Verification of Temporal Constraints in Continuous Time on Nondeterministic Stories^{*}

Eric T. Araujo and Angelo E.M. Ciarlini

Department of Applied Informatics, UNIRIO
Av. Pasteur, 458, Urca, Rio de Janeiro, Brazil, Cep: 22.290-240
{eric.teixeira, angelo.ciarlini}@uniriotec.br

Abstract. This paper proposes a language based on temporal logics to specify constraints on nondeterministic stories, generated by interactive storytelling systems, taking into account that time is continuous and branched. These constraints are checked, in real time, during story generation. Constraints to be checked are described by temporal formulas over the properties of the story. To make possible the verification in real time, it is assumed that these properties are piecewise-linear functions. A prototype, using Constraint Logic Programming for the verification of properties, is operational.

Keywords: Continuous Change, Interactive Storytelling, Constraint Programming, Branching Time.

1 Introduction

Interactive storytelling systems are applications in which stories are generated and told in accordance with the user interference. They have been developed for different purposes, such as helping the authoring of stories, education and training and, of course, in the context of games and entertainment in general. In most of the cases, there are relevant continuous properties that vary over time. The level of tension, the intensity of emotions and soundtrack features are examples of such properties, which we might want to control. We can understand stories as a finite set of events that are logically connected so that goals are achieved. These events should generate peaks in the intensity of emotions, in order to keep the audience's interest in the story. The sequence of events should also generate an overall tension during the narrative, which typically reaches a climax that leads to the end of the story [1].

The author of a story should then be concerned with the variation of these properties. When the story is linear and closed, it is possible to predict, at least intuitively, their values and the degree of involvement of the reader at every moment. In interactive storytelling systems, the task is more complicated because there are different alternatives, generated automatically, to be considered. In particular, if the events of the stories are considered to be nondeterministic, the outcome of each event might lead to different continuations, providing then a greater degree of diversity in plots and more opportunities for user interaction. In this case, interactive stories (or parts of them generated at a given time) are trees of events, in which each path from the root to a leaf is an alternative plot.

^{*} This work has been partially supported by CAPES - Brasil: Project RH-TVD # 133/2008.

A nondeterministic model for the events within an interactive storytelling context is adopted in [2], where an extension of the system Logtell [3-5] is proposed to deal with nondeterminism and weak goals (representing attempts to reach a certain situation, which could either succeed or fail). Nondeterministic planning is used to generate a tree of events for every chapter of a story. The path followed during the 3D dramatization depends on the selection of branches, which occurs either at random or in accordance with (direct or indirect) user intervention. As continuous properties might vary in accordance with the followed path, it is interesting to provide a mechanism to check whether a tree of events satisfies a certain temporal constraint. In this paper, we propose an approach for verifying constraints on continuous properties in chapters generated by the new version of Logtell. The verification of continuous properties occurs just after the generation of each chapter. When constraints are not satisfied, the chapter should be discarded and another alternative chapter (another tree) generated.

As plot generation occurs in real time, in parallel with a continuous 3D plot dramatization, special care is taken to guarantee that the verification process does not hinder the overall storytelling process. Properties vary continuously over time and the evaluation of continuous changes tends to be very time consuming. Furthermore, as an interactive story corresponds to various paths resulting of the branches after each event, we should also be able to talk about these paths. It is therefore a problem to be approached by temporal logics such as CTL (Computation Tree Logic) [6], with the additional requirement of treating an infinite and continuous set of states. In order to be able to verify constraints in real time, we assume that the continuous properties are piecewise linear functions, the variation of the properties being specified by a sequence of constant values for their derivatives assigned to each event. By doing this, we are able to use *constraint logic programming* [7] to verify the satisfaction of constraints. As the height of the trees generated for each chapter is limited and the number of branches after each event is usually a number between one and three, we are able to avoid a combinatorial explosion on the number of different paths to be tested.

