Planning and Interaction Levels for TV Storytelling

Angelo E. M. Ciarlini[1], Marcelo M. Camanho[1], Thiago R. Dória[1] ,
Antonio L. Furtado[2], Cesar T. Pozzer[3] ,Bruno Feijó[2]

[1]UniRio – Departamento de Informática Aplicada
Av Pasteur, 458 – Térreo – Rio de Janeiro, Brazil
{angelo.ciarlini,marcelo.camanho,thiago.doria}@uniriotec.br

[2]PUC-Rio – Departamento de Informática
Rua Marquês de São Vicente, 225 – Rio de Janeiro, Brazil
{furtado,bfeijo}@inf.puc-rio.br

[3]Universidade Federal de Santa Maria - Dep. de Eletrônica e Computação
Campus Universitário – pr. 07, 3º and. Santa Maria, RS, Brazil
pozzer@inf.ufsm.br

Abstract. Interactivity, coherence and diversity of the stories are key issues for the development of interactive storytelling systems. When stories are to be told via interactive TV, special interaction methods are also necessary in order to cope with high responsiveness requirements. In this paper, we describe the extension of the interactive storytelling system Logtell we have developed to run in an interactive TV environment. Planning algorithms have been applied to provide coherence and diversity for the stories at levels of both plot composition and dramatization. A new architecture was designed to combine these algorithms with special interaction methods incorporated to achieve high responsiveness.

Keywords: Storytelling, Planning, Interactive TV, Logics

1. Introduction

Different approaches have been proposed for interactive storytelling, some of them closer to games and others to filmmaking and Literature. In *character-based* storytelling systems [5], the story emerges from the real time interaction between autonomous agents, each one with its own objectives. The main advantage is to facilitate user intervention, since any actions of any character in the story can be influenced, so that the plot may take different directions. In these systems, stories are usually told with a first-person viewpoint and users play the role of characters, as in games. The main challenge for this approach is keeping the coherence of the stories, since user intervention might create inconsistencies. Some other systems have a *plot-based* approach [12,17], wherein a series of rigid rules, built into the plot, guide the narrative, making user intervention far more limited. In such approach, there is a stronger control over the story being presented, preventing the user to stray from the context de-

fined by the author. The general given structure usually includes beginning, middle and ending points previously fixed by the author, and user interaction affects only the way the story will reach these predefined points. One of the main inspirations for this kind of model has been the seminal literary work of Vladimir Propp [19], at the beginning of the twentieth century. Propp examined a large number of Russian fairy tales, and showed that they could all be described by specializations of 31 typical *functions*, such as villainy, hero's departure, reward, etc. In purely plot-based models, the intervention of the user is more limited, but it is much easier to guarantee coherence together with a measure of dramatic power. There are also alternatives that integrate characteristics of both plot-based and character-based approaches. The interactive system Façade [15], for instance, has a drama manager that keeps the characters largely autonomous most of the time, but changes their behaviour to move the plot forward, conciliating higher-level goals, which are essential to the story, with lower-level goals, specific to the autonomous behaviour of the characters. Erasmatron [8], in contrast, starts from the notion of verbs and sentences. Actions are represented by verbs with roles assigned to characters to form sentences.

Automated planning algorithms are important parts of some storytelling systems, because they can be used to create a logical chaining of events. In [5], for instance, hierarchical task network (HTN) planning is used to control the way characters achieve their goals in accordance with user intervention. HTN planning tends to be efficient but less general, demanding the previous construction of a task hierarchy and methods to perform each task. More flexible planning algorithms have also being adopted, as in [20] for example. Such algorithms do not limit solutions to previously defined alternatives; instead of that, they combine events conciliating different objectives with pre-conditions and effects of each event. The computational effort, however, becomes even greater, due to the complex nature of automated planning.

Logtell [6] is an interactive storytelling system we have developed to create plots in accordance with a formal description of the genre and the initial situation of the story. Logtell is based on modelling and simulation. Its basic idea is to capture the logics of a genre and then verify what kind of stories can be generated and told by simulation combined with 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. 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, specifying how situations can bring about new goals for the characters. In the first version of Logtell, our concern on diversity of stories was focused on plot composition. When plots were totally or partially generated, they could be dramatized via an animation of virtual actors in a 3D scenario, but diversity and interaction in the dramatization phase was limited.

