'**.
<?agent_name>	This means that **the following emotional statements are going to apply to *?agent_name.*** *?agent_name* can be initialized with the name of an agent that is a part of the story or with one of the quantifiers 'any' or 'all'.
€?emotion	An emotional statement, asking the algorithm to **look for emotion *?emotion* in the current agent**. *?emotion* can be one of the 16 GAMYGDALA generated emotions (see appendix B).
€?emotionX	The **X** is an integer and talks about **the minimum intensity the agent should experience for this emotion**. Emotions can have an intensity from 1 to 10 (0 means the emotion is not present). It is also possible to define a maximum intensity for emotions. For this the syntax is ^X. Meaning that for example Anger^1 means that the character is not allowed to experience any anger.
#{ 2 * numOfAgents}	The text between #{ } is treated as a mathematical formula. This is calculated before transforming the EmoRegEx string into the corresponding RegEx string. There are different variables possible. For now the variables numOfAgents and lengthOfStory can be used. In future updates of EmoRegEx this can be expanded.
${0,3}<the owner> €Joy#{ 2*numOfAgents }	After 0 to 3 story-events the owner should feel Joy with a minimum intensity of 2 x the total number of storyAgents (= 4).
$*	The * is basic RegEx syntax. It means: '**Repeat the previous statement any number of times**'. Any number can also be zero.
${0,3}<the owner> €Joy4$*<the dog> €Anger4	After 0 to 3 story-events the owner should feel Joy with a minimum intensity of 4, then after any number of actions the dog should feel Anger with a minimum intensity of 4.

Table A.1: EmoRegEx Syntax

B

GAMYGDALA EMOTIONS

In this Appendix the 16 emotions generated by GAMYGDALA are briefly explained. For a full explanation we refer to the paper [28].

Emotion name	Trigger	Description
Joy	$(U(g) > 0 \wedge L(g) == 1) \vee$ $(U(g) < 0 \wedge L(g) == 0)$	A desirable goal succeeds or an undesirable goal fails.
Distress	$(U(g) < 0 \wedge L(g) == 1) \vee$ $(U(g) > 0 \wedge L(g) == 0)$	An undesirable goal succeeds or a desirable goal fails.
Hope	$(U(g) > 0 \wedge L(g) < 1 \wedge \Delta L(g) > 0) \vee$ $(U(g) < 0 \wedge L(g) > 0 \wedge \Delta L(g) < 0)$	A desirable, uncertain goal increases in likelihood of success or an undesirable, uncertain goal decreases in likelihood of success.
Fear	$(U(g) > 0 \wedge L(g) > 0 \wedge \Delta L(g) < 0) \vee$ $(U(g) < 0 \wedge L(g) < 1 \wedge \Delta L(g) > 0)$	An undesirable, uncertain goal increases in likelihood of success or a desirable, uncertain goal decreases in likelihood of success.
Satisfaction	$U(g) > 0 \wedge L(g) == 1 \wedge \lvert\Delta L(g)\rvert < 0.5$	A desirable goal was expected to succeed and it actually does succeed.
Fears confirmed	$U(g) < 0 \wedge L(g) == 1 \wedge \lvert\Delta L(g)\rvert < 0.5$	An undesirable goal was expected to succeed and it actually does succeed.
Relief	$U(g) < 0 \wedge L(g) == 0 \wedge \lvert\Delta L(g)\rvert > 0.5$	An undesirable goal was expected to succeed and it fails.
Disappointment	$U(g) > 0 \wedge L(g) == 0 \wedge \lvert\Delta L(g)\rvert > 0.5$	A desirable goal was expected to succeed and it fails.
Anger	$des(b,g,self) < 0 \wedge Agent(b) = !self$	An undesirable event is caused by another agent.
Gratitude	$des(b,g,self) > 0 \wedge Agent(b) = !self$	A desirable event is caused by another agent.
Guilt	$des(b,g,other) < 0 \wedge$ $Agent(b) == self \wedge$ $like(self,other) > 0$	The agent causes an undesirable event for a liked agent.
Gratification	$des(b,g,other) > 0 \wedge$ $Agent(b) == self \wedge$ $like(self,other) > 0$	The agent causes a desirable event for a liked agent.
Happy for	$des(b,g,other) > 0 \wedge$ $like(self,other) > 0$	A desirable event happens to a liked agent.
Pity	$des(b,g,other) < 0 \wedge$ $like(self,other) > 0$	An undesirable event happens to a liked agent.
Gloating	$des(b,g,other) < 0 \wedge$ $like(self,other) < 0$	An undesirable event happens to a disliked agent.
Resentment	$des(b,g,other) > 0 \wedge$ $like(self,other) < 0$	A desirable event happens to a disliked agent.

