Human Media Interaction

University of Twente
Enschede - The Netherlands

University of Twente
Department of Electrical Engineering,
Mathematics & Computer Science
Faculty of Human Media Interaction

The Plot Thickens

bringing structure and meaning into automated story generation

by

Ivo Swartjes

March, 2006

Examination Committee:	dr. M. Theune
	prof.dr.ir. A. Nijholt
	dr. D.K.J. Heylen

Voor mama
wou dat je er bij kon zijn

Abstract

The Virtual Storyteller is a Multi Agent System (MAS) that can generate stories by simulating a virtual world in which Character Agents pursue their goals. The claim is made that the story emerges from the events in the virtual world.

The goal of this project has been to make the stories more interesting. This thesis describes how to extend the Virtual Storyteller with a Plot Agent that will direct the events to generate more interesting stories. For this, three research questions have been tackled. Firstly, the system needed a better understanding of the emerging story. For this, a fabula structure has been designed that captures causality between story elements such as emotions, goals, actions, events and perceptions. A plot within the fabula revolves around a single goal and contains everything that is affiliated to that goal.

Secondly, plot control techniques have been explored where attention has been paid to the fact that believable characters should be kept autonomous. Four ways to influence the story are: (1) generating events to mediate the plans of characters, (2) influencing their perceptions, (3) changing the setting, (4) suggesting goals or actions.

Thirdly, an exploration has been made as to what decisions the Plot Agent needs to make to direct the story, and to aid these decisions, a creative problem solver has been designed that uses case based reasoning to solve problems.

Preface

"How on earth can a computer *invent* a story?" is a question that numerous people have asked me the past year and it has never failed to put a smile on my face.

The story started four years ago when I stumbled upon automated story generation by coincidence when I read about the Virtual Storyteller. The challenges of this system have fascinated me ever since. Not thinking much further of it, it suddenly came back on my path during my internship. I overheard Mariët Theune talking to somebody about the Virtual Storyteller, and one thing led to another. Cause and effect. This was right before a meeting where the future of the Virtual Storyteller was discussed with Katri Oinonen who wanted to do Ph.D. research on the subject. I attended this meeting and became very enthusiastic. My fate was sealed: I was going to finish my master's degree on the subject of generating stories.

I have had a lot of fun trying to tackle the problem of coming to a form of plot control that makes the stories that are generated by the Virtual Storyteller more interesting. At the moment, five people are doing a Master's thesis on different parts of the subject, including the design of a virtual world, narration of the generated story, adding suspense to the story and generating speech that can tell the story in an engaging way. The plot sure thickens for the Virtual Storyteller.

After a year of research, the journey comes to an end. I am proud to present the results of my research and would like to thank all the people who have made this possible. I want to thank Mariët Theune for our weekly meetings discussing the progress of my work, which was never without a nice cup of tea to accompany us. Mariët showed a lot of faith in me being on the right track. My thanks also goes to Anton Nijholt and Dirk Heylen for their feedback on my ideas and my writing. I want to thank Katri Oinonen for being very ambitious in a way that has been very catching for me. Katri has also instigated a big boost for the Virtual Storyteller as a whole. I want to thank Jasper Uijlings, Nanda Slabbers and Douwe Terluin for the very successful cooperation we had the past year. And of course, I want to thank Joost Vromen and Niels Bloom for making a big contribution to the subject matter of creative problem solving that I have been working on. Finally, I would like to thank the people that have made a contribution to my personal story during this period. My family and friends, who enjoyed nice times with me but have also helped me through some difficult times. In particular I want to thank two very special friends, Bob and Stuart, who have always supported me and cheered me on. That means a lot to me.

Contents

Chapter 1

Introduction

"Storytelling reveals meaning without committing the error of defining it."

— Hannah Arendt (1906 - 1975),
German-American political scientist

"Let's do just that."

— Ivo Swartjes, MSc. graduation student

Human beings are capable of constructing and telling a wide variety of stories. Even children learn to do this to a certain extent at a very early age. Can computer systems do the same? An interesting question with no clear answer. Today still, the tale of automated story generation is one with an open end.

This makes automated story generation a very interesting and challenging subject that combines Narratology with Artificial Intelligence, Psychology and Computer Science. I will explore story generation techniques by contributing to the development of the Virtual Storyteller, a story generation environment where characters inhabit a virtual world, and chase their goals and plans in this world. The assumption is that these goals and plans can be expressed in the form of a story. Aylett (1999) coins the phrase *emergent narrative* to indicate such approaches: the narrative emerges according to what happens in the world.

In order to take the Virtual Storyteller to a next level of more engaging story generation, a form of plot control needs to be introduced in order to prevent the story from becoming a mere simulation of the lives of characters. Characters in the story have a high level of autonomy and control and are not supposed to know that they are playing in a story. Still, the events need to be steered in such a way that a meaningful plot emerges. Tragically however, the plans for an interesting story are sometimes different from the plans for a happy life for the characters in it.

This chapter will describe the context of automated story generation and earlier work done on this area resulting in The Virtual Storyteller. After that, research questions will be formulated.

1.1 Applications of automated story generation

Research into automated story generation can play an important role in applications for entertainment, training and education (Riedl, 2002; Fairclough, 2004). It is clear that stories have been an important form of entertainment for centuries, but up till now the medium for storytelling has been quite static. A traditional book tells the same story every time you read it. With the rising of computers and the development of games, techniques for a whole new form of storytelling are in development. This new form is called interactive storytelling. Many computer games present fixed story lines which are followed by the player. These story lines tend to be pre-scripted in the form of a linear story (linear narrative), sometimes containing a certain number of branching alternatives (branching narrative) (Riedl, 2004). Interactive stories have this same kind of branching narrative. They are different from traditional stories in the sense that they offer the reader choices in the unfolding of the story. A good example is formed by the Choose Your Own Adventure books that contain little fragments of story after which the reader has to make a decision that will lead him to other fragments of the story scattered throughout the book. However, the more interactivity is brought into play, the more theoretically possible stories have to be worked out. This presents challenges for the future because it soon becomes infeasible to write out all possible stories. Authoring might shift to a higher level where it will be done with the aid of automated generation of content. When automated story generation techniques become mature enough, they can be brought into play to automate or at least aid the development of interactive stories. A good example of this is the Façade system (Mateas & Stern, 2003) where an interactive story between two characters and a human player unfolds, a story that can be different every time because it offers a huge amount of freedom for the player.

Several movies portray examples of interactive stories where the participants have total freedom. The Matrix (1999) is a movie that shows a future where human civilization is taken over by artificial intelligence. To power intelligent machines, all human beings are used as biological batteries. Their perceptions are coupled to an immense simulated world where the immersion is so total that the people in it are not aware that they are living in a simulation. A good example of a similar kind of immersion is the Holodeck which is so gratefully used by the characters playing in the television series Star Trek.

Games have a lot in common with interactive stories. Although scenarios like the Matrix and the Holodeck are still very far in the future, it is interesting to see that the computer games industry seems to be heading in this direction, towards more reality and more immersion. Games are more and more taking on the form of a story world in which the player immerses to try and fulfill certain goals. Games like The Sims 2 and World of Warcraft show us that people like to play games in which they are having story-like experiences. These stories originate solely from the interaction that human players have together, which is a form of emergent narrative.

Researchers have come to realize that narrative is the primary mechanism through which we interpret the experiences of our lives (Riedl, 2002). It makes sense that we then also learn and remember through narrative and that computers can provide environments for narrative learning. The ability to present narrative environments that are tailored to the user instead of relying on pre-scripted sequences seems to hold great potential. Van Gils (2005) identifies application areas for education which includes the construction of children's stories and providing training and immersion in virtual scenarios or story environments. There are several advantages above traditional forms of education. More variation can be offered, it is possible to tailor the experience to the user and it can be made more compelling than traditional study books. A trend emerges to develop "serious gaming" applications where the students are trained to make the right decisions in situations that are simulated in a virtual environment.

In fact, games and stories can be seen as a form of education. (Aristotle, trans. 1907) states that poetry is in its essence a way of imitating man and its actions. He also identifies learning as being the biggest kind of entertainment. Reading about man being imitated is pleasurable because the reader learns about mankind. Indeed, in stories, there's always an aspect of characters trying to reach goals by solving problems (Brewer & Lichtenstein, 1981). And I think this learning aspect can be seen in games too, even as simple as learning how to skillfully dodge a monster, or store blocks away efficiently. Games pose problems that have to be solved in the same way that educational challenges do.

1.2 The Virtual Storyteller

The Virtual Storyteller, previously designed and implemented by graduate students at the University of Twente, is a Multi Agent System where virtual characters live in a logic-based virtual story world (Theune, Faas, Nijholt, & Heylen, 2003). The event sequence of the characters is the basis for further processing by a Narrator to generate narrative (Hielkema, Theune, & Hendriks, 2005). The approach is largely inspired by Improvisational Theater and a storytelling system called *Typewriter* that uses a similar approach. See Faas (2002, pp. 14–15) for a more detailed description of Typewriter.

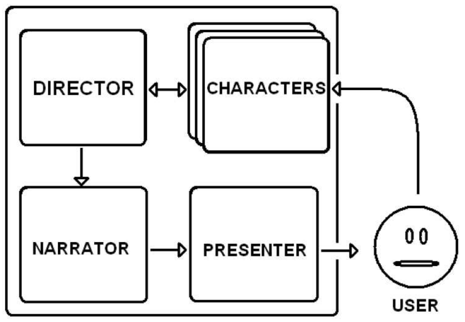

Figure 1.1: *Virtual Storyteller architecture*

Figure 1.1 illustrates the architecture of the Virtual Storyteller. Each character in the story is controlled by a semi-autonomous Character Agent. Such a Character Agent has its own emotions, beliefs and goals within the context of a virtual story world. A story emerges by narrating the events in this world, which currently consist of the emotions and the actions of the characters. There is a Director Agent that has the role of directing the story. In the current implementation this story direction is done by giving the characters goals that –

Once upon a time there was a princess. Her name was Amalia. She was in the little forest. Once upon a time there was a vilain. His name was Brutus. The vilain was in the fields. There is a sword in the mountains. There is a sword in the big forest.

Amalia walks to the desert. Brutus walks to the desert. Amalia experiences fear with respect to Brutus due to the following action: Amalia sees Brutus. Amalia walks to the plains. Brutus walks to the plains. Amalia experiences fear with respect to Brutus due to the following action: Amalia sees Brutus. Amalia walks to the mountains. Brutus walks to the mountains. Amalia picks up the sword. Brutus experiences fear with respect to Amalia due to the following action: Amalia picks up the sword. Brutus kicks the human. Amalia stabs the human. And she lived happily ever after!

Figure 1.2: *The story of Brutus and Amalia (translated into English)*

when reached – mark the end of an episode, and by prohibiting the characters to perform certain actions at certain moments.

Stories that the Virtual Storyteller is able to generate are situated in a simple setting with a couple of adjacent locations. Examples are generated that revolve around a princess called Amalia, and a villain called Brutus. Both characters have an episodic goal; it is Brutus' goal to imprison Amalia, and Amalia's goal to kill Brutus. See Figure 1.2 for an example output.

1.3 Research questions

My work will continue on the Virtual Storyteller approach. I will focus on designing a Plot Agent that replaces the original Director Agent and is responsible for the plot development of the stories generated by the Virtual Storyteller. I will argue that such a Plot Agent should:

- understand the emerging story;
- have ways to control the emerging story;
- have creative decision making abilities for solving story generation problems.

In this thesis I will approach these requirements by coming to a formalization of a structure which gives causal and temporal information about the events behind the text. On top of this, two areas necessary for more interesting stories will be explored: (1) ways to control a plot brought about by characters that are supposed to be autonomous and (2) ways to apply creativity to make decisions about the way the story should unfold given the restrictions of the possible plot control. Section 2.4 will discuss the approach used in more detail.

Chapter 7 will situate this Plot Agent in a modified architecture for the Virtual Storyteller.

1.3.1 Understanding the emerging story

If we look at the story in Figure 1.2, it is clear that the textual representation leaves much to desire. Work on syntactic aggregation (Hielkema et al., 2005) allows for better formulation than displayed here, but has not yet been integrated with the rest of the Virtual Storyteller. However, no matter how well formulated the textual representation becomes, the story would still lack a certain coherence or story line which is not something we can ascribe to an unfinished narration component. The story would be much more interesting if the system could *understand* the story it generated. Let's look at what the story in Figure 1.2 actually is about. It describes how Brutus and Amalia meet each other by coincidence. Then, because Amalia is afraid of Brutus, she flees until she stumbles upon a sword and when she picks that up, the roles suddenly turn! It becomes clear that for such a description to be made, a sense of plot or 'storiness' is very important. The current version of the Virtual Storyteller has no such concept of storiness. All cause and effect of the events exists only implicitly within the Character Agents and are left for the reader to infer when reading the action sequence.

It is very important to understand what this storiness entails. Without this information the task of plot direction is very difficult if not impossible. A plot structure needs to be designed that can be used to capture the events and reason about the development of the plot. A sense of plot is also required for molding the story into a well-structured and interesting textual whole. When the Narrator is able to understand the story, it can then relate the events to a greater plot context, raising and answering reader questions as to why things happen, and what is going to happen next. Chronologically summing up what happens achieves this, albeit in a very straightforward and flat way that has great potential to be boring.

An important part of my research will involve bringing a plot structure into play. It's important to make sure this structure can be generated, reasoned about and that it can be narrated. This leads to the first research question.

RQ1 How can a plot structure be designed that can be generated, narrated and reasoned about?

After Chapter 2 which discusses the form and content of stories, this question can be further specified. Chapter 4 will then provide a solution in the form of a design of such a structure for use in the Virtual Storyteller.

1.3.2 Controlling the emerging story

When the Virtual Storyteller has a concept of story and plot at its disposal, the focus can then shift towards the question how to make the stories that are generated more interesting.

Rensen (2004) recommends expanding the story world and expanding the capabilities of the characters to generate more interesting stories. This is a reasonable extension that has been pursued by Uijlings (2006); the expressiveness of the story world and the actions that are possible within this world is increased. Chapter 3 will describe this work on an explanatory basis.

However, experiments learn that free improvisation by characters in a virtual world is not enough to create interesting stories. The goals that the characters in a story have are different from the goals that the author has for the story, and when stories are based only on the character goals, it leads to narratives without any concept of plot (Mateas & Stern, 2000; Szilas, 2003). Making a huge story world with a lot of characters in it will definitely add to the number of different stories that can be generated, but it will not greatly extend the power and possibilities of the Virtual Storyteller to tell interesting stories.

For interesting stories we need to pursue what I will call *plot goals*, which are goals of generating a more interesting plot. Apart from understanding what these plot goals are like, in other words, what makes a story interesting, it is important to look at the ways in which a Plot Agent can direct the events and the characters in order to reach these plot goals without the characters losing their believability. Techniques of plot control in emerging narrative will be explored to see how we can do this. This forms the second research question.

RQ2 Which techniques can be used to control the plot whilst keeping the characters believable?

Chapter 5 will identify techniques used in other approaches, and proposes plot control techniques for use in the Virtual Storyteller. This thesis will not provide an implementation of these techniques or a specification of a component that *determines* the plot goals, but identifying these techniques is an important start and is useful for specifying the interaction between the Plot Agent and the Character Agents. This interaction will be described in Chapter 7 where a modified architecture for the Virtual Storyteller will be discussed.

1.3.3 Making creative decisions

Chapter 2 will discuss the ingredients of interesting stories. Situations that are interesting for a story can be brought about using the aforementioned plot control techniques. These situations can be described in the form of plot goals. The third research question is based on the assumption that such plot goals exist; the question is how the Plot Agent can decide how to employ the identified plot control techniques to reach these plot goals:

RQ3 How can the Plot Agent use the plot control techniques to reach plot goals?

Chapter 6 will discuss such plot control decisions. It will also come to a design of a creative problem solver that can be used to make these kinds of decisions.

Chapter 2

Theoretical background

"Did you know that the first Matrix was designed to be a perfect human world? Where none suffered, where everyone would be happy. It was a disaster. No one would accept the program. Entire crops were lost. Some believed we lacked the programming language to describe your perfect world. But I believe that, as a species, human beings define their reality through suffering and misery. The perfect world was a dream that your primitive cerebrum kept trying to wake up from. Which is why the Matrix was redesigned to this: the peak of your civilization."

— Agent Smith, 'The Matrix' (1999)

In this chapter I will explore the theoretical background on which I base my research. First I will investigate what actually constitutes a story by investigating the narrative structures in them from a story analysis perspective, and by identifying the difference between content, form and meaning of a story. Second, I will shed a light on the aspects that make a story interesting and emotionally pleasing. Third, I will investigate how automated story generation has been approached in the past. Finally, based on the theoretical background I will further specify the first research question.

2.1 Story structures

In order to achieve any satisfactory result in the generation of stories by computers, it's important to understand what actually constitutes a story. At first glance it is tempting to say that a story is a narration of events, an entertaining description of what happens in a fictive or real world. But there is much more to a story than meets the eye. Much research has been done in the field of story analysis, to try and capture the structure and meaning of stories. It is good to explore these concepts as building blocks for automated story generation.

2.1.1 Grammar approach

First attempts into story analysis have revolved around the idea to approach a story just like one would approach a sentence. It was believed that story grammars can be developed that capture a story and a big interest in grammatical approaches like that, was stirred by the work of Vladimir Propp (Propp, 1968/1997). Propp investigated the structure of Russian folk tales and was able to identify a finite set of elements that occurred in every Russian folk tale in a finite number of possible sequences. This fed the idea that it may indeed be possible to approach a story grammatically, just like a sentence.

Mandler and Johnson (1977); Johnson and Mandler (1980) have investigated such a grammatical approach for the structures of simple stories[1]. They identify the episode as a major unit within a story; the grammar of a simple story can be seen as a tree structure of episodes and other story elements connected by AND, THEN and CAUSE relationships. Episodes can be embedded in other episodes and there's a "matrix episode" overlapping all the other episodes.

Obviously, even though grammatical approaches like this look promising, they heavily rely on the textual form of a story. Wilensky (1983) argued that any approach to story analysis that is purely based on grammars is unsatisfactory exactly for this reason. Such an approach is unable to capture an important element of stories, namely the mental or conceptual structures at the basis of them (plans, goals, etc.) His most important argument is that there exist nonlinguistic stories, such as picture stories or cartoons. As we also know from the fact that many books have also been filmed, the actual story content must somehow be independent from its grammatical surface form.

2.1.2 Three layers of a story

Indeed, more recent literature elaborates on this discrepancy between surface form and content. According to Riedl (2002), narrative theorists decompose narrative into three layers:

Fabula layer: the sequence of events that take place in the story world. It consists of the actual events of the story. It's the essential base from which narrative arises and the layer at which concepts such as causality exist

Story layer: a filtering of the fabula by selecting characters and viewpoint through which to expose parts of the fabula (focalization). It constitutes a narrative structure by organizing the fabula in episodes and relevance of actions.

Text layer: the specific wording and phraseology chosen to tell the story with. A story is told by the author to the reader, and the way this is done constitutes a discourse structure. It also reflects the norms and values of the author, saying or suggesting which actions are good and bad, which is a very important part of a story (Szilas, 2003).

Clearly, the fabula layer defines the content of the story whereas the story and text layer determine the meaning and the structure of it. The meaning forms the rationale of the story. Stories can be seen as a subset of action discourses that contain a 'point' (Van Dijk, 1980). Wilensky (1983) distinguishes external

[1]The term 'simple' in this context has nothing to do with the length, number of events or episodes in the story, but refers to the fact that the story has a single protagonist in each episode.

points (to convey a message, an advice to the reader) from internal points (to evoke emotions).

Stories are usually very structured. There is a beginning, a middle part and an ending, the story is divided in episodes and the story as a whole somehow feels complete. This is an important aspect of good stories that Aristotle (trans. 1907) describes when he talks about unity and totality of plot. He states that a story revolves around one action and nothing in the story can be removed or displaced without disturbing the whole. Sgouros (1998) has used unity and totality of plot as a paradigm to guide automated story generation. He formalizes them as follows:

Unity of plot means that there is only one plot sequence, or if there are more, one is clearly dominant. This dominant plot sequence can be based on one goal around which the story develops, called the *Storyline Goal*.

Totality of plot means that everything that happens is related to the plot sequence, in other words: all the unnecessary is omitted.

Insight into the structure of stories has been greatly complemented by research into the psychological mechanisms facilitating memory and recall of stories. The structure of stories is almost certainly based on these psychological mechanisms (Johnson & Mandler, 1980). Folktales originated and then survived by oral tradition; the evolution due to the telling and retelling of these stories must reflect some organizational tendency of the information-processing system through which it passed (Mandler & Johnson, 1977). Research has supported this by showing that the “higher” structures in a story are more easily recalled, which would indeed suggest a hierarchical organizational structure that helps to memorize and recall a story (Brewer & Lichtenstein, 1981).

These underlying story structures also become evident in developmental psychology research. Such structures can be identified in the development of story comprehension by children. Young children view a story as a sequence of isolated states and actions, whereas at a later age, a temporal ordering can be identified. This can be witnessed when these children tell about their experiences: “and then this happened and then that happened and then...”. Only later do children organize stories into episodes that contain goals, actions and outcomes, developing into a full hierarchical ordering (Trabasso, Van den Broek, & Suh, 1989; Trabasso & Nickels, 1992). Based on this, Trabasso et al. (1989) identify a causal network of setting, goal, attempt and outcome in stories connected by enabling, psychological, motivational or physical relationships. Goals set expectations, outcomes affirm or deny these. The combination goal-attempt-outcome forms an episode. Tapiero, Van den Broek, and Quintana (2002) have researched the strength of these connections in terms of causal importance as perceived by readers.

2.2 Emotions in stories

Having identified the structures that make up a story, we can now look at what makes a story interesting: the emotions that arise, why they arise, and how they can be brought about.

2.2.1 Types of emotions

A good story captures and keeps the reader's attention. Emotions play an important role in achieving that (Szilas, 2003). For the construction of a story, the author will use an implicit reader model to manage the reader's reaction to the story when he imagines what emotions and questions the reader has throughout the story (Szilas, 1999). Oatley (1994) identifies the following taxonomy of emotions that arise in readers when they read stories:

- external emotions, which are emotions about the story itself, e.g. curiosity about what will happen next, or a difference between what the reader expects and what is presented;
- internal emotions, which are emotions that result from the reader engaging in the story world, e.g. sympathy for the characters, (personal) emotion memories and emotions due to identification.

A story focusing on external emotions organizes its events in a way to achieve suspense, surprise or curiosity, followed by a resolution of it. Although external emotions are very important and relevant for story generation, they are more concerned with the presentation aspect of storytelling and will therefore not be looked at in more detail here.

Internal emotions relate to the actual events in the story. The intensity of such emotional reactions can be surprising, knowing that it's 'just' a story apparently doesn't guarantee a detached observation of what's going to happen next. Everyone who has ever read a good horror book knows how much stress and anxiety an innocent piece of written work can achieve in the reader.

Which internal emotions are experienced in narrative depends on the role of the person experiencing it. Kelso, Weyhrauch, and Bates (1992) have done experiments where human actors played in an emerging story and were directed by a human director. Their conclusion is that there is a difference between what participants in a story experience and what spectators experience:

- Participants found the experience more powerful, easily causing immediate, personal emotions, in stead of feelings of empathy for the other characters, that the spectators experienced;
- Participants found behavioral inconsistency of others acceptable, where spectators found these disturbing;
- Experience of time is different for participants; it may be so that participants base their sense of time on the actions that go through their mind, whereas readers base it on the actions as they happen in the story.

For interactive drama in which the emotional experience is focused around a human player, the actions of characters can more easily be motivated by narrative constraints, instead of personal constraints brought about by emotional, psychological or social reasoning. Credibility is enough. This is different for 'normal' drama where much more credibility of the characters in the story is required (Szilas, 1999, 2003).

I will discuss two mechanisms causing internal emotions, namely identification and empathy.

2.2.2 Causes of emotions: identification and empathy

Important internal emotions are those caused by *identification.* When a reader identifies with a liked character in the story, he takes on the emotions and goals of the character as if they are his own. People read stories as if they have to solve the character's problems, making expectations for future events. This makes suspense possible: the fewer solutions the reader can think of, the more suspense is evoked (Young, 2002). Oatley (1994) identifies the elements necessary for this identification to take place, among which:

- Taking on the goals of a protagonist. A plot is the working out of the plans that are supposed to reach these goals in the story world;
- Making a mental model of an imaginary world, in other words mentally (re-) constructing the fabula behind the story;
- Speech acts to the reader that help restructure the story. For instance: "little did Lovely know what was going to happen to her beloved lamb." Speech acts like this are often used to create suspense.

Perhaps more frequently occurring and more credible in stories is the mechanism of *empathy*; readers are in a sense observers. Children display intervention attempts towards liked characters ("Watch out! Look behind you!") that cannot be explained by identification (Zillmann, 1994). Readers simulate the goals and plans, etc. of the characters in a story, and then experience emotions due to the succeeding or failing of them. These are not the same emotions as those of the characters (identification), but their own emotions of sympathy or remembrance of personal situations (Oatley, 1994).

Zillmann (1994) states that affective disposition towards characters in the story and empathy are related: the more positive the affective disposition, the stronger empathy becomes. Counter-empathy is also possible, caused by a negative affective disposition. One way or the other, readers must be made to care about characters either in a positive or negative way so that they are either friends or enemies. The more developed this affective disposition by dramatic events is, the stronger the emotional involvement with the dramatic presentation. If characters aren't developed, even very dramatic events can feel neutral. Judy was killed, so what?

According to Zillmann (1994), characters are not inherently good or bad: trait shifts are possible, depending on the actions of characters. A hero can become a villain. Indeed, an interesting aspect is that reversal of affective disposition heightens emotional involvement of the reader.

2.2.3 Drama: how to evoke these emotions

Now that we have identified the mechanisms that lead to emotional involvement with a story, we can look at the ingredients of an interesting story that captures the reader's attention.

Schank (cited in Wilensky, 1983) has made a classification of interest. He identifies a set of absolute interests (death, danger, power, sex etc). These concepts are not necessarily interesting in and of themselves, but stories about these concepts are more likely to be interesting than those that involve events like going to the movies. Schank also postulates some "interestingness operators" (such as unexpectedness, personal relatedness, etc). Finally, there is situational interest, where the interest arises from juxtaposition, as is the case in for instance irony and dilemmas.

Wilensky (1983) states that stories are often about human dramatic situations: a sequence of goal-related events that contains some problem for a character and usually also solution components. Problems could be provided by goal-subsumption termination (disruption of an equilibrium by disturbing the conditions for that equilibrium, i.e. losing one's job disrupts the equilibrium of having a job), goal conflict (two goals that cannot both be reached), goal competition (two characters striving to fulfill the same goal that can only be reached by one of them) or a difficult goal, where the character's skills are not up to the task or the goal is viewed as intrinsically problematic. Solutions that are difficult could be provided by "fortuitous circumstances", taking more desperate measures that are risky, or overcoming a personal limitation (like lack of courage). These points are all 'static' points. Dynamic points arise from violation of expectations. These could be the expectations of the characters (dramatic irony) or expectations of the reader (surprise or humor).

Sgouros (1998) uses four dramatic situations in a story:

Lifeline is a situation where a character gets a chance to improve his situation;

Rising complication happens when an already bad situation gets worse;

Reversal of fortune is a good situation that turns bad;

Dramatic irony happens when characters don't conform to social action, i.e. actions that are expected due to a character's personality and the social history that characters share. For instance, in the case of betrayal the irony lies in the fact that a favorable action is not returned.

