Rapporto di Ricerca CS-2008-3
Aprile 2008

Annalisa Bossi, Ombretta Gaggi
Automatic validation of SMIL documents

Dipartimento di Informatica, Università Ca’ Foscari di Venezia
Via Torino 155, 30172 Mestre–Venezia, Italy
Automatic validation of SMIL documents

Annalisa Bossi
University Ca’ Foscari of Venice
and
Ombretta Gaggi
University of Padua

In this paper we consider the problem of automatic verification of SMIL documents and present a tool which can assist the user in the complex task of authoring a multimedia presentation. The tool is based on a formal semantics defining the temporal aspects of SMIL tags by mean of a set of inference rules. The rules, in the spirit of Hoare’s semantics, describe how the execution of a piece of code changes the state of the computation of a player. If any temporal conflict is found, the system returns to the user a message pointing out the tag which contains the error and its motivation. This helps the user to correct the error.

Categories and Subject Descriptors: H.5.4 [Information Interfaces and Presentation]: Hypertext/Hypermedia—Theory, User issues; I.7.2 [Document and Text Processing]: Document Preparation—Standards, Markup Languages, Hypertext/hypermedia

General Terms: Verification

Additional Key Words and Phrases: SMIL, authoring, consistency checking

1. INTRODUCTION

Authoring and design of a multimedia presentation is a complex and error-prone activity, especially when the complexity of the temporal structure of a multimedia document increases together with the chance of including a temporal conflict in the synchronization constraints. Moreover, the playback of the media objects may be completely changed by the user, who can follow a link, click on an image, or even simply move the mouse over or out of an item.

Despite many researches address this problem, defining models ([Allen 1983], [Hardman et al. 1994]), languages and tools ([Bulterman et al. 1998], [Jourdan et al. 1998], [Soares et al. 2000]), the authoring of a complex multimedia presentation is far away from being considered as easy as using a word processor or a drawing program. A step toward a solution was the introduction of the standard language SMIL [Bulterman et al 2005] for the output.

Since SMIL’s first appearance, many authoring tools and players have been implemented offering to their users different facilities like visual editors or preview windows. Unfortunately, this tools usually check only the syntactic correctness of the document, and do not validate the temporal consistency of the document.

We call a semantic error the presence of a conflict in the definition of the tags and attributes, like wrong definition of attributes of the same object or wrong definition of the temporal structure of the document. For example, the tag `<text id="txt01" begin="3s" end="5s" dur="5s"/>` presents a conflict since it defines a text message txt01 displayed at time instant 3 and removed after 2 time units, but at the same time defines its duration equal to 5 seconds. The tag:
describes a sequence of two images, each one visualized for 5 seconds. The conflict is between the overall duration of the sequence, i.e., 5 seconds according to its attribute `dur`, and the sum of the durations of the single images it contains.

Semantic errors are particularly dangerous since they usually cannot be automatically corrected because they point out a contradiction in the definition of the behavior of media items, and therefore they require a decision to be solved. Moreover, in presence of temporal conflicts even simple multimedia documents may have different behaviors, according to the chosen player. Therefore the final behavior is almost unpredictable ([Jourdan 2001; Sampaio et al. 2000; Valente and Sampaio 2007]).

Built-in decisions, performed by some authoring system ([Sampaio and Courtiat 2004]), are not always a good solution. As an example, more common players, e.g., GRiNS ([Oratrix]) or RealPlayer ([RealNetworks]), do not point out a temporal conflict to the user but the playback goes on and the duration of an object is equal to the minimum duration defined. This means that, in the first case, text `txt01` lasts for two seconds, and in the second case, image `img02` is not displayed at all and the presentation ends immediately after the first image. Possibly, this is exactly what the author expects but, if not, it is important to individuate the existence of the semantic conflict. In particular, we are not sure that the short interval time of 2 seconds is sufficient to read message `txt01` and the author would not have included `img02` in the presentation if her/his real intention is not to display it. For these reasons we think that a good authoring tool should point out to the user semantic errors and assist him/her while fixing them. The second type of errors is very common and difficult to find out and be fixed, especially when the complexity of the presentation increases. The current state of implementation of the players does not help the user, since it does not always make clear if the misbehavior is due to a semantic conflict or to a bug ([Eidenberger 2003]).

