PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 07/05

A Logic-Based Tool for Interactive Generation and Dramatization of Stories

Angelo E. M. Ciarlini
Cesar Tadeu Pozzer
Antonio L. Furtado
Bruno Feijó

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 07/05 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena February,2005

A Logic-Based Tool for Interactive Generation and Dramatization of Stories *

Angelo E. M. Ciarlini[1] Cesar Tadeu Pozzer Antonio L. Furtado

Bruno Feijó

[1]UniRio – Departamento de Informática Aplicada

angelo.ciarlini@uniriotec.br pozzer@tecgraf.puc-rio.br furtado@inf.puc-rio.br

bruno@inf.puc-rio.br

Abstract: A key issue in interactive storytelling is how to generate stories which are, at the same time, interesting and coherent. On the one hand, it is desirable to provide means for the user to intervene in the story. On the other hand, it is necessary to guarantee that user intervention will not introduce events that violate the rules of the intended genre. This paper describes LOGTELL, a tool we have been developing for interactively generating and dramatizing stories. We focus on the specification of a formal logic model for events and characters' behaviour. Based on the model, the user can interact with the tool at various levels, obtaining a variety of stories agreeable to individual tastes, within the imposed coherence requirements. The system alternates stages of goal inference, planning, user intervention and 3D visualization. Our experiments have shown that the system can be used not only for entertainment but also to help in the creation and adaptation of stories in conformity with a specified genre.

Keywords: Narratives, Storytelling, Planning, Temporal Logic, Simulation

Resumo: Um aspecto vital ao se considerar narração interativa é como gerar estórias que sejam, ao mesmo tempo, interessantes e coerentes. Por um lado, é desejável prover meios para que o usuário possa intervir no processo. Mas, por outro lado, é preciso garantir que sua intervenção não venha introduzir eventos que violem as regras do gênero visado. Este trabalho descreve a ferramenta LOGTELL, que estamos desenvolvendo para gerar e dramatizar estórias. É focalizada a especificação de um modelo formal lógico para eventos e para o comportamento dos personagens. Baseado nesse modelo, o usuário pode interagir com a ferramenta em vários níveis, obtendo uma considerável variedade de estórias capazes de agradar ao gosto de cada um, dentro dos limites impostos pela coerência. O sistema alterna fases de inferência de objetivos, planejamento, intervenção do usuário e visualização 3D. Nossas experiências têm mostrado que o sistema pode ser usado, não apenas para entretenimento, mas também para apoiar a criação e adaptação de estórias, em conformidade com o gênero especificado.

Palavras-chave: Narrativas, Narração de Estórias, Planejamento, Lógica Temporal, Simulação.

* This work has been sponsored by the Ministério de Ciência e Tecnologia da Presidência da República Federativa do Brasil.

In charge for publications:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br

1. INTRODUCTION

In recent years, the convergence of games and filmmaking has been seen as an opportunity to create storytelling systems in which authors, audience, and virtual agents engage in a collaborative experience [15]. The resulting systems may be useful for many different purposes such as writing literary texts, education and training, modelling and decision making and, of course, entertainment. Different approaches have been proposed, using techniques and concepts from many areas such as Computer Graphics, Artificial Intelligence, Cognitive Science, Literature and Psychology. The suitability of each approach depends on the goal of each application.

A first decision to be made before implementing a storytelling system is whether it should be able to actually *create* stories or only to enable the user to *tell* different stories based on previously computed sequences of actions. In the first case, the opportunities of interaction and the variety of different stories tend to be greater, but the coherence of the chaining of actions is more difficult to be kept.

A second important point corresponds to the focus of the story models. The focus can be either on characters or on plots. In a *character-based* approach, the storyline usually results from the real-time interaction among virtual autonomous agents. The main advantage of a character-based model is the ability of anytime user intervention, which means that the user may interfere with the ongoing action of any character in the story, thereby altering the plot as it unfolds. Although powerful in terms of interaction, such an extreme interference level may lead the plot to unexpected situations or miss essential predefined events. Additionally, there is no guarantee that narratives emerging from the interaction of parameterized autonomous agents will be complex enough to create an interesting drama. By contrast, in *plot-based* models, characters should follow rigid rules specified by a plot. A fundamental inspiration for plot-based approaches has been the seminal work of Vladimir Propp in the field of literature [13], at the beginning of the twentieth century. Propp observed that significant actions within a narrative of a certain genre (in his case, Fairy Tales) can be associated with functions and that functions occur in certain typical sequences. In a pure plot-based approach, user intervention might be more limited, but it is usually easier to guarantee coherence and a minimum of dramatic power.

A third decision is whether stories should be told using a first- or a third-person *viewpoint* (cf. the notion of *focalization* in narratology studies [1]). First-person tends to be more suitable for approaches closer to games and third-person for those inspired by writing and filmmaking.

Finally, it is necessary to choose between a *reactive* and a *deliberative* behaviour for the characters. In the first case, efficiency is the main advantage, but modelling an intelligent behaviour is more complicated and the alternatives for the agents tend to be more limited. In the second case, planning and reasoning techniques are usually applied to simulate an intelligent behaviour, but performance is often affected, which might be a problem, especially if the story generation occurs at real-time.