2 A Logic for Nondeterministic Continuous Events

As stories are created by means of an automatic process, formal models for the variation of the properties during the events have to be specified, so that an automatic verification procedure can be applied. A story, for example, may have a chapter in which a villain tries to kidnap a victim, with different possible outcomes. The villain might succeed, but the victim might also escape or die. If the desired level of suspense is reached, the chapter can be considered valid. If it is not reached, the chapter can be discarded. When time is considered to be continuous, the set of space states becomes infinite. Verifying the satisfaction of temporal formulas over an infinite space of states is extremely difficult, because there is no possibility of enumerating all possible states. We have then to resort to symbolic methods, which demand some kind of simplification in the way properties are represented.

In order to be able to verify constraints, we created a model for the specification of continuous properties as functions of the execution time. Each event has a function that specifies the derivative of the property. The derivative can vary during the event,

but only as a sequence of constant values. Each constant value is assigned to a percentage of the execution time of the event. To calculate the level of a property at a certain moment, we can integrate the derivative from the beginning of the story until that moment, considering the specification of the derivative for each event. If we assume that the initial value of the continuous property varies linearly, we are able to use CLP in the verification process, because constraints to be imposed on the values are always linear. Most of the times, properties of stories that vary over time can be nonlinear functions of the time, but it is straightforward to approximate them by means of pieces in which the variation is considered to be linear. This approach simplifies the verification, as each piece is represented by a linear function and the transition from one piece to another occurs instantaneously. In this way, the satisfaction of constraints on the values can be verified by examining the ends of each linear piece.

The logic to be used in the specification of constraints has to be expressive enough to describe relevant constraints that should be satisfied by a (chapter of a) story with continuous and branching time. In this way, our logic uses the CTL path quantifiers **A** (meaning along **all** paths) and **E** (meaning that there **exists** at least one path) to speak about the satisfaction of a formula over the paths. To speak about the satisfaction of a formula over the states of a path, the following quantifiers are used: **U** (meaning that a formula holds **until** another formula holds), **G** (meaning **globally**, i.e. in all states of the path), and **F** (meaning **finally**, i.e. at some state in the path). As time is continuous, the number of states is infinite and we cannot speak about the next state of a certain state, then the CTL operator **X** is not used in our logic. In contrast, we might want to impose constraints on the end of the (chapter of the) story. We added then a new state operator “**end**” which speaks about the satisfaction of formulas at this specific moment. A well-formed temporal modal formula is recursively defined by means of the following grammar (in which a and b represent linear combinations of continuous properties, which have specific values at each moment of the story, α and β represent temporal modal formulas, and the semantics for the satisfaction of basic constraints and formulas with the logical connectives ($\wedge, \vee, \neg$, and $\Rightarrow$) is defined as usual):

$$\begin{aligned} \alpha, \beta \rightarrow & \text{basic_constr} \mid A F \alpha \mid A G \alpha \mid E F \alpha \mid E G \alpha \mid A \alpha \mid U \beta \\ & E \alpha \mid U \beta \mid A \text{end} \alpha \mid E \text{end} \alpha \mid \alpha \wedge \beta \mid \alpha \vee \beta \mid \alpha \Rightarrow \beta \mid \neg \alpha \end{aligned} \quad (1)$$

$$\text{basic_constr} \rightarrow a = b \mid a \neq b \mid a > b \mid a < b \mid a \leq b \mid a \geq b \quad (2)$$

3 Verification of Properties

In our prototype, constraints are specified by means of the temporal logic described in the previous section. To implement a verification procedure, we chose to use a constraint logic programming package, corresponding to a version of Prolog, enhanced with constraint solving. We implemented verification modules that can run either on SICStus Prolog [8] or on SWI Prolog [9]. As explained earlier, nondeterministic stories (or their

chapters) are modeled in the form of trees of events. The final dramatization is always one of the paths from the root to a leaf and depends on the user interaction and selections that occur at random. Figure 1 shows two alternative stories, note that in Story 2 the event after EV3 is EV7, instead of EV6.

