Pontifícia Universidade Católica
do Rio de Janeiro

Fabio Araujo Guilherme da Silva

Emotions in Plots with Non-Deterministic Planning for Interactive Storytelling

Tese de Doutorado

Thesis presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática, PUC-Rio as partial fulfillment of the requirements for the degree of Doutor em Informática.

Advisor: Prof. Antonio Luz Furtado

Rio de Janeiro

April 2015

Fabio Araujo Guilherme da Silva

Emotions in Plots with Non-Deterministic Planning for Interactive Storytelling

Thesis presented to the Programa de Pós-Graduação em Informática, of the Departamento de Informática do Centro Técnico Científico da PUC-Rio, as partial fulfillment of the requirements for the degree of Doutor.

Prof. Antonio Luz Furtado
Advisor
Departamento de Informática – PUC-Rio

Prof. Bruno Feijó
Departamento de Informática – PUC-Rio

Prof. Marco Antonio Casanova
Departamento de Informática – PUC-Rio

Prof. Angelo Ernani Maia Ciarlini
UNIRIO

Prof. Esteban Walter Gonzalez Clua
UFF

Prof. José Eugenio Leal
Coordinator of the Centro Técnico Científico da PUC-Rio

Rio de Janeiro, April 15th, 2015

Fabio Araujo Guilherme da Silva

Obtained a B.Sc. in Mathematics (with a specialization in Informatics) at Universidade do Estado do Rio de Janeiro (Uerj) in 1994, and received his M.Sc. in Information Systems from Universidade Federal do Estado do Rio de Janeiro (UNIRIO) in 2010. He joined the Doctorate program at PUC-Rio in 2010, researching on interactive storytelling, artificial intelligence, user models and games. In 2012, his research on interactive TV comics at VisionLab (PUC-Rio) received honourable mention from the International Telecommunication Union (ITU).

Ficha Catalográfica

Silva, Fabio Araujo Guilherme da

Emotions in plots with non-deterministic planning for interactive storytelling / Fabio Araujo Guilherme da Silva ; advisor: Antonio Luz Furtado. – 2015.
155 f. : il. (color.) ; 30 cm

Tese (doutorado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2015.
Inclui bibliografia

1. Informática – Teses. 2. Narração interativa. 3. Algoritmos de planejamento. 4. Emoções. 5. Modelos de usuário. I. Furtado, Antonio Luz. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

Despite all the difficulties I have faced in these latest years, I cannot complain about any lack of support in my long journey.

First of all, I would like to thank Prof. Antonio Furtado, my advisor, counselor, guide and inspiration with so many ideas, culture and long and fruitful conversations about science, arts, life and all that matters.

I cannot forget to mention Prof. Bruno Feijó for his constant counselling, help and encouragement.

Also all the other professors at PUC-Rio for their professionalism and for sharing their knowledge so brilliantly.

I will always remember Prof. Angelo Ciarlini and Prof. Sean Siqueira from UNIRIO for guiding me so well in the beginning of this path.

My fellows D.Sc. candidates and coworkers at VisionLab, specially Augusto Baffa, Edirlei Soares Lima and Bruno Riodi.

All the staff of the Department of Informatics.

CAPES and PUC-Rio for the granted support, so important to make this work possible.

And, last but not the least, I would like to thank all my friends and family for having, even from a distance, always showed their love and support.

Abstract

Silva, Fabio Araujo Guilherme da; Furtado, Antonio Luz (Advisor). **Emotions in Plots with Non-Deterministic Planning for Interactive Storytelling.** Rio de Janeiro, 2015. 155p. D.Sc. Thesis - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Interactive storytelling is a form of digital entertainment in which users participate in the process of composing and dramatizing a story. In this context, determining the characters' behaviour according to their individual preferences can be an interesting way of generating plausible stories where the characters act in a believable manner. Diversity of stories and opportunities for interaction are key requirements to be considered when designing such applications. This thesis proposes the creation of an architecture and a prototype for the generation and dramatization of interactive nondeterministic plots, using a model of emotions that not only serves to guide the actions of the characters presented by the plan generation algorithm, but also influences the participation of the users. Also, to improve the quality and diversity level of the stories, characters must be able to evolve in terms of their personality traits as the plot unfolds, as a reflection of the events they perform or are exposed to.

Keywords

Interactive Storytelling; Planning Algorithms; Emotions; User Models.

Resumo

Silva, Fabio Araujo Guilherme da; Furtado, Antonio Luz. **Emoções em Enredos com Planejamento Não-Determinístico para Narração Interativa.** Rio de Janeiro, 2015. 155p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Narração interativa é uma forma de entretenimento digital em que os usuários participam do processo de compor e dramatizar uma história. Nesse contexto, a determinação do comportamento dos personagens de acordo com as suas preferências individuais pode ser uma maneira interessante de gerar histórias plausíveis, onde os personagens agem de forma crível. Diversidade de histórias e oportunidades de interação são requisitos fundamentais a serem considerados ao se projetar tais aplicações. Esta tese propõe a criação de uma arquitetura e de um protótipo para a geração e dramatização de enredos interativos não determinísticos, utilizando um modelo de emoções que não só sirva para guiar as ações dos personagens apresentadas pelo algoritmo de geração de planos, mas que também influencie a participação dos usuários. Além disso, para melhorar a qualidade e nível de diversidade das histórias, os personagens devem ser capazes de evoluir em termos de seus traços de personalidade durante o desenrolar da trama, como um reflexo dos eventos que realizam ou a que são expostos.

Palavras-chave

Narração Interativa; Algoritmos de Planejamento; Emoções; Modelos de Usuário.

Contents

List of Figures

1
Introduction

1.1.
Topic Overview

The art of storytelling is one of the most important aspects of human culture. Apart from its obvious entertainment purposes, it also serves as an instrument to preserve knowledge, transmit teachings and communicate complex information. Starting with oral transmission and rock art since prehistoric times, diverse highly popular types of media and art forms to tell stories have been developed over the years, such as books, theatre and films. The advent of computers and digital technologies in general brought the opportunity to explore new ways of telling stories, leading in particular to *interactive storytelling*, which is the central topic of this thesis.

In a society relying increasingly on computational technology while population becomes more and more computer-literate, everything seems set for the next generation of digital entertainment to finally take off (Louchart et al. 2006). In fact, the video game market alone is valued today at over US$93 billion (Meulen 2013), having surpassed film and music industry (Goodkind 2014). In this context, we include interactive storytelling systems, which, employing semi-automatic mechanisms, try to adapt the stories being told according to the preferences expressed by the users. Such systems can be applied to games (Young 2001), stories authoring, general entertainment and even commercial applications (Spierling et al. 2002).

One of the most common problems when creating an interactive storytelling system is how to provide a good level of interactivity engaging the user's participation, while maintaining the coherence of the story that is being generated. With this objective in mind, a valuable resource for the development of such systems is the use of artificial intelligence techniques, such as *planning algorithms* (Ghallab et al. 2004), since they can be directed to automatically create a logical chain of events wherein the required constraints are enforced. Planners,

however, must be equipped with a flexible interface so as to provide, while respecting the coherence requirement, the highest possible degree of freedom to the users in their interaction with the system.

Among the interactive storytelling systems that use planning, we shall highlight **Logtell** (Ciarlini et al. 2005), which dramatizes stories in a 3D environment. **Logtell** first introduced *nondeterminism* in the sense of allowing to dramatize an event in several different ways, lasting more or less time according to the convenience of the storytelling process. This opened the possibility of strong user interference at the dramatization level, with the limitation, however, that the effects of the events could not be changed. Therefore the system still lacked nondeterminism in the effects of the events, an important requisite to promote variability in plot generation.

In the direction of this improvement, the author developed, as part of previous work (Silva 2010), an HTN-based nondeterministic plan generator. With this addition, the specification of the literary genre on hand started to contemplate events with more than one possible outcome. HTN (Hierarchical Task Network) planning (Erol et al. 2004) was adopted to prevent that the introduction of nondeterminism and the treatment of multiple ramifications would affect system performance. The adoption of nondeterminism allows the user to intervene in the plot dramatization through choices that drive the plan generator when determining which ramifications to follow.

An attractive way to conduct such interferences (until now, only simulated by text commands through direct choice of the final state to be generated) would take into consideration emotional attributes related to particular characters and to the entire plot. To further enhance the emotional aspect of the plot, we also propose the use of user models reflecting the choices made in a multiuser environment, which will hopefully provide a valuable feedback to be used as a means to direct the events of the plot in order to please the audience as a whole.

As a starting point for the introduction of models of emotions, the present work uses the behaviour model presented by Barbosa et al. (2014), where characters engage in a three-step decision-making process of goal selection, plan selection and commitment, based on individual preferences originating from a personality model defined by the characters' drives, attitudes and emotions.

1.2. Objectives

The main objective of this work is to present an architecture and a prototype for the generation of interactive plots with nondeterministic events, using a model of emotions that not only serves to guide the actions of the characters in the plan generation, but also influences the users' participation. We also intend to enhance the quality of the stories by permitting the characters' personality to evolve in consequence of the events they perform or are exposed to.

We expect that this approach will proportion more verisimilitude, diversification and engagement to interactive storytelling, and that it may be used as a foundation for further expansions concerning the authoring process and the dramatization of the generated plots.

1.3. Contributions

Our work utilizes nondeterministic planning in order to augment the possibility of plot variations, combining the IPG planner, developed for the original **Logtell** prototypes, with HTN planning to speed up plan generation. The IPG planner itself was modified to work in accordance with the hierarchical nondeterministic planner, incorporating the concept of attempts.

Another significant feature of our work is a decision-making process based on personality traits, which determines the behaviour of the characters participating in a story. The required integration of this feature with the planning algorithms marks a novelty in the treatment of nondeterminism (cf. section 4.1.1). In order to make the plots more believable and bring coherence to the decisions made by the characters, the choice of what they want to do (their goals) and how they do it (their plans) must take into consideration their personality traits. These traits, involving drives, attitudes and emotions, partly determine their individual preferences.

The current version of our prototype has preserved several extensions introduced by previous versions, and has been further extended to make it possible to validate the events against the personality traits of the character currently

performing as agent. The prototype also permits to model one or more users according to some of these personality traits.

Some early results of our research work have been reported in (Silva & Furtado 2011, Silva et al. 2011, Silva et al. 2012, Barbosa et al. 2014).

The main contributions we expect to offer with this thesis are:

- Proposing a general architecture for nondeterministic interactive storytelling incorporating emotion-based decision-making;
- Developing algorithms and techniques to implement the proposed architecture;
- Enhancing user experience through a higher level of interaction and variation, and through more believable characters;
- Presenting a comprehensive overview of the more influential research efforts on interactive storytelling, covering the main concepts, achievements and challenges, while trying to show why it is still a promising and worthwhile field of research.

1.4. Thesis Structure

Chapter 2 presents some theoretical background regarding artificial intelligence and its main branches used in this thesis, namely automated planning and affective computing.

Chapter 3 presents some background on interactive storytelling, including remarks on traditional storytelling. Section 3.8, in particular, surveys related work on interactive storytelling systems with an emphasis on **Logtell**, the interactive storytelling system used as a basis for this work. Some of the systems reviewed deal with character and user models (as is the case in this thesis), so we also refer to these subjects, as also to certain topics from psychology related to the study of personality types.

Chapter 4 presents the proposed architecture of the emotional nondeterministic interactive storytelling system, which essentially combines the previous version of **Logtell** with nondeterministic events, coupled with a decision-making process based on emotional models and further expanded to

incorporate multiuser models and preferences. In special, section 4.2 constitutes the main thrust of the thesis.

Chapter 5 describes the main technical aspects related to the implementation of the proposed system's prototype, including the algorithms for the personality-based nondeterministic planner.

Chapter 6 contains running examples that were performed to test the implemented prototype. They also serve to illustrate how the system can be used, and to give practical evidence of the extended functionality of the new proposed architecture.

Finally, Chapter 7 presents the concluding remarks, as well as a list of suggested research topics for future work.

2
From Artificial Intelligence to Artificial Emotions

Two among the main aspects of our work are the use of automated planning to generate plots, and of affective computing properties to model the personality and emotions of the characters. Both topics belong to the area of *artificial intelligence* (AI). For a better understanding of our approach, we found convenient to provide a preliminary introduction to these aspects.

We begin with a necessarily sketchy exposition on the origins and principles of AI. Next we look at the theoretical background of *automated planning*, with an emphasis on *nondeterministic planning* and *HTN planning*, the latter being our solution to cope with the performance issues raised by the former. Related work combining both approaches is also mentioned in this section.

Finally, we also deal with *affective computing*, a relatively recent research field that purports to improve the sought-after intelligent behaviour of computers by recognizing and simulating emotions.

2.1.
Making Machines That Think

Intelligence is of fundamental importance to humans, to the point that we chose to classify ourselves as *Homo sapiens*. Regarding ourselves as the only rational species on this world, we feel the urge to engage in a never-ending search for other intelligent beings, which is perhaps what leads to programs like the *Search for Extraterrestrial Intelligence* (*SETI*) (Shostak 2014).

Maybe it can also help to explain why the idea of creating artificial creatures capable of thinking has been present in popular imagination since antiquity. Ancient Greek myths such as Galatea, a statue that came to life as a result of the amorous feelings of her sculptor, Pygmalion (Figure 2.1), and the case of Talos of Crete, a mechanical man built by Hephaestus, the Greek god of metallurgy (McCorduck 2004; Russell & Norvig, 2003) show the interest and fascination that such beings have always operated on us. Accordingly, human-like figures

believed to have intelligence and humanoid automata were built by craftsmen from every major civilization, including Egypt, Greece, China and Mesopotamia (Crevier 1993; McCorduck 2004; Needham 1986; Martin 2005).

More recent fiction from the 19th and early 20th centuries, like *Frankenstein* (Shelley 1816) and *R.U.R. (Rossum's Universal Robots)* (Čapek 2001) – which first introduced the word "robot" – has started to discuss the ethical issues and the possible uncontrollable consequences of creating artificially intelligent beings. McCorduck (2004) argues that these are examples of an ancient urge to "forge the Gods", characterizing artificial intelligence as the "scientific apotheosis of a venerable cultural tradition".

Figure 2.1 – Pygmalion and Galatea (Gérôme 1890).

Artificial intelligence is essentially an interdisciplinary field, drawing from a number of different sciences, including philosophy, mathematics, neuroscience, psychology, linguistics, and of course computer science. These fields have, since antiquity, contributed important knowledge and ideas to AI – such as Aristotle's formulation of the deductive reasoning method known as *syllogism* (Nilsson

1998), and George Boole's foundations of propositional logic (Boole 1854). However, only with the advent of the first electronic digital programmable computers in the 1940s – mostly influenced by the ideas set out by Turing (1936) in his seminal paper, *On Computable Numbers* – did AI finally come to fruition.

The first work that is now generally recognized as AI was done by McCulloch & Pitts (1943). Using knowledge from the basic physiology and function of neurons in the brain, a formal analysis of propositional logic (Whitehead & Russell 1910) and Turing's theory of computation, they proposed a model of artificial neurons and showed, among other things, that any computable function could be computed by some network of connected neurons, and that all the logical connectives (*and*, *or*, *not* etc.) could be implemented by simple net structures (Russel & Norvig 2003).

There were a number of early examples of work that can be characterized as AI, but it was Alan Turing (1950) who first articulated a complete vision of AI in his article *Computing Machinery and Intelligence*, introducing the Turing test, machine learning, genetic algorithms, and reinforcement learning (Russel & Norvig 2003).

The term *artificial intelligence* was coined in 1956, when the very field of AI was officially founded at a two-month workshop on the campus of Dartmouth College. Although not leading to any new breakthroughs, apart from the name for the field itself, the workshop served to introduce to each other those scholars who would become the leaders of AI research for the next two decades.

What followed was a period of great enthusiasm, where those prominent researchers and their students wrote programs that were – given the fact that only a few years earlier computers were seen as things capable of doing arithmetic and no more – simply astonishing: computers were solving problems in algebra, proving logical theorems, winning at checkers and even speaking English (Crevier 1993; Moravec 1988; McCorduck 2004; Russel & Norvig 2003). By the middle of the 1960s, AI research was flourishing around the world (McCorduck 2004; Crevier 1993; NRC 1999; Howe 1994), and the founders of the new research field were profoundly optimistic about its future, even predicting that those "machines that think, that learn and that create" would, in a foreseeable future, be able to handle a range of problems similar to that dealt by the human mind (Russel & Norvig 2003).

Reality would eventually show them some difficulties they initially failed to recognize. Attempts to scale up those first successful programs and techniques to cope with applications of practical importance revealed that they could only solve “toy problems” (Nilsson 1998), resulting in a period of reduced funding and interest in AI research known as *AI winter* (Russel & Norvig 2003).

This scenario caused the quest for an overambitious "general-purpose mechanism able to find complete solutions" of the first decade of AI research to be replaced by a search for powerful domain-specific knowledge that can more easily handle typically occurring cases in narrow areas of expertise (Russel & Norvig 2003). An example of this approach is DENDRAL, a system capable of inferring the structure of organic molecules given their chemical formula and information provided by a mass spectrogram (Buchanan *et al*. 1969). DENDRAL was the first successful knowledge-intensive system, giving birth to the commercially successful field of *expert systems*, a form of AI program that simulates the knowledge and analytical skills of human experts on narrowly defined tasks (Russel & Norvig 2003).

Another AI winter followed but, by the end of the 20th and the beginning of the 21st century, a convergence of several factors (including increasing computational power and a firmer adherence to the scientific method) caused AI to achieve its greatest successes, being applied in many areas throughout the technology industry, including logistics, data mining and medical diagnosis (Russel & Norvig 2003; McCorduck 2004; Kurzweil 1995; NRC 1999). Examples of such a success include Deep Blue, the first computer to beat a human chess champion (McCorduck 2004), the Kinect (Figure 2.2), a 3D body-motion interface system for the Xbox videogame consoles (Fairhead 2011), and the widespread use of intelligent personal assistants in smartphones (Rowinski 2013).

Given the astronomically high difficulty to handle the general problem of creating (or even simulating) intelligence, a promising strategy was to break it down into a number of specific sub-problems. These consist of particular capabilities that an intelligent system should possess. Among the corresponding subfields of AI, we can mention *knowledge representation*, *natural language processing*, *machine learning*, *machine perception*, *automated planning and scheduling* and *affective computing* (Russel & Norvig 2003; Luger & Stubblefield

2004; Poole *et al*. 1998; Nilsson 1998). Of particular interest for us are the last two subfields, which we will further explore in the following sections.

Figure 2.2 – User playing with Kinect.

2.2. Planning: The Reasoning Side of Acting

Planning is the process of choosing and organizing actions through the anticipation of its effects. This reasoning process aims to satisfy (by performing actions) some previously established goal. *Automated planning* is the sub-area of artificial intelligence that studies this reasoning process, using the computer (Ghallab *et al*. 2004). Despite this conceptual differentiation, the literature tends to adopt the simpler term "planning" with the same specific sense of "automated planning."

Planning is, due to the wide range of possible practical applications, a field of fundamental importance in artificial intelligence research. In fact, planning techniques have been applied in a wide variety of activities, including robotics, industrial machines control, web information search, autonomous agents, and spacecraft control in space missions.

2.2.1. Classical Planning

The vast majority of research works involving planning concern the so-called *classical planning*. This approach requires a number of assumptions; it being necessary that the planner have complete knowledge about the environment, which must be deterministic, static and finite-state with restricted goals and implicit time (Nau 2007). Besides, it's expected that the generated plans be a finite sequence of linearly ordered actions (Ghallab *et al.* 2004). The adoption of these restrictions leads to a simplification which proved useful in developing planning algorithms and, in fact, several of them were developed based on these assumptions.

Classical planning problems are defined on a domain composed by a set *S* of possible states, a set *A* of actions that can be applied to these states and a state-transition function $\gamma(s, a)$ that establishes which states are reached by applying these action to these states – i.e. $s' = \gamma(s, a)$ if *s'* is the state reached from *s* after applying action *a*. We can also, in some representations of this type of problems, adopt a generalization of the actions through a set *O* of operators with preconditions and effects represented by literals; in this case, the actions in a generated plan are *ground* instances (i.e. without any variable symbol) of these operators. The goal of this type of problem is to find a solution which, applied from a given initial state, leads to a determined goal state.

The simplest classical planning algorithms (*state-space planning*) require the plans to be finite sequences of totally ordered actions. However, it is possible to consider plans that are presented in not so rigid structures as, for example, partially ordered sets of actions (Nau 2007).

This is the case of *plan-space planning*. In classical state-space planning, the search space consists of a graph whose vertices are domain states and whose edges are transitions between states or actions: a plan, in this case, establishes a totally ordered sequence of actions corresponding to a path from the initial state to the final state. In contrast, with plan-space planning the vertices are partially specified plans (defined as a set of partially instantiated actions and a set of constraints), and the edges correspond to refinement operations over these plans. These refinement operations aim at achieving open goals (by adding actions) or

removing possible inconsistencies (by adding constraints). Such constraints can be (Ghallab *et al*. 2004):

- Action precedence constraints (e.g. action *a* must precede action *b*)
- Variable binding constraints
  - Inequality constraints (e.g. $v \neq c$)
  - Equality constraints (e.g. $v = c$)
  - Domain value limitation constraints (e.g. $v > 10$)
- Causal links (e.g. use action *a* to establish the precondition *p* needed by action *b*)

The refinement operations avoid adding constraints that are not strictly necessary for the refinement purpose, according to the *least-commitment principle*. The planning process then, starting with an empty plan, keeps making refinements until a solution plan is obtained whereby all the required goals are correctly achieved.

Plan-space planning differs from state-space planning not only in its search space, but also in its definition of a solution plan – which, in this case, is defined as a set of operators with ordering and binding constraints. This structure, being more general, may no longer correspond to a simple sequence of actions as in state-space planning.

One of the main restrictions of classical planning is the requirement that all actions must be deterministic, i.e. given a determined state *s*, the application of an action *a* will produce as result a unique state *s'*. In this case, the plan that serves as a solution to a planning problem is a linearly ordered finite sequence of actions a_0, a_1, ..., a_n that generate a sequence of state transitions s_0, $s_1 = \gamma(s_0, a_0)$, ..., $s_{n+1} = \gamma(s_n, a_n)$, such that each action a_i is applicable in s_i and s_{n+1} is the goal state.

Unfortunately, few situations in practice are consistently compatible with all the necessary limitations imposed by classical planning. To allow the treatment of most practical problems, it is necessary to relax one or more of the classical planning assumptions. In the next sections, we will see some situations where, by relaxing such assumptions, it is possible to obtain plans applicable to more general contexts wherein the limitations of classical planning cannot be assumed. Moreover, we will see how additional information about the domain, not considered in classical planning, can be used for obtaining more efficiency.

2.2.2. Nondeterministic Planning

Among the relaxations that can be made upon the classical planning assumptions, the acceptance of nondeterminism is perhaps the one that best favours a realistic approach to the world. Its main difficulty is that a plan may result in many paths of execution, which requires efficient ways of analyzing all possible outcomes, and then generating plans that have conditional behaviour and employ trial and error strategies (Ghallab *et al* . 2004). In fact, the treatment of nondeterminism raises efficiency issues even more serious than those of classical planning, because it is necessary to consider the various possible effects of each action.

There are two main approaches to planning with nondeterminism: planning based on *Markov Decision Processes* (MDPs), and planning as model checking. The first requires probabilities associated to the possible outcomes of every action and the attribution of *costs* to these actions, and of *rewards* to the different states. Through the use of these values, the planning problem becomes an optimization problem, where we try to maximize the rewards and minimize the costs.

Planning based on model checking, on the other hand, does not work with probabilities. An action can have more than one possible outcome and we do not know which of them will be effectively established when the action is executed. Both the goals and possible conditions that must be valid during the trajectory of the plan execution are expressed through temporal logic formulae. These formulae can require the fulfillment of the goals and conditions in different degrees. One kind of temporal logic that permits expressing such requirement degrees is *Computation Tree Logic* (CTL) (Emerson 1990), which contains operators to express necessities (i.e. the formula must be true on all the paths from the current state), possibilities (the formula must be true in some path from the current sate) and temporal conditions (e.g. “in some moment in the future”, “in all future states”).

Plans generated by these nondeterministic planning strategies are called *policies*. Unlike the sequence of actions that represent a plan in the classical state-space planning, a policy is an association between states and actions, establishing what action must be performed when a certain state of the world is observed. Such

an approach is of fundamental importance in nondeterministic domains, because the establishment of a unique sequence of actions that is guaranteed to satisfy the goals can just be impracticable. In addition, policies can establish conditional and iterative behaviours appropriate to the nondeterminism of the domain.

A major advantage of planning by model checking is the possibility of using propositional formulae to represent large sets of states, which allows a compact representation that facilitates planning for large-scale problems (Cimatti *et al*. 2003). In this case, the search process is performed by logic transformations on these formulae. BDDs (*Binary Decision Diagrams*) (Bryant 1992) are the most common form of implementation of this technique (Kuter 2005). Several experiments have shown that planning algorithms based on model checking and BDDs can cope well with problems of considerable size.

An example of implementation is MBP (*Model Based Planner*), a system for planning in nondeterministic domains. It consists of a framework to address different problems in nondeterministic domains, planning algorithms for dealing with large state spaces, and functionalities for plan validation and simulation (Cimatti *et al*. 2003).

Despite the expressiveness of CTL regarding the specification of goals (one of the reasons why it became the main formalism used for qualitative planning based on model checking) (Pistore & Traverso 2001), sometimes this language appears to be insufficient to express certain characteristics of the desired solution. This is the case of extended reachability goals that, besides specifying a condition to be achieved at the end of the execution of a policy, also establishes a condition to be preserved (or avoided) in all states visited during the execution of the policy. A proposal to solve this problem is presented by Pereira (2007), in the form of a new version of the CTL temporal logic, called α-CTL. In addition, this work presents the logic and the implementation of a model checker (VACTL) and a planner (PACTL) to address such problems, based on α-CTL itself.

2.2.3. Planning Based on Hierarchical Task Network

Another technique used in order to facilitate the resolution of practical problems is the use of specific knowledge about the problem domain. In this

sense, we can highlight Hierarchical Task Network planning - or, more simply, HTN planning. It is, in some aspects, similar to classical planning, since, in both cases, a sequence of actions is applied from an initial state, generating intermediate known states from one action to another. The biggest conceptual difference between the two approaches lies on what they plan for: while in classical planning actions are chosen based on *goals* to be achieved, in HTN planning the sequence of actions is established on the basis of *tasks* that must be performed.

The different ways how these tasks can be performed are described by *methods*, each of them providing a decomposition of the task associated with a set of smaller tasks, or *subtasks*. By recursive application of methods to *non-primitive tasks* (those which decompose into other tasks), we end up reaching *primitive tasks* (directly related to domain operators), which are then instantiated in tasks to be incorporated into the plan.

In order to describe these tasks and methods, we use knowledge about the domain upon which we are working. It can be said that the main idea of HTN planning, conceptually, is to use the knowledge about the domain in question to optimize the planning process for a given problem. This allows obtaining plans much more efficiently than in classical planning. And it is one of the reasons why HTN is one of the most popular planning techniques, if not the most popular, in practical applications (Ghallab *et al*. 2004).

An example of implementation of HTN planning is SHOP2, an algorithm that allows the establishment of partial ordering between the tasks of a method. SHOP2 produces totally ordered plans, adding each step of the plan in the same order in which it will be executed and is, therefore, able to know the state of the world at every step of the planning process (Nau *et al*., 2001). This knowledge is of great importance for obtaining solutions efficiently, since it permits limiting the search space considerably.

The contents of the following subsections, which should provide a broader understanding of HTN planning, are based on (Ghallab *et al*. 2004).

2.2.3.1. STN Planning

STN (*Simple Network Task*) planning is a particular, simpler case of HTN planning. Although STN planning evidences a loss of richness when compared to more generic HTN, most of the concepts related to the former remain valid with the latter. This fact, together with its easier understanding, makes it interesting enough to justify its introduction before coming to full HTN planning.

For STN planning, we keep the same definitions for operators, actions, plans and for the state-transition function $\gamma(s, a)$ (the result of applying an action a to a state s) that we have in classical planning. The concepts of STN that do not belong to classical planning, and are used both in defining problems and in their solutions, have to do with *tasks*, *methods* and *tasks networks*. These concepts are closely related to how a human being, endowed with specific knowledge of the field on which the planning will be applied, can imagine which domain (sub)problems can be solved.

2.2.3.1.1. Tasks and Task Networks

HTN planning problems (both in general and in the specific case of STN) are solved by performing tasks. Informally speaking, it can be said that a task is what should be done to solve a particular HTN planning problem. Tasks can generally be decomposed into smaller tasks, or *subtasks*, which at the intermediate levels of decomposition are labelled *non-primitive*. The process can be recursively repeated until it reaches tasks directly associated with domain operators that can be executed without the need for further decomposition. Such tasks are called *primitive tasks*.

In a more formal way, we can define a task t as an expression of the form $t(r_1, ..., r_k)$, where t is a *task symbol* and $r_1, ..., r_k$ are terms. The task is primitive (i.e. directly instantiable to describe a domain action) if t is an operator symbol; otherwise it is non-primitive. Moreover, we say that a task is *ground* if all its terms are *ground* (i.e. there is no variable symbol in the expression representing the task); otherwise it is called *unground*. Finally, we say that an action a

accomplishes a task *t* in a state *s* if *a* and *t* are identically named and *a* is applicable in *s*.

Big higher level tasks are decomposed into smaller lower level tasks through the definition of *task networks*. A task network is an acyclic digraph $w = (U, E)$, where *U* is the node set and *E* is the edge set. Each node *u* from *U* contains a task t_u, and each edge *w* represents a partial ordering between a pair of nodes. A task network *w* is considered *ground* or *primitive* if all its tasks are, respectively, ground or primitive; similarly, *w* is respectively considered *unground* or *non-primitive* when the above conditions are not satisfied.

If the ordering defined by the edges of *w* is total, i.e. there is only one possible successor to each of its nodes (except the last, for which there is none), then we say that *w* is *totally ordered*; otherwise we say that it is *partially ordered*.

2.2.3.1.2. Methods

Tasks tell us what we can do, but it is up to the methods to effectively determine which tasks are performed and in what order during the elaboration of a plan. A task can have more than one possible way to be accomplished, in which case the definitions contained in the methods will be used by the planner when it becomes necessary to choose one of them.

Formally, a method consists of a 4-tuple

$$m = (name(m), task(m), precond(m), network(m))$$

in which:

- *name*(*m*) is the *name* of the method, a syntactic expression of the form $n(x_1, ..., x_k)$, where *n* is a unique method symbol and the elements $x_1, ..., x_k$ correspond to the variable symbols occurring in *m*;
- *task*(*m*) is a non-primitive task, with which *m* is associated;
- *precond*(*m*) is a set of literals that are the *preconditions* for the execution of *m*;
- *network*(*m*) is a task network whose constituent tasks are called the *subtasks* of *m*; *task*(*m*) is considered accomplished when all its subtasks are accomplished.

A method *m* is *totally ordered* if *network*(*m*) is totally ordered. In this case, it is possible to replace the digraph specification for *network*(*m*) by a simpler one, *subtasks*(*m*), representing the subtasks of *m* as an ordered list of tasks

$$subtasks(m) = \langle t_1, ..., t_k \rangle,$$

where each task t_i corresponds to a node u_i of *network*(*m*).

2.2.3.1.3. Applicability, Relevance and Decomposition

Although a task can be associated with more than one method, there are criteria to determine which method should be chosen to perform its decomposition. These criteria relate to the *applicability* and the *relevance* of a method instance. We say that an instance *m* of a method is *applicable* to a state *s* if *s* satisfies the preconditions of *m*, i.e.:

$$precond^+(m) \subseteq s \text{ and } precond^-(m) \cap s = \emptyset,$$

where $precond^+(m)$ and $precond^-(m)$ are, respectively, the positive and negative preconditions of *m*.

A method instance *m* is said to be *relevant* to a task *t* if there exists a substitution σ such that $\sigma(t) = task(m)$. If so, the *decomposition* of *t* by *m* under σ, indicated by $\delta(t, m, \sigma)$, is equal to *network*(*m*) – or to *subtasks*(*m*), if *m* is totally ordered.

Whenever a method decomposes a task, a new task network is generated. We indicate by $\delta(w, u, m, \sigma)$ the set of task networks resulting from the decomposition of a node *u* of a task network *w* by a method *m* under a substitution σ. The formal definition of this concept is somewhat complicated, but the idea expressed by it can be intuitively understood: we replace *u* in *w* with a copy of *subtasks*(*m*), and all ordering constraints applied to *u* are then applied to each node in the copy of *subtasks*(*m*).

One of the main difficulties of the decomposition process lies in the fact that there may be more than one possible candidate to be the first task of the set of subtasks (according to the preconditions to be satisfied), making it necessary to establish a set of alternative task networks – one for each of the possible candidates.

In an STN planning process, we are always interested in methods that are applicable to the current state and relevant to a task we want to accomplish.

2.2.3.1.4. Domain, Problems and Solutions

In order to use an automated planner in the search for a solution to an STN planning problem, we have to formally specify how the problem will be represented and what can be considered to be a solution.

Before defining the problem, we need to describe the world to which it applies – or, at least, what matters in this world with respect to the problem. This description corresponds to an *STN planning domain*, a pair

$$D = (O, M),$$

where O is the set of operators and M is the set of methods associated with the domain in question. If all methods in M are totally ordered, D is a *total-order planning domain*.