LOGTELL is based on modelling and simulation. The idea behind LOGTELL is to capture the logics of a genre through a temporal logic model and then verify what kind of stories can be generated by simulation combined with user intervention. In this way, we focus not simply on different ways of telling stories but on the dynamic creation of plots. The model is composed of typical events and goal-inference rules. Plots are generated by multiple cycles of goal-inference, planning and user intervention.

Specifically, we try to conciliate both plot-based and character-based modelling. On the one hand, we borrowed from Propp's ideas, but tried to extend his rather informal notion of function. In our treatment, typical events are described by parameterized operations with pre-conditions and post-conditions, so that planning algorithms can be used for simulation. On the other hand, character-based modelling is added under the form of goal-inference rules for each kind of character. The rules declaratively specify how situations can bring about new goals for the characters.

Instead of creating an immersive experience in which the user takes part in the story as one of the characters, we try to explore the possibilities of generating a large variety of coherent stories. For this reason, our stories are told with a third-person viewpoint. User intervention is always indirect. During the simulation, the user can intervene either passively, just letting the partially-generated plots that seem interesting to be continued, or, in a more active way, trying to force the occurrence of events and situations. These are rejected by the system whenever there is no way to change the story to accommodate the intervention.

Plot dramatization can be activated for exhibiting the final as also the partial plots. During the dramatization, characters are represented by actors in a 3D-world. During the performance of an event, low-level planning is used to detail the tasks involved in each event. In order to integrate dramatization and plot generation, we decided to implement our own graphical engine, so that we could guarantee the compatibility between the logical model of our plots and the corresponding graphical dramatization.

The next section describes related work in the area of storytelling. Section 3 presents LOGTELL's overall architecture. Section 4 describes the main features of the Interactive Plot Generator (IPG), which is the kernel of the system. Section 5 illustrates how the user can interact with LOGTELL to generate stories. Section 6 shows how generated plots are dramatized. Section 7 illustrates the use of the tool with an example. Section 8 contains our concluding remarks.

2. RELATED WORK

The approach adopted in the DEFACTO project [17] uses successive evaluations of rules to control the generation of an interactive story where the user is the protagonist. The interaction among characters' goals is explicitly represented and an Aristotelian conception of plot is used to lead the story to a climax and then resolve it. The chaining of events, however, is not explained by pre- and post-conditions, making the control of what can and what cannot occur more complex. Additionally, it does not allow the use of planning algorithms

to explore sequences of events, instead of single events, for the achievement of goals. The need of user intervention seems to be high if one wishes to generate a complete plot. Goals are inferred by means of rules analyzing the current situation, but the choice of actions to achieve goals seems to be more reactive than deliberative. In LOGTELL, the interaction among characters' goals is embedded in the goal-inference rules. In our plot generation phase (in contrast with our dramatization phase) we are not concerned about quickly choosing a single coherent course of action, because plot generation and dramatization occur at different times. Our objective is to explore many alternatives and select the most interesting ones.

The approach described in [2] adopts a character-based model to make user interventions at any possible time. Characters are autonomous agents, executing plans to achieve their goals, and, from their interactions, it is expected that a narrative will eventually emerge. Users are spectators but can "physically" interact with the context and even advise characters, affecting their decisions and the resulting stories. In order to decide, at real-time, the actions to be performed, characters consult a Hierarchical Task Network (HTN), corresponding to pre-compiled plans. In this way, the system does not have to pay the price of using problem-solving planners while presenting a 3D animation. It might demand more effort to model the behaviour of the characters, but it makes sense if one does not consider maximizing the alternatives as a requirement. The main doubt about pure character-based approaches is to what extent dramatic and engaging narratives may actually result. The task seems to be easier with genres like sitcoms, wherein the climax of a story is not so clearly distinguishable.

The use of Propp's ideas in pure plot-based approaches leads to systems more concerned with the guidance of interactive stories than with their generation [16,8]. For each "Proppian" function within a story of a certain genre, such systems present alternatives to be chosen by the users. Still, we claim that to obtain an effective method to generate stories, it is necessary to extend Propp's ideas, adding semantics to the functions (and to their specializations), so that preconditions, effects and goals can be fully expressed.

Reference [11] presents the Teatrix environment, where Propp's functions are used to model synthetic characters that interact with other characters, directed by children, in a virtual world. Each child directs one character and the synthetic characters are autonomous. All characters have a role in the story, specifying the functions in which they can take part. Synthetic characters have goals that change according to the situation. They plan and try to execute actions (i.e. functions) according to their roles. The approach seems interesting for education, but the control of the consistency of actions and goals and the generation of dramatic situations are not guaranteed. Additionally, the use of predefined plans in the planning process can enhance the performance, but might limit the amount of different stories that can be generated.

The interactive drama FAÇADE [10] is an effort to build an interactive system that integrates characteristics of both plot-based and character-based approaches. A drama manager is responsible for maintaining the story state.

