4 Applying a Plan-Recognition / Plan-Generation Paradigm to Interactive Storytelling

4.1. Chapter Preface

This paper, co-authored with Angelo Ciarlini, Bruno Feijó, and Antonio Furtado, presents our first experiments in adding plan recognition to a storytelling system based on plan generation. It gives a description of the system as whole at the time and was presented at the ICAPS06 Workshop on AI Planning for Computer Games and Synthetic Characters in June 2006.

A slightly extended version of the paper, titled "Improving the Scene: Extending LOGTELL to Support a Plan-recognition / Plan-generation Paradigm" and co-authored with Cesar Pozzer, Angelo Ciarlini, Antonio Furtado, and Bruno Feijó, was later presented at the 5th Brazilian Symposium on Computer Games and Digital Entertainment (SBGAMES'06) in November of the same year.

4.2. 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. The resulting systems can be useful for many different purposes, such as storyboard production, education and training, 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 enable the user to tell different stories based on previously computed sequences of actions. In the former case, the opportunities of interaction and the variety of different stories tend to be greater, but a coherent chaining of actions is more difficult to attain.

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 autonomous agents will be complex enough to create an interesting drama.

By contrast, in plot-based models, characters should follow more rigid rules, specifying the intended plot structures. A fundamental inspiration for plot-based approaches has been the seminal work of Vladimir Propp in the field of literary theory [Propp 1968]. Propp observed that significant events within a narrative of a given genre (in his case, Fairy Tales) can be associated with a fixed repertoire of functions, and that these 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 measure 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 [Bal 1997]. First-person tends to be particularly suitable for applications closer to digital games, whereas third-person is usually more appropriate for those involving filmmaking.

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

LOGTELL is based on modelling and simulation. The idea behind LOGTELL is to try to express the basic structure 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 pre-existing stories, but on the dynamic creation of plots. The model includes typical events and goal-inference rules.

Plots are generated by successive cycles of goal-inference, planning, plan recognition and user intervention. Specifically, we try to conciliate both plot-based and character-based modeling. 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 plot generation. On the other hand, the goal-inference rules model the behaviour of the various actors, thus providing some character-based features. The rules declaratively specify how situations can bring about new goals for each character.

Our objective is not to create an immersive experience in which the user takes part in the story as one of the characters. We endeavour, instead, to explore the possibilities of generating a large variety of coherent stories by means of a plan-recognition/plan-generation paradigm. 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 situations and events.

These are rejected by the system whenever it finds no valid way to change the story to accommodate the intervention. Plot dramatization can be activated for exhibiting the final, and also the partially generated, plots. For 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. It was decided to implement an in-house graphical engine, so that the compatibility between the logical model of plots and the corresponding graphical dramatization could be better guaranteed.

The next section describes related work in the area of storytelling. Section 4.4 presents LOGTELL's overall architecture. Section 4.5 describes the main features of the Interactive Plot Generator (IPG), which is the kernel of the system. Section 4.6 illustrates how the user can interact with LOGTELL to generate stories. Section 4.7 shows how the generated plots are dramatized. Section 4.8 illustrates the use of the tool with an example. Section 4.9 contains concluding remarks.

4.3.Related Work

The approach adopted in the DEFACTO project [Sgouros 1999] 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 rather complex. Additionally, it does not allow the use of planning algorithms to develop sequences of 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 appears to be more reactive than deliberative.

The approach described in [Cavazza et al. 2002] 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 [Spierling et al. 2002]. 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.

[Paiva et al. 2001] 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 [Mateas and Stern 2000] 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 Erasmatron system [Crawford 1999] 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 Mimesis [Riedl and Young 2004] to create plots has many similarities with the decisions made while implementing LOGTELL. In both approaches, a nonlinear, 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 [Riedl and Young 2004] 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.

4.4.The LOGTELL Architecture

LOGTELL comprises a number of distinct modules to provide support for generation, editing and visualization of interactive plots, as shown in Figure 4.1. 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 and handled by a graphical engine. Consequently, each module has specific input and output data.