Table B.1: In this table the different emotions generated by GAMYGDALA are described including the exact state that an agent should be in to experience them. U(g) stands for the utility of a goal. L(g) for the goal likelihood. $|\Delta L(g)|$ is the delta likelihood; the change in likelihood before and after the event. The reader does not need to understand more about des(b, g, ?agent) than: 'the event (e) influences goal (g) owned by agent (self or other) in a desirable (value above 0) or undesirable (value below 0) way. Responsible_Agent(e) is the agent causing event 'e' and finally like(?agent, ?agent) shows a relation between two agents.

C

JEWISH STORY

Once upon a time there was a women named Truth. Truth wanted to share her wisdom for she knew it would make the world a better place. So she went into town and to the town square. She gathered the people around here and spoke to them but no one would listen. The minute she started talking, people ran away and shouted in terror. People went into their houses and shut the doors and windows. In a matter of minutes there were only very few people left in the streets.

What could she do? If people wouldn't even listen to her then how could she deliver her message? In absolute despair Truth did the last thing she could think of to make the people pay attention to her. She took off her clothes and tried again but the result was worse. The people ran away from her. They went into their houses and shut the windows.

All alone she cried and was just about to leave town when something remarkable happened. She heard people leaving their houses again. She heard laughter and joy. People were talking to each other and all despair had left their faces. Truth looked around and then she saw a woman walking towards her. The townsmen were following this woman with happiness in their faces.

The woman noticed Truth and walked towards her. 'Why are you crying?' She asked. Truth was very upset and cried: 'Everyone was scared of me! No one wanted to listen to my words.' The woman looked sympathetic. 'That is because no one likes the truth. Especially the naked truth.' The woman smiled and gave Truth her beautiful sparkling cloak. 'Try again' she said. Carefully Truth tried talking to the townsmen again and like a miracle, this time people listened. They smiled and nodded by her words. No one ran away. For you see, the woman that gave truth her cloak was Story and when cloaked by story even truth is liked by everyone.

- Jewish Story

The Jewish story about the truth and story has existed over many years and has been told and retold many times and in many ways. My first encounter with it was during an online college by Professor Brian Sturm. I remember I was impressed by this story because I felt that it was one of those rare stories that touched the absolute truth behind things. I wanted to tell the story on, hoping that other would experience this feeling as well. The presentation of the story above is assembled with some personal touch and freedom, but hopefully without losing its impact.

D

Little Red Riding Hood

One of the story-domains in the Affective Storyteller is the Little Red Riding Hood domain. In this appendix we would like to show you one of the automatically generated stories by the Affective Storyteller. The story-text is constructed using the logic story-event texts as explained in section 4.4. An abstract fabula text, such as shown to the test subjects in section 6.2, would be:

> Little Red Riding Hood traveled the road, to go and visit her grandma
>
> Grandma went hunting
>
> The wolf gave Red route directions
>
> It became night
>
> Grandma saw some wolf cubs trapped in a river current. The mother wolf was nearby, but did not see the danger for her cubs.
>
> Grandma killed the wolf cubs. The mother wolf was nearby and saw what happened.
>
> The wolf knocked on grandma's door, but she was not home
>
> The wolf ate grandma in the forest
>
> Red arrives at grandma's house

The story-text linked to these events can give this story a much more extensive presentation. We will explain the transition of the first event: 'Little Red Riding Hood traveled the road, to go and visit her grandma'. The story-text that the storyteller prints for this event dependents on whether grandma has already sent Little Red Riding Hood an invitation, and whether it is day or night. The story-text linked to this event can be seen below:

> $3?[<Red>{Got message}]
> (After receiving grandma's message, Little Red Riding Hood quickly grabbed her basket and went out the door.)
> (Little Red Riding Hood decided to go and see her grandmother.) ?[<Wheel of time>Day]
> (She loved to travel the road. Although it could get a bit scary in the dark, at day, the forest road was one of her favorite places in the world. 'I'd better make haste though, before it gets too late', she thought by herself. 'Perhaps it's best to take a shortcut.' Red had only taken the short-cut once before, with her grandma, but she was convinced she would not get lost. Besides, it would save some traveling time.)
> (She hated to travel the road at this time a night, but she didn't want to disappoint her grand-mother, so she went anyway.)