Characters have autonomy most of the time, but their goals and their behaviour can be changed by the drama manager, in order to move the plot forward. The interactive story has the user as the protagonist. The drama manager automatically selects *scenes* to be played. *Scenes* are composed of *beats*, which define the granularity of the interaction between characters and plots. The user can directly interfere in the execution of a beat, determining how the rest of the scene will be played. The approach clearly separates higher-level goals, important for the story, from lower-level goals, more specific of the autonomous behaviour of the characters. Such separation can also be found in LOGTELL. The generation phase deals only with higher-level goals, which are essential for the creation of plots. Lower-level goals are assigned to actors when they have to dramatize an event. The main differences between LOGTELL and FAÇADE result from the objectives of each system. In FAÇADE, the focus is on letting the user experience a story from a first-person perspective. As a consequence, the interaction occurs at real-time, at the level of the beats. In LOGTELL, we focus on the generation of a maximum of different and coherent stories with a third-person viewpoint. The interaction basically occurs during the generation phase. The user is not allowed to interfere in the dramatization phase.

The system Erasmatron [6] is intended to support the authoring process of interactive stories. It tries to balance plot-based and character-based approaches by using the notions of verbs and sentences. Actions are represented by verbs with roles assigned to characters to form sentences. Such a proposal is close to the way we extended Propp's functions in LOGTELL. Functions are implemented as logical operations, with parameters, pre- and post-conditions.

The use of planning in [14] to create plots has many similarities with the decisions made while implementing LOGTELL. In both approaches, a non-linear, least-commitment planner is used to create plots, conciliating actions of many different characters. The main difference is that LOGTELL does not assume the existence of one goal for the story as a whole. Instead, at the beginning of the story and after each planning phase, we use goal-inference rules (defined in a temporal modal logic) to consider new goals induced, for the various characters, by situations arising from the part of the plot so far generated. On the other hand, plans generated according to [14] incorporate information explaining the intention of the actions, which can be useful to help in the dramatization of a plot, in particular to choose a convincing order of events. In LOGTELL, it is up to the user to choose a compatible total order of events to be dramatized.

3. LOGTELL'S ARCHITECTURE

LOGTELL comprises a number of distinct modules to provide support for generation, interaction (management) and visualization of interactive plots, as shown in Figure 1. The arrows represent the dataflow. The general architecture can be seen as a pipeline where data is transformed from morphological functions into real-time 3D animations dramatized by virtual actors handled by a graphical engine. Consequently, each module has specific input and output data. Most of the modules are fully implemented.

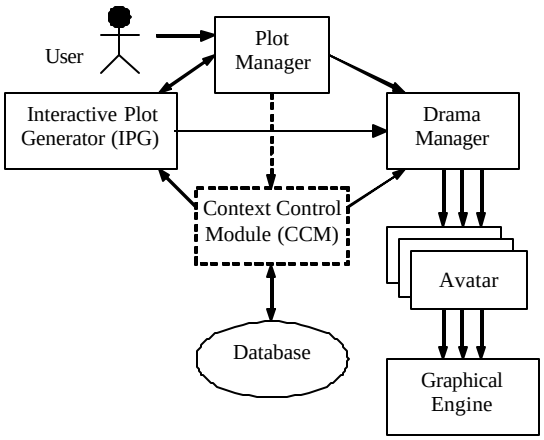

Figure 1: LOGTELL's architecture.

The interface of the system is the Plot Manager. The generation of plots by the Interactive Plot Generator (IPG) is started by the Plot Manager, which receives the partial plots generated so far and allows the user to intervene in the generation process. In order to visualize the dramatization of a plot (final or partial), the user chooses a compatible total order of events, because IPG generates only a partial order, and asks the Plot Manager to activate the Drama Manager.

The Interactive Plot Generator performs simulations using the context specified by the user. Constraints on partial plots, informed by users to the Plot Manager at interaction phases, are taken into consideration when the simulation process is continued.

The Drama Manager is responsible for controlling the dramatization of the plot. In order to do that, it controls actors for each character in a 3D environment running on our game engine. During the dramatization, the Drama Manager consults IPG to keep the coherence between logical and graphical representations of the plot.

For the time being, the context of the stories to be generated and told is directly accessed by the modules and there is a certain replication of data. IPG uses files directly specifying the logical context in Prolog and the Drama Manager uses its own graphical and logical data. In order to eliminate compatibility problems, we are currently implementing the Context Control Module (CCM) to store all data in a single database. CCM will control the access to the data and format the data items to be used by the other modules. We are also extending our interface to help the user to specify the context via the Plot Manager.

4. PLOT GENERATION

IPG [3] semi-automatically generates plots of narratives of a certain genre. Narratives could be both of literary genres and of more mundane ones, such as the context of an information system [7]. In the latter case, IPG is used to simulate systems and help in decision-making. In its use for entertainment, the focus is on checking the logical coherence of a genre and its characters and exploring the variety of stories that can be generated.

The context for the creation of stories comprises the following items:

- an initial configuration, introducing the characters and their initial situation, as well as the description of the scenarios and other static features needed for the generation of stories;
- a set of logical rules, to infer goals to be pursued by each character, as certain situations arise in the course of plots; and
- a library of operations in which characters can take part (typical to the chosen genre).

The initial configuration is specified by a database state to be modified by the execution of operations. Examples of possible *facts* stored in this database in a fairy tale context, using a Prolog notation, are listed below:

- character(villain, dragon): “The dragon is a villain.”
- villain_state(dragon, strength, 0.5): “The initial dragon’s strength level is 0.5”
- hero_state(knight, princess,0.2): “The initial affection level between the knight and the princess is 0.2”.