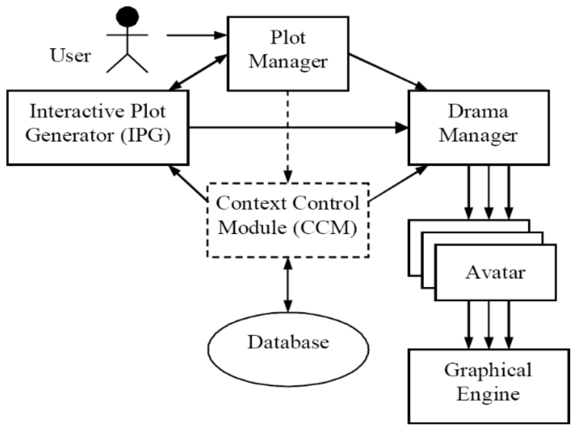
User
Plot Manager
Interactive Plot Generator (IPG)
Drama Manager
Context Control Module (CCM)
Avatar
Database
Graphical Engine

Figure 4.1: LOGTELL Architecture

The user interfaces with the system through 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 total order of events, compatible with the partially ordered sequence generated by IPG, and asks the Plot Manager to activate the Drama Manager.

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 specify the context via the Plot Manager.

4.5.Plot Generation

IPG [Ciarlini 1999] semi-automatically generates plots of narratives of a specific genre. Narratives could be both of literary genres and of more mundane ones, such as the context of a business information system. 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:

- a set of facts (state), 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 limited repertoire of pre-defined operations (typical of the chosen genre) in which characters can take part.

Examples of possible facts in a simple swords-and-dragons context, using a Prolog notation, are listed below:

- dragon('Draco').
- strength('Draco',45).
- affection('Brian','Marian',100).

The facts at a current state change as a consequence of the occurrence of events, which result from the execution of operations by the various characters. For each operation, the following data is supplied:

- a list of arguments, indicating the characters involved in the event, locations, etc.;
- a list of pre-conditions, specifying facts that should or should not hold prior to the execution of the operation;
- a list of post-conditions (effects), specifying facts that hold or cease to hold immediately after the execution of the operation;
- its representation, specifying details about the exhibition of an event caused by the operation.

An example of an operation in the fairy tale context is “kidnap”, having a "villain" as agent and a "victim" as patient. 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 current location”.

Post-conditions are that “the victim will be a captive of the villain” and “both the villain and the victim will be at the villain’s home”. 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 home.

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 [Ciarlini et al. 2000], 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;
- 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 present context 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) $\wedge$ (T1, kidnapped(VIC, VIL) $\rightarrow$

$\exists$ T2 | h(T2, not(kidnapped(VIC, VIL))) $\vee$ 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 fulfil 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 fulfil them. The process alternates goal-inference, plan generation/recognition 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, 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 [Yang et al. 1996] 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.

4.5.1.Composing by Plan Recognition

An alternative way to derive plans for goals is to take, from a conveniently structured library, a pre-existing typical plan, adapting it if necessary to specific circumstances. We use a structure for such libraries of typical plans that also allows plan-recognition by a method proposed by Kautz [1991], and which has been implemented as a complementary feature of IPG. The method consists of matching observed events against the plan definitions (also called complex operations) stored in the library, trying to find one or more plans of which these events may be part.

A structured library with these typical plans (complex operations) is shown in Figure 4.2. Single arrows denote composition (part-of link) and double arrows denote generalization (is-a link).

