UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA
Departamento de Software e Inteligencia Artificial- Lenguajes y Sistemas Informáticos

A COMPUTATIONAL MODEL FOR AUTOMATED EXTRACTION OF STRUCTURAL SCHEMAS FROM SIMPLE NARRATIVE PLOTS.

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Carlos León Aznar

Bajo la dirección del doctor

Pablo Gervás Gómez-Navarro

Madrid, 2011

ISBN: 978-84-694-2084-3

Un modelo computacional para la extracción automática de esquemas estructurales de tramas narrativas simples

Tesis doctoral

Presentada por
D. Carlos León Aznar

Dirigida por
Prof. Dr. D. Pablo Gervás Gómez-Navarro

Facultad de Informática
Universidad Complutense de Madrid
Madrid, 2010

A Computational Model for Automated Extraction of Structural Schemas from Simple Narrative Plots

Ph. D. Thesis

by
D. Carlos León Aznar

Supervised by
Prof. Dr. D. Pablo Gervás Gómez-Navarro

Facultad de Informática
Universidad Complutense de Madrid
Madrid, 2010

Thanks

I WOULD LIKE TO THANK my supervisor, Dr. Pablo Gervás, for everything I have learnt from him and for all his support. This work would not have been possible without him.

Thanks as well to Dr. Mark Riedl, who received me in Los Angeles at the Institute for Creative Technologies, where I learnt so much.

Thanks to the NLG group at Aberdeen University, specially to Dr. Graeme Ritchie, who revised my work and gave absolutely valuable advise for it. This thesis would have not been the same without his influence.

The fruitful collaboration with the NIL group has also been crucially positive during the development of the ideas for this work. Also, the scientific relationship emerging in the Department of Software Engineering and Artificial Intelligence (DISIA) strongly improves every research developed in it. I feel really thankful towards both groups.

Thanks to the reviewers, Dr. Albert Gatt, Dr. Amílcar Cardoso and Dr. Daniel Mozos, the final version of the thesis has been greatly improved.

I would also like to thank my parents and friends for all the support I have received during this period.

Abstract

Building computational systems capable of creating and interpreting narrative content has been an objective of Artificial Intelligence since its beginnings. Its development, however, has been blocked by what is commonly known as the *knowledge acquisition bottleneck*, which does not permit story processing in the large.

In this dissertation, a computational system that tries to take one step forward in the target of making it possible to process narrative content on a larger scale is presented. Two stages of research towards this goal are detailed. A semantic approach to narrative processing not yielding satisfying results is first explained. Then, a different system modelling a focus shifting towards a structural management that provided better results is shown.

The current state of the art is studied in detail. Empirical validation has been carried out to prove to the possible extent the proposed hypothesis (the plausibility of structural processing for narrative content). Discussion about the most important aspects and design decisions is also included, and possible future lines of investigation are also exposed.

Contents

List of Figures

List of Algorithms

List of Stories

List of Formal Stories

List of Story Graphs

Part I

A Computational Model for Automated Extraction of Structural Schemas from Simple Narrative Plots

Chapter 1

Introduction

Computational generation of narrative content has intermittently received the focus of the Artificial Intelligence community from its beginnings. It is possible to find several works on this field based on different points of view and with different results and conclusions. This interest has always been present, as narrations are a basic form of human communication. As far as we know, all human societies use narrations to transmit knowledge of every form. Building systems capable of interacting with humans using narrative structures is therefore useful since it could make it possible to create a more natural way of communication between machines and people.

However, no globally accepted model about how a computational system should manage narrations has been created. This is mostly due to the fact that processing stories in the way humans do requires a strong model of human psychology, and science has not developed such a model yet, at least not a complete one.

Semantic processing of narrations tries to reach, from one or another point of view, human understanding or generation of stories. In general, semantic approaches have been the predominant tendency in automatic story generation systems and other studies in the field [Meehan, 1976, Turner, 1992, Bringsjord and Ferrucci, 1999, Riedl, 2004]. While nothing prevents from success when reproducing the way in which humans perform narrative generation or understanding, current results evidence that this is not an easy task. Problems affecting Artificial Intelligence in general (knowledge acquisition bottleneck, efficiency for complex domains, and others) also appear in Computational Narrative. Since an important amount of knowledge is required for implementing these systems, the cost of building these big knowledge bases has been a barrier in the deployment of narrative creation programs in real applications beyond academic

research.

These drawbacks seem to block the computational implementation of concepts from modern Narratology [Herman, 2000] and other disciplines like Psychology [Kelly, 1955], which conceive narrations as cognitive processes not totally describable in terms of structural properties of stories, as opposed to the structuralist point of view of classic Narratology [Propp, 1928, Barthes and Duisit, 1975]. However, current computational techniques, and not only the state of the art in knowledge representation, are still far from being able to model the inner processes that govern human understanding of stories. Realizing advances in Narratology as computer programs is not directly feasible because Narratology assumes conceptual capabilities exceeding those currently available in machines.

According to these characteristics of Computational Narrative, one major flaw of most narrative generation systems is that, while they are built over generic models, the amount of different stories they can process, once implemented, is somewhat reduced. This is one of the reasons that has prevented computational generation of stories from being applied to production systems in industry. In general, this has occurred because of the *knowledge acquisition bottleneck*. Knowledge acquisition bottleneck is an important defect of declarative approaches to Artificial Intelligence that affects not only Computational Narrative, but also many other fields of computation.

This inherent difficulty has not prevented Artificial Intelligence from trying to build computational models that, while not mimicking the human ways of constructing narrations –because they are still unknown as a whole– yield stories similar to those humans can write. At least, to the extent of being *recognizable* for an external audience as human-written narrations. The relative merits of these systems, while not definitive, have made it possible to advance the current state of the art in Computational Narrative.

This point is crucial in the analysis of Computational Narrative systems: their quality is generally measured in terms of their impact on human judgement. Just as other subfields of Artificial Intelligence do, the target is the final interpretation of the story by humans (although many systems certainly try to implement cognitive models). This approach is the one that has been followed by this research: the objective is to create a computational system whose outcome is acceptable by humans as a story, although the creation process is admittedly far from the way in which humans create stories.

Along the following chapters, a computational system that tries to partially overcome the inherent problems of semantic approaches to Computa-

tional Narrative will be presented. It is based on the previously introduced idea: the objective is not to create a computational model that imitates human behaviour. Instead, the focus is to obtain a system able to generate stories that are identifiable as such by humans.

In particular, the scientific focus of this research has been put on trying to leverage the current capabilities of the scale at which stories can be generated for a single system. As knowledge acquisition is clearly an influential bottleneck, this work proposes a semi-supervised narrative schema acquisition process. The objective is to broaden the scale of generation by allowing to iteratively add new narrative schemas with a reduced effort. In this way, adding domain knowledge can be carried out in a more efficient way (from the user's perspective).

1.1. Motivation of the Research

Developing the current state of the art in Computational Narrative can help to understand better, at least to some extent, the processes underlying narrative creation. If new algorithms are built, models can be empirically tested, presumably in a more efficient way than human behaviour. Therefore, presumably as well, advances in computational Narrative, then, can make it possible to find out inner details about psychological aspects of human narrative.

Furthermore, current interest on Computational Narrative evidences that real applications can benefit from improvements on the field. Videogames can provide a better user experience by having non-player characters create a coherent story, possibly adapting to the user behaviour. Recommender systems can automatically create a narrative fiction based on human experience, and large execution logs can be summed up in a short, meaningful narration. The set of possible applications is vast. Some discussion about this is presented in Chapter 9.

Against this background, while several computational narrative systems already exist, the amount of different stories that they can generate is restricted, as it has been previously introduced. In order to generate new stories, new domain information must be input by hand, which raises the cost of improvement and scaling of the processing capabilities. This is costly and previous evidence has shown that it prevents the previously created and implemented systems from growing.

There exists large interest on finding new methods to reduce the cost of broadening generative capabilities in classic approaches to story generation, like those previously described in literature [Riedl and León, 2008,

Chambers and Jurafsky, 2008, Bringsjord and Ferrucci, 1999]. However, it is well known that automating knowledge acquisition is a difficult task. Science knows little about the underlying mechanisms driving human learning, so computation cannot directly use that knowledge to formalize knowledge or structure acquisition algorithms.

However, instead of trying to solve the knowledge acquisition issue as a whole, it makes sense to restrict the scope of the problem to certain subdomains. Concretely, narrative texts have certain properties which can be used to create constrained models for schema acquisition, and this characteristic can drive a more successful process.

It makes sense, then, to create a model that, if not totally substituting manual management, at least is able to reduce the amount of work that humans must do in order to create new knowledge. Therefore, the main motivation of this research is the benefit that studying the possibilities of this kind of processing in order to help to automate the knowledge acquisition process in computational narrative could bring. Since it is hypothesized that it is possible to reduce the amount of human work needed to extract narrative structures, this is considered a justified motivation.

As it will be shown in next chapters, trying to create a general model covering different types of narrations and able to extract information of any kind from stories is a too ambitious task. Although that would be a very important advance for this field, this research has tried to remain in a focused approach. While several other ways of gathering narrative structures are possible, a full study about every alternative is outside the scope of this work.

1.2. Objectives

Given that improving the generative capabilities of automatic story generation systems could be useful for the research community, the objective of studying how this can be done is proposed. In particular, the objective of this research is to create a model that, at least partially, improves the current state of the art in the automation of the addition of structures needed for story generation systems for generation stories in new domains.

More concretely, the proposed objectives for this work can be summarized in four sub-objectives that must be accomplished:

- Creating a computational system receiving some set of human-written stories as input.

- Extracting some instances of a certain structure from them, minimizing to the possible extent the need for human intervention. This structure will be defined as part of the research
- Using those instances to automatically generate stories.
- Validating the whole process through human judgement, trying to restrict, to the possible extent, the interface between the system and humans to realized stories readable in natural language.

The inherent characteristics of the research carry a problematic issue: it is extremely difficult to prove with absolute certainty that the model generates good stories without checking its quality against human criteria, and these criteria varies depending on the human evaluator. Therefore empirical validation based on human interaction will be used to check that the extent to which the proposed objectives have been reached. The percentage of agreement will also be measured.

It is well known that automatic acquisition of structures which can be used to generate content of similar quality as the one that humans can produce is far from trivial. Therefore, in order to make the objectives feasible, the objectives consider that the complexity of the processed narrations will be low. This means that no complex causality relations or extremely long stories are considered.

Only *simple* narrations are addressed in this work. This is a conflicting definition from a conceptual point of view since there are many different and not necessarily compatible definitions already (as detailed in Chapter 2). For this thesis, a simple narration is a linear *sequence* of time-ordered events. At this point, no assumption is made about the content of simple stories beyond the fact that they must be *coherent*, that is, a reader must consider that all the content is causally connected in a logic unit (all events happen because of a reason). Moreover, the stories that this thesis consider must be *complete*: the reader must consider that the action is over and all the conflicts are solved.

This definition of simple narrations, while slightly loose, cannot be completely formal, since, even with the requirements explained above, there are several possible interpretations of “coherence” and “complete”. The rest of this dissertation will exemplify, with instances of simple narrations, the type of stories that the proposed system can process. Stories with sophisticated order of events regarding time and not completely reasoned causality are considered “complex” and are not addressed in this work.

For instance, “John went to the cinema and, after that, he went home” is a simple story. “When he arrived at home, John felt that he should have

done something else" is not a simple narration because the reader can easily perceive that there are some parts of the story that are not explained (why does John think that he should have done something else?), and, additionally, events are not expressed in time order.

More concretely, the target narrations for this research must satisfy these requirements:

- Narrations will only contain a single narrative thread or plot. No complex, multiple causal lines are allowed.
- Narrations will not be ordered by complex time relations. Strong restrictions are imposed on narrations regarding time:
  - Every event in a narration will last a single *unit* of time.
  - In narrations, events are ordered by time.

A formal definition of narrations and the characteristics that are imposed for them to be processable by the proposed model are presented in Chapter 3. The general objectives of this research, then, will only be applicable to this restricted definition of narration. No claim is made about the capabilities of the system beyond that definition.

1.3. Hypothesis

Section 1.2 just introduced the research objectives for this doctoral work. The proposed solution will be explained in detail in the following chapters and it is based on certain research hypotheses which are explained in this section.

The presented hypothesis were progressively developed during the research work. Initially, the focus was set on pure story generation. Since the intention was to offer some way to partially break the barrier of knowledge acquisition, it was hypothesized that an evaluation function for narrative content, as a change of perspective regarding story generation, could unlock the development.

However, as it will be explained in Chapter 4, the hypothesis was not correct and the evaluation function itself did not yield a satisfying solution. However, the knowledge acquired in the creation of this version of the system helped to analyse the properties of narrations from a different perspective, namely a structural approach. After this analysis, the next hypotheses were formulated:

> A certain set of *structural* patterns in a narration is enough to create stories rated as coherent by humans. That is, once these structural relations are known, the coherence in these terms can be determined just by analysing the presence or lack of them.

That is, it was hypothesized that there exist some structural patterns in narrations that can help to create narrations that humans identify as such. Since it was though that structural patterns were more easily collectible from texts, the next hypothesis, which is based on the previous one, was also formulated as complementary:

> The instances of the hypothetical structural patterns of narrations can be collected under a certain degree of supervision by a computer algorithm. Therefore, patterns extracted from coherent stories and patterns extracted from non-coherent stories can be collected and used as acquired content for generating.

That is, if these structural patterns really exist and they can be formalized as a computational representation, they can be gathered and used.

These two parts of the more general hypothesis state that structural characteristics of text, leaving out semantic features, can be used to perform story processing. This process is explained in the rest of this dissertation. It will be shown how it the plausibility of these hypotheses have been partially proven. An example domain has been used to show that, at least for a restricted domain, the hypotheses make sense.

It is important to make explicit again that this kind of approaches to Artificial Intelligence based on human criteria are always conflictive. As the research assumes coherence based on human judgement, reaching an absolute conclusion about the success of the system is not possible. Only empirical demonstration, which is subject to discussion, can be carried out.

1.4. Methodology

The development of this work has been driven by the application of a research methodology based on the study of previous work in the field and the application of available technologies to prove, to the possible extent, the validity of the previously presented hypotheses.

A sequential approach to propose a scientific solution was carried out: identification of the problem, formulation of a hypothesis, solution proposal, experimentation and analysis of the results.

The first stage of the research consisted on the collection of relevant scientific and non-scientific literature in the area of Computational Narrative. Therefore, work both from Narratology and Artificial Intelligence was taken into account to learn from a rich enough background before tackling the identification of useful objectives.

After the state of the art was studied, a problem was identified and the proposal of an objective was made, namely advancing the current capabilities in the field of knowledge acquisition for Computational Narrative, as detailed in Section 1.2.

It was then hypothesized, based on previous research, that knowledge acquisition for computer generated narrative could be based on the causal properties of typical narrative. After this hypothesis was refined and considered plausible enough, a model based on informed brute force generation and evaluation of narrative structure was created.

Once implemented, the model was tested against human criteria in order to validate the hypothesis. Given that hardly formalizable human opinion is intrinsically involved in the solution, the hypothesis cannot be totally validate for *every single case*. However, tests suggest that the approach is valid *in general* for most simple narrations.

After the evaluation was done, analysis of the results and discussion about the collected conclusions was made, as it can be read in Chapters 8 and 9.

As most research works, the development has been supported by the use of different tools that have prevented from carrying out all the work from scratch and manually. Because the domain is so involved with Computer Science, these tools have mainly been computer programs. Free software has been chosen because of its availability and, at least for the selected tools, its robustness.

Apart from a personal computer (AMD Phenom™9550 Quad-Core Processor), the implementation has been carried out using the SWI-Prolog system [Wielemaker, 2010]. It is a free Prolog interpreter, and it is reasonably efficient and modern, it is also actively developed.

All the documentation has been typeset using the LaTeX typesetting system [The LaTeX Project, 2010], including this dissertation. For statistical analysis a plot representation, the R language and system has been used [R Development Core Team, 2010]. Short post-processing scripts mainly used to store data and bulk conversions have been programmed in Lua [Lerusalimschy et al., 2006] and Ruby [Thomas and Chad Fowler, 2005]. All the development, tests and analysis have been carried out in GNU/Linux (Arch Linux) [Vinet and Griffin, 2010].

1.5. Structure

This dissertation is structured as follows:

- This first chapter, the introduction, has tried to briefly present the motivation and hypotheses leading to this research.
- The second chapter will extensively examine the previous research and work that, either theoretically or practically, has influenced the current research. A special focus is given to similar approaches to story processing with computers.
- Chapter 3 details the formal model of narrations that has been created for developing the current solution. The conceptual assumptions and the computational formulation are included.
- In the fourth chapter a preliminary study of how semantic approaches to story generation based on an explicit evaluation function is proposed. Results of this approach are included, and the conclusions obtained from this work are shown and analysed. This first stage of the doctoral research led to a new model based on structural content.
- This structural content of narrations is addressed in Chapter 5. The structural relation that was created is presented, and the way in which narrations are managed from this perspective is presented.
- Chapter 6 is devoted to the study of a system capable of collecting structural rules for further computational processing of stories.
- The implementation of the theoretical models presented in chapters 5 and 6 is explained in the seventh chapter. The empirical results partially proving the plausibility of the main research hypothesis are included in this chapter.
- Discussion about the whole process, hypotheses and ideas can be read in Chapter 8. The relation of this presented research with other systems, its benefits and its drawbacks are all discussed in detail.
- Finally, as a summary, the last chapter presents the most important conclusions gathered from the development of this research, and the ways in which it could be improved as part of the future work are discussed.

Additionally, the appendix presents the original stories that have been used as corpus. Concrete references to this content can be found in the chapter documenting the implementation, Chapter 7.

1.6. Chapter Summary

This chapter has introduced the main motivations and hypotheses that have led to the development of this research work. A brief description of the narrations that the proposed system will be able to handle has been given, and the objectives, namely the improvement of the current state of the art in narrative generation, have been presented.

Chapter 2

Previous Work

THIS CHAPTER IS DEVOTED TO THE STUDY OF PREVIOUS RESEARCH and systems whose development influences the current work. It mainly captures information about theoretical and practical techniques related to the design and development of this research.

The chapter has been written with the intention of summarizing the most important ideas of the studied research projects regarding the presented work. Many important concepts of previously developed systems will not be addressed here in order to keep the text simple. Therefore, only those concepts influential or relevant have been included.

Special focus has been given to previous approaches to story generation. These works have been radically influential in the proposed system, and, since the final objective of this research is the computational processing of stories, it was decided to put special attention to this field.

After the exposition of the proposed solution for the problems identified in Chapter 1, the relation of the solution with the previous work shown in this chapter will be discussed in Chapter 8. Some of the ideas and contributions detailed in this chapter are directly used, while some other approaches are not. This has been clearly justified in the Chapter 8.

2.1. Narratology

Narratology is the science devoted to the structuralist study of *narrations* and the way in which humans understand and use it. Since the approach to narrative content that Narratology proposes is strongly related with Computational Narrative, it is useful to briefly introduce some of its most important characteristics here.

Narratology is focused on the inner characteristics of narrative and on

its similarities and differences with other kinds of communication. Narratology studies these characteristics trying to be partially isolated from other phenomena like linguistics or semiotics. In this way, Narratology remains as an independent discipline.

Some descriptions of *narration* define it as a tale, some others as the action of telling something, and others focus the description on the structural parts of the narrative discourse [Real Academia Española, 2010]. This structuralist approach to the description of narrations sets the base for modern Narratology, although the science of Narratology is retrospectively considered, perhaps having its roots in Aristotle's *Poetics* [Aristotle, 1974]. Aristotle postulated that the imitation of the real world creates arguments from which the most important units are chosen and ordered in a plot. Imitation of actions in the real world (*praxis*) forms an argument (*logos*) from which events or fundamental units composing the plot (*mythos*) are selected and ordered.

Narratology has its roots on the structuralist analysis of texts, based on the work by Barthes, Genette, Todorov and others. [Barthes and Duisit, 1975, Todorov, 1977, Genette, 1979]. Next sections show some important conceptions and ideas related, to some extent, to the current research.

More modern, post-structuralist perspectives of Narratology have been developed. For instance, Cognitive Narratology [Herman, 2000] considers Narratology as a psychological phenomenon, and proposes a study of narrative aspects in terms of a more cognitive perspective.

Next sections introduce some of the most influential models of narration with regard to classic and modern Narratology. It will be noticeable that it is possible to find clear analogies between different models, although they use to be rooted on different conceptions of narrative.

2.1.1. Fable and Discourse

Chatman divides the narration in *story* and *discourse* [Chatman, S., 1986]. In this way, there is a clear division between the content that is to be transmitted and the manifestation or realization of that discourse. He states that *existents* are those entities which take part in the development of the narration and that the chain of those *events* happening to these existents form the story.

The discourse is the particular realization of a story, and it is manifested and transmitted. The media and techniques can be diverse: voice, perspective, time... Generating a discourse consists not only in choosing adequate words to tell the story. Ordering the events is equally important,

choosing the perspective of locating the action in some place or time is crucial.

Crawford [Crawford, 2005] states that a narration is not about time or spatial relations. He claims that a narration is a high-level structure based on causality. He differentiates characters from narrative events, also from the set of events or event from the world in which everything takes place.

A deeper structure is proposed by Russian formalists, in particular the distinction between what is being told in a story, that is, the content that will be transmitted from the narration to the audience ordered by their natural sequence, called the *fabula*, and the order in which the set of events or facts are laid out in the narration, called the *sjuzet* [Propp, 1928, Viktor Shklovsky, 1917].

In general, full consensus regarding the structures defining Narratology has not been reached. For instance, McKee considers that narrations are structures that can be represented as a tree in which nodes are ordered according to discourse and chronology [McKee, 1997]. In general, his approach is based on films. This tree-layout is also rendered as an ordered set of events. However, this tree structure considers the root node as the *story*, decomposed in *acts* which are decomposed in *sequences*, decomposed in *scenes* which, finally, reach leaf nodes called *beats*. These atomic *beats* contain the narrative meaning. According to McKee, several beats can change the charge of the scene. Field also classifies narrations in different levels in a slightly simpler model [Field, 1998]. Genette identifies five concepts in Narratology to cope with more complex structures of narrations beyond simple tales [Genette, 1966, Genette, 1969, Genette, 1972]. These five concepts are *order*, *frequency*, *duration*, *voice* and *mood*.

Another formalization similar to the one that Russian formalists created is the one followed by Bal [Bal, 1998]. Her formalization considers the *fabula* or the set of entities and events that the author creates in order to shape the story. The discourse is the particular order and selection of events from that fable. Finally, the presentation is the concrete realization of the message of the author (the text in a novel, for instance) which is used to convey the content to the audience.

An analogy can be found between Bal's work and the three stages pipeline structure that Reiter and Dale propose [Reiter and Dale, 2000]. This make it simple to apply a direct mapping between Bal's ideas and Computational Narrative approaches. The first stage as proposed by Reiter and Dale, the *content planning*, can be identified with the fabula, as it consists on a set of concepts, entities and events. The *sentence planning* creates a middle representation in which the content is organized in linguistic entities (sentences), and this could match the "discourse" as Bal

defines it. Finally, the surface realization compares to the presentation stage, since its outcome consists on a human readable version of the initial content. This has been carried out by systems like the recent work by Peinado [Peinado et al., 2008].

Two entities are important in Narratology, in such a way that they deserve special attention: *characters* and *plots*. They are examined in Sections 2.1.2 and 2.1.3.

2.1.2. Characters

Characters are a fundamental constituent in a story. Aristotle considered that the most important element is the action [Aristotle, 1974], but it can be assumed that characters appear, in several forms, in virtually every type of story.

McKee defends that plot characters are just the two parts of the same narrative phenomena, in such a way that it makes no sense to talk about the plot without characters and vice-versa.

Chatman states that entities in a story are divided in *characters* and elements in the *scenario*. Characters are usually humans or humanoid beings (for instance, animals in fables) and the elements in the scenario are places and objects.

2.1.3. Plot

The plot of a story is traditionally seen as a sequence of fact or events in which causality links them meaningfully [Foster, 1941, Chatman, S., 1986].

The first definition of a story in terms of its constituent parts was first tackled by Propp [Propp, 1928]. Propp defines a set of *character functions*, which are those actions that define the role of some character once they have been carried out by her or him. The main contribution of Propp is to determine the functions that take place in *all* tales (at least in the corpus he used, Afanasiev's tales). For instance, the role of the hero receiving a gift, or the princess getting married are classic functions which can be instanced in several domains.

McKee states that plots are the result from the conflicts in the story. Three types of conflict are defined by him: *cosmic* conflicts, those which confront Good and Evil, *social* conflicts, taking place among a group of people, and *personal* conflicts, in which the conflicted character faces her or his own problems.

In general, Narratology has tried to find different abstractions of the types of existing plots, with little consensus. Murray [Murray, 1997] iden-

tifies that there are strong differences between the number of different possible abstract plots that different authors define.

2.2. Computational Approaches to Narrative

Computational Narrative is devoted to the representation and processing of narrations by computers. One of the most studied subfields of Computational Narrative is the generation of stories, which is studied here.

Computational Narrative appears in the 70's with the interest of reproducing the human way of understanding and processing stories.

Roger Schank has been the pioneer of the study of the impact of narrative on humans from a formal, computerizable point of view. Schank stated that the way in which memory works is not only based on processes that manipulate mental data, but instead as continuous recalling and adapting process of previous stories that define our world, like in Case Based Reasoning (whose roots base on Schank's work) [Schank and Abelson, 1977, Schank, 1982].

Schank introduces the term *script*, which consists on short units of sequential knowledge about the typical steps to be carried out in a certain situation. For instance, Schank proposes the classical example of the restaurant: one gets into the restaurant, the he orders food, he eats it, etcetera. This common sense knowledge, according to Schank, is a very important unit of knowledge in humans.

To implement this kind of semantical knowledge in computers, Schank proposed *Conceptual Dependency*. Conceptual Dependency is a formal method based on primitive actions and relations between them and their objects [Lytinen, 1992].

Programs like SAM [Cullingford, 1981] or PAM [Wilensky, 1981] started the development of a theory of Computational Narrative based on characters and the way in which they tried to reach their objectives. Some models of the way in which memory processes could be implemented were also addressed [Kolodner, 1980].

This studies triggered the inverse approach: instead of trying to understand the human methods, it made sense to try to generate narrative through models of the human methods, as it was carried out in TALESPIN [Meehan, 1976, Meehan, 1981] and others.

After this initial stage, Artificial Intelligence applied to computational approaches to narrative in these terms was left out, with the exceptions of some systems [Turner, 1992, Mueller, 1987].

Narrative Intelligence [Mateas and Sengers, 1999] and other multidisciplinary projects on Computational Narrative renew the interest of the field in the XXI century. Theoretical formalization of narrative, development of story generation systems, human narrative knowledge extraction systems, interactive digital narrative and narrative intelligence systems are proposed, from this moment, as possible applications of Computational Narrative.

2.3. Story Generation Systems

This section studies the most relevant work regarding story generation in scientific literature. It is possible to notice that, although families of systems can be clearly identified, they all follow different approaches and focus on different aspects of storytelling. Systems are presented in time order to put focus on the historical evolution of this research field.

2.3.1. Novel Writer

The first storytelling system in literature is NOVEL WRITER [Klein et al., 1973]. NOVEL WRITER generates murder stories within the context of a weekend party. Generation in NOVEL WRITER is tackled by simulating worlds in which character's behaviour is governed by probabilistic rules that, when applied, change the state of the world.

Narrative is then considered to emerge from those changes, and the resulting sequence of events and states conforms the narration. The simulation just changes the state of the characters, and the scenes that link the events in the story are hard wired in order to create the weekend party environment. This approach makes the system rather inflexible, and the use of this fixed structure somehow resembles story grammars.

NOVEL WRITER's input includes the description of the world and the characters' initial relations. Random ingredients in the processing sums to the input to compute the motives of the murder and the events in the story.

Although NOVEL WRITER permits the use of private universes in a somehow uncoupled way (loading the universe in memory from a disk, using it, and then storing it back in the disk), the description available does not explain how to change domains by using different universes, if possible.

Story 1 shows an episode of a story generated using Novel Writer.

NOVEL WRITER shows the first effort for automatic storytelling in scientific literature. Although it only allows to generate one fixed type of story,

The day was Monday. The pleasant weather was sunny. Lady Buxley was in a park. James ran into Lady Buxley. James talked with Lady Buxley. Lady Buxley flirted with James. James invited Lady Buxley. James liked Lady Buxley. Lady Buxley liked James. Lady Buxley was with James in a hotel. Lady Buxley was near James. James caressed Lady Buxley with passion. James was Lady Buxley's lover. Marion following them saw the affair. Marion was jealous.

Story 1: Episode in NOVEL WRITER (part of a larger story).

this system addresses some of the problems which are later studied and, in some cases, solved, like story structure, character behaviour, and the user input when generating the story.

2.3.2. Grammars: Rummelhart and Joseph

Rumelhart proposes a story generation system based on generative grammars. Structural properties of narrations are captured in form of rules in a top-down approach [Rumelhart, 1975]. While this structuralist approach is quite common in some other domains (Narratology, for instance), it is easily observable that, while Rumelhart's system and others alike [Lee, M., 1994, Lang, 1999] practically ensure coherence and correctness, their productivity of original stories rapidly decreases, since the maintenance and update of such complex grammars is too costly.

Although this is a more recent system, Lang's JOSEPH proposes a story grammar (following Rumelhart's work [Rumelhart, 1975]) in which it is claimed that the basic structure of what constitutes a story is preserved, while also reproducing a model of how humans creates stories [Lang, 1999].

An example of the output generated by Joseph can be read in Story 2.

2.3.3. TaleSpin

Meehan's TALESPIN creates stories about woodland creatures, as can be read in Story 3 [Meehan, 1976]. TALESPIN follows a character-centred approach in which goals and events direct the story. TALESPIN creates the story by carrying out a plan-based reasoning with forward-chaining (events to consequences) and backward-chaining, by finding which events are to be executed in order to achieve the goals that the animals in the stories have. The planning system also considers goal decomposition, thus obtaining

once upon a time there lived a peasant. peasant was hungry. one day it happened that the peasant met christ. when this happened, peasant felt awe. in response, peasant begged christ to provide food. christ told peasant to eat ram. when this happened, peasant felt obedient. in response, peasant made it his goal that ram would be eaten. peasant trapped ram. ram whacked peasant. peasant believed it impossible that ram would be eaten. peasant was hungry.

Story 2: A short story generated by JOSEPH.

sub-goals which are fulfilled by sub-planning.

TALESPIN addresses several important issues in storytelling. It fully focuses on characters, and allows to have more than one problem (several characters). Thus, characters' objectives interfere or collaborate. Also, treating characters as intelligent agents which have to solve their problems, the notion of perception and reasoning appears: software characters are no more a tool for the story, but a model of human characters. In TALESPIN characters can compete, dominate, trust other characters (among other relations). These relations and other character's properties like vanity or intelligence serve as motivation for their behaviour.

Regarding the analysis of what constitutes a story, Meehan proposes that the existence of a problem, the degree of difficulty in solving it and the level of such a problem make a valid story. However, this discussion seems to be applied externally to the program.

Story 3 shows an example of TALESPIN.

John Bear is somewhat hungry. John Bear wants to get some berries. John Bear wants to get near the blueberries. John Bear walks from a cave entrance to the bush by going through a pass through a valley through a meadow. John Bear takes the blueberries. John Bear eats the blueberries. The blueberries are gone. John Bear is not very hungry.

Story 3: Story generated by TaleSpin.

2.3.4. Author

Dehn's AUTHOR tries to simulate the author's mind when creating a story [Dehn, 1981]. Dehn states that the environment in which the story happens is developed after the events that happen are decided, in order to justify and frame the author's goals.

Dehn considers that authors have narrative goals at two levels: the particular narrative goals (what is to be told), and a number of meta-level goals like the plausibility or the need for coherence. This meta-level goals may translate into sub-goals, into which the characters may be led. A story is defined as "the achievement of a complex web of author goals". These goals enforce the story structure and guide the construction, but are not explicit in a story. Reformulation of the author's goals, or conceptual reformulation happens in the mind of the author when composing the story: from the initial idea to three acts, then a chain of smaller episodes, a dialogue, and so on.

AUTHOR organizes knowledge by considering the set of events in the story and also the author's experience. It tries to model knowledge in the system by following theories of how the human mind works based both on Psychology and Narrative theory [Schank, 1982, Kolodner, 1980].

Dehn considers story generation to be a creative process, and it must capture two characteristics: the degree of deliberation and the degree of serendipity. Two different meta-goals are postulated to model this: achieving the current narrative goal, and finding better narrative goals to find. The second meta-goal guarantees the duality between direction and serendipity, and it allows for changes when new opportunities appear.

2.3.5. Universe

Lebowitz's UNIVERSE generates short scripts for TV shows [Lebowitz, 1983, Lebowitz, 1985]. In UNIVERSE, several complex characters play out many simultaneous, possibly overlapping stories that never end. This system addresses the creation of characters by modelling them with a complex data structure whose fields are filled in partly automatically, although human input is necessary.

Many storytelling systems are aimed at creating a complete story, from the beginning to the end, but this is not the approach in UNIVERSE. This system generates plots that never end, and it is intended as a help tool for human authors, although it was intended that UNIVERSE could generate full stories autonomously.

Whereas Dehn follows the idea that the plot must be first built in order

to get a good story, and then let the world emerge from the sequence of events, Lebowitz thinks that good stories are achieved by creating a world, and then let the story develop from it.

The implementation of UNIVERSE uses plan-like unit to generate plot outlines. Some characteristics usually present in stories (like dialogues and text generation) are postponed. Plot fragments provide methods for the characters to achieve the author's goals. This intends to lead the generation to conflicts and interesting scenes which would not be reached with plain agent-like planning for characters. The system operates by choosing a goal whose preconditions are fulfilled and performing the appropriate actions. As the planning algorithm is not a depth-first search, the expanded goals can be unrelated between them. Coherence is ensured by maintaining a graph with relations between goals and previous told events in the story.

Regarding creativity, UNIVERSE is able to create new plot fragments by generalizing existing ones and, with the resulting structure, create a new instantiation of the structure. This process ensures correctness without information loss by a causal analysis of the initial fragment, and to ensure a logical sequence, an overall story plot, only a subset of attributes in the generalized structure are subject to change. This makes it possible to identify some fundamental author goals, but the given justification just explains that this is validated by successful experience in the generation.

2.3.6. MINSTREL

Turner's MINSTREL generates short stories about King Arthur and his Knights of the Round Table [Turner, 1992]. MINSTREL is the first story-telling program that explicitly addresses the creativity as an objective and its implications in the generation of stories. MINSTREL could tell about 10 stories half to one page in length, and it could create a number of shorter story scenes.

Goals and plans are used by MINSTREL to perform story generation. It can handle author goals and character goals. On its two levels of operation, MINSTREL switches between planning stage, in which author goals are consumed by either breaking them into smaller goals or passing them to the problem-solving stage, which attempts to solve these problems by adding the required information in the story. Each time a new scene is created, MINSTREL revises the memory of author goals to check whether it is possible to achieve one of them. If this is the case, the goal is processed. This operation resembles Dehn's meta-goal of trying to fulfil author goals.

One of the most interesting MINSTREL's features is the ability of handling episodic memories, which it does by using the so called TRAMs, *Transform Recall Adapt Methods*. Basic TRAMs just instantiate any matching schema in the story from a basic query, but more sophisticated ones perform a basic adaptation on the query, query the episodic memory with it and returning an adaptation and modification of the query. TRAMs can be chained, thus creating a chain of adaptations, which is a kind of generalization. Turner claims this is the base for creativity in MINSTREL.

An example of the generative capabilities of MINSTREL is given in Story 4.

The Vengeful Princess
Once upon a time there was a Lady of the Court named Jennifer. Jennifer loved a knight named Grunfeld. Grunfled loved Jennifer.
Jennifer wanted revenge on a lady of the court named Darlene because she had the berries which she picked in the woods and Jennifer wanted to have the berries. Jennifer wanted to scare Darlene. Jennifer wanted a dragon to move towards Darlene so that Darlene believed it would eat her. Jennifer wanted to appear to be a dragon so that a dragon would move towards Darlene. Jennifer drank a magic potion. Jennifer transformed into a dragon. A dragon moved towards Darlene. A dragon was near Darlene.
Grunfeld wanted to impress the king. Grunfled wanted to move towards the woods so that he could fight a dragon. Grunfeld moved towards the woods. Grunfeld was near the woods. Grunfeld fought a dragon. The dragon died. The dragon was Jennifer. Jennifer wanted to live. Jennifer tried to drink a magic potion but failed. Grunfled was filled with grief.
Jennifer was buried in the woods. Grunfeld became a hermit.
MORAL: Deception is a weapon difficult to aim.

Story 4: Story generated by MINSTREL.

2.3.7. Brutus

Bringsjord and Ferrucci's BRUTUS performs an automatic generation process of short stories about betrayal [Bringsjord and Ferrucci, 1999]. Aspects about creativity in the generative process in Brutus are discussed in the literature, and BRUTUS' authors claim that it is a valid approach to

Computational Creativity. However, the amount of different stories it can generate is quite restricted.

A three level operation governs the behaviour. First, a frame containing the base of a story is instantiated. Then, a character simulation process is run in order to obtain the main events that happen in the story. Finally, a grammar based solution generates the final output.

BRUTUS is a knowledge intensive program which stores the information in several formats depending on the purpose. Story themes and Story frames represent constraints that put limits on what can be generated, resembling Sharples' constraints [Sharples, 1996], in the meaning and the events in the story and in the pure rhetoric. Character behaviour in the simulation is governed by rules that allow them to think and plan, or they just behave in a reactive way. Finally, story grammars at several levels guide the final generation.

Story 5 shows an example of BRUTUS.

2.3.8. MEXICA

Pérez y Pérez's MEXICA generates short stories about the early inhabitants of Mexico City (the mexicas) [Pérez y Pérez, 1999]. MEXICA performs the generation by following Sharples' model of creative writing. MEXICA also puts a heavy focus on emotions, which are considered in this system as a main component of human creativity.

Sharples' account for writing is the base for MEXICA's narrative model [Sharples, 1996, Sharples, 1999]. This system is conceived as a very flexible tool that allows the user to parametrize several aspects of storytelling to experiment MEXICA's generative capabilities.

