

Temporal Integrity Constraints in Interactive Multimedia Documents^{*}

I. Mirbel
Laboratoire I3S
UPRESA 6070 CNRS-UNSA
FRANCE
mirbel@mezzo.unice.fr

B. Pernici
Politecnico di Milano,

ITALY
pernici@elet.polimi.it

M. Vazirgiannis
Athens University of
Economics & Business,
HELLAS
mvazirg@aueb.gr

Abstract

When authoring multimedia scenarios, and in particular scenarios with user interaction, where the sequence and time of occurrence of interactions is not predefined, it is difficult to guarantee the consistency of the resulting scenarios. As a consequence, the execution of the scenario may result in unexpected behavior or inconsistent use of media. The present paper proposes to use temporal constraint verification techniques in a methodology for checking the temporal integrity of Interactive Multimedia Document scenarios at authoring time at various levels.

1. Introduction

An Interactive Multimedia Document (IMD) involves a variety of individual multimedia objects presented according to the IMD scenario. During rendering time, the multimedia objects that participate in the IMD are transformed, either spatially or temporally, in order to be presented according to author's requirements.

At authoring time, the author focuses in separate issues of the presentation at different times and the global consistency of the presentation may be lost. As a consequence, the rendering of the document may result in some of the presentation being lost (i.e., not present to the user) or some unexpected behavior during interactions. Moreover the multitude of events and the related interactions that may occur in an IMD session may create a lot of potential IMD evolution paths which are difficult to follow at authoring time, considering all potential implications. Therefore there is the necessity to provide a formal and sound background for building tools to support authors in designing consistent scenarios and to verify the consistency of existing ones.

In this paper we present a methodology for checking the temporal consistency of an IMD scenario at authoring time. The approach is based on previous work of some of the authors for modeling IMD scenarios [6]. The

methodology is based on the transformation of scenario specifications into networks representing the temporal constraints between multimedia objects and those concerning their evolution. Techniques from temporal reasoning [3] are applied in this context to analyze the resulting networks and examine their consistency and potential problems.

2. The scenario model

In this section we present briefly the scenario model that serves as the basis of the temporal constraint checking scheme. In the present paper, we are going to exploit an existing model for IMD [6]. According to that model an IMD consists of the definition of interaction between the user and the IMD (*external* interaction), the interaction between internal IMD entities (*internal* interaction) and the specification of media synchronization in the temporal and spatial domains.

The scenario consists of a set of autonomous functional units (*scenario tuples*) that include the triggering events (for starting and stopping the scenario tuple), the presentation actions to be carried out in the context for the scenario tuple, related synchronization events and possible constraints. More specifically, a scenario tuple has the form: (*Start_event*, *Stop_event*, *Action_list*, *Synch_events*) where

- *Start_event* represents the event expression that triggers the execution of the *Action_list*;
- *Stop_event* represents the event expression that terminates the execution of this tuple;
- *Action_list* represents the list of synchronized media-presentation actions that takes place when the scenario tuple becomes activated [6];
- *Synch_events* is a pair of events generated at the beginning and at the end of the current tuple execution. These events may be used for synchronization purposes.

^{*} Work partially supported by the TMR CHOROCHRONOS European project.

Hereafter we present a set of Temporal Access Control (TAC) operators [4] that represent the temporal composition of media objects, together with the causality of temporal relationships among presentation intervals. For more details refer to [6]. These operators correspond to the well-known TAC actions: start (>), stop(!), pause(||), resume(|>). For instance, assume a multimedia object A, then “A>” represents the start action of the multimedia instance. We have to illustrate the events arising from the temporal state changes of multimedia object, i.e. when the object A starts its presentation then the “A>” event is raised. Special attention should be paid to the event generated when the object finishes its execution (“A<”) and to distinguish this event from the TAC operation “!”.

Now we define a temporal composition scheme based on the one presented in [6], slightly altered though, to represent the temporal composition of media objects in the context of an IMD scenario. Let time be the length of a temporal interval between two events or TAC operations. Therefore: $\text{temporal_composition} ::= \Theta | (\text{object} (t_event | \text{TAC_operation}) \{ \{ \text{time object TAC_operation} \} \})$ where Θ is the spatiotemporal origin of an IMD session.