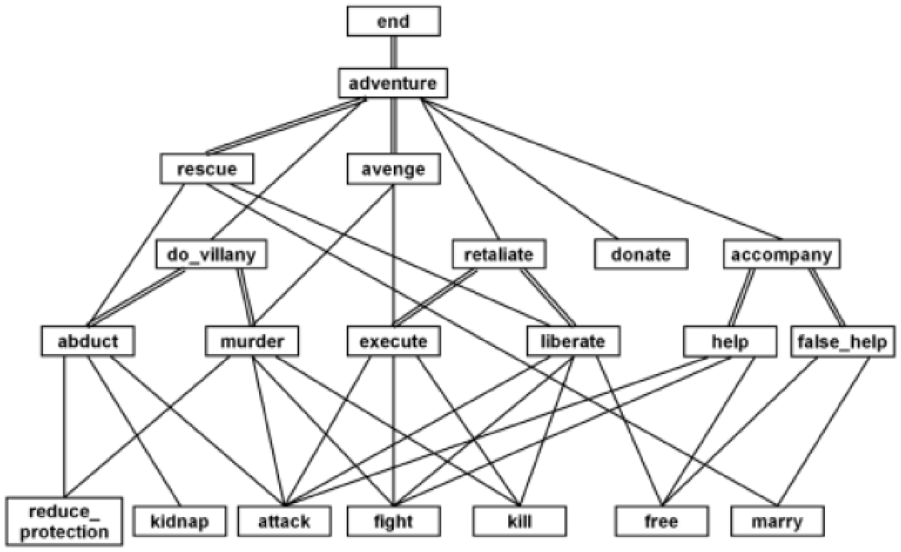

Figure 4.2: Typical plan hierarchy

These complex operations have the same syntax shown for (basic) operations, if the complex operation results from a composition of other possibly complex and/or basic operations, there will be two more parameters, respectively, a list of the component operations, and a list declaring any order requirements holding between them.

Complex operations formed by generalization are also represented, branching down to specialized operations corresponding to alternative ways to reach the same main effects; clauses is_a(<more-specialized-operation>, <more-general-operation>) declare this structural link.

The first step of the plan recognition algorithm is the generation of explanation graphs for the observed (or selected) events. An explanation graph for an event describes in which way this event can be used as part of some end-plan. After the graphs for all observed events are created, they are unified. The final graph will contain all the end-plans where the observed events fit.

Using this approach in LOGTELL, the user can select a group of events and request the possible complex operations that contain them. The system will then insert the complex operations components (if any) in the original plan. More details about this mode of interaction will be provided in the next section.

4.6.User Interaction

People who have no special talent for literary composition, like ourselves, find it difficult to invent interesting plots. Storytelling researchers repeatedly point out that there may be problems when users participating in a game are prompted to function as "authors" [Glassner 2004]. But we usually do not feel so uncomfortable if asked to adapt an existing plot, by introducing small modifications in a gradual fashion.

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.

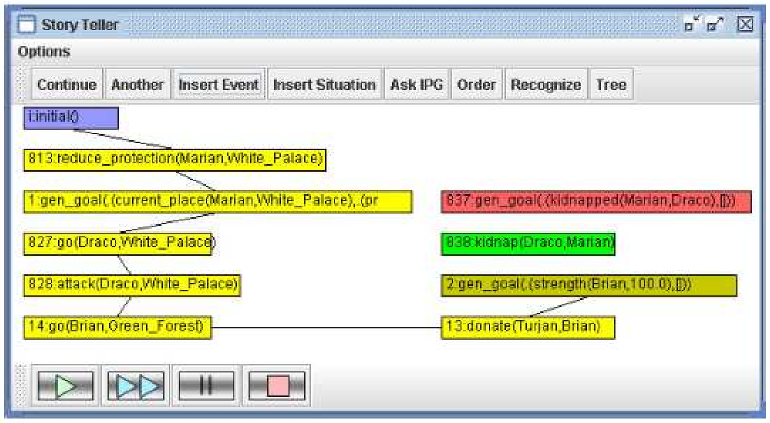
Story Teller
Options
Continue
Another
Insert Event
Insert Situation
Ask IPG
Order
Recognize
Tree
i:initial()
813:reduce_protection(Marian,White_Palace)
1:gen_goal((current_place(Marian,White_Palace),(pr
837:gen_goal((kidnapped(Marian,Draco),[]))
827:go(Draco,White_Palace)
838:kidnap(Draco,Marian)
828:attack(Draco,White_Palace)
2:gen_goal((strength(Brian,100.0),[]))
14:go(Brian,Green_Forest)
13:donate(Turjan,Brian)

Figure 4.3: Plot Manager Interface

In the LOGTELL tool, 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 [Pozzer 2005] 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 4.3). Each event is represented by a rectangular box that may assume a specific color according to its current status.