We can have several different problems on the same domain. What will differentiate one from the other are two factors: its initial state and the task (or set of tasks) to be accomplished. Thus, an *STN planning problem* is a 4-tuple:

$$P = (s_0, w, O, M),$$

where s_0 is the initial state, w is a task network that we call the *initial task network*, and O and M are, respectively, the set of operators and the set of methods of the problem domain D. Again, we can extend the concept of total ordering at a higher level of abstraction: if both w and D are totally ordered, then we say that P is a *total-order planning problem* (otherwise it is a *partial-order planning problem*).

Having defined the problem domain and, with the help of this concept, the problem itself, we still have to define what it means for a plan $\pi = \{a_1, ..., a_n\}$ to be a solution to this problem $P = (s_0, w, O, M)$ – which is equivalent to say that the plan π *accomplishes* w. Intuitively, we can say that the sequence of actions expressed by π is the result of the successive decompositions of w until it reaches primitive tasks, which are then instantiated in actions that follow an ordering consistent to the order of the original tasks.

Formally, for a planning problem $P = (s_0, w, O, M)$, we say that $\pi = \{a_1, ..., a_n\}$ is a solution to P if any of the following three cases occur:

- w is empty and the plan π is also empty (i.e. n = 0).
- there is a node u in w without predecessors associated with a primitive task t_u, the action a_1 of the plan π is applicable to t_u in s_0 and, recursively, the plan $\pi = \{a_2, ..., a_n\}$ is a solution to:

  $$P' = (\gamma(s_0, a_1), w - \{u\}, O, M),$$

  which is the planning problem produced by executing the first action of π and the removal of the corresponding node u from the original task network w.
- If there is a node u in w without predecessors associated with a non-primitive task t_u, there is an instance of a method m in M that is relevant to t_u and applicable in s_0, and there is a task network w' belonging to $\delta(w, u, m, \sigma)$ such that, recursively, the plan π is a solution for:

  $$P' = (s_0, w', O, M).$$

To introduce the procedures for solving STN problems, we will use the *Dock-Worker Robots* (*DWR*) domain. In this simple yet nontrivial domain, we have different *locations, robots* (carts that can be loaded and transport only one container at a time and move to adjacent locations), *cranes* (that can manipulate containers within their same location), *piles* (fixed areas with a *pallet*, denoted as pallet, at the bottom), and *containers* to be manipulated.

The topology of the domain is specified using the following predicates:

- adjacent(l, l'): location l is adjacent to location l'.
- attached(p, l): pile p is attached to location l.
- belong(k, l): crane k belongs to location l.

The current configuration of the domain is specified using the following predicates:

- occupied(l): location l is already occupied by a robot.
- at(r, l): robot r is currently at location l.
- loaded(r, c): robot r is currently loaded with container c.
- unloaded(r): robot r is not loaded with a container.
- holding(k, c): crane k is currently holding container c.

- empty(*k*): crane *k* is not holding a container.
- in(*c*, *p*): container *c* is currently in pile *p*.
- on(*c*, *c'*): container *c* is on some container *c'* or on a pallet within a pile.
- top(*c*, *p*): container *c* sits on top of pile *p*. If pile *p* is empty, this will be denoted as top(pallet, *p*).

Finally, there are five possible actions in the DWR domain:

- *Move* a robot *r* from some location *l* to some adjacent and unoccupied location *l'*.
- *Take* a container *c* with an empty crane *k* from the top of a pile *p* located in the same location *l*.
- *Put* down a container *c* held by a crane *k* on top of a pile *p* located in the same location *l*.
- *Load* with a container *c* held by a crane *k* an unloaded robot *r* that is within the same location *l*.
- *Unload* a container *c* with empty crane *k* from a loaded robot *r* within the same location *l*.

In our problem, we have two locations (*l1* and *l2*), two cranes (*crane1* and *crane2*), a robot (*r1*), two containers (*c1* and *c2*), and four piles (*p11*, *p12*, *p21* and *p22*). The initial state is shown in Figure 2.3. Our goal is to, using the robot *r1*, move containers *c1* and *c2* to the location *l2*.

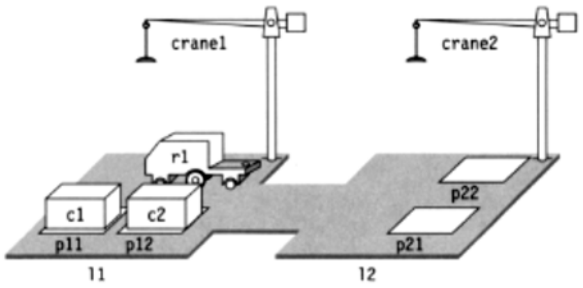

Figure 2.3 – Initial state for our DWR problem (Ghallab *et al*. 2004).

Next we shall see how this problem can be solved using a total-order planning and, afterwards, how we can increase the number of possible solutions

(increasing the chances of finding better solutions) with the use of partially ordered methods.

2.2.3.2. Total-Order STN Planning

One way to solve the problem of Figure 2.3 is through the *TFD* (*Total-order Forward Decomposition*) procedure, used for solving total-order problems. What this procedure does applies directly to the three cases, presented before, for finding a solution to an STN planning problem. TFD is both sound and complete.

Let us consider, as an example, a number of methods that, for the sake of clarity, are presented in a format somewhat different from the 4-tuple format described previously; this set of methods will be called *M1*:

transfer2(c_1, c_2, l_1, l_2, r) *; method to transfer c_1 and c_2*
  task: transfer-two-containers(c_1, c_2, l_1, l_2, r)
  precond: *; none*
  subtasks: ⟨transfer-one-container(c_1, l_1, l_2, r),
  transfer-one-container(c_2, l_1, l_2, r)⟩

transferl(c, l_1, l_2, r) ; *method to transfer c*
  task: transfer-one-container(c, l_1, l_2, r)
  precond: *; none*
  subtasks: ⟨setup(c, r), move-robot(r, l_1, l_2), finish(c, r)⟩

movel(r, l_1, l_2) *; method to move r if it is not at l_2*
  task: move-robot(r, l_1, l_2)
  precond: at(r, l_1)
  subtasks: ⟨move(r, l_1, l_2)⟩

move0(r, l_1, l_2) *; method to do nothing if r is already at l_2*
  task: move-robot(r, l_1, l_2)
  precond: at(r, l_2)
  subtasks: () *; i.e. no subtasks*

do-setup(c, d, k, l, p, r) *; method to prepare for moving a container*
  task: setup(c, r)
  precond: on(c, d), in(c, p), belong(k, l), attached(p, l), at(r, l)

subtasks: ⟨take(*k*, *l*, *c*, *d*, *p*), load(*k*, *l*, *c*, *r*)⟩

unload-robot(*c*, *d*, *k*, *l*, *p*, *r*) *; method to finish after moving a container*

task: finish(*c*, *r*)

precond: attached(*p*, *l*), loaded(*r*, c), top(*d*, *p*), belong(*k*, *l*), at (*r*, *t*)

subtasks: ⟨unload(*k*, *l*, *c*, *r*), put(*k*, *l*, *c*, *d*, *p*)⟩

A formal description of this planning problem should also provide:

- a description of the initial state shown in Figure 2.3;
- an initial task, directly associated with the description of the problem – in this case, the task *transfer-two-containers*(*c1*, *c2*, *l1*, *l2*, *r1*);
- the set of methods *M1*;
- a set of operators including the operators *take*, *load*, *move*, *unload* and *put*.

The resolution of this problem would eventually produce the decomposition tree of Figure 2.4. The initial task *transfer-two-containers*(*c1*, *c2*, *l1*, *l2*, *r1*) is recursively decomposed into subtasks, each of which is instantiated inheriting the substitutions of the task it originated from; the methods selected for generating these subtasks must be relevant to the task, and their preconditions must hold in the current state. The decomposition process continues until it reaches primitive tasks that can be instantiated directly by operators of the DWR domain (these tasks correspond to the shaded rectangles of Figure 2.4).

2.2.3.3. Partial-Order STN Planning

Suppose that, in the DWR problem presented above, the robot *r1* can handle more than one container simultaneously. We could have a better solution, as shown in Figure 2.5, where two tasks are inserted so as to make *r1* transport two containers at once.

In total-order planning, such interleaving is not possible because all methods are totally ordered, hence a task's subtask cannot be accomplished before the predecessor task is fully accomplished. One option would be to change our set of methods *M1* so as to have a method to load all containers, move them only after that, and finally unload both of them at the same time.

Another convenient option to ensure more flexibility is the use of partial-order planning, with partially ordered methods. It allows a subtask of a task *t1* to be executed before another task *t2* is fully accomplished, provided that there be no ordering constraint between *t1* and *t2*.

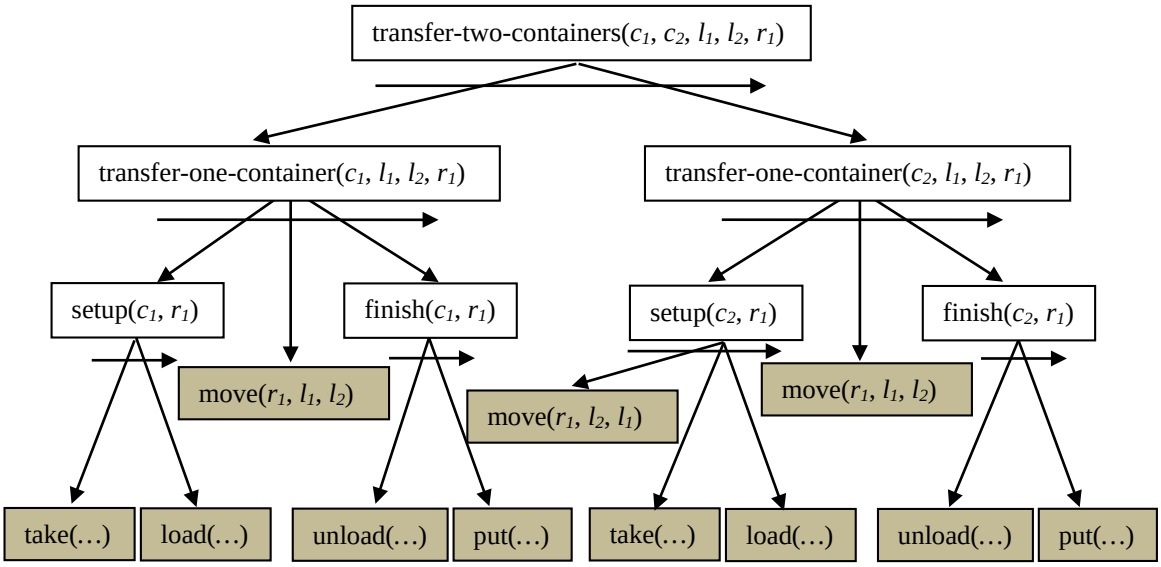

Figure 2.4 – Decomposition tree for the problem of Figure 2.3, utilizing the set *M1* of totally ordered methods. Adapted from (Ghallab *et al*. 2004).

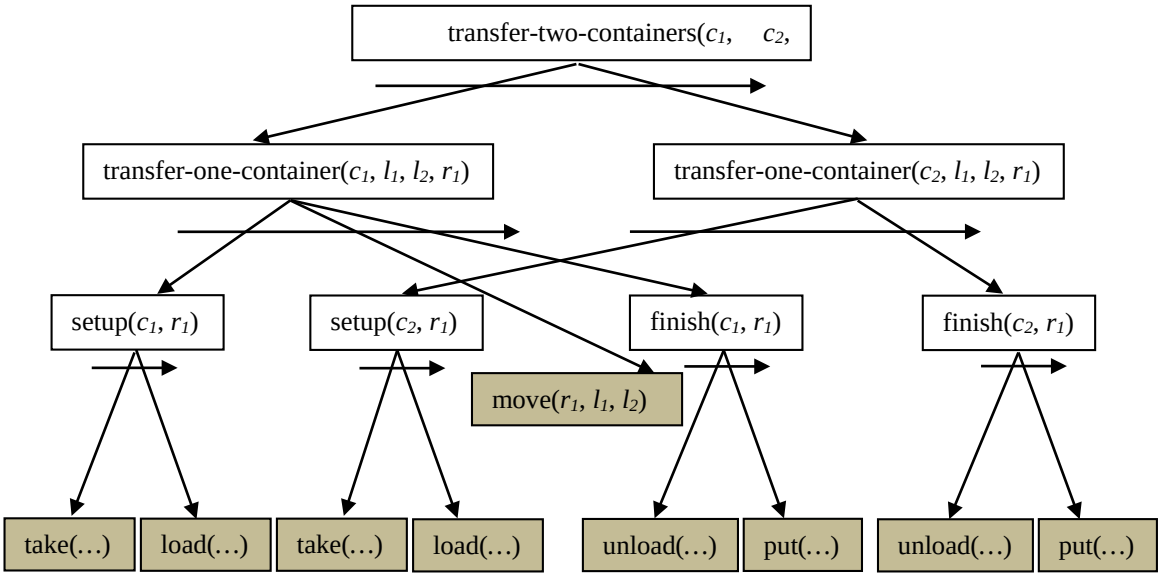

Figure 2.5 – Decomposition tree for the problem of Figure 2.3, with the possibility of *r1* carrying more than one container at once; the use of partial ordering permits the subtasks to be interleaved. Adapted from (Ghallab *et al*. 2004).

A set of methods *M2* enabling such behaviour is described below:

transfer2(c_1, c_2, l_1, l_2, r) *; method to transfer c_1 and c_2*

task: transfer-two-containers(c_1, c_2, l_1, l_2, r)

precond: *; none*

subtasks: u_1 = transfer-one-container(c_1, l_1, l_2, r),

u_2 = transfer-one-container(c_2, l_1, l_2, r),

transferl(c, l_1, l_2, r) *; method to transfer c*

task: transfer-one-container(c, l_1, l_2, r)

precond: *; none*

rede: u_1 = setup(c, r), u_2 = move-robot(r, l_1, l_2),

u_3 = finish(c, r), {(u_1, u_2), (u_2, u_3)}

move l(r, l_1, l_2) *; method to move r if it is not at l_2*

task: move-robot(r, l_1, l_2)

precond: at(r, l_1)

subtasks: ⟨move(r, l_1, l_2)⟩

move0(r, l_1, l_2) *; method to do nothing if r is already at l_2*

task: move-robot(r, l_1, l_2)

precond: at(r, l_2)

subtasks: ⟨⟩ *; i.e. no subtasks*

do-setup(c, d, k, l, p, r) *; method to prepare for moving a container*

task: setup(c, r)

precond: on(c, d), in(c, p), belong(k, l), attached(p, l), at(r, l)

network: u_1 = take(k, l, c, d, p), u_2 = load(k, l, c, r), {(u_1, u_2)}

unload-robot(c, d, k, l, p, r) *; method to finish after moving a container*

task: finish(c, r)

precond: attached(p, l), loaded(r, c), top(d, p), belong(k, l), at (r, t)

network: u_1 = unload(k, l, c, r), u_2 = put(k, l, c, d, p) , {(u_1, u_2)}

The *PFD* (*Partial-order Forward Decomposition*) procedure, shown in Figure 2.6, allows the generation of plans that take advantage of the additional flexibility, attained by using partial-order planning instead of total-order planning. Like TFD, PFD directly implements the definition of a solution to an STN

planning problem, testing each of the three possible situations. Furthermore it is likewise sound and complete.

```
PFD(s, w, O, M)
    if w = ∅ then return empty plan
     nondeterministically choose any u ∈ w that has no
                                        predecessors in w
    if t_u is a primitive task then
        active ← {(a, σ) | a is a ground instance of an
                                        operator in O,
                           σ is a substitution such that
                                        name(a) = σ(t_u),
                           and a is applicable to s}
        if active = ∅ then return failure
        nondeterministically choose any (a, σ) ∈ active
        π ← PFD (γ(s, a), σ(w – {u}), O, M)
        if π = failure then return failure
        else return a.π (i.e. π with a included)
    else
        active ← {(m, σ) | m is a ground instance of a
                                        method in M,
                           σ is a substitution such that
                                        name(m) = σ(t_u),
                           and m is applicable to s}
         if active = ∅ then return failure
         nondeterministically choose any (m, σ) ∈ active
         nondeterministically choose any task network
                                        w' ∈ δ(w, u, m, σ)
         return(PFD(s, w', O, M))
```

Figure 2.6 – PFD procedure for partial-order STN planning. Adapted from (Ghallab *et al*. 2004).

2.2.3.4. HTN Planning

In STN planning, we have two types of constraints for each method: ordering constraints and preconditions. We express ordering constraints by edges in the task network – or, in the case of totally ordered methods, by the tasks ordering in the subtasks list. As for preconditions, we have no resource to keep track of them; in fact, what we do is to coerce them to occur at the right moment, through the appropriate specification of the task networks.

This approach only seems to work adequately with forward-decomposition procedures, such as TFD and PFD. Still, there may be cases where one wants even

greater flexibility, allowing to perform the decomposition process in a different succession (backwards, for example), which makes it necessary to adopt some other approach.

HTN planning is a generalization of STN planning that allows more freedom for the construction of task networks, through the use of a monitoring mechanism that represents the unenforced restrictions. In HTN planning, a task network is represented by a pair $w = (U, C)$, where U is a set of task nodes, and C is a set of constraints that must be satisfied by any plan that is proposed as a solution to the analyzed planning problem. These constraints refer, basically, to the ordering between the task nodes and the establishment of literals before, after or between certain sets of task nodes.

An *HTN method* consists of a 4-tuple:

$$m = (name(m), task(m), subtasks(m), constr(m))$$

in which *name*(*m*) and *task*(*m*) have the same meaning as in STN planning, and (*subtasks*(*m*), *constr*(*m*)) is a task network for HTN planning.

Here is another set of methods applicable to the problem of Figure 2.3, this time using HTN methods:

transfer2(c_1, c_2, l_1, l_2, r) *; method to transfer* c_1 *e* c_2
- task: transfer-two-containers(c_1, c_2, l_1, l_2, r)
- subtasks: u_1 = transfer-one-container(c_1, l_1, l_2, r),
  u_2 = transfer-one-container(c_2, l_1, l_2, r)
- constr: $u_1 \prec u_2$

transferl(c, l_1, l_2, r) *; method to transfer c*
- task: transfer-one-container(c, l_1, l_2, r)
- subtasks: u_1 = setup(c, r), u_2 = move-robot(r, l_1, l_2), u_3 = finish(c, r)
- constr: $u_1 \prec u_2$, $u_2 \prec u_3$

move l(r, l_1, l_2) *; method to move r if it is not at* l_2
- task: move-robot(r, l_1, l_2)
- subtasks: u_1 = move(r, l_1, l_2)
- constr: before({u_1}, at(r, l_1))

move0(r, l_1, l_2) *; method to do nothing if r is already at* l_2
- task: u_0 = move-robot(r, l_1, l_2)
- subtasks: *; no subtasks*

constr: before($\{u_0\}$, at(r, l_2))

do-setup(c, d, k, l, p, r) *; method to prepare for moving a container*

task: setup(c, r)

subtasks: u_1 = take(k, l, c, d, p), u_2 = load(k, l, c, r)

constr: $u_1 \prec u_2$, before($\{u_1\}$, on(c, d)), before($\{u_1\}$, attached(p, l)), before($\{u_1\}$, in(c, p)), before($\{u_1\}$, belong(k, l)), before($\{u_1\}$, at(r, l))

unload-robot(c, d, k, l, p, r) *; method to finish after moving a container*

task: finish(c, r)

subtasks: u_1 = unload(k, l, c, r), u_2 = put(k, l, c, d, p)

constr: $u_1 \prec u_2$, before($\{u_1\}$, attached(p, l)), before($\{u_1\}$, loaded(r, c)), before($\{u_1\}$, top(d, p)), before($\{u_1\}$, belong(k, l)), before($\{u_1\}$, at (r, t))

Both the domain and the HTN planning problem definitions are identical to those of STN planning; thus the only difference lies in the fact that, in this case, the set of methods consists of HTN methods.

A plan $\pi = \langle a_1, a_2, ..., a_k \rangle$ is a solution to a planning problem P if there is a ground instance (U', C') generated by the decomposition of the initial task network w = (U, C) with a total ordering $\langle u_1, u_2, ..., u_k \rangle$ of the nodes of U' so that all the constraints are satisfied.

HTN planning procedures must meet two basic objectives: instantiate operators and decompose tasks. There are numerous ways to accomplish these objectives, and hence there are many possible HTN planning procedures. An example is the *Abstract-HTN* procedure (Ghallab *et al*. 2004), which is generic enough to allow several of these possibilities. For example, it can accommodate HTN versions of both TFD and PFD.

2.2.4. Nondeterministic HTN Planning

An ideal in terms of planning is to unite the more *realistic approach* of nondeterminism to the *efficiency* provided by HTN planning. Kuter & Nau (2004)

present a technique to be employed with planners using forward chaining in deterministic domains, which they adapt to work with nondeterminism. The goal is to take advantage of the efficiency that certain deterministic planners gain by avoiding non-promising states (as it occurs with HTN), without losing neither soundness nor completeness. This technique was applied with the SHOP2 algorithm, generating a nondeterministic version called ND-SHOP2. Experimental comparisons between ND-SHOP2 and MBP showed that, in some cases, while the CPU time of the latter grows exponentially with the problem size, the CPU time of the former still grows in polynomial-time.

Other work in this direction is the Yoyo planning algorithm (Kuter *et al*., 2009), which uses an HTN-based mechanism to restrict its search (like ND-SHOP2) and BDD representation for reasoning about sets of states and state transitions. Like the MBP planner, Yoyo makes use of model checking (implemented with BDDs) to handle nondeterminism; additionally, its plans are also presented as policies. Yoyo is sound and correct and, when compared experimentally to both ND-SHOP2 and MBP, consistently proved to be much faster than at least one of the two and almost always faster than both.

Hermann (2008) proposes the combination of hierarchical planning and planning under Knightian uncertainty (i.e. without considering the distribution of the probabilities of action outcomes) (Knight 1921). The proposed strategy is to apply the transformation process presented by Kuter & Nau (2004), here called *ND-transformation*, to an HTN planning algorithm. A version in the Haskell language (Jones & Hughes 2002) of the SHOP planner (Nau *et al*. 1999), called HSHOP, was submitted to the ND-transformation, resulting in a hierarchical planning system for nondeterministic systems called ND-HSHOP. This planner was experimentally compared to MBP and HMBP (a planner based on MBP, but not using BDDs), overcoming both in the tests performed.

2.3. Can Machines Feel?

In his seminal paper “Computing Machinery and Intelligence”, Turing (1950) proposed the famous question: “Can machines think?” In a time when electronic computers were still a widely advertised but poorly understood new

technology, sometimes popularly described as “giant” or “electronic” brains (Berkeley 1949; Suominen, 2011), it is no wonder that such question was raised by that pioneering computer scientist. In order to clarify his question, Turing remarks that it is fundamental to precisely define the boundaries of what the words “think” and “machine” really mean and, after pointing how difficult such framing is, he suggests a different question, related to the first one but deprived of its ambiguity.

Such question will come with the aid of the so-called “Imitation Game”, which became very well known in Computer Science literature as the *Turing test* (Figure 2.7). In this game, a man and a machine (designed to perform indistinguishably from a human) answer questions posed by a judge (another human) who, based on their answers, must tell which of them is the machine and who is the human being. The participants are separated from each other, so none of them can be seen by the others and, in order to avoid any hint from sensory expression (involving e.g. voice intonation), all communication is made via text (preferently via electronic means). Based on this test, Turing suggests the question that will replace the first: - Can computers conceivably do well in the imitation game?

This line of questioning is affected by differences in the definition of artificial intelligence, reflecting what they are concerned with. We can see the definitions varying along two main dimensions: one is related to whether *thought processes* and *reasoning* or just *behaviour* must be taken into consideration. The other dimension comes from measuring the performance of artificial intelligence systems against either *human performance* or *rationality* (as an ideal concept of intelligence). We have, as a result, four different approaches: *acting humanly*, *thinking humanly*, *acting rationally* and *thinking rationally* (Russel & Norvig 2003). Historically, all four approaches have been followed and brought contributions to AI.

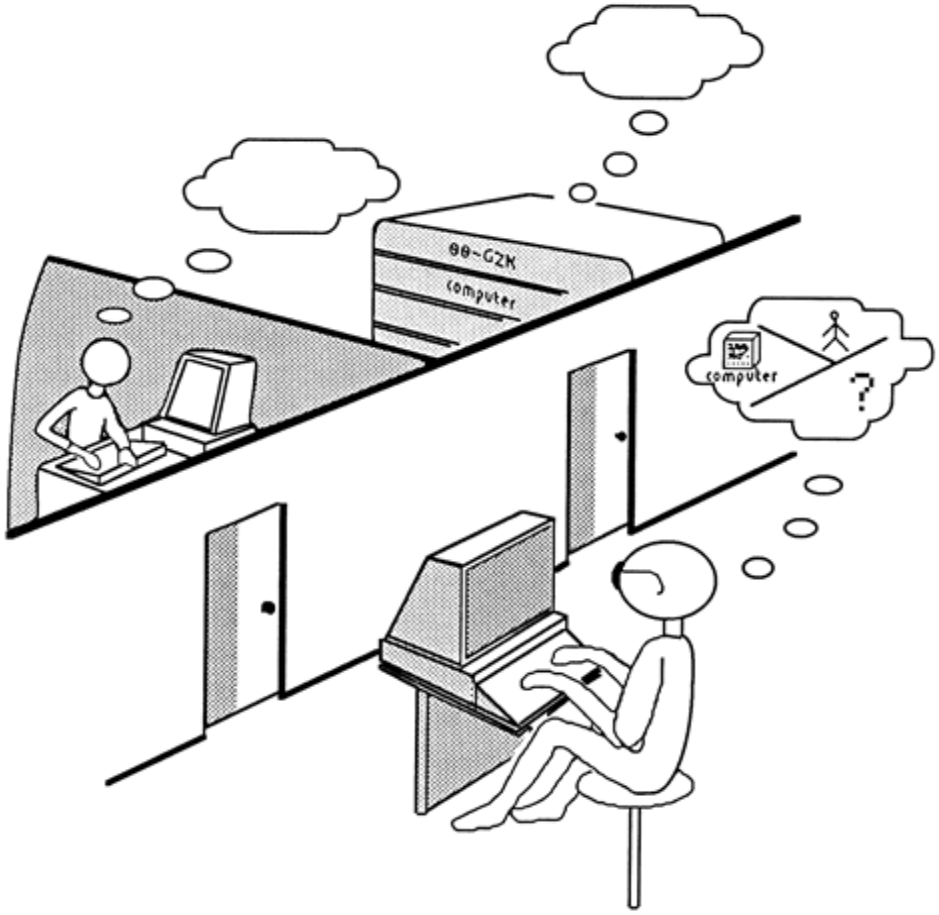

Figure 2.7 – A depiction of the Turing Test (Anthony 2014).

2.3.1. Machines That Act Like Humans

Turing's test clearly belongs to the first group, *acting humanly*, as it proposes to regard intelligence on the basis of an "imitation game" where the computer tries to pose as a human being. However, such imitation, in order to be successful, must, to some extent, contemplate the understanding and simulation of *emotions*. In fact, emotions are a profound aspect of human behaviour and, though the Turing test is designed to remove sensory expression from the game, emotions can still be perceived in text, as well as be elicited by its content and form (Aristotle 1960). As we can see, a machine will not be able to pass the Turing test unless it is also capable of perceiving and expressing emotions (Picard 1995).

The importance of emotions to successfully reproduce human intelligence through behaviour gave birth to a branch of AI named *affective computing*. Starting with an inaugural paper later followed by a book, Picard (1995, 1997)

introduced some issues to be addressed aiming not only at more convincing human simulations, but also at improving the interaction between humans and machines. Some possible applications were also suggested, such as gathering responses from the audience during a film screening, obtaining consumer feedback as a person interacts with a software product, or changing players' avatars according to their emotions. Many of these suggestions have been used since then; for example, a nice implementation of the changing avatars idea is reported in (Hudlicka 2008).

2.3.2. Machines That Think Like Humans

Intelligence and emotion have traditionally been regarded as opposite poles. For example, the popular Myers-Briggs personality-type indicator (Myers *et* al. 1998) – a system that, inspired by the typological theories proposed by Jung (1971), measures psychological preferences in how people perceive the world and make decisions – expresses this polarization very well: among its four axes, there is one with the labels "Thinking" (T) and "Feeling" (F) at the opposite endpoints.

Given that software developers and computer scientists tend to be biased toward the "T" side of this axis – a side that expresses little interest on emotions as compared to logical reasoning – it is not surprising that affective aspects have been marginalized for a long time in the attempts to construct models of intelligence (Picard 1997).

However, emotions are an important aspect of decision-making and thinking in general. Old expressions like "think with your head, not with your heart" show that this second role, acknowledged with noticeable reluctance, is generally considered an undesired alternative to rational thinking. Figure 2.8 shows a humorous depiction of the dichotomy.

Psychological studies have been giving a more positive view on the role of emotion. Gardner (1983) introduced the idea of *multiple intelligences*, which include the mostly emotional aspects of *interpersonal* (regarding the capacity to understand the motivations of other people) and *intrapersonal* (regarding the capacity to understand oneself) intelligences. Two years after that, the term *emotional intelligence* was employed by Payne (1985), but it only became widely

known one decade later, when Goleman (1995) published his best-seller “Emotional Intelligence – Why it can matter more than IQ”. Emotional intelligence (EI) – the ability to monitor one's own and other people's emotions and to use emotional information to, ultimately, guide thinking and behaviour (Coleman 2008) – has, since then, become very popular, assigning a more “intelligent” role to emotions. Sharing this point of view, artificial intelligence pioneer Marvin Minsky (2006) stated that emotion is “not especially different from the processes that we call ‘thinking’".

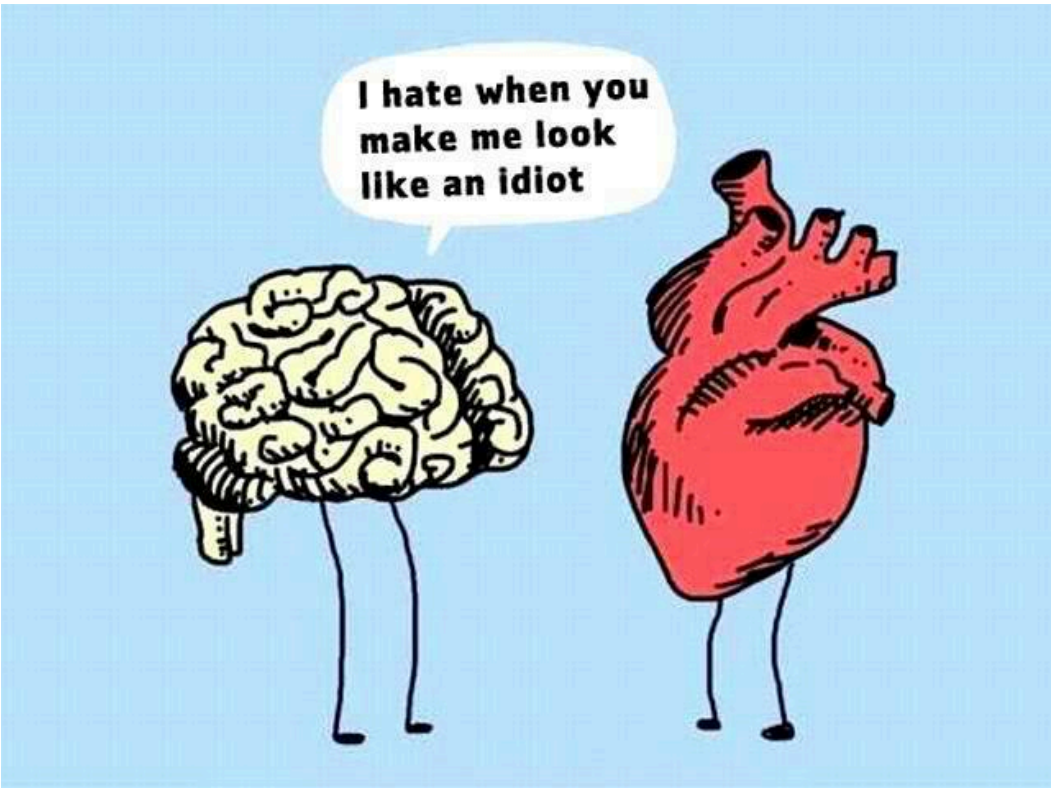

Figure 2.8 – The “head” vs. “heart” conflict (Hammerton 2014).

3
The Art and the Science of Telling Stories

In this chapter, we delve into the main area of our work, *interactive storytelling*. Starting from a brief exposition about traditional storytelling and its importance in our culture, we review some theories about how to compose a coherent and interesting story and to the first attempts to create computer-supported interactive narratives. We shall survey some of these attempts, as well as the technologies they used – with a special attention to those working with planning algorithms, user models and the representation of character personality and emotions. To better understand this latter aspect, we will take a leap into an area outside computer science, seeking some help from psychology studies about personality types, so as to be able to specify plausible personality models.

3.1.
Storytelling in Human Culture

We tell stories since we started communicating. In fact, storytelling can be considered a natural, almost inevitable consequence of human evolution (Crawford 2012). Through the use of words and images, humankind has used narratives as an instrument of entertainment, cultural preservation, education and as a means for transmitting moral values. We tell stories even before we started writing, through oral communication, gestures and rock art (as found in many prehistoric caves). Also, the practice of storytelling is universal, being present in completely unrelated cultures all around the world.