We think that this problem is partially due to the lack of a formal semantics for the SMIL language ([Jourdan 2001; Sampaio et al. 2000]), which can be interpreted differently from different developers. Moreover, as reported in [Jourdan 2001] by Muriel Jourdan, one of the editors of the SMIL 2.0 Timing and Synchronization Module [Patrick Schmitz and Jeff Ayars and Bridie Saccocio and Muriel Jourdan 2001], “...SMIL 2.0 complexity is so great that rejecting the use of formal supports gives rise to a difficult-to-read specification that cannot be free from inconsistency”.

Consistency checking is an important issue for multimedia documents in order to guarantee the generation of a renderable presentation, not only during the authoring phase but also in all their subsequent utilizations. We develop a general tool for the automatic validation of SMIL documents, which can be used both in conjunction with an authoring system, and as the basis for the development of an efficient player. The Semantic Validator Module detects the presence of conflicts in a presentation and points out the wrong values. This information allows a player to avoid starting the playback of a wrong presentation, and an authoring system to help the user while correcting it. Its use as basis for a player is particularly interesting since most
available players are often unstable or not free of charge as reported in [Eidenberger 2003]. The major problem is a robust resolution of start and end time of tags. As output of the validation of a consistent document, our tool generates the correct begin and end time of every media item, which can be used for playback.

The tool is based on a formal semantics for the language SMIL 2.1, defined by mean of a set of inference rules inspired by Hoare logic ([Hoare 1969]). The central feature of Hoare logic is the Hoare triple which describes how the execution of a piece of code changes the state of the computation. This choice brings the advantage that the SMIL structure can be enriched by assertions, expressing the temporal properties, which can be used during the authoring phase when media items are collected in more complex constructs. As an example, our tool can verify the consistency of a multimedia presentation resulting from a context adaptation process. In this case, the document is dynamically build up by selecting media items compatible with the great number of different situations in which a multimedia presentation can be played, in term of availability of resources (e.g., network bandwidth, CPU time), device type (e.g., desktop, laptop, cell-phone) and properties (e.g., screen size, number of colors). This process often generates conflicts which must be solved in order to guarantee the playback.

As we compose a multimedia presentation by nesting a SMIL tag into another, our rules allow us to compose the semantics by evaluating a single tag inside a more complex nesting. In other words, the proposed semantics is compositional and helps the author to modularize her/his work thus mastering the complexity of the verification of a multimedia presentation consistency.

We must note here that this paper does not aim at augmenting or correcting the standard SMIL, but at offering a formal semantics which can help guide SMIL developers, thus improving the standard specification.

The paper is organized as follows. In the next section we give preliminary definitions including an abstract player and the assertion language and its interpretation. Section 3 presents the axiomatic semantics for SMIL language which is at the basis for development of the verification tool described in Section 4. We conclude in Section 5.

A initial subset of the semantics presented in this paper appeared in [Bossi and Gaggi 2007].

2. PRELIMINARIES

This paper presents a tentative approach to the formulation of an axiomatic semantics for the verification of SMIL 2.1 tags using a formal proof system in the Hoare-style. In this section we start by introducing the basic elements and notations used through the paper.

Our framework currently does not consider the whole SMIL language, but the attributes which are missing form a very limited subset of the one allowed by the standard. Moreover, we are not interested in SMIL tags which do not influence the temporal synchronization of the overall multimedia presentation but only its layout or spatial disposition of the media items, e.g. definition of regions, transition effects and animations. Table I describes the set of tags and attributes addressed in this paper as well as their possible values. All of the synchronization considered
in this paper remains valid even in the third version of the standard currently under definition [Bulterman et al 2008]. Note that we do not consider the presence of links to other documents. We plan to fill this gap in our future work.

2.1 A definition of an abstract player

Our inference rules describe how the execution of a piece of SMIL code changes the state of the playback of a multimedia presentation. The notion of state of a player (or of the presentation’s playback) underlying our model, is determined by a set of particular values describing significant aspects of media items. Since we are interested in describing only those aspects that might influence temporal consistency, a state describes significant time instants: the start and end time instants of all SMIL tags contained in the presentation as well as the duration of each continuous object and the user interaction captured by the player.