Important is that the character has trouble fulfilling his goal. This trouble could for instance be in the form of a personal conflict, which Szilas (1999) considers to be the core of dramatic narrative. A personal conflict occurs when a possible action for a character is not compatible with his or her values, like an aristocratic lady not being able to admit her love for a poor servant. Real life is full of these kind of conflicts, which enables readers to identify with the hero in the story.

Summarizing, we could say that interesting stories relate about characters that readers can identify or sympathize with, about goals that these characters have, about difficult problems that arise in fulfilling these goals, and about possible solutions if the story has a happy end.

2.3 Previous approaches

Two categories of storytelling systems can be identified: systems that try to bring a plot into the events of a real interactive environment so that it is entertaining for a player being the protagonist (we will call this 'interactive drama'), and systems that try to produce a story that is entertaining for a reader (we will refer to this as 'automated story generation'). The focus of my research is on the latter.

The automated story generation systems that have been developed can be roughly divided in two categories (Riedl, 2002; Rawlings & Andrieu, 2003):

- Character-centric systems rely on autonomous characters to generate a story. Stories generated by these systems often have strong character believability, but are often not very strong at generating a coherent plot;
- Author-centric systems, which take the plot as focus and are often capable of creating a good plot and story line at the cost of character believability.

An extensive exploration of storytelling systems has been made in previous research by Faas (2002) and Rensen (2004). Some relevant examples will be briefly mentioned here to illustrate character-centric and author-centric approaches.

TALE-SPIN (Meehan, 1981) is a very famous character-centric approach. It is considered to be one of the first serious attempts at computer-generated stories. Just like the Virtual Storyteller, TALE-SPIN generates stories by simulating a virtual world with characters in it that try to reach their goals. The most important lesson learned from TALE-SPIN is the fact that when stories are driven solely by the goals of the characters in the story, uninteresting stories can arise. An example of such a story can be seen in Figure 2.1.

Another famous character-centric approach, focused on interactive drama, is the OZ project by Kelso et al. (1992). The interactive drama system created in this project is based on an experiment where several human characters play out a story guided by a human director, to investigate the possibilities of improvisation and plot control, and the emotional involvement of the players and the spectators.

Author-centric systems on the other hand often have a top-down approach, where a story emerges using some kind of story grammar or using author goals. Examples are MINSTREL (Turner, 1994) and the Actor Conference (ACONF) system (Riedl, 2002).

MINSTREL demonstrated an author-centric creative approach to storytelling to be quite successful. Turner considered the development of a story as a process no different from other forms of creative problem solving, and identified author goals that were solved using case-based reasoning (CBR). Chapter 6 will elaborate on this approach.

The ACONF system uses decompositional planning to assemble a sequence of actions, comprising the narrative. For every character in the story, an expert system is used that understands its assigned character and proposes actions based on its knowledge about the character. Several expert systems like this

ONCE UPON A TIME GEORGE ANT LIVED NEAR A PATCH OF GROUND. THERE WAS A NEST IN AN ASH TREE. WILMA BIRD LIVED IN THE NEST. THERE WAS SOME WATER IN A RIVER. WILMA KNEW THAT THE WATER WAS IN THE RIVER. GEORGE KNEW THAT THE WATER WAS IN THE RIVER. ONE DAY WILMA WAS VERY THIRSTY. WILMA WANTED TO GET NEAR SOME WATER. WILMA FLEW FROM HER NEST ACROSS A MEADOW THROUGH A VALLEY TO THE RIVER. WILMA DRANK THE WATER. WILMA WASN'T VERY THIRSTY ANY MORE.

Figure 2.1: *Narrative generated by* TALE-SPIN

collaborate on the unfolding of coherent narrative, heavily relying on author goals in the form of temporally ordered descriptions of states of the story world that should occur.

2.4 Conclusion

As we have seen in Section 2.1, a story - being a narrative - can be divided in three layers: a fabula layer, a story layer and a text layer. In this perspective, the Virtual Storyteller approach is in essence a character-centric approach that uses believable intelligent characters to generate a fabula layer and attempts to direct the events in such a way that that an interesting plot will emerge.

Interesting stories relate about characters that readers can identify or sympathize with, about goals that these characters have, about difficult problems that arise in fulfilling these goals, and about possible solutions. An approach like the Virtual Storyteller should focus on internal points, on evoking emotions in the reader and not on trying to illustrate a moral or advice to the reader, since an author-centric approach would be much more effective in that.

The approach explained in this thesis is based on the following assumptions about the context of my research:

- Believable characters can be developed. As discussed in Section 2.2.2, this is very important if we want to create an identification with and empathy towards the characters in the story. Work on believable characters can for instance elaborate on the approach of Theune, Rensen, op den Akker, Heylen, and Nijholt (2004).
- The internal state and plans of the characters can be uncovered. As long as the characters are software Agents, this is a fair assumption. For human players this uncovering will be considerably more difficult.
- It is possible to develop a Narrator that can transform the fabula and plot to text. Hielkema et al. (2005) have made some first steps to this end, Slabbers (2006) continues on this approach.

Based on these assumptions, the first research question can be clarified. The fabula layer needs to be captured so that the causal relationships between the story elements become clear, and plot sequences within this fabula layer should be determined to guide plot development. Chapter 4 will tackle this. The second and third research questions will be tackled in Chapters 5 and 6, respectively.

Chapter 3

A Virtual Story World

"This is a sparring program. Similar to the programmed reality of The Matrix. It has the same basic rules...rules like gravity. What you must learn is that these rules are no different than the rules of a computer system. Some of them can be bent...others can be broken."

— Morpheus, 'The Matrix' (1999)

Every story is situated in a certain environment that might be inhabited by characters and objects, governed by certain laws and relationships. Consider the story in Figure 3.1 about Princess Lovely wanting to get a friend. This story is situated in and around a castle with a castle tower, fields and forests around it. There are children in the story, a lamb, a princess and a king. To automate the story generation process, a model has been made which can be used to define such a 'story world' by expressing semantic knowledge about entities and their relationships. Such a story world can be used to generate fabula by introducing Character Agents that can pursue their goals and react to events.

This chapter will describe the design of this story world model. Although this design does not form a main subject of my thesis[1], an understanding of the story world design and the language used to describe it is of importance to the understanding of the subject matter discussed in the following chapters.

There are two aspects of the story world that can be distinguished:

Static world description A static, semantic description of the story world as it is at any given point in time. This description would include facts like: "There is a castle tower", "The children are playing outside", "Lovely is a princess" and "The king owns the castle";

World change description A description of how the world can change in time due to actions and events. This includes for instance a change of locations of things ("Lovely climbs up to the highest tower in the castle", "The sun goes down") or a change in mood ("Lovely becomes lonely").

[1]Most work is done by Jasper Uijlings as part of his Master's thesis, although decisions and ideas have been brought in by Katri Oinonen and me. Argumentation and more detail of the story world can be found in (Uijlings, 2006).

Princess Lovely is the daughter of the King Mikura. Mikura is the King of the Weatherland, and owns the castle, a big army, and all the fields, forests of the Weatherland. Lovely grows up safely and happy inside the protecting walls of the castle, until one day she climbs up to the highest tower in the castle where she is not allowed to go. From the window in the highest tower princess Lovely looks outside of the castle walls for the very first time. Lovely sees three children playing outside. The very important thing that Lovely does not see from the castle, is that the children are hungry. They do not have food, because all the fields and forests are owned by the King. Lovely never saw other children before, and was so fascinated about this that all day long she watched the children play.

When the sun goes down, the princess starts to realize how lonely she really is, and wants to be friends with the other children. She goes to her father King Mikura and asks if she can play with the other children. The king says it is not allowed, because Lovely is a princess and the children are her people. Lovely starts to cry and scream. She does not understand. But the King is hard on her, and even though he loves his only daughter he can not let the princess go to play with the children. Princess Lovely becomes very unhappy. The King cannot bear to see Lovely crying, and decides to give her a little lamb to be her friend. Lovely is very happy with the lamb. They play every day. The love princess feels for the lamb grows day by day. After a week they even sleep in the same bed.

Then, one morning when princess Lovely wakes up, she cannot find the little lamb. She is very worried and cries so loud that the King Mikura commands the whole army of the castle to look for the lamb. After few hour searching the staff realizes that the lamb has fled out of the castle from a door that one servant had forgot to close and shut down. When princess Lovely hears that, she runs out of the castle behind the army to help searching for her only friend. The princess runs faster that anyone else, because she loves the little lamb the most. Lovely is the first one to find the little lamb; slaughtered and hanging above the fireplace. The three children that Lovely wanted to play with had found the lamb, and decided to eat it since they were almost starving from hunger. But this all is too difficult for the little princess to understand, because she had enough food and possessions all her life, and she did not know what it is to be hungry. She was sad and hurt. She went silently back to her castle and she never again wanted to have friends or love in her whole life.

Figure 3.1: *Princess Lovely Story (written by Katri Oinonen)*

First I will discuss the design of a static model of the story world in the form of a two-layer ontology approach. Then, I will discuss how an ontology of world change has been designed.

3.1 Static world description

The static world description is a description of the state of every object and an ontology that describes the meaning of these objects. When I use the term 'ontology', I refer to a vocabulary that defines a theory about the world. The story world ontology, then, is a vocabulary of objects and their possible relationships in a story context. In an ontology for the story world, concepts like princesses, children, castles and forests need to be defined. This ontology is in the form of a classification of these concepts (i.e. a princess is a human being), a definition of possible relationships (i.e. castles have towers, human beings can be the parent of other human beings), and constraints like cardinality (i.e. a king owns at least one castle).

Our approach is to describe the story world ontology in two layers:

- An upper story world ontology that is independent of any writing rules, story structures, or story domains. The ontology describing this world model should be as powerful as possible, making a vast amount of actions and events possible. It is therefore by nature very functional;
- A domain-specific story world ontology that will use the upper story world ontology as a functional basis, and apply this functionality to a certain story domain.

An upper ontology is limited to concepts that are meta, generic, abstract or philosophical, and hence are general enough to address (at a high level) a broad range of domain areas. Kooijman (2004) suggests the use of SUMO (Suggested Upper Merged Ontology [SUMO], n.d.) as an upper ontology to capture the semantics of world knowledge. SUMO is widely used in all kinds of knowledge engineering applications. A big advantage of SUMO is that a mapping exists between concepts in SUMO and a natural language vocabulary in the lexical database WordNet (WordNet, n.d.). However, SUMO is too extensive for our purposes. We have therefore started constructing an upper story world ontology that adheres to SUMO by using the same terms and class hierarchy for concepts that were adopted. We call this upper ontology the Story World Core (SWC). The SWC ontology forms a basis upon which to build more domain-specific semantics. The SWC ontology is defined with functionality in mind; classes are distinguished by their functional differences in a story world. The actions and events that change the world state will only have an effect on properties defined in the SWC ontology. For example:

- The SWC ontology would for instance contain the information that in order to be able to wear an object using a `Dress` action, this object should be a subclass of a `WearableProduct`, which suggests the functional use of such an object as something to wear. When wearing such a product, it has a `wornBy` relationship to the wearer.

- In domain-specific ontology about fairy tales, we could define `Crown` as a subclass of `WearableProduct`, thus making it something to wear according to the SWC ontology. We could have a `Princess` wear such a crown. In a domain-specific ontology about a hospital environment, we could define `Surgeon` and `SurgicalMask` along exactly the same lines.

Globally, the SWC ontology defines and classifies concepts like objects, processes, regions etc., and the relationships that are possible between these concepts. Figure 3.2 shows a selection of the classification hierarchy. A domain-specific ontology still needs to be developed.

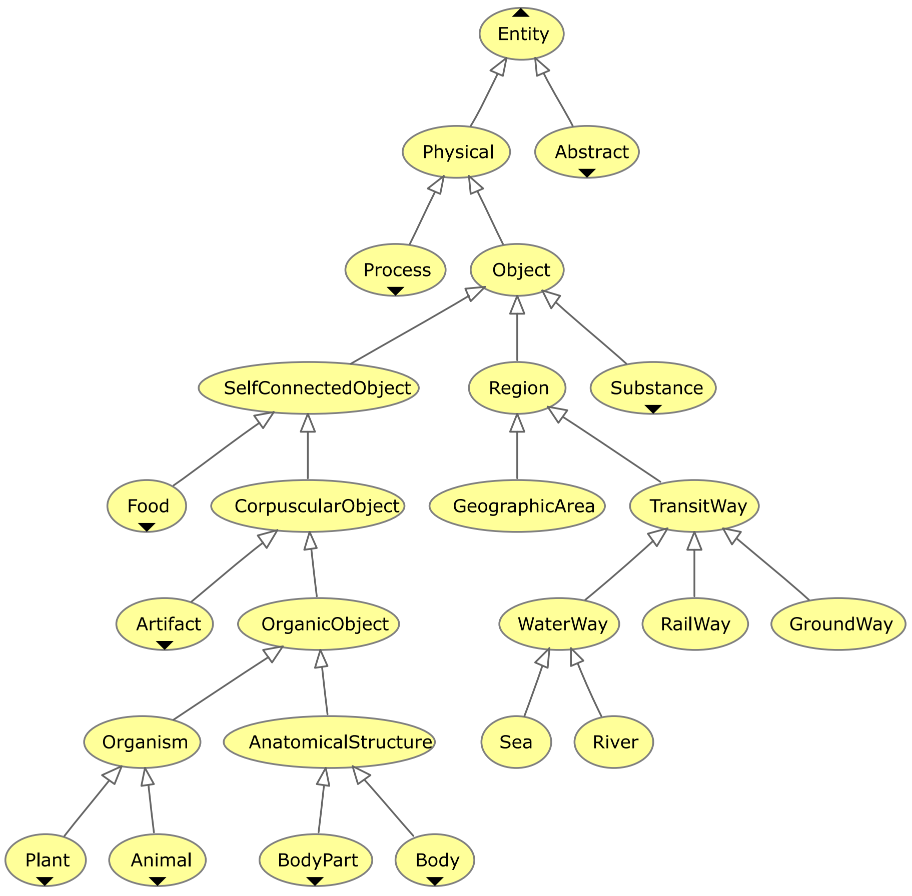

Figure 3.2: *Part of the Story World Core (SWC) ontology*

The most fundamental extension to the previous story world model described in (Faas, 2002) is the way in which the geography of the story world is defined. The previous model defines story world locations using the concept `locale`. Locales can be *adjacent* to each other and characters can move between adjacent locales. We extend this model by introducing a hierarchical topology and the concept of distance. Figure 3.3 sketches the way space is defined in our model as an undirected graph of locations and sub locations, connected by paths with a certain length. Locations, or as SUMO defines them, `GeographicAreas`, form a hierarchical topology, meaning that if an object is located in the tower, it is also located in the castle and also in the kingdom. The real specific location of an object is a location that has no sub-areas. Movement within any such lowest level location does not exist, so objects that are located in such a location are assumed to be equally distant from each other.

The paths or roads (SUMO: `TransitWays`) have a class hierarchy as well. They can be dirt roads, corridors, stairs, rivers, anything that forms a passable connection between two locations. The TransitWays between the GeographicAreas have a certain length. This length is relative and has no unit of measurement, although it makes sense to interpret this length as a geographical distance and not a distance in time. If a TransitWay is very steep, it would take more time to travel over it in one direction than it would take to travel over it in the other direction. The reason that TransitWays have a length is consistency; we want to prevent time-inconsistent stories where a character travels to another country and back to get a medicine, in the same time that his ill grandmother goes to the toilet and back. Section 3.2 will show how this can be avoided.

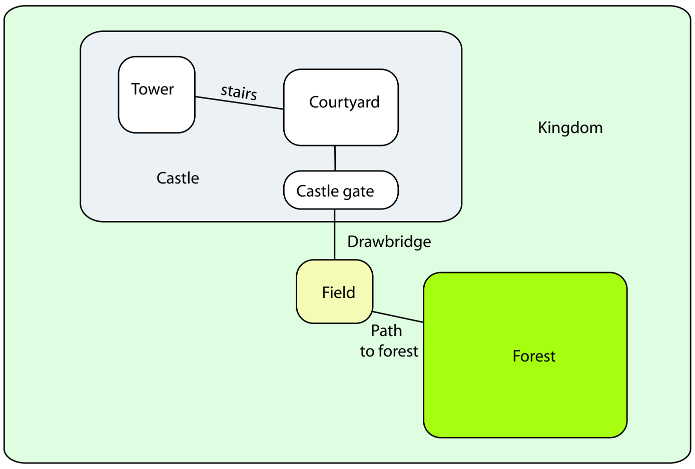

Figure 3.3: *Representation of GeographicAreas*

3.2 World change description

We consider any world state to be a snapshot of the world at a specific time. The world evolves through time steps, and the available world knowledge defines the world state on that one step in time. Actions change the world state to the next slice in time, as is done in Situation Calculus (Russell & Norvig, 1995, pp. 204–206).

To enable story dynamics where actions are executed realistically and can be interrupted, we have designed a temporal model of actions where actions have a certain duration, and an intermediate state of the world during execution. An action has three parts that define its meaning:

Preconditions define constraints on the world state that are necessary for the action to be executed. For instance, preconditions state that in order for

Princess Lovely to walk from the castle to the field, she must be located in the castle, and there must be a TransitWay from the castle to the field.

InterEffects specify the intermediate state change that occurs when the action is in execution. For instance, if Lovely is walking from the castle to the field, she is no longer in the castle (and obviously also not in the field yet). The InterEffects provide extra power to the action model but it is not necessary to define them for every action.

Effects define the world state changes that are applied upon successful execution of the action. Lovely has walked from the castle to the field and will now be located in the field.

Figure 3.4 shows the states that an action can be in, and the requirements for these states. When the action is started by a character, the preconditions must hold. If they do, the action enters an interruptible state. For instance, if Lovely performs an action of leaving the castle, the intermediate state can be interpreted as: Lovely 'is walking' away from the castle, but can still be stopped. The action can be perceived by other characters and the king could for instance quickly lock the door or tell Lovely to stop. Every action contains a specification how long this "interruptible duration" is. At the moment in time where the interruptible duration is over, the preconditions must still hold for the action to become successful. But if for instance the TransitWay to the field has been blocked in the mean time, the action fails. If the preconditions are still valid, the InterEffects are applied and the action is no longer interruptible. The duration of the whole action determines how long the action will be in this state. After the duration, the Effects are applied and the action has been executed successfully. In this way we can keep timing consistent since we can now state that a walk to another country will take a lot of time, whereas going to the toilet is done in an instant.

Uijlings (2006) has developed an ontology of actions for the Virtual Storyteller. This ontology defines primitive actions that are then inherited by more specific actions, forming a hierarchy of actions. Preconditions and (inter-) effects of actions will be inherited from their super actions so that for each action we only need to specify the additional preconditions and effects. Figure 3.5

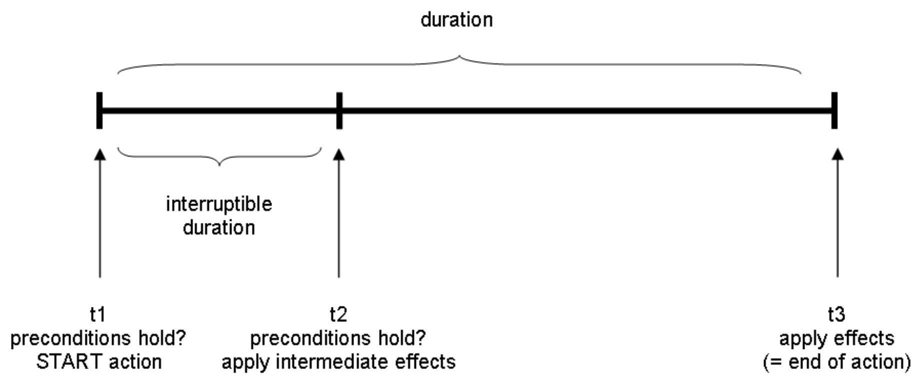

Figure 3.4: *Representation of an Action*

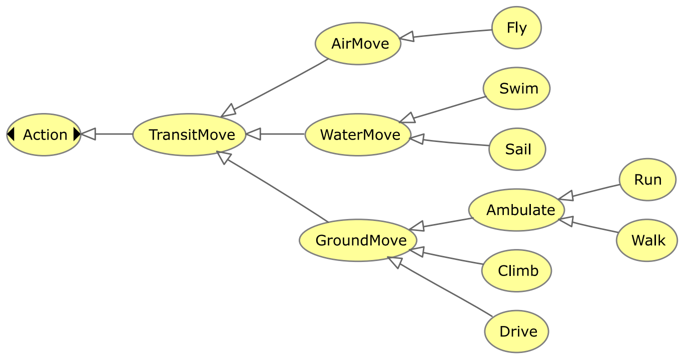

Figure 3.5: *Example from Action Ontology specifying location change actions*

shows a part of the Action ontology. It shows the actions that specify changing one's location. In a similar fashion, other primitive actions are defined for transferring objects (taking something, dressing, inserting etc.), manipulating (locking/unlocking, switching on/off etc.), creating (assembling/disassembling etc.), consuming, attaching (gluing, tying, detaching etc.), controlling and attacking (kicking, punching etc.).

In the same way, the construction of an event hierarchy is needed which can be done in much the same way. The difference is that events are not executed by a character (see Section 4.2.1), but the same model for Preconditions, InterEffects and Effects can be used. For events one can think of dying, falling, collapsing, raining, becoming night, waking up, pricking oneself, etc.

3.3 Knowledge representation

To model the story world, we use Protégé (Protégé, n.d.) to define the ontologies in OWL (Ontology Web Language [OWL], n.d.). OWL is a language recommended by the World Wide Web Consortium (W3C) meant as an extension to RDF. RDF (Resource Description Format [RDF], n.d.) is a format to describe resources, to represent information and to exchange knowledge over the internet using an XML syntax. All this information is in the form of `(predicate, subject, object)` triples as illustrated in Figure 3.6.

$$\begin{tikzcd} \boxed{\mathit{Subject}} \arrow[r, "\mathit{predicate}"] & \boxed{\mathit{Object}} \end{tikzcd}$$

Figure 3.6: *An RDF triple*

RDF can be used to express OWL knowledge. OWL is meant to provide additional vocabulary combined with a formal semantics to allow for a much greater semantic expressiveness and is used to publish and share ontologies, supporting advanced Web search, software Agents and knowledge management. OWL is becoming a standard language for the Semantic Web, which is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. RDF triples form

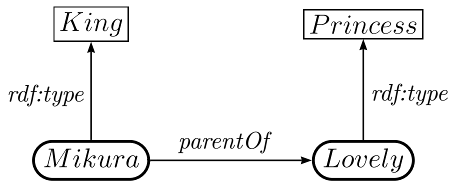

Figure 3.7: *Semantic network for "the King Mikura is a parent of the Princess Lovely."*

a graph or semantic network that defines concepts and relations between these concepts. Every triple defines a relationship[2] between two objects. The objects define the vertices of the graph and the relationships define the edges[3]. For instance:

```
(rdf:type Lovely Princess)
(rdf:type Mikura King)
(parentOf Mikura Lovely)
```

This example uses three RDF triples to express the knowledge that there is a Princess Lovely (Lovely is of the type Princess), a King Mikura, and this King is a parent of Lovely. The first argument is the subject of the relationship, i.e. Mikura, and the second argument defines the object, i.e. Lovely. So, Mikura has a parentOf relationship to Lovely. The graph representation of this knowledge can be seen in Figure 3.7. It is important to make a distinction between the actual knowledge (the graph and its triples), and the representation of it (the language in which these triples are expressed). The above example is represented in the knowledge representation language KIF, but the same knowledge could also be expressed in XML, as is often done when knowledge is shared in the Semantic Web:

```
<Princess rdf:ID="Lovely" />
<King rdf:ID="Mikura">
  <parentOf rdf:resource="Lovely" />
</King>
```

A concept can be prefixed by a namespace description, ensuring that it is clear which ontology defines the semantics of the concept. This namespace is in the form of a URI (Uniform Resource Identifier) indicating the ontology that

[2]In semantic networks, the terms 'predicate', 'relationship' and 'property' have a similar meaning.

[3]Explaining the workings of OWL and RDF in great detail falls beyond the scope of this thesis, however for clarity a very short introduction to semantic networks and in particular OWL will be given. A more extensive description can be found at `http://www.w3.org/TR/owl-features/`. Better insight in semantic networks can be found in (Brachman, McGuinness, Patel-Schneider, Resnick, & Borgida, 1991) and (Russell & Norvig, 1995, pp. 316–325).

describes the concept, but is often abbreviated into a prefix of a few characters. For instance, having the prefix `rdf` in front of the property `type` ensures that we know that the property is from the RDF namespace. In this example, the RDF property `type` was used to specify that Lovely is of the type Princess and Mikura is of the type King.

OWL makes a distinction between Classes and Individuals. In this thesis, I will indicate Individuals using a rounded rectangle, and Classes using a rectangle. In the example of Figure 3.7, the concepts ‘King’ and ‘Princess’ are Classes and the concepts ‘Mikura’ and ‘Lovely’ are Individuals of these Classes. Individuals are defined by having an `rdf:type` relationship to a certain Class. OWL provides means to define classification (i.e. ‘Girl’ is a subclass of ‘Human’) and cardinality (i.e. ‘A castle has only one gate’) which are very useful for defining knowledge about the story world. On top of this, OWL defines several classes of properties that have a special inference meaning:

- inverse properties: if `childOf` is the inverse property of `parentOf`, then from `(parentOf Lovely Mikura)` we can conclude `(childOf Mikura Lovely)`;
- symmetric properties: if `family` is a symmetric property, then from `(family Lovely Mikura)` we can conclude `(family Mikura Lovely)`;
- subproperties: if `fatherOf` is a subproperty of `parentOf`, then from `(fatherOf Lovely Mikura)` we can conclude `(parentOf Lovely Mikura)`.

OWL makes an open world assumption; the absence of knowledge cannot lead to an assumption that the knowledge is false. The fact that Mikura is parent of Lovely doesn’t mean Mikura is the *only* parent of Lovely.

With the story world definition as a basis for automated story generation, the next chapter will discuss a structure necessary to capture the events in this story world.

Chapter 4

Fabula structure

"You see there is only one constant. One universal. It is the only real truth. Causality. Action, reaction. Cause and effect. We are all victims of causality. I drank too much wine, I must take a piss. Cause and effect. Au revoir."

— Merovingian, 'The Matrix Reloaded' (2003)

Organizing actions and events, and causally relating them, is an essential step towards better character-based story generation. Independent of the plot of the story, we can relate character actions to their cause and effect and use this causality to describe the events that form the story.

To recapitulate, the first research question of this thesis is:

RQ1 How can a plot structure be designed that can be generated, narrated and reasoned about?

Section 2.4 concluded that it is necessary to come to a formalization of the fabula layer so that we know what happened, and plot sequences within this fabula layer need to be identified that we can use to guide the plot development, but also to extract a story from the fabula layer.

This chapter will discuss the development of a structure that can capture the fabula layer. I will first discuss the three-layer model of fabula, story and text in more detail to identify what elements of a story belong to these different layers. Then, I will discuss the General Transition Network Model of Trabasso et al. (1989) to come to an understanding of the different elements that are present in stories. Finally, based on this model of story analysis I will describe the design of a structure that can express the fabula behind a story and show how such a structure can be generated by the Virtual Storyteller.