Why do humans transmit information with the aid of stories, instead of expressing directly the plans of action embedded in those stories? One possible answer can be given when we analyze how our mind works. Crawford (2012), based on the theory of multiple intelligences from psychology (Gardner 1983, 1999) and the notion of modularity of mind from cognitive science (Fodor 1983), proposes a model where four basic mental modules are highlighted: visual-spatial, social reasoning, environmental knowledge and language. Also, these modules

can be categorized into two different mechanisms of thinking: the dominant and more natural pattern-based thinking, and the more recent (in evolutionary history) sequential thinking. The older visual-spatial and social reasoning modules are totally dependent on pattern-based thinking, whereas environmental knowledge relies on a mixture of both mechanisms; and language uses sequential thinking only.

Still according to this model, different aspects of human culture are produced as a result of the interactions between these modules. For example, the questions the environmental knowledge module produced out of our curiosity about the natural phenomena, combined with the social reasoning module, would have devised an answer under the form of anthropomorphic deities. This might serve to explain the erratic behaviour of such phenomena, which would be due to personality idiosyncrasies of those all too powerful human-like entities. Worshipping the gods and having especially trained people to communicate with them (druids, shamans, priests) would, then, be natural consequences of this social approach to natural events, giving us all sorts of *myths*.

Similarly, the same questions coming from the environmental knowledge, when combined with the sequential thinking of the language module, would result in *science*. As sequential thinking is not as natural as pattern-based thinking, it would help to explain the longer time it took for us to find scientific explanations to natural phenomena once deemed as the direct consequence of the will of capricious deities. From the standpoint of this model, storytelling would be viewed as a combination of the language and the social reasoning modules. The interactions between these modules are represented in Figure 3.1.

Now, returning to our previous question, why transmit information through stories? Most of the information they contain seems related to social reasoning, concerning interpersonal behaviour: love, trustworthiness, faithfulness, perseverance, and so forth. Assuming that the mental module for social relationships relies on pattern recognition, the use of a sequential medium such as language just does not fit quite right (Crawford compares it to sending parallel data down a serial cable). According to this theory, storytelling works as a *reformatter* that converts one thinking mechanism into another one.

In fact, despite being presented as a linear sequence of events known as *plotline*, stories cannot be understood until the entire content is transmitted.

Reading 90% of a letter is generally not a hindrance to understanding its message; however, if we stop reading a book at 90% of its content, we are likely to miss the whole message and the picture does not seem complete. Stories are then better understood as patterns shown in a linear sequence (Crawford 2012).

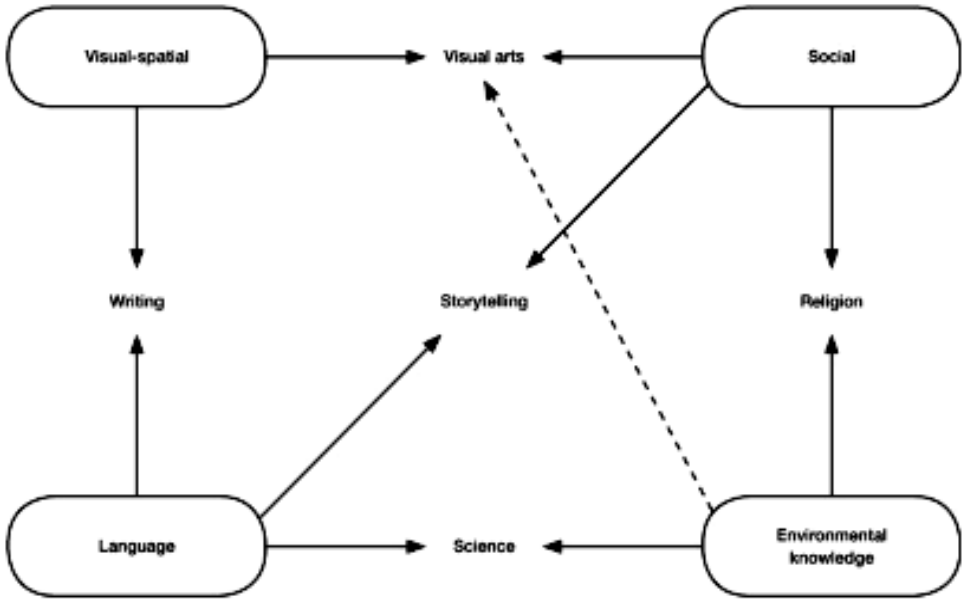

Figure 3.1 – Interactions between mental modules (Crawford 2012).

Anyway, there seems to be a general agreement that storytelling is a universal means for sharing and interpreting experiences, and that it can be used as a method to teach ethical values and cultural norms (Davidson 2004), passing on knowledge in a social context. Indeed, learning is most effective when it takes place in social environments providing authentic social clues about how knowledge is to be applied (Andrews & Hull 2009).

Human knowledge is based on stories and the human brain consists of cognitive machinery necessary to understand, remember, and tell stories (Schank & Abelson 1995). It is easier for us to remember facts as narrative structures in story form; hence, storytelling can also be seen as a foundation for learning and teaching in general.

3.2.
Traditional Story Types

Legends, myths, fables and folktales are traditional forms of transmitting information through storytelling. They are highly associated with an ancient purely oral tradition, and therefore may suffer changes as they are told and retold through the centuries. Some of them, however, have been written and thereby partially "standardized", in several cases by anonymous authors.

Fables are short tales, usually featuring anthropomorphized animals (real or mythical), plants or inanimate objects, with the main purpose to deliver a specific moral lesson (Howard 2013). Fables exist in every culture, but the most famous of them are attributed to the legendary Aesop, supposed to have been a Greek slave who lived around 560 BC. Many disagree whether or not he actually wrote each of the fables we identify as *Aesop's Fables'* today. A *parable* is similar to a fable, except that they feature human characters and usually have a religious tone.

Myths, as foreshadowed in the previous section, are stories meant to explain natural phenomena and answer questions about the human condition: origin and creation stories, stories about human life, death, and life after death. They usually include gods or other beings with supernatural powers and abilities. Myths are sacred, religious stories to the people who believe in them and take them as literal truth, such as, for instance, the myths about Thor have stood for the Old Norse people.

Legends are stories, typically semi-true, about people and their deeds, generally including some historic facts combined with a dose of fantasy or exaggeration. They usually involve heroic characters or wondrous places, and often encompass the spiritual beliefs of the culture wherefrom they originate.

Two examples of notable legends are *King Arthur* and *Robin Hood*. Certain scholars claim that King Arthur was inspired on some British leader who would have lived around the 5th or 6th centuries (Higham 2002), but the stories about the Knights of the Round Table and Merlin the Magician are undisputedly no more than fiction. The point of the Arthurian legend was that the predestined king and his knights defended their people and their land. Similarly, there is modern academic opinion in favour of the thesis that Robin Hood's legend is partially based on a historical person, although there be no consensus about his actual

identity (Ibeji 2011). But no scholar affirms the existence of factual evidence that he lived in a forest with a band who robbed from the rich to give to the poor – but, on the other hand, being a hero who helps people in need is definitely a key factor for the survival of the legend.

A *folktale* is a popular story that was passed on in spoken form, from one generation to the next. Usually the author is unknown and there are often many variants of the tale. The term "folktale" is often used interchangeably with fable, since folktales can propose a moral lesson at the end, although generally their main characters are people rather than speaking animals (Howard 2013). Folktales include fairy tales, stories with magical creatures and other fantastic elements, usually with a happy ending.

3.3. Story Requirements

What comprises a story? Not any sequence of events can be considered a story. Some special cement is needed to bind those events together in a way that we can see a clear picture in the end. Crawford (2012) enumerates some criteria that can help us with this question. First thing: stories are about *people*. Even when this reference is symbolic, as in many fables where the protagonists are animals, we can clearly see that they represent people, many times even acting and behaving just as humans. Inanimate objects, important as they can be in such stories (like magical artifacts), never play an actual role, being just tools used by the real characters – i.e. *people*.

Secondly, stories need some sort of *conflict*. Sometimes conflicts are direct and violent, as in *Star Wars* (Figure 3.2) and *Lord of the Rings*; sometimes they are more subtle, like social or symbolic conflicts (which generally lead to more sophisticated stories), but they must show up somehow.

Finally, stories are about *choices* made by the characters. In many cases, there is a clear key decision that defines the direction taken by the story – for example, Guinevere's decision to act on her love for Sir Lancelot in many versions of the Arthurian legends, and Neo's decision to take the red pill and learn what the world really is in the acclaimed 1999 sci-fi film *The Matrix*. However,

other less dramatically impactful choices throughout the story are also important to establish characters and define their roles along the plotline.

Figure 3.2 – Explicit conflict in a narrative: a lightsaber duel in the 1980 film *Star Wars Episode V: The Empire Strikes Back.*

3.4. How to Make a Story?

If we want to make a system capable of creating stories dynamically, this system needs some model that indicates how to proceed. Actually, creating interesting stories is a hard task even for talented human beings. As a consequence, there are many different studies since the times of ancient Greece about the structural components of narratives and how they are composed. These studies can be useful in interactive storytelling, since they provide models that can, at least theoretically, be used to tell a computer how to make a story with some quality. Indeed, the elements and structures characterized in those models have been used for millennia to create interesting and aesthetically pleasant stories. This section introduces some of the best known and widely utilized studies on this subject.

3.4.1. Aristotle's *Poetics*

Aristotle's *Poetics* (Aristotle 2000), written in the 4th century BC, is the earliest surviving work about dramatic and literary theory, addressing the

fundamental principles on which stories are based. Despite his main focus on tragedy, his comments are applicable to other areas and his work still continues relevant today (Karlsson 2010).

Aristotle divided tragedy into six parts, listed here in decreasing order of importance, according to the author's criteria:

- *Mythos* (plot) – The most important element in Aristotle's work, plot refers to the structure of actions. Actions should follow a logical chain. And they bring more satisfaction if they come about by surprise.
- *Ethos* (character) – In a perfect tragedy, character supports the plot, the cause-and-effect chain of actions being connected and triggered by personal motivations.
- *Dianoia* (thought) – Explanations about the characters or story background.
- *Lexis* (diction) – Related to the quality of speech.
- *Melos* (melody) – Refers to the chorus.
- *Opsis* (spectacle) – The least important element in this list, it refers to the visual aspects of the play.

Aristotle (2004) considers plot and character the most important aspects of a story, with interests of the characters perfectly integrated with the sequence of events that produces the plot as a whole. As we will see later in this chapter, these ideas can be associated with the main approaches used for generating stories in interactive storytelling. He also defines tragedy as "an imitation of an action that is serious" and states that well-formed plots must not begin or end at random, but rather under strictly established conditions.

3.4.2. Structuralism and Story Levels

Structuralism is the theory that elements of human culture must be studied and understood in terms of their relationship to some pre-existing structure. In literary theory specifically, structuralism relates literary texts to some structure on the basis of which they can be characterized as members of a class, be it the conventions of a particular genre, a model of a universal narrative structure, or a

system of recurrent patterns or motifs (Barry 2002). Structuralism argues that there must be a structure in every text; so that everything that is written appears to be governed by specific rules (Selden *et al.* 2005).

A major influence to structuralism as a whole was the movement that took place in Russia in the early 20th century and is known as *Russian formalism*. One of its leaders, Shklovsky (1991), proposed a division of stories in two different levels, called *fabula* and *syuzhet*. Fabula corresponds to the raw material of the story, the chronological sequence of events that composes the plot. *Syuzhet* is how these events are narrated, not necessarily in this same chronologic order – for example, through *flashbacks* or with the use of different points of view.

This concept was later explored by other scholars. Tomashevsky (1965), another Russian formalist, defined the structure of a narrative as resulting from the tension between fabula and *syuzhet*. He states that the *syuzhet* has its own structure, wherein coherence is guided by artistic needs such as suspense and curiosity, and not by time or causality constraints. Still according to him, in order to understand the story, one necessary mental step imposed to readers of a narrative text is to reconstruct the fabula in their mind from the *syuzhet* they receive.

Chatman (1978) takes a similar approach, also dividing stories in two levels: *story* and *discourse*. More recently, Bal (1997) expanded this division to three levels: *fabula*, *story*, and *text*. For her, *story* is the fabula narrated in a certain manner, and *text* is the presentation of the story in a particular medium, like a book or a movie. This approach seems more satisfactory than the previous proposed separation in two levels, given that the same story can be presented in different media – whose peculiarities may require that the story, and sometimes even the fabula, be submitted to extensive adaptations (unfortunately with a disfiguring result, in a number of cases).

3.4.3. Motifs and Tale Types

Folktales around the world and from all times share many common elements, themes and plots. Folklorists use the word m*otif* to name recurring elements with symbolic significance in a story, which can take, among others, the

form of an idea, an object, or a place (Karlsson 2010). Motifs are recognizable and universally repeated constituents of many folktales.

The folktales themselves are grouped into *tale types*, which provide a classification in terms of the entire narrative, determining similar structures that become visible when we abstract the particular, concrete elements of each specific narrative. A tale type is commonly defined by listing the sequence of motifs into which its basic underlying narrative can be decomposed.

The most widely adopted catalogue of tale types is the *Aarne-Thompson Types of the Folktale*. Created by Antti Aarne and later expanded by Stith Thompson (1961), it employs a numeric indexing criterion – the *AT number system*. Examples of their tale types are AT124 – Three Little Pigs; AT333 – Little Red Riding Hood; AT561 – Aladdin; and (to show that not all of them comprise internationally famous tales) AT2031C – The Mouse Who Was to Marry the Sun.

This system was not immune to criticism: some critics pointed a lack of variation and an excessive focus on oral tradition, omitting important tales in written form; others, stated that many "folktale complexes" not previously included in the index could be integrated without difficulty (Uther 2000). These issues led to the ATU system (Uther 2004), an extended reference system. Both the AT and the ATU indexes register tale types with their variants and origins, together with an analysis of their narrative pattern and lists of constituent motifs.

To better clarify the hierarchic classification into types and motifs, Thompson (1989) complemented the taxonomy with the *Motif-Index of Folk Literature*, a compilation of the traditional motifs cross-referenced with the tale types.

Thompson affirmed that the folktale motifs live on because they have been found attractive by generations of tale-tellers (Leach 1972). It gives us a hint on why they can be a useful tool for generating interesting stories. Indeed, Crawford (2005) suggests that this huge dataset could be used in interactive storytelling, mostly by providing a clever way to connect events according to the motifs involved. However, he acknowledges the complexity and difficulty in assembling all this data to all the possible conditions that should trigger or just enable these motifs.

3.4.4.
Morphology of the Folktale

Again back in the early twentieth century, another Russian scholar, Vladimir Propp (2003), strongly influenced by the work of the Russian formalists, analyzed about one hundred Russian folktales in his seminal work *Morphology of the Folktale* and figured out that these folktales contained certain typical actions of the participating characters, which he called *functions*. These functions are defined from the point of view of their importance to the unfolding of the action. Differently from the taxonomy of tale types and motifs, Propp's approach consisted in identifying the purpose of those elementary actions in the plot sequence that formed the tales (Karlsson 2010).

Propp designed a notation for the functions, permitting to represent the plots in a compact, codified form. His first concern resided in knowing *what* was done and not in *who* was doing it or *how*. In fact, although in each story different characters can figure, he claimed that all folktale characters could be resolved into seven broad character roles: the *hero*, the *villain* (the hero's antagonist), the *dispatcher* (who sends the hero off in his quest), the *helper* (who helps the hero in his quest), the *princess* (the hero's love interest), the *donor* (who prepares the hero and/or gives him a magic object) and the *false hero* (who takes credit for the hero's deeds). Accordingly he concluded that the names and attributes of the characters are variable, but the actions (functions) they play are constant. He also declared that there are a limited number of typical functions in all Russian (and not only Russian) folktales (31, to be precise), and that their sequence in the tales is always the same.

Crawford (2005) also proposes the use of Propp's functions in interactive storytelling, stating, however, that the system achieves generality only at the higher levels of abstraction, relegating the concrete manifestation of such abstractions to specific story components. Also, he points out that the scope of a Proppian system is not as large as that of the Aarne-Thompson (1961) catalogue, but adds that this is not necessarily bad since a shorter-sized problem results. For similar considerations, our interactive storytelling system **Logtell** relies strongly on Propp's work, as we will see later in this chapter.

3.4.5.
The Hero's Journey

Another scholar who elaborated a single scheme to fit stories in general was the acclaimed mythologist Joseph Campbell. The importance of his work can be measured by the fact that not only classic myths from many cultures follow its structure (for example, the stories of Osiris, Prometheus, and Buddha), but also because it influenced many modern successful stories, like the *Star Wars* saga (Larsen & Larsen 2002).

Campbell (1968), in his famous book *The Hero with a Thousand Faces*, dealt basically with the hero's figure and his journey in mythological stories, with the help of psychological studies for tracing the origin of the transformations that occur in the hero's mind. In fact, some psychology scholars believe that the rites of passage faced by the heroes in myths represent in some way the phase transitions observed in the human psyche (Karlsson 2010). As we know, Carl Jung followers associate the emergence of universal types and motifs, mythical or not, to the action of the so called *archetypes* of the collective unconscious, as remarked in (Furtado 2006) for instance.

By comparing dreams and numerous stories involving myth from different places and times, Campbell found many similarities among them. He then elaborated a theory according to which all the stories about myths in the world are in fact based on a single outline (Karlsson 2010). Campbell, being an admirer of novelist James Joyce, borrowed the term *monomyth* from Joyce's work (Campbell 1968). The monomyth has also become known as the *hero's journey*.

According to this scheme, the journey consists of a series of stages forming a cyclical diagram (Figure 3.3). Campbell describes 17 stages or steps along this journey. Very few myths contain all 17 stages — some contain many of them, while others contain just a few. Also, some myths may focus on only one of the stages, while others may deal with them in a somewhat different order. These 17 stages are divided into three sections: *Departure* (or *Separation*, the hero's adventure prior to the quest), *Initiation* (the hero's adventures along the way), and *Return* (the hero's homecoming with the gains achieved in the journey).

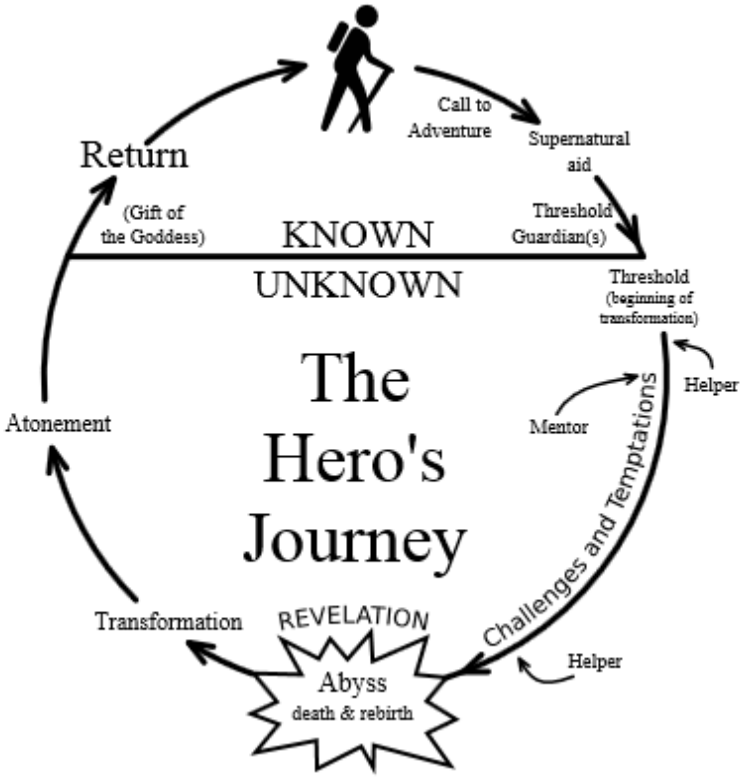

Figure 3.3 – Visual scheme of the monomyth.

3.4.6. Dramatic Situations

Another way to detect similarities in different stories is to categorize every *dramatic situation* that might occur in a story or stage performance. The dramatic situations describe what the story is about; for example whether it is a story of rescue, revenge, or disaster. Some researchers assume that it is possible to enumerate all the dramatic situations found in stories (Karlsson 2010).

George Polti (1945), a French writer, analyzed classical Greek texts, plus classical and contemporaneous French works, besides a handful of non-French authors. He created a list with *The Thirty-Six Dramatic Situations*, published in the book of the same name. Many of these categories are broken down into subcategories. The book also contains extended explanations and examples, in addition to a brief description and a set of character roles for each situation. Again, Crawford (2005) suggests that this taxonomy can be used as a basis for interactive storytelling.

3.5.
The Quest for Interactive Storytelling

Why make stories interactive? Huizinga (2001) argues that cultural practices are characterized by features of their game-like quality, and that humans can be defined as playful creatures, full of curiosity, and moved by love of diversion, explorations and wonder (Karlsson 2010). Roger Caillois [*apud* Huizinga 2001] also states that it is through play that civilization develops. So, particularly as an adjunct to storytelling, playing is also a fundamental aspect of human behaviour and culture. It then seems quite natural that, following a promising (though not unanimously adopted) trend of the entertainment industry, one would try to merge narrative and gameplay in interactive digital systems, which remains an intriguing and challenging field of research.

Actually, the idea of developing interactive stories predates the use of computers for this purpose. Back in 1941, Argentinian writer Jorge Luis Borges laid out the idea of interactive fiction in his short story *The Garden of Forking Paths*, where a Chinese spy for Germany living in Great Britain discusses his ancestor's ambition to write a vastly complex novel that is itself a labyrinth wherein every branching path is determined by the reader's choices (Hendrix 2011).

The concept of interactive fiction was later expanded when, in the 1970s, Edward Packard created the concept of the children's books *Choose Your Own Adventure* (Kraft 1981), whose success subsisted during the 1980s and 1990s. In those books (known as *gamebooks*), the stories are written from a second-person point of view, with the reader assuming the role of the protagonist. At certain points, the reader is called to make choices, moving to a different page depending on which option is chosen.

Eventually, still back in the 1970s (Meehan 1977), researchers started to make some interesting attempts to make computers generate stories. Numerous labels have been used for this field of study, e.g. “interactive story”, “interactive fiction”, and “interactive drama” (Crawford 2012), but we are adopting the most popular and widely used denomination *interactive storytelling* (IS) (Glassner 2004, Crawford 2012).

Regarding IS, it is important to recognize the aspects which distinguish it from digital games. At first glance, it is easy to mistake one for the other – after all, like games, interactive storytelling is played on a computer, is interactive, and is entertaining (Crawford 2012). However, there is a clear difference between the two forms of entertainment. Digital games are essentially about puzzle-solving, hand-eye coordination and resource management skills. Stories in games may come as an embellishment, but the focus of IS is on the plot itself, the characters and their relationships; IS is not "games with stories" (Crawford 2012). The stories presented in games are essentially a support for the challenges presented to the player and, more often than not, they are already defined before the game begins. In such cases the only role of the computer is to tell the story and its possible variations to the players as they proceed along the different phases of the game. As for interactive storytelling, its main purpose is to create attractive stories capable of surprising and captivating the spectators.

In both forms of entertainment, users are able to intervene in the ongoing stories in some way. But, usually, in interactive storytelling the interaction occurs only at specific points of the story and does not require much effort and attention from users (Lima 2014). Thus, interactive storytelling can be defined as a new form of digital entertainment that brings together techniques and tools for the creation, visualization and control of interactive stories through electronic means (Camanho et al. 2008).

Despite these differences, most of the work published in IS is aimed at applications to games. In fact, research on interactive storytelling may offer the opportunity for the development of new forms of digital games, and the recent commercial success of games like Heavy Rain (2010) reveals that integrating interactive stories into the gameplay creates a successful experience for players (Lima 2014).

3.5.1. The Free Will vs. Determinism Dilemma

Nowadays digital games have gained wide acceptance and became a research field on their own, but the field of IS remains unsettled, and still presents many open issues. Although different approaches have been tried to reach the goal

of integrating storytelling and entertainment, such as works of interactive fiction and adventure games, none has succeeded yet to fully match expectations (Karlsson 2010).

There are two important issues related to these systems that, as a rule, come into conflict with each other. On the one hand, we have the control exerted by the users, i.e. the autonomy granted to them in their interaction with the system (which favours a richer experience by providing the feeling of actual participation in the story); on the other hand, we have the coherence of the narrative, which allows the users to understand the causal relationship between the events during the story development (Riedl & Young 2006). Strategies to ensure consistency usually reach their goal by denying the users a greater degree of freedom in how they can change the course of the narrative, while those more focused on user control tend to generate threats to the coherence of the narrative. This creates a problem: how to allow the users to have a satisfactory degree of control, giving them a sense of immersion in the story through the ability to modify the environment presented by the interactive storytelling system, without affecting the coherence of the narrative? This is one of the main challenges of interactive storytelling – the difficult process of generating stories that are both coherent and interesting, allowing, at the same time, meaningful user interaction during their creation.

Crawford (2012) compares this dilemma with a closely related problem: the classical theological problem of free will versus determinism. Shortly put, the problem comes from the inconsistency that arose when Christian theologians confronted the Christian belief in God's omniscience and omnipotence with the free will of human beings. If everything that happens is acknowledged by God and part of His plan, it means that everything is deterministic, so how can humans have free will? On the other hand, if we have free will, God could not control or know what we would do, being neither omnipotent nor omniscient. This sometimes called "omnipotence paradox" has been the object innumerable responses, requiring the superior skill of a St. Thomas Aquinas for a consistent treatment (cf. http://www.ccel.org/a/aquinas/summa/FP/FP025.html#FPQ25A3THEP1).

The parallel with IS is clear: the conflict between determinism and free will is analogous to that between plot (coherence) and user interactivity. The author plays a role that is comparable, of course in a minor scale, to that attributed to

God in the theological analogy. Ironically, the parallel may sound like a warning to authors who do not hesitate to tax their readers' credulousness with patently impossible situations that they do not bother to justify. This godlike attitude is best exemplified in the Greek theater, when the author resorted to what the Romans called the *deus ex machina* mechanism, a crane with pulley and weights used to lower gods from above down to the stage whenever the author could imagine no other way to "untie the knot" of a play (Durant 1939).

Faced with this issue, there are two main options to the development of interactive storytelling. One, inspired by video games, is the *character-based* approach (Cavazza *et al*. 2002; Charles *et al*. 2001; Young 2001), in which autonomous agents, each with their own goals, interact with the environment and the user. The story itself is the result of these interactions, which depend on the decisions and actions of the agents and of the user. So the user gains a high degree of freedom, but it is harder to maintain the coherence of the narrative.

The other option to the development of interactive storytelling, influenced by literature and cinema, is the *plot-based* approach. It keeps a strong control over the flow of the action, preventing the user from straying too far from the original intention of the author. A framework is consequently established for the narrative with well-defined points, and the only permissible effect of user interaction on the narrative is to affect how the story reaches these predefined points (Grasbon & Braun 2001; Spierling *et al*. 2002). This approach is strongly inspired by the work of Propp (2003) with his 31 predefined functions, which are constrained to occur in certain typical sequences (Ciarlini *et al*. 2005).

To ensure narrative coherence, some systems using a character-based approach (in principle) make use of a *drama manager* that monitors the activities of the characters – and user agents – and keeps comparing them with those indicated in a graph plot (a partially ordered graph of events driving the course of the story). When there is the possibility of more than one event occurring, the drama manager determines which one will bring the most satisfactory result, and subtly manipulates the environment and autonomous agents to induce the occurrence of that event (Riedl & Young 2006). In this case, the strategy ceases to be purely character-based, also assuming characteristics of the plot-based approach.

Interestingly enough, we can see an analogy between these approaches and Aristotle's (2004) theories about the elements of a story. As noted before, the Greek philosopher considered plot the most important aspect of a story, followed by the characters, whose desires and intentions set in motion the cause-and-effect chain of events that define the plot – in this sense we can say that Aristotle's analysis is in agreement with the hybrid approach exposed above.

3.5.2. Coordinating Plot Needs with Character Intentions

An example of this hybrid alternative, which integrates plot-based as much as character-based features, is the *Oz Project* (Bates *et al.* 1992), an attempt to use intelligent agent technology to attack the challenges in IS. Its architecture included a simulated physical world, several characters, an interactor, a theory of presentation, and a drama manager. Users communicated with the system using either a text based or a graphical interface.

Another hybrid system is *Façade* (Mateas & Stern 2000). In this case, a drama manager allows the characters to be autonomous during most of the time, but changes their behaviour whenever necessary to advance the plot, combining higher-level goals that are essential to the story with smaller goals specific to the autonomous characters. Figure 3.4 shows a scene from Façade.

Despite the fact that Façade has become a successful experience (Crawford 2012), its architecture requires a huge authorial effort to create new narratives. The authors spent two years creating a narrative that has only one scene, two characters, and takes about 20 minutes to complete (Mateas & Stern 2003). Moreover, in Façade, all the dialogues and the graph of the narrative structure need to be created in advance, not qualifying therefore as truly automatic generation of stories. Its main contribution, nevertheless, was to prove the possibility of generating digital stories with a strong dramatic appeal (Karlsson 2010).

Erasmatron (Crawford 1999) is another system using this hybrid approach. Working on the concept of verbs and phrases, Erasmatron tries to balance the aspects of plot and character within the narrative. Actions are represented by verbs, with lists of roles that can be assigned to the characters to form sentences.

Functions are implemented as logical operators, with parameters and pre- and post-conditions.

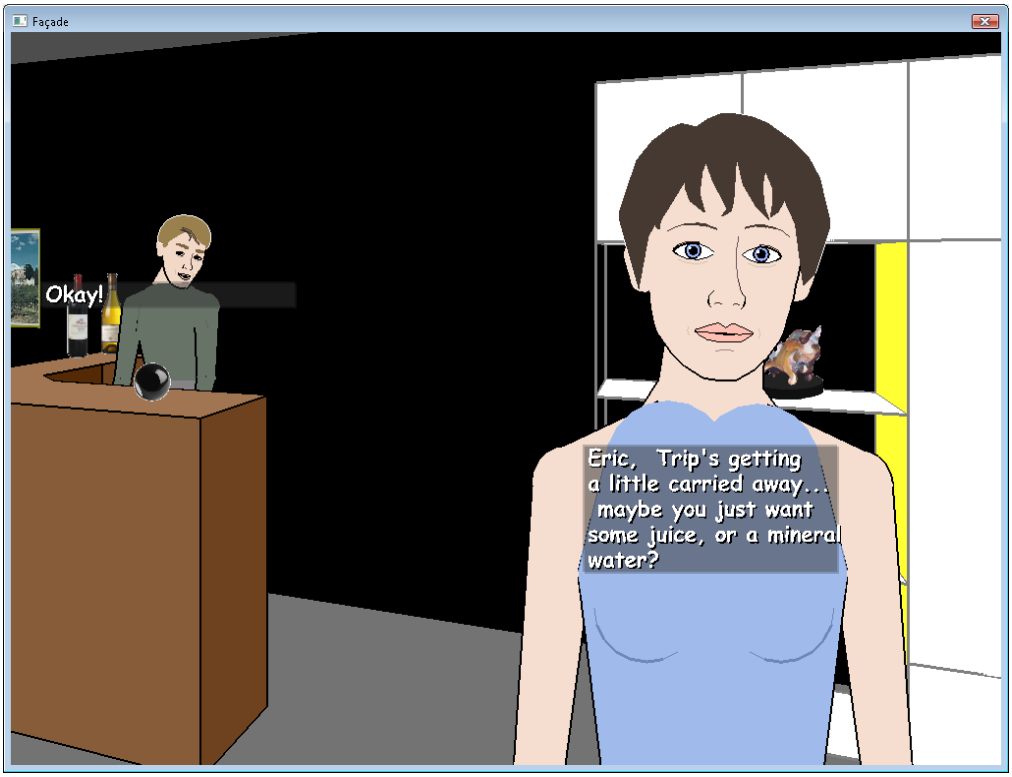

Figure 3.4 – Scene from the *Façade* interactive system.

3.6. Automated Planning in Interactive Storytelling

One of the areas in which automated planning has been cleverly implemented is that of interactive storytelling systems. The main utility of using planning in interactive storytelling is the possibility of automatically generating sequences of events. In this way, it equips the interactive storytelling applications with a tool that permits greater autonomy and dynamism, which ends up being reflected on a greater freedom for the users in their interaction with the system. For this reason, it is fair to acknowledge here the appearance of studies in the area of interactive storytelling in which automated planning assumes a central position.

Tale-Spin (Meehan 1977) was one of the first programs created to address the problem of automatic story generation. Based on a character-based approach, it is capable of describing some simple stories by simulating a virtual world where characters try to reach their goals. Stories are created by a planning algorithm that

is responsible for generating a plan that will be used by the characters. This plan, once generated, is translated into written natural language and then shown to the user. One of the main contributions of Tale-Spin was to show that planning algorithms could be very useful in creating convincing characters in the context of a story. Influenced by Tale-Spin, a number of later story generating systems adopted some sort of planning approach (Karlsson 2010). However, although Tale-Spin is able to generate some interesting stories, most of them cannot be deemed as dramatically interesting. Despite the characters being coherent, stories can turn out to be too short or simply uninteresting.

Universe (Lebowitz 1984, 1985) is another IS system that, as Tale-Spin, generates stories through the use of planning algorithms. But, in this case, the goals and plans generated are not only related to characters' goals, but also and especially to authorial goals. The characters are defined by personality traits, stereotypes and relations to other characters, and are responsible for assuming roles in the plot fragments created by the planner to reach the author's goals. Similarly to Tale-Spin, the story generated by the planner is shown to the user under the form of natural language text.