Most of these information can be retrieved directly from the SMIL documents. The only useful information which is missing concerns the natural duration of each continuous media and the events due to user interaction. More precisely by natural duration we mean the number of time instants for which a continuous media item plays in absence of user interaction or other temporal specifications. We denote by $\mathbb{R}^+$ the set of positive, computer representable, real numbers augmented with the special symbol $\bot$, the “undefined” value. The value $\bot$ is used to represent the absence of information, either because it is not yet available or because it does not exist, like the duration of a static object.

A player enters into a new state in response to an event. SMIL specification considers two types of events, interactive events, i.e., the user interactions like a click on an image or the movement of the mouse, and non-interactive events, i.e., event due to SMIL synchronization, e.g., the start or the end of a media object. The state of a player must record all this information.

Then, an abstract player is fully described by:

— a clock $c$ which is a value that records time progression;
— a function “description" $\delta : \mathbb{I} \rightarrow \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+$ which maps an identifier $id \in \mathbb{I}$
to a triple of values \(\langle b_{id}, e_{id}, d_{id} \rangle\) which denote, respectively, the start time, the 
end time and the natural duration of the tag identified by \(id\);

— a function “event time” \(\tau : ((E \times Id) \cup E) \rightarrow \mathbb{R}^+\) which records the last time 
instant in which an event \(e \in E\) (possibly associated to a tag \(id \in Id\)) occurred.

Let \(\Sigma = \mathbb{R} \times (Id \rightarrow \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+) \times ((E \times Id) \cup E) \rightarrow \mathbb{R}^+\), a consistent state 
\(\sigma \in \Sigma\) of a player is a triple \(\langle \sigma_{e}, \sigma_{d}, \sigma_{\tau} \rangle\) such that:

\[- \forall id \in Id\text{ we have: } \sigma_{\tau} \leq \sigma_{c};\]
\[- \forall id \in Id\text{ and } \sigma_{\delta}(id) = \langle b_{id}, e_{id}, d_{id} \rangle \text{ we have: if } b_{id} \neq \bot \text{ then } 0 \leq b_{id} \leq \sigma_{c}, \text{ if } e_{id} \neq \bot \text{ then } b_{id} \leq e_{id}, \text{ if } d_{id} \neq \bot \text{ then } b_{id} \neq \bot.\]

Moreover, since each time a new state is reached new information about the media 
items playback are added, for any couple of states \(\sigma^1\) and \(\sigma^2\), such that \(\sigma^1 \neq \sigma^2\),
the functions \(\sigma^2_{e}\) and \(\sigma^2_{d}\) are more defined than the corresponding functions in \(\sigma^1\).
This means that: if \(b^2_{id} = \bot\) then \(b^1_{id} = \bot\), and if \(b^1_{id} = t \neq \bot\) then \(b^2_{id} = t\). The same considerations
apply also to the other components of the function \(\sigma_{\delta}\). Differently,
the function \(\sigma_{\tau}\) records the last event occurred, hence it is increasing with respect
to the clock values: if \(\sigma^2_{\tau} = \bot\) then \(\sigma^1_{\tau} = \bot\) and if \(\sigma^1_{\tau} = t \neq \bot\) then \(\sigma^2_{\tau} = t' \geq t\).

We note here that the function \(\tau\) records only the interactive events and not
the non-interactive ones which are fully described by the function “description” \(\delta\).
Interactive events may involve a media item, like a click on a item or the movement
of the mouse over it, but there are also user interactions which are not related to
a specific item like the user keying a key in the keyboard. In the first case, the
function requires as input the type of the event and the \(id\) of the item, in the second
case, the \(id\) is absent. A partial list of supported events can be found in Table I.

As it will be discussed in the following, interactive events may occur several
times, e.g., a user may click in different moments on a button. Therefore, a media
item can be played several times in response to an user event. To represent this
fact we could have associated to each identifier a sequence of tuples representing
the temporal aspects of its various executions. We choose a different solution, that
of creating a new name each time a media item is played. This means that we
can have multiple ids referring to different activations of the same item: they refer
to the same file but are considered completely different from the synchronization
point of view. The new names are generated by the function \(\nu\); thus if \(c\) identifies
a media item then \(\nu(c), \nu(\nu(c)), \ldots, \nu^i(c), \ldots\) are different identifiers for different
instances of the same media item \(c\).