The story-text starts with '$3'. This is an abstract representation showing the location of the event as explained in Appendix A section A.4. It is used by the storyteller to determine when to start a new paragraph. After this, the first logical statement is shown; '?[<Red>{Got message}]'. This is *false* at the start of the story; grandma did not yet sent a message. This means the text between the second two brackets is printed: '*Little Red Riding Hood decided to go and see her grandmother.*' The second statement is: '?[<Wheel of time>Day]'. The story starts at day, so the complete text for this event becomes: '*Little Red Riding Hood decided to go and see her grandmother. She loved to travel the road. Although it could get a bit scary in the dark, at day, the forest road was one of her favorite places in the world. 'I'd better make haste though, before it gets too late', she thought by herself. 'Perhaps it's best to take a shortcut.' Red had only taken the shortcut once before, with her grandma, but she was convinced she would not get lost. Besides, it would save some traveling time. '* These logic statements could also have been nested. For illustrative purposes let's consider the following text:

> $3?[<Red>{Got message}]
> (After receiving grandma's message, Little Red Riding Hood quickly grabbed her basket and went out the door.)
> (Little Red Riding Hood decided to go and see her grandmother. ?[<Wheel of time>Day]
> (She loved to travel the road. Although it could get a bit scary in the dark, at day, the forest road was one of her favorite places in the world. 'I'd better make haste though, before it gets too late', she thought by herself. 'Perhaps it's best to take a shortcut.' Red had only taken the shortcut once before, with her grandma, but she was convinced she would not get lost. Besides, it would save some traveling time.)
> (She hated to travel the road at this time a night, but she didn't want to disappoint her grandmother, so she went anyway.))

Notice that the closing bracket before the second logical statement is missing. This means that this text is part of the text printed if Red did not receive grandma's message. In the example the storyteller still prints the same text. Should Red however have received the message then the storyteller would only print: '*After receiving grandma's message, Little Red Riding Hood quickly grabbed her basket and went out the door.*' Instead of the complete text: '*After receiving grandma's message, Little Red Riding Hood quickly grabbed her basket and went out the door. She loved to travel the road. Although it could get a bit scary in the dark, at day, the forest road was one of her favorite places in the world. 'I'd better make haste though, before it gets too late', she thought by herself. 'Perhaps it's best to take a shortcut.' Red had only taken the shortcut once before, with her grandma, but she was convinced she would not get lost. Besides, it would save some traveling time. '*

The story below shows the final result of this story:

> Little Red Riding Hood decided to go and see her grandmother. She loved to travel the road. Although it could get a bit scary in the dark, at day, the forest road was one of her favorite places in the world. 'I'd better make haste though, before it gets too late', she thought by herself. 'Perhaps it's best to take a shortcut.' Red had only taken the shortcut once before, with her grandma, but she was convinced she would not get lost. Besides, it would save some traveling time.
>
> Looking at her meat supply grandma realized she was running out. 'I'd better go out and get some more.', she thought by herself.
>
> She should have seen it coming. Of course she had gotten lost. The route looked completely unfamiliar. 'What should I do?', she asked herself. It was at that moment that the wolf approached Red. 'Don't be scared child!', the wolf said with her sweetest voice. 'I'm not here to hurt you. I'm here to help you!' Now Red was not completely comforted, but she realized that running would be useless and that without the wolf she might be lost for hours and by that time it would probably be dark. She decided that she could best take her chances with the wolf. 'Can you show me the way to my grandmother's house?', Red asked. 'Well that depends, kid. I know the forest pretty well so I might, but can you explain to me where it should be?' That gave Red some hope. 'Yes, yes I can!' So Red told the wolf where grandma's house should

be and after helping Red with some directions the wolf went her own way.

Night fell over the forest, as dark creatures prepared for the hunt. The road that was so lovely at day had now gotten pretty scary for Little Red Riding Hood.

Out in the forest grandma stumbled upon a disturbing sight. Some little wolf cubs were trapped in a river flow. They were holding on to a branch, but if nothing was done then they would quickly be dragged by the river and head towards the waterfall. She slowly approached them, carefully so she wouldn't fall in the water herself. However, what she didn't know, was that the mother wolf was nearby. Grandma took the last few steps and reached for the branch. The little cubs were squeaking, as grandma looked them in the eyes. It was at that moment that the mother wolf saw grandma. The wolf didn't know that the little wolfs were drowning, but she did see the old lady holding her cubs. She howled into the night and ran towards her cubs. Grandma heard the sound and was reminded to why she had to do this. 'Murderers, that what you will become!' She dropped the cubs into the river and ran away. The mother wolf saw the old lady running and her cubs going down the river. Within a split second she had made up her mind and jumped after her cubs, and into the strong current.