Some facts from the initial configuration will be modified by *events* (denoted by executions of operations) and others will remain till the end of the plot. The library of operations specifies the kinds of events that may occur in the narratives, designed in anticipation of the character’s goals. For each operation, the following data are supplied:

- a list of arguments, indicating the roles played by the characters involved in the event, and other features of the corresponding actions;
- a list of pre-conditions, specifying facts that should hold prior to the execution of the operation;
- a list of post-conditions, specifying facts that hold immediately after the execution of the operation;
- its representation, specifying details about the exhibition of an event effected by the operation.

An example of an operation in the fairy tale context is “the kidnapping of the victim”. Usual pre-conditions are that “the victim should presently be fragile” and that “both the victim and the villain should be present at the victim’s land”. Post-conditions can specify that “the victim will be captured by the villain” and “both the villain and the victim will be at the villain’s land”. The representation of events based on this operation would involve the specification of smaller-grain actions, such as: the villain getting closer to the victim, grasping the victim and taking him / her to the villain’s land.

During the generation phase, plots are represented by partially-ordered sets of events. Partial rather than total ordering is a consequence of the use of non-linear planning during the simulation, establishing temporal constraints only when necessary, which makes the conciliation of goals easier. As a consequence, the truth of a fact at a certain time might depend on the final total order that will be chosen later. For instance, suppose there are two events without a predefined order between them: “the knight gets stronger” and “the knight

fights the dragon". Depending on the order, the knight has different strength levels at the time he fights the dragon.

For each class of characters, there are *goal-inference rules*, specifying, in a temporal modal logic formalism [4], the goals that the characters of the class will have when certain situations occur during a narrative. The rules use the following meta-predicates to speak about the occurrence of an event or the truth value of a literal (a fact or a negation of a fact) at certain times:

- *h(T,LITERAL)*: *LITERAL* is necessarily true at time *T*;
- *p(T,LITERAL)*: *LITERAL* is possibly true at time *T*; and
- *e(T,LITERAL)*: *LITERAL* is established at time *T*; and
- *o(T,EVENT)*: *EVENT* occurred at time *T*.

In order to express constraints relating variables, there are two additional meta-predicates:

- *h(CONSTRAINT)*: *CONSTRAINT* is necessarily true; and
- *p(CONSTRAINT)*: *CONSTRAINT* is possibly true.

An example of goal-inference rule, appropriate to the context of fairy tales, is: "when the victim becomes fragile, the villain will regard that as an opportunity and will have the goal of kidnapping the victim". Another possible rule is that "when the victim is kidnapped, the hero will feel motivated to free the victim". This last rule, is represented in our logic as follows:

$\forall$(VIC,T1,VIL) e(T1,kidnapped(VIC,VIL) $\rightarrow$
$\exists$T2 h(T2,not(kidnapped(VIC,VIL))) $\wedge$ h(T2>T1)

It is important to notice that the rules do not determine the specific reaction of a character. They only indicate goals to be pursued somehow. The events that will eventually achieve the goals are determined by the planning algorithm.

The generation of a plot starts by inferring goals of characters from the initial configuration. Given this initial input, the system uses a planner that inserts events in the plot in order to allow the characters to try to fulfill their goals. When the planner detects that all goals have been either achieved or abandoned, the first stage of the process is finished. The partial plot then generated is presented to the user by means of the Plot Manager and can optionally be dramatized. If the user does not like the partial plot, IPG can be asked to generate another alternative. If the user accepts the plot generated so far, the process continues by inferring new goals from the situations generated in the first stage. If new goals are inferred, the planner is activated again to fulfill them. The process alternates goal-inference, planning and user interference until the moment the user decides to stop or no new goal is inferred.

Notice that, in this process, we mix forward and backward reasoning. In the goal-inference phase, we adopt forward reasoning, so that situations in the past generate goals to be fulfilled in the future. In the planning phase, an event inserted in the plot for the achievement of a goal might have unsatisfied pre-conditions, to be handled through backward reasoning. Also, to establish them before the event, the planner might insert previous events with further unfulfilled pre-conditions, and so on recursively.

The user can also force the occurrence of events at certain times. For instance, in a fairy tale context, the user could well insert “the wedding of the knight with the princess”. It is also possible to specify that some situations should be true at certain times along the narrative, leaving to the system the job of planning the events that bring about such situations. It should be possible to say, for instance, that “the knight will be weaker than the dragon at a certain time”. This kind of intervention is allowed both at the beginning of the process and at the pauses occurring between two simulation cycles The planner tries to conciliate both inferred goals and user specified events and situations.

Our planning tool is a non-linear planner implemented in Prolog, adapted from [19] with extensions. The use of a non-linear planner, as suggested before, seems more suitable because it uses a least-commitment strategy. Constraints (including the order of events) are established only when necessary, making easier the conciliation of various goals. Features to permit the abandonment of goals were included, and also constraint programming techniques for dealing with numerical pre-conditions.

Our plots are not restricted to incorporating only successful plans. In trying to provide adequate means for handling negative interactions happening along a plot, we realized that the solution of conflicts and competitions sometimes requires the presence of totally or partially *failed* plans, which conventional plan generators reject. When a goal is abandoned, events occurring prior to the moment of abandonment must be kept as part of the narrative, and thus influence its continuation.