2.2 The assertion language

The rules provide an axiomatic semantics for the temporal aspects of SMIL tags in
the spirit of Hoare logic. Therefore they allow us to derive judgements in the form
of triplets:

\[\{P\} \quad t \quad \{Q\}\]

where \(P\) and \(Q\) are assertions, respectively the \textit{precondition} and the \textit{postcondition},
and \(t\) is a SMIL tag.

The assertion language used to express pre/post conditions includes a set of basic
functions representing the significant temporal aspects of the media. Assertions
are formed by sets of constraints on values returned by these functions. Table II lists all the functions, and abbreviations, used in the assertions. For instance, if \( c \in \mathbb{d} \) denotes a media, we write \( \text{begin}(c) = 10 \) to mean that the media \( c \) starts its execution at clock time 10. Given an assertion \( B \) which contains the equality \( \text{begin}(c) = t \) (or \( \text{end}(c) = t \)) we use also the notation \( \text{begin}_B(c) \) (or \( \text{end}_B(c) \)) to denote the time instant \( t \) occurring in the corresponding equality. We note here that the SMIL language allows multiple formats of legal clock values, e.g. the values 00:02:33, 2:33 and 153s represent the clock value two minutes and 33 seconds. Since they can be easily translate in the real number representing the number of seconds, we suppose that all our functions return a real value.

The assertion language contains also the functions \( t_{ev} : \mathbb{d} \rightarrow \mathbb{R} \) that represents the current time instant in which the SMIL tag \( id \) is evaluated. By current time instant in which a tag \( id \) is evaluated we mean the clock of the state in which, considering a player executing the presentation, the player evaluates that command. The SMIL Specifications call this time the *implicit syncbase*. We use it in the preconditions.

The occurrence of user interactions is represented in the precondition of a tag by equalities in the form \( \text{times}(ev, id) = (t_1, \ldots, t_n) \) (or \( \text{times}(ev) = (t_1, \ldots, t_n) \)) where the function \( \text{times} : (E \times \mathbb{d}) \cup E \rightarrow \mathbb{R}^* \), returns a sequence of time instants \( (t_1, \ldots, t_n) \in \mathbb{R}^* \) representing the next time instants in which the same event (on the same media item) will occur. By time instant in which an event occurs we mean the value of the clock when the player registers the occurrence of that event. We use also the functions \( \text{time}(ev, id) \) and \( \text{time}(ev) \) to represent just the next occurrence of the event \( ev \), that is the first element in the sequence \( (t_1, \ldots, t_n) \).

Let \( \sigma \in \Sigma \) be a state and \( A \) an assertion, we say that \( \sigma \) satisfies the assertion \( A \), and write \( \sigma \models A \) if the constraint on real values obtained by applying to \( A \) the following transformations holds:

- each occurrence of \( \text{begin}(id) \) is replaced by the first component of \( \sigma_0(id) \);
- each occurrence of \( \text{end}(id) \) is replaced by the second component of \( \sigma_0(id) \);
- each occurrence of \( \text{dur}(id) \) is replaced by the third component of \( \sigma_0(id) \);
- each occurrence of \( \text{times}(ev) = (t_1, \ldots, t_n) \) (or \( \text{times}(ev, id) = (t_1, \ldots, t_n) \)) is replaced by \( \sigma_c \leq t_1 \);
- each occurrence of \( \text{time}(ev) = t \) (or \( \text{time}(ev, id) = t \)) is replaced by \( \sigma_c \leq t \).

A state satisfies the equality \( \text{times}(ev, id) = (t_1, \ldots, t_n) \) (or \( \text{times}(ev) = (t_1, \ldots, t_n) \)) if it enables the first occurrence of that event, i.e., if the value of its clock is less or equal to \( t_1 \). Since the player enters a new state in response to an event, for each time instant \( t_i \), \( 1 \leq i \leq n \) there exists a state \( \sigma^i \) such that \( \sigma^i = t_i \).