Red's directions were good. There it was, the old lady's house. 'Now it is time for this old lady to meet the big bad wolf.' The wolf showed her teeth, in the moonlight it seemed as if she was possessed and was laughing insanely. The wolf jumped at the door, aiming to mimic the sound of a knock. No response. The wolf knocked again. 'Hmm.. it seems no one is home. Perhaps I should try again later.

Grandma was startled after hearing a sound in the bushes. She looked, and there she saw her worst nightmare come true. A giant wolf slowly approached her. 'Murderer!', the wolf cried. Without a second hesitation grandma jumped towards her hunting equipment, but it was too late. The wolf was too fast. With one leap the wolf was at her and in one huge bite, the giant wolf ate grandmother.

When Red arrived at grandma's house, she cried, not knowing what to do without her grandma, but is was already too late. The wolf had gone home again. Perhaps she should just be happy that she got out save herself. Because in real life, things do not always end happily ever after.

- Story generated by the Affective Storyteller

BIBLIOGRAPHY

[1] B. W. Sturm, *The" storylistening" trance experience,* Journal of American folklore , 287 (2000).

[2] A. Simmons, *The story factor: Secrets of influence from the art of storytelling* (Basic books, 2006).

[3] R. a. Mar, K. Oatley, M. Djikic, and J. Mullin, *Emotion and narrative fiction: Interactive influences before, during, and after reading.* Cognition & emotion **25**, 818 (2011).

[4] M. Djikic, K. Oatley, S. Zoeterman, and J. B. Peterson, *On Being Moved by Art: How Reading Fiction Transforms the Self,* Creativity Research Journal **21**, 24 (2009).

[5] M. O. Riedl and V. Bulitko, *Interactive narrative: An intelligent systems approach,* AI Magazine **34**, 67 (2012).

[6] M. Bal and C. Van Boheemen, *Narratology: Introduction to the theory of narrative* (University of Toronto Press, 2009).

[7] P. Gervás, *Computational approaches to storytelling and creativity.* AI Magazine **30**, 49 (2009).

[8] R. J. Gerrig and A. B. Bernardo, *Readers as problem-solvers in the experience of suspense,* Poetics **22**, 459 (1994).

[9] T. Trabasso and L. L. Sperry, *Causal relatedness and importance of story events,* Journal of Memory and language **24**, 595 (1985).

[10] J. Bates *et al.*, *The role of emotion in believable agents,* Communications of the ACM **37**, 122 (1994).

[11] J. Veldman, *De 36 dramatische situaties.*

[12] M. O. Riedl and R. M. Young, *Narrative planning: Balancing plot and character,* Journal of Artificial Intelligence Research **39**, 217 (2010).

[13] I. M. T. Swartjes and M. Theune, *The virtual storyteller: story generation by simulation,* in *Proceedings of the Twentieth Belgian-Netherlands Conference on Artificial Intelligence, BNAIC 2008, Enschede* (University of Twente, Enschede, 2008) pp. 257–265.

[14] R. Aylett, S. Louchart, J. Dias, A. Paiva, M. Vala, S. Woods, and L. Hall, *Unscripted narrative for affectively driven characters,* IEEE Computer Graphics and Applications **26**, 42 (2006).

[15] A. Ortony, *The cognitive structure of emotions* (Cambridge university press, 1990).

[16] S. Klein, J. Oakley, R. Lao, C. Court, M. Foster, S. Converse, J. Aeschlimann, D. Balsiger, and J. Smith, *Automatic novel writing: A status report,* (1973).

[17] D. S. Weld, *An introduction to least commitment planning,* AI magazine **15**, 27 (1994).

[18] M. Theune, S. Faas, D. K. J. Heylen, and A. Nijholt, *The virtual storyteller: Story creation by intelligent agents,* in *TIDSE 2003: Technologies for Interactive Digital Storytelling and Entertainment, Darmstadt, Germany*, edited by S. Göbel, N. Braun, U. Spierling, J. Dechau, and H. Diener (Fraunhofer IRB Verlag, Darmstadt, 2003) pp. 204–215.

[19] N. Dehn, *Story generation after tale-spin.* in *IJCAI*, Vol. 81 (Citeseer, 1981) pp. 16–18.

[20] S. G. Ware and R. M. Young, *Cpocl: A narrative planner supporting conflict.* in *AIIDE* (2011).

[21] H. Zhao, *Emotion in interactive storytelling.* in *FDG* (2013) pp. 183–189.

[22] P. Bleackley, S. Hyniewskab, R. Niewiadomski, C. Pelachaud, and M. Price, *Emotional interactive storyteller system,* in *Proc. International Conference on Kansei Engineering and Emotion Research, Paris, France* (2010).