Yet another story generation system based on planning is *Minstrel* (Turner 1992). Differently from Tale-Spin and Universe, it uses a planning technique called Case-Based Reasoning (Aamodt & Plaza 1994), where pieces of previously known or pre-generated stories are reused to generate new ones. It distinguishes four types of authorial goals: theme, drama, consistency and presentation. *Theme* is concerned with the selection and development of the theme and purpose of the story. *Drama* is concerned with keeping the story interesting by generating suspense, tragedy, presages, etc. *Consistency* concerns the credibility of the actions performed by the characters. Finally, *presentation* is concerned with how the story is told to the reader.

Mimesis (Young 2001) is a system built to be used in digital games, working as an intelligent controller for virtual environments. The system tries to combine story planning with the use of the generated stories, noting that story planning is initially performed without taking into consideration the desires and objectives of the characters. The generated plan is then extended to include information about the goals established by the events and, from this point on, new

goals are generated and the system reasons about the characters' motivations to reach such goals.

The *Liquid Narrative Group* is a research group which has been working with planning-based interactive storytelling aiming at the creation of content for interactive games and other virtual environments. A proposal made by the group to solve the problem of reaching a balance between narrative coherence and user control in the generation of interactive narratives is the *narrative mediation*, a plot-based technique in which an agent-author has control over the actions of characters (Riedl & Young 2006). The system generates a linear narrative that is an ideal story to be told, and then considers all the ways whereby the user can interact with the virtual world and with the other characters. The generated story includes actions to be taken by the characters controlled by the system, and actions expected to be performed by the single character controlled by the user.

With narrative mediation, the generated plan is the narrative itself. The plan has markings indicating both the temporal relationships between actions and the cause and effect relations between them. User actions that threaten the cause and effect relations are called *exceptions*. For each possible exception, the narrative mediation system generates an alternative story plan starting from the point of the exception. The result is a story plan tree (called *narrative mediation tree*) in which each plan represents a complete story – either from the initial state of the world, or from the deviation point – up to its conclusion. This tree is generated before the user interaction phase, for which it serves as a guide. An example of narrative mediation tree can be seen in Figure 3.5.

Users, however, are not aware of the plan generated for the narrative. If they execute some action that is not part of the predefined script, the system checks whether the action causes an exception. If so, the appropriate alternative story is adopted and immediately executed.

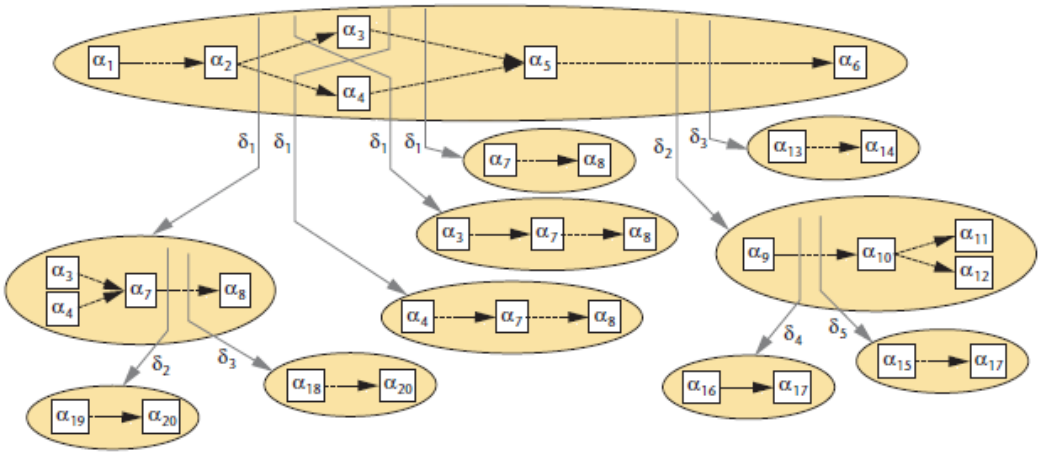

Figure 3.5 – A narrative mediation tree with accommodated exceptions (Riedl & Young 2006).

To prevent an overgrowth of the mediation tree, the system intervenes in some user actions, replacing them either with similar actions or with failure, thus avoiding unwanted effects. A list maintained by the system indicates which faults are associated with each action. Exceptions handled through intervention do not need an alternative story plan, because the original causal relationship is preserved. Interventions are also used in cases where the system would have no way to plan an alternative story - for example, when the user destroys any element or character that may be necessary at some future point in the development of the plot.

3.7. Logtell

3.7.1. Overview of Logtell

Another example of planning applied to interactive storytelling is **Logtell**, a system that, inspired by Propp's work on the typical functions of a narrative, is able to create plots based on a logical specification of a particular literary genre and the story's initial situation and, through 3D animation, dramatize the generated plot (Ciarlini *et al*. 2005). Basically, what **Logtell** does is to establish goals according to the specification of the adopted literary genre and the current plot situation, and create an acceptable chain of events that leads to those goals while taking into consideration the users' interventions. The system has

characteristics of both approaches: the plot-based approach in a larger proportion (via the logic specification of the chosen literary genre), and the character-based approach (through an inference process that assigns goals to be achieved by the characters when certain motivating conditions are observed). Figure 3.6 shows a scene from **Logtell**.

The main goal of **Logtell** is to generate different and coherent stories within a literary genre through different stages of simulation and user interaction. In order to ensure consistency, there is a formal logic-oriented model to capture the specifications of the intended genre. Story diversity is guaranteed through the planning process, which can generate chains of events in different genre-compatible combinations and sequential order. In addition, there are logic rules to define Propp's narrative functions (Propp 1968), tuned in terms of pre- and post-conditions so as to enforce the conventions of the genre, and to cause state-changes towards the satisfaction of the characters' goals. The rules defining the functions, expressed in modal temporal logic, state that, if certain conditions hold before the function's execution, certain other conditions should hold in the resulting state (Ferreira 2013).

Figure 3.6 – Dramatization in **Logtell**.

Logtell prototypes have initially treated the fairy tales sub-genre known as *Swords & Dragons* which, inspired by medieval fantasy, features dragons, wizards, princesses and knights, and assigns to the personages the roles of *villains*, *donors*, *victims* and *heroes* in correspondence with Propp's *dramatis personae*. In this sub-genre, we enable events such as *kidnappings*, *fights*, *spells*, and assume that the characters are basically moved by love and ambition. For example, the villain is subject to a goal-inference rule that states: "*If the victim is alive, and has not yet been kidnapped, then the villain will try to kidnap her*." Even in this simple context, the prototypes are able to produce a considerable diversity of plots.

Stories in the current **Logtell** version are generated in chapters. In each chapter the planner tries to achieve goals specified either by rules or by user interventions. Situations generated by the planned events and user interventions occurring while the chapter is being dramatized will influence the next chapter, and so on. The chapters are represented as contingency trees, where the nodes correspond to nondeterministic events and the edges are conditions that enable the execution of the next event (Silva 2010; Silva *et al.* 2010; Silva *et al*. 2011). A nondeterministic event e_i is executed by a nondeterministic automaton (NDA) (Dória *et al*. 2008) composed of actions a_i (Figure 3.7). These automata contain information about possible sequences of actions and are open to audience's interventions.

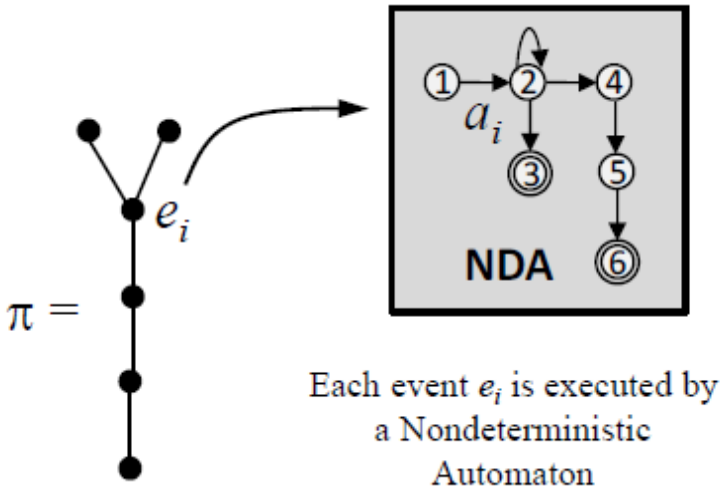

Figure 3.7 – Overview of the story generation process showing a plot π, events e_i, and a nondeterministic automaton with actions a_i; double circles denote final states.

3.7.2. Genre Specification

Logtell works with genre specification schemata, comprising the formal description, plus the information necessary for dramatization and scene control. To clarify how such specification is constructed, this section presents concrete examples of the Swords & Dragons genre. In addition to the genre, it is necessary to define a *context*, i.e. an instantiation of the genre specifying the characters, the scenery locations, and their attributes and relationships (to apply the standard terminology of the well-known Entity-Relationship model) at the initial state. The definition of a genre consists of:

- A logical description of the entities, relationships and narrative roles that may exist in the genre:
    - Entities: *creature*, *person*, *knight*, *dragon*, *princess*, *place*, etc.;
    - Inheritance between entities: a *knight* is a *person*, a *person* is a *creature*, a *dragon* is a *creature* etc.;
    - Relationships: the relationship *home* between a *creature* and a *place*, the relationship *acquaintance* between a *creature* and another, the relationship *married* between a *person* and another etc.;
    - Attributes: a *creature* has a *nature* (good or evil), a *creature* may or may not be *alive*, a *place* has a level of *protection*, *acquaintances* have a level of *affection* etc.;
    - Types of attributes: *alive* is boolean, *protection* is *compound* being kept in a 2-tuple format, indicating the *nature* of the place and its *protection* level etc.;
    - Roles: a *knight* can perform the role of a *hero*, both *knight* and *princess* can play the role of a *victim*, both *knight* and *dragon* can perform the role of *villain* etc.
- A set of parametrized operators, with preconditions and effects, specifying all the possible events;

- A set of goal-inference rules, specified in a modal temporal logic, defining which situations lead the characters to seek determined goals;
- A library of typical plans containing typical combinations of operators to achieve certain goals, organized according to *is-a* and *part-of* hierarchies;
- A non-deterministic automaton associated with each operator to specify the possible forms of dramatization of the event;
- Graphical models of the places and the 3D virtual actors;
- Scripts and a finite-state machine to control the movement of cameras and actors in the 3D virtual environment.

3.7.3. Architecture

Logtell is designed to work in an interactive TV environment. In this context, the diversity of stories and the ability to quickly plan their continuation, (taking into account the user interactions) are required for the generation of the plot. The system has a client/server architecture (Camanho *et al*. 2009) that supports multiple users sharing and interacting in the same or even in different stories. The client-side is in charge of user interaction and dramatization of stories and, at the application server side, there is a pool of servers sharing the responsibility of creating and controlling multiple stories, which are presented in different clients. The audience can interact with the story either by suggesting events to the planning algorithm (to be treated in the next chapter) or by interfering in the nondeterministic automata in a direct or indirect way.

Logtell architecture (Figure 3.8) comprises a number of distinct modules to provide support for generation, user interaction and visualization of plots. In this architecture, story contexts are stored in the *Context Database*, where each context furnishes the description of the genre according to which stories are to be generated, and also some compatible initial state specifying the individual characters and the environment at the beginning of the story. The genre definition consists of:

- **Static schema**: describes the mini-world wherein the plots take place (characters, relations, places…);
- **Dynamic schema**: describes the possible events in which the characters of the story can participate;
- **Behavioural schema**: describes the motives that guide the behaviour of the characters;
- **Personality schema**: details the personality traits for characters and events.

Personality schema: details the personality traits for characters and events.

The context database is accessed by the other modules via the *Context Control Module* (CCM).

The *Simulation Controller* performs the following tasks: (1) inform the Drama Manager, at the client side, the next events to be dramatized; (2) receive interaction requests and incorporate them in the story; (3) select viable and supposedly interesting suggestions for users who decide to perform strong interactions; and (4) control the instances of the *Nondeterministic Interactive Plot Generator* (NDet-IPG), the module responsible for the generation of the plan to be used as input to the dramatization process (further details about this module will be seen in Chapter 4).

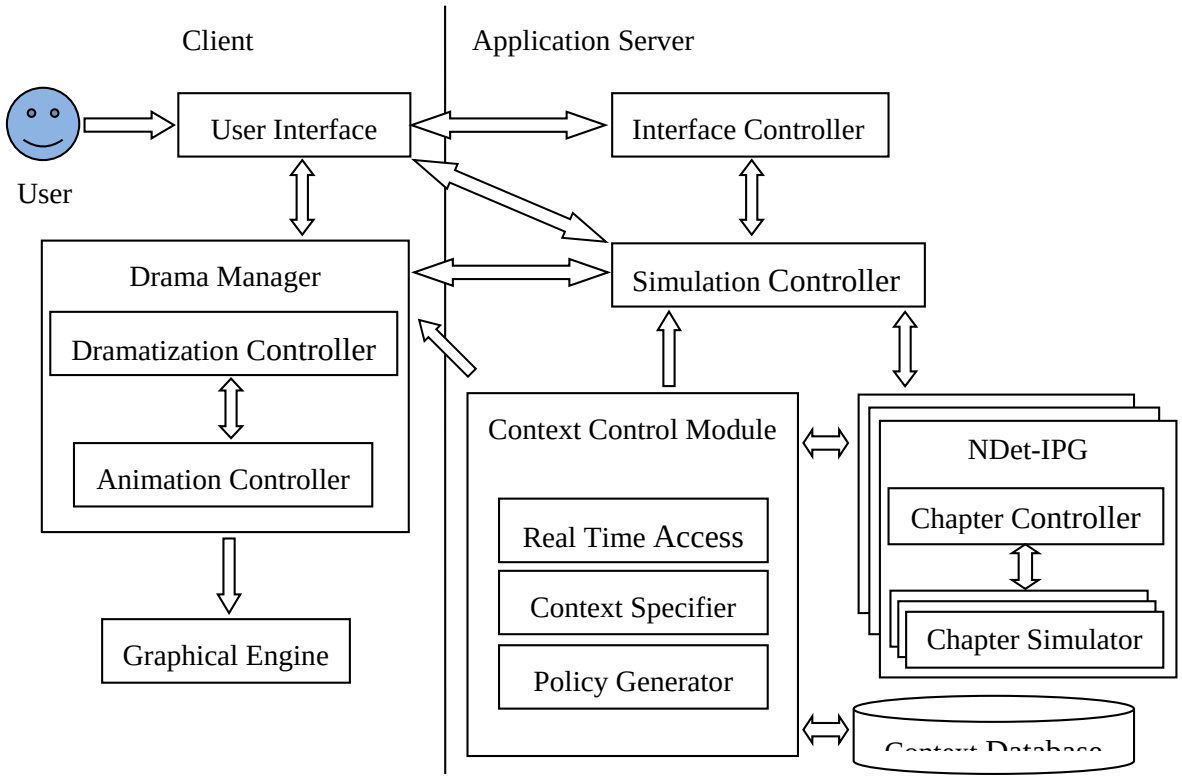

Figure 3.8 – Logtell architecture.

On the client side, the user interacts with the system via the *User Interface*, which informs the desired interactions to the *Interface Controller* that, placed at the server side, controls these user interventions and centralizes the suggestions made by them. The *Drama Manager* requests the next event to be dramatized from the Simulation Controller, and controls actor instances for each character in a 3D environment running on the *Graphical Engine*.

3.7.4. Evolution

Since its original release, **Logtell** has been refined through a series of extensions, including its adaptation to interactive digital TV (Camanho 2009), the use of nondeterminism in the dramatization of events (Dória et al., 2008) and the use of nondeterminism in the generation of plots (Silva 2010; Silva *et al*. 2010; Silva *et al*. 2011) – this latest version being one of the basis of our work. Moreover, further extensions have been produced since then, such as the verification of temporal constraints in continuous time (Araujo 2011), and the

treatment of dramatic properties of the story events (Gottin 2013; Ferreira 2013; Ferreira *et al*. 2013). These extensions are all incorporated in our new version.

3.8. Emotions in Interactive Storytelling

The art of storytelling is closely associated with the interplay of emotions. The way we relate emotionally with characters also partly determines how much we like or dislike a narrative, no matter whether it is presented in a novel, a film or a play. In fact, we tend to remember, for example, the films we watched according to how intensely they made us laugh or cry (Louchart *et al*. 2006). Hardly anyone would be interested in a story that does not arouse any emotion, be it joy, fear, curiosity or even sadness. Thus, it is not surprising that there are plenty of studies trying to associate the treatment of emotions with interactive storytelling. Some models of emotions presented in the literature (Plutchick 1962, 1980) apply particularly well to characters in virtual environments, as argued in the work of Rodrigues *et al*. (2007).

3.8.1. Personality and User Models

Recent research has addressed the usage of user models in interactive storytelling, particularly with the help of *stereotypes* (Rich 1979) to represent users' preferences. El-Nasr (2004a, 2004b) presents an interactive story drama called *Mirage* that makes use of player knowledge to enforce user engagement. This is made possible by borrowing the definition of the characters' behaviour from theatrical acting theory, along with inferred knowledge about the player's actions. Based on predicted player behaviour, the actors choose between different tactics (El-Nasr 2007) to try to reach their goals for a certain scene. When a goal is not reached by a certain behaviour, another one will be selected by the actors so that they can keep trying. The dramatization of Mirage's interactive narrative happens in a rich 3D world, supported by an architecture that implements an agent model and utilizes varying animations attached to the same action with different "adverbs" associated to them – i.e. if a character wants do draw a sword, it can do it slowly, violently, etc. (Karlsson *et al*. 2009). Mirage also makes use of a

scripting language that allows designers to define an evaluation function that influences the way the system estimates a user's character given its actions and story context. To represent a character's personality, Mirage makes use of a vector of stereotypes providing an evaluation separated into five dimensions: *heroism*, *violence*, *self-interestedness*, *truth seeking*, and *cowardice*.

Another interactive storytelling system that uses stereotypes is *PaSSAGE* (Player-Specific Stories via Automatically Generated Events), which uses automatic player modelling to learn the player's preferred style of play (Thue *et al*. 2007). This model is then dynamically used to select the contents of an interactive story from a library of possible events, called encounters, each event being annotated by an author with information concerning which player types it would be suitable for (Karlsson *et al*. 2009). When determining which encounter to run, PaSSAGE examines each encounter's set of branches. To help maintaining a stronger sense of story, encounters are grouped into sets corresponding to the various phases of the monomyth (Campbell 1968).

These encounters can be refined and are implemented via triggers, usually fired when a player approaches some suitable location. Characters satisfying the encounter's trigger conditions assume the behaviours destined for this event, which are tailored to encourage the player's preferred styles of play. The player model vector then changes depending on player action selection. The main difference between the stereotypes in PaSSAGE and those of Mirage is that Mirage tries to model the player's character, defining it in terms of the values attributed to the traits of the character's stereotype. PaSSAGE, on the other hand, categorizes player type stereotypes based on a set of player types published by Robin Laws (2001). These player types include *Fighters* (who prefer combat), *Power Gamers* (who prefer gaining special items and riches), *Tacticians* (who prefer thinking creatively), *Storytellers* (who prefer complex plots) and *Method Actors* (who prefer to take dramatic actions).

3.9. Personality Traits in Psychology

An investment that can improve the effectiveness of user modelling is to borrow some concepts from studies in psychology. Many of them have been made

concerning the finding of a reliable descriptive model, or taxonomy, to describe human personality. The main benefit of such taxonomy is to permit personality researchers to concentrate on certain carefully specified domains of personality characteristics, rather than examining separately the thousands of particular attributes that make human beings individual and unique. Moreover, a generally accepted taxonomy greatly facilitates the accumulation and communication of empirical findings by offering a standard vocabulary, or nomenclature (John & Srivastava 1999).

One starting point for a shared taxonomy is the natural language of personality description, under the assumption that most salient and socially relevant personality differences in people's lives will eventually become encoded into language and that, by sampling language, it is possible to derive a comprehensive taxonomy of human personality traits. Beginning with Klages (1926), Baumgarten (1933), and Allport & Odbert (1936), various psychologists have turned to natural language as a source of attributes for a scientific taxonomy. This work, beginning with the extraction of all personality-relevant terms from the dictionary, has generally been guided by the lexical approach (John *et al*. 1988; Saucier & Goldberg 1996). The lexical hypothesis posits that most of the socially relevant and salient personality characteristics have become encoded in natural language (Allport 1937). Thus, the personality vocabulary contained in the dictionaries of a given natural language provides an extensive, yet finite, set of attributes that the people speaking that language have found important and useful in their daily interactions (Goldberg 1981).

The earliest efforts to find the terms that could distinguish among human being's behaviours started with a gigantic list of 17,953 terms by Allport & Odbert (1936). This list was then divided into four major categories, consisting of: distinct personality traits (e.g., sociable, aggressive, and fearful); temporary states, moods, and activities (e.g., afraid, rejoicing, and elated); evaluative judgments of personal conduct and reputation (e.g., excellent, worthy, and average); and physical characteristics, capacities and talents, and terms that could not be assigned to any of the other three categories (John & Srivastava 1999).

Cattell (1943) used this list as a starting point for a multidimensional model of personality structure, beginning with a subset of 4,500 personality trait terms (Cattell 1943; Cattell 1945a; Cattell 1945b) and identifying 35 major clusters of

personality traits, which he referred to as the "personality sphere". Using this small set of variables, he conducted several oblique factor analyses and concluded that he had identified 12 personality factors, which eventually became part of the 16 Personality Factors (16PF) questionnaire (Cattell *et al*. 1970). This pioneering work stimulated other researchers to examine the dimensional structure of trait ratings. Fiske (1949) constructed much simplified descriptions from 22 of Cattell's variables; the factor structures derived from this study were highly similar to the later "Big Five" factors proposal (to be introduced next). Tupes & Christal (1961) reanalyzed correlation matrices from eight large samples, finding "five relatively strong and recurrent factors and nothing more of any consequence". This five-factor structure has been then replicated by Norman (1963), Borgatta (1964), and Digman & Takemoto-Chock (1981) in lists derived from Cattell's 35 variables. Norman (1963) named these factors *Extraversion* (or *Surgency*), *Agreeableness*, *Conscientiousness*, *Emotional Stability* (versus *Neuroticism*), and *Culture*. These factors eventually became known as the *Big Five* (Goldberg 1981), a title chosen to emphasize how extremely comprehensive each of these factors is. The Big Five structure does not imply that personality differences can be reduced to only five traits. Rather, these five dimensions represent personality at the broadest level of abstraction, and each dimension summarizes a large number of distinct, more specific personality characteristics (John & Srivastava 1999).

Several rating instruments have been developed to measure the Big-Five dimensions (Gosling *et al*. 2003). The most comprehensive instrument is Costa and McCrae's (1992) 240-item NEO Personality Inventory – Revised (NEO-PI-R), which permits measurement of the Big-Five domains and six specific facets within each dimension. Taking about 45 minutes to complete, the NEO-PI-R is too lengthy for many research purposes, hence giving room to a number of shorter instruments. Three well-established and widely used instruments are the 44-item Big-Five Inventory (BFI) (Benet-Martínez & John 1998; John & Srivastava, 1999), the 60-item NEO Five-Factor Inventory (NEO-FFI) (Costa & McCrae, 1992), and Goldberg's instrument comprised of 100 trait descriptive adjectives (TDA) (Goldberg 1992). John and Srivastava (1999) have estimated that the BFI, NEO-FFI, and TDA take approximately 5, 15, and 15 minutes to complete, respectively. Recognizing the need for an even briefer measure of the Big Five,

Saucier (1994) developed a 40-item instrument derived from Goldberg's (1992) 100-item set.

In search for some reliable still shorter instruments, Gosling *et al*. (2003) developed and evaluated 5 and 10-item inventories. Although somewhat inferior to standard multi-item instruments, the shorter instruments reached adequate levels in terms of convergence with widely used Big-Five measures, test–retest reliability, patterns of predicted external correlates, and convergence between self and observer ratings. On the basis of these tests, a 10-item measure of the Big-Five dimensions called TIPI (Ten-Item Personality Inventory) is offered for situations where only very short measures are needed, or personality is not the primary topic of interest, or researchers can tolerate the somewhat diminished psychometric properties associated with very brief measures (Gosling *et al*. 2003) – which are enough, for example, for entertainment purposes such as modelling characters in an interactive storytelling system.

4
Emotional Nondeterministic Interactive Storytelling System

Our work utilizes nondeterministic planning in order to augment the possibility of plot variations. As nondeterministic events have more than one possible outcome, causing a faster growth of the decision tree that represents the plot, we resort to HTN planning to speed up plan generation. Our nondeterministic HTN planner, *ND-HTN* (Silva 2010; Silva *et al*. 2010; Silva *et al.* 2011), is also capable of handling failed attempts to achieve goals, thus increasing dramatic tension and plot diversification. This planner, implemented in the interactive storytelling system **Logtell,** has incorporated several extensions and performance improvements during the last few years (Araujo 2011; Gottin 2013; Ferreira 2013; Ferreira *et al*. 2013), and has been used in other contexts, like automatic web services composition (Valfre 2010). Our ND-HTN planner has preserved these extensions, and has been extended to make it possible to validate the events against the personality traits of the character currently performing as agent.

Another concept our work has incorporated is the decision-making process based on personality traits which determines the behaviour of the characters participating in a story. In our original implementation (Barbosa *et al*. 2014), the process ran along three steps: goal selection, plan selection, and commitment. The selection criteria reflect individual preferences originating, respectively, from drives, attitudes and emotions. The prototype also permitted to model one or more users according to some of these personality traits. The following sections will review these notions in more detail, and show how we combined and extended them in the course of our work.

4.1.
Nondeterministic Planning for Generating Interactive Plots

The original version of **Logtell** had a specific subsystem responsible for plot generation, the *Interactive Plot Generator* (IPG). IPG (Ciarlini 1999) is a

nonlinear planner (i.e. it utilizes plan-space planning techniques, dealing with partially-ordered events) that creates plots as a sequence of chapters, each chapter corresponding to a cycle in which user interaction is incorporated, goals are inferred and planning is used to achieve these goals. In principle, IPG can work with any domain that obeys a predefined set of rules, from fairy tales to corporate games, and is capable of producing a fair number of different stories, even with relatively simple contexts.

However, it soon became clear that nondeterministic planning would be necessary if still more plot diversity were desired. With its help, one could implement events with more than a single possible outcome. The selection through a process of nondeterministic choice of which set of effects will actually be considered for a given event can be delayed, for example until the moment of dramatization, thus permitting different outcomes to be achieved by the events of a chapter. But in order to avoid that nondeterminism might affect the consistency of the stories, the nondeterministic planner has to consider all possible outcomes after each event, and must previously generate branches to ensure the achievement of the goals in each chapter, regardless of which effects will be incorporated to the plot. To increase the degree of variation between the possible ends of a chapter, it is also essential that goals may simply correspond to *attempts* to establish a certain situation. Thus, a chapter can end with the attempted goal being achieved or not, a dilemma that always heightens the dramatic tension of the narrative.

To guide the selection of events that become part of the plan, we make use of hierarchical task networks (HTN). Plan generation based on task networks provides more generality than one can obtain from the establishment of a target state, as happens with IPG. Another advantage of HTN refers to the selection of events, due to the way the methods deal with the tasks they apply to. By considering only promising (and predefined) ways to generate a plan, the process is accelerated – a major bonus for interactive storytelling applications, especially when there are several alternatives that can considerably extend the search space.

The use of hierarchical task networks also influences the number of possible plots, as there may be more than one possible method for accomplishing the same task. Thus, the highest level definitions of the literary genre may result in different sequences of events at the lowest level, as long as different methods are adopted.

Throughout this section we will see the requirements defined in our model, and how **Logtell**'s architecture and specifications were modified in order to encompass the generation and treatment of nondeterministic plots with HTN planning. We will also see how the partial order planner IPG was modified to work seamlessly with the hierarchical nondeterministic planner, incorporating the concept of attempts.

4.1.1. Requirements

To improve the characteristics of **Logtell**, we created a new model to permit the treatment of nondeterministic events, whereby a wider variety is obtained in the generation of stories. To assure that nondeterminism does not affect the system's overall performance, specific techniques were introduced to speed up plan generation. The main technique was the introduction of a hierarchical task network (HTN) planning phase, which uses knowledge about the domain of the stories to cope with the creation of nondeterministic plans.

Any model of nondeterministic plots must attend to certain general requirements, to which we added one that is specifically related to the aims of our work:

- **Support for nondeterministic events.** Provides a structure to add alternatives to the plot, so that the effects actually selected during dramatization are taken into account and lead to the goals of the chapter.
- **Control of the level of nondeterminism.** The generated plans could, if many alternatives are considered, cause a combinatorial explosion on the number of possible continuations of a chapter. To prevent this, the model is able to set a maximum value for the number of branches that can occur in the plan to be generated.
- **"Weak goals".** Incorporated to create dramatic tension corresponding to attempts to establish a situation that can either succeed or fail. These goals are considered fulfilled if there is a chance that the execution of the plan leads to a certain condition. It is also possible to establish how close of reaching that condition the

plot should be. By incorporating weak goals, it is easier to specify that it is acceptable to finish a chapter with completely different alternatives.

- **Speed of plan generation.** The inclusion of events with nondeterministic effects results in a higher level of complexity during the planning stage. In addition, it is necessary to plan all possible continuations for every chapter.
- **Selection of goals and events based on the specification of personality traits.** In order to make the plots more believable and bring coherence to the decisions made by the characters, the choice of what they want to do (their goals) and how they do it (their plans) must take into consideration their personality traits.

4.1.2. Two-Phases Planning Process and New Architecture

In order to fulfill these requirements, our model considers that the planning process should be divided in two phases. In the first phase, a partial-order plan is generated, with which, after incorporating user interventions or inferring new goals, a sketch of the solution is created. This sketch will serve as an initial HTN for the second phase, in which an HTN-planner details the plan, creating a contingency tree that takes nondeterminism into account. The mix of partial-order and HTN planning accelerates the planning process, but at the same time does not restrict the solutions to previously specified HTN methods.

The initial HTN is a partial-order plan containing basic and complex events (that correspond to complex HTN tasks). In the first phase, the number of events should be limited in order to reduce the search space. Nondeterminism at this phase is attained by inserting events that achieve weak goals. Each event has a set of deterministic effects and a set of one or more sets of alternative nondeterministic effects. A set of nondeterministic effects can establish a weak goal, but not a strong goal. Actually, it must be noted that nondeterministic effects can hinder the establishment of a strong goal.

When the original partial-order planner IPG infers more than one goal when applying the predefined goal-inference rules at the current state, it tries to find a

plan that accomplishes all such goals. Our new decision-making process proceeds differently. Whenever more than one goal-inference rule is applicable at the current state, each rule is treated separately – because they are related to different *options* concerning what the characters may be motivated to do given a certain situation. More than one goal may be worthy of consideration for the same character, in which case the goals will be compared on the basis of the character's drives, and then mapped into one or more *different* plans. So we need to plan for *each* of the rules at a time, rather than for *all* of them simultaneously. In order to provide a solution to this problem without altering IPG's previous capabilities (since our intention is to maintain backward compatibility, so that **Logtell** could still be used with contexts without personality traits, the same way it was used before), we are using a "shell" that calls the goal inference process, gets the possibly multiple goals from multiple rules, separates them by each rule, and then finds a plan for each case separately, instead of trying to find a unique plan that accomplishes all the goals together. The chapter simulator can use this shell or access the processes of goal inference and partial order planning directly, depending on a choice parameter.

The second phase must detail the initial HTN generated at the first phase. In order to obtain an executable plan, the complex events have to be decomposed into basic ones. This phase performs in forward-search mode, creating branches (corresponding to the number of alternative nondeterministic effects) whenever a nondeterministic event is incorporated to the plan. The process creates a plan that corresponds to a contingency tree in which the nodes are basic events (which can be directly dramatized) and the edges contain conditions to be tested after the event in order to choose the branch to be followed during dramatization.

Due to the treatment of weak goals at the first planning phase, some events are marked as attempts, meaning that they might not occur. In the branches where their preconditions are not valid, they are immediately discarded.

The two planners are combined in one module in **Logtell**'s architecture, called *Chapter Simulator*. To coordinate the activities exchanged between the Chapter Simulator and other modules of **Logtell**, a control module was created, the *Chapter Controller*. Together, the Chapter Controller and the Chapter Simulator compose the *NDet-IPG* module (Figure 4.1), playing the role previously performed by IPG in **Logtell**'s architecture. Plans generated by the

nondeterministic HTN planner must accomplish the task network for all possible paths represented by the contingency tree (note: 'accomplish the task network' means (Erol et al 1994) “to accomplish all the subtasks in the task network without violating the constraint formula of the task network”). The nondeterminism introduced by this planner allows user-interaction with the current chapter.