The triple \( \{P \} t \{Q \} \) can be read as: whenever the evaluation of the tag \( t \) starts in a state \( \sigma^0 \) which satisfies the assertion \( P \), i.e. \( \sigma^0 \models P \), then it terminates in a state \( \sigma^f \) which satisfies the assertion \( Q \), i.e. \( \sigma^f \models Q \). According with this interpretation, the following rule of consequence holds:
Table II. List of functions and notations used in the definition of the proof rules

<table>
<thead>
<tr>
<th>Function</th>
<th>Where</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{cr} : \text{Id} \rightarrow \mathbb{R} )</td>
<td>Pre</td>
<td>returns the current time instant in which the SMIL tag ( \text{id} ) is evaluated</td>
</tr>
<tr>
<td>( \text{dur} : \text{Id} \rightarrow \mathbb{R} )</td>
<td>Pre</td>
<td>returns a real value representing the time interval for which a continuous media item plays</td>
</tr>
<tr>
<td>( \text{times} : (\mathcal{E} \cup (\mathcal{E} \times \text{Id})) \rightarrow \mathbb{R}^* )</td>
<td>Pre</td>
<td>returns the sequence of all the time instants in which an event occurs</td>
</tr>
<tr>
<td>( \text{time} : (\mathcal{E} \cup (\mathcal{E} \times \text{Id})) \rightarrow \mathbb{R}^\bot )</td>
<td>Pre</td>
<td>returns the time instant of the next occurrence of an event</td>
</tr>
<tr>
<td>( \text{begin} : \text{Id} \rightarrow \mathbb{R} )</td>
<td>Post</td>
<td>returns the time instant media item ( \text{id} ) starts</td>
</tr>
<tr>
<td>( \text{end} : \text{Id} \rightarrow \mathbb{R} )</td>
<td>Post</td>
<td>returns the time instant media item ( \text{id} ) ends</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{begin}_B(c) )</td>
<td>denotes ( t ) if ( { \text{begin}(c) = t } \subseteq B )</td>
</tr>
<tr>
<td>( \text{end}_B(c) )</td>
<td>denotes ( t ) if ( { \text{end}(c) = t } \subseteq B )</td>
</tr>
</tbody>
</table>

CONSEQUENCE \[
\frac{P_1 \Rightarrow P \quad \{P\} \subset \{Q\} \quad Q \Rightarrow Q_1}{\{P_1\} \subset \{Q_1\}}
\]

As a general remark, in the triple \( \{P\} \subset \{Q\} \) the precondition \( P \) contains, among others, the current time instant of the tag \( t \), the natural duration of media items which it defines (if applicable) and the occurrence of events. The postcondition \( Q \) contains the definition of the time instants in which the tags defined in \( t \) begin and/or end. Media items definitions are evaluated through axioms, while for \( \text{par} \), \( \text{seq} \) and \( \text{excl} \) composition more complex rules are needed.

2.3 Notational conventions

In the following section we use a number of special notational conventions to introduce the set of inference rules describing the semantics of the SMIL tags.

Table I lists a set of abbreviations used for the representation of the SMIL tags. For instance \(<\text{cmd } c>\) stands for any tag SMIL with the attribute \( \text{id} = "c" \). Moreover, we use the general form \( \text{end} = "k" \) and \( \text{dur} = "k" \) to represent the attributes of a tag where the meta-variable \( k \) is either any of the admitted values for the particular attribute, or the special value \( \text{void} \). The value \( \text{void} \) represents the absence of that attribute in the command line and allows us to introduce only one general rule for each compound tag. As regards the attribute \( \text{begin} \) we assume it is always defined since its absence can be represented by the value \( k = "0" \). For instance, \(<\text{video id} = "v" \text{begin} = "0" \text{dur} = "5" \text{end} = "\text{void}"/> \) is considered as a synonymous of \(<\text{video id} = "v" \text{dur} = "5"/> \).

The advantage of this representation is that of avoiding repetition of very similar rules, but we need a set of predicates to check the existence of an attribute’s value before using it. We need also to classify the tags which occur in a SMIL document with respect to the values of their attributes \( \text{dur} \) and \( \text{end} \). Hence we introduce some auxiliary predicates and sets whose description can be found in Table III.
### 3. A SEMANTICS FOR SMIL TAGS

SMIL language definition provided by [Bulterman et al 2005; 2008] does not contain a formal specification of tags and attributes semantics. The recommendation is divided into sections, some of which are defined “normative”. Sometimes, an algorithm is provided to better explain how significant time instants are computed, but neither a formal definition nor verification tools have been implemented by Synchronized Multimedia Working group of W3C to check the semantic correctness of SMIL tags.