The second version of Logtell has been implemented to run in an interactive TV environment, complying with requirements of high responsiveness, diversity in the dramatization of stories (not only in their composition) and new interactivity options. A new architecture was proposed, in which Logtell resorts to automated planning at various levels to create plots, to control the dramatization of scenes and to animate virtual actors in a 3D scenario. By using plans based on formal specifications, we try

to guarantee coherence and diversity for plot composition and dramatization. At the same time, we try to offer the users the opportunity to interact at various levels in both active and passive fashions, which might be needed when the user just wants to watch the story unfold by itself.

In this paper, we present the levels of automated planning and their relation with interactivity in the new version of Logtell. Section 2 presents new Logtell's overall architecture. Section 3 describes our use of automated planning for plot composition. Section 4 describes the interaction methods for plot composition in the new architecture. Section 5 shows how dramatization has been structured to provide more diversity for stories and additional opportunities for user interaction. Section 6 contains our concluding remarks.

2. New Logtell's Architecture

Logtell adopts a client-server model as shown in Figure 1. The client-side is responsible for user interaction and dramatization of stories. The application server side is responsible for simulation. All processes run in parallel and are coordinated with each other. For each story being told there is an application running in the server, while one or many applications are kept running in different clients. This takes care of the case wherein multiple users are simultaneously sharing the same story. If clients are set-top boxes for interactive TV, their computational resources are limited, making it hard to perform CPU-intensive tasks such as automated planning. By concentrating simulation tasks in application servers, it is easier to achieve higher scalability. In addition, it is also easier to exert control when a single story is shared by many users.

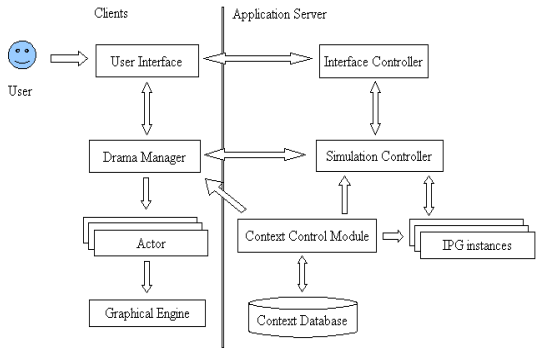


Fig. 1. Logtell's Architecture

The access of all modules to the context of the stories, specified in the Context Database, is always performed via the Context Control Module (CCM), which runs in the server. The context contains the description of the genre, according to which stories are to be generated, and the initial state, specifying characters and the environment at the beginning of the story. The genre is basically described by: (a) a set of parameterized basic operations, with pre- and post-conditions, logically specifying the

events that might occur; (b) a set of goal-inference rules, specified with a temporal modal logic, which define situations that lead characters to pursue the achievement of goals; (c) a library of typical plans, corresponding to typical combinations of operations for the achievement of specific goals, which is organized according to "part-of" and "is-a" hierarchies; (d) logical descriptions of initial situations for the stories, introducing characters and their initial state; (e) a nondeterminisc automaton for each operation specifying alternative ways events based on it can be dramatized; and (f) graphical models of 3D virtual actors.

Plot generation is performed by the Interactive Plot Generator (IPG). IPG generates plots as a sequence of chapters. Each chapter corresponds to a cycle in which user interference is incorporated, goals are inferred and planning is used to achieve the goals. IPG is controlled by the Simulation controller. Multiple stages, each one corresponding to a chapter, usually occur in order to generate a plot.

On the client side, the user interacts with the system via the User Interface, which informs the desired interactions to the Interface Controller placed at the server side. The Drama Manager requests the next event to be dramatized from the Simulation Controller, and controls actor instances for each character in a 3D environment running on our Graphical Engine. On the server side, the Interface Controller centralizes suggestions made by the various clients. When multiple users share the same story, interactions are selected according to the number of clients that requested them. When there is only one client, suggestions are automatically sent to the Simulation Controller. The Simulation Controller is responsible for: (a) informing the Drama Manager, at the client side, the next events to be dramatized; (b) receiving interaction requests and incorporating them in the story; (b) selecting viable and hopefully interesting strong interactions to be suggested to the users; (c) controlling a number of instances of the Interactive Plot Generator, in order to obtain the next events to be dramatized; and (d) controlling the time spent during simulation.