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 are created in an attempt to fulfil 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 **another** command, requests from IPG an alternative solution to achieve the same goals of the step just finished. The **continue** command asks IPG to try to infer new goals and continue the simulation process.

These two commands provide a form of *weak user intervention*. The user merely selects partially-generated plots that seem interesting from his/her perspective to proceed with 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 offers, in addition, two complementary means for *strong user 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 validate the insertions, 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 or not they are consistent with the ongoing plot. If not, IPG tries to fulfil 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.

Besides these interaction modes, in the current system version, the user can also use two other commands, **tree** and **recognize**. The **tree** command displays the available hierarchy of typical plans and can be used, by itself, as a clue to be taken into consideration when inserting new events in the story. Figure 4.4 shows the hierarchy for our swords-and-dragons example; blue edges denote composition (part-of link) and red edges denote generalization (is-a link).

When using the **recognize** command (which is supported by the plan-recognition feature of IPG) the user needs to mark one or more events already inserted and/or being considered for insertion in the Plot Manager interface and the system will try to match these events, as observations, against the library in an attempt to identify one or more typical plans subsuming them. The system will then show the typical Plan Hierarchy representing the story genre in use with the complex operation found (if any) marked in red and its components marked in orange. The user can then choose if the complex operation found is an interesting one or try to change it into another one that fits the intended story. For example, the list of observations [attack('Brian', 'Red_Castle'), kill('Brian', 'Draco')] fits in both rescue and avenge plans and thus suggests two alternative ways to structure the narrative from which the user may draw his preferences. Upon selecting the desired partial plan, its component events will be inserted in the Plot Manger interface.

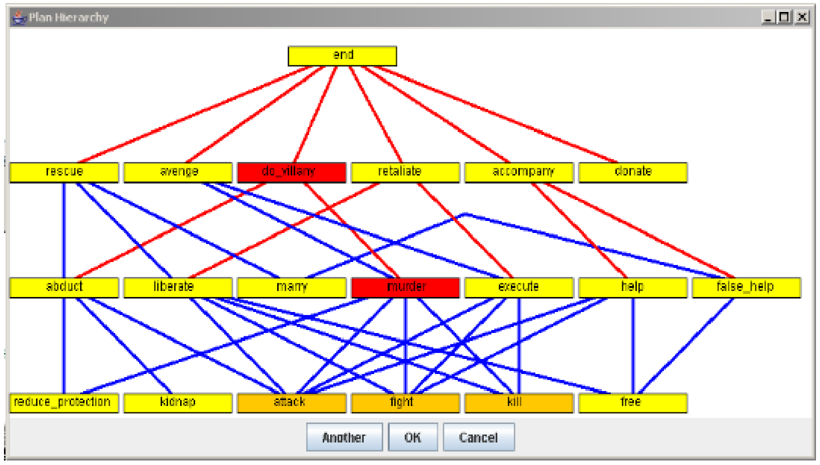

Figure 4.4: Plan Hierarchy Interface

The usage of plan hierarchies can be much enriched if literary indices are made available. For folktales, for example, there is the monumental index compiled in [Aarne and Thompson 1964]. Their identified themes and motifs have always been an inexhaustible source of inspiration for novice and even experienced authors. Treated as fragments of typical plans, they could then be retrieved, to become part of user-composed plots.

Before dramatization, there must be − as said before − one additional user interaction that is actually mandatory, 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 colours 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 source box and drag over the destination box (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 **t**, using temporal modal logic. This feature allows advanced users to find out, for instance, why an operation or goal is not being allowed, and helps authors to revise and tune the story requirements.

4.7.Dramatization

LOGTELL uses a graphics engine developed in-house to support the graphical representation of the plots. It is implemented in C++ and uses the OpenGL graphical API to support real-time rendering of the 3D elements [Pozzer 2005]. Characters in a generated plot are regarded as actors for the dramatization.

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 characters' actions and the overall graphical representation.

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. Figures 4.5 and 4.6 show some snapshots of the dramatization of the generated plots.