Fig. 1. Trees of events that represent two different stories

Suppose that we want to know if, in least one path, there is a time in the future in which the level of *Fear* is higher than the level of *Joy* and, from this time on, in all future paths, the final level of *Joy* is higher than the final level of *Fear*. We could use then the following formula:

$$(EF (Fear > Joy \wedge (A end Joy > Fear))) \quad (3)$$

Fig. 2. Values of the properties in path EV1→EV3→EV6 in Story 1

Figure 2 shows that Story 1, because of the path EV1→EV3→EV6, satisfies this constraint as the condition (*Fear* > *Joy*) is true at a certain moment during EV3 and, in its only continuation, after EV6, the final values satisfy the condition (*Joy* > *Fear*).

Fig. 3. Values of the properties in path EV1→EV3→EV7 in Story 2

Figure 3 shows, on the contrary, that the sequence EV1→EV3→EV7 in Story 2 does not satisfy the constraint. In order to satisfy the formula, Story 2 should have another path where the condition (*Fear* > *Joy*) could be observed at some state.

In our verification procedure, the events are broken into small parts, in which the derivative of all continuous properties is constant and we transform formulas in order to eliminate negations and conditionals. This is easily done using De Morgan's law and the fact that the negation of a basic constraint is also a basic constraint (e.g. the negation of $A > B$ is $A \leq B$).

As we consider the execution of a sequence of events, we impose constraints on the values of each property at various points during the event, and we establish the relation between them in accordance with the corresponding derivatives, here the verification process can be broken in two parts: (i) path verification and (ii) verification of the states of a path. In the first case whenever we have a formula with quantifier **A**, we test all alternative paths to be followed and, whenever we have a formula with quantifier **E**, we test the possible subsequent paths using chronological backtracking and stop the verification when the formula is satisfied. In the second case, whenever we have a formula with quantifier **G**, we impose constraints on all ends of each part of the event. Whenever we have a formula with quantifier **F**, we check, for each part of the event, two possibilities: the formula either holds at a specific point in the interval or it holds after that interval. In the first case, formulas to be checked after that part might be identified and the process might have to go on by checking these formulas. The same occurs if we identify that formulas have to be checked in the subsequent events. Whenever we have a formula $\alpha \text{ U } \beta$ to be verified, we need to verify if α holds in all states until β holds. In order to do that, we mix the strategies used for the verification of formulas with quantifiers **G** and **F**. Constraints are imposed at the ends of intervals to force the satisfaction of α , but we also consider (by imposing constraints) that β might hold at a specific point of the interval, which makes the satisfaction of α unnecessary in the rest of the path.

Formulas with operator “**end**” are verified only after considering the execution of the last event of a path. In order to speed up the verification process, we simplify formulas whenever it is possible. The verification of a formula $(EF \ EG \ p)$, for instance, can be transformed into an equivalent formula $(E \ end \ p)$, which is easier to test.

As the trees of events corresponding to the chapters have a limited number of branches and a limited height, the implemented procedure has shown to be effective to verify constraints in real time, without hindering the overall storytelling process. The verification of the satisfaction of a formula like $(EF \ (\textit{Fear} > \textit{Joy} \wedge (A \ end \ \textit{Joy} > \textit{Fear})))$ by stories with around 58 events and 18 different paths lasts no longer than 141 milliseconds in a PC running a Windows version of SWI Prolog on a Core 2 Duo processor.

4 Concluding Remarks

The real-time verification of continuous events is a difficult task that did not receive much attention in the early ages of Artificial Intelligence, because of the difficulty in modeling the events and technological limitations [10]. This type of verification, however, is extremely important. In many real world situations, there are properties that continuously vary over time. Technological advances in the area of verification of real-time systems led to the appearance of different computational tools.