In the new architecture, there can be various instances of IPG running on the server. Besides the instance corresponding to the current flow of the story, others are used to avoid interruption in the dramatization. The simulation has to be some cycles ahead of the dramatization to keep responsiveness. When there is no user intervention, goals are inferred and events are planned continuously. When users interact with the system, however, they interact in accordance with the events that are currently being dramatized. The Simulation Controller keeps snapshots of the state of the simulation after each cycle, so that simulation can be resumed from the correct point after an intervention. Logical coherence of a requested intervention is always checked before being incorporated, or either discarded if inconsistent. When an intervention is incorporated, simulation has to discard simulation cycles that were previously planned without taking the intervention into account. In order to be prepared for interventions, the system creates other instances of IPG, simulating the incorporation of strong interventions to be suggested to the users. But the suggestions are only communicated if the IPG instance confirms that they are consistent. And if they are accepted, the next events are already planned and there is little risk of interruption.

The time spent for simulation is constantly monitored by the Simulation Controller. When there is risk of interruption in the dramatization because there are not enough events planned, a message is sent to the Drama Manager, so that strategies are used to extend the dramatization of the current events until the situation is normalized.

3. Planning for Plot Composition

IPG semi-automatically generates plots of narratives of a certain genre as a simulation process. The initial situation chosen in the context database is the starting point for the simulation. Facts from the initial configuration might be modified by *events* (denoted by instances of operations). 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 class of characters, there are goal-inference rules, specifying, in a temporal modal logic formalism [7], the goals that the characters of the class will have when certain situations occur during a narrative. 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 situation. 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 chapter of the story is almost finished. During the generation phase, plots are represented by partially-ordered sets of events. Before dramatizing, however, the total order of the events has to be determined, as explained in Section 4. If the user does not like the story, IPG can be asked to generate another alternative for a chapter and to develop the story from this point on. If the user does not interfere in the process, chapters are continuously generated by inferring new goals from the situations generated in the previous chapter. If new goals are inferred, the planner is activated again to fulfill them. The process alternates goal-inference and planning until the moment the user decides to stop or no new goal is inferred. Users can also interfere in the process by choosing alternatives and forcing the occurrence of events and situations as described in Section 4. Notice that, in this process, we mix forward and backward reasoning. In the goal-inference phase, we adopt forward reasoning: 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, handled through backward reasoning. To establish them, the planner might insert previous events with further unfulfilled pre-conditions, and so on.

We use a non-linear planner that works in the space of plans, not assuming a total order for the events. It was implemented in Prolog, adapted from [21], with extensions. A non-linear planner 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. The planner performs a heuristic search for good plans. It works on many plans in parallel and, at each time, selects the candidate with minimal estimated cost for achieving a complete solution, i.e. a configuration in which all goals and all preconditions of all events are satisfied. It then selects a pre-condition of an event that is not necessarily true and tries to make it true. While doing that, the planner generates all possible successors of the current plan. It considers the possibility of using events already in the plan to establish a pre-condition as well as

the insertion of new events. All possibilities for solving conflicts between events in the establishment of pre-conditions are considered.

To meet the requirement of high responsiveness, efficiency of the planning algorithm becomes essential, and possible enhancements to the planner deserve special attention. Most of the successful cases in real-world applications of automated planning are based on algorithms that use hierarchical task networks (HTNs), such as [10, 16]. These algorithms tend to be more efficient because they reduce the search space. They depend however on the previous definition of a hierarchy of tasks and methods to perform the tasks. The task network has to be built for each domain, and all generated solutions are limited to combinations of previously defined ones, giving less flexibility to the creation of alternatives. Algorithms based on HTNs can work with a partial order of events, and HTNs are compatible with the way we specify a hierarchy of typical plans, used for plan-recognition as explained in Section 4. Taking advantage of that, we are enhancing our planner, producing a hybrid planner capable of working with our type of simulation, mixing non-linear planning with HTN planning.

Typical plans are abstract events usually corresponding to various alternatives of combining basic events to achieve ordinary goals. They can have pre-conditions and effects, but these pre-conditions and effects differ from those of basic events because they are necessary instead of sufficient, that is, additional pre-conditions and effects can be observed in a specific alternative.