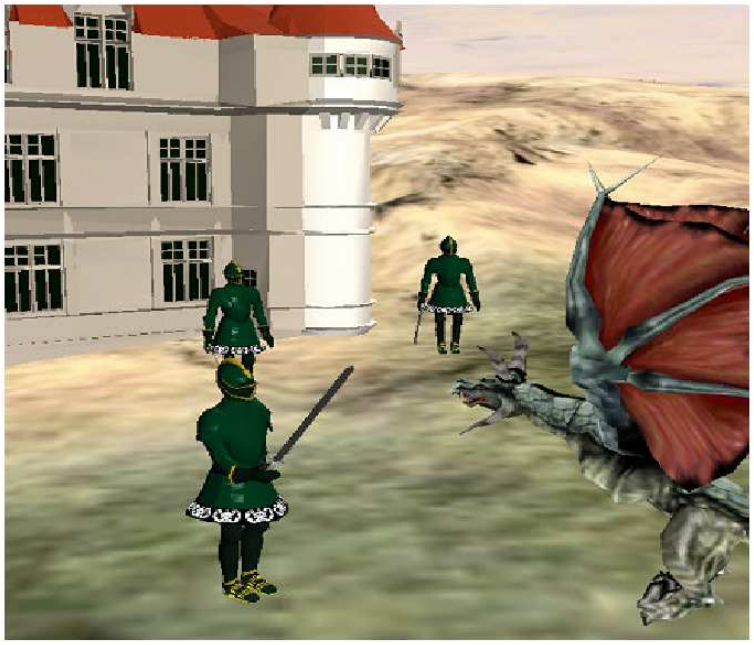

Figure 4.5: Draco attacking Marian’s castle [Pozzer 2005].

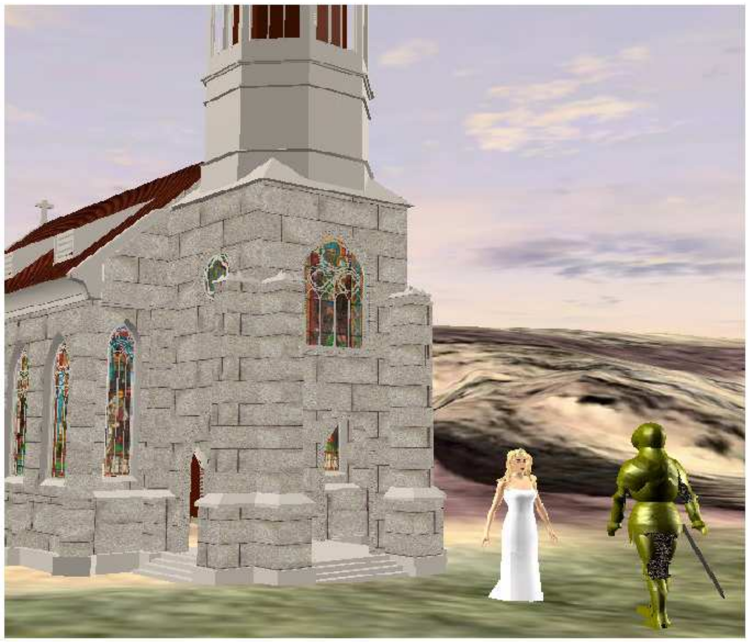

Figure 4.6: Hoel meeting Marian before getting married [Pozzer 2005].

As received from IPG, the plot is organized as a sequence of events, each one associated with a discrete time instant. 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.

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 enacting the associated graphical representation. In our experiments, this may take from a few seconds to about one minute, depending on the kind of operation and on the scenario features.

4.7.1.Scene and Actors

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 3D environment that is suitable for the events and characters that the story is supposed to contain, taking into consideration the conventions of the genre (e.g. the presence of castles).

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 [Pozzer et al. 2004]. 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 achieving an adequate performance, some rudimentary planning resources are 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 the actions each actor is supposed to perform is made by the Drama Manager. It acts as a director that coordinates sequences of 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; some users have found particularly entertaining to look at the scene from a bird's eye perspective, watching the plot 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 used as an example in this paper, based on swords-and-dragons tales, features two heroes, one villain, one victim and a magician. 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 events 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.