4.1 A closer look at the three layers of a story

In the Virtual Storyteller, Character Agents pursue their goals in a virtual story world, a Plot Agent manipulates the world in such a way that 'interesting events' happen, and a Narrator takes these events and presents them to the reader in the form of a narrative. The responsibilities of the Character Agents, the Plot

Agent and the Narrator roughly correspond to the fabula, the story and the text layer that are present in narrative as discussed in Section 2.1. Let's look a bit closer at these layers and the responsibilities of the different Agents towards forming these layers.

4.1.1 Fabula layer

The fabula describes the sequence of events that actually take place in the story world. These events can be things that happen in the world, or things that happen in the body or mind of the characters, i.e. the cognitive, internal processes like emotions, thoughts, intentions, feelings etc. The events that happen in the world can be divided into two parts: the intentional, motivated actions of characters, and more or less coincidental world events. For instance, princess Lovely going up the castle tower is a character action, but the sun going down would be a world event.

The Virtual Storyteller setup is such that the characters are least aware of playing in a story. What is meant by this is that the goals and plans of the characters are motivated from a personal perspective, not a story perspective. They are meant to reach personal happiness, not to cause interesting drama. The goals and plans of the characters will therefore often conflict with those of the story, and as we have seen in Section 2.2.3, that is what makes the story interesting.

It makes sense then, that the Plot and Character Agents together cause fabula to emerge. Characters will plan and execute actions and the Plot Agent monitoring the unfolding action sequence will introduce world events. Character actions are motivated by the characters' goals whereas the world events are motivated by the plot goals. For this, the Plot Agent will have to have a sense of plot, and a means to determine those events that – introduced at the right time – will achieve a good plot.

4.1.2 Story layer

The story layer defines a relevant selection of the fabula layer to form a story that follows a certain plot. Like most stories, the Princess Lovely story in Figure 3.1 is based on fabula that contains much more information than is explicitly told in this particular story. It is not difficult to imagine that in fabula, many plots can coincide. Try to imagine what the fabula of this story would consist of. If we narrate the fabula from the perspective of the children, we get a totally different story about hunger and an appearing lamb, maybe a conflict about killing such a sweet animal for food. But it is easy to see that the two different stories are based on the same fabula, the same sequence of events. So what is this plot all about? What makes these two stories, which are situated in the same fabula, very different?

Aristotle (trans. 1907) identifies unity and totality as an important element of plot. This means that the plot forms a unified whole where nothing is in excess. Sgouros (1998) has formalized this paradigm in his story generation system. He states that there is only one plot sequence, or if there are more, one is clearly dominant. Everything that is told in a story relates to this one plot sequence. Stories generated in his system were revolving around one Storyline Goal being the center of the plot sequence, and he interpreted totality of plot in the sense that everything that happens in the story should be *affiliated* with this goal.

This makes sense if we again consider the Princess Lovely story. When the fabula of this story is told from the children's perspective, the story would revolve around a Storyline Goal of getting food. From the perspective of the princess the story would be about the Storyline Goal of getting friends or solving loneliness. From the perspective of the lamb it could be about successfully escaping the smothering arms of the princess, although that story obviously leads to a very dramatic ending.

Continuing on the approach of Sgouros (1998), let us define the Plot as a subset of the fabula layer containing those elements that form a unified and total whole. It is assumed that every fabula contains such a subset defined by one Storyline Goal. In fact, I would say that any goal can be used as a Storyline Goal. Under this assumption, every fabula contains a plot, however short or boring. The protagonist of the story is the character that is trying to fulfill the Storyline Goal. An interesting way to generate good stories is then to aim at generating those fabula that are most promising to deliver an interesting Plot. This Plot would be a conceptual framework of the story. It would contain relevant events and actions that can be used to narrate the story. Using the Storyline Goal and the concept of affiliation, we have a guideline for extracting a story from the fabula if we can formalize what it means for parts of the fabula to be affiliated with the Storyline Goal.

In order to be able to grasp the concept of affiliation a bit better, imagine again the fabula of the Princess Lovely story. Say that instead of this fabula, a different fabula arose wherein the lamb had escaped from the castle, but never met the children. Instead, it was standing outside in the castle garden enjoying the fresh air and chewing on some grass, and that is where Lovely finds it and lives happily ever after. In this version of the fabula, the children are still hungry, playing, and thinking of ways to find food. The existence of the children and the fact that they are playing is still affiliated to the Storyline Goal because they cause Lovely's loneliness, but the fact that they are desperately hungry, albeit very dramatic, is no longer affiliated to the Storyline Goal. It is irrelevant. Indeed, if this fact is included in the story, it will make the reader confused since it brings up expectations in the reader at the end of the story. What happened to the hunger of the children? They will have expected the hunger to be relevant. Fabula elements are relevant when they have a certain effect on the course of the story, that is, a direct or indirect effect on the Storyline Goal.

4.1.3 Text layer

The text layer is all about how the story is presented to the reader. There is more to the presentation of the plot than simply describing what happens. Presentation not only conveys facts but also moods, emotions and tensions. It suggests the future and refers to the past. Writing techniques such as suspense and dramatization can make the story much more interesting.

For example, when Lovely walks out of the castle to search for her lamb, she is afraid. The presentation could be dark and panicky, chaotic. If she were to walk out of the castle to enjoy the sun, the presentation could be totally different, calm, joyful, birds singing, bright green grass. She could be skipping in the garden which is totally irrelevant in the context of the fabula unless her skipping causes her for instance to sprain her ankle. The emotions of the viewpoint character seem to shape the way the world is described. Had Lovely been sad, then the weather might have been described as cloudy, and she would stroll.

Of course, sometimes the world can also shape emotions. When the weather is dark and cloudy, a character could become a bit down, and singing birds could cheer a character up. Obviously, in this case the weather and the birds *are* part of the fabula, since they have an effect on the behavior of the character.

Writing techniques should be based on the internal world of the viewpoint character and on the dramatic tensions that are already present in the plot. This will be future work and will not be further discussed here except to outline which elements should be part of the fabula, and which elements should be added when the story is presented or narrated.

I make an assumption that the plot can be presentation independent, although (Pérez y Pérez & Sharples, 2004) state that a lack of interaction between generating coherent and interesting event sequences during the development of the story and the text that describes these event sequences is an important shortcoming of several storytelling systems including MINSTREL. One can argue that there are stories where plot and presentation do indeed overlap, sometimes even to a great extent. The plot of detectives for instance relies a lot on presentation: it is all about slowly and partially revealing more and more information. The actual plot is usually very simple and boring: John has died because Mary felt revengeful, and a detective finds this out. The interestingness seems to come almost completely from presentation.

The focus will be on stories where the presentation can in fact be generated independent from the plot. Elements that illustrate the moods of the characters, like singing birds and sunny weather, should not be included in the fabula generation process and should instead be added by the Narrator. There are many advantages to this:

- The Plot Agent and Character Agents are not responsible for writing techniques when generating the fabula. For instance, they don't have to be concerned with adding more detail when things get exciting;
- Presentation independence makes it possible and flexible to have many ways of presentation;
- Search space for character actions can be kept much smaller since actions that have to do with presentation don't have to be simulated. For instance, Princess Lovely doesn't have to include skipping in the garden in her action plans;
- The Narrator has a much more powerful role than it has now. It can shape and structure the Plot in any specific way. A presentation based on visualization could make totally different choices in this.

How to enrich the fabula for presentation purposes is beyond the scope of this thesis. Slabbers (2006) discusses this in more detail.

4.2 General Transition Network Model

The General Transition Network Model of Trabasso et al. (1989) (see Figure 4.1) is a formal model for story analysis. It can be applied to an existing story to capture the underlying structure of it. The model was successfully applied to narrations of a picture story by children and adults (Trabasso & Nickels, 1992). Children and adults were asked to narrate the events that were happening or inferred to be happening in a sequence of pictures, and the clauses that these children used were categorized using the model to expose the underlying narrative structure.

This model can be used to come to a formalization of a practical fabula structure for story generation. The model identifies a causal network of hierarchical episodes. An episode is claimed to be revolving around a Goal and an Outcome for a certain character, occurring within a certain Setting.

A Setting (**S**) enables (e) such an episode in the form of an Event (**E**) which can psychologically cause (ψ) an Internal Response (**IR**) in a character in the form of cognitions, emotions, beliefs etc. or a Goal (**G**). A Goal can and will often be psychologically caused by an Internal Response, and in turn motivates (m) an Attempt (**A**) to reach this Goal, or motivates another (sub) Goal. Each Attempt can enable new Attempts, and eventually physically causes (ϕ) an Outcome (**O**) that is either successful, unsuccessful or neutral with respect to goal attainment. Outcomes, like Events, can physically cause other Outcomes or psychologically cause Internal Responses or Goals. An episode doesn't necessarily start with an Event; a Setting can enable any of the five elements of an episode.

4.2.1 Usability for story generation

The General Transition Network Model is a model of story *analysis* which isn't useful for story *generation* unless we succeed to formalize the way in which the different elements and their connections can be created, and subsequently narrated. While at surface level the model seems to be directly applicable, it is important to realize that the model does not define what the actual elements like Goals, Attempts and Outcomes consist of, but uses these terms to *categorize* the clauses of the story text. For story generation this is a rather superficial approach since on the textual level, the elements of the actual fabula on which the story is based are only *described* and often even left implicit. For instance,

$$\begin{tikzcd}
 & & S \arrow[d, "e"] & & \\
 & & {} & & \\
E \arrow[loop above, "\phi"] \arrow[r, "\psi"] \arrow[rr, bend right=40, "\psi"'] & IR \arrow[loop above, "\psi"] \arrow[r, "\psi"] & G \arrow[loop above, "m"] \arrow[r, "m"] & A \arrow[loop above, "e"] \arrow[r, "\phi"] & O \arrow[loop above, "\phi"] \arrow[lll, bend left=40, "\psi"] \arrow[ll, bend left=30, "\psi"]
\end{tikzcd}$$

Figure 4.1: *General Transition Network Model*

the princess Lovely story of Figure 3.1 doesn't mention the reason why the princess climbed up to the tower. Many readers will infer that she was curious, but this is not mentioned. A model of the fabula of this story should include the reason for her climbing up the tower, but an analysis of this story would not reveal that reason.

In our storytelling Multi Agent System, the Plot Agent as well as the different Character Agents should contribute to the fabula structure, and the Narrator should be able to interpret it. Let's look at the different aspects of the model, and discuss the usability for story generation.

Event

There are two viewpoints on which to discern an Event. On a global viewpoint, an Event is something that happens in the world, with or without causation, e.g. a sun that goes down, a tree that falls down, a twig that breaks. From a personal viewpoint, an Event is something that triggers a character to respond, e.g. the king arriving could be an event from the perspective of the princess, and the princess crying could be an Event from the perspective of the King.

The General Transition Network Model adheres to the second, personal viewpoint. The reason for this is that the model is meant to construct a causal network from the perspective of a single character. It does not attempt to make interpersonal causalities, like an Attempt of one character evoking an Internal Response in other characters. Rather, a single clause can be categorized in different categories for different characters. For instance, an Attempt of one character can be an Event for another character. From such a perspective, Events directly cause Internal Responses or Goals. It makes sense that for this reason, perceptions are not included in their model.

There is a close relationship between an event that happens in the world and a perception of this event. When we look at the (emotional) reactions of characters, an event is the cause of this reaction on a conceptual level, whereas a perception causes it on a functional level. When telling a story, it is the difference between saying that Lovely became worried because the lamb was gone, and saying that she became worried because she *saw* that the lamb was gone. Even though these statements look very similar, they are fundamentally different.

From the viewpoint of a global fabula structure that I intend to design, a distinction is necessary between how the world actually *is*, and how a character *believes* it to be. Therefore I adhere to the global viewpoint and propose to define an Event as any change in the world that is not the direct and planned result of any character action, because this suits the role of the Plot Agent in the generation of fabula, since the Plot Agent will introduce these Events. The characters will respond to these Events using a model of event appraisal.

An Event defined in this way is always independent of the chosen viewpoint; it describes a change in the world as it is. A perception is always personal, changing how a character believes the world to be. Sometimes Events happen that are not perceived by any character, but are still relevant for the story. Sometimes "wrong" perceptions occur, perceptions that don't match the actual Events. A clear example can be seen in Shakespeare's story of Romeo and Juliet, where Romeo sees Juliet and thinks she is dead, while she is in fact still alive. An Event is appraised according to the character's perception of it. A perception of an Event or Setting is therefore maybe best seen as a distinct story element.

Goal

A goal is the main drive for characters to act. Goals are desired or undesired states, activities, or objects. A Goal should describe such a desire to attain, maintain, leave or avoid (Trabasso et al., 1989). This identifies several classes of goals:

to attain a state	to attain an object	to make an activity possible
to maintain a state	to keep an object	to maintain an activity
to leave a state	to lose an object	to stop doing an activity
to avoid a state	to avoid an object	to avoid an activity

Goals like this can be adopted by the characters. Note that the classes of goals about attaining, keeping, losing and avoiding an object are actually goals about the states in which the object is attained, kept, lost and avoided. A state could be a state of the world around a character, or an internal state like an emotion. A Goal could for instance be to attain a state where the character is rich, or to avoid a state of being lonely.

Attempt and Outcome

Under the assumption of the characters being autonomous planning Agents in a story world, the concept of Attempt can be formalized in the form of a collection of goal-driven actions in a way that resembles the notion of ‘plan’ in the domain of planning Agents (Russell & Norvig, 1995, pp. 346–347). The characters have Goals that will motivate other (sub-) Goals, or Attempts to fulfill these Goals, leading to a certain Outcome.

The term Action will be used to indicate a goal-driven, intentional world change brought about by a character. A single Action doesn’t always constitute an Attempt, but an Attempt could be seen as the set of Actions that are motivated by a certain Goal.

The Outcome is closely tied to the world as the character knows it to be, not necessarily to the world as it actually is. When a character believes that its Goal is fulfilled, the Goal will have a positive Outcome. If the character believes that the performed Actions didn’t succeed in fulfilling the Goal, the Outcome will be negative. This enables some interesting dramatic situations. For instance in the story of Romeo and Juliet, Juliet pretends to be dead, but isn’t actually dead. Romeo however thinks Juliet is dead, which is a very negative Outcome for his Goal to be together with her. It even leads him to commit suicide. In the Princess Lovely story, the princess finds the lamb, which is a positive Outcome for her, but then finds out it is dead, which is a negative Outcome for her super goal of getting her friend back. Outcomes are determined by what a character believes.

Internal Response

Trabasso et al. (1989) categorize cognitions, emotions and beliefs, the complete domain of character experiences, under the common denominator ‘Internal Response'. I propose to use the term Internal Element rather than Internal Response. The word 'response' suggests a causation even though there isn't always one. When the children are hungry, there is hardly a cause, at least not at the abstraction level I intend to make. Trabasso et al. would probably categorize this hunger as a Setting or maybe an Event, which is acceptable for story analysis but from the story generation point of view this hunger would arise in a character and is therefore much better in its place categorized as a piece of (internal) state that is not caused by any Event. The term Internal State was considered but would imply that I mean the character's total state whereas I am only referring to an element from this state.

The same goes for the character's beliefs, which I will identify with the term 'BeliefElement', because when a perception leads to a belief update, this update is represented by the specific part of the belief of the character that changes.

BeliefElements can enable (latent) goals and actions. Goals that were previously not possible can suddenly become reachable due to an updated belief. An impossible action needed for a Goal to be reached can suddenly become possible due to perception and belief of new information. BeliefElements can psychologically cause Outcomes because a Goal can be resolved not only by goal-driven actions, but also by updated beliefs due to external events, be it actions of other characters, or 'incidental' events.

Expressing emotional states like Princess Lovely crying is not very straightforward. Is the crying an emotion, or is it an Action caused by sadness? Both make sense. At least we can agree that emotional actions like crying and laughing don't really qualify as a goal-driven Action unless interpreted at a deeper psychological level, for instance when the princess cries to induce a caring response from the king.

An emotional action like screaming or crying or singing could be viewed as a very concrete, perceptible Internal Element. However, the General Transition Network Model doesn't specify how Internal Elements influence other characters, nor should it, since these elements are personal and internal. Therefore it's probably best to categorize emotional actions as Actions that are not goal-driven, i.e. not motivated by a Goal. Instead, these Actions are motivated by Internal Elements. They would typically be used to express the character's internal state in the form of for instance crying, or grimacing[1]. This also leaves open the possibility that characters can fake their feelings to manipulate other characters for their own purposes, which is of course an interesting possibility for a story.

Setting

A Setting can be seen as the state of the world that is not the result of any Events or Actions. For the reader, Settings are interpreted as if the world has been this way ever since the story started, even when they are told later. Unlike Events, which occur in the course of the story. A Setting cannot influence a character by itself but the perception of it can.

A Setting is something that can be changed in the course of the story so long as it is not interpreted by the reader as being a world change. For instance,

[1]On the other hand, making Internal Elements invisible and relying on the characters to express them might make the model overly complex. A different solution might turn out to be necessary.

on an implementation level the children playing outside could be added as a Setting at the moment where Lovely enters the tower because the Plot Agent wanted to cause an interesting Perception. For the reader, it is as if nothing changed, as if the children had always been playing there since the beginning of the story.

4.3 A model of fabula for story generation

As a result of the discussion of the General Transition Network Model, I propose a model of fabula structure as shown in Figure 4.2. This model can express the fabula layer of a story. In contrast to the General Transition Network Model, this model has a global perspective in order to be able to create one network for the whole fabula, in stead of different networks for each character's perspective.

The model therefore introduces Perceptions (**P**) as a central element in the causal network. Perceptions are physically caused by either Actions or Events, and can in turn psychologially cause Internal Elements. Also note the personal role of the Outcome, which has only psychological causes and effects.

Another novelty is that Actions can physically cause Events. This way, unforeseen side effects of Actions can be modeled. When for instance Princess Lovely climbs up the castle tower (A), this can "cause" her to fall (E). This was not Lovely's intention but it is still causally related to her Action.

Note that this structure doesn't form a complete network of everything that happened in the course of the story, but only of those fabula elements that can be structured in a causal network, in other words, only those elements that have either a cause or an effect are captured. Therefore, Settings are left out of this model. They are considered to be the background against which the story unfolds. From a narration point of view, an enabling relation between parts of the Setting and different story elements seems hardly relevant.

$$\begin{tikzcd}
 & & IE \arrow[loop above, "\psi"] \arrow[d, "\psi/e"] \arrow[dr, "m/e"] \arrow[drr, bend left, "\psi"] & & \\
E \arrow[loop above, "\phi/e"] \arrow[r, "\phi"] & P \arrow[ur, "\psi"] & G \arrow[r, "m"] \arrow[loop below, "m"] & A \arrow[ll, bend left, "\phi"] \arrow[lll, bend left=60, "\phi"] & O \arrow[ull, "\psi"'] \arrow[ll, bend left, "\psi"]
\end{tikzcd}$$

Figure 4.2: *Fabula model for story generation*

4.3.1 Causal connections

A prerequisite of the proposed model is that the different causal connections can be generated by the Virtual Storyteller. It is therefore important to have a close look at the different connections to see how and by what part of the system these connections can be made.

$$\begin{tikzcd} E \arrow[loop above, "\phi"] \end{tikzcd}$$

Event physically causes Event An Event can cause another Event, for instance a tree that falls on the bridge causes the bridge to break. Or Sleeping Beauty pricking herself on the spinning wheel causes her to fall asleep. Generating this causality requires either a certain kind of physical reasoning to really model how an Event causes another Event, or it requires a library containing examples of how certain Events physically cause other Events.

$$\begin{tikzcd} E \arrow[loop above, "e"] \end{tikzcd}$$

Event enables Event An Event ontology has not been designed yet, but Events will probably have preconditions and effects in the same way that Actions do. An Event E_1 enables an Event E_2 if the effects of E_1 match or overlap with the preconditions of E_2.

$$\begin{tikzcd} E \arrow[r, "\phi"] & P \end{tikzcd}$$

Event physically causes Perception The Plot Agent determines what characters perceive. When an Event happens, and the Plot Agent sends perceptions of world changes due to this Event to a character, a connection of physical causality can be made.

$$\begin{tikzcd} P \arrow[r, "\psi"] & IE \end{tikzcd}$$

Perception psychologically causes Internal Element The character will appraise the incoming Perceptions using a certain model of Event appraisal. This causes an automatic (psychological) connection between a Perception and an Internal Element that is the result of the Event appraisal.

$$\begin{tikzcd} IE \arrow[loop above, "\psi"] \end{tikzcd}$$

Internal Element psychologically causes Internal Element This connection arises from further processing of Internal Elements by the character. It arises due to cognitive processes, for instance beliefs or cognitions can lead to emotions and emotions can lead to other emotions.

$$IE \xrightarrow{\psi} G$$

Internal Element psychologically causes Goal The characters need a way to determine Goals based on their beliefs, desires and emotions, done by the Character Agents. It leads to a psychological causation for the Goal which is an important element for character believability.

$$IE \xrightarrow{e} G$$

Internal Element enables Goal This mainly applies to BeliefElements. When a character believes something to be true, a Goal can be enabled by this belief. For instance, when Lovely believes the lamb has escaped, she can adopt a Goal to go outside and find it.

$$IE \xrightarrow{m} A$$

Internal Element motivates Action Actions that are causally connected to Internal Elements are emotional Actions that are not goal-driven, as discussed in Section 4.2.1. These are Actions like crying and screaming, that are directly caused by an Internal Element and not by a strategic attempt to fulfill a Goal.

$$IE \xrightarrow{e} A$$

Internal Element enables Action This again mainly applies to BeliefElements. When a character believes something to be true, an Action can be enabled by this belief because the belief matches the Action's preconditions. For instance, when Lovely believes there are stairs that lead to the tower, this enables her to walk up these stairs. Whether there actually *are* stairs, and whether the Action succeeds, is out of her control.

$$IE \xrightarrow{\psi} O$$

Internal Element psychologically causes Outcome When a Goal is about attaining or stopping an Internal Element, this Goal will be resolved due to a new Internal Element, which leads to an Outcome caused by this Internal Element. When a character believes that his plan failed or succeeded, this will also lead to an Outcome, namely a positive Outcome when the plan succeeded, and a negative one when the plan failed. When a character stops wanting the Goal, this will lead to a neutral Outcome.

$$\begin{tikzcd} G \arrow[loop above, "m"'] \end{tikzcd}$$

Goal motivates Goal A Goal G_1 motivates another Goal G_2 when G_2 is a sub goal of G_1, selected for instance based on a library of solutions to problems. For instance, Princess Lovely's goal of wanting the lamb back could motivate a sub goal of finding the lamb. The library will have to contain examples of goal-subgoal structures.

$G \xrightarrow{m} A$

Goal motivates Action Due to the Character Agent's planning algorithm, Actions arise that are goal-driven. When such an Action is planned or executed, it is motivated by a Goal and a motivational causality can be made.

$A \xrightarrow{\phi} P$

Action physically causes Perception Characters should be enabled to perceive Actions. When an Action is perceived by a character, a physical causality occurs. Such a causality is also made when world changes are perceived that are the result of an Action.

$A \xrightarrow{\phi} E$

Action physically causes Event This is how an unforeseen effect of an Action can be modeled. For instance, when Princess Lovely crosses the castle bridge, it can cause the bridge to collapse. When there is no Event, she ends up on the other side, the normal effect of the Action.

$O \xrightarrow{\psi} G$

Outcome psychologically causes Goal When a failed attempt leads to the reinstatement of the Goal, a new episode starts with the same Goal as before, but probably with a different attempt, if the Character Agent is smart.

$O \xrightarrow{\psi} IE$

Outcome psychologically causes Internal Element Appraisal of goal success/failure.

It is important to realize that whereas physical and psychological causation (ϕ and ψ, resp.) also define a temporal ordering of occurrence of the fabula elements, motivational causality (m) and enablement (e) do not. If an Action psychologically causes an Internal Element, both elements really took place, and we can conclude that the Internal Element took place after the Action. But when an Action is motivated by a Goal or enabled by a BeliefElement, this doesn't automatically mean that the Action took place.

Say that Lovely has a Goal to get to the other side of the moat around the castle. She knows that there is a bridge over the moat and therefore an Action to cross the bridge is motivated by the Goal. But when Lovely gets to the bridge, she sees that it is collapsed. This new knowledge will lead to a failed Outcome and an Action that was never performed. However, this Action was motivated by a Goal, and enabled by a BeliefElement that the bridge was there, and will still be included in the fabula structure as an Action that was never performed.

Another possibility is that Lovely *didn't* know there was a bridge, and intended to swim to get to the other side. When she gets to the moat, she sees a bridge and decides to use the bridge because that is easier. The Action to swim is still motivated by the Goal but a new Perception has enabled another Action motivated by the Goal, and the swim Action is never executed.

To see whether an Action was successfully executed, failed or never executed at all, the Action should be annotated with a begin and end time. An Action that has a begin time set, was started. An Action that has an end time, was successfully executed. Similarly, Events should be annotated with a begin and end time as well.

4.3.2 An example

To make it plausible that the fabula of a story can actually be represented using the proposed structure, I have exposed the fabula of a fragment of the Princess Lovely story (Figure 3.1) where the lamb flees, the children find and eat it, and Lovely goes out to find it. This particular selection was made because it represents two conflicting plans revolving around the lamb, leading to an outcome that is totally different for both characters: the children get their food and the princess loses her lamb.

To expose the fabula of an example story, we need to reverse engineer the story and infer its fabula, as many fabula elements are often implicit. Trabasso et al. (1989) organized the clauses of the story into different categories, but this is insufficient for story generation purposes. All the events behind the surface of the story need to be made explicit to display the structure of these events. Table 4.1 and Figure 4.3 show the inferred fabula elements and their causal connections, respectively.

$IE_{c(1)}$:	The children are hungry.
$G_{c(1)}$:	The children want to stop being hungry.
E_1:	The lamb flees.
$P_{L(1)}$:	Lovely hears that the lamb has fled.
$IE_{L(1)}$:	Lovely believes the lamb is gone.
$G_{L(1)}$:	Lovely wants the lamb back.
$G_{L(2)}$:	Lovely wants to find the lamb.
A_L:	Lovely runs out of the castle.
E_2:	The lamb appears on the children's location.
$P_{c(1)}$:	The children see the lamb.
$G_{c(2)}$:	The children want to eat the lamb.
$A_{c(1)}$:	The children slaughter the lamb.
$P_{L(2)}$:	Lovely sees the lamb above the fire.
$IE_{L(2)}$:	Lovely believes the lamb is dead.
$O^+_{L(2)}$:	(Lovely has found the lamb)
$O^-_{L(1)}$:	(Lovely cannot have the lamb back)
$A_{c(2)}$:	The children eat the lamb.
$P_{c(2)}$:	The lamb is eaten.
$IE_{c(2)}$:	The children are not hungry.
$IE_{c(3)}$:	The children believe they have eaten the lamb.
$O^+_{c(2)}$:	(The children have eaten the lamb)
$O^+_{c(1)}$:	(The children are no longer hungry)

Table 4.1: *Fabula of a selection of the Lovely story*

$$\begin{tikzcd}
IE_{c(1)} \arrow[r, "\psi"] & G_{c(1)} \arrow[dr, bend left, "m"] & & & & IE_{c(2)} \arrow[r, "\psi"] & O^{+}_{c(1)} & \\
& P_{c(1)} \arrow[r, "e"'] & G_{c(2)} \arrow[r, "m"] \arrow[rr, bend left, "m"] & A_{c(1)} \arrow[r, "e"] \arrow[ddd, "\phi"] & A_{c(2)} \arrow[r, "\phi"] & P_{c(2)} \arrow[r, "\psi"] \arrow[u, "\psi"] & IE_{c(3)} \arrow[r, "\psi"] & O^{+}_{c(2)} \\
E_1 \arrow[d, "\phi"] & E_2 \arrow[u, "\phi"] & & & & & & \\
P_{L(1)} \arrow[d, "\psi"] & G_{L(1)} \arrow[d, "m"] & & & & O^{-}_{L(1)} & & \\
IE_{L(1)} \arrow[ur, "\psi"] & G_{L(2)} \arrow[r, "m"] & A_L \arrow[r, "\phi"] & P_{L(2)} \arrow[r, "\psi"] & IE_{L(2)} \arrow[r, "\psi"] \arrow[ur, "\psi"] & O^{+}_{L(2)} & &
\end{tikzcd}$$

Figure 4.3: *Structure of fabula selection*

The children are hungry and want to end their hunger. Suddenly, a lamb appears. The children see the lamb and because they are so hungry they want to eat it. Therefore they slaughter it and hang it above the fireplace.
At the same time, Lovely finds out that the lamb has fled. She wants the lamb back and runs out of the castle. When she sees the lamb above the fire, she believes the lamb to be dead. She failed to get her friend back. However, the children are no longer hungry after eating the lamb.

