Multimedia Systems 10: 72–82 (2004)
Digital Object Identifier (DOI) 10.1007/s00530-004-0138-3

Multimedia Systems

Retrieval in multimedia presentations

Augusto Celentano[1], **Ombretta Gaggi**[1], **Maria Luisa Sapino**[2]

[1] Dipartimento di Informatica, Università Ca' Foscari di Venezia, Via Torino 155, 30172 Mestre (VE), Italy
e-mail: {auce,ogaggi}@dsi.unive.it
[2] Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10149 Torino, Italy
e-mail: mlsapino@di.unito.it

Abstract. In this paper we discuss issues concerning the consistent retrieval of parts of multimedia presentations from multimedia repositories. We introduce a class of multimedia presentations made of independent and synchronized media and discuss retrieval requirements of presentation fragments. Then we discuss a retrieval model capable of reconstructing the fragments of a presentation from the atomic components returned by the execution of queries to multimedia presentation repositories. The retrieval model is based on an automaton that formally describes the presentation states entered by the events that trigger media playback. Retrieving a consistent fragment corresponds to building a new presentation with all the media related to the retrieved ones, with their original structural and synchronization relationships.

Keywords: Information retrieval – Multimedia presentations – Media synchronization – Event-based modeling – Finite-state machine

1 Introduction

A multimedia presentation can be modelled as the composition of one or more continuous media files, such as video or audio streams, that are presented to a user together with static documents, such as images or text pages, that are displayed in synchrony with them. From the user point of view the different documents constitute a whole presentation ruled by the synchronization relationships among the component media items.

News-oriented applications, such as news-on-demand and Web advertising, and virtual museums are good representatives of such presentations. News is presented through audio or video clips, and images and texts are displayed according to the particular subject the speaker is talking about. Artworks in a virtual museum are illustrated by multimedia presentations made of text, images, and videos, possibly commented by voice narrations and music.

In both cases the whole presentation is a collection of sections each of which is devoted to a single article or to a single artwork according to a recurring schema. Sections can be grouped into larger entities (e.g., museum rooms), giving the whole presentation a structure that helps the user navigate its contents.

Information retrieval in multimedia presentations can be defined as the task of retrieving items of different media types satisfying some query parameters from a repository of multimedia presentations and returning the user all the presentation fragments in which the retrieved items were originally located. With respect to multimedia information retrieval as defined, e.g., in [8], this task has a number of peculiarities.

The user queries a repository of presentations, expecting (fragments of) presentations as results, but the query processor plausibly retrieves the component media items, which must be integrated with other components in order to return to the user coherent and understandable segments of the original presentation. More specifically, the query result extent must be identified according to a context that takes into account the presentation structure and dynamics.

We do not address here issues related to query processing and retrieval of multimedia data. We simply assume that the user formulates a query with proper parameters that identify media items whose content is relevant, belonging to one or more presentations.

The retrieval system executes the query and returns a (ranked) list of references to media items, possibly of different media types. Generally speaking, each returned reference is a pair $\langle Mid, P\rangle$, where Mid is the media item identifier, i.e., a reference to a descriptor containing a file locator or a URL, a media type, and possibly other information, and P is a reference to the presentation in which it is located. If a media item belongs to several presentations, several pairs are returned with different values of P. Our goal is to reconstruct, for each pair, a coherent and complete fragment of the presentation P to be displayed as result of the query, identifying all objects that belong to the fragment of P that contains the media referenced by Mid. More precisely, a coherent and complete fragment is a portion of the original presentation containing a subset of media components, having the same behavior of the original presentation, i.e., the same reactions to internal and external events. Therefore we must

- Identify the missing media items to be supplied and
- Identify the fragment scope, i.e., the range of the original presentation covered by the fragment.

We assume also that each presentation is described by a schema according to a model able to reveal the relationships among the media components, i.e., the static organization of a presentation that relates media to one another, and the temporal aspects of its behavior, which describe how the different segments evolve.

The paper is organized as follows. After reviewing the related works in Sect. 2, we discuss in Sect. 3 a synchronization model for multimedia presentations and define in Sect. 4 a finite-state automaton describing the presentation dynamic behavior. Section 5 presents the algorithm that builds the presentation fragments associated with retrieved media. Section 6 discusses variants in the identification of meaningful presentation fragments, and Sect. 7 draws conclusions.

2 Related work

Multimedia information retrieval is an issue widely investigated in the literature, but only few works focus on the issues of querying and browsing excerpts of multimedia presentations.

In [1] Adali et al. present an algebra for creating and querying interactive multimedia presentation databases. The authors model a multimedia presentation as a tree whose branches represent different playouts due to user interactions and whose nodes contain the set of active objects and the spatiotemporal constraints between them. The algebra defines operators to select and present paths of the presentation the user is interested in; this allows users to create new presentations, merging parts of existing ones.

In [11] the authors propose an integrated query and navigation model that provides facilities for query writing activities to users that have little familiarity with database contents and schemas. Unlike the approach we propose, the authors are not interested in modelling the temporal behavior of the retrieved results but only in hiding problems related to the querying of heterogeneous data in a distributed environment. They provide an interactive user interface that permits both the use of conventional declarative queries and navigational techniques.

The same approach is used by Chiaramella in [5]. He presents a model that fully integrates browsing and querying of multimedia data capabilities, giving particular emphasis to structured information, but problems related to temporal synchronization in composite documents are not discussed. The model contains two components: the hypermedia component and the information retrieval (IR) component. The hypermedia component allows the user to formulate queries (and browse the results) that explicitly reference both *content knowledge*, i.e., the semantic content of atomic data, and *structural knowledge*, i.e., the types of documents and cross-reference links between documents. The IR component contains information about the resolution of the queries.

GVISUAL [9,10] is a graphical query language for formulating queries on multimedia presentations based on content information. Multimedia presentations are modelled as directed acyclic graphs. A query consists of a head with its name and parameters, and of a body. The query body contains an iconized representation of the objects involved and a *conditions box* with the conditions required. An object's icon contains a name and can be nested to represent a composition relationship. Media items can be connected by edges that represent temporal operators. The authors aim at querying not only the information stored in individual streams but also the *flow of information* that represents the theme of a multimedia presentation.

Other works are more oriented toward building new presentations out of existing multimedia material, thus covering only partially the issues related to the retrieval of presentations. Delaunay^{MM} [6] is a visual tool for querying and presenting multimedia data stored in distributed data repositories, including the Web. It allows the user to define the layout of a virtual document, by arranging graphical icons inside style sheets, and of document content, through ad hoc queries to multiple repositories. Also in this proposed paper, the authors did not address any solution for the specification of temporal synchronization among the objects.

In [2,3] SQL+D, an extension of the SQL language, is presented that allows users to include spatial and temporal specifications in an SQL query. The user can select data needed and specify how it should be displayed. The clause DISPLAY-WITH describes the screen areas, called *panels*, in which the retrieved items are placed, and the clauses AUTOSKIP and SHOW define, respectively, the duration of the media items' playback and the temporal constraints among the objects of a presentation.

This survey would not be complete without briefly mentioning SMIL, *Synchronized Multimedia Integration Language*, a W3C standard markup language that defines tags for presenting multimedia objects in a coordinated way [13]. Synchronization is achieved essentially through the composition of two tags: <seq>, to render two or more objects sequentially, and <par>, to play them in parallel. The tag <excl> is used to model some user interaction. It provides a list of child elements of which only one may play at any given time. The screen is divided into regions in which multimedia objects are placed.