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 positively contribute plot visualization.

The degree of autonomy conceded to the extras leaves them free to perform certain actions randomly, such as walking in different directions; this feature can be improved with the integration of an AI middleware [Karlsson and Feijó 2005] into the Drama Manager. When the actors 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. So, the autonomous actions are not allowed to interfere with the execution of the plot; for instance, the guardians cannot inadvertently kill a leading actor.

4.8.Test Scenario

The test scenario currently in use for LOGTELL corresponds to a small sub-class of the popular swords-and-dragons genre. The possible events were modelled 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(VIC,PL):` the protection of place PL (represented by the number of guardians it has) is spontaneously reduced by the prospective victim VIC;
- `kidnap(VIL,VIC):` the villainous character VIL kidnaps VIC;
- `attack(CH,PL):` character CH attacks place PL (fighting its guardians);
- `fight(CH1,CH2):` character CH1 fights character CH2;
- `kill(CH1,CH2):` character CH1 kills character CH2;
- `free(HERO,VIC):` character HERO frees character VIC, raising the degree of affection of VIC for HERO;
- `marry(CH1,CH2):` the two characters get married;
- `donate(CH1, CH2):` strength level of character CH2 is raised by the magical powers of CH1; and
- `bewitch(CH1,CH2):` the double effect of this operation is to instil an evil nature into CH2 and, at the same time, make him or her much stronger.

Besides these basic operations, a hierarchy of complex operations (structured by `is-a` or `part-of` links) was added:

- rescue, avenge - these are the two species of adventure. The rescue variety has components: abduct, liberate, marry, accompany, donate. The other variety, avenge, has components: murder, execute, accompany, donate.

- do villainy, retaliate, accompany - do villainy specializes into: abduct or murder; retaliate specializes into: liberate or execute; accompany specializes into: help or false help.

- abduct, murder, execute, liberate, help, false help. Abduct has components: reduce protection, attack, kidnap; murder has components: reduce protection, attack, fight, kill; liberate has components: attack, fight, kill, free; execute has components: attack, fight, kill; help has components: attack, fight, free; false help has components: free, marry.

We left out two basic operations from this hierarchy. As operation **go** is in fact a component of practically all others, it is therefore assumed to be always present. And bewitch was deliberately excluded, since any plot including it should not be considered typical in the context of our genre (a sort of tolerated *transgression* of the conventions).

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 strongest character playing a heroic role is still weaker than the villain, this character will want to get stronger.
- 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 her.
- If the affection levels of two characters vis-à-vis each other exceeds a threshold, they will want to marry.
- If a victim is killed, a hero will want to avenge her.

As one of the possible starting configurations, we defined an initial state including the following information:

- Marian is a princess, living in a palace (the victim).
- Brian and Hoel are knights (the heroes).
- Turjan is a forest-dwelling magician (a donor, in Propp's sense).
- Draco is a dragon whose lair is in a red castle (the villain).
- The princess, the dragon, and the magician have protecting guardians around their homes.
- Each character is endowed with a certain strength level for fighting.
- The two heroes have a high affection for the princess, which is not reciprocated by her.
- Turjan is neutral with respect to all the others.

4.8.1.Examples of Interactive Step-wise Plot Composition

Using the tool, it is possible to generate many different plots. An example plot tells the classical happy-ending 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. But, before that, he needs to ask for Turjan's magic to raise his strength. He then goes to Draco's Castle, attacks the castle and fights Draco. He kills Draco and frees Marian, who starts loving her saviour. Motivated by their mutual affection, Brian and Marian go to the church and marry each other."