For instance, the expression: “A> 4 B! 0 C>” conveys “Start A, after 4 time units, stop B, and just after this start C”. Finally, we define the duration d_A of a multimedia object A as the temporal interval between the temporal events “A>” and “A<”. The above-defined set of operators is minimal.

3. Consistency of IMD scenarios

In the present section, we define how we consider temporal consistency in multimedia scenarios, at authoring time. As it has been introduced in other efforts [2] two categories of consistency are recognized: the *qualitative* and the *quantitative* one. The former is related to the validity of the combination of temporal relationships among media objects regardless of their duration, while the latter assumes the first and checks if the specific objects durations cause inconsistency problems. A scenario is said *temporally consistent* at authoring time if (i) the compositions among media objects, as required through the scenario tuple by the designer, are fulfilled all together, (ii) the sequences of TAC actions written in the scenario are legal. We introduce the concept of consistency at two levels: (i) consistency of multimedia objects, (ii) consistency of scenarios, where the internal representation of scenarios is based on temporal constraints.

3.1. Consistency of interactive multimedia objects

A time-dependent multimedia object may be in one of the following temporal states: *active*, *idle*, *suspended*. Moreover there is a set of TAC actions that may be applied that change the object status. When a TAC action

is applied to an object its temporal state changes. For instance when an object is *idle* and the action “>” is applied to, it falls into the *active* state. All the allowed transitions between states and the corresponding actions appear in Figure 1.

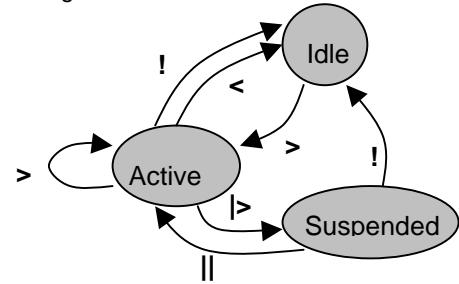


Figure 1. State transition diagram for time-dependent media object

Let us assume the expression: $A \text{ op } t \ A \ \text{op}$, where A is an actor, op is one of the operators: >, <, ||, |>, and t the time elapsed between the two operations. It is important to stress the fact that some combinations are not feasible (“illegal”) and some are feasible under some constraints. An exhaustive combination of TAC operations and their features, at authoring time, appear in Table 1.

Action	Action	Constraint	Action	Action	Constraint
>	>			>	$-d < t < 0$
>	!	$0 < t < d$!	$t > 0$
>		$0 < t < d$			Illegal
>	>	$-d < t < 0$		>	$t > 0$
!	>	$-d < t < 0$	>	>	
!	!	Illegal	>	!	$0 < t < r$
!		$t < 0$	>		$0 < t < r$
!	>	$-d < t < 0$	>	>	Illegal

d: duration of object presentation
r: remaining time of object presentation

Table 1. The constraints on pairs of consecutive TAC actions on the same object

3.2. Scenarios represented as networks of temporal constraints

As a basis for our methodology, we adopt an internal representation of scenarios based on constraint networks. The derivation of constraint networks is a non trivial problem, in particular when considering different possible user interactions during the scenario execution. We assume to derive from scenario descriptions a set of temporal constraint networks which represent the constraints between the different actions performed within a particular scenario execution, and representing also multimedia object internal properties, such as for instance the maximum allowed duration for an object execution. Constraints derived from scenario tuples are represented internally as *conjunctions of bounds of differences (bod)* between variables representing events [3]. Such

differences represent the minimum and maximum possible temporal distances between events, as derived from the specification of actions in tuples and from possible concatenations of tuple executions. For instance, if event a is at least 10 units of time before event b , but b occurs no later than 50 time units after a , we write $10 \leq b-a \leq 50$.

We build networks (or network fragments), where nodes represent events in tuples, and arcs express the minimum and maximum possible distances between two nodes n_1, n_2 in the form $b_1 \leq n_2 - n_1 \leq b_2$.

For each network (built according to the rules illustrated in next section), the consistency of the conjunction of *bod* is checked using temporal reasoning techniques such as the STP framework [3] in $O(n^3)$ time, where n is the number of nodes in the network. The algorithm checking the consistency produces also the *minimal network*, representing all possible solutions, i.e., possible assignments to the variables satisfying the constraints[3].

4. IMD scenario consistency checking

The consistency checking methodology may be summarized in the following.