Figure 4.4: *Possible simple narration of fabula selection*

When Table 4.1 is read from top to bottom it becomes clear how sequentially narrating the elements of the fabula already forms the basis for a story since the causalities are often implicitly clear and inferred by the reader. However, with the causalities explicitly present, this part of the story can be told with much more concept of story. Figure 4.4 gives a possible narration that was generated by hand, but is not far off from texts that the Narrator that is being developed by Slabbers (2006) is able to produce.

Of course, the more expressive the generated fabula structure is, the more engaging the story can be told. In the example we could for instance have included Lovely's love for the lamb, and her panic at seeing the lamb gone, and the children's conflicting feelings of wanting food, but not wanting to kill an innocent animal.

4.3.3 Fabula ontology

A direct translation can be made from the proposed model to an ontology to describe fabula. Such an ontology has been created in OWL using the Protégé ontology editor (Protégé, n.d.). Figure 4.5 shows the concept hierarchy of this ontology. It is not a complete ontology, but forms the basis for further extensions. For instance, one can imagine that the emotional diversity of the Character Agents will be much more extensive than just `Joy` and `Sadness`, which are added as an example. New Character Agents are currently being developed and can for instance follow the model described in (Theune et al., 2004). Note that the Action part of the ontology is not expanded in this figure. It is formed by the Action ontology defined in (Uijlings, 2006), see Section 3.2. The Event part has not been designed yet.

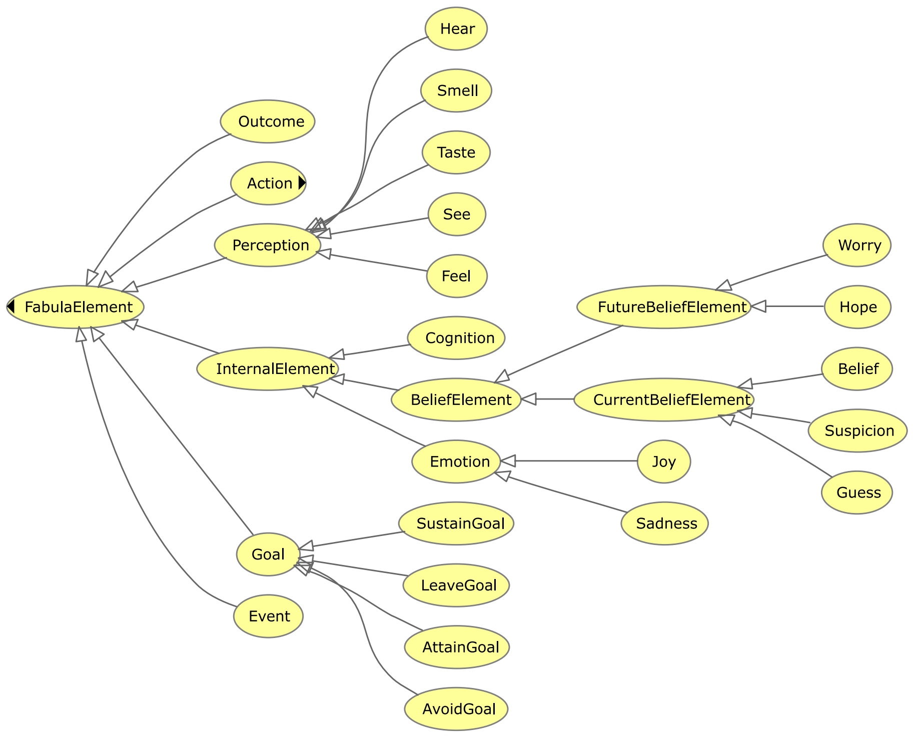

Figure 4.5: *Fabula Ontology*

The properties of the concepts describe the following information about the concepts:

- Causality information. Following the causal relations described in Section 4.3.1, the ontology will define the properties `motivates` (m), `enables` (e), `phi_causes` (ϕ) and `psy_causes` (ψ);
- Time information. Each fabula element will be annotated with an integer indicating the moment in time when the element came into play. This time moment enables a temporal ordering. On top of this, Actions and Events are annotated with the time they started and the time they finished;
- Character information. Each non-Event element is related to a character in the story and this character is annotated in the elements using a `character` property. It specifies for example which character performed a certain Action, or had a certain BeliefElement.

4.4 Adding meaning to the model

Of course, using an ontology of Actions, Events, Goals and Internal Elements to describe the fabula of a story is not very useful if we cannot express what the concepts of this ontology describe. A causal model of fabula elements has been developed, but a way to describe the meaning of these elements – as expressed in textual form in the right columno of Table 4.1 – has not been discussed yet.

Some concepts, such as Actions, Events and emotions, have a meaning that is inherent to the concept. For instance, if we say Lovely is sad, or if we say she performed a Walk action, we know enough. Other concepts, like Goals, BeliefElements and Perceptions, derive their meaning from the knowledge they describe. For instance, if we say that princess Lovely has an AttainGoal, we need to express what this goal is *about*. For instance, Lovely's Goal $G_{L(2)}$ in Table 4.1 is a Goal about attaining a state where she finds the lamb.

In fact, such fabula elements describe world states or parts of world states that describe the present, future or past. For example, a Perception would typically describe a present world state, a Goal describes a possible future world state. A belief is a mental world state and can describe the past (i.e. a memory), the future (i.e. a worry) or the present. The causal relationships connect possible worlds (Schärfe, 2002).

The concepts with inherent meaning are Events, Actions, Emotions and Outcomes. Events and Actions define which real world transitions occur in the form of specified effects. The class hierarchy of these concepts and their semantic definitions of preconditions and effects establish their meaning. For the same reason, feelings such as Emotions are also self-defined. A class hierarchy of these feelings can be modeled that represent the semantics of the feelings[2]. Similarly, an Outcome is just positive, negative, or neutral with respect to a certain Goal. Whether the Outcome is positive, negative or neutral, can be captured in the form of different Outcome classes, or a property.

The concepts that have no inherent meaning are Perceptions, Goals and Beliefs. They encompass parts of possible worlds. We will want to represent knowledge like:

(1) Lovely wants *to possess the lamb.* Lovely has a Goal to attain the state where she possesses the lamb.

(2) Lovely wants *to have a friend.* Lovely has a Goal to attain the state where she has a friend.

(3) Lovely believes *that the children want to play with her.* Lovely has a BeliefElement that specifies the fact that the children want to play with her.

(4) Lovely wants *that Mikura allows her to go outside so she can play with the children.* Lovely has a Goal to attain the state where she is playing with the children, which motivates a Goal to attain the state where Mikura performs an Action to allow her to go outside.

[2]This excludes the possibility of expressing that Amalia is afraid of Brutus attacking her. This can be represented by a Belief that causes an Emotion (Amalia believes Brutus will attack her $\rightarrow^{\psi}$ Amalia is afraid

(5) Lovely believes *that if she cries, Mikura will feel pity and allow her to go out.* Lovely has a BeliefElement that specifies the fact that an Action to cry will cause a Perception by Mikura that will cause him to have an Internal Element of pity, which causes a Goal for him to make Lovely happy, which motivates an Action to allow her to go out.

(6) Lovely sees *that the lamb is not in the castle.* Lovely has a Perception that states the fact that the lamb is not in the castle.

(7) Lovely believes *that someone has stolen her lamb.* Lovely has a BeliefElement that states that someone performed an Action to steal her lamb.

(8) Lovely asks the king, *whether she can go outside.* Lovely performs a speech act that conveys the question that Lovely is allowed to go outside.

Lovely's BeliefElements, Goals and Perceptions in above sentences (in italics) describe parts of possible worlds which can be expressed using concepts from both the story world ontology and the fabula ontology. These parts (as shown in italics) can be seen as little pieces of fabula. Example (8) shows that this is also required for speech acts[3]. To represent this, these sub networks should be included in the general network while at the same time remaining at a different conceptual level, because it concerns different worlds. However, it is not straightforward to embed this kind of knowledge using the flat structure of semantic networks. Let's explore some possibilities.

4.4.1 Contextualizing knowledge: some possibilities

Several ways of expressing knowledge about knowledge in semantic networks will be explored. However, as we will see, all these approaches have disadvantages that make them unsuited for current use.

String representation

The simplest way to represent the contents would be to put them in a string describing the knowledge of the sub networks. For instance, to represent the semantics of (1), this would come down to something like:

```
(contents attainGoal423 "(possesses Lovely Lamb)")
```

The disadvantage of this approach is obviously that the contents are not easily accessible as knowledge; the string that describes the sub network would have to be parsed and interpreted in order to gain access to its meaning.

Named Graphs

Research is being done into extending RDF using Named Graphs. Carroll, Bizer, Hayes, and Stickler (2005) propose to extend RDF triples with an extra value to refer to information sources or more generally, contexts. This proposed extension is intended for several purposes including that of representing beliefs.

[3]Speech acts have not been designed but must be kept in mind for future versions of the Virtual Storyteller.

Named Graphs look very promising for future use, but at the time of writing there seems to be only one implementation of Named Graphs available (Named Graphs API for Jena [NG4J], n.d.), and even this implementation is experimental. The knowledge system that is being used in the Virtual Storyteller (namely JTP (Java Theorem Prover [JTP], n.d.), see Appendix C) does not support Named Graphs and the limited time span of this project prevents a switch to a Named Graphs implementation. These drawbacks make an approach using Named Graphs unsuited for current use.

RDF reification

RDF provides a mechanism for describing knowledge, which is called *reification*[4]. It is a mechanism used to *describe* information rather than represent it. It makes use of the special RDF class `Statement` to reify the knowledge into concrete Individuals. Every `rdf:Statement` describes a triple. For example:

```
(rdf:type state345 rdf:Statement)
(rdf:subject state345 Mikura)
(rdf:predicate state345 isParentOf)
(rdf:object state345 Lovely)
```

describes the triple:

```
(isParentOf Mikura Lovely)
```

In other words, the `rdf:Statement` in the above example states *that Mikura is the parent of Lovely.* The advantage of this representation is that it is easier to access the knowledge in comparison to a representation where the contents of a Goal or BeliefElement are represented as a string. The disadvantages of this approach are numerous though.

The biggest disadvantage is the fact that knowledge described in the form of an `rdf:Statement` exists on a different conceptual level. A collection of Statements *describes* knowledge, but does not *assert* it. The description of a reified triple as in the above example does not imply the presence of the triple itself. This makes it very difficult to reason with reified knowledge. For example, let's say the inverse property of `isParentOf` is called `isChildOf`. When Mikura `isParentOf` Lovely, the fact that Lovely `isChildOf` Mikura is inferred. But when there is a Statement *describing* that Mikura `isParentOf` Lovely, we cannot infer that Lovely `isChildOf` Mikura in this described world without explicitly asserting the described knowledge. The description exists on a different conceptual level than the knowledge itself.

Another known disadvantage of using reification is that RDF provides no way to link the values of the reification to their actual individuals. In the above example, there is no way to make sure that `Lamb` is actually referring to a defined Individual `Lamb` in the same knowledge base. For instance, if the above `rdf:Statement` would be written on a note to the children (i.e. the children find a piece of paper that says: Lovely possesses the lamb), there is no guarantee that the children know of this 'lamb'[5].

[4]See http://www.w3.org/TR/2004/REC-rdf-primer-20040210/#reification

[5]According to http://www.w3.org/TR/2004/REC-rdf-primer-20040210/#reification, this identification must be done by mechanisms outside RDF.

A possible solution using reification despite the limitations described is to use a `contents` relationship which links fabula elements to other fabula elements or `rdf:Statements`. For instance the contents of a BeliefElement can be represented by an Internal Element (Mikura believes that Lovely is sad), the contents of a Goal can be another Goal (Lovely wants Mikura to want to let her go out of the castle), an Event (The lamb wants the gate to open), an Action (Mikura wants Lovely to play with the lamb), or the just introduced `rdf:Statements` (Mikura wants that Lovely possesses the lamb). Simple forms of embedding can now be represented as in Figure 4.6 and Figure 4.7; the `contents` relationship is indicated by the letter c.

$$\begin{tikzcd} BE_L \arrow[r, "c"] & G_C \arrow[r, "c"] & A_C \end{tikzcd}$$

Figure 4.6: *"Lovely believes (BE_L) that the children want (G_C) to eat (A_C) the lamb."*

$$\begin{tikzcd} G_{L(1)} \arrow[r, "c"] \arrow[d, "m"] & A_M \\ G_{L(2)} \arrow[r, "c"] & IE_M \end{tikzcd}$$

Figure 4.7: *"Lovely wants ($G_{L(2)}$) Mikura to feel pity (IE_M) because she wants ($G_{L(1)}$) him to allow her to go outside (A_M)*

$$\begin{tikzcd} & S_1 \\ G_L \arrow[ur, "c"] \arrow[r, "c"] \arrow[dr, "c"] & S_2 \\ & S_3 \end{tikzcd}$$

Figure 4.8: *"Lovely wants (G_L) the children to be friends with her."*

Goals can have more than one element defining their content. More complex knowledge can be described by a set of simple Statements. For instance, a Goal for Lovely could be that the children like her. This could be represented as in Figure 4.8. S_1 represents that there is an object Friendship, S_2 represents that the Children are connected to that Friendship, S_3 represents that Lovely is also connected to that Friendship.

4.4.2 Alternative approach

Where Named Graphs consider the graph to be the context of knowledge, a solution resembling Named Graphs might be to treat the fabula elements as a possible context for knowledge as follows.

- Define a functional property `hasContext` that can be used to link any Individual i_1 to an Individual i_2 identifying the context of i_1. Functional properties have at most one value; Individuals can exist in only one context. i_2 is typically a Perception, BeliefElement or Goal Individual.
- Make sure that properties of Individuals describing world knowledge have as values only Individuals within the same context. This way, we can identify which relationships exist in which context by looking at the context of the Individuals themselves.

Let's look at an example where a BeliefElement *that Lovely loves the lamb*, psychologically causes a Goal *to keep it in the castle*, in other words, to maintain the state where it is in the castle[6]. In this example, the fact that Lovely loves the lamb has to exist within the context of the BeliefElement. The fact that the lamb is located in the castle has to exist within the context of the MaintainGoal.

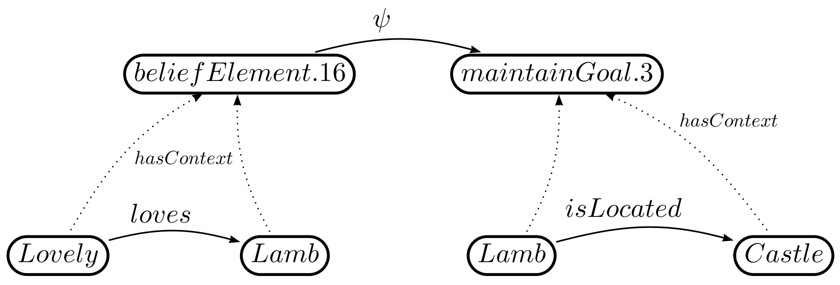

Figure 4.9: *"A belief that Lovely loves the lamb, psychologically causes a goal to keep it in the castle." version 1*

On first sight, Figure 4.9 looks like a good solution. The Individuals have a `hasContext` relationship defining their context. However, the problem is that we haven't ensured that the knowledge *about* these Individuals is contextualized as well. Because in the knowledge base describing the example, the following two triples are true:

```
(loves Lovely Lamb)
(isLocated Lamb Castle)
```

We cannot tell which of these triples belong to the BeliefElement, and which belong to the MaintainGoal. This is because the Individual `Lamb` has two contexts because it has two `hasContext` relationships, which is inconsistent with our definition of `hasContext` being a functional property. When we express

[6]Which character has the BeliefElement or Goal is not relevant for this example, but this would be indicated using a `character` relationship.

knowledge about this `Lamb`, it is impossible to identify to which context this information belongs.

If we make a clear distinction between the actual entities in the story world (i.e. "the actual lamb as an entity in the story") and the Individuals describing them in the fabula knowledge (in this case, the Individual `Lamb`), we can solve this problem by using different Individuals to mean the same entity. An example can be seen in Figure 4.10. The Individuals `Lovely`, `Lamb` and `Castle` are Individuals that exist in 'reality', in the knowledge of the world as it is. The other Individuals, `b_lovely.23`, `b_lamb.3`, `g_lamb.6` and `g_castle.16` exist in someone's Belief, or Goal and act like a certain kind of 'dummy' Individuals.

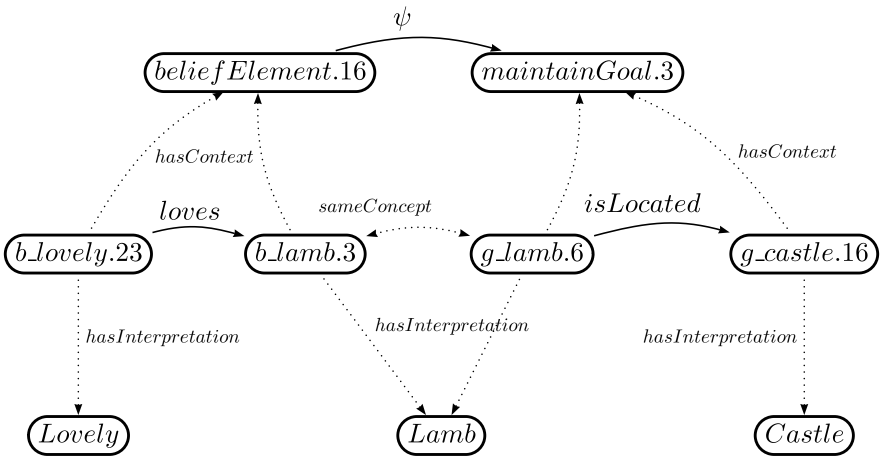

Figure 4.10: *"A belief that Lovely loves the lamb, psychologically causes a goal to keep it in the castle." version 2*

The actual entities are described using Individuals which would typically be Individuals as they are known in the actual story world knowledge, and a `hasInterpretation` property links "anonymous" Individuals to these actual story world entities.

When we want to know whether the Individual "Lamb" in the belief is the same Individual "Lamb" in the Goal, we will have to make sure this identification can be made. The symmetric property `sameConcept` is introduced for this purpose. At first glance, the `sameConcept` property sounds a lot like OWL's `sameAs` property, with the difference that OWL really treats two Individuals linked by this property as the same concept, meaning that the properties that hold for one Individual also hold for the other Individual and that is exactly what we *don't* want.

Having different Individuals for the same concept allows us to represent incomplete beliefs like a belief about a princess but not knowing that this princess is really Princess Lovely, if the `hasInterpretation` relationship is not present.

Of course, the fabula elements themselves can have a context as well, enabling a possibly endless embedding. Expressions like (2) are possible when Brutus' Goal to kill Amalia actually exist within the context of Amalia's Belief. And of course, the action of killing Amalia in its turn exists within the context of Brutus' Goal.

A simple algorithm for contextualizing knowledge can be laid out. Define KB as the knowledge base containing fabula knowledge, a new fabula element f (i.e. a Goal, Perception, etc.) and a sub network of knowledge k.

```
Add $f$ to $KB$
$\forall$ Individual $i \in k$
        create a new Individual $i'$ with a unique name
        assert (hasContext $i'$ $f$)
        assert (sameConcept $i$ $i'$)
        if interpretation $int$ of $i$ is known then
                assert (hasInterpretation $i'$ $int$)
$\forall$ triple (rel $i$ $j$) $\in k$
        assert (rel $i'$ $j'$), where $i'$ and $j'$ are the
            contextualized versions of $i$ and $j$
```

4.5 Example: Plop story

An example of the fabula of a simple story can be seen in Figure 4.11. It sketches the fabula of a gnome Plop being hungry, and having a belief that there is an apple in the house, which causes him to have a Goal to eat that apple. He succeeds by taking the apple, then eating the apple. Seeing that he ate the apple and then believing he did, leads to a positive Outcome which resolves Plop's Goal.

Of course, this fabula describes a very simple story. The fabula is created manually and is represented in OWL using the fabula ontology. Figure 4.11 does not show all relationships, but gives an indication of the Individuals and relationships that form the fabula knowledge about the story.

The `hasContext` relationship is used several times in this example, for instance to express the contents of the BeliefElement Individual `belief_1` (The fact that the apple is located in the house exists within the context of Plop's BeliefElement, in other words: Plop believed *that the apple was located in the house*), or the contents of the Goal Individual `goal_1` formed by an Action (The Action Individual `g_eatApple` of eating the apple exists within the context of Plop's Goal, in other words: Plop wanted *to eat the apple*). An Action is also used to express the contents of the Perception Individual `perception` (Plop perceived *that he ate the apple successfully*). The `hasInterpretation` relationship links the `b_apple` and the `b_house` Individuals that are used in Plop's belief, to the 'real' apple and house Individuals.

An example of the output of the Narrator that is being developed by Slabbers (2006) at the time of writing can be seen in Figure 4.12. It uses the fabula of the Plop story to generate a dutch text representing the story. The causal relationships of the fabula can be found in the structure of the text. Efforts are also being made to visualize the story in the form of a movie that takes the same fabula knowledge as a basis.

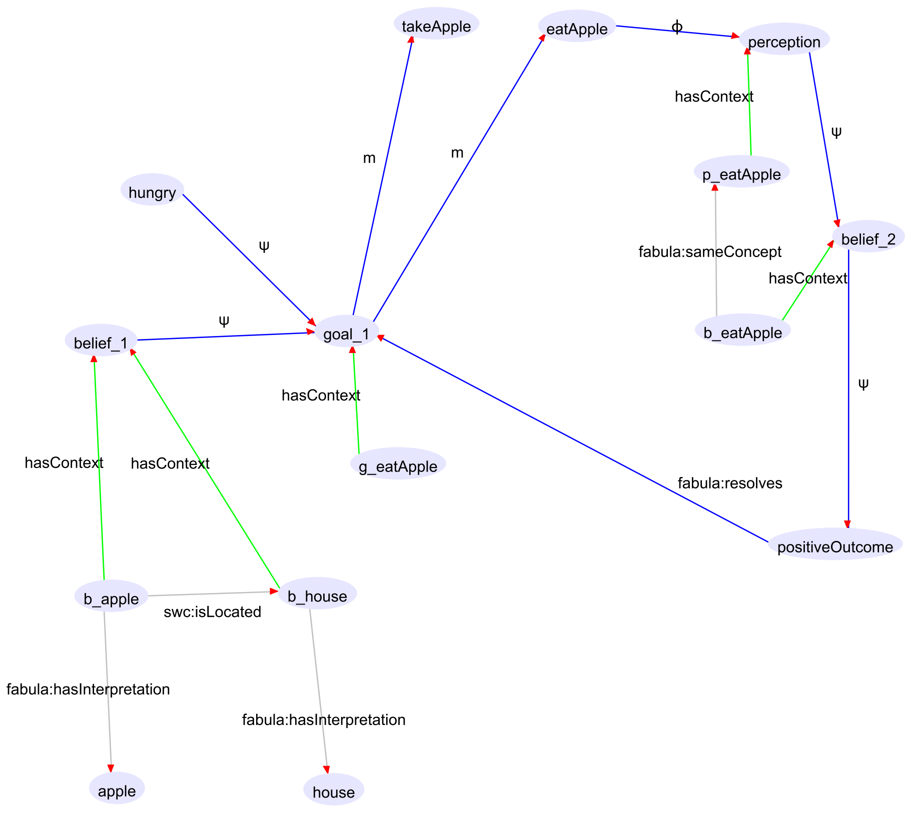

Figure 4.11: *Fabula of Plop story*

Er was eens een kabouter, die Plop heette.
Hij had honger en dacht dat er een appel in een huis lag.
Daarom wilde hij de appel eten.
Nadat kabouter Plop de appel op had gepakt, at hij hem.

Figure 4.12: *Narrator output of Plop Story, taken from (Slabbers, 2006)*

4.6 Conclusion

This chapter has tackled the first research question. It has discussed the design of a structure which allows the Virtual Storyteller to capture the fabula of the emerging story by expressing Goals, Actions, Events, Outcomes etc. and their causes and effects. The use of a Storyline Goal has been suggested to extract a plot from the generated fabula. This plot can be used to not only tell the story, but also to decide how and when to steer the plot to generate more interesting stories. An ontology defining the fabula structure has been developed. The fabula of a very simple example story was created manually has been expressed using this ontology.

A solution has been described to represent the contents of fabula elements in Section 4.4.2, but this solution is not ideal. Using Named Graphs seems like a much better approach which is meant exactly for the purpose of placing knowledge in a certain sub network. It might be worthwhile to gain experience with it.

Chapter 5

Controlling the plot

Oracle: "I'd ask you to sit down, but, you're not going to anyway. And don't worry about the vase." Neo: "What vase?" Neo turns to look for a vase, and as he does, he knocks over a vase of flowers, which shatters on the floor. *Oracle: "That vase." Neo: "I'm sorry." Oracle: "I said don't worry about it. I'll get one of my kids to fix it." Neo: "How did you know?" Oracle: "Ohh, what's really going to bake your noodle later on is, would you still have broken it if I hadn't said anything?"*

— 'The Matrix' (1999)

A simulated virtual world with autonomous characters is sufficient to generate fabula that can be narrated. However, stories that emerge like this tend to be uninteresting, as Tale-Spin demonstrated. The reason for this is the lack of difference between what an author wants to reach with his story, and what the characters in the story want to reach for themselves. The characters will understandably not have any interest in generating conflict or difficulties for themselves. Of course it is possible that conflicts and difficulties emerge by coincidence, but relying on this to occur raises the question whether the stories produced will generally contain enough richness to be interesting.

A form of plot control must be used to bring about situations that are wanted from a plot perspective. This chapter will address the second research question:

RQ2 Which techniques can be used to control the plot whilst keeping the characters believable?

Different techniques are discussed without making any assumptions about when to use these techniques and for what reason. These questions will be answered in Chapter 6.