We use two main mechanisms to handle goal abandonment and competitive plan execution: *conditional goals* and *limited goals*. A conditional goal has attached to it a *survival* condition, which the planner must check to determine whether the goal should still be pursued. Limited goals are those that are tried once only, and have an associated *limit* (expressed as a natural number). The limit restricts the number of new events that can be inserted to achieve the goal.

5. USER INTERACTION

The underlying philosophy of the system consists of providing the user with efficient means for exploring coherent alternatives that the story may allow at a given state, and for guiding the plot at the level of events and characters’ goals.

In general terms, the user has direct control only over the Plot Manager. This module, in turn, communicates with IPG to execute plot generation and enforce coherence, and with the Drama Manager to control plot visualization. The Plot Manager comprises the user graphical interface (implemented in Java), whereby the user can participate in the choice of the events that will figure in the plot and decide on their final sequence (Figure 2). Each event is represented by a rectangular box that may assume a specific color according to its current status.

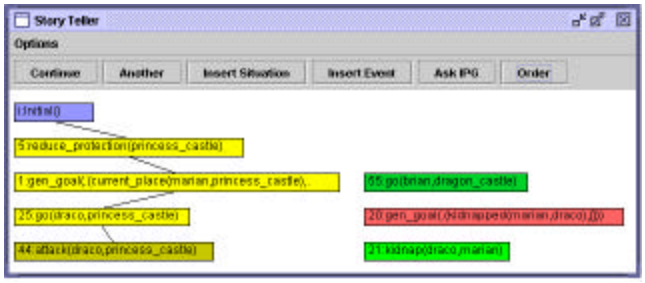

Figure 2: Interface of the Plot Manager.

The user neither has direct control over the scene, nor over the characters themselves. Moreover, user intervention is always indirect, in the sense that any user intervention must be validated by IPG before being incorporated to the current plan.

Plot generation and dramatization are two separate processes, in contrast to pure character-based approaches, where user interaction affects plot structuring at real-time. This means that only during the simulation process the user has an opportunity to intervene in the creation of the plot.

As explained in the previous section, plots result from goals that the characters aim to achieve. At each simulation step, new goals may be inferred and automatically added to the plot, which causes the insertion of a new set of events. The events inserted in the plot so far are sent to the graphical interface for user intervention via the Plot Manager, which offers two commands for automatic plot generation: **another** and **continue**. The command **another**, requests from IPG an alternative solution to achieve the same goals of the step just finished. The command **continue** asks IPG to try to infer new goals and continue the simulation process.

The user intervention may be either weak and/or strong. In a *weak* intervention, the user just selects partially-generated plots that seem interesting from his/her perspective to continue the simulation. This weak form of intervention usually leads the plot to situations that the author of the story has devised beforehand.

The Plot Manager also offers two complementary means for strong intervention in the creation of more personalized stories. Firstly, the command **insert situation** allows users to specify situations that should occur at specific times along the plot by inserting some additional goal to be reached. The specific details of how the goal will be accomplished are left to IPG, which is charged to find a solution, if one exists, using the planning algorithm. It must be noted that, in view of performance considerations, a valid computable plan may fail to be obtained if the search limits currently configured in IPG are exceeded. As in the purely automatic generation, the user may confirm the solution (by indicating **continue**) or request an alternative (**another**), which (as said before), is a case of weak intervention. Secondly, at a lower interaction level, the user is allowed to explicitly insert events into the plot with the command **insert event**. To make the insertions valid, the user must invoke IPG through the **continue** command. At this moment, all user defined operations are submitted to IPG, which runs the planning algorithm to check whether they are consistent

with the ongoing plot. If not, IPG tries to fulfill possible unsatisfied constraints by inserting further new operations in a specific order. The user may also remove user defined operations that were not yet incorporated to (or were rejected by) the planner.

There is one kind of user interaction that is actually mandatory and must be done before dramatization, namely the conversion of the partially-ordered generated plan into a strict sequence, thereby completing the composition of a proper plot. Notice that, if the simulation is resumed afterwards, this addition of new temporal constraints is also an intervention, because it can affect the inference of new goals. To determine the sequence, the user connects the events in a sequential order of his/her choice, respecting the temporal constraints supplied by IPG. The plot's configuration emerges as the user moves the cursor to draw edges linking the operation boxes, starting from the root. To help the user in this process, we utilize colors to distinguish operations that are already connected (yellow), operations that – in view of the temporal constraints – can be immediately connected (green) or cannot yet be connected (red). The starting root is blue and the current operation being rendered is cyan. To connect two operation boxes, the user must click with the mouse over the origin and drag over the destiny (the same process is used to remove a link between two operations). Once the current plot (or part of it) is thus connected into a linear sequence, it can be dramatized by invoking the Drama Manager with the **render** command.

The tool also offers a facility for querying the IPG module about the state of any element of the narrative at a specific time Ti, using our temporal modal logic. This feature is helpful for advanced users to find out, for instance, why an operation or goal is not being allowed, and for authors intent on revising and tuning the story requirements.

6. DRAMATIZATION

We have developed our own engine to support the graphical representation of the plots. It is implemented in C++ language and uses the OpenGL graphical API to support real-time rendering of the 3D elements. Characters in a generated plot are regarded as *actors* for the dramatization. Each actor is implemented as a materialized reactive 3D agent.