When a goal has to be achieved, the system might choose either a basic event or an abstract one to achieve it. In this way, abstract events are initially inserted as if they were basic. When a plan is complete but contains abstract events, HTN planning is applied to create alternatives specializing each abstract event into a set of compatible partially-ordered basic events. Heuristics are used to determine whether basic or abstract events should be chosen for the achievement of a goal (or subgoal). Abstract events tend to be preferred, because a single abstract event is usually necessary to achieve a specific goal. In this case, the planning process is essentially reduced to HTN planning. By doing this, performance is enhanced and, at the same time, we keep the generality and flexibility of our original simulation tool.

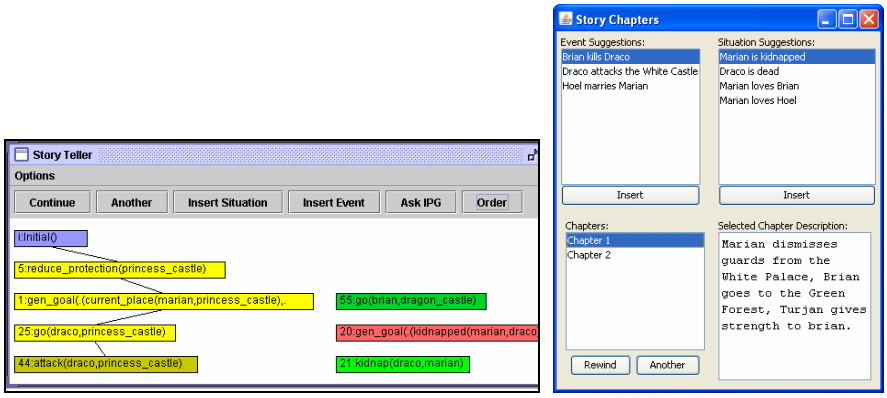
In recent years, some neoclassical planners that work in the space of states, with planning-graph techniques [3,11], have given good results. The use of heuristics and control strategies to prune the search space, as in [9, 13], are also promising strategies. We are also studying the possible adoption of these techniques.

4. Interaction Methods for Plot Composition

The underlying philosophy of the system consists of providing the user with efficient means for exploring coherent alternatives that the story may allow, and for guiding the plot at the level of events and characters' goals. Interaction can occur in a step-by-step mode, in which the user can interfere after the generation of each chapter, or in parallel with dramatization, in continuous mode.

Step-by-step interaction is analogous to debugging a program. When it is activated, the graphical interface presents a graph with the plot generated after each chapter. Figure 2.(a) shows the graphical interface for this kind of interaction. Each event is

represented by a rectangular box that may assume a specific color according to its current status.As explained in the previous section, plots result from goals that the characters aim to achieve. At each simulation step (i.e. a chapter), new goals may be inferred and automatically added to the plot, which causes the insertion of a new set of events. In step-by-step mode, the events inserted in the plot so far are sent to the graphical interface for user intervention, which offers two commands for automatic plot generation: *another* and *continue*. Command *another* requests from IPG an alternative solution to achieve the same goals of the chapter just finished. Command *continue* asks IPG to try to infer new goals and continue the simulation process. This weak form of intervention usually leads the plot to situations that the author of the context has devised beforehand. The Plot Manager also offers two complementary means for strong intervention in the creation of more personalized stories. Commands *insert situation* and *insert event* allow users to specify situations (specified as goals to be achived) and events that should occur at specific times. The specific details of how the goal will be accomplished or the event is inserted are left to IPG, which might insert further events to obtain a consistent plot. Insertions are rejected when they are inconsistent or search limits currently configured in IPG are exceeded. In step-by-step mode, the user has to convert the partially-ordered generated plan into a strict sequence, that can be dramatized. To determine the sequence, the user connects the events in a sequential order of his/her choice, respecting the temporal constraints supplied by IPG. 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.



(a) **(b)**

Fig. 2. Interfaces for interaction: step-by-step mode (a) and continuous mode (b)

Considering that our storytelling system is designed to run on an environment such as interactive television, viable alternatives of interaction should not hinder the experience of watching the dramatization. Interaction cannot demand a high level of attention, as in step-by-step mode, unless the user opts to halt the dramatization in order to interact. The interruption of dramatization to allow the user to interact, which was mandatory in the first version of Logtell, is still allowed, but it is expected to be

an exceptional case, being replaced by other more expedient kinds of interaction in continuous mode.The Simulation Controller plays an important role in the implementation of the new forms of weak and strong interactions.