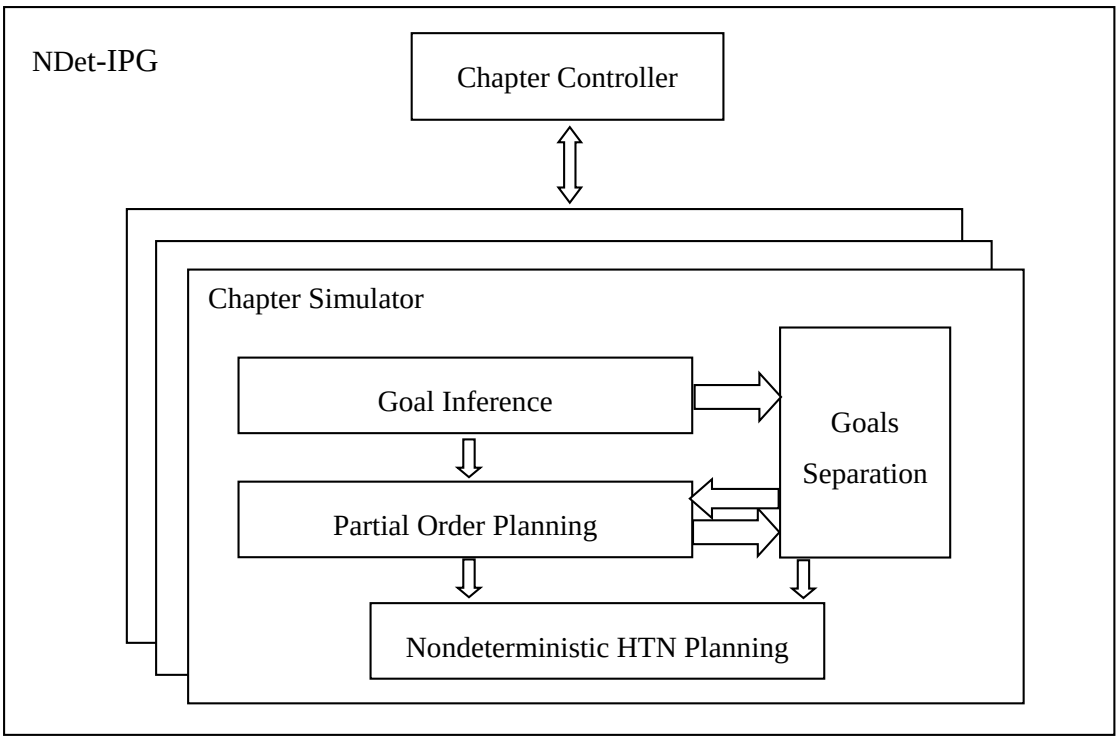

Figure 4.1 – Functional scheme of NDet-IPG.

If the nondeterministic HTN planner succeeds in generating a plan that accomplishes the task network, it is delivered to the Chapter Controller, which will in turn send it forward to the other modules in charge of dramatizing the events. The Chapter Controller will then manage the user interventions and coordinate the generation of new plans from the different final states that can be reached by this nondeterministic plan.

4.1.3. Nondeterministic Operators

The literary genre definitions used originally by IPG establish, for each operator, a set of deterministic effects that specify the facts that become valid (or

no longer valid) when an instance of the operator is executed to mark the occurrence of an event. In order to work with nondeterministic effects, the syntax of these definitions was changed so that they may also include, in addition to those effects that should always be produced, the possible nondeterministic effects.

For example, in the context based on the Swords & Dragons subgenre used in **Logtell**, there is an operator *kidnap*(*VIL*, *VIC*), whereby a character (*VIL*), playing the role of villain, kidnaps a character who plays the role of victim (*VIC*), dragging the latter by the lair. This operator has only deterministic effects: the victim becomes *kidnapped* by the villain and both characters are now at the place where the villain resides. We can add dramatic tension to the plot by replacing such operator by a nondeterministic version, *try_to_kidnap* (*VIL*, *VIC*), which might have either the same outcome of its deterministic counterpart, or some alternative effect, e.g.: the villain would end up killing the victim while pursuing the capture. Thus, the same operator could lead to two different states (kidnapped or dead victim), which would eventually generate strikingly different alternatives in the development of story.

The nondeterministic effects are described as a set of sets of effects, where each set of effects configures an alternative eligible by the dramatization process – noting that, during the planning process, it is not known which of these sets will be ultimately integrated into the plot. Fully deterministic operators are of course characterized by having no such sets.

Also, the description of these operators requires means to indicate their hierarchic position. HTN planning occurs through the decomposition of certain composite operators into sub-operators, which, by their turn, can also be composite operators – so the process is repeated recursively. Thus, the definition of an operator in this model may contain the definition of sub-operators, as well as ordering constraints between these sub-operators. For example, besides the operators *fight* (which indicates a fight opposing two characters) and *kill* (indicating that a character kills another), existing in the previous version of **Logtell**, we could also include a composite operator *defeat* containing both *fight* and *kill* as sub-operators, with the obvious ordering constraint that *fight* must be performed before *kill*.

4.1.4. Partial-Order Planning

The task of generating the plans to be used in the dramatization belongs to the Chapter Simulator. This module combines the functionality of the partial-order planner IPG, with the nondeterministic hierarchical planner. All the process of nondeterministic planning is concentrated in the new planner, specifically developed for this purpose. However, some modifications of the partial-order planning process were required.

One has to do with the specification of the literary genre – particularly regarding the operators. As now each operator may have, in addition to a set of deterministic effects, alternative sets of nondeterministic effects, the Chapter Simulator, when receiving an input from the Chapter Controller indicating a goal state to be achieved, uses an adapted version of IPG to try to build a task network that will achieve that goal state. However, IPG partial-order planning remains deterministic, and hence must adopt an inflexible behaviour: for the establishment of preconditions (and the goal state), considering exclusively the deterministic effects of each operator; on the other hand, when facing the possible threats to preconditions (and the goal state), it must take into consideration all effects, deterministic or not.

To increase the potential for variation of the generated plots, it is convenient that the task networks have as many complex operators as possible, as this would allow a larger number of different instantiations of these operators which can be used to generate alternative plots. The ability provided by IPG to set the value of certain special attributes of the operators, called *costs*, which affect the priorities of the planning algorithm, proved to be very helpful in this regard. Operators with the highest level of generalization, and those with sub-operators, could be set with the lowest *cost* values, thus gaining priority during the partial-order planning process.

The adaptation of the partial-order planning of IPG was necessary to limit the number of operators the initial sketchy plan can have, and to incorporate the notion of weak goals. In order to deal with weak goals, the altered version of the partial-order planner considers weak preconditions of the type *tried*(*L*, *N*), where *L* is a fact or the negation of a fact and *N* limits the number of event occurrences

between a possibly failed attempt and the moment when the weak precondition should be considered fulfilled.

Simply put, *N* is such that, the higher its value, the greater is the number of linked events needed to establish *L* that can fail to happen (because of unsatisfied preconditions). The process works in such a way that events inserted for the establishment of a precondition of the type *tried*(*L*, *N*) are just "attempts" of executing events, which will occur only if their preconditions are valid. Note that events considered as attempts may not occur, because the establishment of their preconditions is possible but not necessary.

More precisely, when *N* = 1, a precondition *tried*(*L*, 1) of an event *E1* is satisfied if there is an event *E0* that will surely occur before, and *E0* has *L* defined as one of its effects – either deterministic or not. This means that there is a chance of establishing the precondition by an event that certainly occurs. Obviously, the establishment of *L* by some effect, deterministic or not, must not be blocked by any event that may occur between the time when it is established and the moment when it should be true. For values of *N* greater than 1, we adopt the concept of attempt for the preconditions of an event that satisfies *tried*(*L*, *N*), but now with a tolerance level decreased by one unit. Thus, a condition of the type *tried*(*L*, *N*) requires the accomplishment of conditions of the type *tried*(*P*, *N*-1) for each precondition *P* of the event that establishes *tried*(*L*, *N*).

An event *E* in these conditions must be considered an attempt, indicated by *attempt*(*E*, *N*-1), which will be incorporated into the final plan only if all of its preconditions are satisfied. According to this interpretation, an event of the type *attempt*(*E*, 0) has to happen, i.e. it requires the deterministic establishment of all of its preconditions. Likewise, operators of the type *attempt*(*E*, *N*) only implicitly establish effects of the type tried(*L*, *N* + 1), whether these effects are deterministic or not. Operators can explicitly specify, among their deterministic effects, the establishment of an attempt of the type *tried*(*L*, *N*). This serves to express that a complex operator can be used to represent an attempt with some specified level of tolerance. If an operator is inserted into a complex plan as an attempt with tolerance level N, its effects of the type *tried*(*L*, *M*) become *tried*(*L*, *M* + *N* + 1). Finally, it is noteworthy that, in order to have the concept of attempt feasible in the partial order planning, effects explicitly defined as being of the type “tried” must be permitted only among the deterministic effects of the events.

4.1.5. Nondeterministic HTN Planning

Most of the effort to implement the new plot generation mechanism was invested on the implementation of our nondeterministic HTN planner. The concepts of HTN subtasks and methods were implemented using a scheme of inheritance and composition of operators. Apart from the primitive domain operators (which we call henceforth *basic operators*), an operator can be either a *generalization* of one or more specific operators and/or a *composition* of partially ordered sub-operators, in both cases to be called *complex operators*. Generalization relates a parent operator to one or more child operators. If the parent operator is also a composite operator, its children will inherit its composition. Children operators can also add new ordering constraints to the composition, and even include new operators among the inherited sub-operators, thus enhancing flexibility in the authoring process.

Especially for this most recent version of the planner, we added the concept of *grouping* operators. Any of the parameters of this new type of complex operators can denote a group of agents or patients of the action represented, and the planner will then replicate the action (with possible differences, as explained next) for each component of the group. Since any operator can be overloaded with different specifications depending on the emotional attributes of the characters they are applied to, a grouping operator applied to a certain group (for example, the guards of a castle, each of them to be individually attacked) can result in different detailed behaviours for each of the members of the group (for example, effectively fighting the bravest guards and only threatening the ones who are easily frightened).

The nondeterministic HTN planner replaces complex operators by their corresponding children and sub-operators, which can be not only basic operators, but also new generalizations, compositions or groupings. These complex operators must be recursively specialized, decomposed or replicated until basic operators are reached and incorporated to the plan. When a parent operator has more than one child operator, their preconditions are evaluated by the planner in

order to choose one of them, just like an HTN method is chosen to accomplish a task.

As the HTN process performs a forward search procedure whenever a primitive operator is selected, the nondeterministic planner updates the current state with its deterministic effects and, continuing from this new current state, makes a new branch for each of its nondeterministic effects. These branches have their own current states (which will be used to plan forward using the remaining task list) updated with their respective nondeterministic effects. Thus, every time a nondeterministic operator is added to the plan, the planner will have new branches in its plan with their own operators and states. The greater the number of nondeterministic actions, the greater will be the number of final states. In the generated contingency tree, the conditions for the selection of each branch correspond exactly to the nondeterministic effects that generated it.

The nondeterministic HTN planner only considers operators of the type *attempt*(*OP*, *N*) in the branches when all the operator's preconditions are satisfied. When this is not the case, the operator is simply not included. Effects and preconditions of the type tried(*L*, *N*) are not considered in the HTN planning phase. Goals of the type *tried*(*L*, *N*) appear in the inference rules (and/or in situations explicitly provided by the user) to increase dramatic tension and the opportunities for user interaction.

4.2. Plot Generation with Character-Based Decisions

4.2.1. The Three-Step Decision-Making Process

In this work, we are incorporating into **Logtell** a method to determine the behaviour of the characters participating in a story, based on affective aspects. As mentioned before, each character engages in a three-step decision process (Figure 4.2), in the course of which they select a goal, then an adequate plan to pursue that goal, and, finally, decide whether or not to commit to the selected plan. The selection criteria reflect individual preferences originating from different species of personality traits: *drives* for goal selection, *attitudes* for plan selection and

emotions for commitment. Four kinds of inter-character relations are also considered, referring to goal and plan interferences.

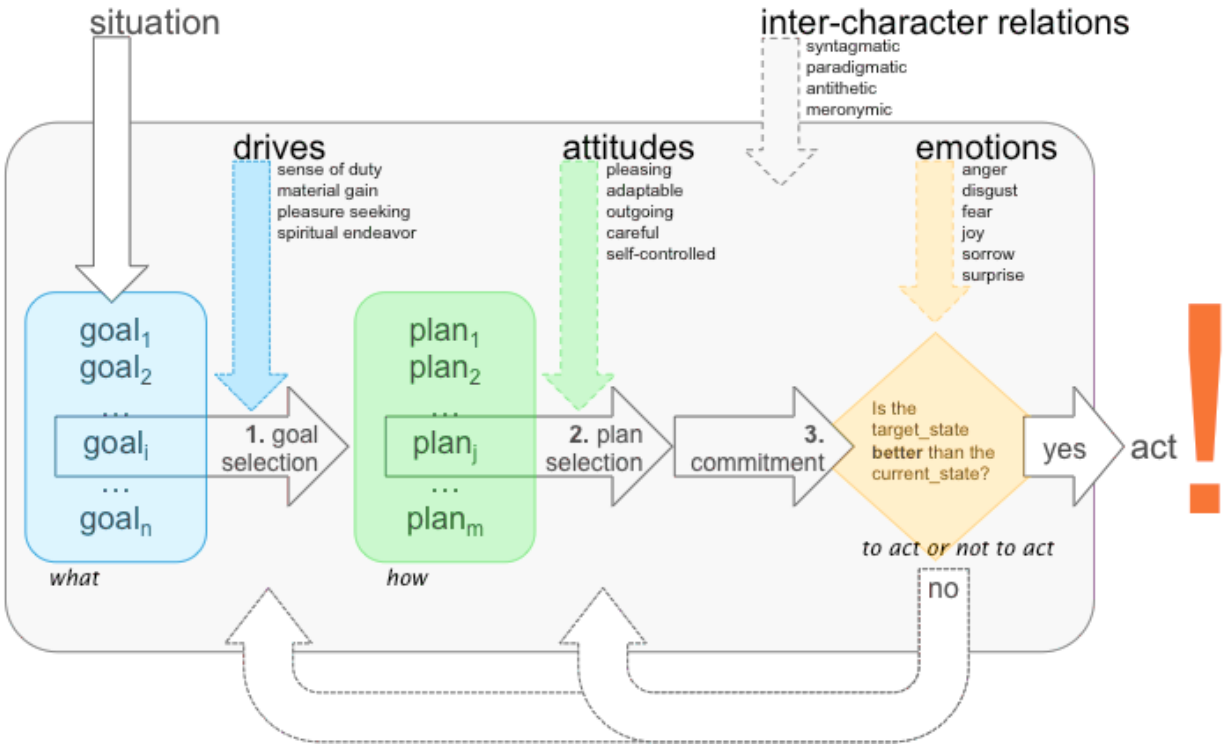

Figure 4.2 – The three-step decision –making process

This decision-making process requires some authorial effort, more than in the early **Logtell** prototypes, to formulate a set of situation-goal rules, associating each given situation with a list of goals. Situations and goals appear as logical expressions involving predicate terms that denote facts of the context scenario, with the goals declaring which terms should be either asserted or denied in the target state. Also, every goal is valued with respect to each of the four drives.

The first decision step is to select a goal from all the lists of goals of the rules triggered in view of situations holding at the current state. Each goal is associated with a 4-element frame which assigns numeric *values* to the goal with respect to each of the four drives. On the other hand, an attribute of each character is a similarly structured frame, which assigns numeric *weights* that indicate how strongly, in a positive or negative sense, the character is affected by each drive. The overall evaluation of a goal for a given character takes then the usual form of a *utility function*, i.e. a straightforward summation of weighted values, and the goal with the highest result is selected for the character.

After finding what to do, the next decision is to choose how to proceed. So, the second step of the decision-making process is to choose a plan from a set of goal-plans rules that associate a given goal with a list of plans. This process is similar to that used in the goal-selection process. Plans are valued with respect to each of the five attitudes, and characters have weights for these attitudes indicating how much they influence their behaviour. The decision-making process to select the plan that a character would use to achieve the goal in question then follows the same sort of overall evaluation that we just described for goal selection.

Once both a goal and a plan have been selected, the character must then assess whether or not the target state resulting from the actions to be executed (which may bring about a number of side-effects besides the achievement of the intended goal) is better than the present state. The third decision step is then to commit or not (Cohen & Levesque 1990) to executing the selected plan. To give a brief description – cf. (Barbosa 2014) for details –, this decision uses specific emotional-factor rules for each character, attributing values to situations with emotional significance to the character, computing and comparing the level of overall emotional satisfaction at the current and at the target state, to finally choose whether or not to commit depending on the gain or loss revealed by the comparison. In the present adaptation to **Logtell,** however, we are only using the emotion frames as a criterion of decision for some events.

The weights given in a personality profile to express the relevance to the character of each of the drives, attitudes and emotions are situated in the range <-4 : 4>. Null and negative values mean, respectively, that the character is immune to some factor, or reacts in opposition to it. If necessary, the weights and values set initially should be gradually tuned by the author until all characters behave in close agreement with their assumed "style".

Our choice of drives, attitudes and emotions was inspired in certain widely accepted notions. The four drives (*sense of duty*, *material gain*, *pleasure seeking* and *spiritual endeavour*), for instance, correspond to the *purusharthas*, the canonical four ends or aims of human life of Hinduism (respectively *dharma*, *artha*, *kama* and *moksha* in the Sanskrit language) (Hopkins 1971). In the same vein, it is not hard to trace the model's five attitudes (*pleasing*, *adaptable*, *outgoing*, *careful* and *self-controlled*) to the Big Five factors (respectively

agreeableness, culture, extraversion, conscientiousness and emotional stability). Finally, we took the six basic emotions (*anger*, *disgust*, *fear*, *joy*, *sorrow* and *surprise*) from the influential work of Ekman & Friesen (1971).

4.2.2. User-Based Character Modelling

Originally, with the model exposed above, the values and weights were estimated in a rather ad-hoc way. Ideally an environment should be provided in which several users would be assembled, each participant being invited to play a role. Through a suitable user-friendly interface, they would be allowed to determine, or to adjust to some extent, the personality traits of the characters they wish to impersonate, in terms that the interface could appropriately translate into numerical values and weights. People often want to play, in fiction, a part completely at variance from their real selves. Sometimes, on the contrary, they may want the chosen character to act just as they usually do, in which case the interface could first submit them to some psychological test based on documented studies, such as the tests for the Big-Five factors (Goldberg 1992).We have begun to explore these two role-playing preferences to tailor the characters' personality traits, as outlined in what follows.

We have modified the prototype to allow one or more users to choose a strategy for establishing the weights of the characters' personality traits: either by explicitly choosing each weight or by choosing a role stereotype and answering a few questions posed by a psychological test. Users can only adjust the weights of "active characters", i.e. characters that may perform actions and thereby influence the story plot. For each character, default weights are assigned in case there are fewer users interested in impersonating or adjusting a character than the number of characters in the story cast.

The role stereotypes are associated to frames with drive values previously established by the author. When the choice is to calibrate the character's personality, values are asked for each of the four drives and five attitudes of the model. These values, to facilitate the interaction, must be situated in the range <1 : 9>. They are then shifted to fit in the range <-4 : 4> used internally by the

prototype and in the characters' specification. For personality testing, we use the Big Five dimensions to set the weights of the characters' attitudes.

4.2.3. Storing Choices in Multiuser Environments

In multiuser environments, as proposed by Camanho (2014), there is, during most part of the time, more than just one viewer watching the story. Different viewers may have different preferences as to the types of events they want to see incorporated to the story. The capture of such preferences can be a valuable resource to offer a richer experience to the viewers.

Indeed, an interesting information that can be extracted from the viewers' interaction history is the qualitative aspect of the choices they make, i.e. the characteristics of the events that were chosen to be incorporated to the story. The idea is to enhance the viewers' experience by capturing their narrative preferences (regarding the events to be incorporated to the story). To accomplish this, we developed a prototype (Lima *et al*.2011) where events in the Swords & Dragons subgenre have been described regarding their *atmospheric traits*. These traits indicate the "feelings" that those events are assumed to bring to the story, and are inspired by the personality traits we can associate to characters.

For the context used in that particular work, five atmospheric traits have been adopted: *adventurous*, *romantic*, *magical*, *violent* and *dark*. Similarly to the personality traits of characters of our character-based decision-making process, these traits are attributed to the events with weights that are in the range <-4 : 4>, indicating how strongly each of them (or their opposites, in case of negative numbers) are associated to each event. A zero weight indicates that the trait does not apply to that particular event. To improve the accuracy of the association between events and their traits, not only the names of the events are taken into consideration, but also their parameters. So, for instance, the event *kill* has always a high level of 'violence'; however, the murder of a hero has an evaluation for 'dark' that is higher than in the case of the murder of a villain.

The recorded history of the choices made by the viewers for every interaction presented is used to find the average atmospheric traits preferred by each viewer. Likewise, a generic model of preferences regarding these traits is

elaborated for the whole group of viewers, considering all the interactions of all the users. These values are then compared to those of the viewers individually, in order to find possible viewers that deviate very much from the rest of the group. This information can be used to suggest that these viewers should move from one group to another that better suits their expectations. In order to bring the possibility of further exploring this possibility, using the personality-based emotional frames associated to the selected plans – and perhaps in an integration with voting systems (Camanho 2014) –, we are also including this evaluation feature in our new version of **Logtell**.

5
Implementation

This chapter describes the main aspects of the implementation of the system presented in Chapter 4, which provides an operational description of the new proposed architecture. Based on the current version of **Logtell**, we developed a prototype in Prolog capable of generating plots with nondeterministic events using our model of personality traits. The first prototype of the character-based decision-making process was implemented in SWI-Prolog, but we chose Sicstus Prolog for programming the new system, as had been done in all previous versions of **Logtell**, one of which already included an early version of our nondeterministic planner.

Our prototype does not incorporate any changes to the Simulation Controller and to the user interface. Thus, it is not possible at this time to operate the dramatization of the events offered by **Logtell**. To validate compliance to the requirements presented in the previous chapter and test how the system reacts to user interventions, we ignore the modules unrelated to planning. User interventions are being done via the Sicstus interface. Throughout this chapter, the new modules and functionalities incorporated into the Logtell architecture will be examined.

As we completed the implementation, we came to appreciate even more the successive enhancements introduced along the time by the other members of the **Logtell** team. Our new features were not difficult to integrate with those supplied in the previous extended versions. In special, for any practical usage of our prototype such as interactive TV applications, the client/server work distribution of **Logtell** (Camanho *et al*. 2009) should prove indispensable.

5.1.
The New Architecture

Besides adding the NDet-IPG module to achieve nondeterministic planning, we found necessary, in order to accommodate the character-based decision-

making process presented in this chapter, to make some changes in the **Logtell** architecture. We thus proceeded to introduce a module where the personality-based decisions are made, the *Emotional Decision-Maker* (EDM). NDet-IPG communicates with EDM every time it has to evaluate some decision that is based on the personality traits, such as checking if a determined character drives are compatible with a selected goal. The Context Database now has an extension specifically with data about the characters' and events' traits, the *Emotional Database*. In addition, we store the interactions of the users with the system in the *Interaction Database*. The new architecture is shown in Figure 2.3.

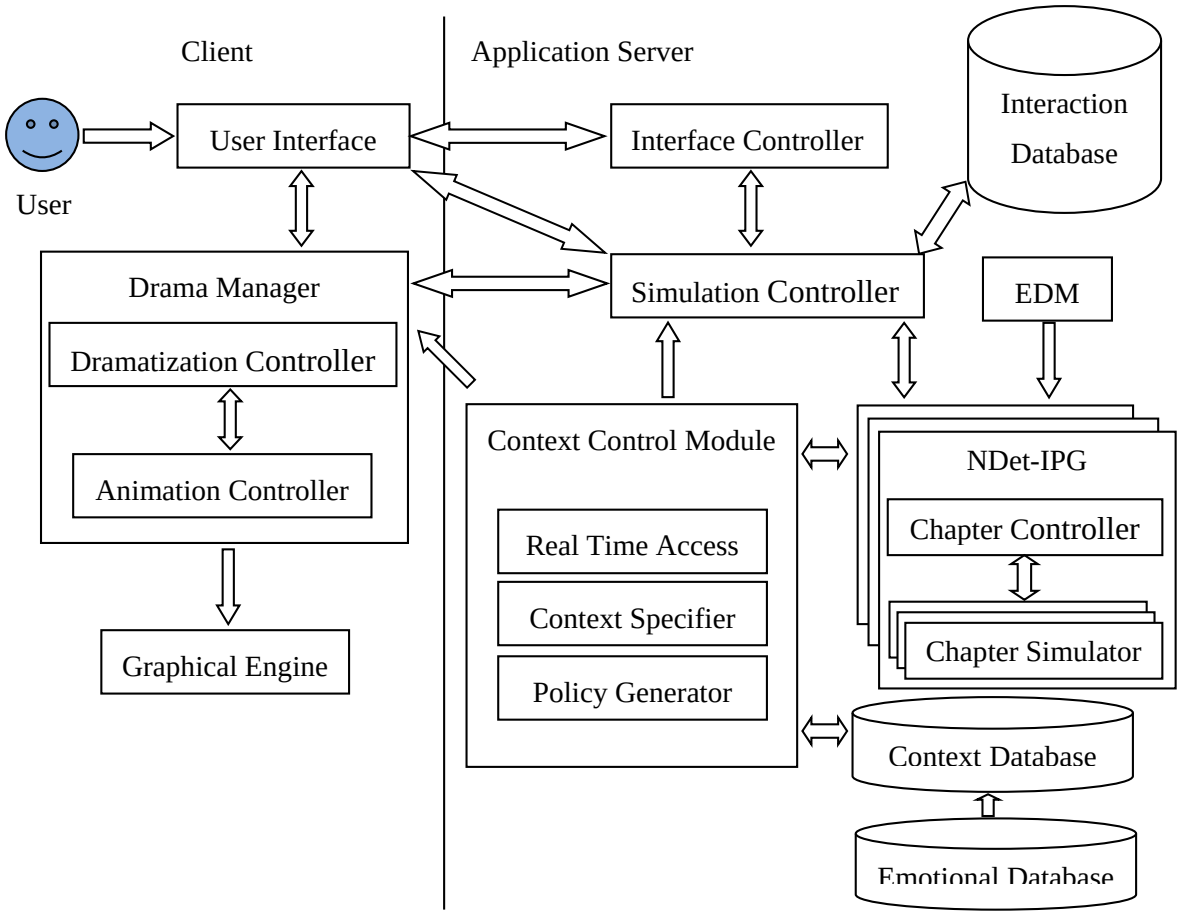

Figure 2.3 – **Logtell**'s proposed new architecture.

5.2. Emotional Decision-Maker

This new module is responsible for taking the decisions related to the character models. Our decision-making process requires the previous formulation,

by the author in charge, of a set of situation-goal rules, associating a given situation with one or more goals. Situations and goals are described by logical expressions asserting or denying the existence and properties of persons, places and all kinds of objects, animated or not.

Suppose that, at the current state S_0 of our context, one or more such rules are triggered, in the sense that their situation components hold at the moment. In our method, the first decision step to be accomplished by each character is to select a goal, after inspecting all lists of goals of the triggered rules. Having in this way found *what* to do, the next step is to choose *how* to proceed towards the selected goal. So, a character who proposes to achieve a goal will have to execute an appropriate *plan*, i.e. a sequence of one or more operations able to lead to a target state wherein the goal will hold, possibly together with a number of other effects which may or may not be to the character's liking. Such plans will later be produced by the plan-generation algorithm. Ideally, the context specification will have some composite operator(s) that establish the selected goal, and then the HTN planning process will find some suitable plan to accomplish it. So, at the second step of the decision process, a character desiring to pursue a goal will choose a plan after inspecting the operators present in the context. During the generation of the plan, some decisions will use specific *emotional-factor* rules for each character C, enumerating and attributing values to situations whereat these factors have any emotional significance to C.

At each step, we use distinct classes of personality traits to provide a decision criterion: *drives* to select goals, *attitudes* to select plans, and *emotions* to select possible variations of these plans. The personality profile of each character provides positive, negative or null weights to be applied to the values attached to drives, attitudes and emotions by the rules governing the three steps.

Also, besides the main acting characters, there may exist groups of lesser participants, whose individual actions will need to be described if the story is to be told at a more detailed level.

5.2.1.
The Personality-Based Decision Process

Before going into details, some general remarks are in order. The personality factors considered here are drives, attitudes and emotions. Thus, according to our proposed model, each character is described in terms of these three factors, using numerical weights to indicate the relevance of each drive, attitude and emotion with respect to the character's behaviour. It may happen that a character is immune to some factor, or may even react in opposition to it. For example, a character may be totally unconcerned with sense of duty (one of the drives we are taking into consideration), or may like the idea of breaching the existing rules, as a typical villain. Thus, weights can be positive, in the range <1 : 4>, negative, in the range <-4 : -1>, or null if the character's behaviour is unaffected by the corresponding specific factor. This 9-point scale is used to measure the weights in the form of a semantic differential scale (Osgood *et al*. 1957), a measurement technique widely used in attitude research.

On the other hand, positive, null or negative values, in the same ranges, are assigned to goals (in situation-goal rules) and plans (in goal-plans rules), to assess goals with respect to the various drives, and plans with respect to the attitudes. Values are also attributed to situations with an influence on the emotions of specific characters (in emotional-factor rules). For a given character, at each of the decision steps, the contribution of each factor is first computed as the product of corresponding weights and values (noting that, whenever weight and value are both negative, a positive contribution results), and then the totals obtained by adding the contributions are applied for ranking purposes.

Only goals and plans for which the totals are positive are retained. The total influences (positive or negative) of prevailing situations on the level of each emotion are added together to assess their overall contribution to emotional satisfaction at the current state.

IPG uses a set of goal inference rules that capture the goals that motivate the agents' actions when certain situations occur during a narrative. We included a new argument in the specification of these rules, an *emotional flag* that, when *true*, causes IPG to check the goal’s value. To calculate the value of a goal G_{ij} (for

a situation S_i that holds at the current state) regarding its drives, we must attribute *goal frames* of the form:

goal_frame(G_{ij}:(*Agent*, [d1:v1$_{ij}$, d2:v2$_{ij}$, d3:v3$_{ij}$, d4:v4$_{ij}$])),

where goal G_{ij} is valued with respect to each of the four drives and *Agent* must be one of its arguments. When unification occurs, the EDM module identifies via this argument the character whose drives will be considered. The values initially arbitrated by the designer are of course subject to later calibration in the course of experiments (the same being true for all numerical measures to be mentioned in the sequel).

On the character's side, frames of the form [d1:w1, d2:w2, d3:w3, d4:w4] must be specified to express by means of weights the influence of each drive in the character's conduct. The expression for the overall evaluation of a goal G_{ij} for a character C is then:

$$V_{G_{ij}}^{C} = \sum_{n=1}^{4} (\mathrm{wn}^{C} \times \mathrm{vn}^{G_{ij}})$$

which resembles ordinary utility functions (Russel & Norvig 2001), except that, in the latter, weights represent probabilities. Also recall that, when both weight and value are negative, their product yields a positive contribution – which equally applies to the other formulae we are going to show. For instance, the sense of duty drive may take a negative value for the goal of taking the unprotected castle of a knight's master (thereby betraying his trust), but a villain, with a negative weight for this drive, would count that as an asset, rejoicing at violating his duty.

Having found the goals suggested by what currently holds in the mini-world of the story, the next task is to proceed to choose *plans* to achieve it. We associate plans with a goal by defining operators that establish such goals. These goals should preferentially be defined as generalizations whose specializations represent different ways of achieving the higher-level goal. Again, we included an *emotional flag* as a new argument, this time for the definition of operators, so that when the flag is true, the planner will check how closely the operator's values agree with the character's attitudes. For this purpose, we must define *plan frames* of the form:

plan_frame(P_{ijk}:(*Agent*, [a1:v1$_{ijk}$, a2:v2$_{ijk}$, a3:v3$_{ijk}$, a4:v4$_{ijk}$, a5:v5$_{ijk}$])).

Similarly to what we indicated for drives, we need, for each character, to define frames that assign weights to the attitudes, of the form [a1:w1, a2:w2, a3:w3, a4:w4, a5:w5]. And to find for a character C the value V_{Pijk} of a plan P_{ijk} able to achieve a goal $G_{ij,}$ we use a formula similar to that used for drives:

$$V_{P_{ijk}}^{C} = \sum_{n=1}^{5} (\mathrm{wn}^{C} \times \mathrm{vn}^{P_{ijk}})$$

In our original prototype, the emotion frames associated with the characters and with particular situations were used as a means of evaluating and comparing the overall satisfaction of the character in the current state and in the prospective state caused by the execution of a plan P, which served as a criterion to decide whether or not to commit to plan P. In the present implementation, commitment to the selected plan is taken for granted. However, we can still evaluate the individual emotional attributes as well as the overall satisfaction as a basis for other decisions (for example, in the choice of operators).

Another feature is that now we can add changes to the values of personality traits as an effect of an operator, thus changing the character's personality during the plot. It is the author who decides up to what level the attributes are to be changed, according to the event under consideration – for example, an ordinary event would not change more than the emotions frame of a character, whereas a very traumatic event could even change a character's drive (a betrayal, for example, can diminish the character's sense of duty).

5.2.2. Tailoring the Character's Personality Traits

If the users decide to calibrate some character's personality traits, they can choose to do it by an explicit choice of values and/or weights for the drives and attitudes, by selecting a previously calibrated role stereotype (e.g. *hero* or *villain*) for the drives or by making a personality test to capture their own characteristics. To capture the user's Big Five dimensions, we utilize the *Ten-Item Personality Inventory* (TIPI), a reliable yet very short instrument for personality tests (Gosling *et al*. 2003). Although not as accurate as longer instruments, the precision showed in the studies made by Gosling *et al*. (2003) is more than what could be considered necessary for a simple simulation with entertainment purposes.

Moreover, most users would probably find it quite boring to be submitted to a long, comprehensive psychological test when their intention was just to have some fun to occupy their spare time. The TIPI, on the contrary, takes about a minute to complete.