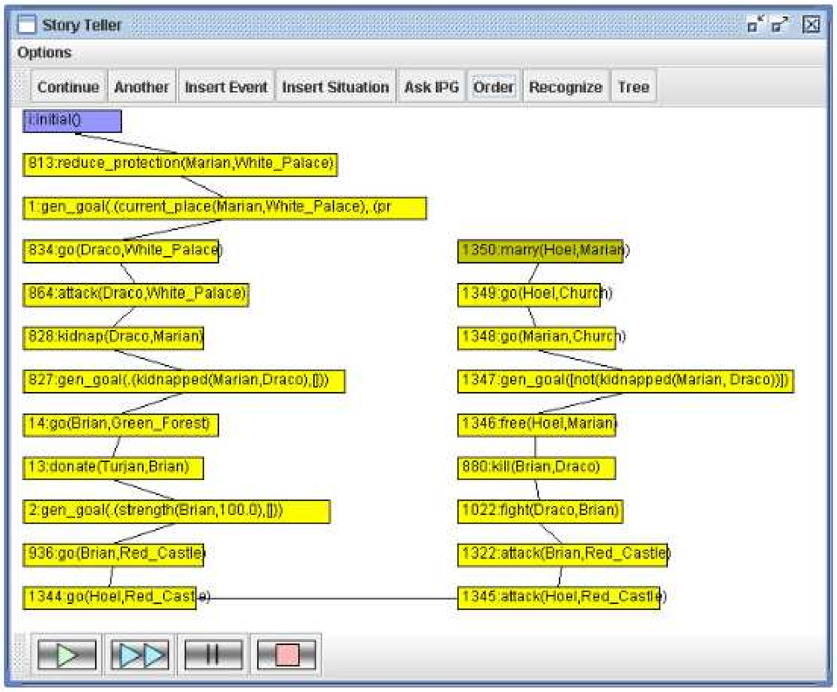
Story Teller
Options
Continue
Another
Insert Event
Insert Situation
Ask IPG
Order
Recognize
Tree
i:initial()
813:reduce_protection(Marian,White_Palace)
1:gen_goal(.(current_place(Marian,White_Palace), (pr
834:go(Draco,White_Palace)
864:attack(Draco,White_Palace)
828:kidnap(Draco,Marian)
827:gen_goal(.(kidnapped(Marian,Draco),[]))
14:go(Brian,Green_Forest)
13:donate(Turjan,Brian)
2:gen_goal(.(strength(Brian,100.0),[]))
936:go(Brian,Red_Castle)
1344:go(Hoel,Red_Castle)
1350:marry(Hoel,Marian)
1349:go(Hoel,Church)
1348:go(Marian,Church)
1347:gen_goal([not(kidnapped(Marian, Draco))])
1346:free(Hoel,Marian)
880:kill(Brian,Draco)
1022:fight(Draco,Brian)
1322:attack(Brian,Red_Castle)
1345:attack(Hoel,Red_Castle)

Figure 4.7: An example of a generated plot.

The plot in Figure 4.7 follows the same course until the point where Marian is kidnapped, but, after that, it can be summarized as follows: “The two knights, Brian and Hoel, propose to save the princess. They both go to Draco’s castle and attack the guardians. But Brian alone fights Draco, and finally defeats and kills it. Hoel then is seen to free Marian, causing her to fall in love with him and become his wife. In spite of doing most of the effort to save Marian, Brian is not able to marry the princess.”

4.9.Concluding Remarks

Having implemented and extended an initial version of LOGTELL, we have been running a number of experiments, which seem to demonstrate that combining goal inference, plan generation/recognition and user participation constitutes a promising strategy towards the production of plots which are both entertaining 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, on the basis of Propp's pioneering 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 a more radical transgression of any conventions should not be so easy to formalize in a systematic way.

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 planning 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 particularly effective help to this interaction is provided by using plan-recognition over libraries of typical plans, which offer expert advice to all kinds of users.

A specific topic for our future research is how to alter the LOGTELL approach in order to offer more advanced dramatization resources, such as investing more on affective computing [Izard 1991, Velázquez 1997] and improving automatic camera control.

To explore the range of applications of LOGTELL is yet another objective of our project. The system could be used, for example, to generate side quests in MMORPGs. Our efforts are now mainly concentrated on the continuing development of our tool, so as to cope with genres involving more sophisticated forms of communication among the characters and a deeper treatment of drives and emotions [Gratch and Marsella 2004].