The graphical engine does not have to perform any intelligent processing. It is merely responsible for rendering, at each frame, the scene and the current actors' aspect and movements, resulting from real-time interactions with the scene and, occasionally, with other actors. In doing that, it follows the ordered sequence of events generated at the previous stages of simulation. The Drama Manager is the module that synchronizes and temporizes characters' actions, as also the overall graphical representation.

6.1 Dramatization Control

The Drama Manager's job is not limited to assigning the actions that specific characters must perform. It translates symbolic operations into fully realized

3D visual graphical animations. And it must guarantee the synchronism and logical coherence between the intended world and its graphical representation.

As received from IPG, the plot is organized as a sequence of events, each one associated with a discrete time instant, and their effects are supposed to occur instantaneously. For the purposes of visualization, a different concept of time is used. The simulation occurs in continuous real-time and the duration of an operation rendering is not previously known. Variable attributes change as the event is dramatized. In order to make logical and graphical representations compatible, the values of the variables before the dramatization of each event must agree with the pre-conditions of the event and the values at the end with its post-conditions.

The dramatization starts by the selection of a specific event and the execution of the command **render** in the Plot Manager. All subsequent chained events from this point to the end are visualized, unless the user interrupts the process. When an event is activated for rendering, the engine uses the current values of the pertinent attributes as a starting point for the representation. For guaranteeing that the scene remains logically compatible with the plot, the user is not allowed to start visualizing events from an arbitrary point. An event can only be rendered if the preceding one has already been processed. Plot dramatization, however, can be activated many times for exhibiting both final and partial plots. If the user orders a second dramatization of the same plot, it can start from any previously dramatized event. In order to do that, when each event starts to be dramatized, the engine stores in a local database all the attributes of all actors and objects of the scene, having as entry key the time associated with the event.

The user can alternate between plot generation and dramatization. In this case, after a dramatization, new events and time constraints can be added either by the user or by IPG. If dramatization is activated again, it can start only at events that occur before the modifications.

The Drama Manager converts all events into actions, which are delegated to specific actors, at specific times, according to the plot order of events. Whenever an event finishes, the Drama Manager asks the Plot Manager to give it the next event. If none exists, the dramatization stops.

The dramatization of an event ends when the involved actors(s) finish to enact the associated graphical representation. In our experiments, this may take from a few seconds to one minute, depending on the kind of operation and on the scenario features.

6.2 The Scene

For the graphical representation of the plots, according to the genre of the story being represented, the engine loads a specific scenario. The scenario is represented by a hand-built 3D environment that is suitable for the events and characters the story is supposed to contain, taking into consideration the conventions of the genre (e.g. the presence of castles is natural in fairy tales).

During the engine initialization, the Drama Manager queries IPG about the state of the actors, including location and other attributes. From this mo-

ment on, only the synchronism previously explained is used to maintain coherence.

Because most events have an association with the place where they are performed, actors should be constrained, while moving through the scene, to maintain graphical coherence with respect to how they follow the plot directions. Buildings, such as castles and other genre-related objects, serve, more than as an ornament, as a conditioning factor to orient the displacements of the characters, the absolute and relative position where an action is to be executed, and the form to treat collisions. We make use of terrain reasoning and path-planning based on waypoints [12].

6.3 The Actors

Actors have a geometric structure amenable to graphical representation, and are provided with a minimum of planning capabilities, at a low level of detail. Since actors are expected to play the assigned roles, and must plan in order to achieve an adequate performance, some simple planning resources become indispensable, so that, in real-time, an actor be able to make decisions and to schedule the necessary micro-actions. In general, simple path-finding algorithms and direct inter-agent communication schemes are sufficient. Each actor must also incorporate behaviours for interacting with the physical environment and with the other actors. Contrary to the generality of the IPG planner, the local planning of each actor must be simplified to ensure short response times.

During graphical representation of the plot, all control of actions each actor is supposed to perform is made by the Drama Manager. It acts as a director that coordinates sequences of linear or parallel actions performed by the whole cast. It continuously monitors the representation process, activating new tasks whenever the previous ones have been finished. As a director, it also controls the positioning of the (virtual) camera, which an option of LOGTELL permits to be transferred to the user. The manual option provides zooming, rotation, and vertical and horizontal shifting via the keyboard keys; particularly entertaining is the ability to look at the scene from a bird's eye perspective, watching the plot to unfold with all locations in view.

For IPG, as the number of characters increase, the computational effort required to control such characters and their interactions may become prohibitive. However, the use of fewer characters - a small number of actors, consequently - may lead to poor graphical representations. The test scenario, based on fairy tales, that was used as an example in this paper (see section 7) features two heroes, one villain and one victim. To enhance the diversity and liveliness of plots, but also to turn the representation more realistic, we introduced a supporting cast, consisting of groups of soldiers (guardians) in charge of the protection of locations where the leading actors live, and where several actions take place. As opposed to the leading actors, whose actions are predetermined by the plot, these extras are endowed with a higher although still limited level of behavioural autonomy.

As will be seen in section 7, for the purposes of our example IPG totally ignores and not even distinguishes individual extras, since only as groups they

have some influence over the plot conduction. For instance, when the plot is being represented, the graphical engine queries IPG about the current protection level of each location. At this moment, a proportional number of guardians is inserted into the scenario, together with the leading characters. We feel that, either as partially or fully autonomous graphical entities, supporting actors do contribute to enrich plot visualization.