Each item in TIPI consists of two descriptors, separated by a comma, using the common stem, “I see myself as:”. Each of the ten items is to be rated on a 7-point scale ranging from 1 (strongly disagree) to 7 (strongly agree). There are two items for each Big Five dimension, one is a positive representation of the dimension, and the other is a negative representation. So, for instance, the Big Five dimension “Extraversion” is represented by the items “Extraverted, enthusiastic” (positive) and “Reserved, quiet” (negative). Negative items have their scores recoded symmetrically within the range <1 : 7>, so that 1 is replaced by 7, 2 is replaced by 6, and so on. The average of the scores for the two items gives the score for the dimension. For our purposes, these scores have to be normalized to fit into the range <4 : 4> before they are assigned to the corresponding character’s attitude.

5.3. Chapter Simulator

The Chapter Simulator is the module responsible for generating the plans to be used in the process of dramatization. This module integrates the partial order planner IPG with the new version of the nondeterministic hierarchical planner NDet-HTN. We also added a shell called IPG2 that is used when this module is generating plots with personality traits. This section presents the algorithm used to implement NDet-HTN (Silva 2010), now adapted to deal with the personality models of the characters, highlighting the solution adopted for the representation of states during the planning process. We also show the changes in IPG’s partial-order planning that proved necessary, and how IPG2 works.

5.3.1. Nondeterministic HTN Planning

Figure 5.1 shows the algorithm used by the nondeterministic HTN planner incorporated to Logtell, NDet-HTN, whose result is a contingency tree linking

events and all paths from the root to a leaf, so as to accomplish the task network. Although inspired by the planning algorithm PFD (mainly as to the treatment of the order in which tasks are selected, according to the defined partial ordering), it has its own logic, mostly to allow an appropriate treatment of nondeterminism. In NDet-HTN, tasks correspond directly to operators, which may be basic, generic, composite or grouping. The input data are:

- *curr_state*: structure containing the sets $facts^+$ and $facts^-$, respectively the facts added to and removed from the initial state.
- *w*: a task network, a partial order of tasks, where each task corresponds to the name of an event (a partially-instantiated basic or complex operator). Tasks may have a modifier *attempt*, establishing that the corresponding event may not occur.
- *O*: the set of domain operators, possibly with specialization, composition and grouping definitions.
- *ndet_level*: positive integer that corresponds to the maximum tolerated degree of nondeterminism.

The following values are returned by the planner:

- π: plan generated by the algorithm. It takes the form of a contingency tree, represented by a structure (*a, L_CONT*) where *a* is a ground instance of a basic operator and *L_CONT* is the set of contingencies to be treated after *a*. Each contingency is a pair (*CONDITION*, π'), where condition is a test on the state after *a* and π' a plan to be executed if *CONDITION* is true.
- *final_states*: final states reachable by the plan, considering all the ramifications of *π*.

NDet-HTN(*curr_state*, *w*, *O*, *ndet_level*, *final_states*)
if *w* = Ø **then return** *EMPTY_PLAN*
nondeterministically choose any $u \in w$ with no predecessors in *w*
//Explicitly enforces ordering constraint based on the chosen node
for each node $i \neq u$ in *w*
insert into *w* every ordering constraint (*u*, *i*) that is not in *w*

```
t'_u ← task associated to the node u, without the attempt modifier
        (if present)
active ← {(a, σ) | a is a ground instance of an operator in O,
                        σ is a substitution such that name(a) = σ(t'_u)}
if active = Ø then return FAILURE
nondeterministically choose any (a, σ) ∈ active
if a is not applicable (also checking personality models) to curr_state then
    if u is an attempt then //It's a failed attempt
            w' ← w – {u}            //Remove the failed node
            π' ← NDet-HTN(curr_state, w', O, ndet_level, final_states)
            return π'
    else return FAILURE             //Not applicable and not an attempt
//Action is applicable and will be accomplished
if t'_u is a basic operator then
    w' ← σ(w – {u})         //Task is removed from HTN
    if t'_u has changes to be applied to some character then apply these changes
    det_state ← application of deterministic effects of a to curr_state
    L_CONT← Ø
    if w' ≠ EMPTY_HTN then
            if t'_u has no nondeterministic effects then
                //Builds subplan
                π' ← NDet-HTN(det_state, w', O, ndet_level,
                                        final_states)
                if π' = FAILURE then return FAILURE
                L_CONT ← {(TRUE, π')}
            else
                π' ← Ø
                for each nondeterministic effect ndet_effect of a
                    ndet_state ← application of ndet_effect to det_state
                    π' ← NDet-HTN(ndet_state, w', O, ndet_level,
                                            final_states)
                    if π' = FAILURE then return FAILURE
                    else L_CONT = L_CONT ∪ {(ndet_effect , π')}
```

```
                    final_states ← {final states reachable by L_CONT}
                    if length(final_states) > ndet_level then
                        return FAILURE
        else
            //w' = EMPTY_HTN → incorporate empty subplan to the
            //plan, closing that branch
            L_CONT ← Ø
            if t'u has no nondeterministic effects then
                final_states ← {det_state}
            else
                final_states ← Ø
                for each nondeterministic effect ndet_effect of a
                    ndet_state ← application of ndet_effect to det_state
                    insert ndet_state into final_states
                if length(final_states) > ndet_level then
                    return FAILURE
    else if t'u is a grouping operator then    // Replicates
        //Task is substituted by a new task network created by the substitution of u
        //by copies of a with the grouping argument replaced by its single elements,
        //similarly to the substitution of a node by its substasks (with substitution σ
        //replacing the grouping parameter by each of its subcomponents)
        w' ← δ(w, u, a, σ)
        π' ← NDet-HTN(curr_state, w', O, ndet_level, final_states)
        return π'
    else if t'u is a generic operator then    // Use a specialization
        specializations ← {e | e is a ground instance (using σ) of a
                                specialization of a in O, and
                                e is applicable to curr_state}
        nondeterministically choose any e ∈ specializations
        w' ← substitution of u by e in σ(w)
        π' ← NDetHPlan(curr_state, w', O, final_states)
        return π'
    else if t'u is a composite operator then
```

```
        //Task is substituted by a new task network created by the
        //substitution of u by the subtasks of a (using substitution σ)
        w' ← δ(w, u, a, σ)
        π' ← NDet-HTN(curr_state, w', O, ndet_level, final_states)
        return π'
    π ← (a, L_CONT)
    return π
end
```

Figure 5.1 – Pseudocode for NDet-HTN.

For the execution of the planning process, we must issue the following initial call to the planner:

NDet-HTN (*initial_state*, *initial_htn*, *O*, *ndet_level*, *Ø*)

The nondeterministic plans are represented in the form of a decision tree, where each node represents an action whose branches correspond to the sub-plan to be selected for each set of nondeterministic effects of that action. Deterministic actions have only one branch and lead to another single node (associated with another action); if there are nondeterministic effects, the branches lead to new nodes that correspond to the actions to be performed according to each condition (nondeterministic effect) assumed. The planner always adopts a possible total ordering compatible to the partial ordering of the task network, and actions without successors in the final plan are associated with terminal nodes, which have no branches.

One of the requirements of this planner is to properly handle the state updates resulting from the application of the effects of an operator instantiated into an action, in order to treat their possible nondeterministic effects. So every time a basic operator is selected and instantiated, the nondeterministic planner updates the current state, initially only with its deterministic effects and, stemming from this new current state, creates a new branch for each of its nondeterministic effects. Each of these branches has its own current state updated with its respective nondeterministic effects, producing new current states that will be used for the continuation of the planning process using the remaining task network (without the basic operator whose nondeterministic effects generated the branches in the plan). So every time a nondeterministic operator is added to the

plan, the planner adds new branches, each with its own sequence of actions and states. The greater the number of nondeterministic actions, the greater is the number of final states generated by the plan.

The subtasks and HTN methods were implemented using an inheritance structure and the composition of operators. Besides the domain operators (which generate the effects on the current state and are instantiated directly into actions), the domain specification also includes the concept of complex operators. An operator said complex may be a generalization of one or more specific operators and / or a composition of partially ordered sub-operators. Specializations relate a parent operator to one or more children operators. If the parent operator is composite, their children also inherit its composition (sub-operator and order constraints). In our proposed model, children operators can also add new constraints and even include new entries among their inherited sub-operators.

Another type of operator introduced in this new version is the grouping operator. It permits the definition of an action to be performed by members of a group – in our context an entity associated with a list of characters – and then the definition of the different sub-plans for the individuals of this group, depending on their characteristics. For example, it is possible to create a list of characters with different personalities – like a coward and a brave one – and associate them with a group of guards. A grouping operator *defeat* can then be applied to the guards as a whole; the planner will replicate it to each of the guards and, finding, as we will see later, a definition of *defeat* for a single character where the agent only threatens the guard if he has a positive level of fear and attacks and kills him otherwise.

The algorithm treats such cases as follows: first, it analyzes whether the operator is basic; if so, it directly performs the necessary modifications to the current state, the plan and the task network. If it is complex, it will first verify if it corresponds to a grouping, then replicating it for each of the elements of the group. If this is not the case, the algorithm verifies if it is a generalization, then seeking a specialization and making the necessary changes in the task network to replace the generic operator by the selected specialization. Finally, if it is neither a grouping nor a generalization, it must be a composition, and the task network is then changed – similarly to what is done in the PFD when a task is replaced by its

subtasks – to replace the operator by its sub-operators. In all cases, the planner is then called recursively with its current task network.

Another important feature treated by the algorithm is the concept of *attempts*. Operators of the type *attempt*(*OP*(a_1, a_2, ..., a_n), *N*) for *N*> 0, are interpreted as an attempt to incorporate the operator *OP*(a_1, a_2, ..., a_n) in the plan. If the preconditions of the operator are accomplished, the algorithm proceeds normally treating it without considering the attempt modifier. If the preconditions are not accomplished and the partial ordering constraints do not permit any other event occurring before the operator, it is simply discarded and removed from the task network as if it had been completed normally. The value of *N*, inherited from the partial-order planner and representing the level of tolerance to failure for the occurrence of the event associated with the operator *OP*(a_1, a_2, ..., a_n), is ignored in the nondeterministic HTN planning process, except in the special case where *N* = 0, corresponding to an event which must necessarily occur, i.e. *attempt*(*OP*(a_1, a_2, ..., a_n), 0) = *OP*(a_1, a_2, ..., a_n).

5.3.2. States Representation

In a nondeterministic planning process it is necessary to reason about various states that can be achieved during its execution. Every action incorporated into the plan causes a change in the current state; these changes have to be constantly recognized by the planner so that the correct actions can be selected. To avoid the need for "materialization" of the current state for every incorporated action – i.e. the complete description of the new state, contemplating the effects caused by the action –, we adopted the strategy of describing the states by indicating the successive changes applied to the initial state. The main objective of this choice is to avoid potential performance degradations caused by traffic in memory of a data structure that contains all literals that describe the state at a given time.

The effects established by the operator (that is, by the actions that instantiate it) may include positive effects (which indicate events that must be true in the new state) and negative effects (which indicate events that must be false in the new state). Changes to the initial state are stored in two sets: one of added facts ($facts^+$)

and one of removed facts (*facts*$^-$) when compared to the initial state. Every new action included in the plan has its effects (already instantiated) compared to these sets and to the initial state.

The manner whereby these two sets are updated causes the set *facts*$^+$ to contain only facts that are not part of the initial state, and the set *facts*$^-$ to contain only facts that are part of the initial state. Thus, positive effects can only have their facts added to *facts*$^+$ or removed from *facts*$^-$, whereas negative effects have their facts just added to *facts*$^-$ or removed from *facts*$^+$. Both the initial state and the two sets of facts are analyzed each time a new action is incorporated into the plan, in order to determine whether each new effect applied to the current state will be added to any of the two sets, removed from some of them, or whether nothing would be done. The algorithm in charge of this task is shown in Figure 5.2 and receives the following parameters:

- *initial_state*: set of positive facts describing the state from which the plot will be generated;
- *current_state*: structure containing the sets *facts*$^+$ and *facts*$^-$;
- *effects*: structure containing a set of positive effects and a set of negative effects.


```
// Update list with facts added to and removed from the initial state
Update_State(initial_state, current_state, effects)
    facts+ ← {positive facts of current_state}
    facts- ← { positive facts of current_state }
    effects+ ← {positive effects of effects}
    effects- ← {negative effects of effects}
    // Add positive effects that still are not part of the current state
    for each effect in effects+
        if effect ∈ facts- then remove effect from facts-
        else
            if effect ∉ initial_state then add effect to facts+
    // Remove negative effects that are not part of the current state
    for each effect in effects-
```

if *effect* ∈ *facts* $^{+}$ **then** remove *effect* de *facts*$^{+}$

else

if *effect* ∈ *estado_inicial* **then** add *effect* to *facts*$^{-}$

end

Figure 5.2 – Pseudocode for states update.

5.3.3. Goal Inference and Partial Order Planning

In order to allow the integration of the new planner with IPG, the latter also had to be modified (Silva 2010). IPG implements a partial order planner, obtaining new plans from a plan that is not yet a solution (i.e. where it is not true that all preconditions of all events are necessarily holding at a given time), as shown in Figure 5.3. The algorithm takes as input a partial plan *P* with some preconditions not established yet for some of its events and generates, if possible, a successor plan *S* to ensure the establishment of such a precondition. During the process the events can be labelled user, establisher or clobberer. An event *u* is considered a *user* of a literal *pre* if *pre* is a precondition of *u*. An event *est* is said to be an *establisher* of *pre* for *u* if *est* can occur before *u* and has some post-condition *pos* that can be unified with *pre*. An event *clob* is said to be a *clobberer* for the establishment of the precondition *pre* of user *u* if *clob* has some effect that can be unified with the negation of *pre*.

successor(P, U, PRE, SUCC)

Input

P: predecessor plan with unresolved precondition.
U: operator of **P** with precondition not necessarily valid.
PRE: precondition of **U** to become valid.

Output

SUCC: successor plan of **P**. Different values are returned by backtracking

1. Define candidates **CAND** to be successors of the plan **P** in which exist an *establisher* **EST** of **PRE** for **U**.
   1.1. Examine events **EST** in **P** that can occur before **U** and that have a post-condition **POS** that can be unified with **PRE**. Define **CAND** as the extension of **P** where **EST** establishes **PRE** for **U**.
   1.2. Examine new events **EST** in the context that can be inserted to establish **PRE** mandatorily before **U**. Define **CAND** as the extension of **P** where **EST** establishes **PRE** for **U**.
2. For each pair <**EST**, **CAND**>:
   2.1. Obtain the list of events (*clobberers*) **LCLOB** that cause conflict for the establishment of the precondition **PRE** of *user* **U** by *establisher* **EST**.
   2.2. Obtain all the possible combinations of the set of restrictions to be added to **CAND** so that no *clobberer* be in conflict with the establishment of **PRE** by **EST** for **U**. For each *clobberer* **CLOB** that establishes a post-condition **negation(Q)**, such as the literal **Q "possibly_codesignates"** with **PRE**, there are the following alternatives for conflict resolution:
      - Variable separation: make the parameters of **Q** and **PRE** be distinct. Each parameter in **Q** and **PRE** specified by variables or constants **X** and **Y**, respectively, creates a possibility of conflict resolution by adding the restriction **dif(X, Y)**. Obviously, it only works when **X** is note equal to **Y** (same variable or constant).
      - Force **CLOB** to occur before **EST**. It is only possible if it is not yet specified that **EST** precedes **CLOB**.
      - Force **CLOB** to occur after **U**. It is only possible if it is not yet specified that **CLOB** precedes **U**.
   2.3. For every set of restrictions, generate a successor **SUCC** to be returned by backtracking.

Figure 5.3 – IPG's *successor* algorithm (Ciarlini, 1999).

The changes made to the *successor* algorithm aim at enabling a proper treatment of attempts and limiting the number of events in the plan. These changes are listed below:

- It is permitted to have preconditions of the type *tried*(*COND*, *N*), where *COND* is a literal and *N* is an integer (the tolerance limit for the attempt). A precondition of type *tried*(*COND*, 0) is equivalent to a precondition *COND* that must be established;
- Events may have to take the form *attempt* (*OP*, *N*), indicating that the event is an attempt to execute *OP*, in which events up to level *N* of antecedence to *OP* in the chain of preconditions are permitted to fail. Operators before these must necessarily occur. An event of the type *attempt*(*OP*, 0) is equivalent to a mandatory occurrence of event *OP*, i.e. it is not an attempt;
- The choice of what can establish a precondition (item 1.1) was changed, due to the fact that a precondition can be of type *tried*(*COND*, *N*):
    - If *PRE* is not of type *tried*(*COND*, *N*), we consider the establishment of *PRE* taking into account only the deterministic effects of events that are already in the plan;
    - If *PRE* is of type *tried*(*COND*, *N*), with *N* = 0, we treat the establishment of *PRE* as the establishment of *COND*, like in the previous item;
    - If *PRE* is of type *tried*(*COND*, *N*), with N> 0, we consider the establishment of *PRE* by:
        - Deterministic effects of events that are not attempts and that establish *COND*;
        - Deterministic effects of events that are not attempts and that explicitly establish *tried*(*COND*, *N1*), with *N1* <= *N*;
        - Effects of nondeterministic events that are not attempts, if *N* = 1;

- Deterministic or nondeterministic effects of events of type *attempt*(*OP*, *N1*) that establish *COND*, with *N1* < *N*;
- Deterministic effects of events of type *attempt*(*OP*, *N1*) that explicitly establish *tried*(*COND*, *M*), with *M* + *N1* < *N*.

- The choice of new events that can be inserted into the plan to establish preconditions (item 1.2) was also changed as follows:
  - For preconditions that are not of type *tried*(*COND*, *N*), we consider only the inclusion of events where *PRE* can be established by the deterministic effects. The new event *EST* is an instance of *OP*;
  - For preconditions that are of type *tried*(*COND*, 1), we consider the inclusion of events where *COND* can be established by deterministic or nondeterministic effects. The new event *EST* is an instance of *OP*;
  - For preconditions of type *tried*(*COND*, *N*), with *N* > 1, we consider the inclusion of events where *COND* can be established by the deterministic or nondeterministic effects of an operator *OP*. The new event *EST* is an instance of *attempt*(*OP*, *N1*), with *N1* = *N* – 1;
  - The inclusion of new events is conditioned to the fact that the current plan does not exceed the limit for the inclusion of new events.
- Item 2.1 has also been modified to reconsider clobberings, in view of the introduction of the concept of attempts:
  - It is necessary to consider all the deterministic or nondeterministic effects that can clobber the establishment of a precondition. We examine the lists of deterministic and nondeterministic effects of all the events, checking whether they are attempts or mandatory events. Effects explicitly defined as *tried*(¬*COND*, *N*) – i.e. attempts to establish the negation of *COND*, whatever the tolerance is – must also be considered for the clobbering of the establishment of *COND*;

- In case *PRE* is of type *tried*(*COND*, *N*), we analyze the clobbering in the establishment of *COND*.

Additionally, the validation of an event precondition at the moment of its execution has to consider the following situations:

- There is an event that necessarily establishes the precondition, according to item 1.1, but is requiring that *EST* comes before *U* and that the effect is already instantiated with the precondition *PRE*;
- There is no other event that may clobber the establishment of the condition, according to the definitions at 2.1.

Finally, we ensured that the goals chosen by IPG match the personality models of the characters responsible for reaching them by adding a validation check on their emotional values, as seen on section 5.2.1.

5.3.4. Goal Separation

The original version of IPG scanned the context database searching for goal inference rules applicable at the current state. If more than one such rule was found, all the related goals were considered and the system would try to find a plan that might accomplish all of them. Since, in our new model, we want to define for the same goal alternative plans that can be performed by the same character, depending specifically on the character's personality traits, we had to make some changes to handle these goals separately, employing different rules. In order to maintain the old functionalities and permit IPG to plan for multiple goals (for example in case we do not want to use the model of personality traits), we created another nodule that uses IPG capabilities but generates a plan for each goal, sending a goal-plan list to the Chapter Simulator, and giving to the user the opportunity to select one of them.

5.4. User’s Preferences

In order to capture the qualitative aspect of the users' choices (cf. section 4.2.2), i.e. the characteristics of the events chosen to be incorporated into the plot, we developed a functionality that, working in a multiuser and multimodal

interaction environment, was able to keep track of the choices the users made by answering “yes” or “no” (or giving no answer at all) to the suggestions made by the system. User logs were stored, and another functionality was introduced capable of reading these logs, checking the users' choices against their characteristics based on atmospheric traits, and giving an average measure of the preferred traits for each user and for the whole group based on their “yes” choices. In such environments, it is possible to check if a user is deviating too much from the group as a whole, and even to make suggestions regarding whether a user should move to some other group. In this work, we added this verification capability, although we still must integrate it to the user interaction process in order to automatically track their choices.

6
Testing the Prototype

In order to test our implementation, we ran a few experiments using a tiny subset of the Chivalric Romance *genre*. It is staged in a mini-world whose initial state can be thus summarized:

Duke Baldwin is absent on a mission, leaving his wife, the lady Elaine, in the solitude of the White Palace. Count Duncan, Baldwin's sworn enemy, sees the duke's temporary absence as an opportunity to invade his domains. Sir Wilfrid, the bravest knight in the realm, is in love with Elaine, but is too shy to confess his feelings; moreover by doing so he would betray the duke, who absolutely trusts him. At the Black Castle lives Prince Morvid, who hates Sir Wilfrid and envies his high reputation.

Our interactive stories are generated according to a story context, which comprises a logical description of the mini-world wherein the narrative takes place, the characters' personality traits, a definition of the events that can be enacted by the characters, and a description of the motives that guide their behaviour.

As already mentioned in section 3.7.3, the elements that compose our story context can be divided in:

- **Static schema**: describes the mini-world wherein the plots take place (characters, relations, places…);
- **Dynamic schema**: describes the possible events in which the characters of the story can participate;
- **Behavioural schema**: describes the motives that guide the behaviour of the characters;
- **Personality schema**: details the personality traits for characters and events.

Specifying such contexts in Prolog is admittedly a worksome task, which remains until now a liability of the **Logtell** approach, requiring the help of authoring tools still to be developed.

While running the tests described in this chapter, we registered the execution times, which were as expected with such small examples. As duly recognized at the beginning of the preceding chapter, the use of client/server work distribution is mandatory for scaling-up the prototype's application.

.

6.1.Static Schema

The formal specification of the static schema is based on the standard syntax and semantics of first order languages and adopts the formalism of the Entity-Relationship model (Batini *et al*. 1992). It is composed by a set of facts indicating the existence of entities, the values of the attributes of entities, the existence of relationships, the values of the attributes of the relationships, and the assignment of roles to entities. An entity can represent anything of interest by itself, material or abstract, animate or not. A set of facts holding at a given instant of time constitutes a state.

The clause patterns used in the specification of our static schema are given below. In conformity with Prolog conventions, square brackets are used for conjunctive lists (with "," as separator) and round brackets for disjunctions (with ";" as separator).

```
entity(<entity-class>,<identifier>).
relationship(<relationship-class>,[<entity-class>,...,
                                                  entity-class>]).
attribute(<entity-class>,<attribute>).
attribute(<relationship-class>,<attribute>).
boolean(<attribute>).
is_a(<more-specialized-entity-class>,<more-general-entity-class>).
role(<role>,(<entity-class>;...;<entity-class>)).
```

In the context of a narrative, the major entity classes are usually the characters and the places where the story unfolds. The characters are identified by their names and may have a number of attributes describing their physical and emotional characteristics. Similarly, locations are also identified by a name and may have some attributes. Between characters and places there may be relationships, for example, indicating ownership or the current place of a character. Similarly, relationships between characters are also contemplated, for

example, indicating that a certain character loves another, or indicating that two characters are married.

The static scheme is created according to the conventions of the genre. The complete entity-relationship diagram of the various components of our context and the connections among them are shown in Figure 6.1.

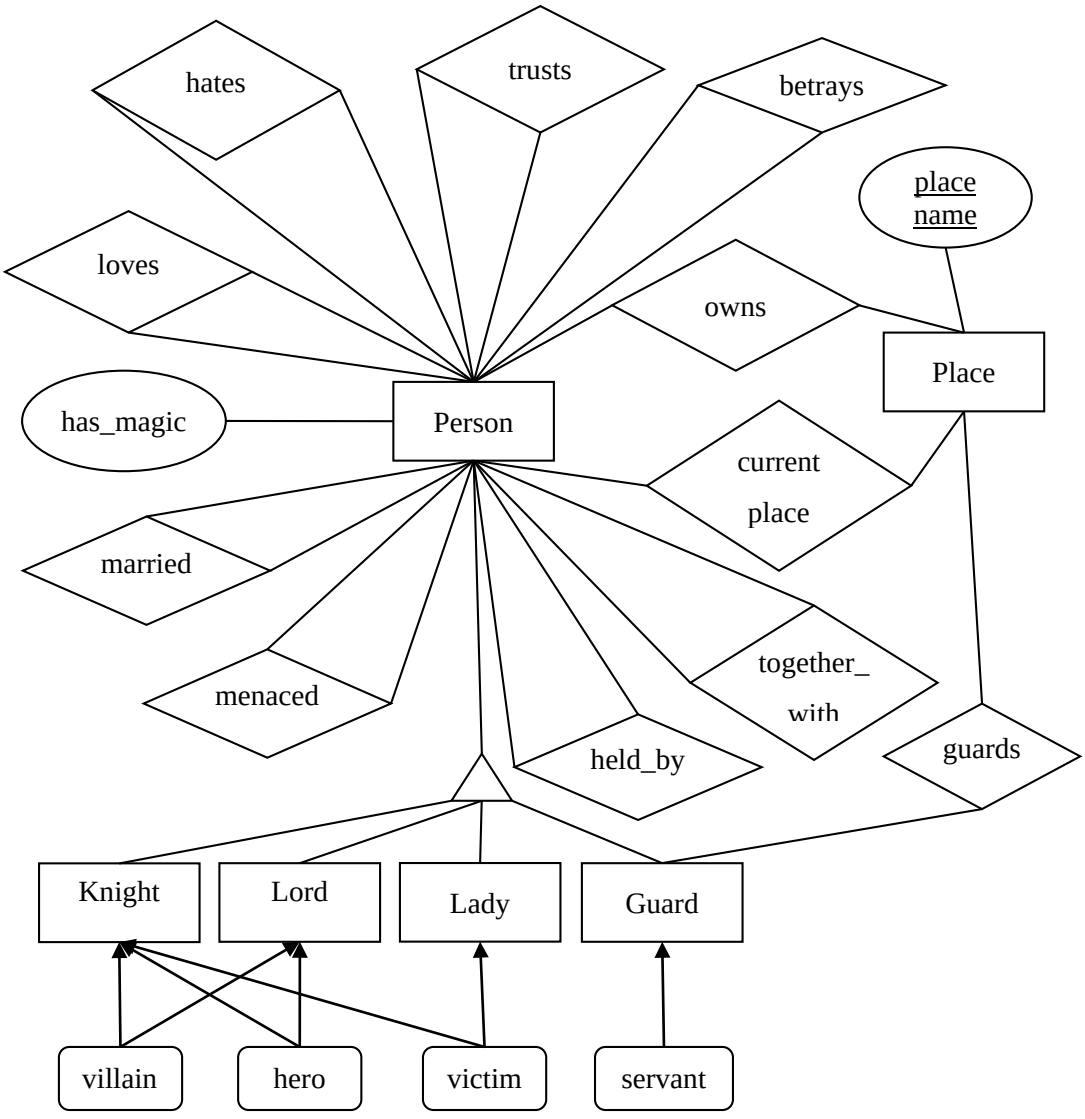

Figure 6.1 – Entity-Relationship diagram of the static schema.

6.2. Dynamic Schema

The dynamic schema contains the definition of the basic and complex operations. It is specified with the following syntax, wherein each operation is defined by an operation frame and an operation declaration:

```
operator_frame(<operator-id>, <operator-name>,
        [<case>:(<entity class or role>;...;
        <entity class or role>),...,
        <case>:(<entity class or role>;...;
        entity class or role>]).

operator(<operator-id>,<operator-name>(<parameter list>),
        [<pre-conditions>],
        [<deterministic effects>],
        [<lists of nondeterministic effects>]
        <estimated cost of operation>,
        [<main effects>],
        [<sub-operators>],
        [<ordering constraints>],
        <emotional flag>,
        [emotional effects]).
```

Of particular interest for us are the *emotional effects*, which are presented in the form `Agent:<personal traits frame>` (where `Agent` must be the same variable as one of the parameters in the parameter list), and permit the character personality to be changed by an event. The planner knows which trait to change because they have different formats for drives, attitudes and emotions.

The operations are specified in correspondence to the events that can occur in the narrative. One example is:

```
operator(11, abduction(M,W),
    [],
    [
        held_by(W,M),
        current_place(W,P1),
        not(current_place(W,P2))
     ],
    [],10,
    [held_by(W,M)],
```

```
        [
            (f1, ride(M,P1,P2)),
            (f2, defeat(M,G)),
            (f3, seize(M,W)),
            (f4, carry(M,W,P1))
        ],
        [(f1,f2),(f2,f3),(f3,f4)],true,[]) :-
                    db0(person(M)),
                    db0(person(W)),
                    db0(home(M,P1)),
                    db0(home(W,P2)),
                    db0(guards(P2,G)).
.
```

The operators can be defined hierarchically, in which case the HTN capabilities can be used to generate different plans for the same goals. One example of how these operators are structured is in Figure 6.2, where a generic operator is shown with its specializations. One is a composite operator and, among its sub-operators, one is a grouping operator (to be replicated into specialized versions, one of which is a composite operator, as will be seen later in this chapter). The formal specification of the specialization of `seduce` in a concrete notation is:

```
specialize(abduction(Agent,W), seduce(Agent,W)).
specialize(elopement(Agent,W), seduce(Agent,W)).
specialize(visit_under_disguise(Agent,W), seduce(Agent,W)).
specialize(proposal_by_proxy(Agent,_Proxy,W), seduce(Agent,W)).
```

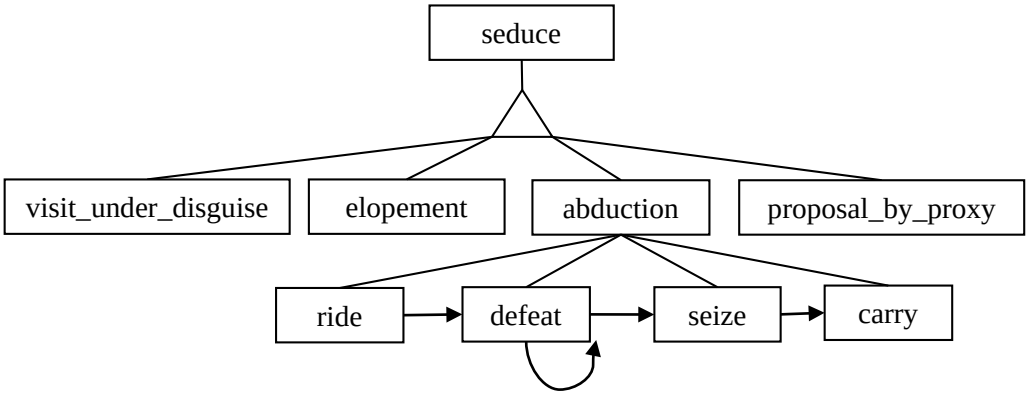

Figure 6.2 – Example of an operator structure containing a generic operator (seduce) with its specializations, one of them being a composite operator (abduction) whose list of sub-operators contains a grouping operator (defeat).

6.3. Behavioural Schema

The behavioural schema describes the motives that guide the conduct of the characters in the narrative. It consists of a set of goal-inference rules that capture the goals that motivate the characters' actions when certain situations occur during the narrative. An example of goal-inference rule for our narrative could be: "When a lord leaves his castle unprotected and his lady unaccompanied, someone will try to seduce her", which can be formally specified as:

```
e(i,person(Agent)) ∧ e(i,married(W,L)) ∧ e(i,owns(L,C))
∧ e(i,not(current_place(L,C)) → ∃T o(T,seduced(W,Agent))
```

where the metapredicate `e(T, LIT)` specifies that a literal `LIT` is established at time `T`, and the metapredicate `o(T, EV)` specifies that the event `EV` occurs at time `T`. The constant `i` is used to reference the initial time of the story. Variables that are not explicitly quantified are taken as universally quantified throughout the formula.

6.4. Personality Schema

In section 5.1.1, we showed the main aspects of the definitions related to this schema. Characters need to have their personality traits defined within frames, as happens to Wilfrid:

```
character('Wilfrid',D,A,E) :-
  D = [d1: 4,d2: 0,d3: 4,d4: 1],
  A = [a1: 3,a2: 0,a3: -4,a4: 1,a5: 1],
  E = [e1: 0,e2: 0,e3:0,e4: 4,e5: -1,e6: 0].
```

Likewise, we have the following goal frames:

```
goal_frame(protected(W,Agent):(Agent,[d1:4,d2:0,d3:0,d4:2])).
goal_frame(conquered(Agent,C):(Agent,[d1:0,d2:4,d3:0,d4:0])).
goal_frame(seduced(W,Agent):(Agent,[d1:0,d2:0,d3:4,d4: -3])).
```

```
goal_frame(pacified(Agent,V,L):(Agent,[d1:1,d2:0,d3:0,d4:2]).
```

And the following plan frames for the plans associated with `seduced`:

```
plan_frame(abduction(Agent,W):
                    (Agent,[a1:-3,a2:-2,a3:2,a4:-3,a5:0])).
plan_frame(elopement(Agent,W):
                    (Agent,[a1:3,a2:-2,a3:2,a4:-3,a5:1])).
plan_frame(visit_under_disguise(Agent,W):
                    (Agent,[a1:0,a2:3,a3:1,a4:-1,a5:3])).
plan_frame(proposal_by_proxy(Agent,P,W):
                    (Agent,[a1:3,a2:2,a3:-3,a4:3,a5:0])).
```

6.5. Running the Tests

All our tests were executed on a notebook ASUS UL30A 64 (1.30 GHz) with 8 GB of RAM and Windows 7 Home Premium. In 10 executions, the average time for the first phase was of 2.29s with a maximum of 3.26s and a minimum of 2.09s, while the average time for the second phase was of 0.08s with a maximum of 0.20s and a minimum of 0.04s.