5.1 Current approach

In the current design of the Virtual Storyteller, a very simple plot control consists of two mechanisms: instilling episodic goals, and prohibiting actions that are not 'interesting' for the plot. Faas (2002) proposes to base the decision to prohibit an action on three things: (1) whether an action would fit the story grammar, or (2) similarity with previous stories, to ensure variation, or (3) enable input of a user like in improvisational theater. Rensen (2004) argues not to make use of the story grammar approach, but suggests episodic constraints which would form a different reason for prohibition of actions.

I would like to argue that control in the form of prohibiting actions is not a satisfying approach. The main reason for this is that such control is based on the assumption that by leaving out uninteresting actions, the actions that remain or happen instead, would form a more interesting whole. I believe this is an oversimplification. Another reason is that simply prohibiting an action because it's not interesting can shatter character believability. Say that princess Lovely is in desperation, looking for her lamb, and finally finds it on the other side of the river. If Lovely is then forbidden to cross the bridge to go to the lamb, because it's more interesting for the plot if the lamb gets away, it makes no sense to the reader why Lovely didn't try to get to the lamb using the bridge. For character believability to remain intact, an obvious action that does not happen should have a good reason to not happen. Instead of prohibiting the action, the Plot Agent should make the action fail with a reason, for instance, by making sure the bridge collapses before Lovely tries to cross it. The reason for prohibiting an action cannot simply be, that the action is not interesting for the story.

Another argument to prohibit actions has been to prevent a boring repetition of actions. Of course boring repetitions can occur. It is possible that Lovely wants to go outside and when king Mikura doesn't allow her to go, she screams, and screams, and screams, and screams until it becomes incredibly boring. However, when such an endless scream sequence happens, subsequent scream actions should not be prohibited simply because they are not interesting. It is not the responsibility of the characters to perform only interesting actions, nor is it necessary for generating a good story. Presenting the interesting parts and leaving out the boring parts is something that the Narrator or a similar presentation component should be concerned with. For instance, consecutive actions like the screaming can be summarized into "Lovely kept on screaming because Mikura wouldn't let her go."

5.2 The dilemma of control

A good story has a coherent plot and characters that are believable. However, the elements of plot coherence and character believability are often very difficult to unite. In a story, the reader assumes that every action of a character is motivated from that character's personal goals and desires. Characters play a double role though. They should act according to their own goals and plans to be believable, but they are also in service of the plot and their actions and reactions must somehow make sense in the context of the story. The characters shouldn't appear to be aware of the plot (Riedl, 2002). The dilemma between wanting the Character Agents to be autonomous and at the same time making them perform desired behavior is a difficult one.

In their research in Interactive Drama, Kelso et al. (1992) have carried out an experiment for the OZ project to investigate this dilemma. In this experiment, a certain story was played out by human actors, wearing a headphone through which they received directions from a human director who was steering the plot. This was an interesting experiment to investigate the interaction between actors and the director that was needed to make a story. One of the conclusions was that a director is needed: neither human actors nor current computer Agents are capable of keeping track of the global flow of the story. Another conclusion was that the actors should be led to a destiny but remain independent. It is an interesting question when and how the director steered the characters, and this project has provided a good impression of that. But there's of course a difference between autonomous human players and autonomous virtual characters.

There should be a way to transparently direct the characters. Too direct control could lead to inconsistent behavior and too indirect control could be ineffective in developing a plot. Blumberg and Gaylean (1997) identify four levels of control over autonomous characters:

1. The motor skill level, where the director wants the character to engage in specific realization level actions. These actions may be "in-character" (e.g. going to the kitchen) or "out-of-character" in the form of changing properties (e.g. changing the location of the Agent);
2. The behavioral level, where the director wants the character to engage in specific but complicated behavior, and trusts the character to do the right thing (e.g. saying "find something to eat");
3. The motivational level, which is not really directing the character but creating a predisposition to act in a certain manner (e.g. saying "you are hungry");
4. The environmental level where the director manipulates the character's behavior by manipulating its environment (e.g. saying "there is bread in the kitchen"). This can be used as a trick to guide Agents in the form of imagination. To get characters to a certain place, for instance, they can be made to believe that there's a desired object at that location, which is only visible for the Agent.

These levels can be used prescriptively (what to do) or proscriptively (what *not* to do) in the form of commands or weighed preferences. Commands can be primary commands, that are executed whenever resources for it are available, secondary commands that are executed if the Agent has nothing better or primary to do, or meta commands, in the form of recommendations of actions (i.e. "if you're hungry, go eat some bread in the kitchen").

I will focus on prescription and recommendations. Proscriptive control will not be focused on, for reasons stated in Section 5.1. Two techniques will be looked at. The first technique is directing the characters in the form of prescriptive control in the form of giving the characters goals or actions. The second technique is narrative mediation, which deals with changing the environment of characters rather than the characters themselves.

5.3 Directing the characters

The previous version of the Virtual Storyteller has been using episodic goals to direct the plot. Characters receive an episodic goal, and when they reach this goal, the next episode starts. These episodic goals are obviously not motivated from the character perspective, but from the plot perspective. The problem of giving a character certain goals, is that goal conflicts can arise. Some goals don't seem to need a specific cause to suddenly occur without apparent reason. For instance: "One day, Lovely decided to climb up the highest tower in the castle". Other goals do seem to need such a cause, because it raises questions as to why the goal arose. For instance: "One day, king Mikura decided to buy Lovely a lamb"[1]. To be believable, these goals should not interfere with goals that the characters have. They would either have to be chosen by the Plot Agent in such a way, that they make sense to the character in respect to what happened before, or they would have to be smartly incorporated with the characters' own goals.

Assanie (2002) has researched two types of goal conflicts for the purpose of designing controllable autonomous characters in a story: consistency conflicts and coherence conflicts.

Goals have a consistency conflict when their goals or components are not compatible with each other. Goals can be specifically inconsistent or generally inconsistent. Goals are specifically inconsistent when goal conditions conflict, when there are possible threats in goal achievement (i.e. if princess Lovely gets hungry she won't adopt the Goal of eating the lamb, since that conflicts with her Goal of keeping it as a friend). Goals are generally inconsistent if they are "out of character", not matching the character having them (i.e. Lovely will not adopt a Goal to slay a dragon).

Goals have a coherence conflict when they conflict in having a clear and logical structure between them and in being understandable in and of themselves. If Lovely first wants to have a friend, and then later on suddenly *doesn't* want to have a friend, this is not very coherent unless the emotional process in between (i.e. Lovely gets hurt by the children) is made clear. Both goal switches and lack of goal switches must seem rational.

Work on consistency and coherence conflicts as done by Assanie (2002) is not yet extensive enough to be used as a way to direct Agents by giving them goals and requiring them to make these goals coherent and consistent. We need to keep the characters responsible for making their own goals according to their own emotional model. Instead of prescribing Goals and Actions, it is therefore better that the Goals and Actions that the Plot Agent invents for the characters take on the form of suggestions or preferences, where characters will only take on these suggestions if they can be made consistent and coherent.

Two methods will be discussed that can be used to solve goal conflicts: improvisation by the characters to make the goals consistent with their own motivation, and manipulation whereby the Plot Agent takes even more control and also prescribes or suggests a good motivation or causation for the goals. Consider the following example. The Plot Agent wants a dramatic event to be happening in the forest and has to make sure that Lovely goes to the forest. Lovely has to have a personally motivated reason to go there. She could go there because she wants to pick some berries. The berries would form a detail necessary for the motivation. Either Lovely has to improvise this by inventing the reason and the berries, or the Plot Agent has to invent the motivation as

[1]Further research should determine what this difference consists of, which Goals can be introduced without problem and which Goals need a motivation.

well and uses this to manipulate Lovely to go there.

Because a Goal is suggested by the Plot Agent for a reason (i.e. to enable a planned dramatic event in the forest), it might not feel entirely coincidental that Lovely went to the forest. This is an interesting aspect since Aristotle (trans. 1907) concluded that the dramatic effect in tragedy is heightened when tragic events are not entirely coincidental, but give a certain sense of inevitability. An approach of suggesting Goals that do not originate in the characters could produce this effect.

5.3.1 Improvisation

When a goal is suggested, a character might have to improvise to make it believable. Improvisation is the key mechanism in many character centric story generation systems (Riedl, 2002). The less autonomy is left to the characters, the more they need to improvise to stay believable as autonomous characters. An important conclusion from the OZ project was that when the actors knew the course of the story, they were much better at their acting. The director could then focus more on the smooth unfolding of the plot, than on “leashing” the characters. But of course, that way the role of the character shifts to that of an actor.

Princess Lovely can think of picking berries as a way to motivate going to the forest. She requests berries in the forest and when it doesn’t lead to an inconsistency[2], the Plot Agent adds these berries to the forest in the Story World. This enables Character Agents to fill in the details purely based on their own character. A princess would go to the forest to pick berries, but a lumberjack would go there to chop some wood. The Plot Agent can leave the details up to the Character Agents and focus on the higher level of plot structure.

Improvisation can also be used when the means to reach goals in any given state of the world are not sufficient. To prevent the characters being limited by a preset story world, certain additional items or knowledge about the world might be needed. This has a lot of correspondence with Improvisation Theater. The director controls the actions or goals, and the characters are responsible for their own believability. Improvisation also makes the world state very flexible since objects are added according to what happens in the story. A disadvantage is that improvisation takes the Character Agent out of its role as autonomous living character that is not aware of the fact that the world it lives in is fictive. But this disadvantage is merely a matter of principle.

5.3.2 Manipulation

Another solution is that the Plot Agent reasons about a motivation or causation for the prescribed goal and instills this motivation in the Character Agent. Manipulation makes for a flexible world state just as well as improvisation but now the Plot Agent is responsible for the believability of the characters. Although it gives the Plot Agent even more control over the story, this is not a clean solution. Another disadvantage is that the Plot Agent should be careful with

[2]For instance, if at this location a very hungry person died because he couldn’t find any food, it might be inconsistent when later on in the story, there are suddenly berries at this exact same location.

this manipulation since it might be very inconsistent when characters suddenly 'know' things without a good explanation. Manipulations should be done in the form of common sense knowledge only. It is plausible that there are berries in the forest and that Lovely knows this. But it is not very plausible when Lovely suddenly knows that her lamb is in the forest and that that is her reason for her to go there. A perception is expected by the reader, for instance a guard telling Lovely that they have seen the lamb in the forest, or Lovely hearing a bleating noise coming from the forest.

Manipulation and suggesting a goal or action should go hand in hand. Control in the form of manipulation alone is too loose. The reason is that if only the motivation is given and not the suggested goal, this could lead to a series of failed manipulations. Consider the following example. The Plot Agent wants Lovely to go to the forest, and invents two reasons for Lovely to want to go there: (1) she is hungry, and (2) she wants to go for a walk. When the Plot Agent prescribes these motivations to get Lovely to the forest without suggesting the goal itself, story fragments like this could arise:

> *Lovely became hungry and decided to take an apple from the bowl in the dining room. Then, she felt like taking a walk so she went to the park and had a nice time. Then, she became hungry again and went to the forest to get some berries.*

It turned out that Lovely found her own ways to act out her wishes. This fragment contains two failed manipulations that now suddenly feel very forced.

The problems with inconsistency and failed manipulations make the use of manipulation rather complicated. I propose to not make use of this technique for the time being and choose for improvisation instead. Improvisation seems to be much more flexible and its biggest advantage is that it allows the characters to fill in the details of the stories in their own personal ways.

5.4 Narrative mediation

Narrative mediation is a technique that leaves as much autonomy to the characters as possible. Instead of influencing the characters directly, the world is influenced so that the effect of the characters' actions are different from the effect they intended such that the effect is more in line with the planned plot development.

If a director or plot monitor has a certain plan about how the story should evolve, and the characters are truly autonomous, like the human players in interactive drama, narrative mediation is very powerful according to Young (2002). He identifies three possible relations between user actions and such a story plan. An action can simply be constituent to the plan. There are also actions that are not according to plan but don't disrupt the plan, which are called consistent with the plan. Finally and most interestingly, an action could be exceptional to the plan. It would disrupt the plan and there are two possibilities:

- Allow the exception and restructure the plan accordingly;
- Prevent execution of the action, to force the player to act according to the plan of the author. This is called narrative mediation. This gives

the player less feeling of being in control, but the performance cost of re-planning the plot doesn't have to be paid.

Young (2002) gives an example of narrative mediation for interactive drama. If a player has a gun and tries to shoot an important character that is not supposed to die, a narrative mediation could be to make sure the gun turns out not to contain any bullets, or the shot misses.

Narrative mediation is a technique that seems very suitable for use in the Virtual Storyteller. We model this by modeling different effects of Actions in the form of (unforeseen) Events, which are caused by these Actions. Boring Actions, or Actions that lead to an unwanted plot situation, can fail. For example: Lovely wants to play with the children and has the plan to go outside and play with them. It would be boring if she was able to just go to them; we can mediate this by letting the "open door" Action fail because the door turns out to be locked, so she needs to ask the king for the key.

5.5 Proposed solutions

To generate good stories, a world will be simulated where a whole range of events takes place. Only part of it will be told in the form of a story. This is an interesting approach when the assumption is made that a plot can be extracted from these events. This assumption has been made plausible in Section 4.1.2 where I proposed to use the Storyline Goal and coming to a better understanding of the concept of affiliation.

If in future designs the Character Agent can be replaced by a human player, realistic fabula are needed because having human players in the story requires much more emphasis on consistency and believability. So creating a story by generating realistic fabula is a good approach if we want to make a transition to interactivity in a later stadium.

The plot control for this kind of fabula should suit the realism. I therefore propose direction of a very lenient nature. Instead of forcing the characters to follow a certain plot development and compromising in believability and consistency, I propose to merely influence the emergent narrative to stimulate a good plot. Furthermore, narrative mediation is interesting because it leaves most autonomy to the characters.

We will use four ways to influence the story:

1. Generating events to mediate the plans of characters;
2. Influencing the perceptions of the characters;
3. Changing the setting;
4. Directing the characters by suggesting goals or actions.

The first three ways leave full autonomy to the characters, keeping them totally independent of their role as a character in a story. Events can be instigated to change the course of the story in general, the lives of the characters, but can also be used very specifically to directly change the outcome of character actions, as discussed in Section 4.3.1. Influencing the perceptions will need to be done with care as believability is easily shattered. When Lovely walks past the fire and doesn't see the children or the lamb because the Plot Agent doesn't want Lovely to find it just yet, readers are going to think Lovely is a bit blind.

Changing the setting of the story can be initiated by the Plot Agent when needed, or by the Character Agents in the form of improvisation. I believe that when characters are allowed to improvise, stories can become more interesting since the story world will take shape according to the goals and plans of the characters. Obviously characters shouldn't get full freedom in this, but their improvised state of the world can for instance be requested from the Plot Agent that can then choose to allow or disallow the improvisation. The decision to allow or disallow has only a small concern with believability since prohibiting the improvised world state will leave the world just as it was. However, attention has to be paid when allowing world changes, since allowing such a change could clash with consistency.

Directing the characters is kept lenient to keep the Character Agents autonomous. The Plot Agent can suggest Goals or Actions to the Character Agent, and the Character Agents can improvise to incorporate the Goals and Actions in a believable way. A suggestion can be interpreted like: "do what you want, make your own goals and plan your own actions, but if you can, give preference to the goals and plans that contain the suggestion". Suggestion is a very open way of plot control and can for instance be implemented using Blumberg and Gaylean's weighed preferences. Of course, a prerequisite of this kind of control is that a character indeed has the ability to form multiple goals and plans, and has the luxury to choose. On top of that, the notion of 'if you can' is a bit vague. If the Plot Agent suggests to Lovely who wants to find her lamb, that she should give preference to a sub goal of going via the adjacent kingdom, she *can* do this but obviously this would not be very believable, whereas going via the nearby forest might be incorporated as a place to look first, for example.

5.6 Conclusion

This chapter has tackled the second research question. I have identified four techniques that can be used to influence the story in a very lenient way:

1. Generating events to mediate the plans of characters;
2. Influencing the perceptions of the characters;
3. Changing the setting;
4. Directing the characters by suggesting Goals or Actions.

These techniques leave the characters very autonomous, which keeps them realistic and believable. They should be allowed to improvise to allow for a story world that is more adapted to the goals and plans of the characters, and to make it easier for them to adopt suggested Goals or Actions.

Chapter 6

Making creative decisions

Tank: "Okay, so what do you need, besides a miracle?"
Neo: "Guns. Lots of guns."

— 'The Matrix' (1999)

Where Chapter 5 has described techniques that can be applied to change the course of the story, the question remains how these techniques should be used to bring about desired changes. There are several decisions that the Plot Agent will have to make in this respect.

The third research question is:

RQ3 How can the Plot Agent use the plot control techniques to reach plot goals?

One of the main pillars of a good storytelling system that claims to produce new and interesting stories is definitely the ability to be creative. In order to make creative decisions about how the story should evolve, a certain model of creativity has to be developed. This chapter consists of three parts. First, I will discuss how creativity and creative problem solving can be automated using Case Based Reasoning (CBR). A general understanding of CBR is useful to understand the second part, which will describe the decisions that the Plot Agent faces in controlling the plot. Finally, I will elaborate on the design of a creative problem solver described in (Vromen & Bloom, 2005). The design uses a form of CBR and, as we will see, can aid in making the decisions that the Plot Agent faces. Having supervised and contributed to the ideas of this design, I will attempt to place the creative problem solver within the context of the Virtual Storyteller, and elaborate on the design to bring it closer to an implementation.

6.1 Case Based Reasoning

Creativity is considered by many to be inherently human. To be creative is to *originate* something. It is a bit difficult to imagine how a computer, which merely follows the programs given to it, can originate something. Yet, several attempts at formalizing creativity and displaying automated creative behavior have been made. Most of the systems attempting creativity use Case Based Reasoning (CBR) in one way or another (Vromen & Bloom, 2005). This section will describe Case Based Reasoning which is used in the creative problem solver discussed in Section 6.3.

Case Based Reasoning is a process of making analogies based on the psychological model of episodic memory. It is believed that in human beings, creativity is the result of cognitive processes that bring together pieces of old knowledge in new ways (Turner, 1994). In CBR, this knowledge is captured in the form of a set of cases that describe a problem, and a solution to that problem. To reason about solutions for problems, the collection of cases is explored in order to find similar situations. The solutions in those situations can be adapted to form a solution to the problem that needed to be solved. Creativity is thus considered to be an extension of problem solving.

The Case Based Reasoning process is usually done in four steps, also called the four R's (Cunningham, 1998):

1. **Retrieve** cases from memory that are relevant to solving a target problem;
2. **Reuse** the solution found by adapting it to the target problem;
3. **Revise** the reused solution, if necessary, based on a test against the real world (or simulation)[1].
4. **Retain** the successful solution by storing the resulting experience as a new case in memory.

It seems that CBR has been applied to storytelling or storytelling-like systems with satisfying results (Mueller, 1987; Turner, 1994; Fairclough, 2004). Minstrel (Turner, 1994) has demonstrated the possibilities of CBR to apply creative problem solving to generate simple stories in the King Arthur domain.

Minstrel defines storytelling goals as problems to solve, and retrieves cases that form a solution to these problems. Usually, CBR retrieves solutions by comparing the problem to solve with similar problems in the case base. For storytelling, it is a difficult to define whether two problems are similar, because as we will come to see in Section 6.4, this is a very knowledge intensive and context dependent quality. Where the somewhat more regular application areas of CBR, like case-based design, planning, diagnosis and information retrieval, use similarity measures to retrieve 'similar' cases that are then reused, Minstrel uses a very active approach to similarity using TRAMs (Transform-Recall-Adapt Methods) to *first* transform the problem to similar problems and *then* look for matching cases. The assumption is that transforming a problem using a TRAM leads to a new problem that is similar to the original problem. It seems easier to design methods to transform problems into similar problems than it is to calculate the similarity between two problems.

A TRAM consists of the following steps:

1. Transform the target problem into a similar problem;
2. Recall past solutions that match the similar problem;
3. Adapt found solution to target problem.

[1]The Reuse and Revise steps are often seen as a single Adapt step.

Inherent to creativity is the fact that creativity errors can arise. A child having a limited concept of objects and holes could have an idea that 'objects fit through holes' which is of course erroneous when a square peg has to go through a round hole. The idea is good, but reality proves that the solution does not work. Therefore we argue that to minimize creativity errors, a distinction should be made between the creative problem solver and the domain-specific model for which it solves problems. Creatively found solutions should be tested against this model, and failed solutions should either be repaired, or discarded. Minstrel does not keep track of such a world model; all knowledge is stored in the cases of the case base.

A further investigation into the strengths and weaknesses of CBR is done by Cunningham (1998). He concludes that CBR can decrease the amount of knowledge that has to be engineered to model the knowledge domain, and that it can greatly help to model causal interactions in the knowledge domain, reducing the need to capture knowledge using 'first principles', such as 'apples grow on trees', 'it is never sunny during the night'. These conclusions seem advantageous to the design of a system that can invent a vast amount of different stories.

6.2 Decision making in the Virtual Storyteller

Vromen and Bloom (2005) have come to a design of a creative problem solver for the Virtual Storyteller. Before I discuss this design, I will first discuss where such a problem solver can be useful in the Virtual Storyteller. I will identify what decisions must be made and how a creative problem solver can help in making these decisions. We will see that such a problem solver is not only useful when making plot control decisions, but can also greatly aid decisions that the Character Agents and the Narrator have to make[2].

6.2.1 Plot decisions

Section 5.5 describes a number of different ways for the Plot Agent to influence the plot. To recapitulate, influencing the plot can be done in four ways:

- Generating events to mediate the plans of characters;
- Influencing the perceptions of the characters;
- Changing the setting;
- Directing the characters by suggesting goals or actions.

Decisions about Event generation are based on the question which Events will contribute to an interesting plot. As discussed before, these are for instance Events that thwart or help a characters Goal. One can imagine that Event-Outcome cases can be expressed using the fabula ontology. We can for instance model how certain Events can eventually cause a negative Outcome for a certain Goal. A problem to solve could for instance be specified as: which Events

[2]The decision making process itself is not discussed in this thesis. An assumption is made that each component contains some form of planning for decision making that can specify problems to be solved by the creative problem solver.

eventually cause a negative Outcome for Goal G? The solution to this problem can then be used to make a decision to initiate an Event.

The second area of concern is perception management. What characters perceive is something that usually follows certain guidelines, but the Plot Agent can deviate from these guidelines based on narrative considerations. For example, when a thief sneaks into the castle and tries to not be seen by the guards, it makes sense for the Plot Agent to decide whether to have the guards discover the thief or not, based on what direction the plot should go. The problem specification is about what the effect of Perceptions will be.

A third area of decisions is how and when to change the setting in the course of the story. Setting management is a powerful tool to guide the plot; deciding whether or not to honor character requests originated by their improvisation, or changes in setting initiated by the Plot Agent itself. Casting new characters could eventually be part of this as well. The Plot Agent can determine what the effect will be of introducing new knowledge; certain cases will become applicable for characters as solutions, or for the Plot Agent as requirements to introduce certain Events. For instance, to thwart an Action of a character to go to the forest, the Plot Agent can introduce a bridge on the way to the forest in order to be able to introduce an Event of the bridge collapsing.

Finally, plot decisions are needed about the suggestions that the Plot Agent can give to the characters. Based on cases that describe the results of Actions, for instance Events that can follow from them, the Plot Agent can reason about which results are desirable, and suggest Actions to reach these results. Similar decisions need to be made for Goals.

6.2.2 Character decisions

I believe that CBR can make the Character Agent a lot more intelligent and believable than would be possible with rule-based reasoning. I will illustrate this claim by discussing three applications:

- Creative action planning;
- Interpretation of perceptions;
- Event appraisal.

The question asked in creative action planning is: what actions does a character perform or what sub goals does it take on to achieve its goals? On top of a 'normal' task planning algorithm, creativity can be brought into play when action planning fails due to incomplete knowledge or when less straightforward solutions are desired or needed.

Interpretation of perceptions is a very important aspect of believable characters. How does a character interpret what it sees? Case Based Reasoning can be used to relate perceptions that the Character Agent receives to similar perceptions described in the form of cases in the case base that also specify the BeliefElements that followed in these cases. This way, a character's belief world can be enriched by a case base of experiences. Maybe even more important, this makes it possible for characters to reason about other characters' beliefs. They can make plans to make other characters believe certain things. For instance, if Lovely enters the forest and hears a bleating noise, and she consults the case base, she might find a case where a lamb bleats causing a shepherd to have a Perception of a bleating noise. She can then adopt the belief that there is a lamb – her lamb – in the forest making the bleating noise. If the story contained a villain that wanted to lure Lovely and imprison her, the villain could imitate a bleating noise just to put Lovely on the wrong track and make her believe her lamb is there, so that she will walk straight into his arms.

Event appraisal answers the question how a character responds to incoming information. Event appraisal can be based on reactions to Perceptions stored in cases. This could be a direct form, where cased in which Perceptions cause Internal Elements (i.e. a case specifying that seeing a monster makes one afraid) are used to generate similar Internal Elements for similar perceptions. Event appraisal can also take on a more complex form, where cases are matched against the current situation to make expectations of what is going to happen, leading to hope, fear, worry, etc. For example, based on a case where a monster has killed a princess, a character can become afraid when it sees a monster.

6.2.3 Presentation decisions

I will illustrate that even a presentation component like the Narrator will be able to benefit from a case base, for instance in the following areas:

- Reader expectations.
- Exposing character flaws.

These two aspects are closely related. Using cases of what happened before in similar situations, reader expectations can be derived under the assumption that the case base is extensive enough to capture these expectations. We can make sure the case base for the Narrator Agent is different from the case base for the Character Agents. For instance, if the Narrator has a case at its disposal where a princess is hurt by someone's actions and asks for an explanation for these actions, and the Character Agent doesn't have this case at its disposal, it allows us to narrate using styles like:

> "Lovely felt sad and hurt by the children *but instead of asking them for an explanation*, she went home and became bitter."

which obviously illustrates a discrepancy between reader expectations and character actions which forms interesting external emotions (see Section 2.2.1) which I think are very important for engaging storytelling. These discrepancies are often meant to expose character flaws or trait shifts. For instance, in the case of character flaws, this discrepancy can be modeled by having certain obvious cases in the Narrator case base that are unavailable to the Character Agent, allowing for situations where a character won't think of obvious solutions that the reader does think of, or doesn't react in a way that the reader would have. In the example above, asking the children for an explanation why they killed her lamb is a stronger, more desired response and Lovely's reaction exposes a character flaw.

6.3 Design of the creative problem solver

Clearly, different components of the Virtual Storyteller can use creative problem solving, each for their own knowledge domain and based on their own goals. As discussed in Section 6.1, a strict distinction will be made between the creative problem solver and the world model for which it solves problems. The creative problem solver has no knowledge of the state of the story world it solves problems for. In this section, I will refer to any part of the Virtual Storyteller that uses creative problem solving as the 'user'. Creative problem solving becomes a cooperative process between the user and the problem solver. See Figure 6.1.