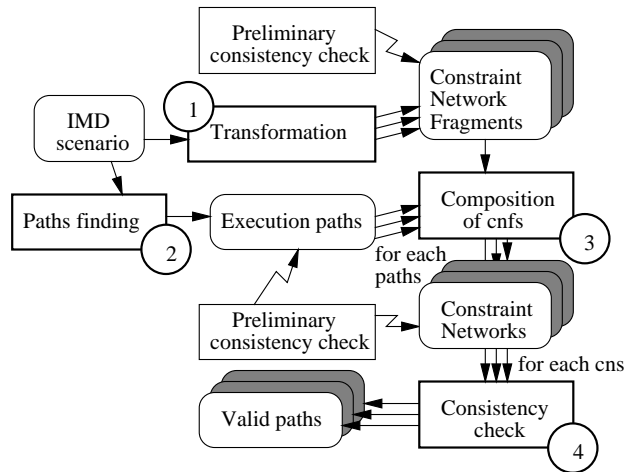


Figure 2. The architecture of the IMD scenario consistency checking methodology

The IMD scenario consists of a set of scenario tuples. Each of them is transformed into a *constraint network fragment (cnf)*. The *cnf* represents the temporal constraints between the start/stop events and the TAC actions/events in the action part of the scenario tuple. Due to the interactive nature of the IMD scenarios there is a lot of alternative sequences of events that will start/stop tuples. Each of these sequences (*paths*) is an *execution plan* for the IMD scenario. In the case of pre-orchestrated scenario, only one execution plan is possible, while in interactive scenarios several alternatives for execution have to be considered. Each execution plan defines a sequence of

tuples, which is mapped in a composition of the corresponding *cnfs*. The result is a composite *constraint network (cn)*. The temporal consistency of the IMD scenario is carried out in the following consecutive stages: (i) *Qualitative checks*, in which we verify in the *cnfs* and *cn*s if the sequences of TAC actions are legal according to the state diagram in Figure 1; (ii) *Quantitative checks*, in which we verify whether the temporal intervals between legal sequences of TAC actions are not creating inconsistencies, according to the criteria set in Table 1; (iii) *Temporal consistency check*, that verifies the global temporal consistency of the *cn*s. This verification is carried out exploiting the framework presented in [3].

The whole procedure is depicted in Figure 2. In the following sections, we elaborate on the phases of the methodology.

4.1. Step 1: network fragment design

This first step of the methodology aims at translating the scenario tuples into temporal networks, called *constraint network fragments (cnfs)*. These *cnfs* allow expressing the temporal constraints of the scenario through *bod* in order to check the temporal consistency of the scenario. The *cnfs* represent the constraints among events (tuple events and TAC events) in the tuple. As the check is based on *conjunctions* of *bod*, if there are alternatives in the tuple (for instance e_1 OR e_2 as stop event), different *cnfs* have to be provided, in order to check the consistency on each possible alternative. Let $NT = \langle N, C, A \rangle$ be the *cnf*, where:

- N is the set of the *nodes* of the network. A *node* represents a tuple event or a TAC event; Tuple events are taken from the Start event, Stop event and Synch event clauses. TAC events are taken from the Action list.
- A is the set of *arcs* between the nodes of the network. An *arc* represents a temporal constraint between two events. It can, for example, link the TAC start event and the TAC stop event of a video.
- C is the set of the *durations* associated with the arcs of the network. A *duration* allows one to express the minimum and the maximum durations associated with an arc. It is the time interval allowed between 2 events.

Thus for each tuple we have to create a *cnf* starting with the start event and continuing with the TAC events. After that we have to connect the stop events to the actions that have to be stopped. On *cnfs* (i.e. their corresponding set of *bod*), it is already interesting to do a preliminary consistency check, which verifies the constraints presented in Table 1 and is performed on the minimal network obtained through temporal reasoning.

4.2. Step 2: finding the paths in the scenario

The second phase of the methodology deals with the identification of possible event sequences resulting in

different order in the tuples execution. Depending on the sequence of occurring events, the different tuples of the scenario start and by this way define global *states* in the scenario. The goal of this first step of the methodology is to identify these different global *states*, with regards to the events (*transitions*) leading from one state to another. This step allows having a global idea of the possible execution plans (*paths*) of the scenario. For this purpose, we use an automaton, representing the different *states* of the scenario while the events allow *transitions* between states.