The autonomy of actors, in particular the extras, leaves them free to perform certain actions randomly, such as walking in different directions. When they are required to participate in some plot event, which has always a higher priority, the Drama Manager makes them interrupt momentarily whatever they were doing. The autonomous actions are restricted to not interfering in the execution of the plot. For instance, the guardians cannot inadvertently kill a leading actor.

7. TEST SCENARIO

The test scenario created for LOGTELL corresponds to a small sub-class of the popular Swords and Dragons genre. The possible events were modeled by just a few parameterized operations, which can nevertheless generate a considerable variety of different plots. The specified operations were the following:

- go(CH,PL): character CH goes to place PL;
- reduce_protection(PL): the protection of place PL (represented by the number of guardians) is spontaneously reduced;
- kidnap(VILLAIN,VICTIM): character VILLAIN kidnaps character VICTIM;
- attack(CH,PL): character CH attacks place PL (fighting the guardians);
- fight(CH1,CH2): character CH1 fights character CH2;
- kill(CH1,CH2): character CH1 kills character CH2;
- free(HERO,VICTIM): character HERO frees character VICTIM, raising the degree of affection of VICTIM for HERO;
- marry(CH1,CH2): character CH1 marries character CH2; and
- get_stronger(CH): strength level of character CH is raised (by a magical power).

The model of the genre was completed by the following goal-inference rules, presented here in English for simplicity:

- If a character plays the role of a victim, this character will spontaneously do something that puts her/him in a less protected situation.
- If the protection level of a victim is reduced, the villain will want to kidnap the victim.
- If a victim is kidnapped, a hero will want to free the victim.
- If the affection levels of two characters vis-à-vis each other exceeds 0.95, they will want to marry.

In order to generate the stories, we defined an initial state with the following information:

- Marian is a princess (a potential victim), who lives in her castle.
- Brian and Hoel are knights (the heroes).
- Draco is a dragon (the villain) and lives in its castle.
- Each place has a certain protection level, given by guardians of the princess (positive protection) or guardians of the dragon (negative).
- Each character has a certain strength level necessary for fighting.
- The heroes already have a high affection for the princess.
- Marian does not have any special affection for any of the knights.

Using the tool, it is possible to generate many different plots. Figures 3 and 4 show the Plot Manager presenting two different plots obtained with weak user interventions.

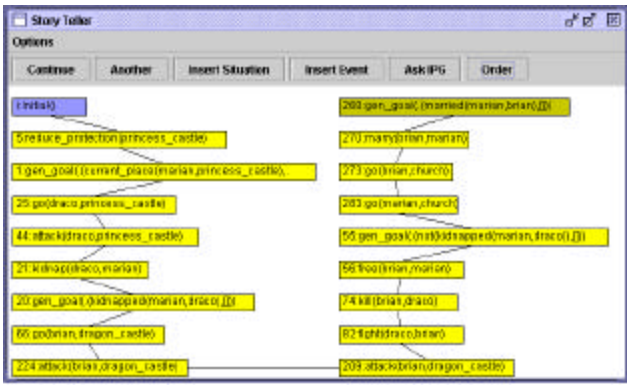

Figure 3: First example of a generated plot.

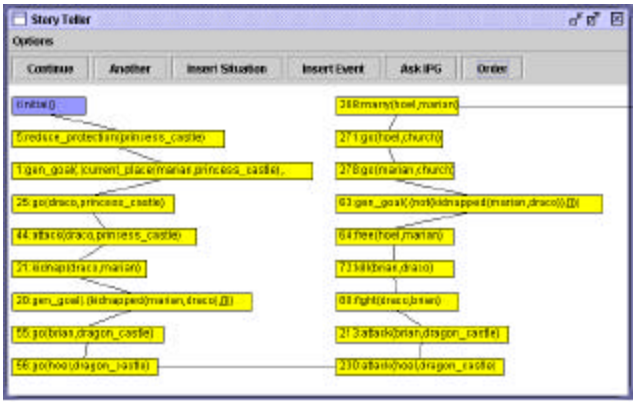

Figure 4: Second example of a generated plot.

Plot in figure 3 is simpler and tells the classical story: “The protection of Marian’s castle is reduced. Draco regards that as an opportunity to kidnap her. Draco then goes to Marian’s Castle, attacks the castle and kidnaps Marian. As a noble knight, Brian feels compelled to save her. He goes to Draco’s Castle, attacks Draco’s castle twice and then fights Draco. Finally, Brian kills Draco and frees Marian, who starts loving Brian as a result. Motivated by their mutual affection, Brian and Marian go to the church and marry each other.”

The Plot in figure 4 is equal until the point where Marian is kidnapped, but, after that, it can be summarized as follows: “As noble knights, Brian and

Hoel feel compelled to save her. Both knights go then to Draco's castle and attack the castle. Brian fights Draco and defeats the dragon. Hoel frees Marian and she falls in love with him and they get married. In spite of doing most of the hard work to save Marian, Brian is not able to marry the princess."

Figures 5, 6, 7 and 8 show some snapshots of the dramatization of the generated plots.

Figure 5: Scenario where plots are dramatized.

Figure 6: Draco attacking Marian's castle.

Figure 7: Brian attacking Draco's castle to save Marian.

Figure 8: Hoel meeting Marian before getting married.