Hytech [11, 12] is a system that works with the satisfaction of temporal formulas in continuous time on hybrid timed automata. The specification of the properties is done using a logic derived from CTL, which treats continuous time with the aid of delay and clocks variables. Hytech also works with derivatives of continuous variables, but it uses Model Checking procedures [13] and approximations based on exact arithmetic to perform verifications. In our approach, as the numbers of events and branches to be considered are limited, we did not have to resort either to the discretization of time or to approximations, which could occasionally cause errors. We chose to perform symbolic executions using constraint logic programming. In addition, we worked with nondeterministic events and proposed a temporal logic that tries to take into account the needs of constraining nondeterministic stories. The prototype is able to verify temporal constraints in continuous time on nondeterministic stories. Our preliminary tests show that such a verification can occur in real time. This is important because the procedure can be applied to: (i) force the occurrence of patterns that are either typical of the genre or can be more appealing to the audience in general and (ii) adapt stories in accordance with the preferences of a specific intended audience.

In spite of being proposed to be incorporated to Logtell, our approach can be used to verify continuous properties of other interactive stories, including electronic RPGs (role playing games). Techniques can be used, for instance, to search for stories in a library of stories, using our logic to specify users' preferences. Moreover, the formalism proposed here may be useful in other systems that require the verification of properties that continuously vary over time.

References

1. Zagalo, N., Branco, V., Barker, A.: Emoção e Suspense no Storytelling Interactivo. In: Actas, Games2004 - Workshop Entretenimento Digital e Jogos Interactivos, Lisboa, Portugal, pp. 75–84 (2004) (best paper prize)
2. Guilherme da Silva, F.A., Ciarlini, A.E.M., Siqueira, S.W.M.: Nondeterministic planning for generating interactive plots. In: Kuri-Morales, A., Simari, G.R. (eds.) IBERAMIA 2010. LNCS, vol. 6433, pp. 133–143. Springer, Heidelberg (2010)
3. Ciarlini, A.: Geração interativa de enredos, DSc. Thesis, Departamento de Informática, PUC-Rio, Rio de Janeiro (1999)
4. Ciarlini, A.E.M., Pozzer, C.T., Furtado, A.L., Feijó, B.: A Logic-Based Tool for Interactive Generation and Dramatization of Stories. In: Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2005), Valencia, pp. 133–140 (June 2005)
5. Dória, T.R., Ciarlini, A.E.M., Andreatta, A.: A Nondeterministic Model for Controlling the Dramatization of Interactive Stories. In: Proceedings of the ACM MM 2008 - 2nd ACM Workshop on Story Representation, Mechanism and Context - SRMC 2008, Vancouver, Canada (2008)
6. Konur, S.: Real-time and probabilistic temporal logics: An overview. Cornell University Library (May 2010), <http://arxiv.org/abs/1005.3200>
7. Marriot, K., Stuckey, P.: Programming with Constraints. MIT Press, Cambridge (1998)
8. SICStus Prolog website, <http://www.sics.se/isl/sicstuswww/site/index.html>
9. SWI Prolog website, <http://www.swi-prolog.org/>

10. McDermott, D.: A Temporal Logic for Reasoning About Processes and Plans. *Cognitive Science* 6(2), 101–155 (1982)
11. Alur, R., Henzinger, T.A., Ho, P.-H.: Automatic Symbolic Verification of Embedded Systems. In: *Proceedings of the 14th Annual Real-Time Systems Symposium (RTSS)*, pp. 2–11. IEEE Computer Society Press, Los Alamitos (1993)
12. Henzinger, T.A., Ho, P.-H., Wong-Toi, H.: HYTECH: A Model Checker for Hybrid Systems. In: Grumberg, O. (ed.) *CAV 1997*. LNCS, vol. 1254, pp. 460–463. Springer, Heidelberg (1997)
13. McMillan, K.L.: *Symbolic Model Checking*. Kluwer Academic Publishers, USA (1993)