To represent stories, MEXICA uses story actions, or actions that can be present in the story and whose meaning is defined in terms of a set of conditions and preconditions. Previous stories are also available for the process, and these stories are defined in terms of story actions.

MEXICA is able to build its own set of knowledge schemas from previous stories, and these schemas are represented using a particular knowledge structure called *Story World Contexts* (SWC), which are instances of context with emotional links and actions. They act somewhat like rules during the engagement phase, associating a given action with the plot so far if it appears in a Story World Context that matches the plot so far. Knowledge in Story World Contexts is used to find the next action in the plot.

Following Sharples' model, in engagement, MEXICA ignores preconditions for allowing a free, unconstrained generative process. In the reflection

Dave Striver loved the university. He loved its ivy-covered clock-towers, its ancient and sturdy brick, and its sun-splashed verdant greens and eager youth. He also loved the fact that the university is free of the stark unforgiving trials of the business world –only this isn't a fact: academia has its own tests, and some are as merciless as any in the marketplace. A prime example is the dissertation defense: to earn the PhD, to become a doctor, one must pass an oral examination on one's dissertation.

Dave wanted desperately to be a doctor. But he needed the signatures of three people on the first page of his dissertation, the priceless inscription which, together, would certify that he had passed his defense. One the signatures had to come from Professor Hart. Well before the defense, Striver gave Hart a penultimate copy of his thesis. Hart read it and told Striver that it was absolutely first-rate, and that he would gladly sign it at the defense. They even shook hands in Hart's book-lined office. Dave noticed that Hart's eyes were bright and trustful, and his bearing paternal.

At the defense, Dave thought that he eloquently summarized Chapter 3 of his dissertation. There were two questions, one from Professor Rodman and one from Dr. Teer; Dave answered both, apparently to everyone's satisfaction. There were no further objections. Professor Rodman signed. He slid the tome to Teer; she too signed, and then slid it in front of Hart. Hart didn't move. "Ed?" Rodman said. Hart still sat motionless. Dave felt slightly dizzy. "Edward, are you going to sign?" Later, Hart sat alone in his office, in his big leather chair, underneath his framed PhD diploma.

Story 5: A short story generated by BRUTUS.

stage, MEXICA checks for correctness, novelty and interest. For each action whose preconditions are not met, the algorithm in the reflection stage searches and inserts actions that met these preconditions. Novelty and interest are kept by comparing the current partial story with the previous stories set. Novelty is computed by comparing the set of actions with the partial story, and interest is computed by examining the emotional tension in the sequence. In case MEXICA finds a poor rating for novelty and interest, the reflection stage sets guidelines for engagement, which modifies the set of possible actions that can be included in the story. This models the influence of previous stories in the generation. When the generation comes up with a situation in which the generation cannot continue (the so called impass), MEXICA seems to add an action chosen from similar contexts in previous stories.

MEXICA clearly follows most closely Sharples account from the point of view of its control flow. However, MINSTREL seems to operate on resources that match more closely the type of constraints that Sharples describes. Story 6 shows a short example of MEXICA's generation abilities.

> Jaguar knight was an inhabitant of the Great Tenochtitlan. Princess was an inhabitant of the Great Tenochtitlan. Jaguar knight was walking when Ehecatl (god of the wind) blew and an old tree collapsed injuring badly Jaguar knight. Princess went in search of some medical plants and cured Jaguar knight. As a result Jaguar knight was very grateful to Princess. Jaguar knight rewarded Princess with some cacauatl (cacao beans) and quetzalli (quetzal) feathers.

Story 6: A short story generated by MEXICA.

2.3.9. The Virtual Story Teller

TALESPIN's approach to storytelling based on simulation of characters has been recently followed by The Virtual Story Teller, which tries to create a scenario in which the characters perform and, as a result of that performance, a story emerges [Theune et al., 2003, Swartjes and Theune, 2006].

THE VIRTUAL STORY TELLER consists on a multi-agent system in which characters are represented by software agents and an extra agent directs the story, like a director. Agents have their own knowledge about the world and own rules that direct their behaviour. The system also includes a narrator agent, which tells the story in natural language.

The director agent has knowledge about classic stories and issues orders to other agents by adding new entities (objects or characters), motivating agents (giving specific actions to characters) and proscribing, disallowing certain actions. In any case, the director can not directly manipulate other characters, which must decide what to do taking into account the director's advices. THE VIRTUAL STORY TELLER also includes metrics for extra-structural content like "impressiveness", surprise and others.

2.3.10. FABULIST

Rield's work on story generation is based on a planning approach to building stories [Riedl, 2004]. In order to create meaningful stories, FABULIST adds character believability to coherent plots. With this definition he offers a modification of a previous partial order planner and creates the IPOCL algorithm (Intent-driven Partial Order Causal Link), which includes characters' intentions.

This research is heavily based on the concept of causality, which is the main relation used to create a *coherent* plot. It is assumed that narrations in which all events are correctly caused are understood as coherent by a human audience. The model is then improved by the addition of character believability, that is, the perception by the audience that the characters playing a role in the story perform because of a coherent set of intentions and beliefs. It is assumed in this work that character believability raises the quality of the stories.

Story 7 shows an example of generation carried out with FABULIST.

2.3.11. ProtoPropp

Peinado's thesis presents a knowledge intensive system to perform story generation [Peinado et al., 2008]. PROTOPROPP is able to generate short stories based on a detailed ontology in which narrative characteristics are described in terms of description logic predicates.

PROTOPROPP knowledge captures both common narrative knowledge and Propp's narratological functions for short tales. In this way, structuralist content is applied to story generation by establishing, trough a graphical user interface, some parameters by hand, like the role that characters must play in a story.

Peinado's work also put explicit focus on creating a set of formal metrics to evaluate the quality of stories, namely *linguistic quality*, *coherence*, *interest* and *originality*. These receive values in the domain of real numbers and

There is a woman named Jasmine. There is a king named Mamoud. This is a story about how King Mamoud becomes married to Jasmine. There is a magic genie. This is also a story about how the genie dies. There is a magic lamp. There is a dragon. The dragon has the magic lamp. The genie is confined within the magic lamp.
King Mamoud is not married. Jasmine is very beautiful. King Mamoud sees Jasmine and instantly falls in love with her. King Mamoud wants to marry Jasmine. There is a brave knight named Aladdin. Aladdin is loyal to the death to King Mamoud. King Mamoud orders Aladdin to get the magic lamp for him. Aladdin wants King Mamoud to have the magic lamp. Aladdin travels from the castle to the mountains. Aladdin slays the dragon. The dragon is dead. Aladdin takes the magic lamp from the dead body of the dragon. Aladdin travels from the mountains to the castle. Aladdin hands the magic lamp to King Mamoud. The genie is in the magic lamp. King Mamoud rubs the magic lamp and summons the genie out of it. The genie is not confined within the magic lamp. King Mamoud controls the genie with the magic lamp. King Mamoud uses the magic lamp to command the genie to make Jasmine love him. The genie wants Jasmine to be in love with King Mamoud. The genie casts a spell on Jasmine making her fall in love with King Mamoud. Jasmine is madly in love with King Mamoud. Jasmine wants to marry King Mamoud. The genie has a frightening appearance. The genie appears threatening to Aladdin. Aladdin wants the genie to die. Aladdin slays the genie. King Mamoud and Jasmine wed in an extravagant ceremony. The genie is dead. King Mamoud and Jasmine are married. The end.

Story 7: Example story generated with FABULIST.

a linear combination of them is output as the final quality. In this way, human judgements about the generated stories are acquired and used to validate his work. In particular, *interest* has influenced the seminal work for this thesis [León et al., 2007a, Hassan et al., 2007a, Hassan et al., 2007b]

Story 7 shows an example a story that PROTOPROPP generated.

> Once upon a time... the swan-geese fell in the trap of the king. The frog used a magic spell against the witch. The king scared somebody. Others and the knight heard about the witch. The swan-geese used a magic spell against the lioness. The king heard something. The swan-geese heard about the king. The little boy shared information with somebody. The little boy said to go outside. Not went outside. The lioness departed with the frog. The king fell in the trap. The lioness enchanted somebody. The lioness went outside.

Story 8: Example story generated with PROTOPROPP.

2.3.12. Summary

Several story generation systems have been presented in the past sections. As a summary, this section presents a table (Table 2.1) which compares some of the most identifiable properties of such systems, in order to create a centralized schema of the differences of the previously explained approaches.

2.4. Evaluation of Narrative

Evaluation and understanding of narrative has been addressed mainly by psychologists. For instance, Kelly analyses how personalities develop according to the structural constructions that humans create through the stories they read or hear [Kelly, 1955]. Applebee studied how the evaluative capabilities of narrative in children grow in parallel with their storytelling abilities [Applebee, 1978].

Literature in Natural Language Processing offers a more extensive set of examples and approaches to evaluation than Computational Narrative [Jones and Galliers, 1996]. In general, evaluation in Computational Narrative is mainly focused on purely narrative properties or in quality in general (as shown in some systems in Section 2.3).

System	Plot goal based	Character goal based	Author goal based	Creativity	Approach
NOVEL WRITER	no	yes	no	no	planning
RUMMELHART AND JOSEPH	yes	no	no	no	grammars
TALESPIN	no	yes	no	no	planning
AUTHOR	no	no	yes	yes	knowledge intensive
UNIVERSE	yes	yes	no	no	planning
MINSTREL	yes	no	no	yes	previous cases
BRUTUS	yes	no	yes	yes	knowledge intensive
MEXICA	yes	no	no	yes	engagement & reflection
THE VIRTUAL STORY TELLER	no	yes	yes	no	planning
FABULIST	no	yes	yes	yes	planning
PROTOPROPP	yes	no	no	no	knowledge intensive

Table 2.1: Comparison of story generation systems.

Regarding explicit computational evaluation of narrative, there is interest on some kind of common metric to compare the quality of story generation systems. Rowe et al. propose StoryEval, a framework for measuring and comparing story generation systems. StoryEval suggests to evaluate these systems by taking into account *narrative metrics*, *cognitive-affective studies*, *director-centric studies*, and *extrinsic narrative evaluations* [Rowe et al., 2009].

2.5. Acquisition of Narrative Schemas

In general, acquisition of structures, information or schemas regarding computational approaches is addressed by a set of techniques commonly grouped as the field of *Machine Learning*. Machine Learning is the discipline devoted to discover and program algorithms which make computers "learn" new knowledge from existing data, without the need for the programmer to directly input that data [Wu, 1992]. While several types of approach are studied in this discipline, none is specifically focused on narrative content. Therefore, this section will remain in the field of acquisition of information, structures or any other form of data in stories.

Overcoming the knowledge acquisition bottleneck in narrative content has been previously addressed from different points of view. For instance, Riedl focus the approach on managing small units of knowledge called *vignettes*. The possible uses of these vignettes are based on acquisition and adaptation of them [Riedl and León, 2008, Riedl and Sugandh, 2008, Riedl and León, 2009]. Vignettes capture in the form of short narrative scripts short episodes, for instance, a fight between a knight and a king or the kidnap of a princess. Through adaptation of vignettes, the domain can be modified while still retaining the action. For this approach to make sense, an initial collection of vignettes is required. Analogy making or Case Based Reasoning are tools being considered to adapt vignettes between different domains.

Finlayson proposes a computational system capable of automatically creating narrative morphologies from an existing corpus of stories, in particular, sort versions of Shakespeare's plays [Finlayson, 2009]. While this work is able to output rules for a new morphology without the need of human intervention for creating them, it has only been tested against a reduced set of inputs.

Chambers and Jurafsky present a statistical approach to learning of narrative schemas from a corpus of news [Chambers and Jurafsky, 2008, Chambers and Jurafsky, 2009]. Their approach is based on the analysis

of the statistical co-occurrence of actions or events in a big corpus, and from this analysis, the quality of the learnt relations is tested in the so-called *narrative cloze*, which tests the quality of the acquired relations by generating full narrations from incomplete parts. A related system, based on stochastic procedures and covering a full generative pipeline, is presented by McIntyre and Lapata [McIntyre and Lapata, 2009].

A similar approach, based, like the one that will be later presented in this dissertation, on a generate-and-test execution pattern was developed by Kaelbling [Leslie Kaelbling, 1994]. In the domain of Boolean formulae, his work shows an efficient solution for learning based on a simple algorithmic approach. While the domain is quite different from the one studied here, the underlying ideas are similar.

2.6. Chapter Summary

This chapter has examined the previous knowledge and approaches (narratological and computational) forming the base for the proposed research. Those systems whose characteristics are related to the current thesis have been highlighted, and alike approaches have also been studied.

Chapter 3

Definition of Narrations

NARRATIONS CAN BE DEFINED IN MANY WAYS, but for a computational system to be realizable, a formal definition must be created. In this chapter the definition of what is considered a formal story for this work and the parts constituting it is shown.

There exist several approaches to formal representation of narrations already. Previous research studied in Chapter 2 showed that every Computational Narrative system uses to create its own representation. This is due to the inherent differences existing between these systems: since each one is focused on a particular aspect of computational narrative, they adapt their representation to their needs. This makes it difficult to just take a previously created model because they are not general.

However, although there exist clear differences, it is possible to identify patterns underlying all representations. The next enumeration, while not containing an exhaustive list, makes explicit some of these characteristics:

- The concept of *sequence* is always considered. That is, narrations are always represented as an ordered list or equivalent.
- The focus of semantic content is set on the *verb* or *action*. Narrations are mainly structured by their actions. When creating knowledge bases, most previous story generation systems define the meaning (in whatever terms) mainly based on the verbs, and only secondarily on the characters carrying out the action.
- Some way to uniquely represent *entities* (characters, places...) is used. This is often carried out by using *tokens* or *logic atoms*.
- First-order logic concepts are commonly used. This does not happen only in Computational Narrative, but in knowledge representation in

general. In particular, the concept of *variable* is often present in most story generation systems.

Since no general definition of how a narration must be formalized for using it in computer systems has been explicitly proposed yet, and although many options exist, it has been considered that the best approach is to create an ad-hoc definition, while still taking into account previous work.

Taking into account previous representation of stories, then, a formal definition of narration has been created and iteratively refined to match the requirements of the proposed approach to computational narrative. During this creation, several formalizations were considered [León et al., 2007b, León and Gervás, 2008, León et al., 2008]. It will be seen that the previously identified characteristics of common formal representation of narrations are present in this proposed formalism. This has not been set as a constraint, instead, those characteristics have arisen during the development of the model as a natural way of representing stories for computational processing.

Therefore, explicitly imitating previous approaches to represent narrations is not intended. No conceptual model has been followed to create this particular formal description of narrations. The computational algorithm for processing stories based on structural properties is the focus of this research, thus the representation fully adapts to the computational requirements, and not the other way around.

3.1. Definition of Narration Regarding Conceptual Models for this Research

No claim about the psychological plausibility of the formal definition of narrations is made. The proposed formalism has only been created for the model to be implementable, and whether these formal structures used to define stories match human mental processes or not is not addressed.

That is, although there exists several models of what a narration is from a human perspective, none has been explicitly followed. This has been done on purpose to permit full freedom in the development of the model of narrations. The specific model of what a narration is from a formal point of view is not the objective in this research. Therefore, it was decided that applying a conceptual model would involve the additional work of proving that the computational definition matched the conceptual definition, and it

was considered useless from the current perspective and for this particular investigation.

This research defines *narration* or *story* as a set of events ordered in chronological time, therefore identifying, to some extent, the terms *fabula* and *narration*, according to the definition of the term *fabula* coined by Russian formalists [Propp, 1928]. It is important to note that this is an *ad-hoc* definition used as nomenclature for this dissertation: no similarity with any psychological, formal or narratological concept is intended nor any definition in these terms is made. From now on, any reference to *narration* or *story* refers to the same concept, except where otherwise stated.

This research is focused on the study of formal story outlines, and not on textual narrative or complex literary characteristics. This focus on outlines has been also chosen in most previous work on story generation [Meehan, 1976, Bringsjord and Ferrucci, 1999, Riedl and Young, 2006].

Therefore, no study about text realization or artistic properties of narrative has been carried out. Although simple text realization has been created in the implementation, it has only been programmed for presenting the story to human evaluators and it is accepted that no real quality has been reached. The intended contribution, then, ranges in the domain of symbolic representation of stories and research on Natural Language Processing, in any of its forms, is not addressed.

3.2. Formal Definition of Narration Used in this Research

According to the lack of imposed conceptual constraints shown in Section 3.1 and based on the common characteristics that have been identified in previous research on computational narrative, a formal definition of narrations has been created for this research. This has been done in order to allow for computations over formal versions of narrations in the implementation and to define the formal model for evaluation and learning.

The formal representation remains at discourse level, and does not assume that the used parts constitute a representation of a domain. It allows for representation of world knowledge (as it is shown in Chapter 4), but the semantics that the cognitive level assigns are not considered in this formalism.

The formal model is presented in ascending order regarding its constituent parts. First, the most basic elements, namely the *tokens*, are shown. Then the *actions*, and then the *narrations* themselves. These

names have been assigned to these parts in order to refer to them, and no similarity with the concepts they represent when using these words in English is claimed. Although the selection of the names is clearly influenced by similar concepts in Narratology, only the formal meaning that is assigned in this research must be interpreted when referring to them.

No claim is made about the further applicability of this definition in other systems. It has been created for this project and, although it could serve to base other research projects, this has not been explicitly intended.

3.3. Constituents of Formal Narrations

Next subsections detail the constituent parts of the formalization of narrations, which is described in Section 3.4.

3.3.1. Tokens

Tokens are atomic elements defining a single thing, character, idea or whatever single instance of any concept. They can represent any entity in some particular domain, but they are restricted so that they cannot refer to any part of the narration. Therefore, tokens cannot represent *actions* nor *narrations* (these are defined in following sections).

Tokens are uniquely specified by their name. This means that two tokens with the same name are actually representing the same concept. Examples of tokens are *john*, *bird*, *house*, *hope* or *sad*. In this sense, *tokens* are similar to logic atoms in first order logic.

Two approaches to story generation are shown in this dissertation: the first one (Chapter 4) uses the semantics that are linked to the concepts that *tokens* define. The second one (Chapters 5 and 6) does not take this into account. The way in which narrations are represented allows for both uses of the information because it does not assume any semantic, therefore allowing for its later inclusion.

3.3.2. Variables

It will be shown that the evaluation and the algorithm for extracting structures for creating new stories both need *variables* to represent rules. Variables in this proposed model are similar to logic variables, and they can represent any token. Variables are denoted by a symbol followed by a question mark. For instance, *x*? or *token_variable*? are valid representations of variables.

Once a variable is bound to a token in an execution of the algorithm (later described), every instance of that variable is bound to that token. There is only one space for variables in the model, so *every* action containing the variable *x*? will have it replaced once it is bound.

Thus, the context, the generation or whatever other mechanism must control the subset of tokens than can be used to bound every particular variable. This formal model for computational definitions of narrations does not address this because it has been created to be general enough to cover all the cases studied in this dissertation. No explicit use for variables is assumed so far, although the computational algorithms shown later will formally define how variables are used.

3.3.3. Actions

Actions are basic constituents of narrations representing an event, property or relation. Actions are defined by this event, property or relation (from now on, the *kernel*) plus an ordered sequence of tokens. The action, then, links a kernel with the tokens, meaning that the action, property or relation is true for them.

A kernel is represented in the form of a single word, which can be compound. For example *love* or *take_to* are kernels. Although this could bring confusion because kernels are formed in the same way as atoms, in practice this does not happen because kernels, in the rest of this model, are always forming part of an action (as defined below). The structure of actions formally specifies the kernel and the tokens.

Although they could be represented in many ways, this model represents actions as first-order logic-like predicates. The kernel in an action takes the place of the property, and the elements take the place of the atoms. Examples of actions are *take*(*john, glass*) or *love*(*ofelia, x*?), where *john*, *glass*, *ofelia* are tokens and *x*? is a variable. Although no explicit semantics are needed for the model so far, actions were defined in this way to mean "John takes the glass" and "Ofelia loves *something*", respectively. This is because the definition of the formal definition for narrations has been iteratively refined and adapted, therefore carrying information from previous versions in which semantic meaning was tightly linked to actions.

The order in the sequence of tokens matters because the place they have in the action determines their role in the action. For instance, the first place is usually the subject that carried out the action. The particular role that each position plays for each action is not fixed, so every action must define this. This design decision has been taken to ease the definition of new terms.

Since this model of narrations does not take into account semantics, it is not affected by the syntactical role of the tokens in the actions. The application of the model is responsible to create a robust layout of positions. The only restriction that the model presented in this chapter imposes is that two actions are *equal* when the kernels are equal, the set of tokens are equal, and the order in which the tokens are positioned in the action is the same. This is why order matters in the model.

The examples show that the kernel of the action is tightly linked to the idea of *verb*. This is not a coincidence. Like several other approaches to knowledge representation in general [Schank and Abelson, 1977] and computational narrative in particular, verbs are assigned a very important role in the definition of narrations because they convey a very important amount of information. While the model for narratives does not address semantic properties, its development was partially linked to a semantic processing of narrations. Therefore, its development generated this definition. Since late development showed that the chosen approach was empirically appropriate (as evidenced in Chapter 7), it was decided to keep this formal representation.

According to the parameters linked to the kernel in the form of tokens or variables, two types of actions are defined:

- *Ground actions* are actions whose elements are tokens (*take*(*john*, *glass*)).
- *Pattern actions* are actions in which at least one element is a variable (*love*(*ofelia*, $x?$) or *attack*($x?$, $y?$)).

The use of these structures in the proposed models for computational processing of stories shown in Chapters 4, 5 and 6 justifies the creation of these definitions.

3.4. Narrations

Narrations are ordered sequences of actions. They nave been defined from the previously described parts (tokens, kernels and actions). Equation 3.1 shows a formal representation of a generic narration.

$$n = [a_1, a_2, a_3, \cdots, a_m] \tag{3.1}$$

where $[a_1, a_2, a_3, \cdots, a_m]$ is a list of ground actions and n is a narration.

For instance, [*go*(*john*, *cinema*), *buy*(*john*, *popcorn*), *watch*(*john*, *movie*), *eat*(*john*, *popcorn*)] is a narration.

Order is fundamental in narrations for this model. Although modern forms of narration do not necessarily require sequential order, like hypertextual narrative [Joyce et al., 1989, Mancini, 2000, Bernstein, 2009], only simple, linear narrations are addressed, following the restrictions shown in Section 1.2. That is, a narration, as this model defines it, must be a single chronologically ordered thread.

More concretely, *simple stories* are sequences of actions in which an additional set of constraints is assumed. That is, the computational model for processing simple narrations in this work must assume the next set of constraints:

- Time is divided in discrete units ($t_1, \cdots, t_n$).
- Time units are totally ordered.
- Every action starts at a single unit of time s and ends at a posterior unit of time e ($e > s$).
- If an action a_i is in the position i and a_j is another action in the same story in position the j, a_i started before a_j is $i < j$ and vice-versa.
- Intervals cannot overlap: an action a always starts after all the preceding actions have ended.
- A *single* narrative thread is assumed. No parallel stories can be expressed in a single *simple story*, therefore any algorithm processing this kind of narrations can safely assume that all actions are semantically related as a single plot.

The existing relation between knowledge representation in computers and time has been previously addressed in literature [Vilain et al., 1986, Ladkin, 1987, Allen, 1991]. While extensive research has developed algorithms for treating time as a logic entity and this could have been used in this model to define a richer formal description of narrations, it has been considered to be outside scope. In order to keep the research focused, time has been simplistically addressed by applying the restrictive constraints previously explained. Nothing prevents, though, from improving the system as part of the future work (Chapter 9).

As an example, the formal Narration 1 shows a story without pattern actions. A sequence in which some element was not an action, for instance, would not be a valid story in the formal terms defined here. Chapter 7 shows other examples that have been used for experimentation.

s =[*carry*(*dog*, *meat*),
cross(*dog*, *river*),
see(*dog*, *the_shadow*),
consider(*dog*, *the_shadow*, *shadow*),
attack(*dog*, *the_shadow*),
drop(*dog*, *meat*)]

Formal Story 1: Example of a formal narration.

3.5. Space of Stories

To define the set of stories, some working domain must be chosen. Since stories represented according to the formalization shown in this chapter will be the members of this set, the valid tokens and patterns have to be selected. Also, the set of particular elements that will belong to the set of stories for some domain depends on the maximum length allowed for stories. It is assumed that the set of variables is unrestricted, that is, there will be as many variables as needed to define all pattern actions.

Therefore, the set of stories $\mathscr{S}$ depends on:

- δ, the set of tokens.
- μ, the maximum number of actions per story.
- π, the set of candidate pattern actions.

3.5.1. Size of the Space of Stories

According to these three parameters (δ, μ and π), the size of the set of stories can be computed using Equation 3.2. Here, τ is the number of possible ground actions. The value for τ is computed from δ and π.

The equation represents that the size of the set is the result of accumulating the amount of stories from length 1 to length μ. The size of the set of stories of length i is equivalent to the number of ground actions (computed by the τ function) powered to the length of the story.

$$|\mathscr{S}_{\delta,\pi,\mu}| = \sum_{i=1}^{\mu} \tau(\delta, \pi)^i \tag{3.2}$$

where $\mathscr{S}$, δ, π, μ are the variables defined in Section 3.5.

The τ function is the number of ground actions in a domain defined by δ and π. Since, a priori, there is no semantic restriction in the way in which ground actions can be constructed, a pattern action can be instantiated (turned to a ground action) with any token.

The τ function is therefore defined as shown in Equation 3.3.

$$\tau(\delta, \pi) = \sum_{p=\pi_0}^{\pi_n} |\delta|^{arity(p)} \tag{3.3}$$

where $arity(p)$ is the number of tokens or variables that a pattern can have, and π_0 and π_n are the first and the last element from the π set. Therefore, the sum traverses the whole π set.

That is, τ returns the number of all possible instantiations ($|\delta|^{arity(p)}$) for all pattern actions in the domain.

The $arity(p)$ function returns the size of the pattern regarding the number of tokens and variables that it allows. For instance, making the call $arity(love(x?, y?))$ returns 2, making the call $arity(give(x?, y?, z?))$ returns 3 and $arity(die(x?))$ returns 1.

3.5.2. Subspaces of *Good* and *Bad* Narrations

The set $\mathscr{S}$ is partitioned into two disjoint subsets, $\mathscr{G}$ and $\mathscr{B}$. The set $\mathscr{G}$ corresponds to the "good" stories, that is, those stories whose quality is *acceptable*. Complementary, the set $\mathscr{B}$ is the set of "bad" stories, or the stories whose quality is not high enough. Any story must fall in either set. The objective of any story generator which intends to generate good stories is, therefore, to be able to generate *only* the *whole* $\mathscr{G}$ set. This task requires a usable definition of *quality*, then. Chapters 4 and 5 show the definitions that have been created for this dissertation.

It could be claimed that the partition of the whole space of stories is more complex than a simple partition in "good" and "bad" stories. The particularities of how to define quality for stories are partially addressed in Chapters 4 and 5 from a computational perspective. *By definition* in this research, a story can be "good" or "bad", and there is no other set. In a real scenario taking into account human criteria this is arguably more complex. Section 8.1.5 addresses this issue in detail.

While this is arguably an over-simplification of the problem, it has been applied in order to keep the models easily implementable and clearer (see Chapters 5 and 7). The author is well aware that human behaviour is much more complex. However, this simplification is considered to be a

good option for the presented prototype, since only coherence (and not levels of coherence) is addressed in the current research.

3.6. Chapter Summary

This chapter has introduced a formal definition of simple narrations. This definition will be used from now on as the formal base for creating the Computational Narrative system devoted to story generation. The space of possible stories that is created by this definition is presented and studied, and the cognitive and psychological aspects of the proposed formalism are discussed.

Chapter 4

Story Generation Based on Semantic Knowledge

The first approach to improve the amount of generated stories while reducing the cost of adding new domain dependent knowledge was the creation of a story generation system based on an evaluation function that, given any story inside some predefined domain, could rate the quality of the story and thus differentiate "good" and "bad" stories.

The main underlying hypothesis at this stage of the research was that the definition of an explicit evaluation function which takes into account narrative and psychological aspects of the narrative act could help to better define a generative process in a more general form, thus creating a model that could be used in general for story generation. In particular, such a function could drive a classic state space search to find "good" stories according to that function.

This system was modelled and implemented as shown in this chapter. Many of the ideas and concepts introduced here set the base for a posterior development of a system that tries to avoid the main drawbacks that were found during the creation of the version of the story generator detailed now.

The formal definition of narrations shown in Chapter 3 was created as part of the work developed in this stage of the research. It has been described in a separate chapter because the concepts that were introduced there are valid both for this chapter, detailing the first version, and for the rest of the dissertation that develops the final proposed model.

This chapter, as it will be detailed, proposes a system based on semantic approaches to perform story generation. While the results were promising, it was evidenced that improving the generation capabilities was difficult due to the system's strong dependence on domain rules. A more detailed

definition of the content of this chapter in the context of Computational Creativity can be found in [León and Gervás, 2010].

To offer a solution for this issue, the research led to the creation of a system that pseudo-automatically extracts structural information from narrations, explained in Chapter 5. Since the research based on semantic processing was the origin of that system, it is explained here to document the carried out research.

4.1. Towards a Story Generation System based on Evaluation

In order to perform story generation *in the large*, it was hypothesized that generating a big set of stories containing both good and bad ones and then choosing those that could be automatically rated as good could improve the amount of stories that the system could generate.

The set of all possible stories $\mathscr{U}$ contains all possible artifacts that can be considered a story by a human reader. It is possible to create a new story in this set by adding a new action to a previously created story, so the $\mathscr{U}$ set is infinite and therefore it is not possible to fully explore it in finite time. Therefore, we want to restrict the exploration in our model to a particular subset of stories, so it is necessary to define the $\mathscr{C}$ set inside $\mathscr{U}$. This space is restricted by a rule-set $\mathscr{R}$, in such a way that it only contains the computationally valid stories in which we are interested. This $\mathscr{R}$ rule-set applies the definition of narrations explained in Chapter 3. Additionally, the $\mathscr{C}$ is constrained with domain rules. The set with domain restrictions, $\mathscr{C}_d$, contains both high-rated, low-rated and meaningless stories according to human criteria.

4.2. Evaluation Function

A function capable of selecting the high-valued stories from among the rest, at least to some extent, is required to identify those elements belonging to the $\mathscr{C}$ set that are certainly "good". Equation 4.1 describes the *story evaluation function*, E, which ranges over the domain of stories in the $\mathscr{C}_d$ set and returns a real value in the range $[-1,+1]$, -1 representing extremely poor quality and $+1$ representing a very good story.

$$E : \mathscr{C}_d \longrightarrow \mathbb{R}_{[-1,+1]} \tag{4.1}$$

The function iterates over the sequence of messages in a story in order, processing their constituent actions. The value for this function is computed from values assigned to a set of significant variables. These values are of three types, corresponding to three different aspects of a story that need to be taken into account: *accumulation of contributions*, *appearance of patterns* and *inference*, as explained next.

A number of variables that take values based on accumulation of contributions from individual action depending on the meaning of the action. It is very important to make clear that this set of variables is by no means a model of human understanding of narrations, and they do not intentionally represent or match any psychological model. They were just chosen as a test set in order to define a computational function, and the selection was only driven by the author's intuition. Therefore, no claim is made about the psychological plausibility of the next list:

- *Interest* models the intention of the reader to continue reading the story.
- *Danger* represents how much danger the reader perceives in the story. When a character is about to die, the *danger* variable is raised.
- *Love* measures the amount of love in the story that the reader perceives. All kinds of love are covered by this variable: romantic love, friendship, and so on. For instance, when a character kisses another character the value of the *love* variable is increased.
- *Tension* captures the sense that an important event is to come in the story.
- *Humanity* is raised when human behaviour is clearly present in the story and characters' reactions are human-alike.
- *Action* represents the amount of change and movement that the story contains. Events involving some kind of action (moving, talking...) raise the *action* variable, whereas descriptive actions (position, feelings) do not.
- *Empathy* models the development of empathy (positive or negative) towards characters. For instance, if some character kills another character, the empathy towards the murderer is lowered.
- *Emotion* represents the perception of emotive actions a heroic fact, fear and other events related with human emotions.

Some variables measure the appearance of particular patterns or relationships between the actions of a story:

- *Causality* measures the number of causality links. If the cause for an action is found, the *causality* is raised.
- *Funny* measures how funny the story is, based on the occurrence of specific templates.
- *Chronology* measures, according to some basic rules, the correctness of the time order of facts in the story.

To model the way humans react to stories, the evaluation function must model the ability people have for "interpreting" stories by adding hypotheses, causes and explanations for what they are told even if those are not explicitly present in the story (some recent work studies these aspects of storytelling [Niehaus and Young, 2009]). To capture the effect of these operations on the overall rating, some variables operate over the number of facts that have been inferred or hypothesized during interpretation (which is computed by ad-hoc, domain dependent rules):

- *Compression* is defined as the ratio between the number of actions that the reader infers and the number of actions that story explicitly includes.
- *Hypotheses* measures the amount of knowledge that the reader hypothesizes when she reads the story. The *hypotheses* variable measures how many hypotheses have been made.

The final rating for the E is a linear combination of these variables. Although several ways of combining these values are possible, the unweighted mean value is used as the overall rating. While being a rather simple approach, the empirical tests show good results using this approach. Other combinations could yield different values that are better fitted to human evaluation following the comparison in Section 4.3.

This set of variables is by no means exhaustive. One can think of several other plausibly valid variables, like surprise. The objective is not to create a full model, but to study the use of evaluation in story generation.

4.2.1. Implementation of the Evaluation Function

The evaluation function has been implemented as a rule based system. Domain specific rules have been encoded in an independent module in such a way that the general evaluation engine can be kept general enough to allow changes of domain at a later stage.

The *E* function is a knowledge intensive rule-based function that receives stories and iteratively processes them to compute a rating value. Thus, the evaluation function sequentially processes every action, just as a reader would do with written text. Although several psychological models and identification of variables regarding its influence on perception of narrative are available [Graesser et al., 1994, Weyhrauch, 1997, Mateas and Stern, 2005], none has been used. Only ad-hoc rules have been created for this prototype, so no psychological plausibility is claimed.

The evaluation function relies on a *context* Γ which stores the partial information state accumulated by a hypothetical reader as the processing evolves. This state includes a partial assignment for evaluation variables.

A rule has preconditions (that must be satisfied by the current context for the rule to be applicable) and postconditions that define the changes that should be applied to the current context to obtain the context after processing the action under consideration.

The *process* function searches in the rule base for rules whose preconditions match Γ_i and e_i. The effects of these rules create a new context which is returned by the *process* function, in this way updating the state of the evaluation. Equation 4.2 shows this relation:

$$\Gamma_{i+1} = process(\Gamma_i, e_i) \tag{4.2}$$

where Γ_{i+1} and Γ_i are the next and current contexts respectively, e_i , is the action being processed and *process* is the function that chooses which rules to apply and applies them.

For creating rules, the authors' intuition has been applied, with special focus on effectiveness of the rules for the working domain. The main objective for rules in the current prototype has been to demonstrate that such an evaluation function, at least for simple narrative, is possible.

The presented prototype has 73 rules. Each rule is only applicable to one type of action. In the rule set there are rules for actions with the *go* kernel, for the *take* kernel, and so on. This means that the kernel in the action is the base when creating rules in the current prototype. Some example rules (translated to natural language) are shown in Table 4.1.

Not every rule is applied for every story. Only those rules whose preconditions are satisfied by the context are used, and the order of application

Context	Event	Variable changes
x? has to pay money to y? and x? did not pay to y? and y? is in p?	x? goes to p?	raise *danger* and raise *humanity* and raise *action*
x? and y? are not friends	x? asks y? for help	raise *humanity* and raise *tension*
z? is the boss of x? and z? hates y?	x? kills y?	raise *humanity* and raise *danger*

Table 4.1: Example of evaluation rules.

is not important because the definition of rules only takes into account the story so far, not the partial results from other rules at previous stages stage.

The main flaw of this design is that the creation of the rules by hand is costly and the rule-set cannot be easily updated without an extra effort to keep consistence on the knowledge base. This is a typical problem of rule-based systems, and it affects storytelling systems like BRUTUS [Bringsjord and Ferrucci, 1999]. Next chapters in this dissertation address this issue.

4.3. Validation of the Evaluation Function using Human Judgment

It is necessary to validate the current model, at least to demonstrate that the task of modeling story evaluation is worth exploring. For this task, 10 stories generated by the exhaustive conceptual space exploration approach (as explained in Section 4.4) were issued to human evaluators (the set of evaluators is detailed later) and they were asked to order them by quality. Seven stories out of 10 were picked from those which the evaluation system rated as "good" (1, 3, 4, 5, 6, 8 and 9) and three from the set rated as "bad" (2, 7 and 10). The selection has not been totally random. Instead, the focus has been put on getting a sample of different stories with a broad range of values from the evaluation function.

Eleven evaluators are male and eight are female, all Spanish. Their ages ranges between 24 and 59 years old and none of them are native English speakers, although all of them consider to have a high level of reading com-

	1	2	3	4	5	6	7	8	9	10
Interest	0.6	-0.3	0.24	0.5	0.3	0.24	0.07	0.07	0.7	-0.2
Causality	0.6	0.3	0.7	0.9	0.5	0.2	-0.1	0.2	0.2	0
Compression	0.7	0.4	0.7	0.6	0.3	0.2	-0.1	0.3	0.4	-0.2
Danger	0.9	-0.2	0.7	0.6	0.5	0.6	0.8	0.6	0.9	0.8
Love	-0.3	-0.6	-0.1	0.5	0.1	-0.2	-0.3	-0.2	-0.1	-0.2
Tension	0.4	-0.2	0.3	0.4	0.4	0.4	0.2	0.02	0.3	0.2
Humanity	0.4	-0.3	0.2	0.64	0.5	0.4	-0.0	0.3	0.5	0.2
Action	0.4	-0.1	0.3	0.6	0.3	0.4	0.5	0.2	0.7	0.5
Hypotheses	0.3	0	-0.3	0.2	0.6	0.2	-0.2	0.2	0.3	0.5
Empathy	0.2	-0.6	0.1	0.3	0.2	-0.2	-0.2	-0.3	0.16	-0.1
Funny	-0.1	-0.4	-0.4	-0.3	-0.4	-0.4	-0.3	-0.4	-0.7	-0.5
Emotion	0.2	-0.6	-0.1	0.2	0.1	-0.2	-0.1	-0.1	0.24	-0.1
Chronology	0.8	0.5	0.6	0.8	0.6	0.4	0.3	0.7	0.9	0
Over. rating	0.6	-0.3	0.2	0.4	0.1	0.2	-0.2	0.2	0.6	0

Table 4.2: Mean values for human evaluation of the set of stories.

prehension in English. Fourteen have graduate or post-graduate academic studies. None have any specialization in narrative.