8. CONCLUDING REMARKS

Having implemented a prototype version of LOGTELL, we have been able to run a number of experiments, which seem to demonstrate that combining goal inference, plan generation and user participation constitutes a promising strategy towards the production of plots which are both interesting and coherent. Moreover, our modelling method, based on temporal logic, has proved adequate to capture the conventions of genres encompassing stories with a high degree of regularity, such as fairy tales (as one could foresee, in view of Propp's seminal work) and, consequently, simple Swords and Dragons narratives.

On the negative side, we must admit that modern and post modern genres, with their emphasis on the "transgression" of any conventions should not be so easy to formalize in any way. To our surprise, we came to realize that, on

the contrary, the application domains covered by most business information systems offer an excellent opportunity for the application of our approach [5]. Systems, such as banking, are obviously constrained by providing a basically inflexible set of operations and, generally, by following strict and explicitly formulated rules. We have also run a number of experiments with such "practical" real-life domains (although only with the IPG simulator until now).

Also, plan generation is unfortunately limited by computational complexity considerations. There is however a continuing research effort to improve its efficiency, and we intend to look into that, to try to upgrade the performance of the IPG algorithms. What we have already verified is that an interactive regime, with the intervention of the user at various stages and at different levels, as our methods and implemented tools favour, does much to expand such bounds.

A specific topic for our future research is how to change the LOGTELL approach in order to cope with continuous instead of discrete time during the generation phase. In addition, much work will be dedicated to offer more advanced dramatization resources, such as investing more on *affective computing* [9,18] and improving automatic camera control.

To explore the range of applications of LOGTELL is yet another objective of our project. At present, we already understand how to use it for entertainment, as well as for design and training in the realm of information systems. But the idea of making it an auxiliary tool for authors of various forms of artistic creation (literature, cinema, interactive TV, etc.), allowing, for example, to develop new plots by *adapting* well-known classic or popular works, appears as a most attractive possibility.

REFERENCES

[1] Bal, M. Narratology - Introduction to the Theory of Narrative. University of Toronto Press, 2002.

[2] Cavazza, M., Charles, F., and Mead, S. Character-based interactive storytelling. IEEE Intelligent Systems, special issue on AI in Interactive Entertainment, 17(4):17-24, July 2002.

[3] Ciarlini, A.. Geração interativa de enredos. PhD thesis, Departamento de Informática, PUC-Rio, Rio de Janeiro, 1999.

[4] Ciarlini, A., Veloso, P., and Furtado, A. A Formal Framework for Modelling at the Behavioural Level. In Proc. The Tenth European-Japanese Conference on Information Modelling and Knowledge Bases, Saariselkä, Finland, 2000.

[5] Ciarlini, A., and Furtado, A. Understanding and Simulating Narratives in the Context of Information Systems. In Proc. ER'2002 – 21st. International Conference on Conceptual Modelling, Tampere, Finland, oct. 2002.

[6] Crawford, C. 1999. Assumptions underlying the Erasmatron storytelling system. In Working Notes of the 1999 AAAI Spring Symposium on Narrative Intelligence. AAAI Press.

[7] Furtado, A., and Ciarlini, A. Operational Characterization of Genre in Literary and Real-life Domains. In Proc. ER'99 Conceptual Modelling Conference, Paris, France, 1999.

[8] 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, p. 337-340, Sankt Augustin, Germany, 2001.

[9] Izard, C. E. The psychology of emotions. New York: Plenum Press, New York, 1991.

[10] Mateas, M., and Stern, A. Towards integrating plot and character for interactive drama. In Socially Intelligent Agents: the Human in the Loop, AAAI Fall Symposium, technical report, p. 113-118, Menlo Park, CA, 2000. AAAI Press.

[11] Paiva, A., Machado,I., and Prada, R. Heroes, villians, magicians, ...: Dramatis personae in a virtual story creation environment. In Proc. Intelligent User Interfaces 2001: 129-136.

[12] Pozzer, C. T., Feijo, B., Ciarlini, A. et al. Managing Actions and Movements of Non-Player Characters in Computer Games. In Proc. of the Brazilian Symposium on Computer Games and Digital Entertainment, 2004.

[13] Propp,V. Morphology of the Folktale, Laurence Scott (trans.), Austin: University of Texas Press, 1968.

[14] Riedl, M.; Young, R. Michael. An intent-driven planner for multi-agent story generation. In the *Proceedings of the 3rd International Conference on Autonomous Agents and Multi Agent Systems,* July 2004.

[15]. Scientific American, Volume 283, Number 5, November 2000. [Special issue on digital entertainment].

[16] Spierling, U., Braun, N., Iurgel, I., and Grasbon, D. Setting the scene: playing digital director in interactive storytelling and creation. Computers & Graphics, 26:31-44, 2002.

[17] Sgouros, N.M. Dynamic generation, managing and resolution of interactive plots. Artificial Intelligence, 107, pp. 29-62.

[18] Velázquez, J. D. Modeling emotions and other motivations in synthetic agents. In AAAI-97: Proceedings of The Fourteenth National Conference on Artificial Intelligence, p. 10-15, Menlo Park, CA, 1997. AAAI Press.

[19] Yang, Q., Tenenberg, J., Woods, S. On the Implementation and Evaluation of Abtweak. In Computational Intelligence Journal, Vol. 12, Number 2, Blackwell Publishers 295-318, 1996.