In this section, we define a formal system which is able to find out temporal conflicts of a multimedia presentation defined using SMIL. The system provides a Hoare-like logic for SMIL by a set of inference rules describing how the execution of a piece of code changes the state of the playback. Since the standard SMIL lacks a formal semantics, soundness and completeness of our approach cannot be formally proved, but we consider that our semantics is correct according to any operational semantics which formalizes the changes in the state of a player described informally in the SMIL recommendation.

We start by considering self contained tags, i.e., SMIL commands whose synchronization do not refer to other media items or tags. Axioms to verify the correctness of statements which define media items are listed in Table IV. The use of interactive events is discussed in sections 3.2 and 3.3.

Assume we want to verify the triple:

\[
\{ P \} <\text{video id='v'} \begin{array}{c}
\text{begin='2'}
\end{array}> \{ Q \}
\]

where the precondition \( P \) is \( \{ \text{dur}(v) = 5, t_{cr}(v) = 0 \} \) and the postcondition \( Q \) is \( \{ \text{begin}(v) = 2, \text{end}(v) = 7 \} \). We can prove its correctness since we can instantiate the axiom \( \text{cont+begin} \) by using the values \( \text{start} = 2, k1 = 2 \) and \( \text{stop} = 7 \).

The system can also be used as the basis for the implementation of a player. In this case, the axiom \( \text{cont+begin} \) can be used to describe the transformation of the state of the player. In fact if the player starts in a state \( \sigma^0 \models P \), i.e., \( \sigma^0 = \langle 0, \sigma_d, \sigma_T \rangle \) such that \( \sigma_d(v) = \langle \bot, \bot, 5 \rangle \). Then it ends in a state \( \sigma^f \models Q \), i.e., \( \sigma^f = \langle 7, \sigma_d^f, \sigma_T^f \rangle \) such that \( \sigma_d^f(v) = \langle 2, 7, 5 \rangle \), thus obtaining the values used to start and stop the video.

The situation is a little more complicated if the media definition contains also an

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>finite(k)</td>
<td>holds if ( k ) is a real value</td>
</tr>
<tr>
<td>indefinite(k)</td>
<td>holds if ( k ) is equal to &quot;indefinite&quot;</td>
</tr>
<tr>
<td>defined(k)</td>
<td>holds if ( k ) is not equal to &quot;void&quot;</td>
</tr>
<tr>
<td>NotDur</td>
<td>contains all the statements with attributes ( \text{dur} ) and ( \text{end} ) equal to void</td>
</tr>
<tr>
<td>Closure(c)</td>
<td>contains ( c ) and all the statements defined inside the tag ( c ), at any level of nesting</td>
</tr>
<tr>
<td>Indef(c)</td>
<td>holds if in ( \text{Closure}(c) ) there are tags with attribute ( \text{end} ) (or ( \text{dur} )) equal to &quot;indefinite&quot;</td>
</tr>
</tbody>
</table>

Table III. List of predicates and sets used in the definition of the proof rules
Since the flexibility of SMIL tags allows us to describe the same temporal behavior

\[ \{A \cup Pre\} \langle static \ m \ begin='k1'\rangle \ \{A \cup Post\} \]

where \( Pre = \{t_{\text{end}}(m) = \text{start} - k1\} \) and \( Post = \{\text{begin}(m) = \text{start}\} \)

\[ \{A \cup Pre\} \langle cont \ m \ begin='k1'\rangle \ \{A \cup Post\} \]

where \( Pre = \{t_{\text{end}}(m) = \text{start} - k1, \text{dur}(m) = \text{stop} - \text{start}\} \)

\( Post = \{\text{begin}(m) = \text{start}, \text{end}(m) = \text{stop}\} \)

\[ \{A \cup Pre\} \langle media \ m \ begin='k1' \ end='k2' \ dur='k3'\rangle \ \{A \cup Post\} \]

where \( Pre = \{t_{\text{end}}(m) = \text{start} - k1, Post = \{\text{begin}(m) = \text{start}\} \cup \text{End}\} \)

\[ \text{End} = \begin{cases} \{\text{end}(m) = \text{start} - k1 + k2\} & \text{if finite}(k2) \\ \{\text{end}(m) = \text{start} + k3\} & \text{if finite}(k3) \\ 0 & \text{otherwise} \end{cases} \]

**APPLICABILITY CONDITION:**

\( (\text{defined}(k2) \lor \text{defined}(k3)) \land ((\text{finite}(k2) \land \text{finite}(k3)) \Rightarrow k3 = k2 - k1) \)

\( \land (\text{finite}(k2) \Rightarrow \neg \text{indefinite}(k3)) \land (\text{finite}(k3) \Rightarrow \neg \text{indefinite}(k2)) \)

**SMIL SPECIFICATIONS [Bulterman et al. 2008]**

- **begin:** defines when the element becomes active;
- **end:** describes the end value as an offset from an implicit syncbase;
- **dur:** specifies the simple duration.