3 Synchronization description in multimedia presentations

For modelling multimedia presentations we refer to a synchronization model previously defined in [4,7]. The model is oriented toward Web-based applications and describes the hierarchical structure of a presentation and the temporal behavior of its components in terms of synchronization relationships among media items.

Other models could describe as well multimedia presentations, e.g., SMIL, which is a standard language with rich synchronization and layout specification constructs, assuring portability across several platforms and players. In a retrieval scenario like the one we face, however, it has a number of drawbacks. SMIL does not provide a global, modular, or hierarchical structure of the whole presentation, the only relationships among media being induced by temporal synchronization. Thus the identification of the retrieved fragment scope and its isolation can be harder. In effect, SMIL is more suitable as an execution language than as a specification language for multimedia presentations. For example, the tags <par> and <seq> can be interchanged with suitable values of attributes to obtain the same presentation behavior. Therefore,

building a timeline representation, to be used to retrieve a coherent fragment for the user, could require anticipating the real execution of the original presentation rather than computing it from the SMIL specification.

According to our model, a multimedia document is a collection of *modules*. A module is a container of different kinds of continuous and static media objects. Static objects like texts or images are referred to as *pages*. Continuous media objects are hierarchically organized: a video or an audio stream is divided into *stories*, which are made up of *clips*, which are media files played continuously. A clip can be divided into *scenes*, which are temporal intervals inside the time span of an object. The scenes are the lowest-level media units with which synchronization events are associated. As a clip plays, the scenes are played in sequence, thus generating events that cause other media to be displayed, or played, or terminated (see Table 1 for a precise definition of each component).

Each medium requires some device to be rendered or played. A media object can use a new browser's window, a frame inside the window, an audio channel, or a combination of audio and video resources (as required by a video file with integrated audio). The model calls a *channel* such a virtual device, which is used by the medium for all the duration of its playback. A channel is *busy* if an active object is using it; otherwise it is *free*. Free channels may hold some content, for example, the last frame of a video clip or a page that has not yet been replaced by a new one.

Synchronization among different components of a multimedia document is described by five synchronization primitives, which define object reactions to some events. Two relations describe the sequential and the parallel composition of two media; the others model objects' reactions to user interactions.

The relationship "*a plays with b*", written $a \Leftrightarrow b$, models the parallel composition of media objects a and b: it states that if one of the two objects is activated by the user or some other event, the two objects play together. Object a acts as a "master" in this relation: as soon as it ends, object b terminates, too, if it is still active. The relation "*plays with*" therefore has a symmetric behavior with respect to media activation but is asymmetric with respect to media termination since only the natural end of object a causes object b to stop.

The relationship "*a activates b*", written $a \Rightarrow b$, models the sequential composition of two objects: when object a naturally ends, object b begins its playback. These two relationships are similar to the tags <par> and <seq> of SMIL [13], but some differences exist that are detailed in [7]. In particular, we distinguish between *internal* events, which are generated by some components of the presentation and *external* events, which are generated by the user, separating the *natural termination* of an object, occurring when it reaches its ending point, from its *forced end*, occurring when the user stops it. In the relationship $a \Rightarrow b$, if the user stops object a, object b is not activated, and in the same situation, if the relationship $a \Leftrightarrow b$ holds, object b is not terminated.

The relationship "*a is replaced by b*", denoted by $a \rightleftharpoons b$, is mainly used with static objects whose time duration is potentially infinite. It states that, when object b starts, it forces a to end, so its channel is released and can be used by b.

Two other relationships model object reactions to user interactions. We describe them for completeness even if they have a less important role in the context of multimedia presentation retrieval. The relation "*a terminates b*", written $a \Downarrow b$, terminates two objects at the same time as a consequence of the forced termination of object a by the user or some other external event. The relation "*a has priority over b with behavior* α", symbolically written $a \overset{\alpha}{>} b$, means that object b is paused (if $\alpha = p$) or stopped (if $\alpha = s$) when object a is activated; a is supposed to be the target of a hyperlink that moves the user focus from the current document (b) to another document or to another presentation. When object a comes to the end, object b is resumed, if it had been paused. The reader is referred to [4,7] for a discussion of details and motivations about this synchronization model.

We introduce a working example in order to show the model application to a nontrivial case. Our example concerns a collection of multimedia presentations about classical music masterworks following a constant structure and containing information about musicians' life, the historical period, and their masterworks. For each masterwork, the presentation contains a description of its musical structure in the form of a graphical score and text comments and a criticism in the form of written text that accompanies the music.

Such a collection was published a few years ago in Italy as a set of CD-ROMs [12]. Figure 1 shows a screen shot from the Beethoven Symphony No. 6 "Pastorale" CD-ROM, featuring a guide to listening.

The overall structure of the work is shown in the lower part of the screen. As the music plays, a bar moves showing the current position in the score. Text commentary that changes as the symphony plays helps the user to understand the different themes, the composer's style, the orchestral arrangement, and so on. This section of the presentation is divided into several *modules* with a constant structure: Fig. 2 illustrates an excerpt of the structure of the first *module*, which plays the symphony's first movement. For the sake of readability it contains only the most relevant relationships.

The soundtrack of the first movement (and therefore of the first *module*) is divided into *stories* ($story_1$, $story_2$, ...), *clips* (for the first story, $c_{1,1}$, $c_{1,2}$, ...), and *scenes* (for the first clip,

Table 1. Definition of the model components

Module	a collection of media items, both continuous and noncontinuous, related to a common topic
Story	a set of continuous media items that constitutes the "master" media stream of the module content
Clip	an atomic continuous media stream; a story is a sequence of clips played continuously
Scene	a time interval in a clip's playback
Page	a static document that, once displayed, stays on the screen until the user or some external event stops it. Pages cannot be paused
Channel	a (virtual) display or playback device like a window, a frame, an audio device, or an application program capable of playing a media file that can be used by one medium at a time

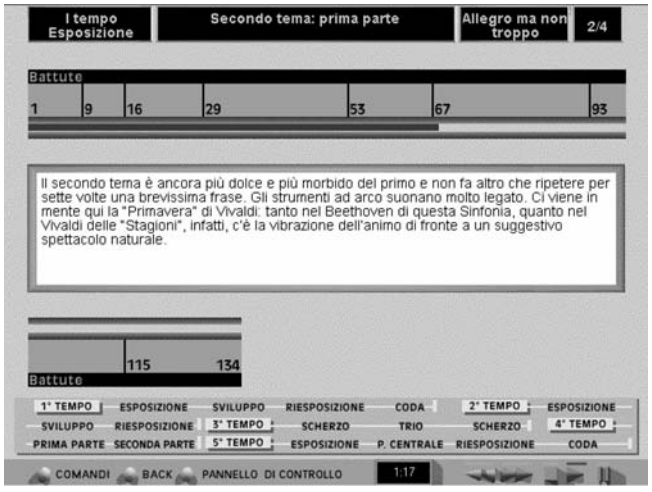

Fig. 1. Guide to listening to the *Pastorale* symphony

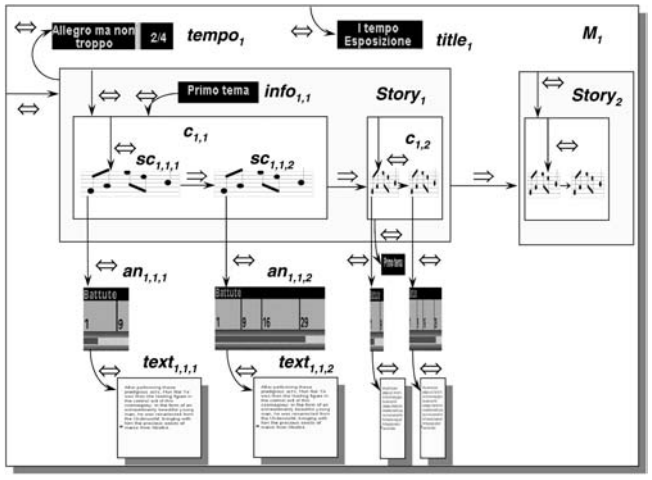

Fig. 2. Synchronization schema for the presentation of the guide to listening to music

$sc_{1,1,1}$, $sc_{1,1,2}$, ...). Such a hierarchical structure corresponds to a musical structure of sections, themes, and parts of the symphony movement that are commented by different text *pages* displayed at proper times during play.