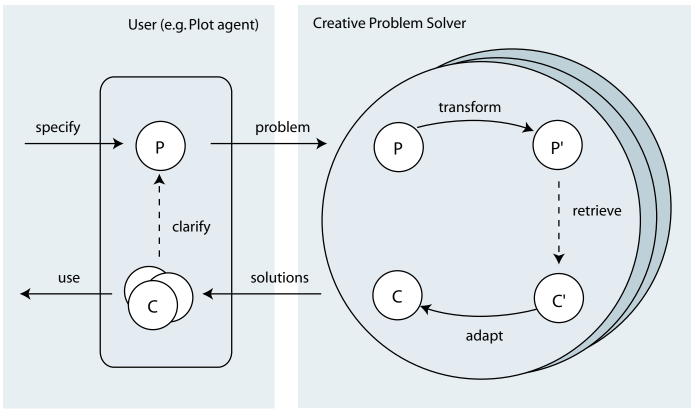

Figure 6.1: *Creative problem solving process*

The user will specify a problem definition for which the problem solver will try to find a solution. For instance, if the user is the Plot Agent trying to achieve a negative Outcome for a certain Goal, the problem specification would be the question: which Events lead to a negative Outcome for this Goal?

This section will define what a problem and a case in the case base consist of, describe the cyclic process of problem solving and show how a user can specify problems and use solutions.

6.3.1 Definition of Problem and Case

Following Vromen and Bloom (2005), a problem P is a tuple $< Pat, Con >$ where Pat is the pattern space defining what a solution must be like, and Con defines the constraint space in the form of information that should not occur in the solution. Section 6.3.2 will illustrate this with an example.

A Case C has the following requirements:

- C demonstrates a *concept*. A concept could for instance be 'hiding from a threat' or 'flying over an area to search for something'. The expression of a concept could be seen as an example of a problem and a solution to it. The problem is for instance the threat, and the solution is to hide. Of course, there can be many cases demonstrating the same concept.
- C is *context complete* with regard to its concept. Cunningham (1998) states that the case representation must capture the predictive features of

a problem. Applied to storytelling problems, we define a context complete case as a case that contains all the elements that are necessary for the case to be viewed as 'logical' by a human reader, regarding the concept it is supposed to express, and expresses nothing more than that.

Section 6.4 will elaborate on these requirements where they become relevant for constructing cases. For now, this explanation will suffice.

6.3.2 The problem solving cycle

Given a certain problem statement, the creative problem solving cycle consists of three steps similar to Turners TRAMs:

1. Transform a problem P into similar problems P';
2. Retrieve cases C' that match P';
3. Adapt C' to form a solution to P.

I will first discuss how to retrieve a Case according to a given Problem, and then look at how these Problems and Cases can be transformed and adapted[3].

Retrieving cases

Consider a Case C and a Problem P. Both C and P can be expressed using OWL using the fabula ontology as designed in Chapter 4. Both Pat_p and Con_p are sets of OWL triples containing uninstantiated variables.

Consider the following example. Say that the Plot Agent wants the lamb to be dead, because that is dramatically interesting. It has reasoned that someone eating the lamb would result in the lamb being dead, and has considered the children to be a good candidate for this, as Lovely wanted to be friends with them. So, to finish its dramatic plan, the Plot Agent looks for a motivation for the children to adopt the Action of eating the lamb. The problem becomes: "find some Goal that motivates the children to eat the lamb".

This example problem can be expressed graphically as in Figure 6.2, where uninstantiated knowledge is displayed using question marks. The problem can be expressed in OWL as follows. Note that variables are used to describe uninstantiated knowledge:

```
(rdf:type i_children Children)
(rdf:type i_lamb Lamb)
(rdf:type ?A fabula:Eat)
(fabula:agens ?A i_children)
(fabula:patiens ?A i_lamb)
(rdf:type ?G fabula:Goal)
(fabula:motivates ?G ?A)
```

We say P *matches* C when there are bindings for the variables in both Pat_p and Con_p such that $C \vdash Pat_p$ and $C \nvdash Con_p$. The pattern space illustrated by Figure 6.2 matches the Case illustrated by Figure 6.3. Obviously, the more

[3]The proposed design does not use a retain step. How this step can be implemented needs further looking in to.

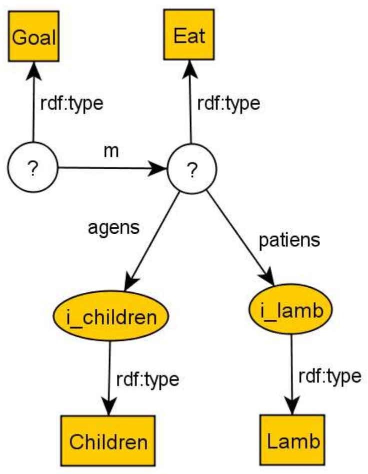

Figure 6.2: *Some Goal motivates the children to eat the lamb*

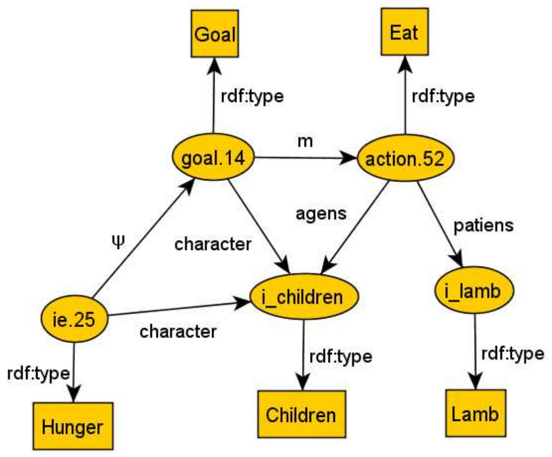

Figure 6.3: *Hunger causes a Goal that motivates the children to eat the lamb*

specific the pattern space is, the more likely it is that the retrieved case forms an adequate solution to the problem. The more general the pattern space is, the more likely it is that the creative problem solver can find a matching case. For instance, the very general pattern space specifying

The children have a Goal.

or specifically:

```
(rdf:type ?G fabula:Goal)
(fabula:character ?G i_children)
(rdf:type i_children Children)
```

would match the above case as well. It is not very likely that cases retrieved using a too general problem specification form a good solution to the problem that is being solved.

Transforming and Adapting cases

When we want novel solutions, we use creativity. When a Problem P doesn't match any case, we *need* creativity. Turner (1994) defines TRAMs to model creativity. In Vromen and Bloom (2005), similar methods are defined which are called creativity heuristics.

A creativity heuristic has three parts.

Match defines whether the heuristic can be applied to a Problem P;

Transform defines how P should be transformed into a similar Problem P';

Adapt defines how a Case C' that matches P' should be transformed to a Case C that forms a solution to P.

In Section 6.5 some possible heuristics will be discussed.

6.3.3 Using the creative problem solver

For creative problem solving, the task of the user is to construct a pattern and constraint space to come to a problem P, which is then used as input for the creative problem solver. The creative problem solver is then responsible for retrieving a set of (possibly transformed) cases that are assumed to solve P. It is up to the user to utilize any of these solution cases:

- as a direct solution to P when the solution fits the context of the problem;
- as a solution that can be repaired into a different solution to P that fits the context of the problem;
- to transform P into a new problem P' consisting of the old pattern space and updated requirements to the constraint space.

Figure 6.1 shows the first two options as one '`use`' arrow, and the last option as a '`clarify`' arrow. When solving problems, the Case Base has no concept of the context of the problem. A found solution might not work in the current context. The found solution should be tested against the Story World and possibly adapted, or constraints should be added to the constraint space to further specify the problem.

For example, consider the following problem. The Plot Agent gives Princess Lovely a suggestion to go to the castle tower. She improvises to make this believable. She uses the creative problem solver: a case where a little boy left a toy in his bedroom and goes there to get it is transformed and applied to this situation. Lovely improvises that she left a toy there. But if the Plot Agent does not agree to introduce a doll in the castle tower, another solution has to be found based on a new problem with an added constraint: *no toy in the castle tower*. Or maybe Lovely knows there is no toy in the castle tower, but does know that there is an apple there. She can repair the solution by substituting the toy in the solution by the apple.

The constraint space can be used to specify constraints that are known in advance. If Lovely knows she is not allowed to go to the castle room, her problem becomes: how can I go to the castle tower *without my father noticing it*?

6.4 Cases

The case base forms the basis for every solution generated by the system. However, it is difficult to determine what a good amount of cases is, or which concepts should be expressed exactly. This will be a process of trial and error. For now, the focus has been on how to construct cases expressing a concept.

So, what does it mean for a case to express a concept? Vromen and Bloom (2005) give several examples of cases that express a concept. Most of these examples express episodes. As seen in Section 2.1.2, an episode is considered to be a Goal-Attempt-Outcome (GAO) combination, where Attempts can be seen as a set of Actions. Most of the examples contain a GAO or GAO-like structure:

- The princess hides under a bush to avoid the dragon, and succeeds (G-A-O)
- The princess climbs out of the window to escape from the castle, and succeeds (G-A-O)
- The dragon eats the princess to kill her (G-A-O)
- The hungry dragon eats the princess to satisfy his hunger (IE-G-A-O)
- The princess shrieks because she is afraid (IE-A)
- The hungry dragon flies over the fields in order to find the princess and eat her, but doesn't find her (IE-G-A-O)
- The princess slams the door, accidentally waking Brutus (G-A-O + E)
- The knight kills the dragon with a sword (G-A-O)

A concept heavily relies on the problem that it is supposed to solve. GAO cases solve the problem of how to bring a Goal to a certain Outcome. When an Event is added, the focus of the case is more on what can happen instead of the normal Outcome. But concepts can also be about expressing the reaction to a perception (P-IE), or about which Actions are a reaction to a certain emotion (IE-A), etc. An extensive analysis is needed as to what exactly entails such a concept, and experience with which concepts make useful cases is needed.

Vromen and Bloom (2005) describe a methodology to construct context complete cases based on a sentence that describes a concept. However, the requirement they pose for a case to be context complete brings about the Qualification Problem that was also faced when designing the Action ontology. This problem is concerned with the fact that in order to make a case fully context complete, there is an infinite amount of context that should be included in the case. I will illustrate that with the following example.

A case illustrating the concept 'escaping a room' could describe a princess climbing out of her bedroom window to escape. But if this case is applied as a solution to escape a castle tower room, obviously this is not a good solution since the room is far too high to climb out of. The case is not context complete, since it requires the room to be at or near ground level. According to Vromen and Bloom, this information should be included in the case to make the solution valid and context complete.

But should we then also specify that the window is big enough to climb through, that there are no guards outside the window, and that the princess is not tied to her bed? Theoretically, a Case is never context complete since new exceptions can always be found. On top of that, having such very specific cases weaken the power of the case based reasoning process. Maybe the constraint on cases to be context complete should be relaxed. Many of these constraints should follow from a test of solutions to the state of the world. Only the very important and logical context should be included but the decision what is important and logical and what is not, is a bit arbitrary.

6.5 Creativity heuristics

Creativity heuristics provide a way to transform a problem into a similar problem, and a way to apply a reverse transformation to a found solution. When searching for a solution, the creative problem solver applies all available creativity heuristics to the problem specification. Problems can undergo a series of these transformations in succession. I agree with the assumption of Vromen and Bloom (2005) that too much transformations will make the solution too unsuited for the problem to solve. Therefore, the number of successive transformations should be limited.

Turner (1994) states that creativity is prone to errors. Of course, the assumption that a limited amount of transformations keeps the solution applicable for the problem at hand, might fail. If we generalize a dragon into an organism, we might find a creative solution where a knight eats a princess to satisfy his hunger. Obviously that is not a good solution. But this is the price we pay for creativity and could even lead to very funny stories since the reasoning behind such a solution is clear, even though it is wrong.

A creativity heuristic in our design needs the following methods:

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

Transform defines how the problem space is transformed;

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

Minstrel has implemented a number of TRAMs and evaluated them in his system. Three of the heuristics that have proven to be most useful have been adapted for the problem solving domain of the Virtual Storyteller: Generalization, Transform Scale and Switch Intention.

6.5.1 Generalization

Generalization is probably the most commonsense heuristic one can think of. If a princess can hide under a bush, most likely any human being can.

The two elements that we can generalize are objects in the story world, and Actions. The generalization heuristic assumes that a problem definition will remain valid when one object or Action element is replaced by a generalization of that element:

Match. The problem should contain at least one Action or story world object.

Transform. Select one Action or object at random, and replace it by a generalized Action or object from their ontology hierarchies described in Sections 3.2 and 3.1, respectively. Only generalize an element one step up in the hierarchy.

Adapt. Identify the element from the recalled solution that fits into the generalized element slot created by the Transform step. Replace all instances of this element with the element that was replaced in the original problem.

Consider a fragment of the problem expressing "The princess runs away from the dragon". Examples of generalizations of this fragment (generalized element in italics):

1. "The princess *moves* away from the dragon";
2. "The princess runs away from *a monster*";
3. "*A woman* runs away from the dragon".

6.5.2 Transform Scale

Consider having to express that a dragon eats a princess in order to satisfy his hunger. The fact that the dragon is hungry, contributes to this case being context complete. The dragon needs to be hungry to eat the princess. But can the dragon also eat the princess when it is less hungry? Transform scale works on values that store state information, i.e. hunger, health, happiness, age. A character can for instance have a health of 14, or an age of 23. Whether a health of 14 is good, or an age of 23 is old, depends on the scale of these values. When cases contain these kinds of scale values, they are somehow important for the case. The Transform Scale heuristic is based on the assumption that a case with a slightly different scale value will still be a valid solution to a problem.

Match. The problem should contain at least one element with a scale value.

Transform. Select one of the scale values of one of the elements at random, and replace it with a value that has a slightly different interval. There are three ways to change an interval:

- Change left boundary, i.e. [4..8] becomes [5..8];
- Change right boundary, i.e. [4..8] becomes [4..9];
- Shift window, i.e. [4..8] becomes [5..9].

The change should be large enough to make a difference, but small enough to keep the problem similar. Vromen and Bloom (2005) propose a maximum difference of 10%, but empirical testing should provide a more accurate guideline.

Adapt. Set all instances of the scale value to the value that was replaced in the original problem.

6.5.3 Switch Intention

Switch intention is based on the idea that if something happens unintentionally, one can try to make this happen intentionally. Actions can cause Events that the Agent did not intent, expect, or failed to take into account. Cases describing this can be transformed into cases where these Events *are* intended:

Match. The problem should contain at least one Goal - Uninstantiated Action - Positive Outcome combination. In other words, the problem is about "what Action brings the Goal to a successful outcome?"

Transform. Using the Goal from the found combination, find an Event that achieves that Goal. Construct a new problem containing a different Goal, motivating an uninstantiated Action, unintentionally causing the constructed Event. The new problem is about "what Action can cause an Event that brings the Goal to a successful outcome?"

Adapt. Insert the original uninstantiated Action with the Action found in the retrieved case.

For example, Lovely wants to go play outside but needs to ask the king for permission. When she finds him, he is sound asleep in his chair. What can Lovely do to wake up the king? The Switch Intention heuristic is based on the premise that in stead of executing an Action to wake up the king in some way, maybe she can cause an Event that wakes him up. If the case base contains a case where a thief wants to close the door and therefore slams it shut, accidentally waking up the house owner, this case can be transformed using the Switch Intention heuristic so that Lovely will slam the door shut in order to wake up Mikura.

6.6 Conclusion

This chapter has discussed the third research question. A concept for a creative problem solver has been designed. This creative problem solver can be used by the Virtual Storyteller to find solutions to plot problems, character problems and narration problems. It can be used to determine how to reach plot goals by deciding which events should be introduced, which Goals or Actions should be suggested, etc. The creative problem solver has not been implemented yet.

Several creativity heuristics for the creative problem solver have been discussed but more should be developed in the future. Furthermore, a methodology to construct cases has been discussed. Using this methodology, a case base can be constructed. Adding new cases should be made easy because the contents of such a case base might vary a lot. It is therefore also important to have insight into what the different cases express. To test the behavior of the creative problem solver, insight in the solutions that the creative problem solver finds, and how these solutions are created, is needed.

Chapter 7

A new architecture for the Virtual Storyteller

"That's how it is with people - nobody cares how it works as long as it works."

— Councillor Hamann, 'The Matrix Reloaded' (2003)

The three research questions posed in Section 1.3 have been explored. A fabula structure to capture the emerging story has been designed, several techniques to control the plot have been identified, and a creative problem solver has been designed that can aid the decision making process. This chapter will combine these subjects and propose a new architecture for the Virtual Storyteller, and in particular the Plot Agent. First, I will propose a new global architecture of the Virtual Storyteller. Then, I will discuss two Agents of this design that are relevant to the subject matter discussed in this thesis: a World Agent that keeps track of the state of a simulated story world, and a Plot Agent which builds and influences the emergent story.

7.1 Global architecture

Kooijman (2004) suggests the use of a World Agent that will keep track of the story world as it is, the physical reality that all the characters live in. The Director in the previous architecture (see Figure 1.1) took on both this task and the task of directing the story. I consider these tasks to be conceptually different. Therefore, a strict distinction is made between an objective, physical representation of the story world, and Agents responsible for changing this world to make a story emerge. In the new Virtual Storytelling architecture, generating a story is done by four types of Agents: a World Agent, one or more Character Agents, a Plot Agent and a Narrator Agent. As the Character Agents and Narrator Agent fall outside of the scope of this thesis, only the World Agent and the Plot Agent will be discussed in more detail, as well as the communications between the Agents.

Figure 7.1 gives a global view of the new architecture. A global division of responsibility is made whereby the World Agent keeps track of the world

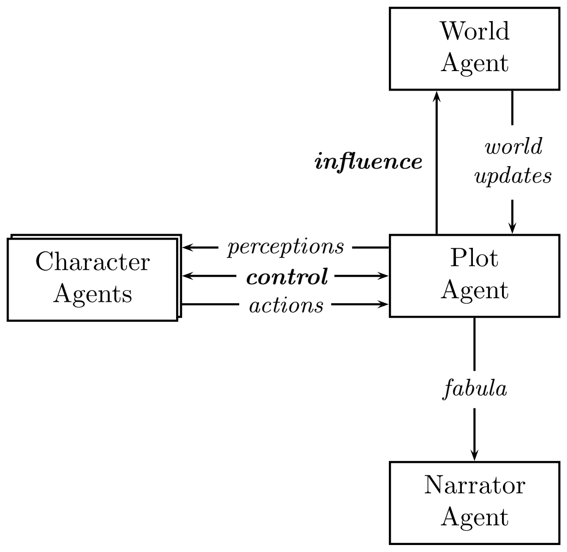

Figure 7.1: *Global design Virtual Storyteller*

as it is, and executes Actions and Events. The Character Agents are able to perceive parts of this 'real' world. They can react to it and perform strategic actions accordingly. The Plot Agent's responsibility is twofold. On one hand it maintains a sense of plot by keeping track of what happens, and building a fabula structure that can be used by the Narrator Agent to be transformed into a story. On the other hand it promotes interesting fabula, which is done in two ways: (1) by influencing the world and (2) by controlling the characters. How the Plot Agent exerts influence on the world, and how the Character Agents are controlled, is discussed in Section 7.1.1.

The advantage of this architecture is flexibility. There can be a World without anything happening in it. We can test and experiment with world models without needing to generate fabula for a story. Furthermore, a Plot Agent can theoretically function without characters. As far as the World Agent is concerned, the Plot Agent carries all responsibility for making things happen, and whether it does this in collaboration with Character Agents or with human users, is not interesting for the World Agent. Finally, the structuring allows full control for the Plot Agent on what Character Agents perceive and do, might it so choose to exert that control.

7.1.1 Influence and control

The Plot Agent influences what happens in the world. The `influence` arrow in Figure 7.1 can be split up in three ways of influence:

1. It sends Actions to perform. These Actions typically originate in the Character Agents and the Plot Agent just passes these on to the World Agent;

2. It sends Events that should happen. Events are similar to Actions with the difference that Events are not chosen by the Character Agents.
3. It sends requests for world updates. These world updates are updates in the *setting*, in other words, updates that are used in the story as if they have always been there in the first place. In principle they will always be honored; it is up to the Plot Agent to make sure that these updates are not inconsistent for the story, as we have discussed in Section 4.2.1.

The `control` arrow in Figure 7.1 consists of three types of information:

1. The Character Agents send their Internal Elements, Goals, and planned Actions to the Plot Agent;
2. The Plot Agent suggests Goals or Actions;
3. The Character Agents can request world changes.

Firstly, the Character Agents expose their internal states by sending their Internal Elements, Goals, and planned Actions to the Plot Agent. This information is used to build a fabula structure on which to base plot control decisions. Secondly, the Plot Agent can suggest Goals or Actions to the Character Agents. In Section 5.5 I have proposed this as one of the two ways for the Plot Agent to control the characters; the other is influencing the character's perceptions, this has been covered by the fact that the Plot Agent determines all perceptions of the Character Agents. Finally, in response to the suggestions or by own initiative, the Character Agents can request world changes that spring from improvisation.

7.1.2 Time step cycle

Time in the story world is measured in time steps. Each time step, a cycle of Agent communication takes place in which the World Agent, Plot Agent and Character Agents collaborate to execute Actions and Events.

From the World Agent perspective, a turn entails a couple of tasks. The World Agent will send World Updates to the Plot Agent, indicates which Actions are finished, and gives information about "what time it is" in time steps. This is reason for the Plot Agent to respond with a list of Actions and Events which should be executed.

The Plot Agent forms the bottleneck that determines how long the cycle lasts. Upon receiving World Updates, it will send perceptions to the Character Agents and when the Plot Agent sends actions to the World Agent, this ends the cycle. The Plot Agent will collect the Actions from the different Character Agents. The Character Agents also communicate what their internal state is (beliefs, emotions, goals, plans, etc.)

Character Agents can request world changes, which the Plot Agent can honor or not. This happens in parallel to the turn; it can happen at any time and a response from the Plot Agent can also happen at any time. Neither the Plot Agent nor the Character Agents will wait on this. If the information arrives too late, then it is used in the next turn. Similarly, the Plot Agent can send suggestions of Goals and Actions to the Character Agent which also happens in parallel to the turn.

Sequence diagrams for this communication can be deployed. For a detailed description, see Appendix A. From this point on I will assume this communication to be there, and will focus on the architectures of the World Agent and the Plot Agent.

7.2 World Agent architecture

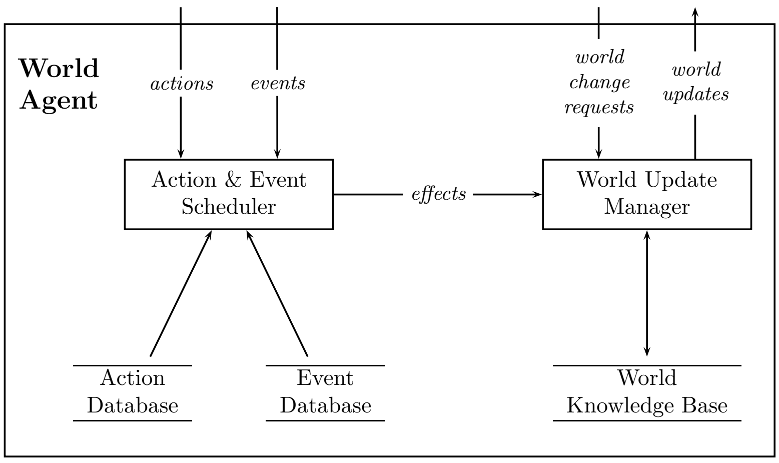

Figure 7.2: *World Agent design*

The World Agent (see Figure 7.2) has the responsibility to manage the Story World as it really is. Its design is quite straightforward; it maintains the true knowledge about the state of the world, and handles actions and events that occur. The Plot Agent will be influencing the world in the form of Actions, Events, and world change requests. World change requests are in principle always honored, as discussed before in Section 7.1.1. The Plot Agent is responsible for not introducing world changes that are inconsistent with the story.

Using an Action and Event database, the World Agent can check the validity of actions and events (i.e. do preconditions hold?) and process their effects on the world when they are finished.

The World Update Manager will explicitly keep track of the changes in the world. The reason for this is that the Plot Agent needs to know what changed, since world changes might lead to perceptions for the Character Agents. The Plot Agent also needs to know what *causes* the changes, as this will be used to make a causal connection between Perceptions and their cause. The cause can be an Action or an Event. Honored world change requests have no cause. The reason for this is again that honored world changes are part of the setting and the resulting knowledge is treated as if it had been there from the beginning of the story.

7.3 Plot Agent architecture

This section will discuss the global architecture of the Plot Agent. The Plot Agent has the responsibility of affecting the emerging event sequence in such a way, that the fabula becomes more interesting from a plot point of view. It will also build a fabula structure using the intentions of the Character Agents (their Internal Responses, Goals, attempted Actions and the causality of each) and the Events.

7.3.1 Decision flow

To come to a design of how the Plot Agent should direct the plot, it is important to first identify the decisions the Plot Agent has to make to influence the plot, and see what to base these decisions on. We have argued in Section 5.5 to use four ways to influence the story, that lead to decisions the Plot Agent has to make:

- Generating events to mediate the plans of characters: which Events should happen and when?
- Influencing the perceptions of the characters: What do characters perceive? And how and when should this deviate from a standard heuristic?
- Changing the setting: which world changes requested by characters should be allowed, and which should not? Which world changes should be actively chosen by the Plot Agent?
- Directing the characters by suggesting Goals or Actions: which character Goals or Actions are preferable?

An answer to these questions should spring from a certain plan that the Plot Agent has for the story. Such a plan can for instance consist of generating conflicts to heighten drama, making connections in the fabula to heighten plot coherence, or thwarting or helping character goals. Just like Character Agents have Goals, the Plot Agent has goals as well, which we will refer to as plot goals, as discussed in Section 1.3.2. For instance in the Princess Lovely story (Figure 3.1), there could be a plot goal to help the children satisfy their hunger using the runaway lamb, so that it will have a certain effect on the Storyline Goal of Princess Lovely wanting her lamb back. These plot goals can be based on the ingredients of a good story that have been explored in Section 2.2.3. Coming to a design of plot goals and a planner for these goals is beyond the scope of this thesis, and is also not an easy task, although it makes sense that the plot goals should somehow be based on the emerging fabula structure. Section 8.2.1 provides a suggestion for such a design.

One of these plot goals can be thwarting or helping a character Goal. Let's take this goal as an example to elaborate on. The main basis on which every other decision is eventually made, is then ultimately the decision whether a certain character Goal should be thwarted, or helped. From a top-down point of view, we can then visualize the decision flow as in Figure 7.3.

First of all, whether Actions should be thwarted or helped is something that flows directly from the Goals motivating these Actions from the character point of view. If this Goal should be thwarted, all Actions motivated by this Goal, according to the intentions of the character, are candidates to be thwarted. If it should be helped, all Actions motivated by it can be helped. Then, exceptions in Perceptions can be made based on the decision whether to help or thwart an

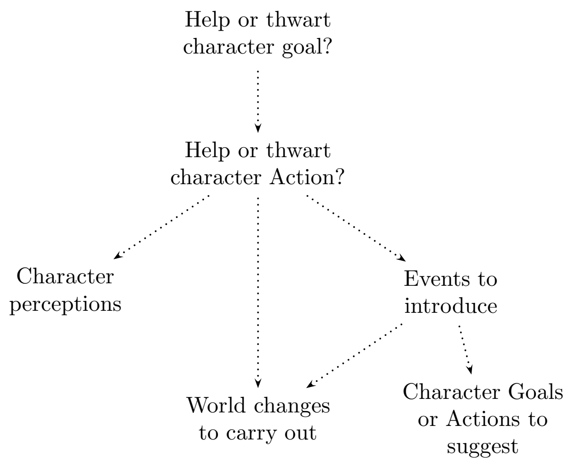
Help or thwart character goal?
Help or thwart character Action?
Character perceptions
Events to introduce
World changes to carry out
Character Goals or Actions to suggest

Figure 7.3: *Plot Agent decisions*

Action. What characters perceive is something that is normally decided by a standard heuristic. Exceptions can be made in certain cases where seeing something that would not be seen according to the heuristic, or not seeing something that would normally be seen, would help or thwart the Action. Third, when Characters request world changes in order to plan Actions for their Goals, it's obvious that decisions in this respect can be made according to the Plot Agent's ideas about helping or thwarting these characters in their Actions. Fourth, the Plot Agent would be introducing Events, again motivated from the idea to help or thwart the Actions that characters plan to execute. Finally, the Plot Agent can suggest Goals or Actions to characters. The motivation behind this is to create situations where certain Events can be applied, or to help or thwart the characters achieving their goals.