A *state* is defined as a set of executing (active) scenario tuples. A state change occurs when a new tuple is started. All the *start events* (Start event, Synch events) have to appear on at least one *transition*. A *transition* is caused by at least one of the *tuple events*. The resulting state-transition diagram must at least contain the *Idle state* (no tuple is running), from which all the paths start, through the *StartApp* event. If there are events that don't lead to a new state, then they have to be written on a transition ending in the pre-defined state called *Undefined state* (path raising a warning). Each *path* starting from the *Idle state* ending to the *Idle state* or *Undefined state*, represents an execution plan of the scenario.

4.3. Step 3: composition of *cnfs*

The goal of this step is to group the different *cnfs* with regard to the *paths*, to check its consistency. Composition is done by (i) merging the set of *bod* of the different fragments under consideration, (ii) adding additional information if possible. The composition of different *cnfs* to obtain a *cn* is done by following a *path*. Therefore, for each *path* deduced from step 2, we compose the *cnfs* representing the tuples involved in the *path*, to get the *cn* corresponding to the whole scenario execution.

Then, for each *cn*, arcs have to be added between events generated by TAC actions dealing with the same object. If the object is time-dependent, the arc durations have to fulfill the rules presented in Table 1. If the object is not time-dependent, the maximal bound of the arc duration has to be unbound.

Then, the events starting the different tuples have to be linked together through arcs indicating their sequence of arrival (*path* under consideration). Since the occurrence time of interactions is not predefined, the duration of these arcs have unknown maximum bound.

A tuple is considered as finished either (i) after stop event occurs (if it exists), (ii) when all the objects presented in this tuple have finished their presentation as defined in the action part. The objects which are not time-dependent stop only if explicitly required through a stop event. Therefore, at this state of the process, a first consistency check can be done with regards to the ending tuples: a warning is raised when an object does not explicitly finish in a given path.

Moreover, *cnfs* are evaluated with regards to the set of constraints representing the allowed sequences of TAC

actions for a given object as it has already been explained at the tuple level in step 1. This set of constraints (cf Table 1) has to be fulfilled for all the objects involved in the *cn*.

Also for each actor A, “A||” and “A|>” cannot appear if “A>” is not present: one cannot pause or resume an object which has not been started. Similarly, “A|>” can not appear without “A||”. These constraints are fulfilled through checks carried out for each object involved in the *cn*.

4.4. Step 4: consistency check

The consistency check is performed according to the STP framework [3] for each of the *cnfs* derived by the *cnf* composition step described above, which takes place for each alternative *path*. During the consistency check, for each *cn*, the corresponding minimal network [3] is derived.

From the consistency check, we can find paths considered as *consistent* or *inconsistent*. Inconsistent paths have to be corrected to avoid their occurrence at runtime.

The consistency check provides also constraints on unbound variables (unknown durations on arcs). It can for example derive the maximum duration necessary between 2 events to assume the consistency of the scenario for a given *path*. Therefore, the scenario can then be enriched with regards to this new constraint(s) in order to improve its robustness.

5. Verification of the approach

As it is evident the consistency checks for an IMD can be computationally very expensive due to the alternative paths that can occur because of internal and external interaction. In the following we attempt a worst case analysis of the approach. Though several heuristic methods and specific features may improve the overall cost. Let *t* be the number of tuples in the scenario, and let the worst case assumption that each tuple has a start event of the type ANY(*k*, *e*₁ .. *e*_{*m*}). In the following we evaluate the time complexity of the steps of the methodology.

Step 1 (Transformation of tuples into *cnfs*): As we mentioned before we assume the worst case in which every tuple has a start event of the form: ANY(*k*, *e*₁ .. *e*_{*m*}). This means that for each tuple we will have $\binom{m}{k}$ different *cnfs* produced. Then checking the consistency of a scenario tuple means: To carry out the checks related to the constraints of Table 1. The former step means that for each pair of actions appearing in the *cnf* we have to verify the constraints of Table 1 which is of linear complexity ($O(n)$); To carry out the consistency check of the *cnfs* according to [3]. It means to verify $\binom{m}{k} * t$ different *cnfs*. The complexity of this task is $O(n^3)$ where *n* is the number of nodes in the *cnf*. Thus the overall worst case complexity is

$$\binom{m}{k} * t * O(n^3).$$

Step 2 (Finding the different paths): This problem can be reduced to all the possible events sequences that may occur. In the general case we should consider all the permutations of the start events. So assuming a number of t tuples we have t start events and therefore all the possible permutations are $t!$. Of course this can be optimized to a great extent since not all the sequences of events are allowed and also always the StartApp event is the first in each permutation. Thus the result of this stage is the set of *legal* permutations