We denote *modules* with uppercase letters $M_1, M_2, \ldots$ and other components with lowercase letters and words. The overall synchronization schema of the symphony's first movement is described by the following relations, where I, K, and H denote the cardinality of the stories, clips, and scenes, respectively:

$$\begin{array}{lll} M_1 & \Leftrightarrow story_1 & \\ story_i & \Rightarrow story_{i+1} & \forall i \;\; i < I \\ story_i & \Leftrightarrow c_{i,1} & \forall i \\ c_{i,k} & \Rightarrow c_{i,k+1} & \forall i,k \;\; k < K_i \\ c_{i,k} & \Leftrightarrow sc_{i,k,1} & \forall i,k \\ sc_{i,k,h} & \Rightarrow sc_{i,k,h+1} & \forall i,k,h \;\; h < H_{i,k}. \end{array}$$

The progress bar is an animation also divided into *clips* and *scenes* (for the first clip of the animation, $an_{1,1}$, $an_{1,2}$, ...). Textual information is organized into *pages*.

Each *module* corresponds to a movement of the symphony. This information is displayed in a header (i.e., a page in the model terminology), $title_1$ for the first module, which is visible for the entire duration of the module. This behavior is modelled by the relation $M_1 \Leftrightarrow title_1$.

Each movement of the symphony is divided into sections that correspond to the *stories*. The relation $story_i \Leftrightarrow tempo_i$ displays the current tempo of the music. Two other information items help the user to understand the current location in the score: a short title is changed each time a new theme (i.e., a clip) begins and the bar progresses in synchrony with the music. The relations $c_{i,k} \Leftrightarrow info_{i,k}$ and $sc_{i,k,h} \Leftrightarrow an_{i,k,h}$ display each title for the duration of the corresponding *clip* and start the animations together with the *scenes* of the soundtrack. The relations $an_{i,k,h} \Leftrightarrow text_{i,k,h}$ complete the presentation modelling by displaying some text comments to help the user interpret the music execution.

Let us suppose that a collection of presentations about different symphonies and musicians are stored in a multimedia repository. Each presentation can be modelled with a structure similar to the one discussed above. In this scenario, if we want to retrieve all the music passages where strings play in a *crescendo* style, we can ask for such a text description in the guide to listening. The retrieval system will return a set of media identifiers, but the user should receive composite parts of the original presentation in order to understand the query result. Browsing has similar problems: once the user has identified relevant information, she may need to broaden the scope and access more complex information in order to understand it.

This synchronization model allows the system to fill the gap between the retrieval of separate media items and the presentation of the composite document. In some cases a presentation is a single file or a set of parallel *tracks*; therefore, a reference from the components to the time interval in which they are played is all that is needed to identify the relevant context. In other cases the presentation is made up of separate parallel files linearly integrated by a player; therefore, something similar to a time stamp could solve the problem. In the general case, however, the temporal dependencies among the media items could be more complex than a simple linear time ordering, and several different files could participate at different times to build the presentation behavior. For example, if we consider a distributed environment like the World Wide Web as the scenario of our discussion, there are at least two issues that make a model based on linearly integrated media unrealistic:

- The media are delivered by a server according to a composite document structure that spans several files and is known as they are delivered to the client.
- A WWW document usually has links that allow the user to explore its structure in a nonlinear way, and the user can also interact by backtracking or reloading the documents as they are displayed. Therefore, a synchronization model more complex than a simple linearly timed composition is needed.

4 A formal approach to multimedia presentation modelling

In the previous section we introduced the relevant components of our model. In particular, we showed that the playout

of a multimedia presentation can be described in terms of the media items involved, the channels used for media playback, the events that cause media to start, pause, and end, and the synchronization relationships that describe dynamic media behavior. Such elements provide an intentional representation of the evolution of the presentation in time. A formal definition is as follows:

Definition 1 (Presentation). A multimedia presentation is a 4–*tuple* $P = \langle \mathcal{MI}, \mathcal{CH}, \mathcal{E}, \mathcal{SR} \rangle$ where

- $\mathcal{MI}$ is a set of media items $\{m_0, m_1, \ldots, m_n\}$;
- $\mathcal{CH}$ is a set of presentation channels $\{c_0, c_1, \ldots, c_h\}$;
- $\mathcal{E}$ is a set of events $\{e_0, e_1, \ldots, e_k\}$, $e_i \in \mathcal{ET} \times \mathcal{MI}$, where $\mathcal{ET} = \{$*start*, *end*, *pause*, *stop*$\}$ is the set of event types;
- $\mathcal{SR}$ is a set of synchronization relationships $\{sr_0, sr_1, \ldots, sr_l\}$, $sr_i \in \mathcal{SP} \times \mathcal{MI} \times \mathcal{MI}$, and $\mathcal{SP} = \{\Leftrightarrow, \Rightarrow, \Downarrow, \rightleftharpoons, \overset{p}{>}, \overset{s}{>}\}$ is the set of synchronization primitives introduced in Sect. 3.

For clarity we shall denote event instances by $e(m)$, where e is an event type and m a media item, and synchronization relationship instances by the symbolic infix notation used in Sect. 3, e.g., $m_1 \Leftrightarrow m_2$ to denote the relationship $(\Leftrightarrow, m_1, m_2)$.

Two mappings describe channel occupation:

- *channel* : $\mathcal{MI} \rightarrow \mathcal{CH}$, which, given a media object, returns the associated channel, and
- *isUsed* : $\mathcal{CH} \rightarrow \mathcal{MI} \cup \{\_\}$, which returns, for every channel, the media item that occupies it at a given time instant. The underscore symbol _ denotes the absence of a media item and is the value associated with free channels.

At any time instant, the presentation is completely described by the set of media that are active at that time and the corresponding channel occupation. This information is captured by the notion of the *state* of the presentation. Before the presentation starts, no media item is active; thus all channels are free. When an event occurs, the state of the presentation changes: some items that were not active become active, some active items end, and other items could be forced to stop due to some interruption.

Definition 2 (State). The state of a multimedia presentation is a triple $S = \langle \mathcal{AM}, \mathcal{FM}, \mathcal{UC} \rangle$, where $\mathcal{AM}$ is the set of active media, $\mathcal{FM}$ is the set of *frozen* (i.e., paused) media, and $\mathcal{UC}$ is the set of pairs $\langle c, m \rangle$, where c is a channel and m is the media item that occupies it as defined by the mapping *isUsed* above. For clarity, in the following discussion, we shall refer to the association between a channel x and an active media y with the functional notation $isUsed(x) = y$.