7.3.2 Modules

The design of the Plot Agent follows from the decisions it has to make. Every decision that the Plot Agent makes is ultimately based on the plot goals. The core of the Plot Agent architecture will be a manager that plans these plot goals. Furthermore, each of the four ways of influence (events, perceptions, setting changes and suggestion) will get its own module. Rather separated from this is the process of building a story by generating the fabula knowledge. This is not a totally separated process though since the plot goals will have to be based on this fabula knowledge.

See Figure 7.4 for the architecture of the Plot Agent. Dotted arrows indicate decision dependencies. The design contains several knowledge bases that are used by the different modules. The World Knowledge Base contains the Plot Agent's version of the state of the world. The Plot Knowledge Base contains information about the emerging fabula. The Event Database contains Events that the Plot Agent can introduce, and the Case Database is used by the Creative Problem Solver module. The Creative Problem Solver is included as a central module that can be used by all other modules. On top of this, the following modules can be identified:

Plot Goal Manager The module that decides what the current plot goals are.

Event Planner The module that plans Events in service of the plot goals. It will use a Case Base to reason about what events to generate to fulfill the plot goals. The Event planner will need to plan Events and try to make the preconditions of planned Events valid.

Perception Manager The module that decides which character gets which perceptions of the world. A simple heuristic to determine what a certain character c can perceive is:

- every object on c's current location, except when it is inside a (non-transparent) `Container`.
- every Action of characters that are on the same location as c, as soon as it starts, is interruptible, or ends.
- every Event that happens on c's location.

The perception manager should keep track of the cause of perceptions, for fabula generation.

Setting Manager The module that decides what changes to the state of the world should be made in order to validate the preconditions of planned Events or to directly thwart or help character Actions. It will handle requests for world changes from characters, and can decide whether or not to permit these changes.

Director The module that is responsible for influencing the characters to do what the Plot Agent wants, in the form of Goal and Action suggestions.

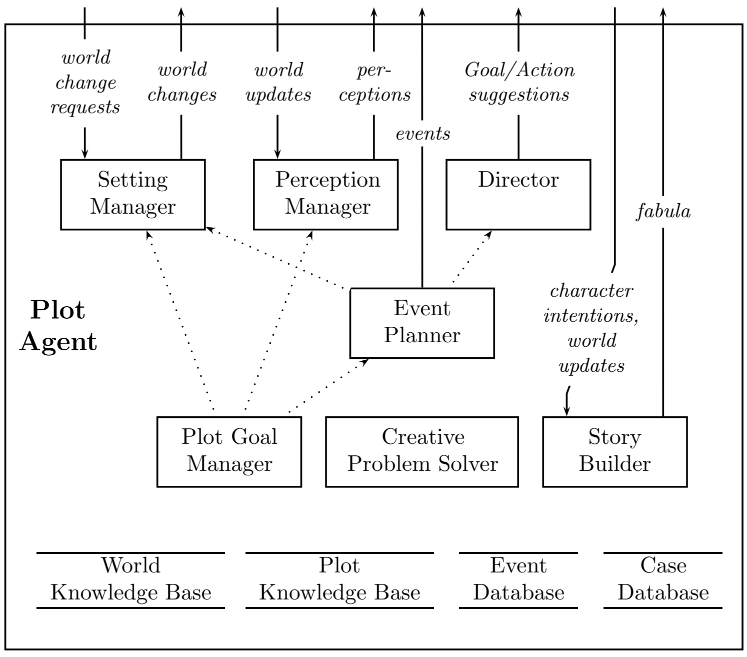

Figure 7.4: *Plot Agent design*

The decision tree of Figure 7.3 indicates that this is based on the question which Events will be introduced, so Goal and Action suggestions are meant to fulfill the preconditions of these Events.

StoryBuilder The module that gathers everything that happens in the course of the story, and forms a knowledge representation of the fabula. This gathering follows directly from the situations described in Section 4.3.1. The story consist of every story element that is affiliated with the Storyline Goal; we have to define what this means. Only the affiliated story elements are communicated to the Narrator. A very simple heuristic would be to say that all story elements that are directly or indirectly causally connected to the Storyline Goal, are affiliated to it.

7.3.3 Implementation

The World Agent and the StoryBuilder component of the Plot Agent have been implemented. The World Agent has been implemented using Java and JADE (Java Agent Development Framework [JADE], n.d.), following earlier implementations. For keeping track of the world state, the World Agent makes use of a knowledge manager which has been built on top of the Java Theorem Prover (JTP, n.d.). This knowledge manager can maintain a knowledge base of asserted and derived OWL facts that constitute a state of the world. Appendix B describes JTP in more detail. Appendix C discusses implementation issues that have been tackled.

To gain insight in the knowledge of the system, a visualization component has been implemented using Parlegraph, a Java graph representation and visualization package. This component can be used in all Agents to gain insight in the knowledge base by drawing objects and their relationships in the form of labeled nodes and edges representing the RDF triples in the knowledge base. See also Section C.3 for a description of Parlegraph.

7.4 Conclusion

A new architecture of the Virtual Storyteller has been defined that is very flexible. It uses a World Agent to keep track of the world state and its changes, and a Plot Agent to create a story using Character Agents. The World Agent and parts of the Plot Agent have been implemented.

Further design is needed to come to a more detailed specification and implementation of the other modules of the Plot Agent. An important part that needs to be designed is be the Plot Goal Manager, which will determine the course of the story by formulating plot goals. The Plot Goal Manager forms the missing link between the emerging fabula developed in Chapter 4 and the decision making abilities that have been identified in Chapter 6.

Chapter 8

Conclusions and recommendations

The Oracle: (making cookies) "That's it. That's the secret. You've got to use your hands."
Sati: "Why?"
The Oracle: "Cookies need love like everything does."

— 'The Matrix Reloaded' (2003)

This chapter will conclude the work described in this thesis which is only a part of the work done on the Virtual Storyteller as a whole. I will identify what is still left to be done for plot control and give some recommendations as to how to proceed from this point on.

8.1 Conclusions

This thesis has researched three areas of emergent story generation. Three research questions have been discussed:

RQ1 How can a plot structure be designed that can be generated, narrated and reasoned about?

RQ2 Which techniques can be used to control the plot whilst keeping the characters believable?

RQ3 How can the Plot Agent use the plot control techniques to reach plot goals?

Firstly, a formal fabula structure has been developed that allows us to come to a better understanding of the emerging story by expressing Goals, Actions, Events, Outcomes etc. and their causal relationships. This fabula structure forms the basis for a concept of plot, revolving around a Storyline Goal and encompassing everything that is affiliated to this goal. An ontology of this fabula structure has been implemented in OWL.

Secondly, techniques to control the plot have been investigated. Besides the introduction of Events and changing the setting of the story, these techniques involve influencing the Character Agents whilst leaving them as much autonomy as possible. Identifying these techniques has lead to a new architecture of the Virtual Storyteller in which the Plot Agent has a more defined role.

Finally, this thesis has explored the way in which plot control decisions can be made. A creative problem solver using creative Case Based Reasoning has been designed which can be used to aid these decisions. Such a problem solver is useful for the Character Agents and the Narrator as well. I have shown how problems and cases can be stated using the fabula ontology.

These three aspects – the fabula structure, plot control techniques and the creative problem solver – have been combined into a new architecture for the Virtual Storyteller that uses a Plot Agent and World Agent. The World Agent and Story Builder component of the Plot Agent have been implemented using OWL for their knowledge representation.

The designs and implementations described in this thesis have not been tested for generating stories. When believable characters are developed to a workable degree, and the Narrator implementation discussed in (Slabbers, 2006) is integrated with the system, tests can be run to generate stories.

8.2 Recommendations for future work

Several areas of future work can be identified. These include further design and implementation of the Plot Agent and its components, extending the structures of fabula and plot that the Plot Agent has, and improving knowledge management of the system.

8.2.1 Further design and implementation of the Plot Agent

Implementation creative problem solver and Event ontology

An important next step is to make an implementation of the creative problem solver that has been designed in Chapter 6. This implementation would form a proof of concept of the design. The implementation relies on two ingredients that have to be created still:

- An ontology of Events;
- A case base.

An ontology of Events can be designed similar to the ontology of Actions described in (Uijlings, 2006), as has been discussed in Section 3.2. With these Events defined, cases can be expressed and a case base can be developed and tested.

Design of the Plot Goal Manager and other Plot Agent modules

Although Chapter 7 described a global architecture of the Plot Agent, its modules have not been fully specified. The Story Builder module has been implemented but the other modules need further design and implementation. Especially the Plot Goal manager will be important. A suggestion for this is to divide a story in three parts:

1. A beginning, where a Storyline Goal emerges;
2. A middle part, where the Storyline Goal is helped and thwarted;
3. An ending, where the Storyline Goal is resolved with either a good or a bad outcome.

One can start in a predefined environment with a predefined Storyline Goal, and focus on generating the middle and ending. Usually in a story the drama increases first before it decreases again. To increase drama, the Plot Agent could form plot goals that focus on making it difficult for the characters, and to decrease it, the Plot Agent can use plot goals to help the characters. To guide decisions about which plot goals to adopt, dramatic arcs can be used, like Freytag's triangle, or the climax structures of Appelcline. Appelcline's climax structures are used to work out stories in role playing games. See (Rensen, 2004) for a description of these structures.

8.2.2 Extension of story understanding

Elaborate fabula

An unsolved area of capturing the fabula is capturing the world state at each point in time. The fabula structure that has been designed only captures those elements that are either the cause or the effect of other elements. Elaborate world states are necessary mainly for presentation and narration of the fabula (Slabbers, 2006). A possible solution that will need looking into is the use of Situation Calculus to keep track of what the state of the world was at any moment in time. Another solution is to "log" the world state by writing it to a file every time step.

The Internal Elements part of the fabula structure has not been worked out to a great degree. Schärfe (2002) defines possible worlds that could extend the fabula structure to split goals and Internal Elements into W-worlds (expressing wishes), O-worlds (expressing obligations), etc. for greater expressiveness. This is worth looking into. However, it doesn't make sense at this time to have the ability to capture more Internal Elements than the Character Agents can actually produce, so the development of an Internal Element ontology will have to go hand in hand with the development of the Character Agents.

Plot understanding

For better plot understanding, taking only the Storyline Goal as a focus might not be enough. For future work on understanding the plot, it might be worthwhile to look at some higher plot structures like recognizing conflicts, tensions and deceptions. Even so, the plot will most likely still remain somewhat weak since that is inherent to having a character-centric approach.

Furthermore, the Plot Agent needs to understand affiliation. The story consist of every story element that is affiliated with the Storyline Goal; we have to define what this means. A simple heuristic would be to say that all story elements that are directly or indirectly (up to a certain distance) causally connected to the Storyline Goal, are affiliated to it. But this heuristic is probably too simple.

For each narrative, the author implicitly uses a reader model to manage the reader's reaction to the story (Szilas, 1999). If such a reader model can be constructed, plot understanding is heightened because there is a better concept of interestingness.

Time leaps for more episodic control

The question is whether generating every detail of the fabula of a story to the full extent doesn't make emergent narrative too inflexible. It makes sense that maybe fabula generation can exist on an episodic level with leaps in time where fabula information can be absent. Just like in theater play the scenery and characters can be taken forward in time where fabula generation continues. This would form a good possibility to introduce episodic goals. For instance in the Princess Lovely story of Figure 3.1 it makes sense to start a new episode when Lovely climbs into the highest tower. The leap in time is depicted by the sentence "one day she climbs up to the highest tower...". Of course a new episode can start with any of the fabula elements, according to Trabasso et al. (1989). For instance the episode initiated by the sentence "one morning when princess Lovely wakes up, she cannot find the little lamb" is likely started by an event of the lamb running away. The days in between don't need elaborate fabula.

8.2.3 Knowledge management

Different solutions to knowledge representation and management

The reasoning capabilities in the Virtual Storyteller are provided by the Java Theorem Prover (JTP, n.d.) developed at Stanford University. Our experiences with JTP have led to the conclusion that it might not be the best system to use. Apart from the fact that it is poorly documented and no longer supported, it is a theorem prover which provides more extensive reasoning than we will use for story generation, which means an unnecessary performance drawback. A possible alternative that has proved its usefulness in many other OWL applications is the OWL reasoner RACER (Racer, n.d.).

Furthermore, the solution to contextualizing knowledge given in Section 4.4.2 is not ideal. Far more ideal would be to use Named Graphs, discussed in Section 4.4.1. JTP is built on top of Jena, a Java framework for the Semantic Web. An extension to Jena providing Named Graphs is available, which is called NG4J (NG4J, n.d.). Another possible alternative is to leave out JTP and make direct use of Jena and NG4J. Query languages exist and are being developed for graph reasoning, i.e. TriQL and SPARQL (Carroll et al., 2005).

Authoring and managing knowledge

An important issue in the Virtual Storyteller is the authoring of knowledge. Our experiences are that even with a very simple story and a very limited amount of facts in the knowledge base, it is already difficult to gain insight in exactly what is happening on a knowledge level. We need to be able to easily author this knowledge, but also to easily gain insight in the state of the world at any time.

Two areas spring to mind where authoring knowledge is concerned. The first one is managing the case base. Adding, reading and removing cases should be made easy; it is very important to come to a good way of authoring these cases, maybe this can be done in a graphical way. The other one is the ability to create story world settings for the stories to start in, which could even be enriched with back stories for the characters, stating how they came to their initial state and goals. It makes sense that authoring knowledge in the form of adding facts to a text file might have its shortcomings in that respect.

Appendices

Appendix A

Agent communication

This appendix describes sequence diagrams for the communication between the Agents in the new Virtual Storyteller architecture (see Chapter 7).

For this, an ontology of communication message concepts has been defined in Protégé-2000 version 1.7, similar to Rensen (2004). The meaning of these concepts follows from the sequence diagram descriptions. The concepts can be converted in to Java Beans using the Java Bean Generator. These Beans can contain information and can be sent to other Agents using the Java Agent platform JADE. See Figure A.1.

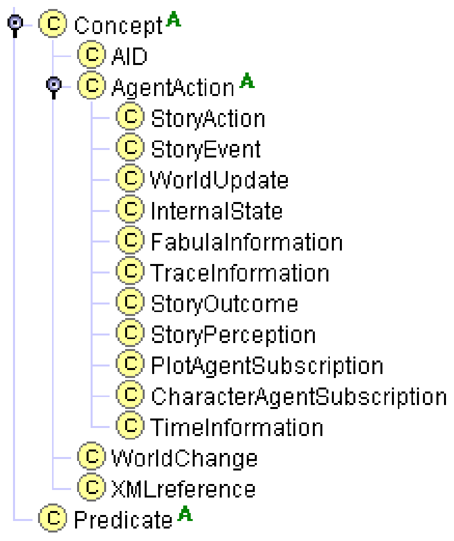

Figure A.1: *Communication ontology*

A.1 Subscription

The Plot Agent makes itself known to the World Agent by searching for a World Agent within the Agent platform, and subsequently sending a SUBSCRIBE message to it. See Figure A.2.

Character Agents subscribe to a Plot Agent in a similar way. The Character Agents will also communicate which character in the story they play. See Figure A.3, which shows a Character Agent subscribing itself for the character that is identified in the story by the Individual `sws:Amalia`.

Characters can subscribe before the story starts, but can also subscribe at a later stage; if a Character Agent enters the story at a later moment in time, when the story generation process has already started, it will be getting perceptions and given the possibility to perform Actions the next turn.

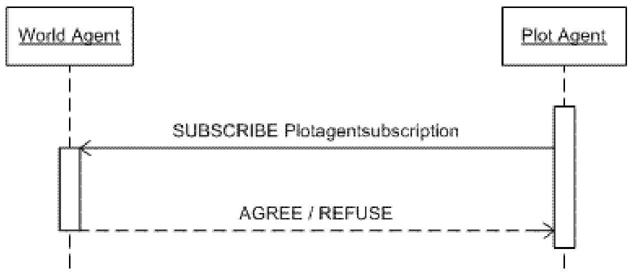

Figure A.2: *Plot Agent subscription*

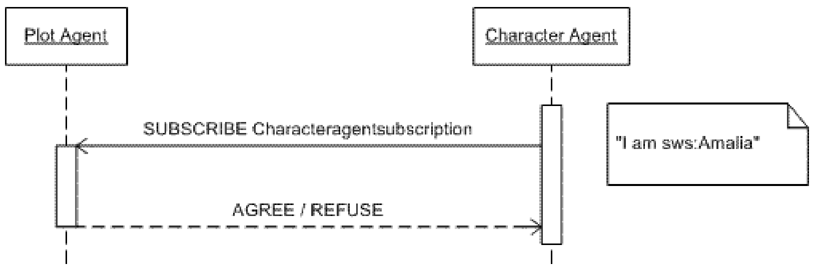

Figure A.3: *Character Agent subscription*

A.2 Turn

The World Agent will send World Updates to the Plot Agent, indicates which Actions are finished, and gives information about “what time it is” in time steps. The Plot Agent responds with a list of StoryAction and StoryEvent Objects that indicate which Actions and Events should be executed.

The World Agent takes the responsibility for the synchronization of the different Agents. It keeps track of the Story World’s time using time steps. When all Character Agents have proposed an action to execute, and the Plot Agent has proposed an Event (or a timeout has occurred) and the World Agent has processed all these world changes, the time step is increased and all Agents are notified of this by a pulse.

From the Plot Agent perspective the Actions are collected from the different Character Agents, which also communicate what their internal state is (beliefs, emotions, goals, plans, etc.) See Figure A.4.

A.3 Control

Character Agents can request World Updates, which the Plot Agent can honor or not. This happens in parallel to the turn; it can happen at any time and a response from the Plot Agent can also happen at any time. If the information arrives too late, then it is used in the next turn. See Figure A.5. Similarly, the Plot Agent can send suggestions of Goals and Actions to the Character Agent. See Figure A.6.

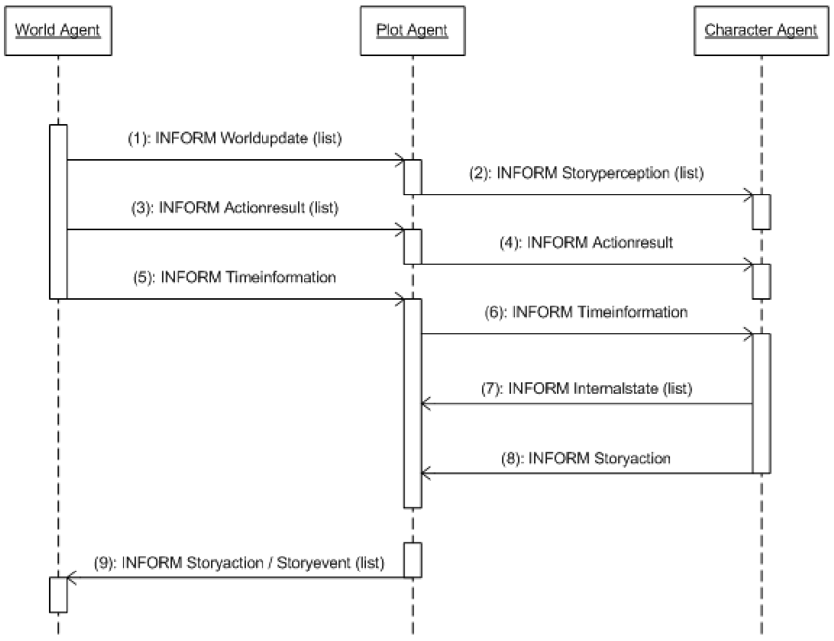
World Agent
Plot Agent
Character Agent
(1): INFORM Worldupdate (list)
(2): INFORM Storyperception (list)
(3): INFORM Actionresult (list)
(4): INFORM Actionresult
(5): INFORM Timeinformation
(6): INFORM Timeinformation
(7): INFORM Internalstate (list)
(8): INFORM Storyaction
(9): INFORM Storyaction / Storyevent (list)

Figure A.4: *A turn*

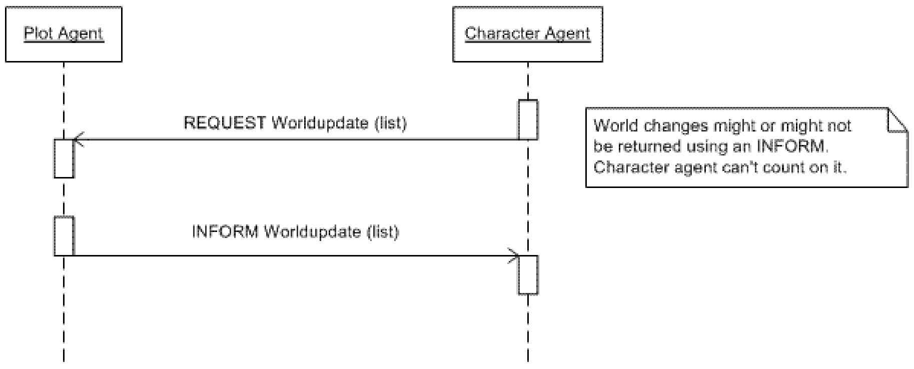
Plot Agent
Character Agent
REQUEST Worldupdate (list)
INFORM Worldupdate (list)
World changes might or might not be returned using an INFORM. Character agent can't count on it.

Figure A.5: *World Update request*

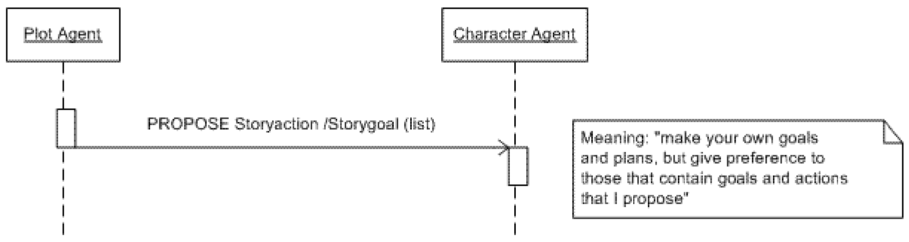
Plot Agent
Character Agent
PROPOSE Storyaction /Storygoal (list)
Meaning: "make your own goals and plans, but give preference to those that contain goals and actions that I propose"

Figure A.6: *Suggestion*

Appendix B

Java Theorem Prover (JTP)

`JTP` is an object oriented reasoning system, implemented in Java. This chapter will describe the implementation of a simple reasoning system in `JTP`. The parts of the system will be discussed and several problems that can arise will be identified. A brief JavaDoc description of the `JTP` classes can be found on the `JTP` website: http://www.ksl.stanford.edu/software/JTP/api/

The basis for a reasoning system is the Reasoning Context. Every Reasoning Context in `JTP` extends the base class `jtp.context.BasicReasoningContext`.

B.1 Representation of knowledge

Knowledge is represented in the Conjunctive Normal Form (CNF). A CNF sentence consists of a collection of clauses that are all true and conjunct (`clause` $\wedge$ `clause` $\wedge$ `...`). A clause is represented as a collection of disjunct literals (`literal` $\vee$ `literal` $\vee$ `...`). A literal can be a positive literal (p) or a negative literal ($\neg p$).

When knowledge is added to the system, it is converted to CNF. A sentence like $A \rightarrow B$ would be converted to $\neg B \vee A$, with A and B being a literal.

A literal consists of a relation and arguments between which the relation holds. In a literal `(ancestor Mary Sue)` (KIF), `ancestor` would be the relation, and `Mary` and `Sue` its arguments.

Arguments of a literal can also be variables. A variable starts with a question mark ('?') followed by the variable name. This way, for instance transitivity can be expressed in KIF as follows:

```
(=>
  ( and (ancestor ?x ?y) (ancestor ?y ?z) )
  (ancestor ?x ?z)
)
```

B.1.1 Java representation

Knowledge is represented using a couple of Java classes:

`jtp.fol.CNFSentence` consists of a `Collection` of (conjunct) `Clauses`, so the `CNFSentence` containing `Clauses` `A`, `B` and `C` would have the KIF representation `(and (A) (B) (C))`.

`jtp.fol.Clause` consists of a `Collection` of (disjunct) `Literals`, so the `Clause` containing `Literalss` `A`, `B` and `C` would have the KIF representation `(or (A) (B) (C))`.

`jtp.fol.Literal` is a simple Clause that consists of a relation `Symbol` and arguments which is a `List` whose elements are either a `Symbol`, a `Variable`, or other objects like an `Integer`, `Float` or `Boolean`. It also has a polarity: when the polarity of a literal p is $false$, then $\neg p$ is $true$.

`jtp.fol.Symbol` is a (symbol) element, having a package (namespace) that can be empty and a name. `owl:inverseOf` can be represented as a `Symbol` with package `owl` (or in fact `http://www.w3.org/.../owl#`) and name `inverseOf`.

`jtp.fol.Variable` is a variable, for instance `?loc`.

B.2 Reasoning Context

In `JTP`, several Reasoning Contexts are implemented. Each of these contexts offers the possibility to store (tell) knowledge and retrieve (ask) it. The knowledge is inputted using sentences in KIF, and are converted by the context to CNF sentences. Internally, these CNF facts are used to derive other facts.

A Reasoning Context consists of Reasoners and Dispatchers. Dispatchers receive a query or assertion and pass them on to suitable Reasoners. Telling reasoners use knowledge in a forward chaining manner; they process consequences of asserted knowledge. Asking reasoners process knowledge using backward chaining: with every query the Reasoner looks for proofs.

The simplest Reasoning Context is implemented by the class `jtp.context.-BasicReasoningContext`. This context offers the possibility for processing assertions and queries:

- Assertions are delegated by a `jtp.modelim.BreadthFirstForwardReasoner` Reasoner to a `jtp.disp.SequentialDispatcher`;
- Queries are delegated by a `jtp.context.IterativeDeepening` Reasoner to a `jtp.disp.SequentialDispatcher`.

To store knowledge, a `jtp.gmp.ClauseOrientationKB` has to be added to a `BasicReasoningContext`. The actual storage of knowledge is done in this class using the private variables `pos` and `neg`.

B.2.1 Specialized contexts

Within `JTP` there are 2 extensions on the basic context as implemented by `BasicReasoningContext`: the classes `jtp.frame.fc.Context` and `jtp.frame.-listen.Context`. Both class offer support for forward chaining rules using special reasoners, but the way in which rules are supported is different for the two classes.