6.5.1. Personality Decisions and Changes

When we activate our prototype, it first calls IPG, which will find all the goals applicable to the current state (*possible goals*). Since in our context there is no operator establishing the goal `protected(C1, C2)` (for any characters `C1` and `C2`), the two goals following this pattern are discarded when the process returns the *goals with plans*.

```
Possible goals:
[[seduced(Elaine,Morvid)],
 [seduced(Elaine,Wilfrid)],
 [pacified(Wilfrid,Duncan,Baldwin)],
 [protected(Elaine,Duncan)],
 [protected(Elaine,Wilfrid)],
 [conquered(Duncan,White Palace)],
 [conquered(Morvid,White Palace)]]

Goals with plans:
[seduced(Elaine,Morvid)]
[seduced(Elaine,Wilfrid)]
```

```
[pacified(Wilfrid,Duncan,Baldwin)]
[conquered(Duncan,White Palace)]
[conquered(Morvid,White Palace)]

Elapsed planning time (in seconds): 2.152.

Selected Goal: [seduced(Elaine,Morvid)]

Operations

i:initial
 condition:true
8:seduce(Morvid,Elaine)
 condition:true
1:gen_goal([seduced(Elaine,Morvid)])
 condition:true
g:goal([])
 condition:true

Constraints on Time:

8-[1]
1-[]

Constraints on Data:


Continue/Stop/Another/Query? (C/S/A/Q)
|:
```

If we then choose to continue, the prototype will call the nondeterministic HTN planner that will try to expand this plan with actions that are suitable to Morvid's personality. The first choice is `abduction` which, as shown in Figure 6.3, is a composite operator, with the sub-operators `ride` (to move the abductor to the place where the lady is), `defeat` (the guards who are protecting the place where the lady is), `seize` (the lady) and `carry` (the lady to the abductor's place). But note in the same figure that `defeat` is a grouping operator. The guards are associated to the place they protect by the following Prolog predicate:

```
guards('White Palace',['Eustace','Briol'])).
```

The definition of operator `defeat` establishes, as one of its conditions, that character `M` defeats `G`, the guards in charge of protecting place `P` (where `M` is):

```
operator(102, defeat(M,G), [ current_place(M,P) ], … ) :-
                           db0(person(M)), db0(guards(P,G)).
```

The (meta-)predicate `db0` is used to signalize facts of the initial state. We defined `defeat` as a grouping operator, indicating which of its arguments must be

expanded (in this case, the variable `Group` after the delimiter ":" will be unified with the second argument):

```
grouping(defeat(_,Group):Group).
```

One by one, the characters were specified:

```
db0(guard('Eustace')).
db0(guard('Briol')).
```

and a value for fear was supplied via the predicate `v_fear`, whose first parameter is a character and the second is a list of pairs `s:v`, where `s` is a situation and `v` the value of fear at situation `s` for the character in question. For Eustace we specified:

```
v_fear('Eustace',[true:10]).
```

with a single `s:v` pair. Situation `s` being `true` means that Eustace has always a level of fear equal to 10, i.e. Eustace is a coward.

We also created two new definitions for `defeat`, both of them using the predicate `fear` that gets the possible values (according to the current state) of `v_fear`:

```
operator(1021, defeat(M,G), [ current_place(M,P) ], …,
             [(f1, attack(M,C)), (f2, kill(M,C))],
             [(f1,f2)] ) :-
                    db0(person(M)), db0(guard(C)), fear(C,0).

operator(1022, defeat(M,G), [ current_place(M,P) ], …,
             [(f1, threaten(M,C))],
             [] ) :-
                    db0(person(M)), db0(guard(C)),
                    fear(C,F), F > 0.
```

For Eustace the value will always be equal to 10, as noticed. As we have not defined any clause `v_fear` for Briol, `fear` returns zero for him. The planning process goes as exposed below. Right below the HTN line, we can see the plan as it is stored internally by the planner. Next we see the contingency tree that

represents it in a more visual way, with all the possible final states listed – in this case, only one, wherein Elaine is held by Morvid at the Black Castle.

```
HTN = [(8,seduce(Morvid,Elaine))],[]

Plan = ride(Morvid,Black Castle,White Palace),
[true,(attack(Morvid,Eustace), [true,(kill(Morvid,Eustace),
[true,(threaten(Morvid,Briol), [true,(seize(Morvid,Elaine),
[true,(carry(Morvid,Elaine,Black Castle), [true,(leaf,[])])])])])])]

==========================================================================
PLAN:
==========================================================================

PLAN

   Initial State
      []
      []

   HTN
      [(8,seduce(Morvid,Elaine))],[]

   Plan
      -> ride(Morvid,Black Castle,White Palace)
      -> attack(Morvid,Eustace)
      -> kill(Morvid,Eustace)
      -> threaten(Morvid,Briol)
      -> seize(Morvid,Elaine)
      -> carry(Morvid,Elaine,Black Castle)

   Final States
------------------------------------------------------------
       FINAL STATE ID: 1

       Facts +
              [current_place(Elaine,Black Castle),held_by(Elaine,Morvid)]
       Facts -
              [current_place(Elaine,White Palace)]

------------------------------------------------------------

Number of leaves: 1.
Elapsed planning time (in seconds): 0.15.

Please select the desired outcome (0 to finish) -->
```

If, instead of continuing the process, we choose to select another plan, the prototype will pick the next goal (`seduced(Elaine,Wilfrid)`), giving us the following result:

```
Selected Goal: [seduced(Elaine,Wilfrid)]

Operations

i:initial
 condition:true
9:seduce(Wilfrid,Elaine)
 condition:true
2:gen_goal([seduced(Elaine,Wilfrid)])
 condition:true
```

```
g:goal([])
 condition:true

Constraints on Time:

9-[2]
2-[]

Constraints on Data:


Continue/Stop/Another/Query? (C/S/A/Q)
|:
```

If we then choose to continue the process, it will now look for a *seduction* plan suitable to Wilfrid's personality. Since in his case the attitude *outgoing* has a negative weight, the best match for him is `proposal_by_proxy`, whose value is coincidentally negative for this attitude.

```
HTN = [(9,seduce(Wilfrid,Elaine))],[]

Plan = proposal_by_proxy(Wilfrid,Duncan,Elaine), [true,(leaf,[])]

========================================================================
PLAN:
========================================================================

PLAN

   Initial State
      []
      []

   HTN
      [(9,seduce(Wilfrid,Elaine))],[]

   Plan
      -> proposal_by_proxy(Wilfrid,Duncan,Elaine)

   Final States
------------------------------------------------------------
      FINAL STATE ID: 1

      Facts +
             [likes(Wilfrid,Duncan),loves(Elaine,Wilfrid),
             together_with(Wilfrid,Elaine),betrays(Wilfrid,Baldwin)]
      Facts -
             []

------------------------------------------------------------

Number of leaves: 1.
Elapsed planning time (in seconds): 0.059.

Please select the desired outcome (0 to finish) -->
```

Here, Duncan acts as a proxy for the shy Wilfrid, informing Elaine that Wilfrid loves her. However, by doing so, Wilfrid indirectly betrayed his lord, as attested above by (`betrays(Wilfrid,Baldwin)`). In this case, if we inspect

Wilfrid's drives, we see that his sense of duty (d1) has changed from the maximum value (4) to zero:

```
D = [d1: 0,d2: 0,d3: 4,d4: 1]
```

because the specification of `seduce` has, as one of its effects when the seduced lady is married to someone who trusts the seducer, the change of d1 from a positive value to zero.

Figure 6.3 illustrates the computation by which the `proposal_by_proxy` plan was obtained.

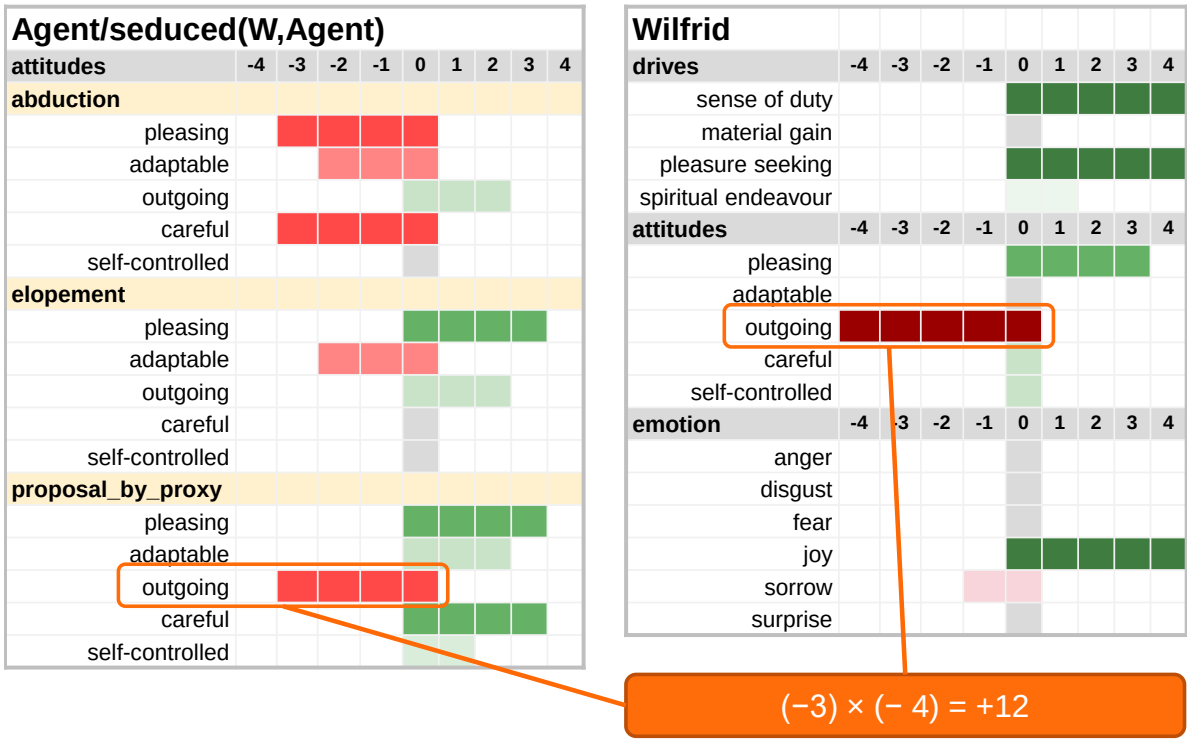

Figure 6.3 – Obtaining a ranking value for the plan, considering the plan's values and the character's attitude weights.

6.5.2. Nondeterminism

One way of testing the use of nondeterministic events and attempts is by replacing the deterministic event `kill(M,C)`, whereby `M` kills `C`, by the nondeterministic event `try_to_kill(M,C)`, where either `M` or `C` – or even both – can be killed. In this case, the sub-operators `seize` and `carry` become *attempts* of executing these operators in the modified definition of `abduction` (Figure 6.4) and, having both the precondition that the agent be alive, they will fail and not be

executed in states where Morvid is dead; and yet the planning process succeeds and it is possible to continue the story from any of its three branches.

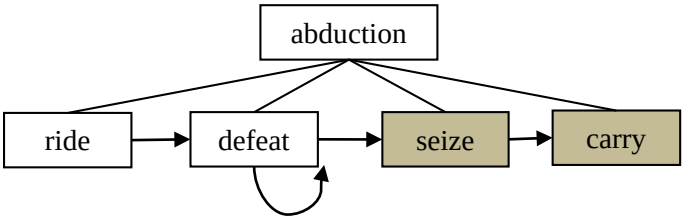

Figure 6.4 – New definition for abduction: gray boxes represent attempts to execute the respective operators.

Here is the new plan for `seduce(Morvid,Elaine)`:

```
HTN = [(8,seduce(Morvid,Elaine))],[(8,1)]

Plan = ride(Morvid,Black Castle,White Palace),
[true,(attack(Morvid,Eustace),[true,(try_to_kill(Morvid,Eustace),
[[[not alive(Morvid)],(leaf,[])],
[[not alive(Eustace)],(threaten(Morvid,Briol),
[true,(seize(Morvid,Elaine),[true,(carry(Morvid,Elaine,Black Castle),
[true,(leaf,[])])])])])],
[[not alive(Morvid),not alive(Eustace)],(leaf,[])]])])]

==========================================================================
PLAN:
==========================================================================

PLAN

   Initial State
      []
      []

   HTN
      [(8,seduce(Morvid,Elaine))],[(8,1)]

   Plan
      -> ride(Morvid,Black Castle,White Palace)
      -> attack(Morvid,Eustace)
      -> try_to_kill(Morvid,Eustace)
      [?][not alive(Morvid)]
      [?][not alive(Eustace)]
         -> threaten(Morvid,Briol)
         -> seize(Morvid,Elaine)
         -> carry(Morvid,Elaine,Black Castle)
      [?][not alive(Morvid),not alive(Eustace)]

   Final States
--------------------------------------------------------------------------
      FINAL STATE ID: 1

      Facts +
         [current_place(Morvid,White Palace)]
      Facts -
```

```
         [alive(Morvid),current_place(Morvid,Black Castle)]

------------------------------------------------------------------------
      FINAL STATE ID: 2

      Facts +
         [current_place(Elaine,Black Castle),held_by(Elaine,Morvid)]
      Facts -
         [alive(Eustace),current_place(Elaine,White Palace)]

------------------------------------------------------------------------
      FINAL STATE ID: 3

      Facts +
         [current_place(Morvid,White Palace)]
      Facts -
         [alive(Eustace),alive(Morvid),current_place(Morvid,Black
          Castle)]

------------------------------------------------------------------------

Number of leaves: 3.
Elapsed planning time (in seconds): 0.151.

Please select the desired outcome (0 to finish) -->
```

6.5.3. Calibrating the Values

We can also check how the changes in the personality traits can cause variations in the plot. When first started, the prototype presents a scenario to introduce the mini-world to the user, to whom it offers the opportunity to impersonate one of the active characters:

```
   Duke Baldwin is absent on a mission, leaving his wife, the lady
Elaine, in the solitude of the White Palace. Count Duncan, Baldwin's
sworn enemy, sees the duke's temporary absence as an opportunity to
invade his domains. Sir Wilfrid, the bravest knight in the realm, is in
love with Elaine, but is too shy to confess his feelings; moreover by
doing so he would betray the duke, who absolutely trusts him. At the
Black Castle lives Prince Morvid, who hates Sir Wilfrid and envies his
high reputation.

Please choose the character you want to impersonate:

[1] Sir Wilfrid
[2] Count Duncan
[3] Prince Morvid
[0] to finish character selection.
```

Upon selecting a character, the user is prompted to choose between the two different strategies for tailoring the drives:

```
Your choice --> 1.
Viewer has chosen to impersonate Sir Wilfrid.
Please choose how you want the drives to be modelled:

[1] I want to model the character explicitly.
[2] I want to choose a stereotype for the character.
```

```
Your choice --> |: 2.

[1] Hero.
[2] Villain.

Your choice --> |: 2.
```

As a consequence of the above interaction, Wilfrid assumes a villain stereotype. Now it is time to calibrate the attitudes:

```
Please choose how you want the attitudes to be modelled:

[1] I want to model the character explicitly.
[2] I want to be submitted to a short personality test and have the
character modelled after me.

Your choice --> |: 1.

Here are a number of personality traits associated to the character that
you chose to impersonate. Please write a number next to each personality
trait to indicate how strongly you want that trait to affect the
character's personality (in a positive, negative or neutral way).

-------------------------------------------------------------------------
very strong|strong|average|weak|neutral|weak|average|strong|very strong
negative   | neg  | neg   | neg|       |pos | pos   | pos  |positive
<----------<------<-------<----<------->----->------->------>------------>
     1         2      3      4     5     6      7      8        9
-------------------------------------------------------------------------------------------------------------------------------
1. Pleasing.        Your choice --> |: 3.
2. Adaptable.       Your choice --> |: 4.
3. Outgoing.        Your choice --> |: 2.
4. Careful.         Your choice --> |: 1.
5. Self controlled. Your choice --> |: 7.

Outgoing:        -3
Pleasing:        -2
Careful:         -4
Self Controlled:  2
Adaptable:       -1

Wilfrid:
   Before:[d1:4,d2:0,d3:4,d4:1][a1:3,a2:0,a3: -4,a4:1,a5:1]
   After :[d1: -4,d2:3,d3:4,d4:0][a1: -2,a2: -1,a3: -3,a4: -4,a5:2]
```

So the chosen option was to explicitly define the values for the attitudes and, in the end, the prototype shows us the effects of these changes. The prototype then removes Wilfrid from the list and gives the opportunity to adjust another character. Suppose now that the user chooses to modify Morvid's personality by explicitly changing his drives and utilizing the personality test to calibrate the attitudes:

```
Please choose the character you want to impersonate:

[1] Count Duncan
[2] Prince Morvid
[0] to finish character selection.
```

```
Your choice --> |: 2.
Viewer has chosen to impersonate Prince Morvid.
Please choose how you want the drives to be modelled:

[1] I want to model the character explicitly.
[2] I want to choose a stereotype for the character.

Your choice --> |: 1.

Here are a number of personality traits (drives) associated to the
character that you chose to impersonate. Please write a number next to
each personality trait to indicate how strongly you want that trait to
affect the character's personality (in a positive, negative or neutral
way).

-------------------------------------------------------------------------
very strong|strong|average|weak|neutral|weak|average|strong|very strong
negative   | neg  | neg   | neg|       |pos | pos   | pos  |positive
<----------<------<-------<----<------->----->------->------>------------>
     1         2      3      4     5     6      7      8        9
-------------------------------------------------------------------------
1. Sense of duty.          Your choice --> |: 9.
2. Material gain.          Your choice --> |: 1.
3. Pleasure seeking.       Your choice --> |: 1.
4. Spiritual endeavour.    Your choice --> |: 9.

Sense of Duty:        4
Material Gain:       -4
Pleasure Seeking:    -4
Spiritual Endeavour:  4

Please choose how you want the attitudes to be modelled:

[1] I want to model the character explicitly.
[2] I want to be submitted to a short personality test and have the
character modelled after me.

Your choice --> |: 2.

Here are a number of personality traits that may or may not apply to you.
Please write a number next to each statement to indicate the extent to
which you agree or disagree with that statement. You should rate the
extent to which the pair of traits applies to you, even if one
characteristic applies more strongly than the other.

-------------------------------------------------------------------------
Disagree|Disagree  |Disagree|Neither agree|Agree   |Agree     |Agree
strongly|moderately|a little|nor disagree |a little|moderately|strongly
<-------<----------<--------<------------>-------->---------->-------->
    1        2        3         4          5        6       7
-------------------------------------------------------------------------
I see myself as:

1. Extraverted, enthusiastic.           Your choice --> |: 7.
2. Critical, quarrelsome.               Your choice --> |: 1.
3. Dependable, self-disciplined.        Your choice --> |: 1.
4. Anxious, easily upset.               Your choice --> |: 2.
5. Open to new experiences, complex.    Your choice --> |: 6.
6. Reserved, quiet.                     Your choice --> |: 2.
7. Sympathetic, warm.                   Your choice --> |: 4.
8. Disorganized, careless.              Your choice --> |: 6.
9. Calm, emotionally stable.            Your choice --> |: 7.
10. Conventional, uncreative.           Your choice --> |: 1.
```

```
Extraversion:            6.5
Agreeableness:           5.5
Conscientiousness:       1.5
Emotional Stability:     6.5
Openness to Experiences: 6.5

Outgoing:         3
Pleasing:         2
Careful:         -3
Self Controlled:  3
Adaptable:        3

Morvid:
   Before:[d1: -4,d2:3,d3:4,d4:0][a1: -3,a2: -3,a3:3,a4: -3,a5:0]
   After :[d1:4,d2: -4,d3: -4,d4:4][a1:2,a2:3,a3:3,a4: -3,a5:3]
Please choose the character you want to impersonate:

[1] Count Duncan
[0] to finish character selection.

Your choice --> |: 0.
```

If the planner is called again, we can now see that the list of goals has changed, now including one where Morvid tries to make peace between Duncan and Baldwin (something that better suits his new drives), and another one where Wilfrid, now a villain, wants to conquer the White Palace:

```
Goals with plans:
[seduced(Elaine,Wilfrid)]
[pacified(Morvid,Duncan,Baldwin)]
[conquered(Duncan,White Palace)]
[conquered(Wilfrid,White Palace)]

Elapsed planning time (in seconds): 2.327.

Selected Goal: [seduced(Elaine,Wilfrid)]

Operations

i:initial
 condition:true
7:seduce(Wilfrid,Elaine)
 condition:true
1:gen_goal([seduced(Elaine,Wilfrid)])
 condition:true
g:goal([])
 condition:true

Constraints on Time:

7-[1]
1-[]

Constraints on Data:


Continue/Stop/Another/Query? (C/S/A/Q)
|: c
```

If the user chooses to continue, it will happen that, because of Wilfrid's new attitudes' frame, the same goal of seduction that previously led to a proposal by proxy will now lead to an elopement:

```
HTN = [(7,seduce(Wilfrid,Elaine))],[]

Plan = elopement(Wilfrid,Elaine),[true,(leaf,[])]

==========================================================================
PLAN:
==========================================================================

PLAN

   Initial State
      []
      []

   HTN
      [(7,seduce(Wilfrid,Elaine))],[]

   Plan
      -> elopement(Wilfrid,Elaine)

   Final States
--------------------------------------------------------------------------
      FINAL STATE ID: 1

      Facts +
         [together_with(Wilfrid,Elaine)]
      Facts -
         []

--------------------------------------------------------------------------

Number of leaves: 1.
Elapsed planning time (in seconds): 0.031.

Please select the desired outcome (0 to finish) -->
```

7
Conclusion

7.1.
Concluding Remarks

Despite the fact that interactive storytelling (IS) has been researched for decades already, it still brings a feeling of “new media”, possibly due to the fact that it has not established itself until now as a mainstream subject. The involvement with the huge and rich games industry may help fomenting research and bring some exposition to IS, but, as a research field on its own right, it still has a long road to run.

One of the biggest challenges is to find a flexible and attractive manner of presenting the stories. Finding a visually appealing way to dramatize the plots, like animations and video, while making this dramatization capable of reflecting subtle variations in the plot is difficult and costly. Fortunately some recent proposals, like the comics-based (Lima *et al*. 2013) and video-based (Lima 2014) approaches, appear to indicate promising directions to deal with this issue.

Another challenge is the difficulty encountered when proceeding to author interactive narratives. In IS, as in any form of storytelling, the author is the key factor in the production of a successful story, but authoring for interactive storytelling involves logically specifying the context of the story. This includes thinking about possible events in terms of parameters, preconditions and effects, whereas story writers are seldom familiar with these tasks – and even for researchers trying to test their implementations it is until today a demanding and time-consuming task. The development of good and flexible authoring tools is a problem yet to be solved.

But probably the most pressing challenge in IS is to generate stories that are interesting enough to the user – in fact, this is the aspect we are endeavouring to cope with in this work. It requires both the creation of flexible tools, hopefully including the ones we proposed here, and an arduous authorial effort to specify the intended genre in a form intelligible to the computer processes. And, despite

all precautions, the question remains: will computers, at some point, create stories as good as those created by humans?

We believe that this question can be addressed in different ways. One is to focus on the aspect of creativity: indeed, stories are highly praised if they are deemed to possess this quality. Simulate or replicate human creativity using a computer is the main goal of the field of *computational creativity*, a multidisciplinary mix of artificial intelligence, cognitive psychology, philosophy and arts. One of its major problems is to find a way to induce a machine, that was initially fabricated to do only what it is programmed to do, to break conventions and act in an unexpected manner, as creativity requires (Amabile 1983). One speculative answer may come from an analogy with the supposition that, once endowed with a complete understanding of the laws of the Universe and enough computational speed and sensory capabilities, one would end up regarding no physical event as random – not even the throw of a dice or the flip of a coin. Similarly, a sufficiently complex computer system, unless programmed to strictly censor the communication of responses, could eventually bring forward some unexpected and interesting results. However, the prospect of such devices coming to life, at least as far as interactive storytelling systems are concerned, seems very distant from the present reality.

A more optimistic way to address this question is to put aside the preoccupation with mechanical creativity and use and explore the structures and themes that some scholars have already identified as recurring in traditional human storytelling. Resorting to these elements ought to be a fairly secure way of making interesting stories, as they have been approved by people for millennia. Some creativity should be proportioned with the mutual aid of good IS systems and good IS authors.

Finally, another approach to the question is to consider that IS is a whole new subject and, as such, it does not have to be compared with more established storytelling vehicles, such as literature, theatre or cinema – in the same way as it would be wrong to compare cinema with theatre when cinema had scarcely been invented. The key novelty of interactive storytelling, that differentiates it from other storytelling forms, none of them able to compete in this regard, is *interactivity* whereby users gain the power to imprint a personal bias on the resulting diversity of plots. Of course minutely specified computer support still

has to be provided to convey recommendations to the user, but interactivity and the consequent customized plot variations really make a difference.

7.2. Major Contributions

In a brief summary, our main contribution was to present an architecture and a prototype for the generation of interactive plots, whose functionality included among other features: (1) nondeterministic planning in order to augment the possibility of plot variations; (2) the combination of the IPG planner, developed for the original **Logtell** prototypes, with HTN planning to speed up plan generation; (3) a decision-making process based on personality traits, which determines the behaviour of the characters participating in a story; (4) the ability to model one or more users according to some of these personality traits. In what follows, we shall provide a more detailed account.

Working on the topic of interactive storytelling (IS), we proposed an architecture that combines and extends previous versions adopted in the **Logtell** project, and, to enable practical experiments, an interactive storytelling system capable of formulating decisions concerning the acting characters' goals and plans, based on a model of modifiable (either by the user or automatically) personality traits. The system also copes with nondeterministic and failed events and permits the definition of composite, abstract and grouping operators that are instantiated into plot events. When combining these features, we aimed at one basic objective: to achieve a greater diversity in the generation of plots. Directly related to this contribution is the fact that the system confers a greater flexibility to the authoring process, allowing to adopt a mixed plot-based and character-based perspective when specifying the intended genre.

Before utilizing the ability to modify personality traits, individual users are offered the option to submit themselves to a personality test borrowed from psychology. This is especially convenient if they want to impersonate some character, letting them feel more engaged in the narrated plots. A realistic calibration of the characters' personalities generally leads to a more convincing chain of events, guiding the plot to some plausible outcome.

Believable stories can indeed be achieved only if all characters behave in a manner consistent with their personality, over and above their other physical and mental attributes. The fact that personality traits can change according to the events of the plot is another source of verisimilitude, since human beings are bound to evolve and change – as is extensively explored in the *Bildungsroman* (novel of education) genre. Moreover, the evolution of characters also permits dramatic turns, again contributing to flexibility and user engagement, with the possibility of more interesting stories.

As a basis for our work, we used the latest available version of **Logtell**, which already incorporated some extensions over the initial version of the nondeterministic planner of our M.Sc. dissertation. The modifications we made to its architecture and algorithms, besides further extending the system and permitting new possibilities, stay compatible with those extensions. Also, the way our new version was implemented does not prevent **Logtell** to run over contexts that do not consider the new features exposed in this work, which qualifies our system as a proper extension of **Logtell**, conforming to the backward compatibility requirement.

We also contributed an overview and some general ideas related to interactive storytelling, and how it can borrow from traditional human storytelling as well as from sciences and arts, notably psychology and literature. The works reviewed here show many interesting concepts and implementations. And, although much remains to be done before IS reaches maturity as an academic research field, we believe that it is a promising area of research, and expect that our effort has somehow contributed to sustain this opinion.

7.3. Publications and Awards

During the research that led to this work, we addressed certain aspects of interactive storytelling in papers submitted to conferences on artificial intelligence and entertainment computing. The first version of the nondeterministic planning process described here was published in the *International Conference on Tools with Artificial Intelligence* (Silva *et al*. 2011). Another aspect of IS tackled during the research, but not explored in the final version of the thesis, concerns the

incorporation of information-gathering events in story plots. Our work on this was first published in the series *Monografias em Ciência da Computação* (MCC 07/11), PUC-Rio (Silva & Furtado 2011), and later in the *International Conference on Entertainment Computing* (Silva *et al*. 2012).

We also collaborated in a paper introducing a decision-making method based on personality traits, which was published in the journal *Computers in Entertainment* (CiE) of the *Association for Computing Machinery* (ACM) (Barbosa *et al*. 2014), and would eventually constitute one of the main foundations of the thesis.

Another work, developed in collaboration in the course of our doctoral research, was the elaboration of a comic-based interactive narrative designed for interactive TV, called *Little Gray Planet*. This project received an honorable mention on "Interactivity" in the *2nd ITU IPTV Application Challenge competition* (2012), sponsored by the *International Telecommunication Union* (ITU, the United Nations agency for information and communication technologies).

7.4. Future Work

The research for this work revealed some new issues, and highlighted old ones that still deserve to be addressed, towards the advancement of digital storytelling. One of the main issues faced by every initiative in the **Logtell** project is the effort taken by the authoring process, especially in view of the option, adopted since the beginning of the project, that every element of the story genre and context would be specified in logic programming notation, and implemented via Prolog code. This always comprised, among other items, character roles, all sorts of entities, attributes, relationships, operationally-defined events with their pre- and post-conditions (with multiple branches in nondeterministic events) and in some cases hierarchical structure, and goal-inference rules. And, from now on, it also includes personality traits to which numeric values and weights must be adequately assigned to influence the course of the events. So, any kind of *authoring tools* would be a most welcome addition to **Logtell**.

Any inconsistency or other kind of mistake, even in the form of trivial typos, can be a threat to the faithful specification of the genre, with the

consequence that the planner might fail to generate the plots intended by the author. Therefore a *consistency-checking* mechanism built into the author's interface should also prove to be a useful resource, and indeed an investigation regarding this feature has actually been contemplated during our preliminary studies, but had to be left for a later moment due to its complexity.

Another fact to be taken into consideration is that we are not covering all the three levels prescribed in narratology studies, namely fabula, story and text (Bal 1997). We treat no more than the generation of plots, which corresponds to the *fabula* level wherein *what* happens is indicated, and the final *text* level wherein the plot is materialized in some medium (written text, movie, animation, etc.), and which we handle at the dramatization phase. For fully-fledged composition of narratives one still has to address the intermediate *story* level, wherein several artistic skills pertaining to literary craft are applied to figure *how* to tell what happens (in reality or fiction), a frequently occurring example being the decision to organize the course of action in non-chronological sequence.

Also, in order to effectively explore the entertainment potential of our extended planning prototype, it is necessary to go further in its integration within **Logtell** in order to establish its linkage to the various dramatization tools developed by other members of the team. It goes without saying that such integration would bring another benefit: moving from the bare Prolog standard environment to the more user-friendly graphical user interface of **Logtell**, after adaptations to allow running our additional facilities, such as handling nondeterministic events and permitting users to modify personality traits as the narrative proceeds.

Also desirable is to rethink the way the three-level decision making process is now working as an integral part of the planning algorithms. In the originally proposed version of the process – as well as in our implemented prototype – drives are used for goal-selection and attitudes for plan-selection. However only in the original proposal (not in the prototype, as yet) emotions were used to decide whether or not to commit, i.e. to either execute the chosen plan or backtrack to generate another plan and again test for commitment. To lead to a decision on the basis of a criterion of emotional satisfaction, the original process compares the character's situation at the current state with the prospective situation at the state to be reached by the plan. In our prototype, implicit commitment to the chosen

plan is assumed, even though emotions are taken into consideration as character attributes that are subject to change pursuant to the occurrence of events.

The decision making process itself could be enhanced by being combined with the multiuser model proposed by Camanho (2014) for **Logtell**. The introduction of multiuser environments would also permit the creation of mechanisms to produce game-like stories, letting every user be in charge of one character (using the idea of multiple character calibration presented in this thesis), in a competition to reach some determined goal, the winner being the first to achieve it.

An interesting idea in IS research is to rely on well-known stories, occasionally told and retold in multiple versions, where users can easily recognize famous characters with whom they could possibly identify (Furtado 2015). Starting from such stories, one would then create a context to try further variations, and even transpositions to different times and places (e.g. reviving the King Arthur or the Robin Hood universe). In addition, we could expand the array of Propp's roles in order to make it possible to produce variations not only in the story events, but also in character-role casting. So, for example, the main antagonist (playing the role of a villain) of a Robin Hood story could be either the Sheriff of Nottingham or Sir Guy of Gisbourne – and, in consequence of their different attributes (especially personality traits), the stories should differ substantially, while still remaining plausible within a well-built and flexible enough context specification.

We also think that a combination of Proppian functions (as in **Logtell)**, with Aarne-Thompson catalogue of folktale types and motifs (incorporating some ideas already tackled by Karlsson (2010)), plus Campbell's monomyth scheme or Polti dramatic situations, can be a means to gain extra power towards generating more interesting and diversified stories.