[23] D. Pizzi and M. Cavazza, *Affective storytelling based on characters' feelings,* in *Intelligent Narrative Technologies: Papers from the AAAI Fall Symposium* (2007) pp. 111–118.

[24] D. L. Roberts, H. Narayanan, and C. L. Isbell, *Learning to influence emotional responses for interactive storytelling.* in *AAAI Spring Symposium: Intelligent Narrative Technologies II* (2009) pp. 95–102.

[25] K. J. Blom and S. Beckhaus, *Emotional storytelling,* in *IEEE Virtual Reality Conference, Workshop "Virtuality Structure"* (2005) pp. 23–27.

[26] M. Cavazza, D. Pizzi, F. Charles, T. Vogt, and E. André, *Emotional input for character-based interactive storytelling,* in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1* (International Foundation for Autonomous Agents and Multiagent Systems, 2009) pp. 313–320.

[27] M. O. Riedl and R. M. Young, *An objective character believability evaluation procedure for multi-agent story generation systems,* in *Intelligent Virtual Agents* (Springer, 2005) pp. 278–291.

[28] A. Popescu, J. Broekens, and M. van Someren, *GAMYGDALA: An Emotion Engine for Games,* IEEE Transactions on Affective Computing **5**, 32 (2014).

[29] S. Marsella, J. Gratch, and P. Petta, *Computational models of emotion,* A Blueprint for Affective Computing-A sourcebook and manual , 21 (2010).

[30] C. D. Elliott, *The Affective Reasoner: A Process Model of Emotions in a Multi-agent System*, Ph.D. thesis, Evanston, IL, USA (1992), uMI Order No. GAX92-29901.

[31] W. S. Reilly, *Believable Social and Emotional Agents.*, Tech. Rep. (DTIC Document, 1996).

[32] R. Pérez y Pérez, *MEXICA: a computer model of creativity in writing.*, Ph.D. thesis, University of Sussex (1999).

[33] P. Gervás, *Story generator algorithms,* The Living Handbook of Narratology. Hamburg: Hamburg University Press. http://hup. sub. uni-hamburg. de/lhn/index. php (2012).

[34] J. McCoy, M. Treanor, B. Samuel, M. Mateas, and N. Wardrip-Fruin, *Prom week: social physics as gameplay,* in *Proceedings of the 6th International Conference on Foundations of Digital Games* (ACM, 2011) pp. 319–321.

[35] J. McCoy, M. Treanor, B. Samuel, B. Tearse, M. Mateas, and N. Wardrip-Fruin, *Authoring game-based interactive narrative using social games and comme il faut,* in *Proceedings of the 4th International Conference & Festival of the Electronic Literature Organization: Archive & Innovate* (2010).

[36] M. Mateas and A. Stern, *Façade: An experiment in building a fully-realized interactive drama,* in *Game Developers Conference*, Vol. 2 (2003).

[37] R. Aylett, J. Dias, and A. Paiva, *An affectively driven planner for synthetic characters.* in *Icaps* (2006) pp. 2–10.

[38] S. Rank, S. Hoffmann, H.-G. Struck, U. Spierling, and P. Petta, *Creativity in configuring affective agents for interactive storytelling,* in *International conference on computational creativity* (2012) p. 165.

[39] I. Swartjes and M. Theune, *A fabula model for emergent narrative,* in *Technologies for Interactive Digital Storytelling and Entertainment* (Springer, 2006) pp. 49–60.

[40] I. Swartjes, *Using narrative cases to author interactive story content,* in *Entertainment Computing–ICEC 2007* (Springer, 2007) pp. 205–210.

[41] M. Theune, S. Rensen, R. op den Akker, D. Heylen, and A. Nijholt, *Emotional characters for automatic plot creation,* in *Technologies for Interactive Digital Storytelling and Entertainment* (Springer, 2004) pp. 95–100.

[42] S. Bringsjord and D. Ferrucci, *Artificial intelligence and literary creativity: Inside the mind of brutus, a storytelling machine* (Psychology Press, 1999).

[43] R. M. Young, *Story and discourse: A bipartite model of narrative generation in virtual worlds,* Interaction Studies **8**, 177 (2007).

[44] R. P. y Pérez and M. Sharples, *Three computer-based models of storytelling: Brutus, minstrel and mexica,* Knowledge-based systems **17**, 15 (2004).

[45] M. Schröder, P. Baggia, F. Burkhardt, C. Pelachaud, C. Peter, and E. Zovato, *Emotionml–an upcoming standard for representing emotions and related states,* in *Affective Computing and Intelligent Interaction* (Springer, 2011) pp. 316–325.