Weak interactions work basically around the "normal" flow of the story, as one would have with *continue* and *another* commands in the first version of Logtell. The Simulation Controller directs the flow of the story by automatically selecting alternatives and total order of the events to be dramatized. Such selections can be done either in a random way or based on user satisfaction models. The idea is that stories are worth of telling even if the user only watches the dramatization, with no intervention. Even in this case, we keep the possibility of watching different but still coherent stories based on the same initial configuration.

Other kind of weak interaction is to return to a previous chapter in the narrative, so that alternative directions for the plot can be chosen. In step-by-step mode, at the end of a simulation phase, the user could examine the new events not yet incorporated, and decide whether or not to consider them interesting; if not, the user would ask for the generation of an alternative. In continuous mode, with parallel dramatization, it becomes highly desirable to extend the backtracking range, to allow the user to undo the narrative up to any previous stage, and have a chance to find how it would develop if different alternatives were chosen at such point. For this objective, different snapshots of the simulation process are kept in memory by the Simulation Controller, corresponding to the end of each chapter.

Suggestions of strong interactions correspond either to the insertion of specific events in the plot or to the directive that a certain situation should occur at a specific time. These suggestions can be made based on the events already inserted in the story and on an analysis of the context of the genre. We shall describe two methods for their creation. Independently of the method utilized, the Simulation Controller checks the consistency of the suggestion with the current narrative, before sending it to the user. The first method to obtain a suggestion of strong interaction uses our library of typical plans. Typical plans usually consist of certain combinations of events whereby the various characters pursue their goals, but they can also correspond to motifs, i.e. recurring structures compiled in the course of critical studies on the genre [1]. IPG contains a procedure for the recognition of plans, based on an algorithm specified by Kautz [14]. The procedure is able to discover that some given events are compatible with a motif for which we have a typical plan, enabling the Simulation Controller to suggest the inclusion of additional events contained in the plan. The second method to obtain a suggestion of strong interaction is based on the application of the goal-inference rules against the current plot. A suggestion, in this case, is triggered by observing that the insertion of a fact at a specific time might lead to the inference of a new goal.

Figure 2 (b) shows the interface used for user interaction in continuous mode. Events and situations to be incorporated in the next chapter are continuously suggested to the user. By selecting a suggestion, the user asks the system to try to force a certain event or situation in the next chapter. If it is logically possible, the new suggestion is likely to be incorporated as a strong interaction; if not, it is discarded. By selecting a chapter and pressing button *rewind*, the story backtracks up to the desired point. By pressing *another*, the story also backtracks up to the desired point but an alternative for the selected chapter is adopted and story continues from this point on.

Notice that there is also a window listing the events corresponding to the selected chapter.

Besides the strategies for incorporating weak and strong interactions that we had in the first version of Logtell, further kinds of interactions are under consideration, such as letting users to: insert abstract events and situations in the story, which are automatically specialized by the simulation process; tune narrative tensions by means of numeric scales referring to levels of violence, romantic turns, etc.; share stories with other users, so that more than one user influence the same story; and suggest the insertion of events and situations using natural language.

5. Planning and Interaction at the Dramatization Level

The Drama Manager converts all events into actions, which are delegated to specific virtual actors, at specific times. In the first version of Logtell, for each event there was a single combination of actions for its representation. There was little diversity in the dramatization and the time for each event almost did not vary. In the second version, we introduced nondeterminism for the specification of each event. Each event is assigned to a nondeterminisc automaton, created by authors and stored in the context database. By doing this, we try to increase the number of possible dramatizations, to enable the users to interfere in the story at this level, and to vary the time for the dramatization of an event, according to the convenience of the overall narration of the story. In each automaton, states are described by invariants, that is, logical formulae relating dramatization attribute variables. An automaton is said to be in a certain state if and only if its invariant is satisfied. Transitions correspond to actions delegated to actors. These actions are nondeterministic and can lead the automaton to alternative states. The new state may depend on the interference of the user or can be achieved according to the autonomy of the actors. Figure 4 shows an automaton created to represent possibilities for the dramatization of an event, in which a villain kidnaps a victim.