Human judgements have been compared to the ordering that the evaluation function puts on the stories. The evaluation function creates this *quality order* by assigning a value to each story and then ordering stories accordingly. Stories used for the validation are shown in Figure 4.1.

It is important to note that, since these stories have been randomly generated, some of them do not conform to the definition of *simple narrations*, as defined in Chapter 3. Humans, however, do not take this into account when evaluating the previously detailed variables. This leads to a certain amount of divergence between the intended model and the obtained results. However, this is considered to have only a small influence on the overall conclusions because the rules just ignore those patterns which do not match the definition of simple narrations. Thus, the value for the corresponding variables will be adjusted to what the rules can process.

Human evaluators are asked for a rating in the integer range $[1, 5]$ for every evaluation variable presented in Section 4.2. That interval has been chosen in order to use positive integer values instead of real numbers, which has been considered to be simpler for evaluators.

These human evaluation values have been normalized to match the range $[-1, +1]$. Additionally, the opinion about the overall value has been gathered, in the same range. Nineteen people have been queried. Table 4.2 shows the mean gathered values.

Table 4.3 shows the values computed by the implementation of the eval-

1. John was in the bus stop. A man was in the bus stop. John realized that it was late. He was surprised. John asked the time to the man, and he said that it was two o'clock. John supposed that he was going to die. Some time before, John had agreed with a godfather that he would pay him some money before 2 o'clock. John wanted to ask for help to the man in the bus stop. The man in the bus stop, then, said that it was too late, and he killed John.

2. John was in the bus stop. John went to the warehouse. John gave some money to a man. The godfather was the boss of the man. The man gave the money to the godfather. The godfather said to the man that John had to pay him that money. The guy said goodbye to John. John said goodbye to the man.

3. The godfather hated a man. The godfather was the boss of the man. The man was a friend of John. The godfather was the boss of John. John was the friend of the man. The godfather told John to kill the man. John killed the man. John was sad.

4. The godfather was sad. The man killed John some time before. The godfather desired John to be alive. The godfather told the man that he hated him. The man loved the godfather. The man was sad. The man killed himself.

5. The man desired that John was in the bus stop. The man was in the bus stop. The godfather told the man to kill John some time before. The man was afraid. The man wanted to escape. The man supposed that the godfather would get angry. The man escaped.

6. The man was angry. John was angry. The godfather told the man that he would pay him some money some time before. The godfather told John that he would pay him some money some time before. John supposed that the man would kill the godfather. John found the godfather. The godfather was dead. The man supposed that John had killed the godfather.

7. John took the gun. John was friend of the godfather. The godfather was friend of the man. The man was friend of John. The man had some money. The man gave some money to John. John gave some money to the godfather. The godfather gave some money to the man. John killed the man.

8. The godfather was surprised. The man had killed John before. The man told the godfather that it was late. The godfather told the man that it was 2 o'clock. The godfather took the money. The godfather gave the money to the man. The man was happy.

9. John was in the bus stop. The godfather was in the bus stop. The man was in the bus stop. John realized that the godfather took the gun. The man realized that the godfather took the gun. The godfather killed John. The godfather killed the man. The godfather killed himself.

10. John was angry. The godfather realized that the man was in the bus stop. The godfather told John that he supposed that it was late. The man took the gun. The man went to the warehouse. The man killed John. The man was surprised. The godfather told the man that he had killed John. The godfather was happy.

Figure 4.1: Stories used for validation. They have been generated using the algorithm explained in Section 4.4. They have been translated to natural language using simple templates and the text in some sentences has been corrected by hand.

	1	2	3	4	5	6	7	8	9	10
Interest	0.8	-0.2	0.2	0.6	0.2	0.2	-0.2	0.2	0.6	-0.6
Causality	1	-0.2	1	1	0.8	0.6	-0.2	0.2	0.4	-0.3
Compression	1	-0.6	1	1	0.8	0.7	-0.4	0.6	0.6	-0.5
Danger	1	-0.2	1	0.6	0.6	0.6	-0.4	0.6	1	0
Love	-0.2	-0.6	0.6	0.2	0.2	0	-0.2	-0.2	0.2	-0.5
Tension	0.6	-0.2	-0.2	0.6	0.4	0	-0.6	-0.2	0.9	-0.3
Humanity	0.9	-0.6	0.6	0.6	0.6	0.6	-0.2	0.2	0.4	-0.4
Action	0.2	-0.4	0.2	0.6	0.1	0.2	-0.2	0.2	0.6	-0.5
Hypotheses	1	0.6	-0.2	0.2	0.8	0.8	-0.2	0.6	0.2	-0.8
Empathy	0.8	-0.6	0.6	0.6	0.4	-0.2	-0.2	0.2	0.6	-0.7
Funny	0	-0.6	-0.4	-0.6	-0.5	-0.4	-0.6	-0.6	-0.2	-0.6
Emotion	0.2	-0.6	-0.1	0.2	0.1	0.1	-0.2	0.3	0.4	-0.7
Chronology	1	-0.2	1	1	1	1	-0.4	1	0.9	-0.7
Over. rating	0.6	-0.3	0.4	0.5	0.4	0.3	-0.3	0.2	0.5	-0.5

Table 4.3: Values computed by evaluation function for the set of stories.

uation function. The overall rating is the mean value of all the variables.

Figure 4.2 shows the comparison between the overall value computed by the implementation of the evaluation function for the test stories against the overall value gathered from humans. A very nice fitting can be seen between rating in human evaluation and the implementation. The mean quadratic error between the two sets of values is 6.21%. On the other hand, a perfect fitting would be difficult to interpret as strong validation of the evaluation function: there are so many aspects in story evaluation, and there are so many possible interpretations, that there is not a *correct solution*. Therefore, a "nice" fitting of the evaluation function output values is, in general, useful enough.

To show more specific results, the next list includes the Pearson' correlation coefficient between human and machine results for every variable in the executed experiment, which show a clear correlation:

- Interest: 0.91.
- Causality: 0.88.
- Compression: 0.75.
- Danger: 0.5.
- Love: 0.67.
- Tension: 0.59.

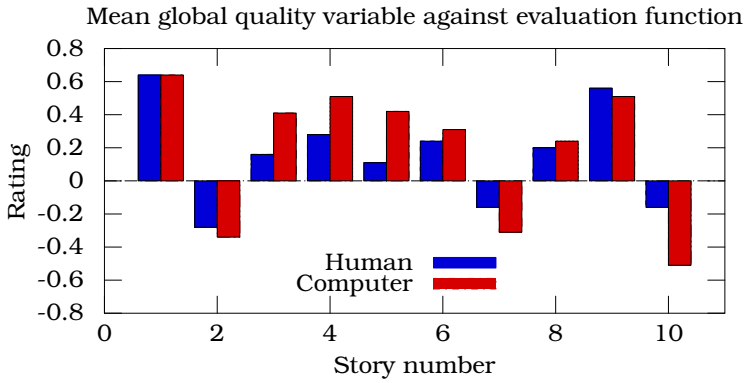


Figure 4.2: Mean global quality variable against evaluation function.

- Humanity: 0.79.
- Action: 0.52.
- Hypothesis: 0.24.
- Empathy: 0.78.
- Funny: 0.74.
- Emotion 0.72.
- Chronology: 0.8.
- Causality: 0.74.
- Overall quality: 0.9.

This result suggests that the implementation and the selection of the variables is promising for simple stories as those we present.

There is a big deviation for story 10. The automatic evaluation system rates that story as a very bad one, whereas human do not set such a low value. This is because story 10 is meaningless for the current implementation of the evaluation function and humans tend to create meaning in a more powerful way. This case shows that a more detailed study of how humans assign meaning to stories must be made in order to add coverage for more complex stories in the evaluation function.

4.4. Exhaustive Exploration of the Space of Stories for Generation

A simple *generate and test* approach, using PROLOG in the implementation, has been used to perform an exhaustive exploration of the $\mathscr{C}_d$ set of stories. The generation iteratively constructs the $\mathscr{C}_d$ set, and every created story is evaluated with the implementation of the E function.

The implementation consists on a simple generative approach based on an incremental strategy: a story with a single action is generated and evaluated, then a new story with that action and an additional one is generated, and so on. This is done for every combination of actions and tokens (characters, places, etcetera), and considering the maximum length of stories (defined in Equation 3.2).

The order in which actions are generated depends on the order in which action generation rules are ordered in the PROLOG program. Events are instances of basic actions (the set of possible actions in the current prototype is described in Equation 3.2). Tokens for are taken from the δ set. The generation of messages is carried out incrementally by building messages for all available actions.

Stories can be created based on the way actions are generated. First, all stories with one message are sequentially created. This means that the system will generate one story per different possible message. Then, the process is repeated for all possible stories of length 2, and so on until the number of messages reaches φ.

Two sets are created during generation: the set of *good* stories and the set of *discarded* stories. Stories are classified into one or the other based on the results of the evaluation function E described in Section 4.2.1. A user given threshold τ is used to include stories in one set or another. Those stories whose rating falls above τ are good, and the rest are considered bad and they are included in the set of discarded stories.

The implemented prototype has 3 terminals for different characters, 2 different locations, 2 objects and 26 different types of actions. Therefore, it can generate 473,979 different actions. This number is obviously dependent on the number and the type of parameters of the action. For a maximum depth of 10 actions per story, according to Equation 4.3, the conceptual space $\mathscr{C}_d$ has a size of:

$$|\mathscr{C}_d| = \sum_{i=1}^{\delta} \mu^i = \sum_{i=1}^{10} 473,979^i = 5.7226 \cdot 10^{56} \text{ different stories} \qquad (4.3)$$

4.4.1. Results of the Exhaustive Exploration Approach

A threshold τ of 0.01 has been used. The execution was run in a single core Intel Centrino 2.16 GHz computer with 3GB of RAM memory using SWI PROLOG version 5.7.15 [Wielemaker, 2010] on a Windows 7 machine.

After 4 hours and 25 minutes (99% of processing time for the generation) the execution was manually stopped. In total, 15,932,143 stories had been created and evaluated. Only 4,234 stories received a rating over the threshold (0.01), which means a rate of only 2 good stories in every 10,000 generated stories. Not every story rated as good by the implementation was high-valued according to humans and many discarded stories were probably high-valued. As checking several million stories by hand is quite a difficult task, only a random sample has been picked from the set of good stories to check that the stories are, at least, meaningful. "Bad" stories in the discarded set are also checked, and most elements of the "bad" set were found to be meaningless. A subset of this sample and some stories in the discarded set were chosen as the test set for the evaluation function explained in Section 4.3.

From the set of good stories, the mean overall quality value was 0.14. The best story received a rating of 0.64:

> John was in the bus stop. A man was in the bus stop. John realized that it was late. He was surprised. John asked the time to the man, and he said that it was two o'clock. John supposed that he was going to die. Some time before, John had agreed with a godfather that he would pay him some money before 2 o'clock. John wanted to ask for help to the man in the bus stop. The man in the bus stop, then, said that it was too late, and he killed John.

Only 319 stories received a rating over 0.25 (7.53% of stories). It is possible to state that (approximately) this should be the set of stories that are not only meaningful but also high-valued to some extent, although checking all these stories and evaluating them using human criteria would be needed.

4.5. Improving Conceptual Space Exploration by Constraining

The previous section showed that the amount of stories that the system can generate is so high that practical generation is unmanageable. In this

context, it makes sense to constrain the bounds of the conceptual space so that the number of generated stories becomes smaller. The previous section showed that 99,97% of stories were discarded. Considering that the required time for generating any story is approximately equal, a large percentage of total computing time was spent on incorrect stories. Also, humans do not achieve creativity by trying all possible alternatives and then choosing the most appropriate one.

These two aspects of the exhaustive generation approach have led to the modification of the basic generation system to include the possibility of a *pruning function*, $\mathscr{P}$. The second version of the prototype includes a domain dependent function that returns a boolean value: if it returns true, the current branch in the generation is explored. Otherwise, the generation stops at that point for the current branch, making the algorithm try a different option.

Although the current prototype permits any implementation of this pruning function, we want to explore the possibility of basing this pruning on the evaluation function that has been defined in Section 4.2. The definition of this pruning can be formally represented by Equation 4.4:

$$\mathscr{P}(s) = \mathscr{E}(s) > \tau \tag{4.4}$$

where τ is any real parameter in the range $[-1, 1]$, as defined in Section 4.4.

4.5.1. Adapting the Evaluation Function for Use as a Pruning Function

To adapt the evaluation function so that it can be used as a pruning function, an iterative modification based on the exhaustive generation approach results has been carried out. The overall point is to adapt the evaluation function in such a way that it detects unfinished stories, and treats them so that an adapted procedure can be applied to them.

If a partial story is evaluated as if it was a complete story, the overall rating it is likely to be quite low. If it was discarded before completion, the system would be discarding a potentially high-valued story. The basic E function must be adapted so that it considers partial stories in a different way.

This is done by providing a set of additional rules. New rules have the same format as those described in Section 4.2.1, but they are designed so that they avoid low rating of partial stories when extensions of the story may achieve high ratings. If the story rates as an unfinished story above

a user given threshold, the partial story is considered to be promising and it is extended. Otherwise, the story is considered not promising and it is discarded.

The definition of the rules is based on experimental results. The process of creating good rules for constraining the search in the conceptual space takes four steps:

1. Using the current E function (the function which *does not* take into account partial stories) a sample of stories L is generated. The size of this sample is kept small so it can be checked manually.

2. Using the E_p function (the function which *does* take into account partial stories) the generation is repeated, getting the L' set (in less time).

3. Checking the generated logs from the execution in step 2 in comparison with those in step 1, those non-promising stories (according to E_p) that yielded good stories (according to E) are identified as the ∂ set.

4. Each rule in E_p is adapted so that it accepts the stories in ∂ as promising. This process involves several evaluations of every element in ∂, trying new rules and testing them so that the evaluation fits the authors' criteria.

Steps 2–4 are repeated until $L = L'$. When this happens, a different sample set is chosen for generation.

4.5.2. Results of the Constrained Conceptual Space Exploration Approach

To compare both approaches (exhaustive exploration and constrained exploration), the constrained version was run to generate exactly the same number of good stories than in the previous experiment, 4,234. Using the same machine, the generation took 53 minutes. The obtained set of stories, however, was not the same, so the application of the pruning function does not only affect time, but also the output set itself in terms of Wiggins' formalism. This suggests that the same $\mathscr{T}$ combined with a different $\mathscr{E}$ leads to a different set of points within the conceptual space.

Using constraints in this way, the mean value for stories rose to 0.22, and 814 stories received an evaluation greater or equal to 0.25. That is, 19.22% were "good" stories. Again, this should be proved by checking the

real quality in stories. However, the maximum value was 0.60, which is slightly lower. It corresponds to the story presented below. It is quite similar to the best story in the exhaustive exploration presented in Section 4.4.1:

> John realized that it was late. John was in the bus stop. He was surprised. John asked the time to the man, and he said that it was two o'clock. John supposed that he was going to die. Some time before, John had agreed with a godfather that he would pay him some money. John wanted to ask the man in the bus stop for help. The man in the bus stop killed John.

These results show that constraining the conceptual space search saves time and discards non-promising stories.

4.6. Benefits and Drawbacks of the Semantical Approach

As advanced in the introduction, this study served, among other things, to check that semantic approaches to narrative generation are useful, but the cost of improving the knowledge base is too high for the system to be really maintainable in production scenarios. While this was known before the system was built, the hypothesis that an explicit evaluation function could serve to reduce the required cost was not validated.

While the results showed promising results, and in-depth analysis of the system detailed in this chapter reveals that the rules are too *ad-hoc*. The author was responsible of programming them, and, although not intentionally, the definition of the characteristics of a "good" narration and the output of the rules were too coupled and tweaked to yield satisfactory results.

In a more general approach where the definition of a "good" story and the creation of the rules were totally independent, the results would not be so good. Therefore, although defining the model and the rules by hand offers full control on the specific behaviour of the system, the amount of knowledge that a human can insert into a computational system is so limited that the effort is usually not valid for a broad range of domains. This is considered a typical drawback of knowledge intensive systems, and it could not be solved by the creation of the evaluation function previously described.

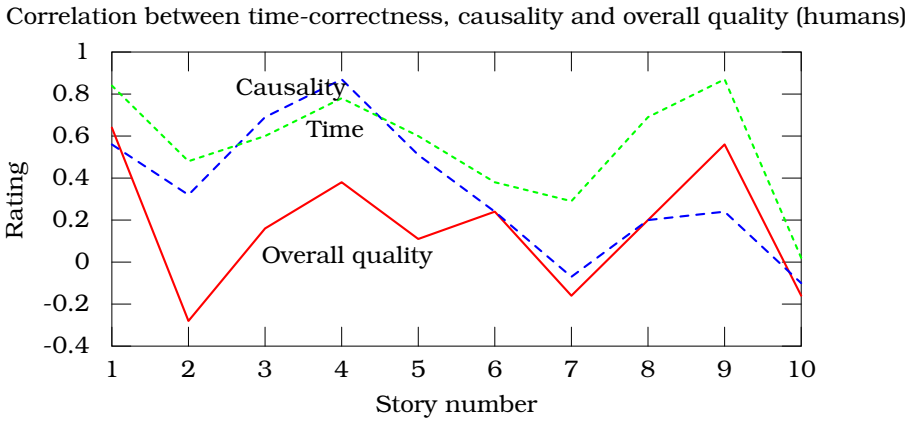


Figure 4.3: Graphical representation of the *causality*, *time-correctness* and *overall quality* (human evaluation).

As conclusion for this chapter, then, it can be said that the experience of building a knowledge based story generation system helped not to clearly identify the bottleneck that rule creation presents (which was known a priori), but to hypothesize that a different approach could be tested. When creating the variables, the definition suggested non-semantic aspects of the properties of a good narration, in the sense of the lack of external knowledge to handle it. It was hypothesized that some structural aspects could be computationally managed, which has been done, as explained in Chapter 5.

Also, the explicit evaluation function has been kept, but it was strongly modified so that it only captures structural patterns of simple stories, as it will be shown in next chapter.

4.6.1. Influential Variables

As Herman studies, there is an existent correlation between causality, chronology and overall quality [Herman, 2000]. This can be graphically depicted as shown in Figure 4.3, where a slight correlation can be perceived. Moreover, there is a strong agreement on the human evaluation of these aspects of narrations.

The Pearson's correlation coefficient between chronology and mean quality in humans is 0.67, and between causality and mean quality it is 0.6. This suggests that there is a correlation between these variables. These results will be used in next chapter as inspiration for the creation of

a structural pattern, as it will be explained.

4.7. Chapter Summary

This chapter has presented the first stage of the research, namely a computational approach to story generation based on a knowledge intensive evaluation function. While the results were positive, it has been made clear that the knowledge acquisition bottleneck is still totally present, therefore being necessary to build a different approach. The main results from the execution of the experiments suggest some ideas that form the base for the next chapter.

Chapter 5

Structural Processing of Stories

This chapter explains an artificial structural relation in narrations that is used as a basic relation to process stories in this model. This relation has been called *preconditional-link*, and next sections explain its main characteristics. The creation of a simple relation allows to computationally analyse structural characteristics of stories and build an algorithm that extracts rules for preconditional links, the so called *preconditional rules* (Chapter 6).

This relation has been synthetically defined from observation of instances of *simple stories*, as defined in Section 3.4. Thus, no cognitive model is proposed. Instead, obtaining satisfactory practical results through empirical measurements is the objective.

The *preconditional link*, as it will be later shown, is just a structural property that helps to analyse and build simple stories (Section 3.4). This kind of structural relation is valid for the restricted type of narrative presented in this research, processing more complex stories by these means will probably lead to incorrect results according to human criteria about the coherence of the generated stories. Advanced forms of narrations are not addressed.

The proposed structural relation, then, is presented with the intention of proving that structural analysis of narrative makes sense. While more extensive work is needed for the approach to be generally applicable (as addressed in Chapter 9), preliminary results suggest that this way of processing narrations is promising. In this way, the intended contribution is to show that classic semantic approaches can be complemented (and not substituted) with structural processing.

5.1. From a Cognitive Model to a Structural Definition

The system introduced in Chapter 4 made explicit several flaws inherently linked to the semantic processing of stories. These flaws constrained the development capabilities of the implementation due the fact that updating and broadening the domain information is very costly.

As explained in Chapter 2, Schank introduced the concept of *script* as a basic mental structure for memorizing in humans. This scripts are basically sets of typical actions or short stories which occur frequently. Schank and Abelson stated that this subtype of cognitive schema is basic in human knowledge [Schank and Abelson, 1977].

This sequential property of scripts led the author to hypothesize that structural properties of scripts or narrations can be used in a more useful way by computers. Putting together experience and theory, the system based on semantic processing, despite of its inherent limitations, suggested a possible change of focus. The ad-hoc definition of the variables for evaluation yielded promising empirical results, even being domain-specific. And the evaluative nature of the solution offered explicit analysis of certain features of narrations, some of them involving structural aspects of stories. Since the definition of the features, while cognitive, just modelled an approximation to evaluation of stories, the study of the system led to the hypothesis that empirical *rules of thumb* about properties of narrative could permit the processing of narratives.

That is, it was hypothesized that just by taking into account structural properties of formalized story plots, stories understandable by humans as such could be generated, as introduced in Section 1.3.

There was a clear correlation between *causality*, *chronology* and *overall quality*: stories receiving ratings for *causality* and *chronology* above zero also received rates for *overall quality* above zero. Zero was set as the threshold between acceptable and non-acceptable stories.

This has been previously identified in literature as a common psychological process when humans understand narrative [Herman, 2000], so it was considered that correctness could be indirectly modelled by replicating these kinds of understanding processes. But modelling causality and chronology requires handling massive amounts of semantic knowledge, as evidenced in Chapter 4 and other previous research.

At this point, a synthetic relation was defined, not based on any cognitive aspect. Instead, it was only *inspired* by them. The intention was to discover patterns ensuring coherent causality and chronology. These

patterns had to remain totally structural, that is, they did not have to rely on any additional knowledge base. As Herman states [Herman, 2000],

> People do seem to rely on a story-based rule of thumb when they bind strings of successively occurring events into causal and chronological wholes-e.g., I am out in a storm with a group of my peers and only I am struck by lightning; therefore I have earned the ire of the gods or, in another story, I in particular have been brought low by the turning of Fortuna's wheel.

Therefore, successive events in a story capture some amount of information, and this is a structural characteristic. Additionally, Trabasso suggests a graphical representation of causality in which events in a story are represented by nodes and causality is represented by arcs between those nodes [Trabasso and Sperry, 1985]. This kind of representation has been followed by others [van den Broek, 1988, Riedl, 2004]. In particular, Riedl suggests that *story coherence*, as opposed to *plot coherence*, requires that all elements in the graph (all actions in the narration) must be part of a *causal chain* leading to the outcome of the story [Riedl, 2004].

Based on this ideas, a heuristic was created to capture this: the *preconditional link*. Preconditional links try to describe the structural patterns in narrations that are involved in the human recognition of causality and chronology, ignoring the semantic content of these two concepts.

Next section (Section 5.2) details the model of coherence based on structural patterns of stories and Section 5.3 introduces the *preconditional link* relation, whose definition is based on this structural approach to narrative coherence.

5.2. Structural Properties as Structural Coherence

In order to correctly create a formal definition for preconditional links able to determine which stories are "good" and which are not (regarding coherence), the characteristics of "good" stories as complete units were studied, and then, the preconditional link was defined based on the definition of *structural coherence*.

Experimentation with variables shown in Section 4.2 led to the conclusion that, while the approach was promising ("good" stories could be generated), it was difficult to build a robust model of every variable. It is not claimed that modelling quality in that way is impossible, this research

only estimates that such a model requires a long term scientific investigation because the currently available technology is unable to fully automate knowledge acquisition in computers. Therefore, conclusions from this previous work about the way in which narrative quality could be computationally figured out were used to narrow down the scope.

It was decided that the common aspect of "good" quality is the *coherence*. *Narrative coherence* is *defined* for this research as the property as being recognizable as a narration by humans. While this is admittedly a conflicting definition –human criteria on this field is too variable to be formally used without adaptation–, it is used as the conceptual base for a formalization. That is, instead of looking for the generation of really good stories, the scope of the research was narrowed down to a more restricted objective. It is assumed that every good story is, at least, coherent (discussed in Section 8.1.3), so it is therefore concluded that any advance on achieving coherence in story processing is useful as a step towards the goal of narrative quality in computational generation of stories.

This assumption is not new in the field. Previous literature on formal or computational approaches to narrative assume that a narration is recognizable as such if it is *coherent* [Trabasso and Sperry, 1985, Riedl, 2004, Pérez y Pérez et al., 2007, Chambers and Jurafsky, 2008]. The meaning of coherence, although addressed from different perspectives and with slightly different definitions, conveys the idea that everything occurs for a logic reason and there is some kind of conclusion in the story. As an approximate summary of the ideas regarding this concept, coherence in Narratology and conceptual approaches to computational narrative define a *coherent* narration as a narration in which everything occurs for a logic reason according to the rules governing the domain in which the action occurs and there is an ending, outcome or solution for that narration.

However, the proposed approach in this work does not try to match any cognitive model. The proposal consists on shifting from purely cognitive to mostly structural approaches, and consequently coherence in narrations is defined in these terms, without any cognitive assumptions, as state next:

> A story is coherent in the proposed model if it is rated as such by human evaluators.

That is, the problem is passed to humans, and thus empirical demonstration of the capabilities of this computational system will have to be tested against human criteria. While this definition could seem circular, it is not. For instance, Wiggins proposed an analogous definition for Computational Creativity [Wiggins, 2006]. Therefore, if common human criteria

in empirical evaluation rates a computationally generated story as *coherent*, it will be considered that the generation program has been successful in these terms.

However, the previous definition about coherence is only useful as evaluation criteria: since no cognitive model is assumed in that definition, it does not help to generate stories efficiently. By assuming *only* that definition of coherence, one could only generate candidate stories and feed an evaluation system based on human judgements. This would be obviously impracticable.

In order to avoid that, an *additional* definition is created. This is the definition of *structural coherence*:

> A story is *structural-coherent* if three structural patterns are present in it: *focus*, *unique linkage* and *full connection*.

Focus, *unique linkage* and *full connection* are later defined. These have not been explained first because its definition is linked with the notion of preconditional links. The lack of cognitive base has lead to a definition of these properties in terms of the preconditional link, which is later explained.

It will be seen how the definition is circular: preconditional links are defined in terms of focus, unique linkage and full connection, and vice-versa. This definition is not useful by itself, then, but it will be shown, in next chapter, how the definition is useful when running the algorithm for extracting patterns because only preconditional links appropriate for an input corpus of coherent stories is accepted (Chapter 6). On the other hand, the definition could have been given in terms of graph-properties of the stories, which would have broken the circularity in the definition. It has been considered that the chosen way is more intuitive, and thus, more appropriate.

It is again emphasized that only *simple* narrative is addressed: surreal novels, for instance, are not covered by this definition (since surrealism, by definition, does not follow the domain rules governing the world in which the action occurs). Therefore, only the kind of stories fulfilling this requirement can be processed by the proposed model. The structural approach to story processing is only valid, according to the focus of this research, for simple narrations. No further assumptions are made about more complex stories.

A formal definition of structural coherence for a story s is shown in Equation 5.1. The implementation (Chapter 7) is based in this equation.

$$\begin{aligned} is\_structurally\_coherent(s) &= is\_focused(s) \\ &\wedge\ is\_fully\_connected(s) \\ &\wedge\ is\_uniquely\_linked(s) \end{aligned} \tag{5.1}$$

The definition of structural coherence have been formalized as a *boolean* evaluation function. The current model presented in this dissertation states that a story *is* structurally coherent or *it is not*. Again, this is a simplification that does not cover the complexity of human criteria. It was decided to adopt this simplification in order to keep the model simple. Although the evaluation function shown in Chapter 4 addressed a real range, it is considered that it does not add relevant theoretical content to the description of the structural approach, which is the target of this part of the research. Further work explained in Chapter 9 will address the improvement of the model so that structural coherence could be rated in a real interval.

Sections 5.6.1, 5.6.2 and 5.6.3 define the structural patterns introduced in the definition of structural coherence. Section 5.3 shows how these patterns are used and Section 5.6 formally defines them in terms of preconditional links.

5.2.1. Focus

A story is *focused* when it has a *single* outcome. That is, it ends in a single action or event. This definition is based on the work by Trabasso about plot coherence [Trabasso and Sperry, 1985]. Intuitively, if a story is told and no conclusion can be extracted from the narration, it could be considered incomplete. This has been interpreted as a lack of coherence. A formal definition of focus can be examined in Section 5.6.1.

5.2.2. Full Connection

A story is *fully connected* when every action is related to each other. This definition tries to intuitively capture the restriction of simplicity that only one single thread or plot can be told in simple stories. Fully connection both captures the fact that no action in the story must be told without a reason and that all relevant facts must be told (otherwise, the connection would be lost). Full connection is formally defined in Section 5.6.2.

5.2.3. Unique linkage

A story is *uniquely linked* if two events are related by a single idea or conclusion. This is the conceptual definition of another part of simplicity in coherent narrations. If two events are related to each other by several relations, more than one thread is present. Therefore, the story is not *simple* in the defined terms. This conceptual definition of unique linkage is defined in Section 5.6.3.

5.3. Preconditional Links

Once the definition of structural coherence (that is formally defined later) has been introduced, the created relation which will be used for this formal definition is explained in this section: the *preconditional link*.

The definition of preconditional link is only inspired by the way in which, heuristically, humans perform story understanding, but it does not try to capture any cognitive process. Instead, the current proposal defines it as a heuristic for machines, and not for humans. That is, the definition is strictly bound to computationally processable information: not necessarily the information that humans use and not necessarily processed in the way in which humans do.

Pure causality was initially considered to form the base of the structural approach. The basic concept of causality is not used in this model, while there exist some other systems that make extensive use of it [Riedl, 2004, Trabasso and Sperry, 1985].

Classic causality is not used because it is a very complex concept subject to philosophical discussion. As an example, if a mother orders her daughter to cross the river to deliver a meal to her grand-mother, and the daughter drowns in the river, did the mother cause the death? Or was it the fact that humans cannot breath under water? This deep description of causality (tightly related to guilt, in this case), makes it difficult the formal definition of *cause* beyond its semantic properties, so the structural definition of preconditional links was tackled. This relation is defined next:

> In a story formalized as the sequence of actions $\{e_1, \cdots, e_n\}$, the actions $\left\{e_i, \cdots, e_j\right\}$ (the *preconditions*) are *preconditionally linked* to e_k (the *consequence*) if they appear before e_k and the directed graph resulting from the preconditional links for all actions in the story presents structural coherence:
>
> 1. *focus*, by having all its actions converging to a single action in the story,

2. *full connection*, if has all its actions directly or indirectly rooted in a single action and
3. *unique linkage*, for which every pair of connected actions is connected by one link *at most*.

where $1 \leq i, j, k \leq n$.

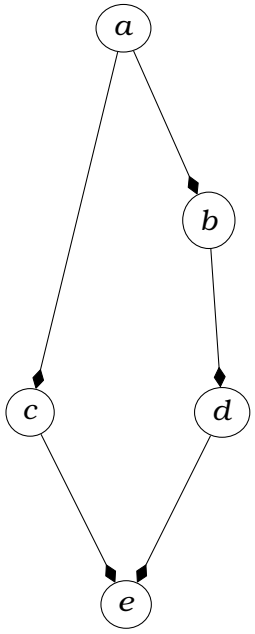
It could seem circular that preconditional links are defined in terms of focus, full connection and unique linkage and these three patterns are later formally defined in terms of preconditional links. This is because being a synthetic relation, neither the patterns nor the relation itself make sense without each other. This is way the definition is coupled. Empirical evidence for this definition has been studied in order to prove its use. This empirical study is detailed in Chapter 7.

This definition is obviously structural, that is, it only captures surface properties of stories according to a synthetic relation. As an example, if a story is composed by the actions $\{a, b, c, d, e\}$, valid preconditional links would appear in Figure 1, but not in Figure 2.

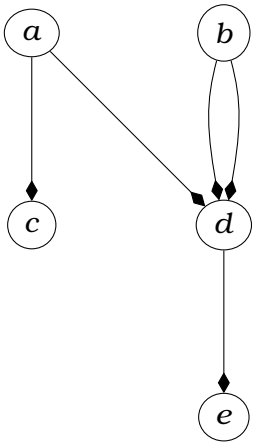
The set of actions in the precondition of the preconditional link represents conjunction. The model does not consider representing disjunction since this would lead to non-determinism, which is left out in this model. The reasoning process about causality would be more complex if a story did not uniquely determine the preconditional links of some fact.

As shown in Graphs 1 and 2, simple graphical depiction of preconditional links is used to represent the graph of preconditional links in a story in a more visual way. It is based on previous work regarding narrative causality in which causality is depicted by arrows between events [Trabasso and Sperry, 1985]. This graphical way of showing causality represents actions as labeled nodes or edges. In these graphs, the formal representation of the action is the label (see Section 3.4), and directed vertices represent causality. Origin of vertices starts in the action that is part of the precondition and ends in the action that is part of the conclusion. Therefore, graphical representation of causality forms directed, acyclic graphs. Since it is required that any member of the precondition in a preconditional link appears strictly before the conclusion, graphs cannot be cyclic.

In order to create a different symbol to make explicit that preconditional links do not represent causality, the style of the arrow has been slightly modified. Previous depictions of causality use to use classic arrows, so it has been considered that adapting the symbol could help to visually differentiate both relations.



Graph 1: Valid set of preconditional links. The arrows represent preconditional links.



Graph 2: Non-valid set of preconditional links. The arrows represent preconditional links.

5.4. Other Properties of Preconditional Links

While preconditional links are theoretically defined in Section 5.3, the current prototype has imposed additional restrictions on the properties that valid preconditional links must satisfy in order to be computable.

These restrictions are not included in the general definition of the model because they have been imposed ad-hoc and some other decisions could have been taken. In some cases, easing the implementation has been the objective, in other cases focusing on the scope of the research on narrative generation has been intended.

The first restriction establishes that preconditional chains are set to be *transitive* by definition. If there exists a chain between a_1 and a_2, and another chain between a_2 and a_3, then there exists a chain between a_1 and a_3. This is just a definition that makes easy to conceptually design an algorithm for computing preconditional links, as it will be shown in Section 5.8.

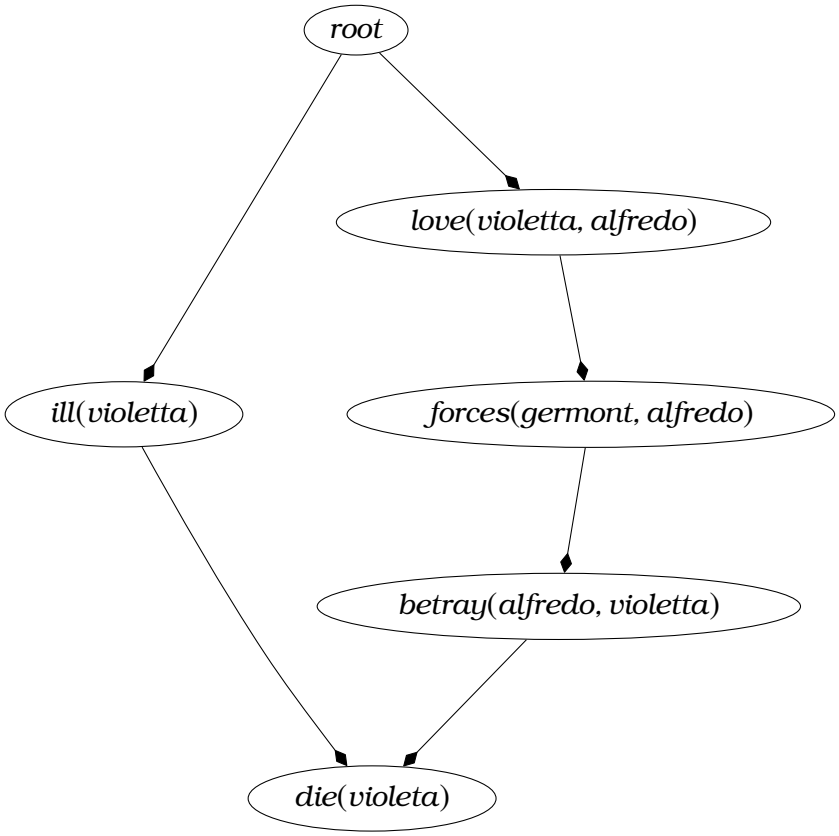
A special action for creating the root for stories was created. It is denoted by the word *root*. In order to create a preconditional link for every action all of them would have to have a previous action. The first action in the story cannot have a previous action, so, in order to keep the formal definition of structural coherence unchanged, this virtual action was added. The *root* node is used to preconditionally link the first action. It actually can be preconditionally linked to every action in the story, but in practice (shown in Chapter 6), it is commonly linked to the first or the two first actions in the story.

While it would have been possible to relax the definition of preconditional link, it was decided to add the *root* node as an additional property for the implementation because the definition of the algorithms was easier (the algorithm is shown in Section 5.8.

A preconditional link for an action a, therefore, must be either *root*, or a set of actions $a_1, \cdots, a_n$. Not only the first action in the story can be the consequence of *root*, instead, every other action can be its consequence.

Cycles have been forbidden for preconditional links. The definition makes it impossible to theoretically find a story with cyclic preconditional links, but forbidding it in the computational model has made possible to write simpler algorithms. That is, if there is a preconditional link in a story stating that $a \diamond b$, the preconditional link $b \diamond a$ would be forbidden, where a and b are actions. Since preconditional links are transitive, non-direct cycles are also disallowed. For instance, if $a \diamond b$ and $b \diamond c$, the preconditional link $c \diamond a$ would not be correct according to the definition. Disallowing cycles permits easier computational algorithms, since cycles could lead to infinite exploration and the system would have to handle it.