The set $\mathcal{S}$ of the possible states for a presentation is finite, since both the set of media items and the set of channels are finite. The set of frozen media items $\mathcal{FM}$ is a subset of the set of active media $\mathcal{AM}$, since a frozen media item still holds a channel, e.g., for the last frame of a video. The channel can be used only by other media items for which a relation $\overset{p}{>}$ with the paused object exists.

If we observe the system in time, the only relevant time instants are the *observable time instants*, i.e., the time instants in which an event occurs, or the effects of an event are assessed and perceivable. Indeed, these are the time instants in which something in the state of the presentation might change, as a consequence of the occurred event.

The state of a multimedia presentation is thus a function of *observable time instants*. We assume that at any observable time instant at most one "master event" occurs, i.e., either a media item is activated, paused or stopped by the user, or a media item naturally ends. Anyway, at the same time instant other items may be activated, paused or stopped, according to the synchronization relationships characterizing the presentation.

It is important to notice that, given the set of synchronization primitives associated with the presentation, the effects of any observable event are deterministically implied by (i) the set of currently active media, (ii) the set of frozen media, (iii) the current channel occupation, and (iv) the occurred event. Thus, all the possible evolutions in time of a multimedia presentation P can be described by a finite-state automaton, defined as follows.

Definition 3 (Automaton). Let $P = \langle \mathcal{MI}, \mathcal{CH}, \mathcal{E}, \mathcal{SR} \rangle$ be any presentation. Its associated finite-state automaton is the 5–*tuple* $AUT(P) = \langle \mathcal{S}, \mathcal{E}, s_0, $ *next*, *Final*$\rangle$, where

- $\mathcal{S}$ is the set of possible states for the presentation P;
- $\mathcal{E}$ is the set of possible event instances in the form $start(m)$, $end(m)$, $pause(m)$ and $stop(m)$, $m \in \mathcal{MI}$;
- s_0, the initial state, is $\langle \mathcal{AM}_0, \mathcal{FM}_0, isUsed_0 \rangle$, where $\mathcal{AM}_0 = \emptyset$, $\mathcal{FM}_0 = \emptyset$, and $isUsed_0(c) = \_$, for all $c \in \mathcal{CH}$;
- The transition function $next : \mathcal{S} \times \mathcal{E} \rightarrow \mathcal{S}$ is the mapping that deterministically associates any state s to the state s' in which s is transformed by an event $e \in \mathcal{E}$;
- $Final$, the set of states that correspond to the end of the presentation playback. Details on the set $Final$ will be given in Sect. 6.

The definition of the function *next* requires some extra notions, that we introduce in the following. First, we need to be able to capture all the consequences that an event might have on the presentation, given the current state, according to the synchronization rules. Starting and ending events might indeed activate a cascade of simultaneous media activations or stops. The following notions of *closure of an item*, with respect to some category of rules, capture these effects.

Definition 4 ($(\Leftrightarrow)Closure$). Let a be a media item in $\mathcal{MI}$. The $(\Leftrightarrow)Closure$ of a is the set inductively defined as follows:

- $a \in (\Leftrightarrow)Closure(a)$;
- for any item $b \in \mathcal{MI}$, if $b \in (\Leftrightarrow)Closure(a)$ and $b \Leftrightarrow c \in \mathcal{SR}$, then $c \in (\Leftrightarrow)Closure(a)$.

$(\Leftrightarrow)Closure(a)$ captures the asymmetry[1] and the transitivity of the *plays with* relation. In particular, we shall use the set $(\Leftrightarrow)Closure(a)$ to take care of the asymmetric consequences of the end of a, in the presence of *plays with* rules.

Definition 5 ($(\Leftrightarrow)1step$). Let a be a data item in $\mathcal{MI}$. The $(\Leftrightarrow)1step$ of a is the set defined as follows:

- For any item $b \in \mathcal{MI}$, if $a \Leftrightarrow b \in \mathcal{SR}$ or $b \Leftrightarrow a \in \mathcal{SR}$, then $b \in (\Leftrightarrow)1step(a)$;

[1] As defined in Sect. 3

$(\Leftrightarrow)1step(a)$ captures the symmetry of the *plays with* relation, as defined in Sect. 3. In particular, we will use the set $(\Leftrightarrow)1step(a)$ to take care of the symmetric consequences of the start of a, in the presence of *plays with* rules.

Definition 6 ($(\Downarrow)Closure$). Let a be a data item in $\mathcal{MI}$. The $(\Downarrow)Closure$ of a is the set inductively defined as follows:

- $a \in (\Downarrow)Closure(a)$;
- for any item $b \in \mathcal{MI}$, if $b \in (\Downarrow)Closure(a)$ and $b \Downarrow c \in \mathcal{SR}$, then $c \in (\Downarrow)Closure(a)$;

Intuitively $(\Downarrow)Closure(a)$ contains all the media items that, according to the *terminates* relation, are required to stop if a terminates.

The following two definitions take care of the hierarchical structure of the media items by relating a story with its component clips, and a clip with its component scenes.

Definition 7 ($ComponentOf$). Let a and b be data items in $\mathcal{MI}$. $ComponentOf(a, b)$ evaluates to true if and only if at least one of the following conditions holds:

- a is a story (clip) and b is a clip (resp. scene) and $a \Leftrightarrow b \in \mathcal{SR}$;
- a is a story (clip) and b is a clip (scene) and n clips (scenes) $x_1 \ldots x_n$ exist such that $a \Leftrightarrow x_1$, $x_i \Rightarrow x_{i+1}$ for all $i = 1 \ldots n-1$ and $x_n \Rightarrow b$.

Definition 8 ($IsLast$). Let a and b data items in $\mathcal{MI}$. $IsLast(a, b)$ evaluates to true if and only if both the following conditions hold:

- $ComponentOf(a, b) = True$;
- For any item $c \in \mathcal{MI}$, if $b \Rightarrow c \in \mathcal{SR}$, then $ComponentOf(a, c) = False$.

Definition 8 lets us identify the last clip (scene) of a story (clip). When it naturally ends, the whole story (clip) ends.

The state transformation caused by an event might be a complex procedure. For the sake of readability, we introduce two parameterized functions that take care of the two most complex actions, i.e., activating a nonactive media item and restarting a paused media item. Then we shall define the state transition function algorithm.

The ACTIVATE function activates object x: it starts x if it is not active or resumes x if it is paused. Then it controls which other objects must be activated by the beginning of x, i.e., the objects that belong to $(\Leftrightarrow)Closure(x)$. Step by step, the function activates all media items $y \in (\Leftrightarrow)1step(y)$ if their channel is free. Otherwise the function controls if relationships of type $\rightleftharpoons$ or $\overset{\alpha}{>}$ exist. Then it stops the media that use the same channel as y and starts y.

The RESTART function resumes object x from the pause. It resumes also all other objects y such that $y \in (\Leftrightarrow)Closure(x)$.