The use of IS out of the scope of entertainment is another aspect to be further explored. Ciarlini & Furtado (2002) have already addressed the use of **Logtell** in a business context. We might equally apply these IS ideas to corporate training, business games or even cultural assimilation (Fiedler *et al*. 1971), with the system presenting narratives that expose the user to culturally challenging situations, something especially useful for companies intent on training employees assigned to work overseas. In this case, the personality models could be

particularly apt to model cultural stereotypes. Another idea is the usage of IS for educational purposes, for example by re-enacting historical events, trying alternative outcomes and asking what-if questions, thus making students engage and interact with the facts they only knew from school books.

Finally, a more comprehensive integration with some of the extensions recently added to **Logtell** remains in order, in particular to combine the dramatic properties of the story events (Gottin 2013; Ferreira 2013) with our model of personality traits. The incorporation of novel features developed outside the scope of **Logtell**, like information-gathering events in the plots (Silva *et al*. 2012) and capturing the general preferences of an audience (Baffa 2015), should also be part of efforts to bring together distinct but complementary approaches.

References

AAMODT, A.; PLAZA, E. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *Artificial Intelligence Communications*, 7 (1), pp. 39-52, 1994.

AARNE, A. *The Types of the Folktale: A Classification and Bibliography*. Translated and Enlarged by Thompson, S. 2nd rev. ed. Helsinki: Suomalainen Tiedeakatemia / FF Communications, 1961.

ALLPORT, G. W. *Personality: A psychological interpretation*. New York: Holt, 1937.

ALLPORT, G. W.; ODBERT, H. S. *Trait-names: A psycho-lexical study*. Psychological Monographs, 47, No. 211, 1936.

AMABILE, T. *The social psychology of creativity*. New York, NY: Springer-Verlag, 1983.

ANDREWS, D.; HULL, D. Storytelling as an Instructional Method: Descriptions and Research Question. *The Interdisciplinary Journal of Problem-Based Learning*, 3(2): 6–23, 2009.

ANTHONY, S. Eugene Goostman Becomes the First AI to Pass the Turing Test, Convincing Judges That He's a 13-year-old Boy. *ExtremeTech*, 2014. Available at: <http://www.extremetech.com/extreme/183851-eugene-goostman-becomes-the-first-computer-to-pass-the-turing-test-convincing-judges-that-hes-a-13-year-old-boy>. Retrieved 20 March 2015.

ARAUJO, E. T. *Verificação de Restrições com Tempo Contínuo em Storytelling Não-Determinístico.* M.Sc. Thesis. Departamento de Informática, Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2011.

ARAUJO, E. T.; CIARLINI, A. E. M. Verification of Temporal Constraints in Continuous Time on Nondeterministic Stories. *In: 10th International Conference*

on Entertainment Computing, 2011, Vancouver. Lectures Notes in Computer Science. Berlin: Springer, v. 6972, p. 28-34, 2011.

ARISTOTLE. *The Rhetoric of Aristotle*. New York, NY: Appleton-Century-Crofts. An expanded translation with supplementary examples for students of composition and public speaking, by L. Cooper, 1960.

ARISTOTLE. Poetics. *In*: *Classical Literary Criticism*. P. Murray et al. (trans.). Penguin, 2000.

BAL, M. *Narratology*. Toronto: University of Toronto Press, 1997.

BATINI, C.; CERI, S.; NAVATHE, S. B. *Conceptual Database Design: an Entity Relationship Approach*. Addison-Wesley, Boston, USA, 1992.

BAUMGARTEN, F. Die Charktereigenschaften. *In: Beitraege zur Charakter- und Persoenlichkeitsforschung* (Whole No. 1). A. Francke, Bern, Switzerland, 1933.

BAFFA, A. C. E. *Storytelling Based on Audience Social Interaction*. D.Sc. Thesis, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brazil, 2015.

BARBOSA, S. D. J.; FURTADO, A. L.; CASANOVA, M. A. C. A Decision-making Process for Digital Storytelling.*Proceedings of SBGames 2010,* Florianópolis, 2010.

BARBOSA, S. D. J.; SILVA, F. A. G.; FURTADO, A. L. *Early cases of Bertillon, the logic programming sleuth*. Monografias em Ciência da Computação Series (MCC 08/12), PUC-Rio, Brazil, 2012.

BARBOSA, S. D. J.; SILVA, F. A. G.; FURTADO, A. L.; CASANOVA, M. A. Plot Generation with Character-Based Decisions. *Computers in Entertainment*, v. 12, p. 1-21, 2014.

BARRY, P. Structuralism. *Beginning theory: an introduction to literary and cultural theory*. Manchester University Press, Manchester, pp. 39–60, 2002.

BATES, J.; LOYALL, A. B.; REILLY, W. S. An Architecture for Action, Emotion, and Social Behavior. *In: Proceedings of the Fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, S. Martino al Camino, Italy, 1992.

BENET-MARTÍNEZ, V.; JOHN, O. P. 'Los Cinco Grandes' Across cultures and ethnic groups: Multitrait-multimethod analyses of the Big Five in Spanish and English. *Journal of Personality and Social Psychology*, 75, 729–750, 1998.

BERKELEY, E. C. *Giant Brains, or Machines That Think*. John Wiley & Sons, Inc. New York, USA, 1949.

BOOLE, G. *An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities*. Macmillan, 1854. Reprinted with corrections, Dover Publications, New York, NY, 1958. (Reissued by Cambridge University Press, 2009).

BORGATTA, E. F. The structure of personality characteristics. *Behavioral Science*, 9, 8-17, 1964.

BREAZEAL, C. Emotion and sociable humanoid robots. *Int. J. Human-Computer Studies*, 59, pp. 119–155, 2003.

BRYANT, R. E. Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams. *ACM Computing Surveys, Vol. 24, No. 3* (September, 1992), pp. 293–318, 1992.

BUCHANAN, B. G.; SUTHERLAND, G. L.; FEIGENBAUM, E. A. Heuristic DENDRAL: A program for generating explanatory hypotheses in organic chemistry. *In*: Meltzer, B., Michie, D., and Swann, M. (Eds.), *Machine Intelligence 4*, pp. 209-254. Edinburgh University Press, Edinburgh, Scotland, 1969.

CAMANHO, M. M. *Conciliando Coerência E Responsividade Em Storytelling Interativo*. M.Sc. Thesis, Departamento de Informática Aplicada, Universidade Federal do Estado do Rio de Janeiro, Brazil, 2009.

CAMANHO, M. M. *A Model for Stream-based Interactive Storytelling*. D.Sc. Thesis, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brazil, 2014.

CAMANHO, M. M.; CIARLINI, A. E. M.; FURTADO, A. L., POZZER, C.T.; FEIJÓ, B. Conciliating Coherence and High Responsiveness in Interactive Storytelling. *In: Proceedings of the 3rd ACM International Conference on Digital Interactive Media in Entertainment and Arts (DIMEA 2008)*, Atenas, 2008.

CAMANHO, M. M.; CIARLINI, A. E. M.; FURTADO, A. L.; POZZER, C. T.; FEIJÓ, B. A Model for Interactive TV Storytelling. *In: VIII Brazilian Symposium on Digital Games and Entertainment*, Rio de Janeiro, Brazil, pp. 197-206, 2009.

CAMPBELL, J. *The hero with a thousand faces.* Princeton University Press, USA, 1968.

ČAPEK, K. *R.U.R.* translated by Paul Selver and Nigel Playfair. Dover Publications, 2001.

CATTELL, R. B. The description of personality: Basic traits resolved into clusters. *Journal of Abnormal and Social Psychology*, 38, 476-506, 1943.

CATTELL, R. B. The description of personality: Principles and findings in a factor analysis. *American Journal of Psychology*, 58, 69-90, 1945a.

CATTELL, R. B. The principle trait clusters for describing personality. *Psychological Bulletin*, 42, 129-161, 1945b.

CATTELL, R. B., EBER, H. W., TATSUOKA, M. M. *Handbook for the Sixteen Personality Factor Questionnaire (16PF)*. Champaign, IL: IPAT, 1970.

CAVAZZA, M.; CHARLES, F.; MEAD, S. Character-based interactive storytelling. *IEEE Intelligent Systems, special issue on AI in Interactive Entertainment*, 17(4):17-24, 2002.

CHARLES, F.; CAVAZZA, M.; MEAD, S. *Character-driven story generation in interactive storytelling*. Technical report, VSMM, Berkeley, 2001.

CHATMAN, S. *Story and Discourse: Narrative Structure in Fiction and Film*. Cornell: Cornell University Press, USA, 1978.

CIARLINI, A. E. M. *Geração Interativa de Enredos*. D.Sc. Thesis, Departamento de Informática, PUC-Rio, Rio de Janeiro, 1999.

CIARLINI, A. E. M.; CASANOVA, M. A.; FURTADO, A. L.; VELOSO, P. A. S. Modeling interactive storytelling genres as application domains. *Journal of Intelligent Information Systems*, v. 35, p. 347-381, 2010.

CIARLINI, A. E. M.; FURTADO, A. L. Understanding and Simulating Narratives in the Context of Information Systems. *In: ER2002 - 21st International Conference on Conceptual Modeling*, Tampere, Finland, 2002.

CIARLINI, A. E. M.; POZZER, C. T.; FURTADO, A. L.; FEIJÓ, B. A Logic-Based Tool for Interactive Generation and Dramatization of Stories. *In: Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2005*), 2005, Valencia. pp. 133-140, June 2005.

CIMATTI, A.; PISTORE, M.; ROVERI, M.; TRAVERSO, P. Weak, Strong, and Strong Cyclic Planning via Symbolic Model Checking. *Artificial Intelligence*, v.147 (1-2), pp. 35-84, July 2003.

COHEN P.; LEVESQUE, H. Intention is choice with commitment. *Artificial Intelligence*, 42, 3, pp. 213–261, 1990.

COLEMAN, A. *A Dictionary of Psychology (3 ed.)*. Oxford University Press, 2008.

COSTA, P. T., JR.; MCCRAE, R. R. *Revised NEO Personality Inventory (NEO-PI-R) and NEO Five-Factor Inventory (NEO-FFI) professional manual*. Odessa, FL: Psychological Assessment Resources, 1992.

CRAWFORD, C. Assumptions underlying the Erasmatron storytelling system. *In: Working Notes of the 1999 AAAI Spring Symposium on Narrative Intelligence*. AAAI Press, 1999.

CRAWFORD, C. *Chris Crawford on interactive storytelling*. New Riders, 2005.

CRAWFORD, C. *Chris Crawford on Interactive Storytelling (2nd Edition)*. Pearson Education. Kindle Edition, 2012.

CREVIER, D. AI: The Tumultuous Search for Artificial Intelligence. New York, NY: BasicBooks, 1993.

DAVIDSON, M. A phenomenological evaluation: using storytelling as a primary teaching method. *Nurse Education and Practice* 4 (3): 184–189, 2004.

DIGMAN, J. M.; TAKEMOTO-CHOCK, N. K. Factors in the natural language of personality: Reanalysis and comparison of six major studies. *Multivariate Behavioral Research*, 16, 149-170, 1981.

DÓRIA, T. R.; CIARLINI, A. E. M.; ANDREATTA, A. A Nondeterministic Model for Controlling the Dramatization of Interactive Stories. In: *Proceedings of the ACM MM2008 - 2nd ACM Workshop on Story Representation, Mechanism and Context (SRMC08)*, Vancouver, Canada, 2008.

DURANT, W. *The Life of Greece*. New York, NY: Simon and Schuster, p. 379, 1939.

EKMAN, P.; FRIESEN, W. V. Constants across cultures in the face and emotion. *Journal of Personality and Social Psychology*, 17, pp. 124-129, 1971.

EL-NASR, M. S. A user-centric adaptive story architecture: borrowing from acting theories. *In: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, 2004, Singapore. pp. 109-116, 2004a.

EL-NASR, M. S. An Interactive Narrative Architecture Based on Filmmaking Theory. *International Journal on Games and Simulation* 3(1) 29-36, 2004b.

EL-NASR, M. S. Interaction, Narrative, and Drama Creating an Adaptive Interactive Narrative using Performance Arts Theories. *In: Interaction Studies*, Vol. 8, No. 2, 2007.

EMERSON, E. A. Temporal and Modal Logic. *In: van Leeuwen, J. (Ed.), Handbook of Theoretical Computer Science, Volume B: Formal Models and*

Semantics, Elsevier Science Publishers B. V., Amsterdam, 1990, The Netherlands. pp. 996-1072, 1990.

EROL, K.; HENDLER, J.; NAU, D. S. UMCP: A sound and complete procedure for hierarchical task-network planning. In: *Proceedings of the International Conference on AI Planning Systems (AIPS)*. 249-254, 1994.

FAIRHEAD, H. *Kinect's AI breakthrough explained*, 2011. Available at: <http://www.i-programmer.info/news/105-artificial-intelligence/2176-kinects-ai-breakthrough-explained.html>. Retrieved 24 March 2015.

FÉNELON, F. *Telemachus*. Riley, P. (ed. and trans.). Cambridge University Press, 1994.

FERREIRA, P. A. *Modelo de planejamento temporal não determinístico considerando propriedades dramáticas em mudança contínua para storytelling interativo*. M.Sc. Thesis, Departamento de Informática Aplicada, Universidade Federal do Estado do Rio de Janeiro, Brazil, 2013.

FERREIRA, P. A.; GOTTIN, V. M.; CIARLINI, A. E. M.; ARAUJO, E. T.; FURTADO, A. L.; FEIJÓ, B.; SILVA, F. A. G.; POZZER, C. T. A Nondeterministic Temporal Planning Model for Generating Narratives with Continuous Change in Interactive Storytelling. *In: AIIDE Proceedings*, Boston, 2013.

FIEDLER, F. E., MITCHELL, T., & TRIANDIS, H. C. The culture assimilator: An approach to cross-cultural training. *Journal of Applied Psychology*, 55, 95-102, 1971.

FISKE, D. W. Consistency of the factorial structures of personality ratings from different sources. *Journal of Abnormal and Social Psychology*, 44, 329-344, 1949.

FODOR, J. *The modularity of mind*. MIT Press, Cambridge, MA, 1983.

FURTADO, A. L. *Mitos e Lendas: Heróis do Ocidente e do Oriente*. 1. ed. Rio de Janeiro: Nova Era, 2006.

FURTADO, A. L. *Storytelling variants: the case of Little Red Riding Hood*. Monografias em Ciência da Computação Series (MCC 01/15), PUC-Rio, Brazil, 2015.

GARDNER, H. *Frames of mind: The theory of multiple intelligences*. New York: Basic Books, 1983.

GARDNER, H. *Intelligence reframed: Multiple intelligences for the 21st century*. New York: Basic Books, 1999.

GÉRÔME, J.-L. Pygmalion and Galatea. Oil on canvas, 88.9 X 68.6 cm. Whitford and Hughes, London, UK / Bridgeman Art Library, 1890.

GHALLAB, M.; NAU, D.; TRAVERSO, P. *Automated Planning: Theory and Practice*. 1ed. Amsterdam: Morgan Kaufmann Publishers, 2004.

GLASSNER, A. *Interactive Storytelling: Techniques for 21st Century Fiction*. AK Peters, 2004.

GODDKIND, N. How the video game industry became bigger than movies and music. *Yahoo! Finance*, 2014. Available at: <http://finance.yahoo.com/blogs/daily-ticker/how-the-video-game-industry-became-bigger-than-movies-and-music-171225174.html>. Retrieved 14 March 2015.

GOLDBERG, L. R. Language and individual differences: The search for universals in personality lexicons. *In:* L. Wheeler (Ed.), *Review of personality and social psychology*, (Vol. 2, pp. 141-165). Beverly Hills, CA: Sage, 1981.

GOLDBERG, L. R. *The Development of Markers for the Big-Five Factor Structure*. Psychological Assessment, v4, n1 pp. 26-42, 1992.

GOLEMAN, D. *Emotional Intelligence*, New York, NY, England: Bantam Books, Inc., 1995.

GOSLING, S. D.; RENTFROW, P. J.; SWANN, W. B. A very brief measure of the Big-Five personality domains. *Journal of Research in Personality*, 37, 504–528, 2003.

GOTTIN, V. M. *Verificação abstrata de propriedades dramáticas contínuas em eventos não determinísticos*. M.Sc. Thesis. Departamento de Informática, Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2013.

GRASBON, D.; BRAUN, N. A morphological approach to interactive storytelling. *In: Proc. CAST01, Living in Mixed Realities. Special issue of Netzspannung.org/journal, the Magazine for Media Production and Inter-media Research, Sankt Augustin, Germany*. 337-340, 2001.

HAMMERTON, B. Heart vs. Head: Who has the upper hand? *The Garland Messenger*, 2014. Available at: <http://thegarlandmessenger.com/heart-vs-head-who-has-the-upper-hand/>. Retrieved 26 March 2015.

HEAVY RAIN. Quantic Dream, Sony Computer Entertainment, 2010.

HENDRIX, G. *Choose Your Own Adventure*, 2011. Available at <http://www.slate.com/articles/arts/culturebox/2011/02/choose_your_own_adventure.single.html#pagebreak_anchor_2>. Retrieved 30 March 2015.

HIGHAM, N. J. *King Arthur, Myth-Making and History*. London: Routledge, 2002.

HOPKINS, T. J. *The Hindu Religious Tradition*. Wadsworth Publishing, 1971.

HOWARD, M. The Fable, Folktale, Myth, Legend: Differences and Examples, 2013. Available at: <http://study.com/academy/lesson/the-fable-folktale-myth-legend-differences-and-examples.html>. Retrieved 28 March 2015.

HOWE, J. *Artificial Intelligence at Edinburgh University: a Perspective*. The University of Edinburgh School of Informatics, 1994. Available at: <http://www.inf.ed.ac.uk/about/AIhistory.html>. Retrieved 24 March 2015.

HUDLICKA, M. Affective computing for game design. *In: Proceedings of the 4th North American Conference on Intelligent Games and Simulation (GAMEON-NA)*, McGill University, Montreal, Canada, 5-12, 2008.

HUIZINGA, J. *Homo Ludens*. Translation: João Paulo Monteiro. Perspectiva, Brazil, 2001.

IBEJI, M. *Robin Hood and his Historical Context*, 2011. Available at: <http://www.bbc.co.uk/history/british/middle_ages/robin_01.shtml>. Retrieved 28 March 2015.

JOHN, O. P.; ANGLEITNER, A.; OSTENDORF, F. The lexical approach to personality: A historical review of trait taxonomic research. *European Journal of Personality*, 2, 171-203, 1988.

JOHN, O. P.; SRIVASTAVA, S. The Big-Five trait taxonomy: History, measurement, and theoretical perspectives. *In:* L. A. Pervin & O. P. John (Eds.), *Handbook of personality: Theory and research* (Vol. 2, pp. 102–138). New York: Guilford Press, 1999.

JUNG, C. G. Psychological Types. *Collected Works of C.G. Jung, Volume 6*. Princeton University Press, 1971.

KARLSSON, B.; GUERRA, F. W.; FURTADO, A. L. *On the Craft of Interactive Stories*. Monografias em Ciência da Computação Series (MCC 36/09), ISSN 0103-9741, Department of Informatics/PUC-Rio, Rio de Janeiro, Brazil, 2009.

KARLSSON, B. *A Model and an Interactive System for Plot Composition and Adaptation, based on Plan Recognition and Plan Generation*. D.Sc. Thesis, PUC-Rio, Brazil, 2010.

KLAGES, L. *The science of character* (Translated 1932). London: Allen and Unwin, 1926.

KRAFT, S. He Chose His Own Adventure. *The Day*, 1981. Available at: <https://news.google.com/newspapers?id=_nUfAAAAIBAJ&sjid=XXUFAAAAIBAJ&pg=1663,2191360&dq=choose-your-own-adventure&hl=en>. Retrieved 30 March 2015.

KURZWEIL, R. *The Singularity is Near*. Penguin Books, 2005.

KUTER, U. *Planning under Uncertainty: Going Forward*. Invited Talk at the Seminars on Artificial Intelligence Planning, National ICT Australia, Sydney, Australia, 2005.

KUTER, U.; NAU, D. S. Forward-Chaining Planning in Nondeterministic Domains. *In: Proceedings of AAAI-2004*, 2004

KUTER, U.; NAU, D.; PISTORE, M.; TRAVERSO, P. Task Decomposition on Abstract States for Planning under Nondeterminism. *Artificial Intelligence. Special issue on Advances in Automated Planning*. 173:669--695, 2009.

LARSEN, S.; LARSEN, R. *A Fire in the Mind: The Life of Joseph Campbell*. Rochester, Vermont: Inner Traditions, 2002.

LAWS, R. *Robin's Laws of Good Game Mastering*. Steve Jackson Games, 2001.

LEACH, M. (Ed.). *Standard Dictionary of Folklore, Mythology, and Legend*. New York: Funk & Wagnalls, USA, 1972.

LEBOWITZ, M. Creating characters in a story-telling universe. *Poetics*, 13 (3), 1984.

LEBOWITZ, M. Story-Telling as planning and learning. *Poetics*, 14 (6), 1985.

LIMA, E. S. *Video-Based Interactive Storytelling*. D.Sc. Thesis, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brazil, 2014.

LIMA, E. S.; FEIJÓ, B.; BARBOSA, S. D. J.; SILVA, F. A. G.; FURTADO, A. L.; POZZER, C. T.; CIARLINI, A. E. M. Multimodal, Multi-User and Adaptive Interaction for Interactive Storytelling Applications. *In: Proceedings do X Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames 2011)*, Salvador, p. 206-214, 2011.

LIMA, E. S.; FEIJÓ, B.; FURTADO, A. L.; BARBOSA, S.; POZZER, C. T.; CIARLINI, A. E. M. Non-Branching Interactive Comics. *In: Proceedings of the 10th International Conference on Advances in Computer Entertainment Technology (ACE 2013)*, Enschede, v. 8253, p. 230-245, 2013.

LOUCHART, S.; AYLETT R.; ENZ S.; DIAS, J. Understanding emotions in drama, a step towards interactive narratives. *In:* Tim Kovacs and James A. R. Marshall (Eds.). *Proceedings of AISB'06: Adaptation in Artificial and Biological Systems*, 38-44, April 2006.

LOEWENSTEIN, G.; LERNER, J. S. The role of affect in decision making. *In*: *Handbook of Affective Sciences*. Davidson, R.J.; Scherer, K.R.; Goldsmith, H.H. (Eds.). Oxford University Press, pp. 619-642, 2003.

LUGER, G.; STUBBLEFIELD, W. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving (5th ed.)*. Benjamin/Cummings. ISBN 0-8053-4780-1, 2004.

MATEAS, M.; STERN, A. Façade: An Experiment in Building a Fully-Realized Interactive Drama. *In: Game Developers Conference*, p. 4-8, 2003.

MATEAS, M.; STERN, A. Structuring content in the Façade interactive drama architecture. *In: Proc. Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2005)*, 2005.

MCCORDUCK, P. *Machines Who Think (2nd ed.)*, Natick, MA: A. K. Peters, Ltd., 2004.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115-137, 1943.

MEEHAN, J. R. TALE-SPIN, An Interactive Program that Writes Stories. *In: Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 1977.

MEEHAN, J. R. TALE-SPIN. *In: Inside Computer Understanding: Five Programs Plus Miniatures*, Schank, R., Riesbeck, C. (Eds.), Lawrence Erlbaum Associates, Hillsdale, USA, pp. 197-226, 1981.

MEULEN, R. van der. *Gartner Says Worldwide Video Game Market to Total $93 Billion in 2013*, 2013. Available at: <http://www.gartner.com/newsroom/id/2614915>. Retrieved 14 March 2015.

MINSKY, M. *The Emotion Machine*. Simon & Schuster, 2006.

MORAVEC, H. *Mind Children*. Harvard University Press, 1988.

MYERS, I. B.; MCCAULLEY M. H.; QUENK, N. L.; HAMMER, A. L. *MBTI Manual (A guide to the development and use of the Myers Briggs type indicator)*. Consulting Psychologists Press; 3rd edition, 1998.

NAU, D. S. Current trends in automated planning. *AI Magazine* 28(4):43–58, 2007.

NEEDHAM, J. *Science and Civilization in China: Volume 2*. Caves Books Ltd., 1986.

NICK, M. Al Jazari: The Ingenious 13th Century Muslin Mechanic. *Al Shindagah*, 2005. Available at: <http://www.alshindagah.com/marapr2005/jaziri.html>. Retrieved 24 March 2015.

NILSSON, N. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann, 1998.

NORMAN, W. T. Toward an adequate taxonomy of personality attributes: Replicated factor structure in peer nomination personality ratings. *Journal of Abnormal and Social Psychology*, 66, 574-583, 1963.

NRC (United States National Research Council). Developments in Artificial Intelligence. *Funding a Revolution: Government Support for Computing Research*. National Academy Press, 1999.

OSGOOD, C. E.; TANNENBAUM, P. H.; SUCI, G. J. *The Measurement of Meaning*. Urbana: University of Illinois Press, 1957.

PAYNE, W. L. A study of emotion: developing emotional intelligence; self integration; relating to fear, pain and desire. Dissertation Abstracts International, 47, p. 203A (University microfilms No. AAC 8605928), 1985.

PEREIRA, S. *Planejamento sob Incerteza para Metas de Alcançabilidade Estendidas*. D.Sc. Thesis, IME-USP, 2007.

PICARD, R. W. *Affective Computing*. M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 321, 1995.

PICARD, R. W. *Affective Computing*. MIT Press, Cambridge, 1997.

PISTORE, M.; TRAVERSO, P. Planning as Model Checking for Extended Goals in Non-Deterministic Domains. *In: 17th International Joint Conference on Artificial Intelligence*, pp. 479-484, Washington, 2001.

PLUTCHIK, R. *The emotions: Facts, theories, and a new model*. Random House: New York, 1962.

PLUTCHIK, R. *Emotion: Theory, research, and experience: Vol. 1. Theories of emotion 1*. New York: Academic, 1980.

POLTI, G. *Thirty-Six Dramatic Situations*. Whitefish: Kessinger Publishing, USA, 1945.

POOLE, D.; MACKWORTH, A.; GOEBEL, R. *Computational Intelligence: A Logical Approach*. New York: Oxford University Press. ISBN 0-19-510270-3, 1998.

PROPP, V. *Morphology of the Folktale***,** 2 ed. Austin, Texas: University of Texas Press, 2003.

RICH, E. User modeling via stereotypes. *Cognitive Science* 3, pp. 329-354, 1979.

RIEDL, M. *Narrative Planning: Balancing Plot and Character*. PhD Thesis, North Carolina State University, 2004.

RIEDL, M.; YOUNG, R. M. From Linear Story Generation to Branching Story Graphs. *IEEE Computer Graphics and Applications*, v. 26, n. 3, pp. 23-31, doi:10.1109/MCG.2006.56, May/June 2006.

RODRIGUES, P. S. L.; FEIJÓ, B.; VELHO, L. C. P. R. *Um sistema de geração de expressões faciais dinâmicas em animações faciais 3D com processamento de fala*. 161 f. D.Sc. Thesis, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Brazil, 2007.

ROWINSKI, D. *Virtual Personal Assistants & The Future Of Your Smartphone [Infographic]*, 2013. Available at: <http://readwrite.com/2013/01/15/virtual-personal-assistants-the-future-of-your-smartphone-infographic>. Retrieved 24 March 2015.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach (2nd ed.)*, Upper Saddle River, New Jersey: Prentice Hall, 2003.

SAUCIER, G. Mini-markers: A brief version of Goldberg's unipolar Big-Five markers. *Journal of Personality Assessment*, 63, 506–516, 1994.

SAUCIER, G.; GOLDBERG, L. R. The language of personality: Lexical perspectives on the five-factor model. *In:* J. S. Wiggins (Ed.), *The five-factor model of personality: Theoretical perspectives*. (pp. 21-50): Guilford Press, New York, NY, US, 1996.

SCHANK, R. C.; ABELSON, R. P. *Knowledge and Memory: The Real Story*. Hillsdale, NJ: Lawrence Erlbaum Associates. pp. 1–85, 1995.

SELDEN, R., WIDDOWSON, P., BROOKER, P. *A Reader's Guide to Contemporary Literary Theory*, Fifth Edition. Harlow: p. 76, 2005.

SHELLEY, M. *Frankenstein*. Harmondsworth: Penguin Classics, 1816.

SHKLOVSKY, V. The Novel as Parody: Sterne's Tristram Shandy. *In: Theory of Prose*, p. 147-170. Normal: Dalkey Archive Press, USA, 1991.

SHOSTAK, S. *Using Radio in the Search for Extraterrestrial Intelligence*, US Congress, 2014. Available at <http://science.house.gov/sites/republicans.science.house.gov/files/documents/HHRG-113-SY-WState-SShostak-20140521.pdf>. Retrieved 24 March 2015.

SILVA, F. A. G. *Geração de enredos com planejamento não-determinístico em storytelling para TV interativa.* M.Sc. Thesis, Departamento de Informática Aplicada, Universidade Federal do Estado do Rio de Janeiro, Brazil, 2010.

SILVA, F. A. G.; CIARLINI, A. E. M.; SIQUEIRA, S. W. M. Nondeterministic Planning for Generating Interactive Plots. In: Proceedings of the 12th Ibero-American Conference on Artificial Intelligence, Bahía Blanca, Argentina. *Lecture Notes in Computer Science*. 1ed.: Springer Berlin Heidelberg, v. 6433, p. 133-143, 2010.

SILVA, F. A. G.; CIARLINI, A. E. M.; SIQUEIRA, S. W. M. A Planning Algorithm for Incorporating Attempts and Nondeterminism into Interactive

Stories. *In: 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, Boca Raton, Florida, EUA. ICTAI 2011. p. 96-101, 2011.

SILVA, F. A. G.; FURTADO, A. L. *Information gathering events in story plots*. Monografias em Ciência da Computação Series (MCC 07/11), PUC-Rio, Brazil, 2011.

SILVA, F. A. G.; FURTADO, A. L.; CIARLINI, A. E. M.; POZZER, C. T.; FEIJÓ, B.; LIMA, E. S. Information-gathering Events in Story Plots. In: Proceedings of the 11th International Conference on Entertainment Computing, Bremen, Germany. *Lecture Notes in Computer Science*. 1ed.: Springer Berlin Heidelberg, 2012, v. 7522, p. 30-44, 2012.

SPIERLING, U.; BRAUN, N.; IURGEL, I.; GRASBON, D. Setting the scene: playing digital director in interactive storytelling and creation. *Computers & Graphics*, v. 26, n.1, pp. 31-44, 2002.

SUOMINEN, J. Atorox, Finnish Fictional Robot with a Changing Personality in the Late 1940s. *In:* Ferro, D. L., Swedin, E. G. (Eds.) *Science Fiction and Computing: Essays on Interlinked Domains*, kindle Edition, NC, USA and London, UK, pp. 68–82, 2011.

TAO, J.; TAN T. Affective Computing: A Review. *Affective Computing and Intelligent Interaction*. LNCS 3784. Springer. pp. 981–995, 2005.

THOMPSON, S. *Motif-Index of Folk-Literature*. A Classification of Narrative Elements in Folktales, Ballads, Myths, Fables, Mediaeval Romances, Exempla, Fabliaux, Jest- Books and Local Legends. Rev. & enlarged ed. 6 vols. Bloomington: Indiana University Press, USA, 1989.

THUE, D.; BULITKO, V.; SPETCH, M.; WASYLISHEN E. Interactive Storytelling: A Player Modelling Approach. *In: Proceedings of the Artificial Intelligence and Interactive Digital Entertainment conference (AIIDE)*, Stanford, USA, pp. 43-48, 2007.

TOMASHEVSKY, B. Thematics. *In: Russian Formalist Criticism: Four Essays*, p. 61-95, Lincoln: University Nebraska Press, USA, 1965.

TUPES, E. C.; CHRISTAL, R. C. *Recurrent personality factors based on trait ratings*. Technical Report, USAF, Lackland Air Force Base, TX, 1961.

TURING, A. M. On computable numbers, with an application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society*, Ser. 2--42, 230--265, 1936.

TURING, A. M. Computing machinery and intelligence. *Mind*, 59, 433-560, 1950.

TURNER, S. *MINSTREL: a computer model of creativity and storytelling*. PhD Thesis, Computer Science Department, University of California, 1992.

UTHER, H.-J. The Third Revision of the Aarne-Thompson Tale Type Index (FFC 184), *In: FFNetwork* 20, pp. 11-13, 2000.

UTHER, H.-J. *The Types of International Folktales: A Classification and Bibliography Based on the System of Antti Aarne and Stith Thompson*. Vols 1-3. FF Communications No. 284-86, Helsinki: Academia Scientiarum Fennica, 2004.

VALFRE, G. B. Composição e Monitoração Automáticas e Contínuas de Serviços Web Não-Determinísticos. M.Sc. Thesis. Universidade Federal do Estado do Rio de Janeiro, 2010.

WHITEHEAD, A. N.; RUSSELL, B. *Principia Mathematica*. Cambridge University Press, Cambridge, UK, 1910.

YOUNG, R. An overview of the mimesis architecture: Integrating narrative control into a gaming environment. *In: Working notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, March 2001, Stanford, CA. AAAI Press, pp. 78-81, 2001.

YOUNGBLOOD, G. M.; DIXIT, P. N. Understanding Intelligence in Games using Player Traces and Interactive Player Graphs. *Game Programming Gems* 7 (AI Section). Charles River Media, 2008.