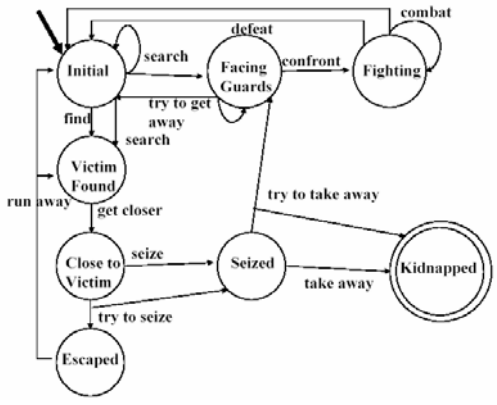


Fig. 2. Example of automaton for the representation of an event.

The Drama Manager has now an abstract control layer that controls the automaton corresponding to the current event. The control layer checks the values of the attribute variables and detects the current state. Since the current state is known, an action is chosen to be performed by the actors. The choice of the action depends on alternative plans for leading the automaton from the initial state to a final state. These plans are called policies and they map states into actions to be performed.

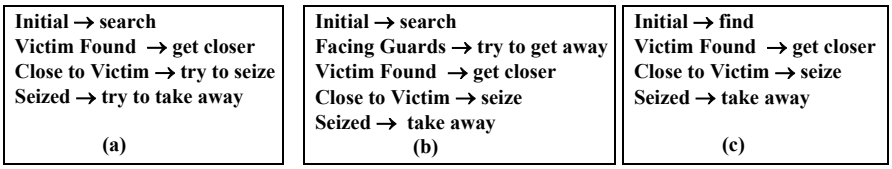


Fig. 3. Examples of policies: weak (a), strong with cycles (b) and strong (c).

In order to create policies, we adopt planning as model checking [2,18], which is an approach for planning under uncertainty. Model checking [4] is a formal method to check whether a certain logical formula is a model for a structure. Reachability of a certain state in an automaton is a typical application of model checking. In planning as model checking, the planning algorithm examines the automaton in order to create policies that take the automaton from one state to another state. Generated policies can be weak, strong with cycles or strong. In weak policies, there is at least one path from the initial state to the goal state, but states from which it is not possible to reach the goal according to the policy can be reached. When a policy is strong with cycles, the goal state is always reachable but cycles can occur, so that the time to reach the goal might be virtually infinite. Strong policies guarantee that, from any state in the policy, the goal state is reached at some moment. A state is said to be safe if it has at least one strong policy leading the automaton from this state up to the final state. In our automata, only safe states are allowed. Figure 6 shows weak, strong with cycles and strong policies for our example.

Based on the automaton for an event, various policies of all kinds can be built beforehand using planning as model checking. In this way, there is no need to spend time planning during dramatization. The set of policies correspond to alternatives for leading the automaton from specific states to the final state. Occasionally, policies to postpone the end of an event, leading the automaton to intermediate states might be useful. Policies are chosen and replaced during dramatization in accordance with some principles: (i) In order to obtain variety during dramatization, the choice should occur at random up to a certain extent. (ii) At the beginning of the dramatization of an event, weak policies are accepted (and even preferred) because they tend to be more realistic, there are usually many weak policies available (which guarantees diversity) and duration is not yet a problem. When a state not mapped to any action in the current policy is reached, the policy has to be replaced. (iii) At a certain point of the dramatization, there might be time to accept cycles, but it might not be convenient to postpone termination for a long time. Unnecessary change of policies should be avoided and policies that are strong with cycles are preferred. (iv) When there is little time available, strong policies are needed to force termination.

The main difficulty for user interference in the dramatization is the fact that the scene might be modified in such a way that the story would become inconsistent. By using planning as model checking, it is possible to introduce opportunities for interaction at this level. If the nondeterminisc automaton considers opportunities for user intervention and this intervention is restricted to automaton's transitions, there is no risk of not reaching the final state of the current event. Forms of interaction at the dramatization level are still under investigation, but two alternatives seem to be straightforward. The first one corresponds to interactions in which the user would choose the result of a nondeterministic action explicitly. In the second one, users would control actions of avatars, but restricted to possibilities specified by the automaton.

Besides the use of automated planning for plot composition and for controlling the dramatization, our virtual actors have 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. The local planning of each actor must be simplified to ensure short response times.