ACTIVATE (x: media object; $\mathcal{AM}$, Δ^+, Δ^-, Δ_F^+: set of media items; oldUsed, newUsed: channel-media mapping)

```
// x: media item to be activated,
// AM: set of currently active media,
// Δ+: set of media items to be added to the set of
//       active media,
// Δ−: set of media items to be removed from the set
//       of active media,
// Δ+F: set of media items to be added to the set of
//       frozen media,
// oldUsed: channel occupation function when
//            ACTIVATE is called,
// newUsed: channel occupation function after
//            ACTIVATE is executed
begin
                        // the set of items to be activated
  Set = {x};            // after x's start
  while Set ≠ ∅ do
   begin
     pick any y from Set;
     if oldUsed(channel(y)) = _ then
      begin             // the channel of y is free, start y
        Δ+ = Δ+ ∪ {y}; newUsed(channel(y)) = y
      end
     else
      begin             // check if some relation exists
                        // that releases y's channel
        z = oldUsed(channel(y));
        if z ⇌ y ∈ SR or y >s z ∈ SR then
         begin          // replace y for z in z's channel
           Δ+ = Δ+ ∪ {y}; Δ− = Δ− ∪ {z};
           newUsed(channel(z)) = y;
                        //stop all media w | w ⇓ z ∨ z ⇓ w
           for all w ∈ (⇓)Closure(z)
              if w ∈ AM then
               begin    // stop w
                 Δ− = Δ− ∪ {w};
                 newUsed(channel(w)) = _
               end
         end
        else if y >p z ∈ SR then
         begin          // z pauses, then y starts
           Δ+F = Δ+F ∪ {z}; Δ+ = Δ+ ∪ {y};
           newUsed(channel(y)) = y
         end
                        // if y's channel is used by one
                        // of its components, add y to
                        // the set of active media
        else if ComponentOf(y, z) then
           Δ+ = Δ+ ∪ {y}
                        // if y is a component of z
        else if ComponentOf(z, y) then
         begin
                        // start y
           Δ+ = Δ+ ∪ {y};
           newUsed(channel(y)) = y
         end
      end;
     // if y is activated then try to activate all
     // media in (⇔)1step(y)
     if y ∈ Δ+ then
        Set = Set ∪ ((⇔)1step(y) \ Δ+)
   end
end.
```

```
RESTART (x: media item; AM, Δ_F^-: set of media items;
         oldUsed, newUsed: channel-media mapping)
// x: media item being restarted (with items in its closure),
// AM: set of currently active media,
// Δ_F^-: set of media items to be removed from the set of
//        frozen media,
// oldUsed: occupation function, before restart of x,
// newUsed: occupation function after restart of x and of
//          media displayed in parallel with it

begin
    for all y ∈ (⇔)Closure(x)
        // restart all paused media y whose channel is
        // free or already assigned
        if (y ∈ AM) and (oldUsed(channel(y)) = y or
            oldUsed(channel(y)) = _ ) then
          begin
            Δ_F^- = Δ_F^- ∪ {y}; newUsed(channel(y)) = y
          end
end.
```

Definition 9 (State transition function). The state transition function $next : \mathcal{S} \times \mathcal{E} \rightarrow \mathcal{S}$, where $\mathcal{S}$ is the set of all possible states and $\mathcal{E}$ is the set of events, is the function that, given a state s and an event e at the observable time instant n, returns the state $s' = next(s, e)$ at the observable time instant $n + 1$ where $s = \langle \mathcal{AM}_n, \mathcal{FM}_n, isUsed_n \rangle$, $s' = \langle \mathcal{AM}_{n+1}, \mathcal{FM}_{n+1}, isUsed_{n+1} \rangle$, $\mathcal{AM}_{n+1} = \mathcal{AM}_n \cup \Delta^+ \setminus \Delta^-$, $\mathcal{FM}_{n+1} = \mathcal{FM}_n \cup \Delta_F^+ \setminus (\Delta_F^- \cup \Delta^-)$, and $isUsed_{n+1}$, Δ^+, Δ^-, Δ_F^+, and Δ_F^- are defined according to the following process in which e is the occurring event and m the media item to which the event applies:

```
begin
    Δ^- = ∅; Δ^+ = ∅; Δ_F^+ = ∅; Δ_F^- = ∅;
    for all c ∈ CH do
        isUsed_{n+1}(c) = isUsed_n(c);

case e = start(m):
    if m ∈ FM_n then
        RESTART(m, AM_n, Δ_F^-, isUsed_n, isUsed_{n+1});
    if m ∉ AM_n then
        ACTIVATE (m, AM_n, Δ^+, Δ^-, Δ_F^+, isUsed_n,
                  isUsed_{n+1});

case e = end(m):
    x = m;
    while ∃y ∈ MI such that ComponentOf(y, x) do
                // if x is the last component of y then
                // consider the event end(y)
        if IsLast(y, x) then x = y
        else exit while ;
    if x ∈ AM_n then [2]
      begin     // x stops and releases its channel
        Δ^- = Δ^- ∪ {x}; isUsed_{n+1}(channel(x)) = _ ;
                // in relation a ⇔ b when a ends b must
                // be stopped
        for all y ∈ (⇔)Closure(x)
            if (y ∈ AM_n) then
              begin
                // stop media y which were activated by x
                Δ^- = Δ^- ∪ {y};
                isUsed_{n+1}(channel(y)) = _ ;
                // stop all media z | z ⇓ y ∨ y ⇓ z
                for all z ∈ (⇓)Closure(y)
                    if z ∈ AM_n then
                      begin
                        Δ^- = Δ^- ∪ {z};
                        isUsed_{n+1}(channel(z)) = _
                      end
              end;
        for all y ∈ MI such that x ⇒ y ∈ SR do
            if y ∈ FM_n then
                RESTART(y, AM_n, Δ_F^-, isUsed_n,
                        isUsed_{n+1});
            if y ∉ AM_n then
                ACTIVATE (y, AM_n, Δ^+, Δ^-, Δ_F^+,
                          isUsed_n, isUsed_{n+1})
      end;

case e = pause(m):
    if m ∈ AM_n then
      begin
        Δ_F^+ = Δ_F^+ ∪ {m};
        for all x ∈ (⇔)Closure(m)
            if (x ∈ AM_n) then Δ_F^+ = Δ_F^+ ∪ {x}
      end;

case e = stop(m):
    if m ∈ AM_n then
      begin
        Δ^- = Δ^- ∪ {m};
        isUsed_{n+1}(channel(m)) = _ ;
        // stop all media x | m ⇓ x ∨ x ⇓ m

        for all x ∈ (⇓)Closure(m)
            if x ∈ AM_n then
              begin
                Δ^- = Δ^- ∪ {x};
                isUsed_{n+1}(channel(x)) = _
              end
      end
end.
```

Figure 3 illustrates the finite-state automaton of the example presentation introduced in Sect. 3, where a subset of the states is considered. Specifically, we do not consider the events related to user interaction since we assume that the query must return a fragment of the original presentation and thus of its *original behavior*. Then the user is free to interact with the returned result, possibly reaching other states of the automaton that are not illustrated in Fig. 3.

From the initial state s_0 the activation of the presentation is given by the event $e = start(M_1)$ that starts the first module. The function $next(s_0, start(M_1))$ returns the state $s_1 = \langle \mathcal{AM}_1, \emptyset, isUsed_1 \rangle$ where $\mathcal{AM}_1 = \{M_1, title_1, story_1, tempo_1, c_{1,1}, info_{1,1}, sc_{1,1,1}, an_{1,1,1}, text_{1,1,1}\}$ and $isUsed_1$ associates with each media item its channel, whose details are not relevant here. Thus the transition from s_0 to s_1 captures the fact that the first module activates the music play and a set of text pages that show the position inside the masterwork and other information about the music.

[2] Nonactive items cannot end; the same holds for stop and pause events.