### Table IV. Proof rules for media items definitions

**end**, or **dur**, attribute. As an example, consider the triple:

\[ \{t_{\text{end}}(v) = 0\} \langle \text{video id='v' begin='2' end='3' dur='void'}/\rangle \{Q\} \]

where \( \{Q\} = \{\text{begin}(v) = 2, \text{end}(v) = 3\} \).

As discussed in Section 2.3, \( \langle \text{video id='v' begin='2' end='3' dur='void'}/\rangle \) is a synonymous of \( \langle \text{video id='v' begin='2' end='3'}/\rangle \). Therefore, we can instantiate the rule **MEDIA+BEGIN+END+DUR** with the values \( \text{start} = 2, k1 = 2, k2 = 3 \) and \( k3 = \text{void} \). The applicability conditions hold since \( k2 \) is finite, and \( k3 = \text{void} \).

In this case, according to the SMIL recommendation, if the player starts in a state \( \sigma \) which satisfies the precondition, i.e. such that \( \sigma_c = 0 \), then the final state reached by the player is \( \sigma = (3, \sigma'_f, \sigma'_v) \) where \( \sigma'_f(v) = (2, 3, 5) \) and thus \( \sigma = Q \).

Note that the rule **MEDIA+BEGIN+END+DUR** defines the end time of a media item \( m \) only if both \( k2 \) and \( k3 \) are not equal to **\text{indefinite}**. Moreover, media items definition does not lead to temporal conflicts unless the author defines both the **dur** and the **end** attributes. The applicability condition disallows the application of the rule in presence of uncorrect values of these attributes; for instance when both the attributes **dur** and **end** are finite, the relation \( k3 = k2 - k1 \) must hold, otherwise the applicability conditions point out the temporal conflict to the user.

### 3.1 Rules for parallel and sequential composition

When media definitions are nested into parallel and sequential composition, the evaluation of these structures requires the definition of more complex rules.

Since the flexibility of SMIL tags allows us to describe the same temporal behavior
using both a \texttt{par} or a \texttt{seq} tag, we base the discussion of this section mainly on
the description of the rules for the parallel composition. The sequential composition is
discussed at the end of this section.

\begin{table}
\begin{tabular}{ | l | }
\hline
\textbf{Parallel Composition Rule}\texttt{(par + begin + end)} & \\
\hline
\multicolumn{1}{| c |}{ $\{A_i \cup \{t_{cr}(c_i) = init + k1\}\} \ c_i \ B'_i \ \forall i \ 1 \leq i \leq n$} & \\
\hline
\end{tabular}
\end{table}

where $A' = \bigcup_{i=1}^{n} A_i \cup \{t_{cr}(c) = init\}$

$B = \bigcup_{i=1}^{n} B_i \cup \{begin(c) = init + k1\} \cup End$

$\text{stop} = \begin{cases} 
    \text{init + k2} & \text{if finite}(k2) \\
    \max_{c_i} \{\text{end}_B(c_i)\} & \text{if} \ -\text{defined}(k2)
\end{cases}$

$\text{End} = \begin{cases} 
    \{\text{end}(c) = \text{stop}\} & \text{if} \ -\text{Indef}(c) \\
    \emptyset & \text{otherwise}
\end{cases}$

$B'_i = \begin{cases} 
    B_i \setminus \{\text{end}_c(c_i) = \text{stop}\} & \text{if} \ c_i \in \text{NotDur} \land \text{finite}(k2) \\
    B_i & \text{otherwise}
\end{cases}$

\textbf{Applicability Condition:}

finite$(k2) \implies (\neg \text{Indef}(c) \land k2 \geq k1) \land \forall c_i (\text{end}_B(c_i) \leq \text{stop} \lor c_i \in \text{NotDur})$

\land \text{Indef}(c) \implies (\neg \text{defined}(k2) \lor \text{finite}(k2))$

\land \forall c_i \begin{cases} 
    \text{begin}_B(c_i) \geq \text{init} + k1
\end{cases}$

\textbf{SMIL Specifications [Bulterman et al 2008]}

A par container, short for “parallel”, defines a simple time grouping in which multiple ele-
ments can play back at the same time. The implicit syncbase of the child elements of a par
is the begin of the par. […] The implicit duration ends with the last active end of the child

elements.