Graph 3 depicts a story with its preconditional links.



Graph 3: Story with preconditional links.

5.5. Preconditional Chains and Preconditional Networks

Two actions a_p and a_c are preconditionally linked if there exists a preconditional link in which a_p belongs to the *preconditions* and a_c is the conclusion.

Chains of preconditional links (or *preconditional chains*) are defined as ordered sequences of actions in which the action a_i belonging to the chain c is preconditionally linked to a_j if $i < j$, where i and j are positions in the list.

$$chain_{example} = [a_1, a_2, a_3] \tag{5.2}$$

in this example, a_1 is preconditionally linked to a_2, and a_2 is preconditionally linked to a_3.

The *network of preconditional chains* or *preconditional network* of a story s is the set of all preconditional chains for s.

5.6. Formal Definition of Structural Patterns

Once the formal definition of the main structural relation for this model not based on semantic processing of stories has been described, the structural patterns leading to *structural coherence* can be formally defined as well.

It will be seen in Sections 5.6.1, 5.6.2 and 5.6.3 that the formal equations for the structural characteristics of coherent stories, namely *focus*, *full connection* and *unique linkage* are defined based on several functions which assume the existence of preconditional links. This existence is made possible through the computation of the links as shown in Section 5.8.

5.6.1. Focus

How to compute whether a story is focused or not is formulated in Equation 5.3. This equation states that a story is focused if the number of actions belonging to that story that can be considered an outcome for the story is equal to 1.

$$is\_focused(s) \leftrightarrow |\{a \mid is\_outcome(s, a)\}| = 1 \tag{5.3}$$

To compute whether an action a is an outcome for the story s or not, the model considers the definition formalized in Equation 5.4. This equation

mathematically represents that an action is the outcome of a story if there exist some preconditional chain from *every other action* in the story to it. In Equation 5.4, the existence of a function *exist_preconditional_chain*(s, a', a) is assumed. This boolean function returns *true* if there exists a preconditional chain from a' to a, and false otherwise.

$$is\_outcome(s, a) \leftrightarrow exists\_preconditional\_chain(s, a', a) \; \forall \; a' \neq a \qquad (5.4)$$

5.6.2. Full Connection

Equation 5.5 shows the formal definition for *full connection*. The graph formed by the preconditional links is fully connected if all its actions are fully connected.

$$is\_fully\_connected(s) \leftrightarrow action\_fully\_connected(s, a) \; \forall \; a \in s \qquad (5.5)$$

An action if fully connected, according to Equation 5.6, if the special action *root* is preconditionally linked to it or all the actions in the preconditions for all preconditional links having it as consequence are fully connected.

$$\begin{aligned} action\_fully\_connected(s, a) \leftrightarrow\ & precondition\_al\_link(s, a) = root \\ & \vee \; action\_fully\_connected(s, a') \\ & \forall \; a' \in precondition\_al\_links(s, a) \end{aligned} \qquad (5.6)$$

The definition assumes the existence of *preconditional_links*(s, a). It returns a set containing all preconditional links of the action a in the story s. How this set of preconditional links is computed is explained in Section 5.8.

5.6.3. Unique linkage

Unique linkage is computed by checking that every action in the story is uniquely linked, as shown in Equation 5.7.

$$is\_uniquely\_linked(s) \leftrightarrow action\_uniquely\_linked(s, a) \; \forall \; a \in s \qquad (5.7)$$

And, to check that an action a is uniquely linked, it is checked that there are no duplicated preconditional links between any action in the story and a, as formalized in Equation 5.8.

$$action\_uniquely\_linked(s, a) \leftrightarrow \neg\ duplicates\_in(pl) \quad (5.8)$$
$$\text{where } pl = all\_preconditional\_links\_to(s, a)$$

5.7. Preconditional Rules

An action a can be the consequence of different preconditions, that is $b \diamond a$ and $c \diamond a$ is permitted in the model. In fact, it is possible to abstract the particular information present in stories about concrete preconditional links and create *preconditional rules*.

Preconditional rules are pairs of *preconditions* and one *consequence* in which the preconditions are sets of pattern-actions (as defined in Section 3.3.3), and the consequence is a single pattern-fact. Expression 5.9 shows an example.

$$go(x?, y?) \wedge see(x?, z?) \diamond want(x?, z?) \quad (5.9)$$

The example shown in Expression 5.9 has a meaning analogous to first-order logics: it means that if someone goes to some place and she sees something, that person wants what she sees. That is, this full abstraction of preconditional links makes it possible to use the information present in one story as part of the definition of some domain.

5.8. Computing Preconditional Links in a Story

Computing preconditional links in a story consists on the process of assigning the appropriate preconditional links to the corresponding facts of a story, finally obtaining a structure like the one shown in Plot 3. This structure must satisfy the requirements shown in previous sections.

For it to be feasible, the system needs a set of preconditional rules. Basically, computing the preconditional links for a story consists on *choosing* among all possible preconditional networks that can be assigned to the story through some input set of rules. Figure 5.1 depicts a black box model of the process.

Algorithm 1 shows a pseudo-code version of the non-deterministic algorithm for finding a correct preconditional network for a story. As depicted in Figure 5.1, the input of the algorithm consists on a story and a set of preconditional rules.

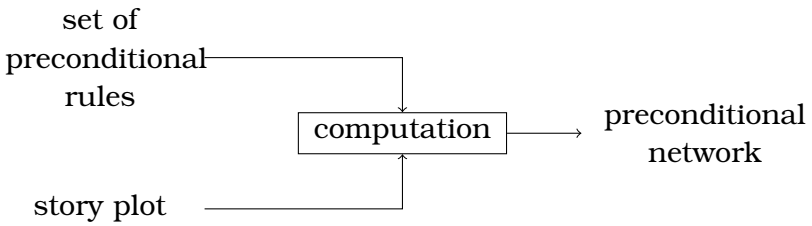


Figure 5.1: Black box model of the preconditional links computation.

In this algorithm the set R is introduced. It contains a set of *rules* that provide candidate preconditional links for each input story. These rules define groups of events that can form valid preconditional links. The details of that set, how it is constructed and the use in the model is presented in detail in Chapter 6.

Algorithm 1 Pseudocode algorithm for assigning a preconditional network to a story s.

```
1: s ← current story
2: R ← set of preconditional rules
3: for candidate_network ∈ candidate_preconditional_networks(R, s) do
4:    l ← apply candidate_network on s
5:    if is_structurally_coherent(l) == true then
6:       return candidate_network
7:    end if
8: end for
9: return "Could not find any preconditional network for s."
```

The created algorithm for computing preconditional links consists on the search for a valid assignment of these links to *every* action in the story. The algorithm tries to compute this links based on the structural model proposed along the previous sections of this chapter. It has been designed as a non-deterministic algorithm in which the function returning the candidate preconditional networks (whose pseudo-code definition is shown in Algorithm 2) does not impose any order in the way in which networks are returned. The implementation of the function in the current prototype (Chapter 7) actually imposes an order for efficiency reasons.

In Algorithm 2, **yield** returns a value but stores the state of the computation, in such a way that, in case the returning value is not valid by the calling environment, the execution continues from that point. This definition tries to approximately represent a behaviour similar to PROLOG's

backtracking or RUBY's *yield* statement.

Algorithm 2 *candidate_preconditional_networks*: pseudocode algorithm for finding candidate preconditional networks.

```
1: R ← set of preconditional rules
2: s ← current story
3: for a ∈ s do
4:    for any r ∈ R do
5:       if a can be instanced with r then
6:          add links between r's preconditions and a
7:       end if
8:    end for
9: end for
10: yield network
```

The onus of the system is then on finding adequate preconditional rules to compute preconditional links producing coherent stories according to human judgement. Chapter 6 explains how to partially automate this task. It will be shown how human intervention can drive a semi-supervised algorithm carrying out this process.

5.9. Chapter Summary

This chapter presents a structural model for narrative processing, as opposed to the knowledge intensive version shown in the previous chapter. The model is based on the *preconditional link*, an artificial relation created for this work, and used for computational purposes (not trying to model psychological or narratological concepts). The formal definition of this relation is given and its graphical depiction is also presented.

Chapter 6

Automatic Extraction of Preconditional Rules

CHAPTERS 3 AND 5 HAVE SET THE BASE for creating the definition of a rule extraction process which, when seen as a black box model, receives as input a set of coherent stories (according to the definition in Chapter 5), a set of human judgements (through supervision), and outputs a set of preconditional rules. This chapter explains how this rule gathering process is carried out and its relation with the model.

Figure 6.1 depicts a graphical schema of the pseudo-automatic extraction process. This graph intends to show the cyclic nature of the proposed algorithm.

Along next sections it will be explained how human criteria is gathered through a refined set of preconditional links. This refinement is carried out by applying partial human opinion and simple boolean query for coherence is asked to a human evaluator. An iterative rule extraction process is proposed. On each iteration, the set of preconditional rules already collected are used to generate stories. These stories are then given to human evaluators in order to check for appropriateness (coherence). After the generated stories are evaluated, the generated set is divided in "good" and "bad" stories. Then, new preconditional rules are extracted and added to the current set, thus getting an updated set of rules that is used to perform a new iteration. The rule-gathering process stops when the percentage of "good" stories falls above some given threshold [Gervás and León, 2010].

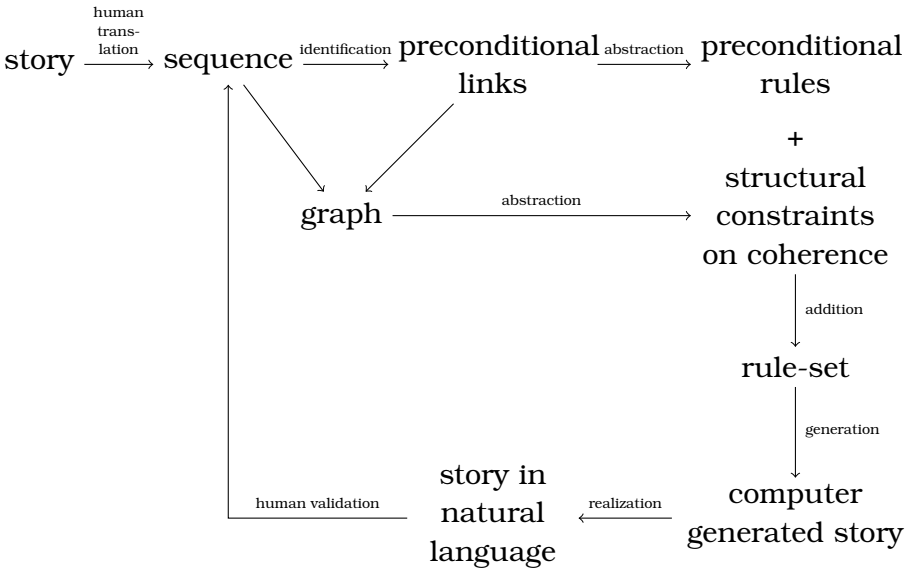


Figure 6.1: Graphical depiction of the rule extraction process.

6.1. Set of Possible Preconditional Rules

Section 5.7 explained the formal structure used to represent preconditional rules. A pair in the form of *preconditions* $\diamond$ *consequence* was used to express that some set of actions (patterns actions) could lead to some consequence (represented as a pattern action as well).

Considering some set of candidate pattern actions π (among other elements), it is possible to specify a set of preconditional rules determining the domain. The definition of the domain follows the same rules than those presented in the definition introduced in Section 3.5.

The P_D set is defined as the set of all possible preconditional rules in some domain D in which the set of pattern actions is π_D. The elements of P_D are $\{r_1, r_2, \cdots, r_n, \cdots\}$, where r_i is a rule as defined in Section 5.7, and they are composed by pattern actions taken from π_D.

For instance, Expression 6.1 shows an example of the preconditional rules for some domain D_j.

$$P_{D_j} = \{take(x?, y?) \diamond give(x?, z?);\ jump(?x) \wedge clumsy(?x) \diamond fall(?x)\} \quad (6.1)$$

P_D is an infinite set if there is no limit on the size or in the allowed pattern actions in the preconditions of the rules. That is, it would be

always possible to create a new rule since, having previously generated the rule $p_1 \wedge p_2 \wedge \cdots \wedge p_n \diamond e$, a new rule is created in the form $p_1 \wedge p_2 \wedge \cdots \wedge p_n \wedge p_m \diamond e$. Here, p_m is a new pattern action taken from π_D in which all variables are unbound and they are different from the variables in $p_1 \wedge p_2 \wedge \cdots \wedge p_n$.

While that definition makes the set infinite, a finite subset is useful to extract preconditional rules, as it will be explained in the next sections of this chapter. A simple approach to a finite subset is to set a limit on the size of the set of preconditions.

If it is considered that two sets or preconditions are equal if their pattern actions have the same structure, constraining the size is enough. Two pattern actions have the same structure if their kernels are equal, their ground actions are equal as well and they are located in the same parameter, and their variables, even if different, show the same structure regarding the entity they represent. For instance, $go(x?, park)$ and $go(y?, park)$ are defined as structurally equal pattern actions.

Regarding preconditional rules, then, $go(x?, park) \wedge eat(x?, food)$ are structurally equal to $go(y?, park) \wedge eat(y?, food)$, but not to $go(x?, park) \wedge eat(y?, food)$.

Summing up, taking into account these assumptions the only needed parameter for this approach to a non-infinite set is n, the maximum number of pattern actions per precondition. Section 3.5.1 shows how this is used in the rule-extraction algorithm.

6.1.1. Size of the Set of Rules

Previous sections have introduced a way of constraining the infinite set of preconditional rules. Since the knowledge acquisition algorithm that is proposed for this research carries out a generate and test pattern, as said in the introduction of this chapter, setting some restrictions on the set that can be generated is useful because in that way the algorithm will finish. This would not happen with an infinite set if it was totally explored.

According to this, knowing *a priori* the size of the set that is going to be explored is useful because it allows to estimate the cost of the exploration in the worst case. This worst case happens when no solution can be found, which produces a full exploration of the set (because, until the last candidate is discarded, a solution can still be found).

To compute the size of the set, the number of possible single pattern actions in some domain D must be known first. Assuming a finite number of kernels, any simple counting process outputs this number, p. This value, p, depends on the number of kernels and on the allowed patterns of variables for it. That is, for a kernel *eat*, the patterns of variables $x?, y?$

and $x?, x?$ could be allowed, thus producing $eat(x?, y?)$ and $eat(x?, x?)$. If those were the only pattern actions in the domain D, p_D would have a value of 2.

The set of possible preconditional rules for some type of action a is denoted by P_a, where a is a kernel in some domain D. Knowing the values for p_D and n (which is a parameter), the definition of the amount of possible preconditional rules can be computed according to the expression in Equation 6.2. In this equation, n is the maximum number of actions that can be part of the precondition (that is, the size of the precondition in the preconditional link is restricted), and v is the cardinality of the set of kernels.

For simplicity, it is assumed, in Equation 6.2, that every kernel can be instanced in the same way. That is, every kernel can create p_D pattern actions.

$$|P_a(v, p_D, n)| = 1 + \sum_{i=1}^{n} (p_D \times (v - 1))^i \tag{6.2}$$

where 1 is added to the expression because the precondition *root* is always a candidate, and since cycles are not allowed in the definition of preconditional links, the set of actions that can belong to the preconditions is equal to the set of actions, except the action that is on the effect, that is why $v-1$ and not v multiplies p.

From the definition shown in Equation 6.2, the cardinality of the set of preconditional rules, $\mathscr{C}$, can be defined. Equation 6.3 represents the formalization of this cardinality.

$$|\mathscr{C}(v, p_D, n)| = \left(1 + \sum_{i=1}^{n} (p_D \times (v - 1))^i\right)^v \tag{6.3}$$

where v, p_D and n are defined in the same way as before.

Instancing the parameters of Equation 6.3 can give an example of how big the set is. For example, let us say that $v = 15$, (15 kernels compose the domain), $n = 2$ (1 or 2 actions can be in a precondition) and $p_D = 4$ (4 types of pattern actions can be created for each kernel). With a domain composed by 15 verbs, which is the analogous definition when interpreting the domain with human knowledge, the space of preconditional rules in which valid subsets can be explored has a cardinality shown in Equation 6.4.

$$|\mathscr{C}(15,4,2)| = \left(1 + \sum_{i=1}^{2} (4 \times (15-1))^i\right)^{15} \tag{6.4}$$
$$= 365 \times 10^{50}$$

That is, 365×10^{50} possible sets of causal rules are candidates. According to the measurements on a AMD Phenom^{TM}9550 Quad-Core Processor, approximately 3000 sets are explored per second. This makes a full exploration of the set last 3.86×10^{41} years. While this would happen, according to the proposed algorithm, in the worst case, it is obviously an intractable solution that should be avoided or reduced to make the experiments feasible.

The rule acquisition method must process the set of preconditional rules. Since the set is so big, some solution must be created to turn its exploration a tractable problem. In order to bypass this issue, the search is more aggressively informed and the explored subset of the space is smaller. The used techniques are explained in Section 6.2.

6.2. Constrained Set of Preconditional Rules

The previous section has shown how big the set of candidate preconditional rules is. Although finite, it is too big to be tractable. Like most Artificial Intelligence problems facing this issue with space exploration, informing the search, that is, applying restrictions by the inclusion of additional knowledge in the process, can be the key to transform an intractable solution into a tractable one. This has been the chosen solution.

6.2.1. Restrict the Search for Candidates in Stories

The main improvement on the restriction of the set has been to generate only promising preconditional rules. This has been done by defining a set whose members in the form of preconditional rules are only generated from elements from analysed stories instead of taking them from the whole domain.

For instance, if preconditional rules are going to be extracted from Formal Story 2, the domain would be composed by the kernels *go*, *buy*, *watch*, *like* and *leave*. If for each of these kernels all possibilities were examined, the set would be too big. However, many preconditional rules are actually not valid for this story. Given the definition of preconditional rules

explained in Section 5.3, the preconditions must appear before the consequences. Therefore, it makes sense to include in the set only those rules in which the actions included in the preconditions appear strictly before the preconditions in the story.

go(*john*, *cinema*)
buy(*john*, *popcorn*)
watch(*john*, *movie*)
like(*john*, *movie*)
leave(*john*, *cinema*)

Formal Story 2: Example formal story for extracting preconditional rules from a more reduced set.

According to this restriction, pattern actions formed by the kernel *go* (*go*($x?, y?$), for instance) can only be the conclusion of *root*, because no action is before that one in the story. The only reasonable preconditions for pattern actions generated for *buy* are those generated for *go* or *root*, and so on. This means a very important improvement in the computational definition of the set, and therefore has a big impact on the performance.

Equation 6.3 can be used to estimate the cardinality of this set of sets of preconditional rules without this improvement, which is computed in Equation 6.5. With 4 instantiations of each kernel as pattern actions and a maximum of 2 actions per precondition, the size of the set is 15×10^{11}, which would be equivalent approximately to 16 years with the used computer in the worst case (around 3000 elements per second on a AMD Phenom^{TM}9550 Quad-Core Processor).

$$|\mathscr{C}(5,4,2)| = 15 \times 10^{11} \tag{6.5}$$

With the explained improvement, the cardinality set of candidate actions for each action in the preconditions is computed by Equation 6.6.

$$candidates(position, p, n) = 1 + \sum_{i=1}^{min(n,position)} ((position - 1) \times p)^i \tag{6.6}$$

where *position* is the order of the particular action in the story and n is the maximum amount of allowed actions per precondition and p is the number of possible generated pattern actions.

The cardinality of the set is noticeably reduced, as shown in Equation 6.7, considering that the system looks for a rule for each action in the story. The result of the cardinality of the $\mathscr{C}'$ set (with the restriction applied) for the example story is computed by accumulating the number of rules of the first action plus the second action, the third one, and so on.

$$\begin{aligned} |\mathscr{C}'_{example\ story}(5,4,2)| &= \sum_{i=1}^{5} candidates(i,4,2) \\ &= 1+5+73+156+272 \\ &= 507 \end{aligned} \tag{6.7}$$

The improvement on the size of the set is obvious. While this is a toy example, it is clear by examining the equations that less time is required. This way of constructing the set makes it possible to perform experimentation.

6.2.2. Limiting the Candidates by Distance

While the modification of the trivial definition of the set shown in Section 6.2.1 severely reduced the required computational cost, the equations show that the problem is still computable only in exponential time.

To permit a more fine-grained improvement, the model accepts a new parameter, v_p, which sets the maximum number of actions that can be searched backwards in a story for including as members of the precondition.

For instance, if v_p is set to 2, only 2 actions can be searched backwards for *like* according to Formal Story 2. Thus, only pattern actions created from the kernels *buy* and *watch* can compose the preconditions of pattern actions generated for *like*. While this does not noticeably affect short stories, the amount of sets of preconditional rules for long ones (those with many kernels) can be reduced by tweaking this parameter. Equation 6.6 is therefore transformed to Equation 6.8.

$$candidates'(position,p,n,v_p) = 1 + \sum_{i=1}^{min(n,position)} \left((min(position-1,v_p)) \times p\right)^i \tag{6.8}$$

6.3. Rule-Extraction Algorithm

The rule extraction algorithm is based on the function for identifying coherent stories (the *is_structurally_coherent* function shown in Section 5.2) and a rule generation algorithm later explained. This rule extraction algorithm iteratively generates candidate sets of preconditional rules according to the order imposed by the algorithm shown in Section 6.3.2 and checks whether the stories in the corpus are coherent when coherence is evaluated using those generated rules. The rule-set which makes all the stories in the corpora be evaluated as coherent is chosen as the acquired set of preconditional rules. If no set of rules is found, the algorithm finished with an error return value.

6.3.1. Input Corpus of Simple Stories

The first step in the rule gathering algorithm consists on the processing of a set of stories as a corpus. The algorithm requires a corpus composed by stories that are deemed as coherent *a priori*. The rules collected by this first stage will be taken as preconditional rules defining a generation process for "good" stories.

The corpus is sequentially processed. Each story is analysed as explained in Section 6.3.2, and the corresponding preconditional rules are extracted. After that, a story is generated from those rules (as detailed in Section 6.3.3) and human criteria about it is obtained. That new story is then used as a corpus to repeat the process until the stop condition is satisfied (Section 6.3.6).

It would have possible to perform preconditional rule extraction from the corpus with all the input stories at once. In such a version of the model all kernels from all stories would have been used to search for plausible rules. While this is possible, the cost of the exploration of the set would be much higher because, according to Equation 6.7, the cost grows exponentially with the number of kernels. Additionally, it does not add any theoretical improvement to the model and, in the preliminary tests that were carried out where all stories were taken together, the was no difference in the quality of the results.

Any corpus of stories formalized as shown in Chapter 3 is processable by the proposed model and by its implementation (detailed in Chapter 7). However, the results will only be valid if the content of those stories satisfies the constraints imposed by the definition of *simple stories*, as previously defined.

The input corpus is only explicitly used in the first iteration of the preconditional rules acquisition process. After that first iteration, the rules are refined by querying human opinion (Section 6.3.4), and thus the input corpus is not used again as such. The impact of the quality of the corpus, therefore, is somewhat reduced by this direct human intervention. It is difficult to quantify the extent of the impact of the corpus because it depends on the particular quality of the stories and the way in which a human supervisor rates the stories generated with the rules collected at some stage of the process.

In fact, on each iteration, a new story is generated, so it could be said that a new corpus based on the initial one is being created: nothing would prevent from using the coherent stories generated in the process as input corpus for another execution, although this has not been addressed. Additionally, human supervision makes it possible to create "bad" stories, or non-coherent ones when they are rated.

This corpus of non-coherent stories can only be synthetically generated in practice (as in this model) because there does not exist a corpus of "bad" stories. While the set of public coherent stories written by humans is huge, non-coherent stories, in the terms defined in this dissertation, are hardly available.

6.3.2. Generating Preconditional Rules

Both for each story in the input corpus and for each generated story (as shown in following sections), preconditional rules are extracted. This is carried out as detailed in Algorithm 3. This algorithm is analogous to the algorithm for creating preconditional links shown in Algorithm 1, and it implicitly abstracts the rules from the links in a variabilization process [Charniak and McDermott, 1985].

Algorithm 3 Pseudo-code algorithm for acquiring preconditional rules from a corpus of stories.

```
1: s ← current story
2: for next_candidate ∈ next candidate set of preconditional rules do
3:    s_linked ← compute preconditional links of s with next_candidate
4:    if ϵ(s_linked) == true then
5:       return next_candidate
6:    end if
7: end for
8: return "Could not finish rule gathering process."
```

In the implementation the set is not generated a priori and the candidates extracted, instead it is iteratively generated. The algorithm consists on a *generate and test* pattern in which sets of preconditional rules are iteratively created and checked against the input story. If the current set of preconditional links makes it possible to rate the story as structurally coherent according to the definition presented in Section 5.6, the set of preconditional links is considered acceptable and it is returned.

From a theoretical point of view the order of extraction of preconditional rules is not specified in the model. That is, any order of generation could work, so no restrictions are imposed. On the other hand, the order in which the rules are tested influences the result of the algorithm, so in practice, the order must be taken into account.

Although this is somewhat part of the particular implementation of the solution (fully detailed in Chapter 7), the order of generation is defined here for completeness.

The story that is being processed is s, and it is an ordered sequence of actions: $s = \{a_1, a_2, \cdots, a_n\}$. The kernels for the actions can be extracted following the order of the actions in the story, avoiding duplicates. Then, the set of kernels is stored in an ordered set K. Therefore, $K = \{k_1, k_2, \cdots, k_m\}$. To generate the rules, this order establishes the order of the generated rules.

Each candidate set of preconditional rules is generated as described in Algorithm 4. The formal definition of the *candidate_rules* function can be examined in Algorithm 5.

Algorithm 4 Order of generation of the candidate sets of preconditional rules.

```
1: s ← current story
2: K ← set of kernels from s
3: R ← {}
4: for i ← 1 until i = |K| do
5:     add candidate_rules(K_i, s) to R
6: end for
7: return R
```

For the function *candidate_rules*, a new parameter is included: the maximum number of rules per kernel, m. So far, only one rule per kernel was considered, but this is not complete. It could be the case that two actions with the same kernel are present in some story, as exemplified in Formal Story 3.

In this story, a preconditional rule in the form of $root \diamond go(\_?, \_?)$ will

go(*john*, *cinema*)
buy(*john*, *popcorn*)
go(*john*, *home*)

Formal Story 3: Example story with two actions sharing the same kernel (*go*).

be found by the algorithm. This is because the first action in the story is *go*(*john*, *cinema*) and *root* will always be preconditionally linked to the first action in a narration, as explained before. The notation _? is used here to represent *any* variable (x?, y?...).

However, it could be the case that the kernel that corresponds to *go*(*john*, *home*) needs another preconditional rule. This rule must be different from *root* $\diamond$ *go*(_?, _?) to be able to find a structurally coherent preconditional network. This is why more than one preconditional rule per kernel is allowed (from 1 to m preconditional rules, in this model).

Algorithm 5 Candidate rules for each kernel.

1: $s \leftarrow$ current story
2: $k \leftarrow$ current kernel
3: $m \leftarrow$ maximum number of rules per kernel
4: **for** $i \leftarrow 1$ **until** $i = m$ **do**
5: **yield** i rules for k in story s
6: **end for**

Algorithm 6 Algorithm for yielding a concrete number of rules (auxiliary algorithm for Algorithm 5).

1: $s \leftarrow$ current story
2: $k \leftarrow$ current kernel
3: $j \leftarrow$ number of rules to be generated
4: *rules* $\leftarrow \{\}$
5: **for** $i \leftarrow 1$ **until** $i = j$ **do**
6: add a new rule to *rules* according to *single_rule*(s, k)
7: **end for**
8: **yield** *rules*

The **yield** statement in Algorithms 5 and 6 returns a value but stores the state of the computation, in such a way that, in case the returning

value is not valid by the calling environment, the execution continues from that point, as defined in Section 5.8.

The function *single_rule*(*story, kernel*) yields rules according to the next order. It is assumed that there exists a function iteratively creating patterns of variables $var_a, \cdots, var_b$, as later explained:

1. First, the *root* token: $root \diamond kernel(var_1, \cdots, var_n)$.
2. Then, the kernel for the previous action in *story*: $kernel_{prev\ action}(var_i, \cdots, var_j) \diamond kernel(var_x, \cdots, var_y)$.
3. After that, step two is repeated by going backwards in the story until the first action. That is, if the story contains the actions $\{a, b, c\}$, the tested preconditions for the action c will be formed by the kernel in b, and then by the kernel in a.

This order assumes some set of variables $\left\{var_1, \cdots, var_j\right\}$. The variables are created by analysing the tokens of the actions in which the creation of preconditional rules is based. The process first examines all the different tokens in the story, yielding a set $T = \{t_1, \cdots, t_n\}$. Then, a corresponding set of variables is created, mapping each token to a new variable: $V = \{v_1, \cdots, v_n\}$.

When the search for preconditional rules starts and the kernels must be completed with variables to create pattern actions, the set of variables for each pattern action is assigned using the mapping between tokens and variables according to the story.

For instance, if a mapping $\{john \rightarrow x?, cinema \rightarrow y?, home \rightarrow z? \cdots\}$ was created, searching for a preconditional rule for *go* would yield the pattern actions $go(x?, y?)$ and $go(x?, z?)$ because the story only contains the actions $go(john, cinema)$ and $go(john, home)$.

Summing up, the example story in Formal Story 3 would perform the search for a set of preconditional rules for the kernel *go* by exploring the set of next candidates for a maximum value of actions per preconditional rule of 2 and a maximum value of 2 rules per kernel (the search would end before exploring this whole set):

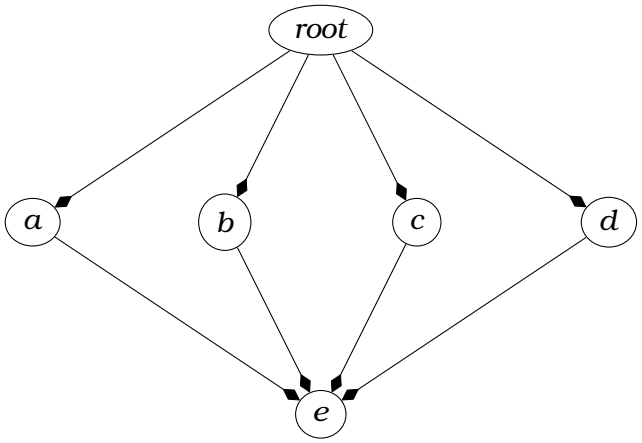
$$root \diamond go(x?, y?)$$
$$root \diamond go(x?, y?); buy(x?, a?) \diamond go(x?, z?)$$
$$buy(x?, a?) \diamond go(x?, z?)$$
$$buy(x?, a?) \diamond go(x?, z?); go(x?, y?) \diamond go(x?, z?)$$
$$go(x?, y?) \diamond go(x?, z?)$$

The returned set of preconditional rules is next:

$$root \diamond go(x?, y?); buy(x?, a?) \diamond go(x?, z?)$$

The order in which preconditional rules are gathered (generated, actually) and the value for the parameters certainly *affect* the output of the rule-acquisition process. Other rule selection order and other parameter tweaking would yield very different structural patterns for stories, which would lead to acceptance or rejection of different stories as coherent.

For instance, an order in which every action would be first preconditionally linked to *root* and every action but the last one would be preconditionally linked to the last action is possible and would create structurally coherent stories. For example, Graph 4 shows an instance of this structural pattern[1].



Graph 4: Example of the resulting structural pattern when changing the selection order.

If this pattern was commonly found when extracting preconditional rules, these would be too specific and they would not be useful to create new content, since the rule-collection process for every story would be yielding these rules:

$$root \diamond a$$
$$root \diamond b$$
$$root \diamond c$$
$$root \diamond d$$
$$a, b, c, d \diamond e$$

[1]Thanks to Dr. Peinado for discovering this issue.

These rules are of little use outside the input story. While the system would still output new content, it would be very difficult to improve the generative capabilities of any story generation system with such rules. It is therefore concluded that the order and the particular restrictions in which preconditional rules are generated affect not only the efficiency of the algorithm, but also the quality of the results.

The chosen parameters for the implementation are explained in Section 7.2.2. These have been empirically tweaked to perform acceptable preconditional rule extraction.

6.3.3. Story Generation

Intervention of humans regarding knowledge acquisition in the development of Artificial Intelligence systems must be properly isolated from the rest of the system in order to avoid non-intended influence [Ritchie, 2008]. This increments not only the maintainability of computational systems, but also helps to identify the limits of the system.

To ensure to the possible extent this property still including human supervision to perform preconditional rule gathering, human criteria is collected by querying a simple boolean evaluation of coherence over automatically generated stories (this process is detailed in Section 6.3.4). The proposed model tries to establish clear bounds on the participation of human knowledge by obtaining very specific information (yes/no) from very specific sources (computer generated stories). As it is explained in Section 7.3, the experiments that have been carried out do not allow for any other interaction between the evaluators and the system.

Automatic story generation is therefore addressed in this work. The theoretical model that is being detailed in this chapter does not impose a single way of creating stories. From an abstract point of view, any story generation process whose outcome is directly influenced by the gathered preconditional rules can be used to adapt the collection of preconditional rules. If this condition is satisfied, the system can automate the refinement of the preconditional rules. If it is not, the refinement is not ensured.

Since the theoretical model is not imposing any restriction, the definition of the computational system for generating stories is detailed as part of the implementation, in Section 7.3. It is important to make clear at this point that the quality of this story generator is far from good, and it is an accepted flaw. However, since the focus of this work is not set on this part of the system but instead on the acquisition of the structural elements used to generate, better story generation is planned as part of the future work 9.

6.3.4. Gathering Human Criteria

As introduced in Section 6.3.3, human criteria must be used to differentiate "good" stories from "bad" ones. The process considers stories generated by any procedure fulfilling the requirements previously explained and a judgement process based on human criteria is fed with these stories.

Analogously to story generation, the theoretical model does not impose any particular way of applying human criteria on the generated stories. From a black box perspective 6.2, a story is inputted in this stage of the process, and a boolean output value is expected: *true* if the story is coherent and *false* if the story is not, according to the evaluator criteria.

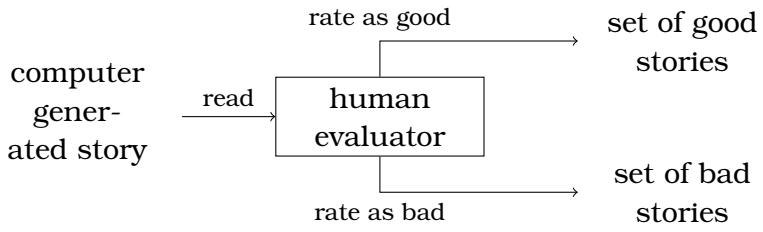


Figure 6.2: Black box model of the judgement gathering process.

More concretely, for a simple implementation (shown in Section 7.3) the model proposes a straightforward criteria-gathering process: a natural language version of the story is created and given to the human, who simply reads it and classifies it in these terms (boolean value for coherence).

A boolean classification of the quality of a story is admittedly an oversimplification of the problem. As it has been suggested in Chapter 4, it is clear that the set of variables influencing the judgement of quality contains more aspects, and these are not being taken into account. While this limits the scope of the solution, it is considered to be a reasonable option. It narrows down the scope of the research and makes it possible to put the focus on the process itself (the framework for structural schemas acquisition). A more detailed discussion is presented in Section 8.1.5.

6.3.5. Gathering and Merging "Good" and "Bad" Preconditional Rules

Algorithm 3 extracts preconditional rules from any set of stories as long as there is some possible assignment of causes making the evaluation of coherence return a valid value. This produces a set of rules that can be

applied in several systems, particularly in automatic storytelling, as it has been done in the prototype implementation.

In a simple approach, preconditional rules extracted *only* from coherent stories were used as basic rules in which a simple unification pattern generated a story valid accordingly. Such a system was implemented. When testing the implementation of this proposed generation system, non-coherent stories were generated as if they were coherent according to the rules that the system had collected so far. They were obviously not coherent from a human point of view and also taking into account the intended meaning of actions in the training corpus: a dead character was doing things after his death. Formal Story 4 shows that generated story that allowed the identification of this error.

love(*alfredo, violetta*)
love(*germont, violetta*)
love(*violetta, alfredo*)
die(*germont*)
force(*germont, violetta*)
die(*violetta*)

Formal Story 4: Non-coherent story generated by a simplistic version of the model. This story helped to detect some limitations that were later addressed in the definitive algorithm.

After analysis, the reason of this lack of coherence in the first prototype without bad stories was unveiled: simply joining rules from each iteration of the system, assuming they were coherent, led to a wrong solution: preconditional rules were coherent only for their originating stories, but once merged with other rules the resulting set did not ensure coherence.

To fix this important flaw, the inclusion of a process for identifying also non-coherent rules (and therefore discarding them from the set for generation) was addressed as previously explained in Section 6.3.4.

In order to do this, the current model considers a rule-set partitioned in *good rules* and *bad rules* after human intervention. To abstract this simple classification to insert the rules in either set, after the story has been classified, the preconditional rules from the stories are extracted (as previously detailed), and then, these criteria is followed:

- The set of good rules contains those abstracted preconditional links present in stories tagged as correct by humans.

- The set of bad rules contains the abstracted preconditional links present in stories tagged as non-correct *if* these abstracted preconditional links are not in the set of good rules.

That is, if a story considered correct s_i, after analysis, created the preconditional rules $\{r_1, r_2, r_3\}$ and a non-correct story s_j created rules $\{r_1, r_4, r_5\}$, the resulting *good* subset would contain the rules $\{r_1, r_2, r_3\}$ and the *bad* subset would contain the rules $\{r_4, r_5\}$.

There are many more ways of creating the rule set, but for prototyping the system and for explanation purposes this approach is simple and useful. The study of more sophisticated ways will be addressed in further research 9.

6.3.6. Stop Condition

Once the set of "good" and "bad" rules has been updated with new preconditional rules, the next iteration starts. As shown in Figure 6.1 and Algorithm 3, the iteration performs the loop in the story generation process. That is, the updated set of preconditional rules is used to create a new story. This new story could be coherent or not, and so it will be rated by the human evaluator.