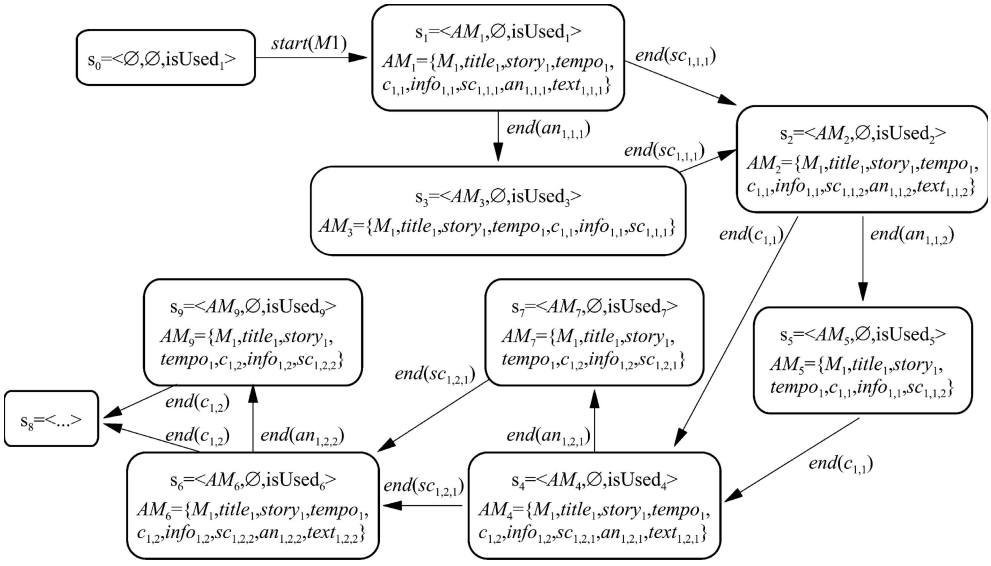

Fig. 3. The automaton of the *Pastorale* Symphony guide to listening (excerpt)

Without considering user interaction, the only possible events are $end(an_{1,1,1})$ and $end(sc_{1,1,1})$ since master events concern the lowest-level media items (i.e., scenes) and not the structured ones (i.e., clips, stories, and modules). Moreover, static media like pages have an infinite time duration; therefore end events are not meaningful.

In the first case, $text_{1,1,1}$ is stopped, since $text_{1,1,1} \in (\Leftrightarrow)Closure(an_{1,1,1})$, but remains on the screen till another page needs the channel. The automaton reaches the state s_3.

In the case of the natural termination of the first scene $[e = end(sc_{1,1,1})]$, the second scene starts, together with its associated page and animation. The automaton reaches a new state $s_2 = \langle \mathcal{AM}_2, \emptyset, isUsed_2 \rangle$ where $\mathcal{AM}_2 = \{M_1, title_1, story_1, tempo_1, c_{1,1}, info_{1,1}, sc_{1,1,2}, an_{1,1,2}, text_{1,1,2}\}$.

If animation $an_{1,1,2}$ ends, the associated page $text_{1,1,2}$ is terminated and the automaton reaches state s_5. When the second scene naturally ends, the clip reaches its ending point since $IsLast(c_{1,1}, sc_{1,1,2}) = True$; thus event $e = end(sc_{1,1,2})$ is equivalent to event $e = end(c_{1,1})$. The automaton goes to state s_4, with $\mathcal{AM}_4 = \{M_1, title_1, story_1, tempo_1, c_{1,2}, info_{1,2}, sc_{1,2,1}, an_{1,2,1}, text_{1,2,1}\}$. The media item $info_{1,2}$ shows which part of the masterwork is currently playing.

The second clip has the same structure and relationships of the first one, and thus the same natural behavior. At the end of scene $sc_{1,2,2}$, clip $c_{1,2}$ naturally ends, together with $story_1$ in which it is contained, since $IsLast(story_1, c_{1,2}) = True$. Since $tempo_1 \in (\Leftrightarrow)Closure(story_1)$, $info_{1,2} \in (\Leftrightarrow)Closure(c_{1,2})$, and $(\Leftrightarrow)Closure(sc_{1,2,2}) = \{sc_{1,2,2}, an_{1,2,2}, text_{1,2,2}\}$, all these media are stopped and their channels are released. By the relation $story_1 \Rightarrow story_2$ the second story is activated. Since each story has the same structure of $story_1$, this behavior repeats till the end of the presentation.

5 Retrieving consistent presentation fragments

The automaton describes formally how the presentation evolves; therefore it contains in each node the information about which media play together, under all possible conditions about triggering of events related both to unattended play and to user interaction. It is therefore the candidate source of information for reconstructing consistent presentation fragments after some media items have been retrieved according to a user query. As stated in Sect. 1, the retrieval engine is assumed to return a set $\mathcal{R}$ of pairs $\langle Mid, P \rangle$, each pair denoting a media item *Mid* and a reference to the presentation P.

The set of consistent presentation fragments containing the retrieved media items is built according to the following procedure, which is executed for each presentation P occurring in $\mathcal{R}$. For the sake of readability, in the following discussion we omit any explicit reference to P, implicitly assuming that media, states, events, and synchronization relationships are related to the same presentation.

1. For each item $r \in \mathcal{R}$ returned, let Mid_r be the media item identifier and $\mathcal{S}_r$ the set of states in which Mid_r is active. Each state $s_i \in S_r$ identifies the set of media items $\mathcal{AM}_i$ that play together and the corresponding channel assignments.
   Let $\mathcal{S}_f$ be the set of states that identify the fragments that must be returned to the user as the answer to his/her query. Initially, $S_f = S_r$. If Mid_r is active in two states s_1 and $s_2 = next(s_1, e)$ for some $e \in \mathcal{E}$, we consider only state s_1, i.e., $\mathcal{S}_f = \mathcal{S}_f \setminus \{s_2\}$. If Mid_r is active in two non-sequential states and inactive in the middle, we take both states for subsequent analysis.
2. For each state $s_i \in \mathcal{S}_f$, identify the event that activates the retrieved media item Mid_r. Since media items are activated by *plays with*($\Leftrightarrow$) or *activate*($\Rightarrow$) relations, all states are entered by *end* events, except for state s_1, which is entered by event $start(M_1)$. In terms of the presentation synchronization relationships, state s_i is entered under one of two conditions:

(a) An event $start(M)$ has occurred where $M \in \mathcal{AM}_i$ is a module, or
(b) A relation $x \Rightarrow m \in \mathcal{SR}$, where $m \in \mathcal{AM}_i$, $x \in \mathcal{AM}_j$, and $s_i = next(s_j, end(x))$.

In case 2(a), the module itself acts as the "starting" component of the presentation fragment. In case 2(b), only one media item in each state acts as the "starting" object, the others being activated in parallel with it or being already active by virtue of previous events. Let us call it m_0.