6. Concluding Remarks

Logtell has been extended to work in an interactive TV environment, conciliating requirements of coherence and diversity of stories with high responsiveness. In this paper, we presented the various levels of automated planning and interaction we have incorporated to perform tasks that, in filmmaking, are typical of authors, directors and actors. Planning techniques have shown to be important to provide diversity and coherence for our stories. In order to cope with high responsiveness, new interaction methods have been investigated and implemented and, as efficiency became essential, the planner for plot generation has been enhanced. Planning as model checking has also been introduced in dramatization, which is important to diversify the ways events are dramatized, to control dramatization time and to let users interact with the story at the dramatization level.

Several other research topics are being investigated to achieve realism and good quality in the narration of stories by Logtell, including models for representing beliefs and emotions of characters, alternative ways to control the camera, and automatic text generation for supporting dialogs and the explicit narration of stories.

References

1. Aarne, A. *The Types of the Folktale: A Classification and Bibliography*. Translated and enlarged by Stith Thompson, FF Communications, 184. Helsinki: Suomalainen Tiedeakatemia (1964)

2. Bertoli, P., Cimatti, A., Roveri, M., and Traverso, P.: Planning in Nondeterministic Domains under Partial Observability via Symbolic Model Checking. In: 17th International Joint Conferences on Artificial Intelligence, pp. 473-478. Washington (2001)
3. Blum, A. L. and Furst, M. L.: Fast Planning through planning graph analysis. *Artificial Intelligence*, 90:281-300 (1997)
4. Burch, J.R., Clarke, E.M.; McMillan, K.L., Dill, D.L. and Hwang, L.J.: Symbolic Model Checking: 10^{20} states and beyond. Information and Computation, 98(2):142-170 (1992)
5. 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 (2002)
6. Ciarlini, A.E.M., Pozzer, C. T., Furtado, A.L. and Feijo, B.: A Logic-Based Tool for Interactive Generation and Dramatization of Stories. In: *Proc. ACM SIGCHI International Conference on Advances in Computer Entertainment Technology* (ACE 2005), Valencia (2005)
7. 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) .
8. Crawford, C.: Assumptions underlying the Erasmatron storytelling system. In: *Working Notes of the 1999 AAAI Spring Symposium on Narrative Intelligence*. AAAI Press (1999)
9. Doherty, P. and Kvarnström, J. TALplanner: A temporal logic based planner. *AI Magazine*, 22(3): 95-102 (2001)
10. Erol, K.; Hendler, J.; Nau, D. S. UMCP: A sound and complete procedure for hierachical task-network planning. In: *Proceedings of the International Conference on AI Planning Systems (AIPS)*, pp. 249-254 (1994)
11. Geverini, A. and Serina, I. LPG: A planner based on local search for planning graphs. In Proceedings of the International Conference on AI Planning Systems (AIPS), pp. 968-973 (2002)
12. Grasbon, D. and 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*, pp. 337-340, Sankt Augustin, Germany (2001)
13. Hoffman, J. FF: The Fast-Forward planning system. *AI Magazine*, 22(3):57-62 (2001)
14. Kautz, H. A.: A Formal Theory of Plan Recognition and its Implementation. In: Allen, J. F. et al (eds.)*: Reasoning about Plans*. Morgan Kaufmann, San Mateo (1991)
15. Mateas, M. and Stern, A. Towards integrating plot and character for interactive drama. In: Dautenhahn, K., editor, *Socially Intelligent Agents: The Human in the Loop, AAAI Fall Symposium, Technical report*, p. 113{118, Menlo Park, CA. AAAI Press (2000)
16. Nau, D. S.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, W.; Wu, D. and Yaman, F.: SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20:379-404 (2003)
17. Paiva, A., Machado, I. and Prada, R. Heroes, villains, magicians, ... Dramatis personae in a virtual story creation environment. In Proc. *Intelligent User Interfaces* (2001)
18. Pistore, M., Traverso, P.: Planning as model checking for extended goals in non-deterministic domains. In: *Proc. 17th International Joint Conference on Artificial Intelligence*, pp. 479--484. Washington (2001)
19. Propp, V. *Morphology of the Folktale*, Laurence Scott (trans.), Austin: University of Texas Press (1968)
20. Riedl, M.and Young, R. M. An intent-driven planner for multi-agent story generation. In: the *Proceedings of the 3rd International Conference on Autonomous Agents and Multi Agent Systems* (2004)
21. Yang, Q., Tenenberg, J. and Woods, S.: On the Implementation and Evaluation of Abtweak. *Computational Intelligence Journal*, Vol. 12, Number 2, pp. 295-318, Blackwell Publishers (1996)