The algorithm was set to stop when a certain number of stories were rated as correct in a row. The stop condition has been empirically set, so it is therefore explained in detail in Chapter 7, where the empirical results led to a conclusion from which the stop condition was deduced: *saturation*, explained in Section 7.5.1.

6.4. Chapter Summary

This chapter proposes, based on the notion of preconditional link and on the definition of *narration* that has been detailed in previous chapter, a human-supervised process for acquiring structural schemas, in the form of rules. These rules can then be used to generate stories, as it will be explained in the next chapter.

Chapter 7

Implementation and Results

This chapter shows the implementation of the model that has been proposed along past chapters and how this implementation has been used to carry out some experiments that try to validate, to the possible extent, the original hypothesis.

Additionally, the experimentation was not only valid to check that the initial ideas were relatively correct. It also has helped to identify new problems and challenges that are either limitations of the approach or new objectives targeted as part of the future work.

Both the implementation and the tests have been carried out on a AMD Phenom^{TM}9550 Quad-Core Processor. Only one processing core has been used when executing the searches for rules and the generation of stories, although the nature of the searches of the different proposed algorithms would allow for a parallel solution. This has not been implemented since it has been considered that the study of how to parallelize the code efficiently and the cost of that implementation would be too high. In the best case a speedup of 4 would have been reached. While this would have been noticeable for the executed tests, the nature of the problem would not have been turned the whole exploration of the proposed sets into a tractable problem, so the effort would have been useless if the search space had not been not reduced. Additionally, parallelization is outside the scope of this research.

7.1. Corpora of Stories

Three domains have been used during the development of this research: murder stories, Aesop's fables and short opera plots. They have been used at different stages of the development of the proposed system, and they

have incrementally led to conclusions that have made the final results possible.

They are all detailed here for completeness purposes, although the only shown execution example here was run with operas (Section 7.1.3). The experiments with murder stories is shown in Chapter 4, and the preliminary experiments with fables were not satisfactory enough to carry out a full experimentation with external human judges (it is later explained why). It was considered that including here the other studied domains could help to justify the selection of that domain.

7.1.1. Murder Stories

The first working prototype of the evaluation function was built for short plots of murder stories. The intention was to study the plausibility of an evaluation function for stories, not only bound to pure correctness [León and Gervás, 2010]. The results from this chosen domain are shown in Chapter 4. This implementation considered a bigger set of variables, which were chosen without any narratological or psychological model. The selection was based on the author's intuition about what constitutes a good murder story.

The empirical results were promising. Experimentation with humans showed that creating an evaluation function measuring several, heterogeneous variables based on a sequential analysis of a formalized version of texts can match human criteria. However, it was soon clear that just by shifting the focus was not enough: the knowledge acquisition bottleneck was still present.

Once this was identified, the domain of murder stories was abandoned because it was necessary to manage a usable external corpus. This kind of resource would ensure coherence (assuming that all the stories in the corpus are coherent). There exist many murder stories written by humans and they are freely available, but they are too heterogeneous and too different between them to create a good corpus without a very big effort. This was why a new domain was chosen for shifting to structural processing of stories.

7.1.2. Aesop's Fables

After the development of the first version of the evaluation function, it was clear that refining the model and reducing the research scope was needed for a robust model to be feasible. The focus was then redirected to a more clear set of narrative characteristics. Therefore, a new domain was

chosen. In order to make this decision, several aspects were taken into account:

- Simplicity of representation. This reduces the effort needed to formalize stories, and yields the current average computational power of a personal computer as completely valid for implemented tests (the current tests have been run on a AMD Phenom™9550 Quad-Core Processor).
- Grounded semantics. The meaning of the events is plain and it does not require to understand anything beyond the action.
- Appropriate narrative properties. The domain must fulfill the requirements that this research sets on the scope of application of the solution.
- Availability of corpora. Since a training algorithm is to be applied on the input data, freely availability of the data is mandatory.

Fables were chosen as the next working domain[1] because they have several advantages as domain for research and study:

- Fables are usually short. This is an important advantage of fables over novels, for instance, and it makes it possible a simple representation.
- Fables are simple. Content in fables does not analyse complex human behaviour, and story plots are not excessively complicated. This ensures grounded semantics and, again, a simpler representation than for novels, for instance.
- Fables are available. Fables written by many authors, those from Aesop [Aesop, 1992], Perrault [Neil and Simborowski, 1993] and others are freely available for download on the Internet, and without any restricting licence. A large corpus is available.
- Most classic fables are translated to many languages. This allows for experimentation with fables and human judges with different native languages, at least for preliminary testing.
- Fables define common human behaviour. It is true that, in general, classic fables have animals as protagonists. However, these protagonists really play the role of human beings.

[1] Thanks to Dr. G. Ritchie for this suggestion.

- Fables do not contain "magic" or other non-real content. Since modelling *magic* is never easy, fables are more suited for scientific analysis than fairy tales, for example. Grounded semantics are, again, possible.

- Fables are usually known by people. Although this is just an assumption based on the author's experience, in general it is possible to say that people (from western culture) has more information about fables than about experimental narrative, for instance.

All these reasons influenced the selection of fables as the base domain for this research. Although some other domains could make sense. For instance, fairy tales, if magic is omitted, could have been a good candidate. However, no other common narrative domain was considered to be more adapted to the research domain.

Having decided that fables were the target domain, a set of narrations to include in the training corpus was chosen by following a pseudo-random process: it was decided that the first book in Project Gutenberg [The Project Gutenberg Team, 2010] of the fable writer whose name appeared first on Wikipedia when looking for "fable" in the English version[2] would be the source for narrations. The first author, following chronological order, is *Aesop*. The resulting book was *Aesop's Fables* [Aesop, 1992].

An example fable taken is shown in Story 9. The formal version that was created is shown in Formal Story 5. The process of translating fables in English to the corresponding formal version is explained next.

> It happened that a Dog had got a piece of meat and was carrying it home in his mouth to eat it in peace. Now on his way home he had to cross a plank lying across a running brook. As he crossed, he looked down and saw his own shadow reflected in the water beneath. Thinking it was another dog with another piece of meat, he made up his mind to have that as well. So he made a snap at the shadow in the water, but as he opened his mouth the piece of meat fell out, dropped into the water and was never seen more.
> *Beware lest you lose the substance by grasping at the shadow.*

Story 9: Example fable: "The Dog and the Shadow" [Aesop, 1992].

[2] http://en.wikipedia.org/wiki/Fable

$$carry(dog, meat)$$
$$cross(dog, river)$$
$$see(dog, the\_shadow)$$
$$consider(dog, the\_shadow, shadow)$$
$$attack(dog, the\_shadow)$$
$$drop(dog, meat)$$

Formal Story 5: Example of a formal version of a fable ("The Dog and the Shadow").

Transforming fables in English to a formal representation processable by a computer was made by hand. The process consists on reading each sentence of a fable in English and creating a single action with the appropriate tokens representing the entities that are referenced in the sentence. For instance, the fable "The Two Crabs" starts as follows:

> One fine day two Crabs came out from their home to take a stroll on the sand.

This is translated to a set of facts represented in the partial Formal Story 6. It can be observed that the translation carries out a very important loss of content. First, literary structures like "one fine day" are removed, and "came out" is translated to "go", which is a simpler and more generic form. The formalized stories follow the conventions set for the definition of *simple stories* explained in Section 1.2.

$$go(crab_1, home, sand)$$
$$go(crab_2, home, sand)$$

Formal Story 6: Example of simple translation of a fable.

This way of creating the formalization of human-written stories is an important point of discussion because the author's knowledge about the system is clearly involved in the process, and therefore the influence he is exerting on the output can impact the results of the experiments. This issue is discussed in detail in Section 8.1.2. While this kind of manual translation is clearly a problem, doing it in a fully automatic way is a hard Artificial Intelligence problem for which no solution has been proposed so far.

Fifteen fables from a set of 82 were selected because it was considered that they well represented a good corpus for a prototype. Formalizing them required a little effort, so considering too many narrations would have been too costly. A number too low would have not been useful to perform a complete analysis. Without any special further reason about how many narrations to study, it has been considered that 15 items were enough. This, of course, is open to discussion. Section 8.3.1 studies this situation.

Once the limits of the knowledge based approach were clearly experienced, the structural processing of fables was carried out in the way explained along Chapters 5 and 6. A problem related with the particular domain was found: although fables have a very simple and similar structure, many different events occur in them. Since this happens, the number of acquired preconditional rules per kernel is low: there are typically only one or two actions for each kernel in the whole corpus. This makes it more difficult to experiment in the proposed terms because having only one or two rules leads to generation of stories extremely similar to the original ones. They would be rated as coherent most of the times but they would have been copies.

Therefore, the required domain needs to include many instances of a relatively reduced set of kernels. Operas were chosen as a good candidate because it was identified that they typically contain plots about the same type of stories (tragedies).

7.1.3. Operas

Once it was identified that fables were not well suited for acquiring rules through the application of the proposed rule extraction algorithm because of the previously explained problems, a new working domain with an appropriate corpus was selected. The requirements of this different corpus were quite similar to the ones introduced in Section 7.1.2, plus the constraint of a high proportion between number of available samples and verbs.

Operas from the XIX century follow a very classic plot based on extreme passion and a tragic ending. This suggested that it was possible to adapt operas in such a way that a basic plot was extracted and used as a short story. This adaptation was required because operas do not fulfill the requirement of being short and single-plotted. However, basic plots can be identified in operas.

The translation of the chosen operas has been carried out by the author in the same way as fables were translated. This translation has significantly reduced the content of the operas, much more than for fables.

Additionally, the personal interpretation of the author, while unintended, has probably affected the interpretation as well. Section 8.1.2 discusses this issue.

The exact ratio between the number of kernels and the stories is hard to set in general because its influence on the generation system depends on the acquired preconditional rules, the input stories, and the story generation process human criteria. However, it has been empirically analysed for this corpus (operas) and for fables. While the mean ratio of appearance between kernels and stories in the formal version of Aesop's fables is 0.26, this ratio raises to 0.4 in the chosen operas. Fables were not useful in the final prototype and operas were appropriate (as empirically checked). A deeper study of how to set this ratio a priori is planned as part of the future work.

The formal version of the opera plots can be examined in Appendix A.

7.2. Implementation

The implementation of the system has been carried out using SWI-PROLOG 5.8.3. PROLOG has been the used language and execution system because:

- The rule-extraction algorithm was designed as a generate and test pattern, and PROLOG's execution algorithm is based on the same pattern.
- The rule-extraction algorithm creates rules with variables, and the algorithm finding the particular preconditional links for each evaluated story binds those variables. PROLOG's engine performs this task.
- The evaluation function can only be applied after having found a valid preconditional network for a story. PROLOG's engine allows for a natural description of this search in terms of the properties the preconditional network must fulfill.

PROLOG also is well suited to tackle knowledge representation without the need to use an external formalism, since its data representation system is based on first order logic predicates, a very typical way for knowledge representation and the one that has been used for this research. The implementation of the description of narrations shown in Chapter 3 was easily implementable in PROLOG as a list of logic predicates that represented actions. Tokens have been represented through the use of atoms.

7.2.1. Implementation of the Computation of Preconditional Links

The implementation of the program responsible for extracting preconditional links from stories was straightforward because the algorithm was designed as a generate and test pattern, therefore perfectly matching PROLOG's classic programming approaches.

The implementation of this part of the system, the corresponding to Chapter 5, was divided in two modules: the implementation of the function for evaluating structural coherence (Section 5.8) and the implementation of the algorithm for extracting the preconditional links, which is based on the evaluation function.

According to the algorithmic design, the implementation receives a story, applies a candidate preconditional network to it, and checks that the story together with the preconditional network is structurally coherent. If it is not, the system backtracks and tries, for the same story, another candidate network. The process is repeated until a structurally coherent preconditional network is found or until there are no more candidates. In this case, the system cannot satisfy the predicate and the PROLOG query fails.

7.2.2. Implementation of the Preconditional Rules Extraction Algorithm

As it can be seen in the pseudo-code version, the implementation of the preconditional rules extraction algorithm consists on a search over some set of candidate sets of causal rules. This search finishes when a set of rules make it possible to find a coherent explanation for the story.

This part of the implementation corresponds to the algorithms shown in Chapter 6. Each algorithm shown in that chapter has been implemented as a predicate (with auxiliary code) trying to match, to the possible extent, the high level conceptual definition. This has not been always possible: for instance, *for* loops, which are not present in PROLOG, have been implemented with *forall* predicates. However, the algorithms have been coded in such a way that they are very similar to pseudo-code. Clarity has been preferred over brevity and efficiency.

7.2.3. Optimization of the Implementation

Experiments are focused from the perspective of applicability. Since criteria about structural coherence is strongly involved in the evaluation

of the quality of this work, taking into account human opinion is a requirement. Thus, the experiments have been designed to be as much user-friendly as possible, as that has been considered to be a requirement to ease testing.

As a crucial part of this requirement, the experimentation has to be performed in such a way that the user who is experimenting does not have to wait too much time to receive the next question. Since the model is designed to solve search problems which are slow, the implementation needed to be optimized.

The algorithmic optimization shown in previous chapters regarding the reduction of the space to be traversed was aggressive enough to allow for experimentation in which stories were analysed to find preconditional rules in real time. Before those optimizations were realized, a particular optimization of the implementation was carried out: *memoization*.

Memoization is a programming pattern in which computed results from functions are stored to avoid computing the same values more than once. Memoization is specially useful when computing these results is much more expensive in terms of time than storing and retrieving the previously computed values [Michie, Donald, 1968].

Memoization has improved the execution times during the experiments because of its iterative definition. Since preconditional links must be generated many times, it is often the case that the same set of links must be processed. Through memoization the computation for a given input is only executed once, therefore saving computation time.

7.3. Simple Story Generation

To check that the system is acquiring preconditional rules in an appropriate way, the experiment generates short stories based on the rules that have been collected so far. In this way, human criteria about the generated stories can be queried and therefore that can serve to indirectly assess the quality of the extracted rules. More details about the relation of this stage with the rest of the model are given in Section 6.3.3. In that section it was introduced that the particular implementation of the story generation system does not affect the model (that is, it is seen from a black-box perspective), but the current set of rules must be taken into account for the iterations to refine the set of rules.

The proposed story generation system is straightforward and simple. It must be taken into account that pure story generation is not the focus of this research, therefore this approach to automatic storytelling can be

easily improved. However, this is left as future work (Chapter 9).

Referring back to Section 3.5, the space of stories was defined as a set where all possible stories were included. The restriction making $\mathscr{S}$ finite requires to determine the value for the parameters constraining that set, δ, μ and π: the valid tokens, the maximum number of actions per story and the set of valid candidate pattern actions, respectively.

According to this, the values for these parameters must be compatible with the domain whose rules the system is to acquire. In this implementation of the model, simplified opera plots are the target, so the parameters for the set S_{opera}, the set of operas, are defined in the next list:

- δ_{opera}, the set of entities, is set to {*violetta, alfredo, germont, annina*}. This is just a set of logic atoms and no additional meaning is assigned to them. They are named after the main characters from *La Traviata*, the Verdi's opera.

- μ_{opera}, the maximum number of facts per story is set to 10. No opera plot in the corpus of short versions of operas is longer, so this value was chosen in order to generate stories alike.

- π_{opera}, the set of candidate pattern actions, is set to be the same as the pattern actions in the input domain. The kernels, along with their meaning and their respective allowed patterns actions are shown in Table 7.1.

With this definition of the set S_{opera}, and according to Equation 3.2, the size of this set can be computed. This computation is shown in Equation 7.1.

$$\begin{aligned} |\mathscr{S}_{opera}| &= \\ |\mathscr{S}_{\delta_{opera},\pi_{opera},\mu_{opera}}| &= \\ \sum_{i=1}^{\mu_{opera}} \tau(\delta_{opera}, \pi_{opera})^i &= \\ \sum_{i=1}^{10} 220^i &= 2.6681 \times 10^{23} \end{aligned} \tag{7.1}$$

Even with the severe imposed restrictions, this is a too big number to generate the whole set. However, the point is not to generate the whole $\mathscr{S}_{opera}$ set, but to generate one of its two partitions according to Section 3.5.2: the $\mathscr{G}$ set.

Kernel	Pattern facts	Meaning
ill	*ill*(_)	The character is ill, without chance of being cured.
die	*die*(_)	The character dies.
back	*back*(_)	The character is back.
love	*love*(_, _)	The characters love each other.
jealous	*jealous*(_, _)	The first character fells jealous towards the second.
breakup	*breakup*(_, _)	The first character breaks up with the second.
escape	*escape*(_, _)	Both characters escape.
together	*together*(_, _)	The characters are together as a couple.
forgive	*forgive*(_, _)	The first character forgives the second.
despise	*despise*(_, _)	The first character fells despise towards the second.
help	*help*(_, _)	The first character helps the second to reach her main objective.
kill	*kill*(_, _)	The first character kills the second.
chase	*chase*(_, _)	The first character chases the second to harm or imprison her.
forces	*forces*(_, _)	The first character forces to second to do something she does not want to do.
want	*want*(_, _)	The first character loves the second, but the second does not love the first.
kidnap	*kidnap*(_, _)	The first character kidnaps the second.

Table 7.1: List of 16 verbs for the sample story generation. This list defines π_{opera}. The symbol _ represents any variable in the set $\{x?, y?, z?, w?\}$, corresponding to the three elements in δ_{opera}.

It is not possible to know *a priori* which elements fall in the set $\mathscr{G}$ (this research would be useless if that was possible), and the rules that generate *only* elements belonging to that set are the objective of this system. Previous research shown that partitioning a set like S_{operas} in "good" and "bad" subsets yields a "bad" set, $\mathscr{B}$, much bigger than $\mathscr{G}$ [León and Gervás, 2010]. Uninformed traversal of the whole set, then, yields many more bad stories than good ones. However, it can be hypothesized that story generation based on this approach can be improved by adding some heuristics or using better traversal functions.

The proposed story generation informs the traverse operation by the application of the gathered preconditional rules. That is, instead of generating candidates without any restriction, only those candidates which can later satisfy the acquired preconditional rules are created. Since these rules have been collected by the application of the acquisition model from structurally coherent stories (assuming that every story in the corpus is, indeed, coherent), the rules should capture, to some extent, coherent narrative schemas. The tests for this hypothesis are detailed in next sections.

7.3.1. Rule Application

The way in which collected preconditional rules are applied to perform story generation is very straightforward. It has been done in this way because pure story generation is not the research focus, so a simplistic approach was considered to be better suited for the current purposes. Iteratively, a rule is randomly chosen from the set of rules. Then, it is instanced with a token from the δ set (in this instance, δ_{opera}).

The instantiation of the variables in the pattern actions of the preconditional rules ($x?$ or $y?$ in $take(x?, y?) \diamond drop(x?, y?)$) is done randomly, but it is important to take into account that rules in the set share variables and, therefore, they share them as well when those are instanced to tokens. For instance, in the next case in which the set of preconditional rules contain two rules:

$$love(x?, y?) \diamond jealous(x?, y?)$$
$$go(x?, z?) \diamond remember(x?, z?)$$

After the first rule is instanced, both the $x?$ and $y?$ variables get unified. Therefore, and although the second rule has not been used yet, it is partially instanced as well. The resulting set of preconditional rules after binding $x?$ to *violetta* and $y?$ to *alfredo* would be this one:

$$love(violetta, alfredo) \diamond jealous(violetta, alfredo)$$
$$go(violetta, z?) \diamond remember(violetta, z?)$$

After a rule has been chosen from the set and it has been instanced using any valid set of tokens, as previously exemplified, the preconditions of the rule are checked at the current point of the story. If all the actions in the precondition are present in the story generated so far, the conclusion is added to the story.

Before computing whether the story is complete or not, the implementation makes use of the "bad" rules to check that there are no structural patterns satisfying any of these "bad" rules. For instance, if the generative process using the set of "good" rules generates ..., $die(alfredo)$, $take(alfredo, something)$ and there exists the "bad" rule $die(x?) \diamond take(x?, z?)$, it is identified that the story is not structurally coherent and it is discarded.

After checking that no bad rules can be applied to the story, the structural coherence for the partially generated story is tested (according to the definition shown in Chapter 5). If the story happens to be structurally coherent, it is considered a complete one and it is yielded as a new story.

Randomly choosing preconditional rules to generate content could lead to the generation of the same story more than once per execution. This is avoided by including a set of *visited* stories containing all stories that have been generated so far. If the new story is already a member of this set, it is discarded and a new one is created.

7.3.2. Simple Surface Realization

Non experts cannot read opera plots rendered in the proposed formal representation without being first trained, which is too costly and leads to errors. However, the generated short operas are not internally represented in natural language (this is beyond the scope of this research). To bypass this limitation, simple surface realization based on trivial patterns has been used.

A mapping function receiving an individual fact and returning a string corresponding to its translation to a natural language representation has been created. The translation patterns themselves have been written by the author, who assigned the meaning to kernels. Following a pattern-based approach and slightly polishing the way in which entities and verbs are dumped into text strings, readable versions of the operas can be created. These are repetitive and simple, but they can be understood by external evaluators without any previous explanation.

To translate a whole opera, every action is first converted. Then the list of facts is joined keeping the same order in the short plot and separated by stops. For instance, Formal Story 7 would be rendered as Story 11 in English, and as shown in Formal Story 10 in Spanish[3].

ill(*violetta*)
love(*violetta, alfredo*)
together(*alfredo, violetta*)
forces(*germont, violetta*)
breakup(*violetta, alfredo*)
despise(*alfredo, violetta*)
forgive(*alfredo, violetta*)
die(*violetta*)

Formal Story 7: "La Traviata" in formal representation.

Violetta padecía una grave enfermedad. Violetta y Alfredo se querían con locura. Alfredo y Violetta eran novios. Germont forzó a Violetta a que hiciera lo que no quería. Violetta abandonó a Alfredo. Alfredo mostró su desprecio hacia Violetta. Alfredo, finalmente, perdonó a Violetta. Violetta murió, trágicamente.

Story 10: "La Traviata" after simple surface realization, in Spanish as originally generated.

This surface realization is admittedly very poor. Several systems performing surface realization for different languages already exist in literature [Elhadad and Robin, 1996, Gervás, 2007, Gatt and Reiter, 2009], and most of them can be used without problems. However, not being the focus of this research, proper surface realization is proposed as future work (Chapter 9).

[3]The surface realizer performs the translation in Spanish, this version has been translated to English by the author from the original version.

Violetta was terribly ill. Violetta and Alfredo loved each other passionately. Alfredo and Violetta lived together. Germont forced Violetta to do something that she did not want to do. Violetta left Alfredo. Alfredo showed his despise towards Violetta. Alfredo, finally, forgave Violetta. Violetta died, tragically.

Story 11: "La Traviata" after simple surface realization, after being translated to English by hand from the text shown in Story 10.

7.4. Adjusting Values for Parameters in the Implementation

The presented algorithms and models in Chapters 3, 5 and 6 are defined in terms of some parameters. In order to adjust the experiments and to allow a fast execution, these have been assigned by hand. Apart from the definition of the sets involved in the definition of the domain, which has been explained in Section 7.3, the parameters used to tweak rule acquisition are explained in this section. These parameters are introduced in 6.1.1.

The first parameter to be examined is n, which is the maximum number of allowed pattern actions in the precondition of the preconditional rules. This value ranges between 1 (a preconditional rule without precondition makes no sense) and the length of the longest story minus 1.

The execution of the experiment was carried out having set n to 2. According to the definition of the algorithms, the preconditional rules are searched first against all preconditions of size 1, then all preconditions of size 2, then those of size 3, and so on. Therefore, it means that setting a value higher than 2 would be useless if a valid candidate is found when the loop in the algorithm is creating rules with two preconditions. Even if n was set to 3 or higher, the result would have been the same. Other implementations in which the preconditional rules were generated as candidates following a different order would require a deeper study of the best approach to setting the value for this parameter.

The number of kernels is υ. This value for this parameter is directly computed by counting the number of kernels in the set π.

The number of possible patterns of variables per kernel that form the pattern actions for creating rules is p. The way in which the assignment of the variables is computed by the algorithm makes it automatically set. Since the number of tokens δ_{opera} is translated to a corresponding set of

variables and these are assigned according to the layout of token in the actions in the story, the value of p is fixed and, in this implementation, it cannot be set by hand (other approaches could have changed this). The value of p is, then, 1 in this experiment. This has severely reduced the cost of the search for preconditional rules, on the other hand it has restricted the number of different rules that could make it possible to generate coherent stories.

The v_p parameter represents the maximum number of actions that are examined backwards when looking for candidate pattern actions for creating a new preconditional rule. The value of this parameter does not affect the system as long as it is over a minimum. That is, if the system cannot find an appropriate action for evaluating a story as structurally coherent, it returns an error. As long as v_p is able to cover a minimum set of previous actions in the story so that this can be found, the value acceptable.

Since the algorithm always takes the first solution as the output, raising v_p to be higher than this minimum value is useless. As the algorithm for finding preconditional rules was improved in terms of efficiency and execution speed is not the main purpose of this experiment, v_p was set to be equal to the length of the longest formal opera. In this way, it is ensured that the parameter does not affect the results. It would have been possible to adjust it to be lower and the execution would have worked, but it was considered that finding this value was not necessary.

7.5. Execution Example

In order to clarify a real execution of the proposed validation process, an example of the learning procedure is shown. It tries to detail the experiment with the intention of showing through the example how the algorithm works, in this way completing the exposition of the tests.

The next commented execution log is the test carried out by a female, 28 years old, native Spanish speaker with no particular studies on narrative nor on Computer Science. The text, generated by the simple templates by computer in Spanish, have been translated to English by the author, it is assumed that the loss of meaning is not significant since the content is very similar. To make this assumption only the knowledge of Spanish and English of the author, the responsible for the translations, has been taken into account. The chosen corpus for this test is the one shown in Section 7.1.3, operas.

The experiment started by giving the user short instructions about what she was about to see and what she was supposed to do. These were orig-

inally written in Spanish because all the human evaluators are Spanish. The corresponding English translation can be read in Figure 7.1.

> You are about to be presented short story plots that you can probably recognize as opera plots. These are written in simple Spanish, and they have been generated by a computer. The facts in the story are ordered by time.
> This experiment tries to generate coherent stories. A story is coherent if it is completely understandable (i. e. everything happens for a logic reason), it finishes with a clear, logic ending, and everything in the story is relevant (i. e., there is no spurious content).
> Don not pay any attention to any other feature of the stories. It is not important whether it is interesting or not, only coherence is being evaluated.
> All you have to do is to answer *yes* is you think this story is coherent and *no* otherwise, according to the previous definition of coherence. The experiment will automatically finish after having evaluated some stories.

Figure 7.1: Instructions for carrying out the test.

The initial set of preconditional rules for coherence is extracted from the base corpus of operas by the execution of the algorithms shown in Chapter 6. Initially, the set of rules for non-coherent relations is empty because "non-coherent" human written stories are not used. This is due to the fact that non-coherent stories in the sense used in this research are not available, at least not in the same proportion as coherent stories.

This rule-set collected from the extraction of preconditional links (and then, rules) is shown next:

$$\begin{gathered}
breakup(x?, y?) \diamond back(x?) \\
breakup(x?, y?) \wedge together(y?, x?) \diamond back(x?) \\
back(x?) \wedge ill(x?) \diamond die(x?) \\
breakup(x?, y?) \wedge love(x?, y?) \diamond die(x?) \\
chase(x?, y?) \wedge love(x?, y?) \diamond die(x?) \\
chase(z?, y?) \wedge kill(y?, z?) \diamond die(x?) \\
despise(y?, x?) \wedge forgive(y?, x?) \diamond die(x?) \\
kill(x?, y?) \wedge love(x?, y?) \diamond die(x?) \\
kill(x?, z?) \wedge love(x?, y?) \diamond die(x?) \\
back(x?) \wedge together(x?, z?) \diamond die(y?) \\
root \diamond ill(x?) \\
escape(x?, y?) \diamond breakup(x?, y?) \\
forces(z?, x?) \diamond breakup(x?, y?)
\end{gathered}$$

$$jealous(y?, x?) \diamond breakup(x?, y?)$$
$$love(x?, y?) \diamond breakup(x?, y?)$$
$$want(y?, z?) \diamond breakup(y?, x?)$$
$$breakup(y?, x?) \diamond chase(x?, y?)$$
$$root \diamond chase(z?, y?)$$
$$escape(y?, x?) \wedge kill(y?, a?) \diamond chase(z?, y?)$$
$$root \diamond chase(z?, a?)$$
$$forces(x?, y?) \diamond chase(a?, x?)$$
$$ill(x?) \wedge breakup(x?, y?) \diamond despise(y?, x?)$$
$$help(y?, x?) \diamond escape(x?, y?)$$
$$love(x?, y?) \diamond escape(y?, x?)$$
$$kidnap(z?, y?) \diamond forces(x?, y?)$$
$$chase(z?, y?) \diamond forces(z?, x?)$$
$$together(y?, x?) \diamond forces(z?, x?)$$
$$love(x?, y?) \diamond forgive(y?, x?)$$
$$kill(x?, z?) \diamond help(y?, x?)$$
$$chase(z?, a?) \diamond forces(z?, x?)$$
$$forces(z?, x?) \diamond jealous(x?, y?)$$
$$love(y?, x?) \diamond jealous(y?, x?)$$
$$root \diamond kidnap(z?, y?)$$
$$chase(a?, x?) \wedge love(x?, y?) \diamond kill(z?, y?)$$
$$jealous(x?, y?) \diamond kill(x?, y?)$$
$$root \diamond kill(x?, z?)$$
$$forces(z?, x?) \diamond kill(x?, z?)$$
$$root \diamond kill(y?, z?)$$
$$root \diamond kill(y?, a?)$$
$$root \diamond love(x?, y?)$$
$$root \diamond love(y?, x?)$$
$$root \diamond together(x?, z?)$$
$$root \diamond together(y?, x?)$$
$$love(x?, y?) \diamond together(y?, x?)$$
$$root \diamond want(y?, z?)$$

After acquiring rules for the system against stories in Appendix A, the obtained set of rules (the process can be examined in Chapter 6) is used for the simple pseudo-random story generation, shown in Section 7.3. The set of obtained rules can be examined in the previous rule-set.

After this story was generated (it can be read in Formal Story 8) and flushed to the screen, the user was asked if she considered that the story was coherent. She answered, according to her interpretation, `y`, "yes". Although the exact meaning of the verbs was clearly explained, she under-

stood that Germont forced Violetta to act like jealousy, and then she died of grief. It was decided to let her own interpretation drive the experiments so that the system gathered and classified the acquired rules according to the user's criteria.

$$love(violetta, alfredo)$$
$$together(alfredo, violetta)$$
$$forces(germont, violetta)$$
$$jealous(violetta, alfredo)$$
$$kill(violetta, alfredo)$$
$$die(violetta)$$

Formal Story 8: First story generated in the execution of the test.

After the user had classified the story, the rule extraction algorithm is run again for this generated story and a new set of rules is obtained. Since the story has been marked as *coherent*, the new set of rules is merged with the one obtained when the experiment started because this original set only contains rules for coherent preconditional links. The new set of rules that will be added to the previous one is next:

$$kill(x?, y?) \diamond die(x?)$$
$$together(y?, x?) \diamond forces(z?, x?)$$
$$forces(z?, x?) \diamond jealous(x?, y?)$$
$$jealous(x?, y?) \diamond kill(x?, y?)$$
$$root \diamond love(x?, y?)$$
$$root \diamond together(y?, x?)$$

Redundant rules are not added to the set of learnt rules since it is useless. Therefore, only *new* rules, those which are not previously included in the set are inserted. In this case, the only new rule is:

$$kill(x?, y?) \diamond die(x?)$$

This rule is then merged with the previous set by performing a simple inclusion. Then, with the set of rules resulting from the merge, a new story different from the input set is generated. The generation system keeps a record of the stories generated so far, so that they will not be generated again. Although it is not explicitly shown in this example, there are cases in which the generated story only involved rules that have been already

$$ill(violetta)$$
$$love(alfredo, violetta)$$
$$jealous(alfredo, violetta)$$
$$breakup(violetta, alfredo)$$
$$despise(alfredo, violetta)$$

Formal Story 9: Story rated as non-coherent by the user during the tests.

learnt so the classification is carried out but no new content is added to any set of preconditional rules ("good" or "bad"). The generated story is shown in Formal Story 9.

The user, now, answers `n`, "no". She considers that the story lacks coherence, so some information must be extracted from the story so that the set of rules evolves. Therefore, the rule extraction algorithm is executed on this story, yielding the next set of rules:

$$root \diamond ill(x?)$$
$$jealous(y?, x?) \diamond breakup(x?, y?)$$
$$ill(x?) \wedge breakup(x?, y?) \diamond despise(y?, x?)$$
$$love(y?, x?) \diamond jealous(y?, x?)$$
$$ill(x?) \diamond love(y?, x?)$$

This set of rules represents what should not be generated. However, there are conflicting rules: some of them are present in the set of rules for coherent plots, so these are given preference. In this way, the only rule identified as creating non-coherent stories is:

$$ill(x?) \diamond love(y?, x?)$$

The experiment goes on by presenting more stories to the evaluator and storing her answers. In total, 22 stories were needed to reach a point in which stories were evaluated always as coherent: 16 stories rated as coherent by the user and 6 stories rated as non-coherent. When 5 stories were considered coherent, the experiment finished, considering that the system has acquired enough rules and most stories are correct.

The sequence of classfications is represented in Figure 7.2.

In order to represent more graphically the evolution of criteria about the generated stories, it is possible to represent the proportion between coherent and non-coherent stories for the last 6 generations.

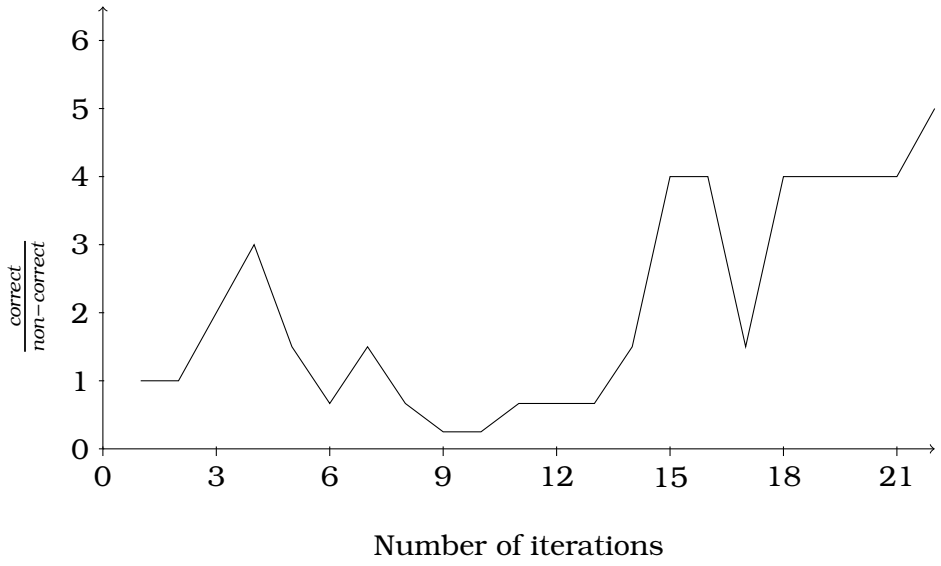


Figure 7.2: Learning curve from the execution example.