3. If condition 2(a) holds, the fragment to be returned is the module itself.
4. Otherwise, the minimum presentation fragment enclosing m_0 to be returned to the user is described by a schema obtained from the original presentation schema with the following transformations:
   (a) If relation $M \Leftrightarrow m_0 \notin \mathcal{SR}$, where M is the presentation module containing m_0, add it to $\mathcal{SR}$ and remove from $\mathcal{SR}$ any other relation $M \Leftrightarrow m_{ch}$, where $m_{ch} \in \mathcal{AM}_i$ and $channel(m_0) = channel(m_{ch})$;
   (b) If $ComponentOf(x, m_0)$ and relation $x \Leftrightarrow m_0 \notin \mathcal{SR}$, add it to $\mathcal{SR}$ and remove from $\mathcal{SR}$ any other relation $x \Leftrightarrow m_{ch}$, where $m_{ch} \in \mathcal{AM}_i$ and $channel(m_0) = channel(m_{ch})$;
   (c) For each media item $m \in \mathcal{AM}_i$, if $m \notin (\Leftrightarrow) Closure(m_0)$ and the relation $m \Leftrightarrow m_0 \notin \mathcal{SR}$, add relation $m_0 \Leftrightarrow m$ to $\mathcal{SR}$;
   (d) Iteratively remove all media items $x \notin Reach$, where $Reach$ is the set of reachable objects defined as follows:
      i. For any media item $m \in \mathcal{AM}_i$, $m \in Reach$;
      ii. For any media items x and m such that $m \in Reach$, if $m \Rightarrow x \in \mathcal{SR}$ and $\exists k \mid x \in \mathcal{AM}_k \wedge s_k \in S_r \wedge s_{j+1} = next(s_j, e) \wedge i \leq j < k$, for any event $e \in \mathcal{E}$, then $x \in Reach$;[3]
      iii. For any media item x, if $m \Leftrightarrow x \in \mathcal{SR}$ or $x \Leftrightarrow m \in \mathcal{SR}$ and $m \in Reach$, then $x \in Reach$.

   In this way the resulting fragment contains only the media items that in the original presentation play together with the items retrieved by the query. The items that temporally precede the retrieved ones are removed and the fragment ends when the retrieved media items are no longer active. In step 4(d)ii, the procedure retains the part of the presentation corresponding to the sequence of states $s_i \ldots s_k$ such that $s_{j+1} = next(s_j, e), i \leq j < k$ where the retrieved media are active in all the states s_j and not active in s_{i-1} and s_{k+1}.
   In terms of the presentation synchronization relations, if state s_k does not lead to a final state, an $x \Rightarrow y$ relation must exist for a medium x in state s_k. If this relation is removed from the presentation, and iteratively all the unreachable media items are also removed, the presentation stops playing when leaving state s_k.

This procedure assures also that if two media items m_1 and m_2 belonging to the same automaton state s are retrieved, the same presentation fragment is returned for both. Finally, channels are preserved since they are associated with the media statically.

[3] S_r is defined in step 1 of this procedure.

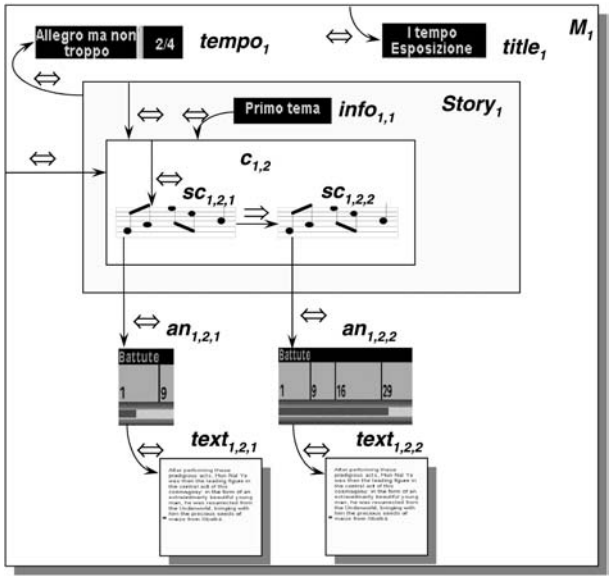

Fig. 4. A returned presentation fragment

Referring to our example, let us suppose that a user's query returns a pair $\langle c_{1,2}, P \rangle$. We can easily identify in the automaton of Fig. 3 the set of states in which $c_{1,2}$ is active, $S_r = \{s_4, s_6, s_7, s_9\}$. Since $s_6 = next(s_4, end(sc_{1,2,1}))$, $s_7 = next(s_4, end(an_{1,2,1}))$ and $s_9 = next(s_6, end(an_{1,2,2}))$, we consider only $S_f = \{s_4\}$. The set of objects that play with $c_{1,2}$ is then equal to $\mathcal{AM}_4 = \{M_1, title_1, story_1, tempo_1, c_{1,2}, info_{1,2}, sc_{1,2,1}, an_{1,2,1}, text_{1,2,1}\}$, which correspond to the text information about the position inside the music work, the bar animation, and the text comments that help to understand the music execution. Since the synchronization rule $c_{1,1} \Rightarrow c_{1,2}$ exists, $c_{1,2}$ is the "starting object" m_0. Then the synchronization relationships $M_1 \Leftrightarrow story_1$ and $story_1 \Leftrightarrow c_{1,2}$ are replaced by the relations $M_1 \Leftrightarrow c_{1,2}$ and $story_1 \Leftrightarrow c_{1,2}$. The resulting fragment is shown in Fig. 4 where the unreachable items have been removed together with the synchronization relationships involving at least one of them.

6 Discussion

Given an automaton $AUT(P) = \langle \mathcal{S}, \mathcal{E}, s_0, next, Final \rangle$, since we are not interested in recognizing acceptable streams of events, but our investigation is based on the information content of the states, we can consider $Final$ equal to the empty set. If we want to recognize the sequence of events that naturally lead from the starting point of the presentation to its end, that is, to the point where no media item is active, and all the channels are free, we let $Final = \{\langle \emptyset, \emptyset, isUsed_0 \rangle\}$, where $isUsed_0(c) = \_ \forall c \in \mathcal{CH}$.

The presentation fragment built by the procedure in Sect. 5 stops its execution when the media returned as query results are no longer active. Let us call $\mathcal{RM}$ the set of media items returned by the query; then $Final = \{s_i \mid s_i = next(s_j, e)$ for some j and e, $\mathcal{RM} \bigcap \mathcal{AM}_j \neq \emptyset$, $\mathcal{RM} \bigcap \mathcal{AM}_i = \emptyset\}$. We could ask if this behavior, which is consistent with the

way the presentation is transformed, is correct for the user's expectations. Indeed, such a sharp identification of the temporal scope of the fragment could lessen the significance of the result.

The presentation once started could follow its complete execution up to its end. In this case, the system returns to the user an *access point* and leaves the user free to stop the presentation playback at will.[4] Another choice could be to identify the fragment scope based also on the presentation static structure: a meaningful fragment ends when the module that contains it ends. These solutions can help the user to better understand the context and the significance of the resulting fragment. Therefore, while it is easy to set the beginning of the relevant scope of the fragment, its end is more a matter of meaning and semantics than of structure.

A second comment concerns the hierarchical structure of the presentation in terms of modules, stories, and sections. They build up a hierarchy of contexts that can give the user different levels of access to the presentation content. The retrieval model we have presented identifies only a minimal scope of the fragment of the presentation and a set of media temporally related. The design structure of the presentation can integrate this dynamic information with other information related to the identification of different "meaning scopes" in the presentation. For example, let us suppose that the user's query returns as two different results the text comments $text_{1,1,1}$ and $text_{1,1,2}$. Our procedure considers them separately.

Since they are active in two different states, s_1 and s_2 (Fig. 3), the system returns two different fragments. With reference to Fig. 5, (a) is the fragment returned for $text_{1,1,1}$ that corresponds to state s_1 and (b) is the result that contains $text_{1,1,2}$ corresponding to state s_2. Since both $text_{1,1,1}$ and $text_{1,1,2}$ belong to the same section, and s_1 and s_2 are two consecutive states, it is reasonable to return a single fragment that contains both the media items, much as the fragment depicted in Fig. 4. The choice between these two possibilities depends on the level of access required by the user.