Step 3&4 (Composition & evaluation of the cns): Assuming that we found the legal permutations of events for our document we carry out the construction of the alternative *cns*. The complexity of the above algorithm is based on the number of alternative *paths* (and hence *cns*) which in the worst case is $O(t!)$. Since the number of different alternative *cns* because of the complex start events is $O\left(\binom{m}{k}^t\right)$ and the number of nodes in the *cn* which is n , then the overall worst case complexity is:

$$O(t! * \binom{m}{k}^t * n^3).$$

Of course the vast majority of the paths are not legal and we are working on heuristic methods to reject them in early stages of the methodology, so that the approach is computationally feasible.

6. Related work

In [2], a synchronization model for the formal description of multimedia documents is presented, where temporal synchronization is achieved through a set of library functions for which there is no indication if they can cover all the potential temporal composition requirements, especially in the case of composing more than two objects. We alternatively propose a minimal set of TAC operators which can represent any temporal composition adopting vacant temporal intervals among the actions. The TAC functionality is not fully covered. Our concept of consistency is richer since we define consistency as an integration of : (i) specific qualitative constraints on the sequence of TAC actions, (ii) quantitative constraints on the temporal differences between actions on the same object, and (iii) on global qualitative and quantitative check. Our approach deals also with reachability, and provides a minimal network and constraints derivation on unknown durations.

In [1], a temporal constraint satisfaction algorithm is presented, but without a clearly defined temporal composition model. In addition, in our approach we introduce the aspect of TAC action sequence consistency related to an object temporal state.

In [5] an approach is presented that addresses the key issue of providing flexible multimedia presentation with user participation. This effort has a sound theoretical

background from the Petri Nets theory that has been used often to model temporal synchronization of multimedia objects. The interaction space is different to the interaction we handle in our model: it refers to the ability of the user to react with the presentation as one single object and to change global features of the presentation such as duration. The sequence of media objects is generally fixed (with the exception of skipping the remaining time of it) unlikely to our approach. There is no provision for interactions with the objects that participate in the presentation.

7. Concluding remarks

In this paper we proposed a methodology for checking the temporal consistency of IMD scenarios. The approach is based on the description of scenarios adopting the modeling language proposed in [6]. The consistency check on a scenario consists of a set of distinct methodological steps which allow the verification of the consistency of the behavior of single objects evolving in the scenario, as well as portions of the scenario or its complete possible evolution during execution. Rules for building the constraint networks corresponding to the fundamental functional units of the scenario are defined and, together with temporal constraint verification techniques introduced in [3], provide the basis for checking the consistency of the scenario. Moreover, the temporal consistency check is robust since it is based on well-accepted and thoroughly studied research results [3].

8. References

- [1] M.C. Buchanan, P.T. Zellweger, "Automatically generating consistent schedules for multimedia documents", *ACM-Multimedia Systems Journal*, vol. 1 (2), pp. 55-67.
- [2] J.P. Courtiat, R.C. De Oliveira, "Proving Temporal Consistency in a New Multimedia Synchronization Model", *ACM Multimedia Conference*.1996.
- [3] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks", *Artificial Intelligence*, Vol. 49, 1991, pp. 61-95.
- [4] T.D.C. Little, A. Ghafoor, "Interval-Based Conceptual Models for Time-Dependent Multimedia Data", *IEEE Trans. on Data and Knowledge Engineering*, Vol. 5, No. 4, August 93.
- [5] B. Prabhakaran and S.V. Raghavan "Synchronization Models for Multimedia Presentation With User Participation", *ACM Journal of Multimedia Systems*, vol.2, no. 2, August 1994.
- [6] M. Vazirgiannis, Y. Theodoridis, and T. Sellis, "Spatio-Temporal Composition and Indexing for Large Multimedia Applications", *ACM Multimedia Systems Journal*, 1998.