The final rule-set of preconditional rules for story processing can be read next. As the model introduces, two subsets of rules have been extracted: a set of rules for coherent relations, and a set of rules for non-coherent ones.

$$breakup(x?, y?) \diamond back(x?)$$
$$breakup(x?, y?) \wedge together(y?, x?) \diamond back(x?)$$
$$escape(x?, y?) \diamond back(y?)$$
$$back(x?) \wedge ill(x?) \diamond die(x?)$$
$$ill(x?) \wedge kill(x?, z?) \diamond die(x?)$$
$$breakup(x?, y?) \wedge love(x?, y?) \diamond die(x?)$$
$$chase(x?, y?) \wedge love(x?, y?) \diamond die(x?)$$
$$chase(z?, y?) \wedge kill(y?, z?) \diamond die(x?)$$
$$despise(y?, x?) \wedge forgive(y?, x?) \diamond die(x?)$$
$$kill(x?, y?) \diamond die(x?)$$
$$kill(x?, y?) \wedge love(x?, y?) \diamond die(x?)$$
$$kill(x?, y?) \wedge love(x?, z?) \diamond die(x?)$$
$$kill(x?, z?) \wedge kill(z?, y?) \diamond die(x?)$$
$$kill(x?, z?) \wedge love(x?, y?) \diamond die(x?)$$
$$kill(y?, z?) \diamond die(x?)$$
$$kill(z?, x?) \wedge love(x?, y?) \diamond die(x?)$$
$$root \diamond die(y?)$$

$$back(x?) \wedge together(x?, z?) \diamond die(y?)$$
$$breakup(y?, x?) \wedge love(x?, y?) \diamond die(y?)$$
$$escape(z?, y?) \wedge love(x?, y?) \diamond die(y?)$$
$$kill(y?, x?) \wedge together(x?, y?) \diamond die(y?)$$
$$want(z?, y?) \diamond die(y?)$$
$$root \diamond ill(x?)$$
$$die(x?) \wedge escape(x?, y?) \diamond breakup(x?, y?)$$
$$escape(x?, y?) \diamond breakup(x?, y?)$$
$$forces(z?, x?) \diamond breakup(x?, y?)$$
$$help(z?, y?) \diamond breakup(x?, y?)$$
$$jealous(y?, x?) \diamond breakup(x?, y?)$$
$$love(x?, y?) \diamond breakup(x?, y?)$$
$$forces(z?, x?) \wedge love(x?, y?) \diamond breakup(y?, x?)$$
$$forces(z?, y?) \diamond breakup(y?, x?)$$
$$forces(z?, y?) \wedge love(x?, y?) \diamond breakup(y?, x?)$$
$$want(y?, z?) \diamond breakup(y?, x?)$$
$$forces(x?, y?) \wedge want(x?, y?) \diamond breakup(y?, z?)$$
$$breakup(y?, x?) \diamond chase(x?, y?)$$
$$forces(y?, x?) \diamond chase(y?, z?)$$
$$root \diamond chase(z?, x?)$$
$$together(y?, z?) \diamond chase(z?, x?)$$
$$root \diamond chase(z?, y?)$$
$$despise(y?, z?) \wedge love(x?, y?) \diamond chase(z?, y?)$$
$$escape(y?, x?) \wedge kill(y?, w?) \diamond chase(z?, y?)$$
$$root \diamond chase(z?, w?)$$
$$forces(x?, y?) \diamond chase(w?, x?)$$
$$ill(x?) \wedge breakup(x?, y?) \diamond despise(y?, x?)$$
$$jealous(y?, x?) \diamond despise(y?, x?)$$
$$want(x?, y?) \diamond despise(y?, x?)$$
$$kill(y?, z?) \diamond despise(y?, z?)$$
$$root \diamond escape(x?, y?)$$
$$help(y?, x?) \diamond escape(x?, y?)$$
$$kill(x?, z?) \diamond escape(x?, y?)$$
$$love(x?, y?) \diamond escape(y?, x?)$$
$$kill(z?, x?) \diamond escape(z?, y?)$$
$$root \diamond forces(x?, y?)$$
$$help(z?, y?) \diamond forces(x?, y?)$$
$$kidnap(z?, y?) \diamond forces(x?, y?)$$
$$root \diamond forces(y?, x?)$$
$$chase(z?, x?) \diamond forces(z?, x?)$$
$$chase(z?, y?) \diamond forces(z?, x?)$$

$$chase(z?, w?) \diamond forces(z?, x?)$$
$$together(y?, x?) \diamond forces(z?, x?)$$
$$root \diamond forces(z?, y?)$$
$$die(y?) \diamond forces(z?, y?)$$
$$want(z?, y?) \diamond forces(z?, y?)$$
$$love(x?, y?) \diamond forgive(y?, x?)$$
$$root \diamond help(x?, y?)$$
$$kill(x?, z?) \diamond help(y?, x?)$$
$$love(x?, y?) \diamond help(y?, x?)$$
$$love(z?, y?) \diamond help(z?, y?)$$
$$want(z?, y?) \diamond help(z?, y?)$$
$$forces(z?, x?) \diamond jealous(x?, y?)$$
$$root \diamond jealous(y?, x?)$$
$$forces(z?, y?) \diamond jealous(y?, x?)$$
$$love(y?, x?) \diamond jealous(y?, x?)$$
$$kill(z?, x?) \diamond kidnap(x?, y?)$$
$$root \diamond kidnap(z?, y?)$$
$$root \diamond kill(x?, y?)$$
$$jealous(x?, y?) \diamond kill(x?, y?)$$
$$root \diamond kill(x?, z?)$$
$$forces(z?, x?) \diamond kill(x?, z?)$$
$$love(y?, z?) \diamond kill(x?, z?)$$
$$jealous(y?, x?) \diamond kill(y?, x?)$$
$$root \diamond kill(y?, z?)$$
$$help(y?, x?) \diamond kill(y?, z?)$$
$$want(z?, x?) \diamond kill(y?, z?)$$
$$root \diamond kill(y?, w?)$$
$$root \diamond kill(z?, x?)$$
$$chase(z?, x?) \diamond kill(z?, x?)$$
$$despise(y?, x?) \diamond kill(z?, x?)$$
$$forces(z?, x?) \wedge love(x?, y?) \diamond kill(z?, y?)$$
$$root \diamond love(x?, y?)$$
$$chase(y?, z?) \diamond love(x?, z?)$$
$$root \diamond love(y?, x?)$$
$$despise(y?, x?) \diamond love(y?, z?)$$
$$die(y?) \diamond love(z?, y?)$$
$$root \diamond together(x?, y?)$$
$$escape(x?, y?) \wedge love(x?, y?) \diamond together(x?, y?)$$
$$root \diamond together(x?, z?)$$
$$root \diamond together(y?, x?)$$
$$love(x?, y?) \diamond together(y?, x?)$$

$$root \diamond together(y?, z?)$$
$$back(y?) \wedge love(x?, y?) \diamond together(z?, y?)$$
$$root \diamond want(x?, y?)$$
$$forces(x?, y?) \wedge kidnap(x?, y?) \diamond want(x?, y?)$$
$$root \diamond want(y?, z?)$$
$$root \diamond want(z?, x?)$$
$$root \diamond want(z?, y?)$$
$$help(x?, y?) \diamond want(z?, y?)$$

Next rule-set contains all the acquired non-coherent rules during the execution of the experiment.

$$breakup(x?, y?) \wedge love(x?, y?) \diamond die(x?)$$
$$breakup(z?, x?) \wedge love(x?, y?) \diamond die(x?)$$
$$despise(x?, z?) \wedge forces(z?, x?) \diamond die(x?)$$
$$despise(y?, x?) \wedge forgive(y?, x?) \diamond die(x?)$$
$$root \diamond ill(x?)$$
$$escape(x?, y?) \diamond breakup(x?, y?)$$
$$forces(z?, x?) \diamond breakup(x?, y?)$$
$$despise(y?, x?) \wedge love(x?, y?) \diamond breakup(y?, x?)$$
$$help(z?, y?) \diamond breakup(z?, x?)$$
$$love(x?, y?) \diamond despise(x?, z?)$$
$$ill(x?) \wedge breakup(x?, y?) \diamond despise(y?, x?)$$
$$jealous(y?, x?) \diamond despise(y?, x?)$$
$$help(y?, x?) \diamond escape(x?, y?)$$
$$ill(x?) \wedge kill(z?, y?) \diamond forces(z?, x?)$$
$$together(y?, x?) \diamond forces(z?, x?)$$
$$love(x?, y?) \diamond forgive(y?, x?)$$
$$kill(y?, z?) \diamond help(y?, x?)$$
$$want(z?, y?) \diamond help(z?, y?)$$
$$root \diamond jealous(y?, x?)$$
$$root \diamond kill(y?, z?)$$
$$want(z?, x?) \diamond kill(z?, y?)$$
$$root \diamond love(x?, y?)$$
$$ill(x?) \diamond love(y?, x?)$$
$$love(x?, y?) \diamond together(y?, x?)$$
$$together(y?, x?) \diamond want(z?, x?)$$
$$root \diamond want(z?, y?)$$

However, these rules are not definitive since several rules included in this set conflict with rules in the set of rules for coherent stories. These

are removed from the non-coherent set and the next one is obtained as the definitive set.

$$breakup(z?, x?) \wedge love(x?, y?) \diamond die(x?)$$
$$despise(x?, z?) \wedge forces(z?, x?) \diamond die(x?)$$
$$despise(y?, x?) \wedge love(x?, y?) \diamond breakup(y?, x?)$$
$$help(z?, y?) \diamond breakup(z?, x?)$$
$$love(x?, y?) \diamond despise(x?, z?)$$
$$ill(x?) \wedge kill(z?, y?) \diamond forces(z?, x?)$$
$$kill(y?, z?) \diamond help(y?, x?)$$
$$want(z?, x?) \diamond kill(z?, y?)$$
$$ill(x?) \diamond love(y?, x?)$$
$$together(y?, x?) \diamond want(z?, x?)$$

The whole experiment took approximately 8 minutes and a half. This means an average time of 25 seconds per story, approximately. Although the creation of the rules by hand from the same domain has not been carried out, it is claimed that this pseudo-automatic process is much faster, based on experience. Therefore, the solution is promising according to its objectives. At least, for simplistic domains as the one that has been tested. It is planned to study more complex domains as part of the future work.

7.5.1. Saturation

A limit for the experiment has been imposed: once the user reaches a point where most stories are rated as coherent, the experiment stops. It can be seen in the previous section that this limit has been set to 5. This is because during the early tests a *saturation level* was discovered.

After some classifications, the current procedure is unable to extract more rules in a robust way. While it is possible to find more rules, this does not happen as fast as in the first stage (*non-saturation*) of the execution. It was empirically found that once 5 or 6 stories in a row are rated as coherent, the probability of being in the saturation stage is quite high, so it was decided to stop the experiment once this happens.

Saturation happens because of the experimentation system itself, the abstract model for gathering rules is not creating the saturation. The way in which modifications are made to generated stories to create new ones has a limit. In order to keep resemblance with the stories that can be generated by only applying rules, some constraints are cutting too many possibilities. The amount of different patterns it can create is small, therefore eventually exhausting this set of patterns. The proposals for future

work, in Chapter 9, examine other possible approaches to story generation in order to overcome this limitation.

7.6. Overall Results

In order to evidence that useful domain information can be extracted in general, and not only in one single case, the test explained in the previous section was repeated for other subjects. In total, the test was run for 14 people (including the subject from the previous example). Their age ranged between 25 and 60 years old, and all of them are native Spanish speakers, so the surface realization for stories was only implemented for Spanish. It is considered that the messages are so simple that the particular natural language is not affecting the identification of coherence, although discussion is pertinent about this matter (Section 8.1.6).

The test was run once per each human evaluator. Each experiment finished when the saturation zone was reached. The experiment was stopped at that point and the evaluator was not asked to run the test again. Therefore, 14 experiments have been executed. The statistical data graphically shown in 7.3 is acquired from that number of experiments, then.

Elapsed time to reach saturation was also measured. If acquiring rules pseudo-automatically for some domain is slower than doing it by hand, the use of the solution becomes discussable. The mean elapsed time was 8.78 minutes, with a standard deviation of 2.752. The fastest classification took 4.224 minutes and the slowest one took 13.407 minutes.

The average time per story was 26.20 seconds with a standard deviation of 7.02.

Regarding the comparison against crafting the rules by hand, it has to be taken into account that the proposed system is able to generate a set of rules from some input corpus without human intervention, if it is desired. While it has been shown how this approach would be incomplete, it is clear that the benefit in terms of time is noticeable.

Figure 7.3 shows the average structural coherence refinement process from all the evaluators. It can be seen how the proportion between coherent and non-coherent stories is almost lineally raised during the execution of the tests.

It is assumed that the content of the stories is affecting the execution, that is, not every type of plot or any input set would yield these results. The reasons for this are explained in this chapter, mainly in Section 7.1. While the results are promising and the empirical results shows that pseudo-automatically acquiring rules is possible, the application of this approach

Figure 7.3: Average proportion of coherent vs. non-coherent stories during the refinement process. The relation between the last 5 stories is shown.

to other domains (mainly more complex ones) is planned as part of the future work.

For instance, if the tests are run using just one story as corpus, the learning process will be much faster, as it can be seen in Figure 7.4. While it is just an example and, depending on the evaluator and on the input story it does not have to happen in all cases, the learning process is quite straightforward because all the generated stories are coherent according to human criteria. This is just because acquiring rules from one single story leads to a rule-set that only creates stories very similar or even only equal to the source story (which is coherent a priori). This is one of the cases in which this approach is not really useful: more stories are needed.

7.7. Chapter Summary

This chapter details the implementation of the theoretical model described in the previous chapters. Additionally, it shows the experiments that have been run with that implementation to prove, to the possible extent, that the main hypotheses are plausible. Graphical depiction of some executions are shown as well.

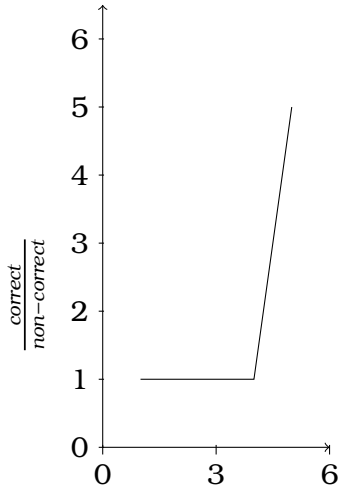


Figure 7.4: Example result of the rule acquisition process when using one single story as input corpus.

Chapter 8

Discussion

THIS CHAPTER DISCUSSES ALL THOSE DECISIONS that have been taken during the development of this research, trying to show the justification of having chosen them. Given the characteristics of the current research, some assumptions are subject to discussion because they are outside the pure formal or computational domain. While some of them are hard or impossible to demonstrate with absolute certainty, this part of the dissertation tries to show that, at least to some extent, the taken approach is useful according to the current focus of the development.

This chapter have been thematically organized, instead of keeping the order followed in the previous chapters. It has been considered that a chapter devoted to discussion is better understandable with such a layout.

8.1. Conceptual Aspects of the Current Approach

This research is based on an adoption of a structural focus to computational narrative processing. Therefore, the extent to which this is really useful and plausible from a conceptual point of view must be discussed.

The main assumption in this research is that empirical results are valid, and the fact that the modelled method does not mimic human psychological processes is not influential. As introduced in Chapter 1, the reduced knowledge available about underlying human mental processes on narrative makes it very difficult to implement systems that *really* replicate humans.

8.1.1. Number of Used Stories in the Rule-Extraction Process

Algorithm 3 only learns from one single story. While results show useful outputs (Chapter 7), the system could learn content from more than one story. This would be beneficial in principle because there is implicit content in the *set* of stories that could be extracted, not only in the stories individually. For instance, the system could extract a rule r_a from a story s_a and a rule r_b from the story s_b. Taken independently, r_a and r_b could be useful, but they could be incompatible together.

Since the rule extraction algorithm must return a single set of rules, these must be coherent as a set, and not only independently. So far, the rule extraction process only ensures coherence for single stories.

While developing the modification of the algorithm to make it able to process more than one story, the search space became much bigger. Because of the exponential nature of the solution based on search, adding more candidates made the problem intractable again. Additionally, extracting rules from the whole corpus at once turned the system into a non-scalable one, because adding more stories to the corpus would make the search space grow exponentially instead of growing lineally. This is why it was decided to process one story at a time. However, apart from the execution cost, nothing prevents from analysing more than one story.

8.1.2. Impact of the Input Corpus in the Final Results

The chosen set of input stories affects the overall results of the experiments shown in Chapter 7 because they are the base for the generated stories. Different corpora would create different stories with different characteristics. While it is assumed that the resulting rules will depend on the input content, it is important to discuss the way in which the *quality* of the results is affected by the process of creation of the corpus.

In the particular implementation that has been shown, operas have been summarized and translated to a formal version by hand, which has introduced the author's criteria in the execution. It makes sense then to question if the author's knowledge has influenced the results.

According to this, the summarization and translation of human-written stories to formal representation processable by the algorithm is considered to be one of the main drawbacks detected in the development of this research. In this stage, author's knowledge and intuition have admittedly influenced the whole process. In the instantiation of the model for operas, the real *libretto* is very different from the outline, and even the outline of the

events taking place in the opera are inherently different from their formal version.

While other summaries for the operas and other possible formalizations would have been possible and they would have possibly led to different values, it is claimed that, since the influence of the author's knowledge takes place in the first stage only, the quality of the results is not really affected because it is driven by external humans' opinion. That is, if the stories coming from the author's translation were totally non-coherent, the rules acquired by the algorithm would create non-coherent stories as well. On the other hand, if coherence in the input stories led to coherent results, all generated stories in the rule-gathering process would lead to acceptable stories *always*, which is not the case, as it has been shown in Chapter 7.

This does not mean that this translation is the best method, though. The option of translating the stories by hand was chosen to reduce the needed effort to get the system running. Training other humans to create short summaries of operas and then having them formalize this outlines to the required formal representation was not always possible, specially when the formal model and the computational requirements of the implementation have been updated several times during the development. If other individuals had created the processable versions of the stories, they would have been forced to do it several times and with several formalisms.

External influence on translations, or even pseudo-automatic translation, which is hypothetically true, would be, in the opinion of the author, better options for future versions of the proposed system.

8.1.3. Influence of Aspects Different from Coherence in Classification

As shown in Section 7.5, humans were required to focus on narrative coherence when rating stories in the experiments. This explicit restriction was clearly stated in the questionnaire to reduce the influence of other aspects of stories on the rating. Experience from the work presented in Chapter 4 showed that the opinion about the overall quality that readers perceive when they read a story influences the rating of particular variables. That is, if a reader finds the story funny or interesting, she or he will consider that the story is good, therefore it is possible to find coherence in a story that is actually lacking good coherence.

However, it is difficult to measure the extent to which this has been correctly reached. Since psychological aspects are unavoidably involved when measuring coherence, and human psychological processes for iden-

tification of coherence are not completely known, no formal study about this is possible.

In order to partially address this issue, the model assumes that validation against human criteria is the best way to evaluation. That is, if human evaluators rate a story as coherent, the rules are correct. This decision is a consequence of the structural definition of this work, which tries to avoid cognitive models. If the stories are not coherent from a narratological or psychological point of view but evaluators rate it as coherent, it is valid for the system.

8.1.4. Influence of Human *Opinion* in the Preconditional Rule Extraction Process

Establishing a general definition of coherence in stories valid for every possible story is not possible because particular human opinion is involved in the process. There is no way of formalizing human opinion and therefore it does not make sense to give more importance to one opinion over another in all cases. Therefore, the particular set of human evaluators driving the rule gathering process explained in Section 6.3 affects the final set of acquired rules. That is, the final set can be non-valid for other humans.

The results collected in Section 7.6 suggest that the evaluation for coherence is reasonably uniform for a simple domain under somewhat restricted conditions, but this does not have to be the case in all scenarios. It is hypothesized that more complex domains in which the evaluation of coherence in stories will not be uniform can be found. The study of this hypothesis and its consequences is targeted as future work.

8.1.5. Good and Bad Stories

Narrations have been divided in the model in two partitions: *good* and *bad*. That is, according to the model, two good stories are not comparable to each other to test which one is better, they are just *good*. An analogous method is followed for bad stories.

From a cognitive point of view, this is far from complete. Not only stories can be “good”, “normal” or “very good”, for instance, but also “good” stories can be considered “bad” as well if several human judgements are taken into account.

This common aspect of evaluation is only partially addressed as a possible side effect of the classifications obtained from different human evaluators, but the theoretical model does not take this into account. This is

a limitation of the model, which should be improved in future versions to take this into account.

On a different tack, the set of "good" stories was defined in terms of coherence: structural coherence from the point of view of the computational system, and coherence as understood by humans in the experiments. It is accepted that *quality* is not totally covered by coherence, and both short and complex plots, even on its high level representation (leaving out literary aspects), are rated differently about its quality taking into account many more aspects that only coherence. While it is assumed that coherence is a basic starting point for describing quality in narrations, the model accepts this restriction in order to offer a working solution, instead of a broader but incomplete one. Whether coherence is one of parts of basic quality is discussable as well, but that discussion is considered to be outside the scope of this dissertation.

8.1.6. Language for Text Realization

Experiments have been carried out by presenting the stories in natural language to human evaluators. As all the evaluators where native Spanish speakers and this was known before implementing the text realization, simple patterns, the experiments were therefore carried out with texts in Spanish.

The quality of the texts is low, but the patterns and their joint as a full story is grammatically correct. No special check about this correctness was carried out because basic knowledge of Spanish was enough to ensure it.

Both aspects could be discussed regarding text realization: the influence that the particular way of telling the plots exerts on evaluators, and the way in which the particular language (Spanish, English or whatever other language) affects human criteria.

Regarding the first aspect, the influence of the quality of the texts, it is assumed that the patterns are so simple that no real influence is being exerted. As a loose test, it was checked during the execution of the tests that evaluator's criteria were very similar to the author's criteria for every story, with only slight differences. This suggests that the text patterns created by the author, which had defined the meaning for the kernels of the actions, expressed what he meant, and that the evaluator was approximately interpreting the same thing. This is obviously not a formal check of the assumption, but text realization is not an objective of this work and it is therefore accepted that the results are valid.

The example of realized story shown in Formal Story 11 shows the text in English, and the one shown in Formal Story 10 shows the Spanish

version. It is hard to evaluate their similarities without speaking both languages, but these examples try to show an evidence (for those who speak English and Spanish) that the meaning of the realized actions is very similar in both version.

8.2. Comparison Against Other Approaches

The proposed method is a very particular approach to story generation. There are, as it has been shown in Chapter 2, many other computational approaches to narrative. This section compares and discusses those points in which the comparison is interesting.

8.2.1. Classic Machine Learning and Rule-Gathering Algorithm

To be able to gather rules, the algorithm needs, as input, a set of stories for use it as a "training corpus". This corpus, however, must satisfy some constraints. Any set of texts put together is not enough to make the system work. The approach is based on narrative structural knowledge that short, simple narrations contain, not on statistical properties of elements.

Most machine learning algorithms can learn most of the times no matter the quality of the corpus, the accuracy of the learning depends on the quality of the corpus. However, the proposed algorithm lacks this ability: if the input stories are not processable by the evaluation function, nothing will be learned from them.

On the other hand, the results from this learning process are totally usable after the acquisition: the set of rules is perfectly readable and modifiable by humans. Additionally, certain approaches to Machine Learning have already developed useful techniques for some aspects related with the current work, like, for instance, acquisition of time relations [Mani et al., 2006, Bramsen et al., 2006].

The other approach considered worth the comparison is the acquisition of causal rules from an external knowledge base. The overall benefit is clear: there is no need of creating a knowledge base, and this saves time and effort. However, possible options like OpenCyc [CyCorp, 2010] or ConceptNet [Liu and Singh, 2004] have several characteristics which make it difficult to apply the current approach to evaluation of narrative content, namely:

- Nothing ensures that information about the working domain is avail-

able on these databases. While they capture (in different ways) common sense knowledge, some structure may well not be present.

- Causality is considered, but not as much developed as in the proposed learning algorithm. Full focus on causality is put in the presented system (because it is the structure on which the definition of preconditional link relies), but acquiring robust link information from these knowledge bases is not simple because that information is not as abundant as some other type of relations (as existential properties, for instance).

- Multiple links between concepts are hard to find. Cases in which some fact is the result of the *combination* of two different facts are seldom found, and that construct is quite frequent, at least, in the domain used for the implemented prototype.

- Improving the knowledge base for adapting it to new circumstances (new domains) requires, in the case in which the domain knowledge is not present, hand-writing rules, which implies the problems previously analysed.

It is finally concluded that the option of learning the rules is the most profitable one. Its cost is quite low compared to the option of hand-writing rules, and it provides a more robust coverage inside concrete domains. It comes at the cost of developing the learning algorithm (which is not low), but once this has been created, the adaptation to new domains is, in principle, straightforward.

But that approach is not perfect, since the results are not optimal and it is quite sensible to errors. Putting out human supervision carries the risk of letting the system infer wrong rules, although this supervision can be obviously done *post hoc*. Another benefit of this approach is that any improvement on the learning algorithm can directly benefit all working domains. The current approach to learning is not the optimal one, but it can probably be improved for future versions of the evaluation function.

8.3. Discussion related to the Implementation

The implementation of the theoretical model presented in chapters 3, 5 and 6, as detailed in Chapter 7, has involved additional assumptions and restrictions that have realised the algorithms in a real program. How these adaptations have affected the collected results is discussed in this section.

8.3.1. Appropriate Number of Input Stories for the Corpus

For carrying out the presented results, 9 adaptations of operas have been used. As discussed in Section 8.1.2, it is assumed that the way in which the input corpus is used does only partially affect the results because human supervision is what really drives the rule acquisition process.

However, the amount of input stories affects not only the particular set of rules that is extracted after the execution of the process, but also the time needed to reach the saturation point. If only *one* story was used and this story was coherent, the acquired set of rules used to generate stories would lead to only one single structure of narrations, very similar to the original story. Correcting this set of rules through classification of generated stories would only require to check a few examples (the exact amount would depend on the length of the story). So the amount of different stories that can be created is directly proportional to the set of preconditional rules, which is directly proportional to the number of input stories in the corpus, *a priori*.

This does not have to happen always because the actions in the stories are very influential in the rules. In the most extreme case, the system could be fed with many copies of the same story, this would be exactly the same as feeding it with one instance of that story. In the opposite case each story has a set of actions totally different from the actions in the rest of input stories. This would lead to many rules, but only one rule for each action, which would make the iterative process of refining the set of rules quite slow.

As it has been intentionally carried out in the experimentation, the pseudo "optimal" case takes place when the availability of several stories in which the set of actions is not the same, but every actions appears in several stories with different previous actions. This leads to several rules for each action, which helps to create instances and to iteratively create a useful set of "good" and "bad" preconditional rules.

8.3.2. Application of the Structural Information Extraction Algorithm to Other Domains

Results obtained after this study prove that the gathering of structural relations in simple stories can be pseudo-automatically acquired. This is considered an important merit of the research, but further studies must be carried out in order to prove that the proposed method is valid for other domains.

It is well known, as it has been presented in Chapter 7, that the concrete set of actions in the input story and the proportion in which they appear against the amount of input stories in the corpus determine the final quality of the rule-refinement process, which affects itself the generative capabilities.

On the one hand, the imposed restrictions about what a *simple* story is reduces the scope of the approach. That is, these structural approaches are not necessarily valid for complex narrative forms like novels, which are not addressed. Although the study of the inherent differences between complex narrations and simple stories has not been carried out, many assumptions that have been made for this work to be feasible are not satisfied in complex narratives, so the proposed algorithm is quite unlikely to work in those cases.

On the other hand, short versions of opera plots are not the only possible input domain. While reduced operas are useful for the particularities explained in Section 7.1.3, many other domains fulfill the same requirements, so they could be used as well, and nothing prevents, in principle, a successful execution.

The main point, then, is that the definition of "simple story" is quite restrictive, but that many forms of narrative fall in this category. Fables (if a good subset of them could be found), tales or short descriptions of travels. Although future work must address this to prove it, it is hypothesized in advance that it makes sense to keep on studying the structural analysis of narrative as a useful computational method.

8.4. Chapter Summary

This chapter discussed the most interesting and conflicting points of the research. Both the strong parts and the weaknesses of the current approach are addressed, and future lines of discussion, outside the scope of this thesis, are proposed.

Chapter 9

Conclusions and Future Work

THIS LAST CHAPTER OF THE DISSERTATION summarizes the presented work and proposes different possible further ways in which the explained research could be improved.

While there is still much to be done in this field (structural approaches to computational narrative), this dissertation has tried to create a plausible and useful proposal that can be used, if not for solving problems in the industry, at least for triggering new research in these terms.

9.1. Summary

All along the previous chapters, these have been the subjects that have been studied:

- The objectives and hypotheses that motivated this work were presented in the first chapter.
- The previous research setting the base for this research was studied in detail in Chapter 2.
- The formal and conceptual definition of *narration* used in this research has been defined in Chapter 3.
- The preliminary work on a computational system for processing narrations based on a semantic approach is then detailed.
- A structural approach to computational processing of narrative has been modelled and described in Chapters 5 and 6.
- Then, the implementation, experiments and obtained results are shown in Chapter 7.

- Finally, the most relevant points are discussed and summarized.

All this dissertation has been written trying to focus on what were considered the most important ideas and contributions of the research, without leaving out all the techniques, algorithms and references that base the work.

9.1.1. Scientific Contributions

The main scientific contributions are summarized in this list:

- A formalism for representing narrations, and a definition of *simple narration* that is adopted in this work. This is described in Chapter 3.
- The development of a knowledge intensive system in which an explicit evaluation function drives a generative process, and its partial validation through human supervision (Chapter 4).
- The definition of an artificial relation used as schematic unit to perform schema acquisition and rule learning, the *preconditional link* (Chapter 5).
- A set of algorithms for evaluating a story in terms of its structural properties and for acquiring instances of schemas and rules (Chapters 5 and 6).
- A framework for human-supervised acquisition of generation rules and narrative generation (Chapter 6).

9.2. Conclusions of this Research

This section tries to resume the conclusions that have been gathered after the obtained results were analysed. The most important part of these conclusions is the validation –or rejection– of the main hypothesis according to the results of the empirical tests.

The hypotheses, defined in Section 1.3, state that a structural approach to narrative not taking into account cognitive models can be implemented for some domains in such a way that stories recognizable as such by humans are generated. It was also hypothesized that the defined structural properties could be pseudo-automatically acquired.

After the analysis of the results shown in Chapter 7 it can be concluded that the hypotheses have been *partially* validated. Through the implementation and the tests it has been shown how *it is possible* to build a system performing structural processing on narrations that creates stories that are considered *coherent* by humans.

The other part of the model, the proposed rule-acquisition algorithm, has been implemented as well and tested as part of the experiments, so the second part of the hypotheses, the plausibility of acquiring rules pseudo-automatically (in the proposed way) has also been *partially* demonstrated.

A special emphasis is put on the fact that the hypotheses have only been *partially* proven. This is considered so because:

- The experiments involve human criteria. This means that the evaluation of coherence is only partial, and that many more human evaluators could be used to perform a more specific study. However, due to the prototype characteristics of the implementation, this was considered unnecessary at this point.

- The inherent focus of this thesis is related to narrative, which is a very complex concept which does not have a fully accepted definition. Thus, stating that the stories *are coherent*, while plausible, cannot be formally proven.

Although only partial validation has been carried out in this research, the overall conclusion is that the hypotheses, to the possible extent, are plausible enough to keep on researching on the field of structural processing of narrations in these terms.

9.3. Benefits and Drawbacks

No research project is perfect, and this one is not an exception. This sections briefly summarizes the main benefits and drawbacks that have been presented along the exposition of the previous chapters.

It has been concluded that the current contribution reduces, to some extent, the human effort required to improve the domain of generation of narratives for an application. While this is considered an obvious benefit, it must also be made clear that the proposed system, so far, is only applicable to simple domains and short and plain narrations. While more work could lead to an improved model providing coverage to more sophisticated forms of stories, this has not been carried out yet.

The supervision system is very simple and based on a straightforward boolean classification. It is claimed that this is a clear benefit because it is very easy for humans to classify content in this way. Additionally, such a simple boolean system does not necessarily have to be explicit, it is hypothesized that this can be queried by other means (although this study is outside the scope of this research). But this simple classification comes at a price: only a basic conception of coherence can be gathered using this method. Interest, for example, is not boolean, there are virtually infinite levels of interest.

Theoretically, the proposed model covers the full space of solutions for all the proposed algorithms. This permits a theoretical full coverage of the solutions, which is good. However, this process has not been feasible in practice because the spaces are so big that traversing them completely is not a tractable problem. Therefore, the theoretical advantage is hindered by the practical limitations of current computational power.

Finally, it is considered that an important part of the contribution is the proposal of a totally structural paradigm for computationally management of narrative content. It has been shown how this can be positive with an original model. But it is hypothesized that, while this could open new paths for research, it is quite likely that only structural processing will not be enough by itself, and semantic computation will be required to achieve really good quality, at least when comparing the stories generated by computers to those generated by humans.

9.4. Future Work

The previous chapters have defined the bound of this research, leaving several points as part of the future work. This section summarizes those improvements on the system or related research lines that are being currently considered. While there could be many others, these are directly related and planned as the next issues to study.

9.4.1. Improving the Model for Structural Definition and Processing of Narrations

It has been empirically shown that structural processing of stories makes sense, at least for some domains. However, the application of the proposed approach is not limited to the presented system, and it is hypothesized that more sophisticated and powerful solutions can be developed.

For instance, the classification of the stories could be improved so that the evaluation function and the human criteria are not boolean, but range on some real interval. In this way, plain separation between *coherent* and *non-coherent* stories could be improved and a *ranking* of stories could be performed. This would lead to the concept of a story which is "more coherent" than some other. This was the approach followed in the cognitive system developed and presented in Chapter 4.

Figure 6.1 shows a diagram of the proposed preconditional rule acquisition method. While it has been proven that it makes sense to build rule-sets in this way, some improvements can be applied to the algorithm so that the rule gathering approach is made more powerful. For instance, the first accepted candidate set of rules is taken as the solution. This is not necessarily optimal, and more options could be considered. For instance, it would be possible to select the *best* candidate based on a non-boolean classification of story according to their coherence.

The model could be also improved by adding more complex time relations to the definition of *simple stories*. The assumptions that were made are considered to be extremely restrictive. This was done to keep the presentation of the model simple and focused, but it is hypothesized that the model in its current form can be applied to stories in which the duration of each action is higher than a single unit of time, for instance. More complex time relations would broaden the range of application of the model.

In general, this expansion of the range of stories for which the structural model is able to give a correct solution must be improved as the next step in this research for the system to be really useful. It is assumed that humans handle stories that are much more complex than those presented in this dissertation, and reaching, at least partially, that expressive power is a very important –and, perhaps, ambitious– objective that, in the opinion of the author, should be targeted.

9.4.2. Improved Story Generation

The rule-acquisition method is based on creating candidate stories that are classified under supervision. The quality of the story generation system affects both the required time to reach the saturation point in the experiments and the capabilities of finding new rules.

As it has been designed, the acquisition process can only collect the preconditional rules that are implicitly present in the input stories because no new possibilities are included. The saturation point is reached when there is nothing new to "learn", that is, a local maximum regarding the preconditional rules has been hit. In order to avoid this, *noise* could be

introduced in the model.

That is, to be able to gather new preconditional rules, relatively original stories, slightly different from those which can be generated with the rules from the original stories, could be created. In this way, the rule-acquisition process could expand its capabilities by changing the way in which stories are built.

This opens new issues to solve, as for instance the amount of noise to add to the generated stories. Too much noise will generate an excessive amount of non-coherent stories, and too little noise will not be useful at all.

It would be interesting as well to use more sophisticated Natural Language Generation systems for text realization. The one that has been shown here is too poor, and the human criteria about stories is probably being affected by the realization and not only the underlying content. This text realization is, however, supposed to happen at a different level than the rest of the system. That is, improving the text realization, while important, should not influence the rest of the algorithm, at least in principle.

9.4.3. Improving the Implementation

The implementation of the theoretical model has been carried out for testing purposes, and it has not been adapted for use beyond this. To make it possible further research of the model, and given that all the demonstration of the validity of such a kind of model is carried out empirically, it makes sense to create a version of the implementation that can be used in a more robust way.

On the one hand, the implementation could be executed faster. During the middle stages of the development of the prototype, parallelization of the code was taken into account. While it was finally discarded in order to keep a fast development process, due to the type of algorithms, parallelization can make it possible to explore bigger subsets when searching candidate preconditional rules and links and when performing story generation.

On the other hand, the development of such a model is intended to be useful to the research community, so modifying the prototype following a software design approach that makes it possible to implement and release the system as a library –or whatever other form of distributable and usable system– makes sense.

9.5. Applications of the Structural Processing of Narratives

The overall objective of this research problem is not only the development of a theoretical model for managing narrative texts, but also the application of the results to real applications. While the current version of the implementation is still far from being applicable in a real scenario, it is considered that the ideas, once built together in a working system, could improve or ease certain human tasks.

The first application that could benefit from the use of this model is plain story generation. While, as it has been shown, the solution is yet unable to produce high-quality stories, nothing prevents from applying the concepts developed in this research to create a simple story generator. This hypothetical generation system would probably reach a higher quality of generation by creating a mixed approach in which both structural and cognitive methods are used.

Not only classic story generation could be the objective of this model. One of the main benefits of the proposed algorithms is that rule acquisition can be carried out pseudo-automatically, which leverages the possible range of applications to any scenario in which human criteria can be gathered. Since the rating that are required for the system to work are boolean, one can imagine several situation in which this is easily doable.

For instance, in narrative videogames, the player could play an automatically generated story and then rate it as coherent or non-coherent. Then, rules could be extracted from this rated story, and they could therefore, as the model proposes, be included in supervised rule-sets. This would provoke an iteratively refined generation system which would lead to the generation of coherent stories without any additional human intervention in principle.

The synthetic proposed implemented scenario for demonstrating that the preconditional rule acquisition is possible does not represent well real scenarios in general. While a commercial system could require preliminary training of the generation process, a more user-friendly system would perform this rule extraction in a more indirect way.

For instance, in an even more sophisticated approach, the opinion about coherence could be extracted in a more automatic way using non-explicit methods. Instead, the way in which the player plays the game could be analysed. For a trivial solution, non-coherence stories could be directly rated as such if the player stops playing it without having finished the story, for instance. This is just a hypothetical scenario and no real

solution is being proposed, but it is considered that this kind of systems could be built as an application of this model. Of course, rating coherence in this way would be polluted with the implicit rating of many other aspects as interest or engagement. This would have to be addressed in the creation of a system like this one.

Many other uses of this research could be found, and this section has only intended to propose a brief perspective of how the structural processing of narrative can be applied. While some of this applications are in general solvable by any approach to story generation or understanding, doing it with algorithms based on structure can offer new things that, at least partially, can make the solution of these long term problems more reachable.

9.6. Chapter Summary

This chapter concludes the thesis by examining the main outcome, and proposes further lines of investigation. Both the benefits and the drawbacks of the research process are highlighted, and the extent to which the hypotheses suggested in the first chapter have been demonstrated is discussed.

Parte II

Resumen de la tesis en español

Capítulo 10

Introducción

El tratamiento semántico de los textos narrativos trata de alcanzar, desde uno u otro punto de vista, la interpretación de cómo los humanos procesan historias. En general, las aproximaciones semánticas han predominado en los sistemas de generación [Meehan, 1976, Turner, 1992, Bringsjord and Ferrucci, 1999, Riedl, 2004]. Mientras que nada evita que la investigación en este tipo de sistemas pueda llegar a tener éxito, no es una tarea fácil. Los problemas de la IA en general (el cuello de botella de la adquisición del conocimiento, y otros) también aparecen en la Narrativa Computacional. Como una cantidad importante de información se requiere para implementar este tipo de sistemas, el coste de construir estas bases de conocimiento ha sido una barrera en el desarrollo de programas de este tipo.

De acuerdo con estas características de la Narrativa Computacional, una importante desventaja de los sistemas de generación es que, mientras que están construidos sobre modelos genéricos, el tamaño de historias que pueden procesar es reducido en comparación al coste de inserción de conocimiento. Esta es una de las razones que ha evitado que la generación de historias se aplique en la industria. En general, esto ha sido así por el *cuello de botella de adquisición del conocimiento*.

En particular, el foco científico de esta investigación se ha puesto en intentar mejorar la escala a la que se genera en relación al esfuerzo de añadir conocimiento a mano. El proyecto propone un sistema de adquisición semiautomática de esquemas narrativos. El objetivo es ampliar la escala de generación mediante la reducción de esfuerzo para incluir nuevas reglas.

10.1. Motivación de la investigación

Varios sistemas de generación ya existen, pero la cantidad de historias que pueden generar es muy reducida. Para generar nuevas historias, nueva información del dominio tiene que ser incluida a mano, lo cual hace aumentar el coste de mejora y escalado de las capacidades de proceso. Esto es costoso y la experiencia prueba que impide crear sistemas muy complejos.

En vez de tratar de resolver el problema de adquisición de conocimiento el bloque, tiene sentido restringir el alcance del problema a un subdominio. En concreto, los textos narrativos tienen ciertas propiedades que pueden ser usadas para crear modelos restringidos de adquisición de esquemas.