In the same way, we suggest that a narrower scope than the one defined here could come from computing the $(\Leftrightarrow)Closure$ and $(\Leftrightarrow)1step$ sets on the retrieved media items. The scope is narrower because it does not consider media that are not directly connected by synchronization relationships to the retrieved ones. For example, background music in a cultural heritage presentation that starts at the beginning of the presentation and lasts up to the end does not bear a meaningful content to the presentation evolution; therefore it could be left out from the presentation of retrieved results, at least in a first browsing phase where the relevance of the returned items is evaluated by the user.

In the same way, elements like generic menus, banners, advertisements, and sidebars often surround the core information in Web documents are not only scarcely relevant but can divert the user's attention from the primary results of information retrieval. We could cut off these components from the fragment to return with a deep analysis of the automaton that describes the behavior of the presentation. Consider the following definitions:

Definition 10. A presentation *naturally ends* if the set of final states in the corresponding automaton is $Final = \{s_f\}$, where $s_f = \langle \emptyset, \emptyset, isUsed_f \rangle$, $isUsed_f(c) = \_ \; \forall c \in \mathcal{CH}$ and there exists a path that leads to the final state that is an ordered sequence of events $e_0, e_1, \ldots, e_n$ such that $e_0 = start(m_0)$, $e_i = end(m_j)$, for some medium m_j, and $s_{i+1} = next(s_i, e_i)$ $\forall i = 0 \ldots n$.

The ordered sequence of events might contain multiple instances of the same event $end(m_j)$, corresponding to different observable time instants.

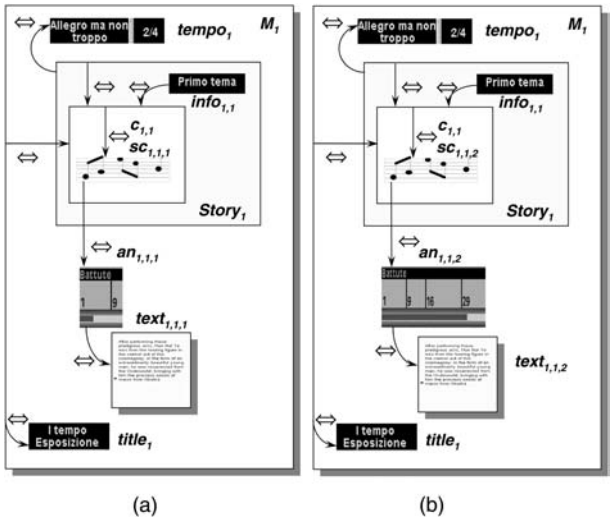

Fig. 5. Fragments returned for $text_{1,1,1}$ and $text_{1,1,2}$

Definition 11. Let P be a naturally ending presentation and $start(m_0) \ldots end(m_j) \ldots end(m_k)$ the sequence of labels of edges in the minimal path leading from the initial state s_0 to the final state s_f. We call *master objects* of the presentation the media objects that appear in the labels of the edges of the given path whose timing makes the presentation evolve in its natural behavior.

By Definition 11, if a presentation naturally ends, we can recognize which are the master objects that make the presentation to evolve and which media items are played as a consequence of the progress of the presentation. Then, once the set of states in which the user is interested is retrieved, the master objects contained in them must belong to the fragment returned. Other media items may belong or not according to other considerations, as for example, the level of access given to the user, the resources available at the client side, and so on.

As an example, in the result depicted in Fig. 4, $c_{1,2}$ is the master object (as are its scenes $sc_{2,1}$ and $sc_{2,2}$) and must be returned to the user. Since the object $tempo_1$ is not directly related to the clip or to the scene, it can be omitted. The text comments $text_{2,1}$ and $text_{2,2}$ can be returned or not according to other information, like, e.g., the user preferences.

[4] This behavior is obtained removing step 4(d)ii from the procedure described in Sect. 5.

7 Conclusion

Information retrieval in distributed multimedia documents requires modelling of the relationships among the media objects that build the presentations. We have illustrated a synchronization model that takes a step further, with respect to other models defined in the literature, in considering also user actions among the events that drive the dynamics of a multimedia presentation.

In this paper we have addressed the problem of identifying consistent fragments of a presentation given a set of media items contained in it. Based on the synchronization schema of the presentation, we have discussed an algorithm able to find media items that play at the same time and have shown how such an algorithm can be used in building answers to queries to multimedia presentation repositories. We have also discussed how to identify different fragments based on a deeper analysis of the media relations in terms of scopes and environments.

Technical issues concerning retrieval of multimedia data and index construction for complex composite documents deserve, of course, great attention and investigation in order to move from a modelling approach like the one described here to a prototype implementation.

The retrieval model is based on an automaton that formally describes the presentation states entered by the events that trigger media playback. The same automaton can be used to study some properties of the presentation and its behavior. The automaton allows one to classify media objects, e.g., which ones are the master objects that make the presentation evolve. Our future work will discuss this property with deeper consideration.

References

1. Adali S, Sapino ML, Subrahmanian VS (2000) An algebra for creating and querying multimedia presentations. Multimedia Sys 8(3):212–230
2. Baral C, Gonzalez G, Nandigam A (1998) SQL+D: extended display capabilities for multimedia database queries. In: Proceedings of ACM Multimedia 1998, Bristol, UK, September 1998, pp 109–114
3. Baral C, Gonzalez G, Son T (1998) A multimedia display extension to SQL: language and design architecture. In: Proceedings of the international conference on data engineering, Orlando, FL, February 1998
4. Celentano A, Gaggi O (2000) Synchronization model for hypermedia document navigation. In: Proceedings of the ACM symposium on applied computing (SAC2000), Como, Italy, March 2000, pp 585–591
5. Chiaramella Y (1997) Browsing and querying: two complementary approaches for multimedia information retrieval. In: Proceedings of Hypertext – Information Retrieval – Multimedia '97, Dortmund, WA, September 1997, pp 9–26
6. Cruz IF, Lucas WT (1997) A visual approach to multimedia querying and presentation. In: Proceedings of the 5th ACM international conference on multimedia '97, Seattle, WA, November 1997, pp 109–120
7. Gaggi O, Celentano A (2004) Modeling synchronized hypermedia presentations. Multimedia Tools Appl (in press)
8. Gupta A, Jain R (1997) Visual information retrieval. Commun ACM 40(5):71–79
9. Lee T, ShengL, Hurkan Balkir N, Al-Hamdani A, Ozsoyoglu G, Meral Ozsoyoglu Z (2000) Query processing techniques for multimedia presentations. Multimedia Tools Appl 11(1):63–99
10. Lee T, Sheng L, Bozkaya T, Hurkan Balkir N, Meral Ozsoyoglu Z, Ozsoyoglu G (1999) Querying multimedia presentations based on content. IEEE Trans Knowl Data Eng 11(3):361–385
11. Miller RJ, Tsatalos OG, Williams JH (1995) Integrating hierarchical navigation and querying: a user customizable solution. In: Electronic proceedings of the ACM workshop on effective abstractions in multimedia, San Francisco, CA, November 1995
12. Enda Multimedia: Il Mondo della Musica Classica. Enda Srl, Milano, Italy, 1996
13. Synchronized Multimedia Working Group of W3C (2001) Synchronized Multimedia Integration Language (SMIL) 2.0 Specification, August 2001