A survey of the reasoners that are used by `jtp.frame.listen.Context`:

- Telling reasoners:
  - `jtp.frame.LinkAsserter`
  - `jtp.frame.listen.VCListenerCreator`
  - `jtp.frame.SlotValueTellingReasoner` (coupled to a `FrameKB`)
- Asking reasoners:
  - `jtp.frame.SlotValeuAskingReasoner` (coupled to a `FrameKB`)

The class `jtp.frame.listen.Context` forms the basis for the class `jtp.-context.rdf.RDFContext`. An `RDFContext` adds the following Asking reasoners to the inherited reasoners:

- `jtp.frame.HoldsReasoner`
- `jtp.frame.EnumeratingReasoner`
- `jtp.func.Inequal`

The class `jtp.context.owl.OWLContext` is a further specialisation of the `RDFContext`. An `OWLContext` adds the following Telling reasoners:

- `jtp.frame.listen.dl.IntersectionTypeReasoner`
- `jtp.classifier.ClassifierTellingReasoner`

RDF/OWL functionality as described in http://www.ksl.stanford.edu/software/-JTP/doc/owl-reasoning.html is done by loading the following knowledge files when setting up an RDF or OWL reasoning context:

```
jtp/context/owl/owl-assumptions.kif
jtp/context/owl/owl-rules.xml
jtp/context/rdf/rdf-assumptions.kif
jtp/context/rdf/rdf-rules.xml
```

In the XML files, forward chaining rules are defined, whereas in the KIF files, definitions are given of OWL and RDF facts like:

```
(rdfs:domain rdf:type rdfs:Resource)
(rdfs:range rdf:type rdfs:Class)
(rdf:type owl:Thing owl:Class)
(owl:complementOf owl:Thing owl:Nothing)
```

B.3 Specialized reasoners

Depending on the use of the context, specialized reasoners might have to be added. `JTP` contains a number of specialized reasoners that are not documented, but will turn out to be very useful, as we will see.

To use a custom Reasoner in a Reasoning Context it should be added to the Dispatcher of this context. For example, maybe we want to add a `Greater` Reasoner, which is an Asking Reasoner that can reason about the truth of assertions like `(> 4 2)` (is 4 greater than 2?). This can be done as follows:

```
DispatcherUtils.addToDispatcher( new jtp.func.Greater(),
    myContext.getAskingDispatcher());
```

B.3.1 Unprovable

A special Reasoner that takes some attention is the `Unprovable` Reasoner. Its purpose is to answer the question whether a literal can *not* be proven. It can be used to ask whether something like "there is no known son of Bob", in other words, it should not be possible to find any `?x` such that `(fatherOf Bob ?x)` is true. When the `Unprovable` Reasoner is added to the Reasoning Context, this can be asked as follows:

```
(unp (fatherOf Bob ?x) )
```

The argument of `unp` must be a literal.

The `Unprovable` Reasoner needs another Reasoner to handle the argument query, then results a dummy `Iterator` (which binds the variables to the value `null`) if no proof can be found for the argument. This Reasoner should be set first using the method `setAskingReasoner` and the code for adding an Unprovable Reasoner to the Reasoning Context becomes:

```
// Add unprovable Reasoner
  Unprovable unpReasoner = new Unprovable();
  unpReasoner.setAskingReasoner(myContext.getAskingReasoner
      ());
  DispatcherUtils.addToDispatcher(r,myContext.
      getAskingDispatcher());
```

B.3.2 GetSetof

Another very useful Reasoner, when looked closely at it, is the `GetSetof` Reasoner. It can be used to reason about sets. Just like with the `Unprovable` Reasoner, another Reasoner should be set using the `setAskingReasoner` method. A query that is handled by the `GetSetof` Reasoner looks like:

```
(get-setof (fatherOf Bob ?x) ?x (setof Mark Stuart))
```

This query succeeds when Bob is the father of Mark and Stuart, and of nobody else. What happens internally is that the first argument, `(fatherOf Bob ?x)`, is processed as a query, after which every binding x_i of the second argument (`?x`) is used to construct a literal of the form:

(`setof` x_1 x_2 `...` x_n)

which is then unified with the third argument. If this unification succeeds, the query succeeds. If the third argument is a variable, it is bound to the value of the constructed literal.

This can be used for instance to construct literals like:

- Bob has exactly two sons.
  `(get-setof (fatherOf Bob ?x) ?x (setof ?one ?two) )`
- Bob has no sons. (using an empty `(setof ...)` literal)
  `(get-setof (fatherOf Bob ?x) ?x (setof ) )`
- Who are Bob's sons? `?sons` gets a binding like `(setof Mark Stuart)`
  `(get-setof (fatherOf Bob ?x) ?x ?sons)`

B.3.3 ForIn

The `ForIn` Reasoner completes the functionality of asking complex queries. It can be used to see if a query holds for every binding of a variable from a set. It can be used to ask: *are Mark and Stuart married?*

```
(for-in ?x (setof Mark Stuart) (married ?x ?y))
```

The first argument is the variable to loop through, the second argument is a set of the bindings that the variable should get, and the third argument is the (Literal) query to process for each binding of the variable. Internally, the query above is translated into a conjunction ($\wedge$) of the following queries:

```
(married Mark ?y)
(married Stuart ?y)
```

This Reasoner, combined with the `GetSetof` Reasoner, can be used to create powerful queries like: *are all of Bob's sons married?*

```
(and
    (get-setof (fatherOf Bob ?x) ?x ?sons)
    (for-in ?x ?sons (married ?x ?y))
)
```

Or even more complex, using the `Unprovable` Reasoner: *none of Bob's sons are married.*

```
(and
    (get-setof (fatherOf Bob ?x) ?x ?sons)
    (for-in ?x ?sons (unp (married ?x ?y)))
)
```

B.3.4 Math reasoners

There are several mathematical reasoners, like `Plus` and `Multiply`. These can be used to reason with values:

- 3 plus 4 equals 7.
  `(+ 3 4 7)`
- What is 3 times 5?
  `(* 3 5 ?answer)`
- Is Bob's age greater than Marks age and Stuarts age combined?

  ```
  (and
      (age Bob ?b)
      (age Mark ?m)
      (age Stuart ?s)
      (+ ?m ?s ?sum)
      (> ?sum ?b)
  )
  ```

B.4 Adding and removing knowledge

Adding knowledge to a Reasoning Context is done by asserting facts. Asserted facts can activate forward chaining rules, that is why there are two ways to assert knowledge: one returns a `ReasoningStepIterator`, the other returns an integer. The `ReasoningStepIterator` can be used to iterate through the forward chained information, the integer simply indicates the number of derived facts.

Furthermore, the fact to be asserted can be entered as a `String` or as an `Object`. This leads to the following 4 methods in the class `BasicReasoningContext` with which a fact can be asserted:

```
int tell ( Object assertion );

int tellString ( String assertion , URL url );

ReasoningStepIterator getAssertionResults (
    DirectAssertion assertion );

ReasoningStepIterator getStringAssertionResults (
    String assertion );
```

The difference between passing an `Object` or a `String` is that a `String` will first have to be converted to an object. The difference between the methods that return a `ReasoningStepIterator` and the other methods, is that with these methods, the assertion and its consequences are only asserted to the system as the returned `ReasoningStepIterator` is walked through (using `next()`).

To delete knowledge from the Knowledge Base, it should be untold. Only facts that were directly told (and not derived from other facts) can be untold. This also applies that only literals can be used to untell. When a fact is untold, all its consequents no longer hold either. Knowledge can be untold using the following `jtp.context.rdf.RDFReasoningContext` method:

```
void untellString(String literal);
```

B.5 Retrieving knowledge

Retrieving knowledge is done by posing a query to the Knowledge Base. The context `BasicReasoningContext` offers the method `ask (String query)` for this. Internally, the query is translated by the `FirstOrderLogicTranslator` of the context, that in turn uses the parser that is set for the context at that time (see `setParser()`). JTP sometimes changes the used parser (for instance when a Knowledge Base is loaded using `loadRDFKB()`, setting an RDF parser), that's why it's recommended to invoke the `AskingReasoner` directly:

```
getAskingReasoner().process(translator.translate(
    query ))
```

Queries that consist of a simple literal (for instance `(fatherOf ?x ?y)` are processed by the model elimination theorem prover, implemented in the class `jtp.modelim.ModelEliminationReasoner`. When a query has a more complex nature, it is given to the class `jtp.modelim.AskingQueryReasoner`. This Reasoner handles the query in a special way:

- A temporal `Literal` is constructed with relation $aux, and its arguments being the collection of unbound variables from the original query;
- An aid rule is added to the Knowledge Base. This rule is: $query \rightarrow \$aux$.
- Then a new query: `($aux)` will be posed. This way a complex query can be handled like a simple query again, handled by the model elimination theorem prover.

Let's look at an example. Say, the query is `(a ?x)` → `(b ?y)`. The free (unbound) variables in this query are `?x` and `?y`. An aid literal is constructed: `($aux ?x ?y)`. The following rule is added to the KB: `( ((a ?x)` → `(b ?y))` → `($aux ?x ?y) )`. Internally, the query `($aux ?x ?y)` is passed to the `ModelEliminationReasoner`.

A result of using the `ModelEliminationReasoner` is that more and more rules will be added to the KB with every query, since the `$aux` aid rules are never removed. It is therefore advised to create a snapshot of the knowledge using the `UndoManager` (see Section B.5.2) and restore the knowledge after a query.

If you want to use a variable for a relation, for instance to ask which relations hold between two objects, this can be asked using the special construct `holds`:

```
(holds ?rel Bob Stuart)
```

This might for instance result in the variable `?rel` bound to the relation `fatherOf`.

B.5.1 Result of a query

The result of a query is a `ReasoningStepIterator`, which is an important class within `JTP`. An instance of the class `jtp.ReasoningStep` is in fact a proof of a query. A `ReasoningStep` consists of the following elements:

goal is the sentence to be proven;

subgoals exist when the goal is not a literal but a conjunction of literals, the sub goals are these literals;

subproofs are proofs of every sub goal;

bindings are the free variables that occur in the sentence and the values that they are substituted for while creating a proof.

B.5.2 The Undo manager

The knowledge in a `JTP` Reasoning Context is stored in attributes of the class `ClauseOrientationKB` and is not directly available. Instead, knowledge can be iterated using the class `UndoManager`. Knowledge that is asserted using the `tell` methods is managed by the undo manager, so that facts can later be retracted using `untell`. This functionality is encapsulated in the class `jtp.-undo.SnapshotUndoManager` which builds on the Java SDK classes in the package `javax.swing.undo`. Using this class to retrieve knowledge is done as follows:

1. Using `BasicReasoningContext.getUndoManager()` the undo manager is retrieved;
2. A snapshot is made using `SnapshotUndoManager.getSnapshot()`;
3. Knowledge that has been added since creating the snapshot can be retrieved with the method `SnapshotUndoManager.getObjectsToldSinceSnapshot()`.

When a fact is untold, this is implemented as follows. All knowledge is undone up to the state right before the fact was told. Then, each fact that was told since this state is re-told, skipping the fact to be untold. This is a very time-consuming method, especially when untelling a fact that has had many facts told after it.

Another problem with this method is that possible snapshots made *after* the fact to be untold, are lost, since new instances are made of the subsequent states due to the re-telling of facts. This is not tested yet, but the comments in the `JTP` code state this.

B.6 Packages

The different packages within `JTP`:

jtp contains basic classes and interfaces that will be implemented/extended in sub packages

jtp.cache for caching a Knowledge Store (doesn't seem to be used in `JTP`)

jtp.classifier for classification algorithms (OWL), for instance inferring that an object is a building because it has four walls and a roof

jtp.context contains Reasoning contexts for DAML, RDF, OWL

jtp.demod seems to be something for transforming knowledge

jtp.disp contains Dispatchers, that dispatch queries to Reasoners

jtp.fol contains the classes used for First Order Logic, classes like `CNFSentence`, `Clause` etc. Also contains parsers for this knowledge, to transform `Strings` to Java objects.

jtp.frame for frame-based reasoning:

jtp.frame.vocab vocabular for RDF, OWL etc.

- **jtp.frame.fc** for forward chaining using rules
- **jtp.frame.listen** "listens" to incoming knowledge to draw conclusions
- **jtp.frame.vc** Value collection, for searching through it and working with it

jtp.func contains functional reasoners

- **jtp.func.math** contains mathematical reasoners

jtp.gmp contains Modus Ponens reasoners

jtp.iw contains Inference Web reasoners

jtp.modelim contains Model elimination reasoners

jtp.proof manages proofs: goals, subgoals, and reasoning rules like and-introduction

jtp.rs contains the classes for Reasoning Steps

jtp.time contains Time reasoners

jtp.ui is a user interface to `JTP`

jtp.undo is used for retracting knowledge

jtp.util contains several utility classes

B.7 Using JTP

When writing a Java application that uses `JTP`, the following files have to be included in the `CLASSPATH` parameter of `java` and `javac`:

```
Antlr.jar
Base64.jar
http.jar
icu4j.jar
iiop.jar
jade.jar
jadeTools.jar
jakarta-oro-2.0.5.jar
jdom.jar
jena.jar
jtp.jar
log4j.jar
xercesImpl.jar
xmlParserAPIs.jar
```

Appendix C

Some implementation issues

C.1 Knowledge Manager

The Knowledge Manager in the `vs.knowledge` package has an interface providing the following functionality:

- Loading and saving of knowledge base;
- Asking queries;
- Telling facts;
- Retracting knowledge.

The class `BasicKnowledgeManager` provides this basic functionality for an OWL reasoning context. The class `ReasoningKnowledgeManager` extends this functionality with certain reasoning capabilities such as some basic queries, getting the inverse and symmetric relations, and the `DELETE` algorithm.

C.1.1 Retractions

In `JTP`, told facts are stored together with their derived facts. Because the Virtual Storyteller will keep track of the current state of the world, old knowledge will be deleted constantly and new knowledge will be added. Truth should be maintained throughout this process and this is a problem that takes some attention. There is a difference between `retract(`p`)` and `tell(`$\neg p$`)`. The first allows us to conclude neither p nor $\neg p$, whereas the latter allows us to conclude both p and $\neg p$, which is of course inconsistent (Russell & Norvig, 1995).

So retracting a fact is different from simply telling its inverse fact. Retract has some extra work to do. Let us consider a fact p which is of the form `(property subject object)`. Using `JTP`, there are two possible cases:

- p is a directly asserted fact. `JTP` allows us to `retract` this fact without problems, and its derived facts will automatically be removed as well.
- p is an inferred fact. Normally we would not be retracting these kind of facts but there are some cases where this might occur. p could be a consequent of the `owl:inverseOf` rule where a fact with its inverse property

was directly asserted. The same could be the case when p contains a property of the type `owl:SymmetricProperty` or `owl:TransitiveProperty`. And finally, p could be derived when it is derived from a sub property.

Let's define:

`sub(`p`)` as the collection of those facts that have the same subject and object as p but a property that is a subclass of the property of p.

`inverse(`p`)` as a fact with the same subject and object as p but the inverse property of p's property according to `owl:inverseOf`.

`symmetric(`p`)` as a fact with swapped subject and object in comparison to p and p's property being of the type `owl:SymmetricProperty`.

The `DELETE` algorithm for retracting a fact p of given form would be:

```
function DELETE($p$)
    retract $p$
    if $\exists$ inverse($p$) then retract inverse($p$)
    if $\exists$ symmetric($p$) then retract symmetric($p$)

    for each sub($p$) as $q$ do
        DELETE($q$)
    end
end
```

Even though the inverse and symmetric facts of a directly asserted fact will most often be inferred facts, it is also possible that these facts are directly asserted facts as well. The `retract` function in above algorithm means: "try to retract this fact". It might succeed, and it might not. The `DELETE` algorithm doesn't guarantee that after execution, p will not be true anymore.

C.2 Actions

The Class `ActionDB` gathers the Actions from the Action ontology as described in (Uijlings, 2006). It transforms them into `Action` objects representing the Java structure of an action. Internally, an Action object has the following structure.

It has a name, which is a `Symbol` corresponding to a concept in the Action ontology, for instance `|http://www.owl-ontologies.com/FabulaOntology#|::|PickUp|`.

Preconditions are stored in `m_preconditions` as a `Precondition` class. A Precondition consists of multiple `ConjunctiveVectors`. Each `ConjunctiveVector` again consists of `Literals`. Note that a `Literal` has the form: `(Relation Object_1 Object_2)`. One `ConjunctiveVector` represents an independent part of a precondition. With independent it is meant that between two `ConjunctiveVectors`, there are no mutual variables. This allows for checking each `ConjunctiveVector` on valid conditions separately.

It also has a conjunctive list of `Effects`, for intereffects to be added and deleted, and for effects to be added and deleted. An `Effect` contains a `CNFSentence` with the actual effect, and an `operator` that determines whether it is an effect to `ADD` or `DELETE`.

The `Map` `m_variableValues` contains entries that connect a variable name to a binding `Object` to replace the variable with. A possible entry of this map could be: `"AGENS" -> amalia`, mapping the variable `?AGENS` to its `Symbol` value `amalia`.

The variables of a certain `Action` can be set using a `Map` with variable mappings (assume `brutus` and `amalia` are `Symbols`):

```
// Brutus kicks Amalia
  Map m = new HashMap();
  m.put("AGENS", brutus);
  m.put("PATIENS", amalia);
  kickAction12.setVariables(m);
```

The conditions of an `Action` can be retrieved using the following methods:

```
public Preconditions getPreconditions();

public Vector<Effect> getInterEffects();

public Vector<Effect> getEffects();
```

The structures that these methods return are the original preconditions and effects, without its variables replaced. To get the versions in which the variables are replaced (dereferenced) by their bindings contained in `m_variableValues`, one must use the methods:

```
public Preconditions getDereferencedPreconditions();

public Vector<Effect> getDereferencedInterEffects();

public Vector<Effect> getDereferencedEffects();
```

C.3 Visualizing knowledge

The system uses Parlegraph to visualize knowledge. There is a Class `GraphGui` that implements a graph interface. The contents of the graph are defined using the Class `GraphFiller` which makes a model of the answers of the queries in the method `makeGraph()`. The vertices are being identified by their name, which is a `String`. The `GraphGui` is constructed using a certain `GraphFiller`.

So if a new graph needs to be made, extend the Class `GraphFiller` and implement the abstract method `makeGraph()`. This way, two `GraphFillers` have already been made as an example: `LocationGraphFiller` shows the location of the story world objects, and `FabulaGraphFiller` shows the fabula structure.

The `FabulaGraphFiller` has been used to visualize the very simple story of Plop, see Figure 4.11.

Some useful things to know about ParleGraph:

- The keys z, r, g and b make the color of a selected vertex black, red, green or blue, respectively.
- The DEL key removes a vertex and its connected edges.
- In the classes that implement the different View elements (V...), methods can be added to determine if you can move them, resize them, etc:

```
protected int getDefaultResizeMode () {
    return super.getDefaultResizeMode () &
                                        ~RM_ALLOW_RESIZE ;
}
```

This has been done for the Class `VEllipticTextVertex`.

C.4 Debugging and logging

The package `vs.debug` is intended to provide debug and logging classes for the Virtual Storyteller. It can contain Test classes and also provides a class called `LogFactory` which uses Java's `java.util.logging.Logger` class. See the Java SDK for a description of its functionality.

A very convenient log format is used: log messages are written to HTML files in the subdirectory `vs`
`log`. The `LogFactory` takes care of creating these log files and generating an index HTML file (`index.html`) where every log file is listed.

Use logging as follows:

```
import java.util.logging.Logger;
import vs.debug.LogFactory;

// To set it up:
  Logger logger = LogFactory.getLogger(this);

// Use it:
  logger.info ("This is an info log message");
  logger.severe ("This is important stuff!");
```

To create an index file, if you want, use the `LogFactory.generateIndex()` method.

References

Aristotle. (1907). *Poetics* (S. Butcher, Trans.).

Assanie, M. (2002). *Directable synthetic characters* (Tech. Rep.). University of Michigan Artificial Intelligence Lab.

Aylett, R. (1999). Narrative in virtual environments - towards emergent narrative. In *AAAI fall symposium on narrative intelligence* (pp. 83–86).

Blumberg, B., & Gaylean, T. (1997). *Multi-level control for animated autonomous agents: Do the right thing... oh, not that...* (Tech. Rep.). MIT Media Lab.

Brachman, R. J., McGuinness, D. L., Patel-Schneider, P. F., Resnick, L. A., & Borgida, A. (1991). Living with CLASSIC: When and how to use a KL-ONE-like language. In J. Sowa (Ed.), *Principles of semantic networks: Explorations in the representation of knowledge* (pp. 401–456). Morgan-Kaufmann.

Brewer, W. F., & Lichtenstein, E. H. (1981). Event schemas, story schemas and story grammars. In J. Long & A. Baddeley (Eds.), *Attention and performance* (pp. 363–379). Hillsdale, NJ.

Carroll, J. J., Bizer, C., Hayes, P., & Stickler, P. (2005, june). Named graphs. *Journal of Web Semantics*, *3*(4). Preprint, retreived December 15, 2005, from `http://www.websemanticsjournal.org`.

Cunningham, P. (1998). *CBR: strengths and weaknesses* (Tech. Rep.). University of Dublin, Computer Science Department.

Faas, S. (2002). *Virtual storyteller: An approach to computational story telling.* Unpublished master's thesis, University of Twente, Department of Electrical Engineering, Mathematics and Computer Science.

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

Hielkema, F., Theune, M., & Hendriks, P. (2005). Generating ellipsis using discourse structures. In J. Spenader & P. Hendriks (Eds.), *Proceedings of the esslli '05 workshop on cross-modular approaches to ellipsis* (pp. 37–44). Heriot Watt University, Edinburgh.

Java Agent Development Framework (JADE). (n.d.).
`http://jade.tilab.com/`

Java Theorem Prover (JTP). (n.d.).
`http://www.ksl.stanford.edu/software/JTP/`

Johnson, N. S., & Mandler, J. M. (1980). A tale of two structures: underlying and surface forms in stories. *Poetics*, *9*, 51–86.

Kelso, M. T., Weyhrauch, P., & Bates, J. (1992, December). *Dramatic presence* (Tech. Rep.). School of Computer Science, Carnegie Mellon University.

(Submitted to PRESENCE, The Journal of Teleoperators and Virtual Environments, MIT Press, 1992)

Kooijman, R. (2004). *De virtuele verhalenverteller: voorstel voor het gebruik van een upper-ontology en een nieuwe architectuur.* (Tech. Rep.). University of Twente, Department of Electrical Engineering, Mathematics and Computer Science.

Mandler, J. M., & Johnson, N. S. (1977). Remembrance of things parsed: Story structure and recall. *Cognitive Psychology*, *9*, 111–151.

Mateas, M., & Stern, A. (2000). Towards integrating plot and character for interactive drama. *Working notes of the Social Intelligent Agents: The Human in the Loop Symposium.*

Mateas, M., & Stern, A. (2003). *Façade: An experiment in building a fully-realized interactive drama* (Tech. Rep.). Literature, Communication and Culture and College of Computing, Georgia Tech.

Meehan, J. (1981). Inside computer understanding - five programs plus miniatures. In R. Schank & . K. Riesbeck (Eds.), (pp. 197–226). Lawrence Erlbaum Asoociates.

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

Named Graphs API for Jena (NG4J). (n.d.).
`http://www.wiwiss.fu-berlin.de/suhl/bizer/ng4j/`

Oatley, K. (1994). A taxonomy of the emotions in literary response and a theory of identification in fictional narrative. *Poetics*, *23*, 53–74.

Ontology Web Language (OWL). (n.d.).
`http://www.w3.org/TR/owl-features/`

Pérez y Pérez, R., & Sharples, M. (2004). Three computer-based models of storytelling: BRUTUS, MINSTREL and MEXICA. *Knowledge-Based Systems*, *17*, 15–29.

Propp, V. (1997). *De morfologie van het toversprookje: vormleer van een genre.* (S. Butcher, Trans.). Utrecht: Spectrum. (Original work published 1968)

Protégé. (n.d.).
`http://protege.stanford.edu/`

Racer. (n.d.).
`http://www.sts.tu-harburg.de/~r.f.moeller/racer/`

Rawlings, J., & Andrieu, J. (2003). *Player-protagonist motivation in first-person interactive drama: A framework for aristotelian interactive drama* (Tech. Rep.). Realtime Drama, Inc.

Rensen, S. (2004). *De virtuele verhalenverteller: Agent-gebaseerde generatie van interessante plots.* Unpublished master's thesis, University of Twente, Department of Electrical Engineering, Mathematics and Computer Science.

Resource Description Format (RDF). (n.d.).
`http://www.w3.org/RDF/`

Riedl, M. O. (2002). *Actor conference: Character-focused narrative planning* (Tech. Rep. No. 03-000). Liquid Narrative Group, North Carolina State University.

Riedl, M. O. (2004). *Equivalence between narrative mediation and branching story graphs* (Tech. Rep. No. 04-004). Liquid Narrative Group, North Carolina State University.

Russell, S., & Norvig, P. (1995). *Artificial intelligence: A modern approach.* Prentice Hall.

Schärfe, H. (2002). Possible worlds in narrative space. *IMPACT - an electronic journal of formalisation in media, text and language.*

Sgouros, N. M. (1998). Dynamic generation, management and resolution of interactive plots. *Artificial Intelligence*, *107*, 29–62.

Slabbers, N. (2006). *Narration for virtual storytelling.* Unpublished master's thesis, University of Twente. (to appear)

Suggested Upper Merged Ontology (SUMO). (n.d.). `http://ontology.teknowledge.com/`

Szilas, N. (1999). *Interactive drama on computer: beyond linear narrative* (Tech. Rep.). American Association for Artificial Intelligence.

Szilas, N. (2003). *IDtension: a narrative engine for interactive drama* (Tech. Rep.). IDtension.

Tapiero, I., Van den Broek, P., & Quintana, M.-P. (2002). The mental representation of narrative texts as networks: The role of necessity and sufficiency in the detection of different types of causal relations. *Discourse Processes*, *34*(3), 237–258.

Theune, M., Faas, S., Nijholt, A., & Heylen, D. (2003). The Virtual Storyteller: Story creation by intelligent agents. In *Technologies for Interactive Digital Storytelling and Entertainment (TIDSE)* (pp. 204–215).

Theune, M., Rensen, S., op den Akker, R., Heylen, D., & Nijholt, A. (2004). Emotional characters for automatic plot creation. In S. Göbel et al. (Eds.), *Technologies for Interactive Digital Storytelling and Entertainment (TIDSE)* (pp. 95–100). Springer-Verlag.

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

Trabasso, T., Van den Broek, P., & Suh, S. Y. (1989). Logical necessity and transitivity of causal relations in stories. *Discourse Processes*, *12*, 1–25.

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

Uijlings, J. (2006). *Designing a virtual environment for story generation.* Unpublished master's thesis, University of Amsterdam. (to appear)

Van Dijk, T. A. (1980). Story comprehension: An introduction. *Poetics*, *9*, 1–21.

Van Gils, F. (2005). *Potential applications of digital storytelling in education* (Tech. Rep.). University of Twente, Department of Electrical Engineering, Mathematics and Computer Science.

Vromen, J., & Bloom, N. (2005). *Creative problem solving for the virtual storyteller* (Tech. Rep.). University of Twente, Department of Electrical Engineering, Mathematics and Computer Science.

Wilensky, R. (1983). Story grammars versus story points. *The Behavioral and Brain Sciences*, *6*, 579–623.

WordNet. (n.d.). `http://wordnet.princeton.edu/`

Young, R. M. (2002). *Story and discourse: A bipartite model of narrative generation in virtual worlds* (Tech. Rep.). Liquid Narrative Group, NC State University.

Zillmann, D. (1994). Mechanisms of emotional involvement with drama. *Poetics*, *23*, 33–51.