Tiene sentido, por lo tanto, crear un modelo que, si bien no sustituye totalmente la intervención manual, al menos puede reducir la cantidad de trabajo que los humanos deben desarrollar para crear nuevo conocimiento. Por lo tanto, la principal motivación de esta investigación es el beneficio que el estudio de estas posibilidades puede darse. Como se hace la hipótesis de que es posible reducir la cantidad de trabajo humano necesario para extraer estructuras narrativas, esto es considerado una motivación justificada.

10.2. Objetivos

- Crear un sistema computacional que reciba historias escritas por humanos como entrada.
- De ellas, extraer instancias de algún tipo de estructura, minimizando hasta el límite posible la necesidad de intervención humana. Esta estructura será definida como parte de la investigación.
- Usar esas instancias para generar automáticamente historias.
- Validar el proceso completo a través de evaluación humana, intentando restringir, en la medida de lo posible, el interfaz hombre-máquina a texto en lenguaje natural.

Las narraciones objetivo tienen que cumplir estos requisitos:

- Las narraciones sólo contendrán un sólo hilo narrativo. Complejas y múltiples líneas causales no están permitidas.
- Las narraciones no estarán ordenadas por relaciones temporales complejas. Se imponen estas restricciones:

- Cada evento en las narraciones durará una sola unidad de tiempo.
- Los eventos estarán ordenados por tiempo en las narraciones.

10.3. Hipótesis

Inicialmente, el foco fue puesto en la generación de historias. Como la intención era romper parcialmente el límite que la adquisición de conocimiento pone en este campo, se formuló la hipótesis de que una función de evaluación explícita podía ser suficiente [León and Gervás, 2010].

Sin embargo, esta hipótesis no fue correcta y la función de evaluación no sirvió para mucho. Sin embargo, el proceso llevó a la formulación de una nueva hipótesis (que resultó definitiva), basada en el tratamiento estructural:

> Un cierto conjunto de relaciones *estructurales* en una narración es suficiente para crear historias evaluadas como *coherentes* por humanos. Es decir, una vez que estas relaciones son conocidas, la coherencia puede ser analizada en estos términos.

Adicionalmente, se completó la hipótesis con esta sub-hipótesis:

> Las instancias de estas relaciones estructurales hipotéticas pueden ser adquiridas semiautomáticamente mediante un algoritmo computacional. Por lo tanto, los patrones extraídos desde historias coherentes y no coherentes pueden ser recogidos como contenido para la generación.

10.4. Metodología

El desarrollo de este trabajo ha sido dirigido por la aplicación de metodología de investigación basada en el estudio del trabajo previo en el campo que nos ocupa y la aplicación de las tecnologías disponibles para probar, hasta el punto posible, la validez de las hipótesis previamente enunciadas.

Una aproximación secuencial para la propuesta científica ha sido llevada a cabo: la identificación del problema, el enunciado de la hipótesis, la sugerencia de una solución, y la experimentación y el análisis de los resultados.

La primera etapa de la investigación consistió en la lectura de literatura científica y no científica en el área de la Narrativa Computacional. Por lo

tanto, los avances en Narratología e Inteligencia Artificial fueron tenidos en cuenta para aprender de un sustrato rico antes de afrontar la identificación de objetivos útiles.

Aparte de un ordenador personal (AMD Phenom™9550 Quad-Core Processor), la implementación ha sido llevada a cabo usando el sistema SWI-Prolog [Wielemaker, 2010]. Todo el sistema de documentación ha sido maquetado con LaTeX[The LaTeX Project, 2010], incluyendo este documento. Para el análisis estadístico, el programa y lenguaje R ha sido usado [R Development Core Team, 2010]. Scripts de post-procesado de datos han sido creados con Lua [Lerusalimschy et al., 2006] y el lenguaje de programación Ruby [Thomas and Chad Fowler, 2005]. Todo el desarrollo, pruebas y análisis han sido ejecutados en GNU/Linux.

Capítulo 11

Trabajo previo

Este capítulo está dedicado al estudio de la investigación científica previa, cuyo desarrollo influencia este trabajo. Principalmente trata de analizar las técnicas teóricas y prácticas relacionadas con el diseño y el desarrollo de esta tesis.

11.1. Narratología

La Narratología es la ciencia dedicada al estudio estructuralista de las *narraciones* y de la manera en la cual los humanos las entendemos y las usamos. Dado que la aproximación a las narraciones que propone la Narratología está estrechamente relacionado con la Narratología Computacional, es útil comentar algunas de sus más importantes características.

La Narratología se centra en las características internas de la narrativa y en las similitudes y diferencias con otros tipos de comunicación. La Narratología estudia estas características, intentando quedar parcialmente aislada de otros fenómemos como la lingüística o la semiótica. De este modo, la Narratología permanece como una disciplina independiente.

Algunas descripciones de *narración* la definen como un cuento, algunas otras como la acción de contar algo, y otras se centran en la descripción de las partes estructurales del discurso narrativo propiamente dicho [Real Academia Española, 2010]. Este enfoque estructuralista a la descripción de las narraciones establece la base de la Narratología moderna, aunque la ciencia de la Narratología se considera de un modo retrospectivo, quizá enraizando en Aristóteles *Poetica* [Aristotle, 1974]. Aristóteles postuló que la imitación del mundo real crea argumentos de los cuales las unidades más importantes son escogidas y ordenadas en una trama. La imitación de las acciones en el mundo real (*praxis*) crea un argumento

(*logos*) del cual las unidades (*mythos*) son escogidas.

La Narratología moderna tienes sus orígenes en los análisis estructuralistas de los textos, basados en el trabajo de Barthes, Genette, Todorov y otros [Barthes and Duisit, 1975, Todorov, 1977, Genette, 1979]. La Narratología más moderna, post-estructuralista, crea perspectivas en las cuales la ciencia cognitiva hace que se consideren las narraciones como fenómenos psicológicos, y propone un estudio de la narrativa en términos de una perspectiva cognitiva [Herman, 2000].

11.2. Narrativa Computacional

La Narrativa Computacional se dedica a la representación y procesado de las narraciones por ordenadores. Uno de los subcampos más estudiados de la Narrativa Computacional es la generación de historias. Aparece en los años 70 del siglo XX con el interés humano de entender y procesar historias.

Roger Schank fue el pionero del estudio del impacto de la narrativa en humanos desde un punto de vista formal y computable. Schank postuló que la manera de la que la memoria funciona no está sólo basada en procesos que manipulan datos mentales, sino que es un proceso continuo de recuerdo y adaptación de historias previas que definen nuestro mundo, como en el Razonamiento Basado en Casos (que se origina en el trabajo de Schank) [Schank and Abelson, 1977, Schank, 1982].

Schank propone el término *script*, que consiste en unidades cortas de conocimiento secuencial sobre los pasos típicos que han de ser tomados en una cierta situación. Por ejemplo, Schank propone el clásico ejemplo del restaurante: alguien entra en un restaurante, pide comida, come, etcétera. Este conocimiento común, según Schank, is muy importante en el conocimiento humano.

Para implementar este tipo de conocimiento semántico en ordenadores, Schank propone la *dependencia conceptual*. La dependencia conceptual es un método formal basado en acciones primitivas y relaciones entre ellas y sus objetos [Lytinen, 1992].

Programas como SAM [Cullingford, 1981] or PAM [Wilensky, 1981] comenzaron el desarrollo de una teoría de narrativa computacional basada en personajes y en la manera de la que intentaban alcanzar sus objetivos. Algunos modelos del modo en el que los procesos de la memoria pueden ser implementados han sido también estudiados [Kolodner, 1980].

Estos estudios provocaron la aproximación inversa: en vez de intentar entender los métodos humanos, tenía sentido intentar generar narrati-

va a través de modelos de métodos humanos, como se hizo en TALESPIN, [Meehan, 1976, Meehan, 1981] y otros.

Tras esta etapa inicial, la Inteligencia Artificial aplicada a la narrativa en estos términos fue abandonada, con las excepciones de algunos sistemas [Turner, 1992, Mueller, 1987].

La Inteligencia Narrativa [Mateas and Sengers, 1999] y otros proyectos multidiscipinares en Narrativa Computacional renuevan el interés del campo en el sigo XXI. La formalización teórica de la narrativa, el desarrollo de los sistemas de generación de historias, sistemas de extracción de conocimiento humano narrativo, y otros proyectos, están, de nuevo, acrecentando el interés en el campo.

11.3. Evaluación de Narrativa

La evaluación y la comprensión de la narrativa han sido estudiadas principalmente por la Psicología. Por ejemplo, Kelly analiza cómo la personalidad se desarrolla de acuerdo con las construcciones estructurales que los humanos creamos a través de las historias que leemos u oímos [Kelly, 1955]. Applebee estudió cómo las capacidades evaluativas en narrativa de los niños crecen en paralelo con sus capacidades de contar historias [Applebee, 1978].

En relación con la evaluación computacional de la narrativa, hay interés en encontrar algún tipo de métrica para comprar la calidad de los sistemas de generación de historias. Rowe et al. proponen STORYEVAL, un armazón para medir y comparar sistemas de generación de historias. STORYEVAL sugiere evaluar estos sistemas teniendo en cuenta *métricas narrativas*, *estudios cognitivo-afectivos*, *estudios centrados en el director* y *evaluaciones narrativas características* [Rowe et al., 2009].

Capítulo 12

Definición de narraciones

Teniendo en cuenta la representación de historias que se ha hecho en investigaciones previas, una definición formal de narración ha sido creada y refinada iterativamente para encajar con los requisitos de la aproximación computacional que se trata en esta tesis. Durante el proceso de creación de este modelo, diferentes formalizaciones fueron consideradas [León et al., 2007b, León and Gervás, 2008, León et al., 2008].

12.1. Definición de narración en su relación con modelos conceptuales

Esta investigación define *narración* o *historia* como un conjunto ordenado de hechos por orden cronológico. Por lo tanto se igualan, hasta cierto punto, los términos de *fábula* y *narración* de acuerdo con la definición de algunos formalistas rusos [Propp, 1928]. Es importante notar que esto es una definición *ad-hoc* usada como nomenclatura para esta tesis: no se asume ninguna similitud psicológica, formal o narratológica. De aquí en adelante, cualquier referencia a *narración* o *historia* se refiere al mismo concepto, excepto donde se especifique lo contrario.

Esta investigación se centra en el estudio formal de tramas de historias, y no en características narrativas o ligüsticas más complejas. Este foco en las tramas ha sido también seguido en la mayor parte de sistemas de generación narrativa que se presentan en la literatura [Meehan, 1976, Bringsjord and Ferrucci, 1999, Riedl and Young, 2006]. Por lo tanto, no se hace ningún estudio sobre la realización superficial como texto o las propiedades artísticas. Aunque una generación simple de texto ha sido llevada a cabo por motivos de experimentación, se asume que no se ha conseguido una calidad suficiente.

12.2. Definición formal de narración en este trabajo

La representación formal se centra en la capa de discurso, y, en particular, la definición formal de narraciones no se enmarca en ningún dominio semántico en concreto. Permite, así, la representación de conocimiento del mondo [León and Gervás, 2010]. La semántica que el nivel cognitivo asigna no se tiene en cuenta.

El modelo formal está presentado en orden ascendente en relación con sus partes constituyentes. Primero, los elementos más básicos, los *átomos*, se muestran. Después, las *acciones*, y después las *narraciones* propiamente dichas. Estos nombres de las partes han sido creados para tener una palabra de referencia, y no se asume ninguna similitud con los conceptos que representan en el idioma común. Aunque la selección de los términos está claramente influenciada por conceptos similares de Narratología, sólo el significado formal que se asigna en esta investigación debe ser interpretado cuando se haga referencia a ellos.

No se hace ningún asunción sobre la aplicabilidad posterior de este modelo de narraciones en otros sistemas. Ha sido creada para este proyecto de investigación y, aunque podría servir para basar otros trabajos, éste no ha sido el objetivo.

12.3. Constituyentes de las narraciones formales

12.3.1. Átomos

Los *átomos* son elementos atómicos que definen una sola cosa, carácter, idea o cualquier instancia simple de cualquier concepto. Pueden representar cualquier entidad en un dominio particular, pero están restringidos de tal modo que no pueden hacer referencia a ninguna parte de la narración. Por tanto, los átomos no pueden representar *acciones* ni *narraciones* (definidas a continuación).

Los *átomos* están unívocamente especificados por su nombre. Esto significa que dos átomos con el mismo nombre representan el mismo concepto. Ejemplos de átomos son *john*, *bird*, *house*, *hope* o *sad*. En este sentido, los *átomos* son similares a los de la lógica de primer orden.

12.3.2. Variables

Más adelante se verá que la evaluación y el algoritmo para la extracción de estructuras para la creación de nuevas historias necesitan *variables* para representar reglas abstractas. Las variables que se proponen en este modelo son similares a las variables lógicas, y pueden representar cualquier átomo. Las variables están representadas por un símbolo seguido de un signo de interrogación. Por ejemplo *x*? ó *token_variable*? son representaciones válidas de variables.

Una vez que una variable está ligada a un átomo en una ejecución de los algoritmos (descritos más adelante), todas las instancias de esa variable están ligadas a ese átomo. Sólo hay un espacio de variables en el modelo, de tal modo que *todas* las acciones que contienen la variable *x*? verán ligado su valor una vez se ligue en un punto de la ejecución.

12.3.3. Acciones

Las *acciones* son los constituyentes básicos de las narraciones, y representan eventos en la historia. Las acciones están definidas por estos eventos, propiedades o relaciones (de aquí en adelanto, el *núcleo*) y una secuencia ordenada de átomos. La acción, por tanto, enlaza un núcleo con átomos, significando que la acción, propiedad o relación es cierta para ellos.

Un núcleo está representado en forma de una palabra, que puede ser compuesta. Por ejemplo *love* o *take_to* son núcleos. Aunque esto podría dar lugar a confusión porque los núcleos están formados de la misma manera que los átomos, en la práctica esto no ocurre porque los núcleos, en el resto del modelo, siempre forman parte de una acción (tal y como se define después). La estructura de las acciones define formalmente la unión entre el núcleo y los átomos.

De acuerdo con los parámetros enlazados al núcleo en forma de átomos o variables, dos tipos de acción se han definido:

- Las *acciones instancias* son acciones cuyos elementos son átomos (*take*(*john*, *glass*)).

- Las *acciones de plantilla* son acciones en las cuales algún elemento es una variable (*love*(*ofelia*, *x*?) o *attack*(*x*?, *y*?)).

12.4. Narraciones

Las *narraciones* son secuencias ordenadas de acciones. Han sido definidas a partir de las partes anteriormente descritas (átomos, núcleos y acciones). La ecuación 12.1 muestra una representación formal de una narración genérica.

$$n = [a_1, a_2, a_3, \cdots, a_n] \tag{12.1}$$

donde $[a_1, a_2, a_3, \cdots, a_n]$ es una lista de acciones instanciadas y n es una narración.

Capítulo 13

Procesamiento estructural de narraciones

La primera parte de la investigación tuvo como objetivo la realización de un prototipo funcional en el que se estudiaba la generación de textos narrativos a gran escala mediante un enfoque semántico. El resultado final fue que, si bien tenía sentido la aproximación, el coste de la inserción de conocimiento era muy alto. Por tanto, se decidió centrar la investigación científica en un ámbito puramente estructural, tal como se describe en este capítulo. El trabajo y los resultados de la primera parte de la investigación pueden encontrarse en [León and Gervás, 2010].

Este capítulo muestra cómo una relación estructural de las narraciones, en particular, puede ser la base del proceso automatizado de narraciones. Esta relación ha sido llamada *enlace precondicional*, y las siguientes secciones explican sus principales características. La selección de una relación simple permite analizar computacionalmente las características estructurales de las historias y construir un algoritmo que extrae reglas para los enlaces precondicionales, las llamadas *reglas precondicionales* (capítulo 14).

13.1. De un modelo cognitivo a una definición estructural

Estudiando los resultados recogidos de la evaluación con humanos en el generador semántico de historias, fue detectado que había una correlación muy clara entre dos variables: la *causalidad* y la *cronología*. La causalidad representaba la percepción de que todo pasa por una razón lógica, y la cronología evaluaba la correcta disposición de los eventos en el

tiempo (por ejemplo, que las causas aparecían antes que las consecuencias). Como se trabajó en un dominio muy simple, fácilmente interpretable por todos los evaluadores, todos ellos evaluaron la calidad de estas dos variables con muy pocas diferencias.

Una heurística fue creada para capturar esto: el *enlace precondicional*. Los enlaces precondicionales tratan de describir un patrón estructural que está implicada en el reconocimiento humano de la causalidad y la cronología, ignorando el contenido semántico de estos dos conceptos.

13.2. Propiedades estructurales como coherencia estructural

La aproximación propuesta en este trabajo no pretende emular ningún modelo cognitivo. La propuesta consiste en el cambio desde aproximaciones puramente semánticas a otras más estructurales, y consecuentemente la coherencia en las narraciones está definida en estos términos, sin ninguna asunción cognitiva, como se establece a continuación:

> Una historia es coherente en el modelo propuesto si es evaluada como tal por evaluadores humanos.
>
> Una historia es *estructuralmente coherente* is tres patrones estructurales están presentes: el *foco*, en *enlace único* y la *conexión completa*.

El *foco*, el *enlace único* y la *conexión completa* se definen más adelante. No han sido explicados antes porque su definición están enlazada con la noción de enlace precondicional. La ausencia de una base cognitiva ha llevado a la definición de estas propiedades en términos del enlace precondicional, que es detallado más adelante.

Una definición formal de coherencia estructural en una historia s se muestra en la ecuación 13.1. La implementación (capítulo 15) se basa en esta ecuación.

$$\begin{aligned} \mathit{is\_structurally\_coherent}(s) &= \mathit{is\_focused}(s) \\ &\wedge\ \mathit{is\_fully\_connected}(s) \\ &\wedge\ \mathit{is\_uniquely\_linked}(s) \end{aligned} \tag{13.1}$$

La definición de la coherencia estructural ha sido formalizada como una función de evaluación booleana. El modelo establece que una narración *es* estructuralemente coherente o *no*.

13.2.1. Foco

Una historia tiene *foco* cuando tiene un *sólo* final. Es decir, termina en una sólo acción o evento. Esta definición se basa en el trabajo de Trabasso sobre la coherencia en tramas [Trabasso and Sperry, 1985]. Intuitivamente, si una historia se cuenta y no se puede extraer ninguna conclusión de la narración, podría ser considerada que está incompleta. Esto ha sido interpretado como una falta de coherencia. Una definición formal de foco puede verse en la sección 13.4.1.

13.2.2. Conexión completa

Una historia está *completamente conectada* cuando todas las acciones están relacionadas entre sí. Esta definición trata de capturar intuitivamente la restricción de simplicidad de que sólo un hilo narrativo puede existir en las historias simples. La conexión completa captura el hechos de que ninguna acción en la historia debe ser contada sin una razón y que todos los hechos relevantes deben ser contados (de otro modo, la conexión estaría perdida). La conexión completa está formalmente definida en la sección 13.4.2.

13.2.3. Enlace único

Una historia está *únicamente enlazada* si dos eventos están relacionados por una single idea o conclusión. Esta es la definición conceptual de otra parte de la simplicidad en las historias coherentes. Si dos eventos están relacionados con otro por varias relaciones, más de un hilo está presente. Por lo tanto, la historia no es *simple* en los términos establecidos. Esta definición conceptual de enlace único está definida en la sección 13.4.3.

13.3. Enlaces precondicionales

La definición de enlace precondicional sólo se *inspira* en la manera en la que, heurísticamente, los humanos llevan a cabo la interpretación de historias, pero no intenta capturar ningún proceso cognitivo. En vez de eso, la propuesta actual lo define como una heurística para máquinas, y no para humanos. Es decir, la definición está estrictamente ligada a la información computacionalmente procesable: la información que los humanos usamos y la manera en la que lo hacemos no es necesariamente compatible.

En una historia formalizada como la secuencia de acciones $\{e_1, \cdots, e_n\}$, las acciones $\left\{e_i, \cdots, e_j\right\}$ (las *precondiciones*) están *precondicionalmente enlazadas* a e_k (la *consecuencia*) si aparecen antes que e_k y el grafo dirigido resultante de crear enlaces precondicionales para todas las acciones tiene coherencia estructural

1. *foco*, teniendo todos los enlaces convergentes a una sola acción.
2. *conexión completa*, si todas sus acciones enraízan directa o indirectamente en una sólo acción, y
3. *enlace único*, según el cual cada par de acciones conectadas está conectada por un sólo enlace *como mucho*.

donde $1 \leq i, j, k \leq n$.

Esta definición es obviamente estructural, es decir, sólo captura propiedades superficiales de historias de acuerdo a una relación sintética. Como ejemplo, si una historia está compuesta por la acciones $\{a, b, c, d, e\}$, enlaces precondicionales válidos aparecerían en la figura 5, pero no en la figura 6.

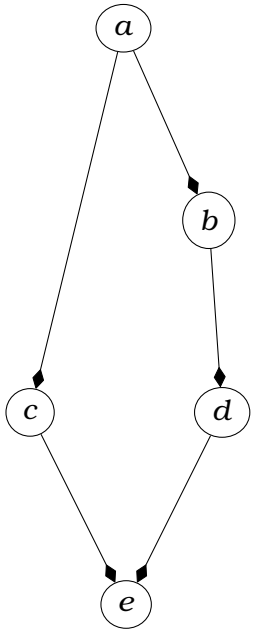
El conjunto de acciones de la precondición de un enlace precondicional representa conjunción. El modelo no considera representar disyunción porque esto llevaría al no-determinismo, que se ignora en este modelo. El proceso de razonamiento sobre la causalidad sería más complejo si una historia no determinara únicamente los enlaces precondicionales de algún hecho.

13.4. Definición formal de patrones estructurales

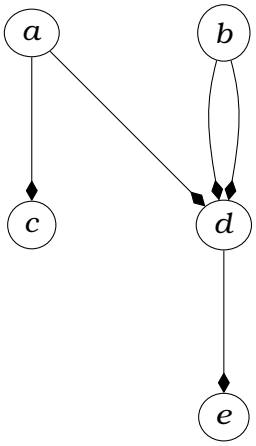
13.4.1. Foco

$$is\_focused(s) = |\{a \; \forall \; a \; \in \; s, \; if \; is\_outcome(s, a)\}| == 1 \tag{13.2}$$

$$is\_outcome(s, a) = \forall a' \in s - \{a\}, exist\_preconditional\_chain(s, a', a) \tag{13.3}$$



Graph 5: Conjunto válido de enlaces precondicionales. Las flechas representan enlaces.



Graph 6: Conjunto no válido de enlaces precondicionales. Las flechas representan enlaces.

13.4.2. Conexión completa

$$is\_fully\_connected(s) = action\_fully\_connected(a)\ \forall a \in s \tag{13.4}$$

$$action\_fully\_connected(s,a) = preconditional\_link(s,a) == root \vee action\_fully\_connected(s,a') \quad \forall\ a' \in preconditional\_links(s,a) \tag{13.5}$$

13.4.3. Enlace único

$$is\_uniquely\_linked(s) = action\_uniquely\_linked(a)\ \forall a \in s \tag{13.6}$$

$$action\_uniquely\_linked(s,a) = \neg\ duplicates\_in(pl) \quad \text{where } pl = all\_preconditional\_links\_to(s,a) \tag{13.7}$$

13.5. Reglas precondicionales

Las reglas precondicionales son pares de un conjunto de *precondiciones* y una *consecuencia* en la cual las precondiciones son acciones de plantilla (según se definen en la sección 12.3.3), y la consecuencia es una acción de plantilla sencilla. La expresión 13.8 muestra un ejemplo.

$$go(x?, y?) \wedge see(x?, z?) \diamond want(x?, z?) \tag{13.8}$$

13.6. Cálculo de los enlaces precondicionales en una historia

El cálculo de los enlaces precondicionales en una historia consiste en el proceso de asignar los enlaces precondicionales apropiados a los hechos correspondientes en una historia, de modo que se obtenga finalmente una estructura concreta.

El algoritmo 7 muestra una versión en pseudocógido del algoritmo no determinista para calcular una red precondicional correcta en una historia. Como se muestra en la figura 13.1, la entrada del algoritmo consiste en una historia y un conjunto de reglas precondicionales.

Algorithm 7 Pseudocódigo para asignar una red precondicional a la historia s.

```
1: s ← historia actual
2: R ← conjunto de reglas precondicionales
3: for candidate_network ∈ candidate_preconditional_networks(R, s) do
4:    l ← apply candidate_network on s
5:    if is_structurally_coherent(l) == true then
6:       return candidate_network
7:    end if
8: end for
9: return "No se puede encontrar una red precondicional para s."
```

Algorithm 8 *candidate_preconditional_networks*: algoritmo en pseudocódigo para encontrar las redes precondicionales candidatas.

```
1: R ← conjunto de reglas precondicionales
2: s ← historia actual
3: for a ∈ s do
4:    for any r ∈ R do
5:       if a puede ser instanciada con r then
6:          añadir enlaces entre las precondiciones de r y a
7:       end if
8:    end for
9: end for
10: yield network
```

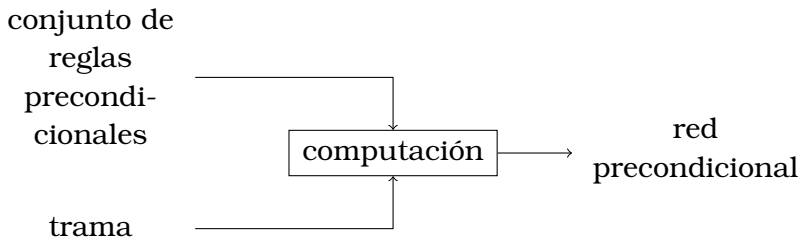


Figura 13.1: Modelo de caja negra el algoritmo de cómputo de enlaces precondicionales.

Capítulo 14

Extracción automática de reglas precondicionales

Este capítulo explica cómo el proceso de adquisición de reglas se lleva a cabo en relación con el modelo.

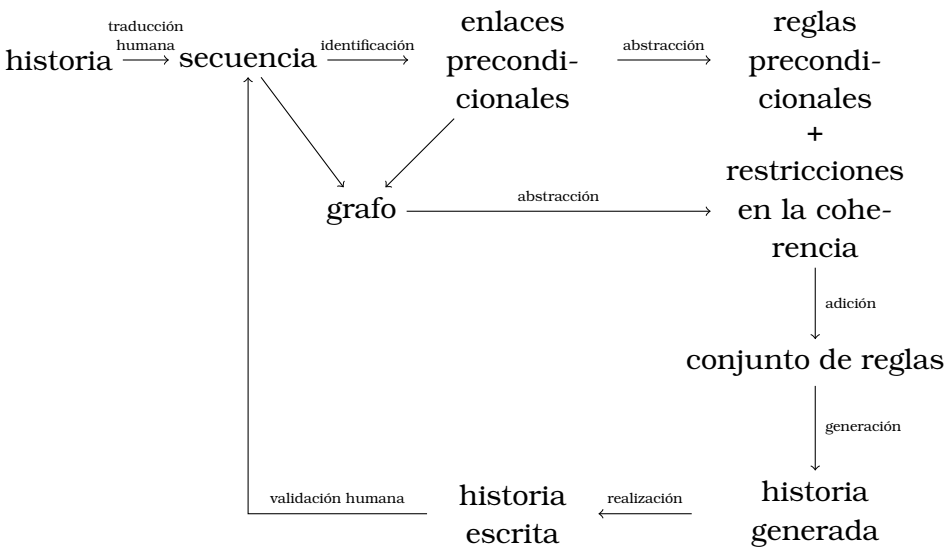


Figura 14.1: Descripción gráfica del proceso de extracción de reglas.

14.1. Algoritmo de extracción de reglas

14.1.1. Corpus de historias de entrada

Se tiene un corpus fijo de entrada, de historias coherentes, al empezar la ejecución. En cada iteración, una nueva historia es generada. Así, se podría decir que un nuevo corpus basado en el anterior se crea: nada evitaría usar las historias coherentes generadas en el proceso como corpus de entrada para otra ejecución, aunque esto no hay sido tratado en este trabajo.

14.1.2. Generación de reglas precondicionales

Para cada historia en el corpus de entrada y para cada historia generada (tal como se muestra en la secciones siguientes), se extraen reglas precondicionales. Esto es llevado a cabo como se detalle en el algoritmo 9. Este algoritmo es análogo al algoritmo para crear enlaces precondicionales mostrado en 7, y abstrae implícitamente las reglas de los enlaces en un proceso de variabilización básico [Charniak and McDermott, 1985].

Algorithm 9 Algoritmo en pseudocódigo para extraer un conjunto de reglas de un corpus de entrada.

```
1: s ← historia actual
2: for next_candidate ∈ siguiente conjunto candidato de reglas do
3:    s_linked ← calcular los enlaces precondicionales de s con next_candidate
4:    if ϵ(s_linked) == true then
5:       return next_candidate
6:    end if
7: end for
8: return "No se puede encontrar un conjunto de reglas."
```

Cada conjunto candidato de reglas precondicionales es generado como se describe en el algoritmo 10. La definición formal de la función *candidate_rules* puede ser examinada en el algoritmo 11.

La función *single_rule*(*story, kernel*) devuelve reglas de acuerdo con el siguiente orden. Se asume que existe una función que crea iterativemente patrones de variables $var_a, \cdots, var_b$.

1. Primero, el átomo especial $raizraiz \diamond nucleo(var_1, \cdots, var_n)$.

2. Después, se genera el kernel de la siguiente acción de la historia *story*: $nucleo_{accion\ previa}(var_i, \cdots, var_j) \diamond nucleo(var_x, \cdots, var_y)$.

Algorithm 10 Orden de generación del los conjuntos de reglas precondicionales candidatos.

```
1: s ← historia actual
2: K ← conjunto de núcleos de s
3: R ← {}
4: for i ← 1 until i = |K| do
5:    add candidate_rules(K_i, s) to R
6: end for
7: return R
```

Algorithm 11 Reglas candidatas para cada núcleo.

```
1: s ← historia actual
2: k ← núcleo actual
3: m ← número máximo de reglas por núcleo
4: for i ← 1 until i = m do
5:    yield i reglas para k en la historia s
6: end for
```

Algorithm 12 Algoritmo de creación de un número concreto de reglas. Auxiliar del Algoritmo 11.

```
1: s ← historia actual
2: k ← núcleo actual
3: j ← número de reglas que van a generarse
4: rules ← {}
5: for i ← 1 until i = j do
6:    añadir una nueva regla a rules de acuerdo con single_rule(s, k)
7: end for
8: yield rules
```

3. Tras eso, el paso 2 se repite yendo hacia atrás en la historia hasta la primera acción.

Este orden asume un conjunto de variables $\left\{var_1, \cdots, var_j\right\}$. Las variables son creadas mediante el análisis de los átomos de las acciones en las cuales la creación de reglas precondicionales está basad. El proceso primero examina los diferentes átomos de la historia, devolviendo un conjunto $T = \{t_1, \cdots, t_n\}$. Después, un conjunto correspondiente de variables es creado, mapeando cada átomo con una nueva variable $V = \{v_1, \cdots, v_n\}$.

Resumiendo, una historia de ejemplo haría que se llevara a cabo la búsqueda de reglas precondicionales para el kernel *go* mediante la exploración de los candidatos siguientes, tras parametrizar, con este resultado:

$$root \diamond go(x?, y?)$$
$$root \diamond go(x?, y?); buy(x?, a?) \diamond go(x?, z?)$$
$$buy(x?, a?) \diamond go(x?, z?)$$
$$buy(x?, a?) \diamond go(x?, z?); go(x?, y?) \diamond go(x?, z?)$$
$$go(x?, y?) \diamond go(x?, z?)$$

Finalmente se devolvería este conjunto de reglas:

$$root \diamond go(x?, y?); buy(x?, a?) \diamond go(x?, z?)$$

14.1.3. Generación de historias

Como el modelo teórico no impone ninguna restricción, la definición del sistema computacional para generar historias se detalla como parte de la implementación en la sección 15.2. Es importante dejar claro en este punto que la calidad de este generador de historias está lejos de ser alta, y es un defecto aceptado de la implementación que se propone, por no ser uno de los objetivos de la investigación. En el capítulo 17 se propone un sistema de generación de historias más sofísticado.

14.1.4. Adquisición de criterio humano

De forma análoga a la generación de historias, el modelo teórico de este capítulo no impone ninguna manera particular de aplicar criterio humano en las historias generadas para clasificación. Siguiendo una perspectiva de caja negra 14.2, una historia es insertada en esta parte del proceso, y se espera un valor booleano por parte de un humano que coincida con su opinión sobre la coherencia de la narración.

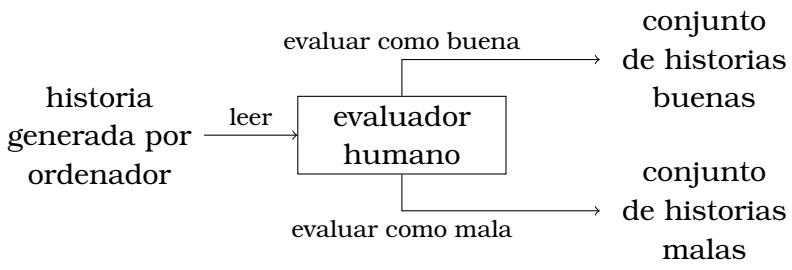


Figura 14.2: Modelo de caja negra de la clasificación booleana de historias por humanos.

14.1.5. Recogiendo y mezclando reglas precondicionales buenas y malas

En una aproximación simple, las reglas precondicionales extraídas *sólo* desde historias coherentes eras usadas como reglas básicas en las cuales un patrón de unificación simple generaba una historia válida. Tal sistema fue implementado. En las pruebas de la implementación de este sistema, historias claramente no coherentes fuere generadas, lo cual era un error.

Para solucionar esto, el modelo actual considera un conjunto de reglas particionado en *reglas buenas* y *reglas malas* tras la intervención humana. Para abstraer esta clasificación de reglas, tras la clasificación de *la historia*, sus reglas son extraídas y añadidas a uno u otro conjunto.

- El conjunto de reglas buenas contiene los enlaces precondicionales de las historias buenas.
- El conjunto de las reglas malas conjunto de reglas de historias consideradas no coherentes siempre que estas reglas no estén en el conjunto anterior.

Es decir, si una historia considerada correcta, s_i, tras el análisis, genera las reglas precondicionales $\{r_1, r_2, r_3\}$ y una historia s_j, no correcta, crea las reglas $\{r_1, r_4, r_5\}$, el conjunto *bueno* contiene las reglas $\{r_1, r_2, r_3\}$ y el conjunto *malo* contiene las reglas $\{r_4, r_5\}$.

Capítulo 15

Implementación, evaluación y resultados

15.1. Corpus de historias

15.1.1. Historias de asesinatos

El primer prototipo funcional de la función de evaluación fue construido para tramas cortas de historias de asesinato. La intención era estudiar la plausibilidad de una función de evaluación para historias, no enlazada únicamente a la corrección [León and Gervás, 2010]. Esta implementación consideraba un conjunto de variables mayor, las cuales fueron elegidas sin ningún modelo narratológico o psicológico. La selección estaba basada en la intuición del autor sobre lo que constituye una buena historia.

Los resultados empíricos era prometedores. La evaluación humana mostró que crear una función de evaluación que mide varias variables, basada en el análisis secuencial de versiones formales de los textos puede llegar a asemejarse al criterio humano. Sin embargo, se hizo patente que simplemente cambiando el foco no era suficiente: el cuello de botella de adquisición de conocimiento aún estaba presente.

15.1.2. Aesop's Fables

Una vez que los límites de la aproximación basada en conocimiento fuera claramente experimentados, el proceso estructural con fábulas fue llevado a cabo según se explica en los capítulos 13 y 14. Se encontró un problema relacionado con el dominio en particular: Aunque las fábulas tienen una estructura similar y muy simple, muchos eventos diferentes aparecen en ellas (muchos verbos). Somo esto ocurre, el número de reglas

precondicionales adquiridas por cada núcleo es muy bajo. Generalmente, en este caso, sólo hay una o dos reglas por cada caso en todo el corpus, lo cual es muy bajo. Este caso hace que sea muy difícil experimentar dado que muy pocas reglas por núcleo siempre dan lugar a las mismas historias. Es decir, aunque estas nuevas historias serían coherentes, serían muy parecidas unas a otras.

15.1.3. Óperas

Después de descubrir el problema de las fábulas, se decidió trabajar con óperas de siglo XIX. Estas óperas clásicas siguen una patrón muy clásico basados en las pasiones humanas y finales trágicos. Esto sugirió que era posible adaptar las óperas de tal modo que una traba básica fuese abstraída como historia simple. Esta adaptación fue requerida porque las óperas no cumplen el requisito de ser cortas y con un sólo hilo. Sin embargo, fue posible identificar hilos simples principales en las óperas.

15.2. Generación de historias simple

La generación de historias propuesta es muy simple y directa. Tiene que ser tenido en cuenta en la generación de historia como tal no es el objetivo de esta investigación, por lo tanto, esta aproximación a la generación de historias puede ser fácilmente mejorada. Sin embargo, esto ha sido dejado como trabajo futuro (capítulo 17). El sistema de generación de historias propuesto informa la operación de exploración mediante la aplicación de las reglas precondicionales extraídas hasta el momento de la generación. Es decir, en vez de generar candidatos sin ninguna restricción, sólo aquellos candidatos que pueden posteriormente satisfacer las reglas preconcionales son creados. Como estas reglas han sido cogidas de la aplicación del proceso de adquisición, las reglas deberían capturar, hasta cierto punto, esquemas narrativos coherentes. Las pruebas para esta hipótesis se detallan en las secciones siguientes.

15.3. Ejecución de un usuario

La secuencia de evaluaciones se presenta en la figura 15.1.

Para representar de una manera más gráfica la evolución del criterio sobre las historias generadas, es posible representar la proporción entres las historias coherentes y no coherentes para las últimas 6 generaciones.

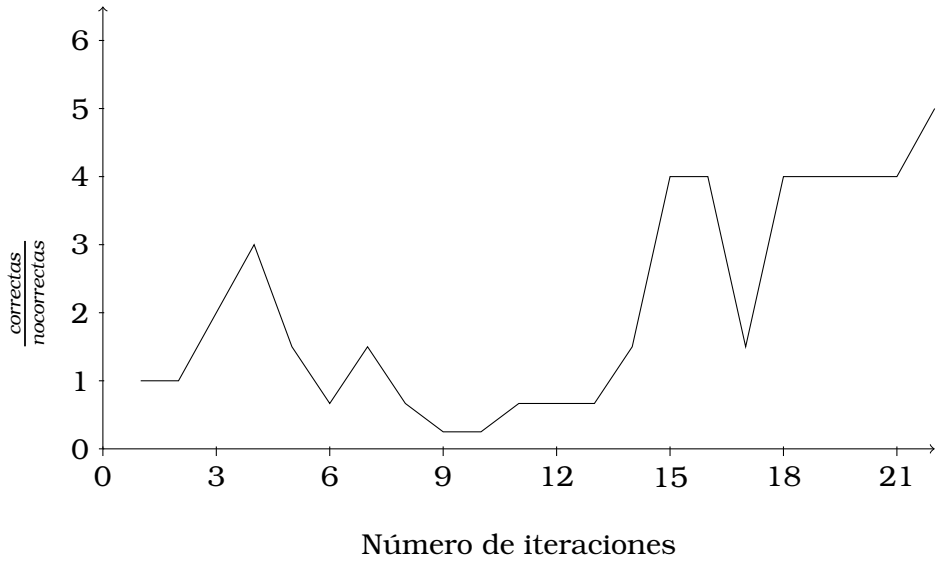


Figura 15.1: Curva de aprendizaje para un usuario.

El experimento concreto tardó en ejecutarse aproximadamente 8 minutos y medio. Esto significa un tiempo medio de 25 segundos por historia, aproximadamente. Aunque la creación de reglas a mano para el mismo dominio no ha sido llevada a cabo, se asume que este sistema pseudoautomático es mucho más rápido, basándonos en la experiencia. Por tanto, la solución es prometedora de acuerdo con sus objetivos. Al menos, para dominios simples como el que se muestra en estos resultados. Se planea estudiar dominios más complejos como parte del trabajo futuro.

15.3.1. Saturación

Un límite se impuso en el experimento: una vez que el usuario llega a un punto en el que la mayoría de las historias son evaluadas como coherentes, el experimento se detiene. Puede comprobarse en la sección anterior que este límite ha sido establecido en 5. Esto es así porque durante los primeros ensayos se encontró un punto de *saturación*.

Tras algunas evaluaciones, el procedimiento actual fue incapaz de extraer más reglas de una manera robusta. Mientras que es teóricamente posible encontrar más reglas, esto no ocurre tan rápido como en la primera fase de la ejecución (*fase de no saturación*). Fue empíricamente comprobado que una vez que 5 ó 6 historias seguidas son clasificadas como coherentes, la probabilidad de estas en la zona de saturación es muy alta,

así que se decidió estableces este punto como la condición de parada.

La saturación no tiene lugar por el sistema de experimentación en sí mismo. La manera en la que se hacen modificaciones a las historias generadas para crear nuevas tiene un límite. Para mantener la semejanza con las historias que pueden ser generadas mediante la pura aplicación de reglas, algunas restricciones están podando demasiadas posibilidades. La cantidad de patrones diferentes que esto puede crear es pequeña, por lo tanto, se agota, en un momento dado, este conjunto de patrones. Las propuestas para el trabajo futuro, en el capítulo 17, examinan otras aproximaciones posibles a la generación de historias para superar esta limitación.

15.4. Resultados globales

Se ha medido el tiempo medio en alcanzar la saturación. Si la adquisición pseudoautomática de reglas para un dominio es más lenta que hacerlo a mano, la utilidad de esta solución es discutible. Como media, se tardó 8.78 minutos, con una desviación estándas de 2.752. La evaluación más rápida tardó 4.224 minutos, y la más lenta 13.407. El tiempo medio por cada historia fue de 26.20 segundos, con una desviación estándar de 7.02.

En relación a la comparación con la creación manual de historias, tiene que ser tenido en cuenta que el sistema propuesto es capaz de generar un conjunto de reglas a partir de un corpus de entrada sin intervención humana, si se desea. Mientras que ha sido demostrado que esta aproximación es incompleta, es claro que el beneficio en términos de tiempo es notable.

La figura 15.2 muestra la coherencia estructural media del proceso de adquisición de reglas de todos los evaluadores con los que se ha experimentado. Puede ser comprobado cómo la proporción entre historias coherentes y no coherentes aumenta casi linealmente durante la ejecución de las pruebas.

Se asume que el contenido de las historias afecta a la evaluación, es decir, no todas las tramas y no cualquier corpus de entrada daría estos resultados. Las razones se explican en este capítulo, principalmente en la sección 15.1. Mientras que estos resultados son prometedores y la evaluación empírica muestra que adquirir reglas de forma pseudoautomática es posible, la aplicación de esta aproximación a otros dominios (principalmente a aquellos más complejos) se planea como parte del trabajo futuro.

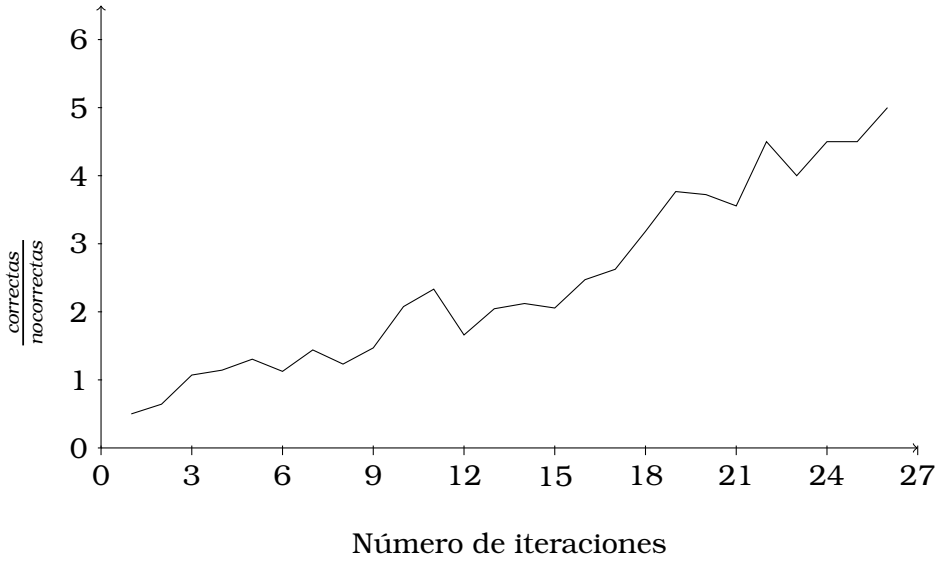


Figura 15.2: Proporción media de historias coherentes y no coherentes durante el proceso de adquisición de reglas. Se muestra la proporción de las últimas 5 historias.

Capítulo 16

Discusión

16.1. Aspectos conceptuales de esta solución

La principal asunción en esta investigación es que los resultados empíricos son validos, y que el hecho de que el método modelado no imite los procesos psicológicos humanos no es influyente. Como se presentó en el capítulo 10, la reducida cantidad de conocimiento disponible sobre los procesos mentales humanos en narrativa hace muy difícil implementar sistemas que emulen *realmente* a los humanos.

16.1.1. Número de historias en el proceso de extracción de reglas

El algoritmo 9 sólo aprende de *una* historia. Mientras que estos resultados ofrecen resultados útiles (capítulo 15), el sistema podría aprender contenido de más historias al mismo tiempo haciendo una variación básica en el algoritmo. Esto sería beneficioso en principio porque hay contenido implícito en el *conjunto* de historias que podría ser extraído, no sólo de las historias individualmente. Por ejemplo, el sistema podría extraer una regla r_a de una historia s_a una una regla r_b de s_b. Aunque por separado podrían ser útiles, juntas no lo serían.

Como el algoritmo de extracción de reglas debe devolver un solo conjunto de reglas, deben ser coherentes como conjunto, y no sólo de forma independiente. Hasta ahora, el algoritmo sólo asegura la coherencia para historias simples.

16.1.2. Impacto del corpus de entrada en el resultado final

El conjunto de historias elegido como corpus de entrada afecta a los resultados generales de los experimentos mostrados en el capítulo 15 porque son la base de las historias generadas. Corpus diferentes crearía diferentes historias con distintas características. Mientras que se asume que las reglas resultantes dependerán en el contenido de entrada, es importante discutir la manera en la que la *calidad* de los resultados es afectada por el proceso de creación del corpus.

En la implementación en particular, las óperas han sido resumidas y traducidas a una representación formal a mano, lo que introduce el criterio del autor en la ejecución. Tiene sentido, entonces, tener en cuenta que esto es una fuente de error. Actualmente, sin embargo, no hay manera de hacer estoy automáticamente.

16.1.3. Influencia de otros aspectos diferentes a la coherencia

Tal y como se muestra en el capítulo 15 se les pedía a los evaluadores que se centraran en la coherencia narrativa cuando evaluaban historias. Esta restricción explícita fue claramente expuesta en el cuestionario para reducir la influencia de otros aspectos de las historias en el resultado. La experiencia adquirida presentada en la primera parte de la investigación [León and Gervás, 2010] muestra que la opinión sobre la calidad global que los lectores perciben cuando leen una historia influye en la valoración de otras variables más concretas. Es decir, si un lector encuentra que la historia es graciosa o interesante, considerará que la historia es buena, por lo cual es posible encontrar una buena valoración para una historia que no es realmente coherente.

Para afrontar este hecho parcialmente, el modelo asume que la evaluación con criterio humano es la manera global de evaluación. Es decir, si los evaluadores humanos consideran una historia coherente, las reglas son *correctas*. Esta decisión es la consecuencia de la definición estructural de la propuesta, que intenta evitar modelos cognitivos. Si las historias que no son coherentes son evaluadas como tal por humanos, es válido para el sistema y el alcance de esta investigación.

16.1.4. Influencia de la *opinión* humana en el proceso de extracción de reglas

Establecer una definición general de coherencia en historias válida por todas las historias posibles no tiene sentido porque la opinión de individuos particulares está implicada en el proceso. No hay manera, actualmente, de formalizas la opinión humana, y no hay razón, a priori, para das más relevancia a una opinión que a otra en todos los casos. Por lo tanto, el conjunto particular de evaluadores humanos en el proceso de adquisición de reglas definido en la sección 14.1 afecta al conjunto final de reglas adquiridas. Es decir, el conjunto final puede ser inválido para otros humanos.

Los resultados recogidos en la sección 15.4 sugiere que la evaluación de la coherencia es razonablemente uniforme para un dominio simple bajo ciertas condiciones más o menos restringidas, pero esto no tiene que ser el caso para todos los escenarios. Se hace la hipótesis de que dominios más complejos en los cuales la evaluación de coherencia en historias no será uniforme pueden ser encontrados. El estudio de esta hipótesis y sus consecuencias se planea como trabajo futuro.

16.2. Comparación con otras aproximaciones

Para que se puedan recoger reglas, el algoritmo necesita, como entrada, un conjunto de historias para usarlo como “corpus de entrenamiento”. Este corpus, sin embargo, debe satisfacer ciertas restricciones. Cualquier conjunto de textos reunidos no es suficiente, en general, para hacer que el sistema funcione. Esta aproximación está basada en conocimiento estructural en narrativa que narraciones simples y cortas incluyen, no es datos estadísticos.

La mayor parte de algoritmos de aprendizaje máquina pueden aprender la mayor parte de las veces sin importar la calidad del corpus. La exactitud del aprendizaje depende de la calidad del corpus. Sin embargo, el algoritmo propuesto carece de esta habilidad: si las historias de entrada no son procesables por la función de evaluación, nada podrá ser aprendido.

Por otro lado, los resultados de este proceso de aprendizaje son útiles totalmente tras su adquisición: el conjunto de reglas es perfectamente modificable y legible por humanos.

La otra aproximación considerada que merece la pena comprar es la adquisición de reglas causales desde una base de conocimiento externa. El beneficio general es claro: no hay necesidad de crear una base de conocimiento, y esto ahorra tiempo y esfuerzo. Sin embargo las opciones posibles

como OpenCyc [CyCorp, 2010] o ConceptNet [Liu and Singh, 2004] tienen ciertas características que hacen difícil aplicar la aproximación actual a la evaluación de contenido narrativo.

Se concluye que la opción de aprender reglas es la más beneficiosa. Su coste es bastante bajo en comparación con el coste de crear las reglas a mano, y provee una cobertura robusta dentro de dominios concretos. Tiene el coste de desarrollar el algoritmo, pero una vez éste ha sido creado, la adaptación a nuevos dominios es, en principio, directa.

Capítulo 17

Conclusiones y trabajo futuro

17.1. Conclusiones de la investigación

Tras el análisis de los resultados mostrados en el capítulo 15 puede ser concluido que las hipótesis han sido *parcialmente* validadas. A través de la implementación y las pruebas ha sido demostrado cómo *es posible* construir un sistema que lleva a cabo un proceso estructural que crea historias que son consideradas *coherentes* por humanos.

Un énfasis especial es puesto en el hecho de que las hipótesis han sido sólo demostradas *parcialmente*. Esto es considerado así porque:

- Los experimentos implican criterio humano. Esto significa que la evaluación de coherencia es sólo parcial, y que muchos más evaluadores humanos podrían haber sido usados para llevar a cabo un estudio más específico. Sin embargo, debido a que el sistema es un prototipo, ha sido considerado innecesario.
- El foco inherente de la tesis está relacionado con narrativa, que es un término muy complejo que no tiene una definición única aceptada. Por lo tanto, sentenciar que las historias son *coherentes*, mientras que plausible, no puede ser formalmente demostrado.

17.2. Beneficios e inconvenientes

Se ha concluido que la contribución actual reduce, hasta cierto punto, el esfuerzo humano que se requiere para mejorar el dominio de generación de narrativas para una aplicación. Mientras que esto se considera un beneficio claro, tiene que quedar claro que el sistema propuesto, hasta aquí, sólo es aplicable a dominios simples y a narraciones cortas y sencillas.

Mientras que un trabajo más detallado podría llevar a un modelo mejorado que provea cobertura a formatos narrativos más sofisticados, esto aún no se ha hecho.

Teóricamente, el modelo propuesto cubre el espacio completo de soluciones para todos los algoritmos propuestos. Esto permite una cobertura teórica de las soluciones, lo cual es bueno. Sin embargo, este proceso no ha sido realizable en la práctica porque los espacios en los que se trabaja son tan grandes que una exploración completa es intratable en términos de computación. Por lo tanto, la ventaja teórica está bloqueada por las limitaciones prácticas.

Finalmente, se considera que una parte importante de la contribución es la propuesta de un paradigma totalmente estructural para el proceso computacional de contenido narrativo. Ha sido mostrado cómo esto puede ser positivo con un modelo original. Pero se hace la hipótesis de que, mientras que esto podría abrir nuevos caminos de investigación, es bastante probable que el proceso estructural por sí mismo no sea suficiente, y que aproximaciones semánticas sean necesarias para alcanzar un grado decente de calidad, al menos en comparación con las historias de otros sistemas de generación automática y de humanos.

17.3. Trabajo futuro

17.3.1. Mejora del modelo

Se ha demostrado empíricamente que el proceso estructural de historias tiene sentido, al menos para algunos dominios. Sin embargo, la aplicación de la aproximación propuesta no está limitada al sistema presentado, y se hace la hipótesis de que soluciones más sofisticadas y potentes pueden ser desarrolladas.

Por ejemplo, la clasificación de historias podrías ser mejorada de forma que la función de evaluación y el criterio humano no sean booleanos, sino que estén en un intervalo real. De este modo, una separación simple entre historias *coherentes* y *no coherentes* podría ser mejorada y un *ranking* de historias podría ser creado. Esto llevaría a un concepto de historia en el que unas son “más coherentes” que otras.

La figura 14.1 muestra un diagrama del sistema de adquisición de reglas precondicionales expuesto. Mientras que ha sido demostrado que tiene sentido construir conjuntos de reglas de este modo, alunas mejoras pueden ser aplicadas al algoritmo de modo que la aproximación a la adquisición de reglas sea más potente. Por ejemplo, el primer candidato aceptado se toma

como solución. Esto no es óptimo necesariamente, y más opciones pueden ser tenidas en cuenta. Por ejemplo, también, sería posible elegir el *mejor* candidato en base a una clasificación no booleana según la coherencia.

El modelo puede ser mejorado también mediante la adición de relaciones de tiempo más complejas a la definición de *historias simples*. Las asunción que se han hecho consideran un modelo extremadamente restrictivo. Esto se hizo para mantener el modelo simple y enfocado, pero se hace la hipótesis de que el modelo en su estado actual podría ser aplicado a historias en las que la duración de cada acción es mayor que una sola unidad e tiempo, por ejemplo. Relaciones temporales más complejas ampliarían el campo de aplicación.

17.3.2. Generación de historias mejorada

El proceso de adquisición de reglas se basa en crear historias candidatas que están clasificadas bajo supervisión. La calidad de el sistema de generación de historias afecta al tiempo requerido para llegar al punto de saturación en los experimentos y en la capacidad de encontrar reglas nuevas.

Tal y como ha sido diseñado, el proceso de adquisición puede sólo recoger las reglas precondicionales que están implícitamente presentes en las historias de entrada porque no se incluyen otras posibilidades. El punto de saturación se alcanza cuando no hay nada nuevo que "aprender", es decir, un máximo local en relación a las reglas precondicionales se ha alcanzado. Para evitar esto, se puede introducir *ruido* en el proceso.

17.3.3. Mejoras en la implementación

La implementación del modelo teórico ha sido llevada a cabo con motivos de prueba, y no ha sido adaptada para mejor uso tras esto. Para hacer posible que se lleve a cabo una investigación más avanzada, y dado que la demostración de la validez de tal modelo se hace empíricamente, tiene sentido crear una versión de la implementación que puede ser usada de una manera más robusta.

Por una parte, la implementación podría ser ejecutada más rápido. Durante las fases medias del desarrollo del prototipo, la paralelización del código fue tenida en cuenta. Mientras que fue finalmente descartada para mantener un proceso de desarrollo simple, debido al tipo de algoritmos que se usan, la paralelización puede hacer posible expirar subespacios mayores en la búsqueda de reglas precondicionales y enlaces en la generación de historias.

Por otra parte, el desarrollo de tal modelo intenta ser útil para la comunidad científica, de tal modo que modificar el prototipo siguiendo un patrón de diseño software que haga posible implementar y publicar el sistema como una librería –o cualquier otra forma de sistema distribuible– tiene sentido y utilidad.

Bibliography

[Aesop, 1992] Aesop (1992). *Aesop's Fables*. Houston: Advantage International, The PaperLess Readers Club.

[Allen, 1991] Allen, J. F. (1991). Time and time again: The many ways to represent time. *International Journal of Intelligent Systems*, 6:341–355.

[Applebee, 1978] Applebee, A. (1978). *The Child's Concept of Story: Ages Two to Seventeen*. Chicago: University of Chicago Press.

[Aristotle, 1974] Aristotle (1974). *Poética*, volume 8. Coleccion Biblioteca Románica Hispánica IV.

[Bal, 1998] Bal, M. (1998). *Narratology: Introduction to the Theory of Narrative*. University of Toronto Press, second edition.

[Barthes and Duisit, 1975] Barthes, R. and Duisit, L. (1975). An introduction to the structural analysis of narrative. *New Literary History*, 6(2):237–272.

[Bernstein, 2009] Bernstein, M. (2009). On hypertext narrative. In *HT '09: Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 5–14, New York, NY, USA. ACM.

[Bramsen et al., 2006] Bramsen, P., Deshpande, P., Lee, Y. K., and Barzilay, R. (2006). Inducing temporal graphs. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 189–198, Morristown, NJ, USA. Association for Computational Linguistics.

[Bringsjord and Ferrucci, 1999] Bringsjord, S. and Ferrucci, D. (1999). *Artificial Intelligence and Literary Creativity: Inside the mind of Brutus, a StoryTelling Machine*. Lawrence Erlbaum Associates, Hillsdale, NJ.

[Chambers and Jurafsky, 2008] Chambers, N. and Jurafsky, D. (2008). Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797, Columbus, Ohio. Association for Computational Linguistics.

[Chambers and Jurafsky, 2009] Chambers, N. and Jurafsky, D. (2009). Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610, Suntec, Singapore. Association for Computational Linguistics.

[Charniak and McDermott, 1985] Charniak, E. and McDermott, D. (1985). *Introduction to artificial intelligence*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Chatman, S., 1986] Chatman, S. (1986). *Story and Discourse: Narrative Structure in Fiction and Film*, volume Second Edition. Cornell University Press, USA.

[Crawford, 2005] Crawford, C. (2005). *On Interactive Storytelling - New Riders: Game Design and Development*. Peachpit Press, Berkeley, California, first edition.

[Cullingford, 1981] Cullingford, R. E. (1981). *Inside Comptuer Understanding: Five Programs Plus Miniatures*, chapter SAM. Lawrence Erlabaum Associates.

[CyCorp, 2010] CyCorp (2010). Opencyc. http://www.opencyc.org/.

[Dehn, 1981] Dehn, N. (1981). Story Generation After Tale-Spin. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 16–18.

[Elhadad and Robin, 1996] Elhadad, M. and Robin, J. (1996). An overview of surge: a reusable comprehensive syntactic realization component. Technical report, Ben-Gurion University of the Negev.

[Field, 1998] Field, S. (1998). *Screenplay: The Foundations of Screenwriting*. Amazon.com, third edition.

[Finlayson, 2009] Finlayson, M. (2009). Deriving narrative morphologies via analogical story merging. *New Frontiers in Analogy Research*, pages 127–136.

[Foster, 1941] Foster, E. M. (1941). *Aspects of the Novel*. Edward Arnold, London.

[Gatt and Reiter, 2009] Gatt, A. and Reiter, E. (2009). SimpleNLG: A realisation engine for practical applications. In *ENLG-2009*.

[Genette, 1966] Genette, G. (1966). *Figures I*. Seuil.

[Genette, 1969] Genette, G. (1969). *Figures II*. Seuil.

[Genette, 1972] Genette, G. (1972). *Figures III*. Seuil.

[Genette, 1979] Genette, G. (1979). *Narrative Discourse: An Essay in Method*. Cornell University Press.

[Gervás, 2007] Gervás, P. (June, 2007). Tap: a text arranging pipeline. Technical report, Natural Interaction based on Language Group, Universidad Complutense de Madrid, Spain.

[Gervás and León, 2010] Gervás, P. and León, C. (2010). Story generation driven by system-modified evaluation validated by human judges. In *First International Conference on Computational Creativity*, Lisboa, Portugal.

[Graesser et al., 1994] Graesser, A. C., Singer, M., and Trabasso, T. (1994). Constructing Inferences During Narrative Text comprehension. *Psychological Review*, (101):371–395.

[Hassan et al., 2007a] Hassan, S., León, C., Gervás, P., and Hervás, R. (2007a). A computer model that generates biography-like narratives. In *International Joint Workshop on Computational Creativity*, London.

[Hassan et al., 2007b] Hassan, S., Pavón, J., Arroyo, M., and León, C. (2007b). Agent based simulation framework for quantitative and qualitative social research: Statistics and natural language generation. In *The Fourth European Social Simulation Association Conference*, Toulouse, France. F. Amblard, F. Amblard.

[Herman, 2000] Herman, D. (2000). Narratology as a cognitive science. *Image and Narrative*.

[Jones and Galliers, 1996] Jones, K. S. and Galliers, J. R. (1996). *Evaluating Natural Language Processing Systems: An Analysis and Review*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Joyce et al., 1989] Joyce, M., Kaplan, N., McDaid, J., and Moulthrop, S. (1989). Hypertext, narrative, and consciousness. In *Hypertext'89 Proceedings, November 5-8, 1989, Pittsburgh, Pennsylvania, USA*, pages 383–384. ACM.

[Kelly, 1955] Kelly, G. (1955). *The Psychology of Personal Constructs*, volume I,II. Norton, New York.

[Klein et al., 1973] Klein, S., Aeschliman, J. F., Balsiger, D., Converse, S. L., Court, C., Foster, M., Lao, R., Oakley, J. D., and Smith, J. (1973). Automatic novel writing: A status report. Technical Report 186, Computer Science Department, The University of Wisconsin, Madison, Wisconsin.

[Kolodner, 1980] Kolodner, J. L. (1980). *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model*. PhD thesis, University of Yale, New Haven, CT, USA.

[Ladkin, 1987] Ladkin, P. B. (1987). The logic of time representation.

[Lang, 1999] Lang, R. R. (1999). A declarative model for simple narratives. In *Proceedings of the AAAI Fall Symposium on Narrative Intelligence*, pages 134–141. AAAI Press.

[Lebowitz, 1983] Lebowitz, M. (1983). Storytelling as planning and learning. In *nternational Joint Conference on Artificial Intelligence*.

[Lebowitz, 1985] Lebowitz, M. (1985). Storytelling as Planning and Learning. *Poetics*, 14:483–502.

[Lee, M., 1994] Lee, M. (1994). A Model for Story Generation. Master's thesis, University of Manchester.

[León and Gervás, 2010] León, C. and Gervás, P. (2010). The Role of Evaluation-Driven rejection in the Successful Exploration of a Conceptual Space of Stories. *Minds and Machines*, 20(4):615–634.

[León et al., 2008] León, C., Peinado, F., and Navarro, A. (2008). An intelligent plot-centric interface for mastering computer role-playing games. In *ICIDS08, 1st International Conference on Interactive Digital Storytelling*.

[Lerusalimschy et al., 2006] Lerusalimschy, R., Figueiredo, L. H. d., and Celes, W. (2006). *Lua 5.1 Reference Manual*. Lua.Org.

[Leslie Kaelbling, 1994] Leslie Kaelbling (1994). Associative Reinforcement Learning: A Generate and Test Algorithm. In *Machine Learning*.

[León and Gervás, 2008] León, C. and Gervás, P. (2008). Creative storytelling based on transformation of generation rules. In *5th Internation Joint Workshop on Computational Creativity*.

[León et al., 2007a] León, C., Hassan, S., and Gervás, P. (2007a). From the event log of a social simulation to narrative discourse: Content planning in story generation. In Olivier, P. and Kray, C., editors, *Conference of the Artificial and Ambient Intelligence*, page 402–409, Culture Lab, Newcastle University, Newcastle upon Tyne, UK.

[León et al., 2007b] León, C., Hassan, S., Gervás, P., and Pavón, J. (2007b). Mixed narrative and dialog content planning based on bdi agents. In *XII Conferencia de la Asociación Española para Inteligencia Artificial*, Salamanca, Spain.

[Liu and Singh, 2004] Liu, H. and Singh, P. (2004). ConceptNet: A Practical Commonsense Reasoning Toolkit. *BT Technology Journal*, 22.

[Lytinen, 1992] Lytinen, S. L. (1992). Conceptual Dependency and its Descendants. *Computers and Mathematics with Applications*, 23(2–5):51–73.

[Mancini, 2000] Mancini, C. (2000). From cinematographic to hypertext narrative. In *HYPERTEXT '00: Proceedings of the eleventh ACM on Hypertext and hypermedia*, pages 236–237, New York, NY, USA. ACM.

[Mani et al., 2006] Mani, I., Verhagen, M., Wellner, B., Lee, C. M., and Pustejovsky, J. (2006). Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 753–760, Morristown, NJ, USA. Association for Computational Linguistics.

[Mateas and Sengers, 1999] Mateas, M. and Sengers, P. (1999). Narrative Intelligence: An Introduction to the NI Symposium. In *Working Notes of the Narrative Intelligence Symposium*, pages 1–10.

[Mateas and Stern, 2005] Mateas, M. and Stern, A. (2005). Structuring content in the Façade interactive drama architecture. In *Proceedings of AIIDE*, pages 93–98.

[McIntyre and Lapata, 2009] McIntyre, N. and Lapata, M. (2009). Learning to tell tales: a data-driven approach to story generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL-IJCNLP '09, pages 217–225, Morristown, NJ, USA. Association for Computational Linguistics.

[McKee, 1997] McKee, R. (1997). *Story. Substance, Structure, Style and the Principles of Screenwriting*. Regan Books, New York.

[Meehan, 1976] Meehan, J. (1976). *The Metanovel: Writing Stories by Computer*. PhD thesis, Yale University.

[Meehan, 1981] Meehan, J. R. (1981). *Inside Comptuer Understanding: Five Programs Plus Miniatures*, chapter Tale-Spin and Micro Tale-Spin. Lawrence Erlabaum Associates.

[Michie, Donald, 1968] Michie, Donald (1968). Memo Functions and Machine Learning. *Nature*, 218:19–22.

[Mueller, 1987] Mueller, E. T. (1987). *Daydreaming and Computation: A Computer Model of Everyday Creativity, Learning and Emotions in the Human Stream of Thought*. PhD thesis, UCLA, Computer Science Department.

[Murray, 1997] Murray, J. H. (1997). *Hamlet on the Holodeck. The Future of Narrative in Cyberspace*. MIT Press, Cambridge, MA.

[Neil and Simborowski, 1993] Neil, P. and Simborowski, N. (1993). *The Complete Fairy Tales of Charles Perrault*. Houghton Mifflin Harcourt.

[Niehaus and Young, 2009] Niehaus, J. and Young, R. M. (2009). A Computational Model of Inferencing in Narrative. In *Intelligent Narrative Technologies II*. AAAI Press.

[Peinado et al., 2008] Peinado, F., Navarro, A., and Gervás, P. (2008). A Testbed Environment for Interactive Storytellers. In *International Conference on Inteligent Technologies for Interactive Entertainment*. ACM Digital Library.

[Pérez y Pérez, 1999] Pérez y Pérez, R. (1999). *MEXICA: A Computer Model of Creativity in Writing*. PhD thesis, The University of Sussex.

[Propp, 1928] Propp, V. (1928). *Morphology of the Folk Tale*. TX: University of Texas.

[Pérez y Pérez et al., 2007] Pérez y Pérez, P., Sosa, R., and Lemâitre, C. (2007). A Computer Model for Visual-Daydreaming. In *AAAI 2007 Fall Symposium on Intelligent Narrative Technologies*. AAAI.

[R Development Core Team, 2010] R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

[Real Academia Española, 2010] Real Academia Española (2010). Diccionario de la Real Academia Española.

[Reiter and Dale, 2000] Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press.

[Riedl, 2004] Riedl, M. (2004). *Narrative Planning: Balancing Plot and Character*. PhD thesis, Department of Computer Science, North Carolina State University.

[Riedl and León, 2008] Riedl, M. and León, C. (2008). Toward vignette-based story generation for drama management systems. In *INTETAIN, Workshop on Integrating Technologies for Interactive Stories*.

[Riedl and León, 2009] Riedl, M. and León, C. (2009). Generating story analogues. In *Proceedings of the 5th Artificial Intelligence and Interactive Digital Entertainment*.

[Riedl and Young, 2006] Riedl, M. and Young, R. (2006). Story Planning as Exploratory Creativity: Techniques for Expanding the Narrative Search Space. *New Generation Computing, Computational Paradigms and Computational Intelligence. Special Issue: Computational Creativity*, 24(3):303–323.

[Riedl and Sugandh, 2008] Riedl, M. O. and Sugandh, N. (2008). Story planning with vignettes: Toward overcoming the content production bottleneck. In *Proceedings of the 1st Joint International Conference on Interactive Digital Storytelling (Formerly TIDSE and ICVS)*.

[Ritchie, 2008] Ritchie, G. (2008). Uninformed Resource Creation for Humour Simulation. In *Proceedings of the 5th International Joint Workshop on Computational Creativity*, pages 147–151, Madrid.

[Rowe et al., 2009] Rowe, J. P., McQuiggan, S. W., Robison, J. L., Marcey, D. R., and Lester, J. C. (2009). Storyeval: An empirical evaluation framework for narrative generation. In *AAAI Spring Symposium*.

[Rumelhart, 1975] Rumelhart, D. E. (1975). Notes on a schema for stories. *Representation and Understanding: Studies in Cognitive Science*, pages 211–236.

[Schank and Abelson, 1977] Schank, R. and Abelson, R. (1977). *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. L. Erlbaum, Hillsdale, NJ.

[Schank, 1982] Schank, R. C. (1982). *Dynamic Memory : A Theory of Reminding and Learning in Computers and People*. Cambridge University Press.

[Sharples, 1996] Sharples, M. (1996). An account of writing as creative design. *The Science of Writing*.

[Sharples, 1999] Sharples, M. (1999). *How We Write*. Routledge.

[Swartjes and Theune, 2006] Swartjes, I. and Theune, M. (2006). A Fabula Model for Emergent Narrative. In S. Gbel, R. M. and Iurgel, I., editors, *Proceedings of the Technologies for Interactive Digital Storytelling and Entertainment, Third International Conference, TIDSE 2006*, number 4326 in Lecture Notes in Computer Science, pages 49–60, Heidelberg.

[The LaTeX Project, 2010] The LaTeX Project (2010). LaTeX - a document preparation system.

[The Project Gutenberg Team, 2010] The Project Gutenberg Team (2010). Project Gutenberg.

[Theune et al., 2003] Theune, M., Faas, E., Nijholt, A., and Heylen, D. (2003). The virtual storyteller: Story creation by intelligent agents. In *Proceedings of the Technologies for Interactive Digital Storytelling and Entertainment*, pages 204?–215.

[Thomas and Chad Fowler, 2005] Thomas, D. and Chad Fowler, A. H. (2005). *Programming Ruby: The Pragmatic Programmers' Guide*. Pragmatic Bookshelf, Raleigh, NC, 2. edition.

[Todorov, 1977] Todorov, T. (1977). *The Grammar of Narrative*, pages 108–119. The Poetics of Prose. Ithaca: Cornell UP.

[Trabasso and Sperry, 1985] Trabasso, T. and Sperry, L. (1985). Causal relatedness and importance of story events. *Journal of Memory and Language*, 24:595–611.

[Turner, 1992] Turner, S. (1992). *MINSTREL: A Computer Model of Creativity and Storytelling*. PhD thesis, University of California at Los Angeles, Los Angeles, CA, USA.

[van den Broek, 1988] van den Broek, P. (1988). The effects of causal relations and hierarchical position on the importance of story statements. *Journal of Memory and Language*, 27:1–22.

[Viktor Shklovsky, 1917] Viktor Shklovsky (1917). *Art as Technique*. University of Nebraska Press.

[Vilain et al., 1986] Vilain, M., Kautz, H., and Beek, P. (1986). Constraint propagation algorithms for temporal reasoning. In *Readings in Qualitative Reasoning about Physical Systems*, pages 377–382. Morgan Kaufmann.

[Vinet and Griffin, 2010] Vinet, J. and Griffin, A. (May, 2010). Arch Linux. http://www.archlinux.org/.

[Weyhrauch, 1997] Weyhrauch, P. (1997). *Guiding interactive drama*. PhD thesis, Computer Science, Carnegie Mellon Univ., Pittsburgh, PA.

[Wielemaker, 2010] Wielemaker, J. (2010). SWI-Prolog. http://www.swi-prolog.org/.

[Wiggins, 2006] Wiggins, G. (2006). A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems*, 19(7).

[Wilensky, 1981] Wilensky, R. (1981). *Inside Comptuer Understanding: Five Programs Plus Miniatures*, chapter PAM, pages 136–179. Lawrence Erlabaum Associates.

[Wu, 1992] Wu, X. (1992). Inductive learning: Algorithms and frontiers. *Artificial Intelligence Review*, 6.

Part III

Appendix

Appendix A

Operas

This appendix shows the short formalizations that have been created to test the system. They have been formalized by hand.

ill(*violetta*)
love(*violetta, alfredo*)
love(*alfredo, violetta*)
together(*alfredo, violetta*)
forces(*germont, violetta*)
breakup(*violetta, alfredo*)
despise(*alfredo, violetta*)
forgive(*alfredo, violetta*)
die(*violetta*)

Formal Story 10: Verdi's *La Traviata*.

ill(*mimi*)
love(*rodolfo, mimi*)
jealous(*rodolfo, mimi*)
breakup(*mimi, rodolfo*)
back(*mimi*)
die(*mimi*)

Formal Story 11: Puccini's *La Bohéme*.

love(*pinkerton*, *butterfly*)
love(*butterfly*, *pinkerton*)
together(*butterfly*, *pinkerton*)
breakup(*pinkerton*, *butterfly*)
back(*pinkerton*)
together(*pinkerton*, *kate*)
die(*butterfly*)

Formal Story 12: Puccini's *Madama Butterfly*.

love(*carmen*, *jose*)
love(*jose*, *carmen*)
kill(*carmen*, *girl*)
help(*jose*, *carmen*)
escape(*carmen*, *jose*)
breakup(*carmen*, *jose*)
love(*carmen*, *escamillo*)
die(*carmen*)

Formal Story 13: Bizet's *Carmen*.

love(*leonora*, *alvaro*)
love(*alvaro*, *leonora*)
kill(*alvaro*, *marques*)
escape(*alvaro*, *leonora*)
chase(*carlo*, *alvaro*)
kill(*alvaro*, *carlo*)
die(*leonora*)

Formal Story 14: Verdi's *La Forza del Destino*.

love(*tosca*, *caravadossi*)
love(*caravadossi*, *tosca*)
want(*scarpia*, *tosca*)
chase(*scarpia*, *caravadossi*)
forces(*scarpia*, *tosca*)
kill(*scarpia*, *caravadossi*)
kill(*tosca*, *scarpia*)
die(*tosca*)

Formal Story 15: Puccini's *Tosca*.

love(*salud*, *paco*)
want(*paco*, *carmela*)
breakup(*paco*, *salud*)
chase(*salud*, *paco*)
chase(*salvador*, *paco*)
die(*salud*)

Formal Story 16: Falla's *La Vida Breve*.

love(*duke*, *gilda*)
love(*gilda*, *duke*)
kidnap(*courtiers*, *gilda*)
forces(*duke*, *gilda*)
chase(*rigoletto*, *duke*)
kill(*sparafucile*, *gilda*)

Formal Story 17: Verdi's *Rigoletto*.

love(*otello*, *desdemona*)
chase(*iago*, *cassio*)
forces(*iago*, *otello*)
jealous(*otello*, *desdemona*)
kill(*otello*, *desdemona*)
die(*otello*)

Formal Story 18: Verdi's *Otello*.