Table V. Proof rule for the parallel composition when the attribute \texttt{dur} is equal to \texttt{void}

We start our analysis by considering the parallel composition expressed by the tag
\texttt{par} when the attribute \texttt{dur} is not present (i. e. \texttt{dur} = ‘‘\texttt{void}’’), the attribute \texttt{begin}
is present (possibly equal to zero) and the attribute \texttt{end} is \texttt{void} or \texttt{indefinite} or
a real number. The \texttt{par + begin + end} rule described in Table V defines the semantics
of the parallel composition in these cases. In the postcondition we distinguish the
components $B_1 \ldots B_n$, to make it clear that the postcondition contain information
about each $c_i$, be it a media object or a synchronization structure.

To prove the correctness of the tag \texttt{<par c> c_1 ... c_n</par>}, each $c_i$ must be
proved to be correct by assuming the current time instant of the parallel tag plus the
offset given by the attribute \texttt{begin} as its current time instant, i. e., if $(t_{cr}(c) = \text{init})$
is contained into the precondition of the tag $c$, the precondition of each tag $c_i$ must
contain $(t_{cr}(c_i) = \text{init} + k1)$ where $k1 \geq 0$ is the value of the attribute \texttt{begin} and
\texttt{init} is the time instant at which the statement \texttt{par} is evaluated.

The evaluation of the end time instant of a par tag is a little more complicated,
and not always possible. As a general remark, it is not possible to calculate the
end time of a media item in two cases: if it is a static object and it does not
have an attribute \texttt{end} or \texttt{dur} defined, or if it has an attribute \texttt{end} or \texttt{dur} equal
to ‘‘\texttt{indefinite}’’. In the same way, the ending time of a par statement cannot
be calculated if its attribute \texttt{end} (or \texttt{dur}) is equal to ‘‘\texttt{indefinite}’’, or it
is not defined and one of its children has the attribute end (or dur) equal to ‘‘indefinite’’. In this case, the tag ends together with the overall presentation.

Once we are able to decide whether a parallel composition terminates, we must calculate the time instant stop. The semantics which describes the evaluation of stop is complex since four different cases have to be considered:

1. the tag \( c \) does not contain the definition of attribute end (i.e. end = ‘‘void’’): in this case, the tag \( c \) ends when all its children (which are not static objects in NotDur) have finished their playbacks, i.e. at time instant \( \text{stop} = \max_{c_i} \{ \text{end}_B(c_i) \} \);
2. all tags contained in the par tag end up before or together with the par statement’s end, more precisely before time instant \( \text{init} + k2 \);
3. some continuous media items defined inside \( c \) have a natural duration wider than the duration of \( c \);
4. some items defined inside \( c \) have a duration, defined with an attribute dur or end, wider than the duration of \( c \).

Cases 1, 2 and 3 are all correct. In the first two cases, each media object or statement within \( c \) lasts for a period of time equal or shorter than the duration of \( c \). If a static media item has not a duration defined, its duration is equal to the duration of \( c \). In case 3, if a continuous media \( c_i \) has a natural duration longer than the duration of \( c \), its playback will be truncated at \( c \)'s end.

Case 4 is not correct since the author gives a double, and contradictory, definition of the duration of the involved tags, thus generating a temporal conflict. Note that case 4 includes also the case in which the parallel composition has a finite duration, but contains some children with an indefinite duration, which is, by definition, longer than any other finite value.

We can apply the PAR+BEG+END rule in cases 1, 2 and 3, since the applicability conditions are satisfied. In case 1 and 2, all tags end before time instant \( \text{stop} \), by hypothesis (case 2) or since it has been chosen as the maximum value. In case 3 all media items ending after the time instant \( \text{stop} \) belong to NotDur, therefore finite\( (k2) \) \( \implies \forall c_i \text{ end}_B(c_i) \leq \text{stop} \) \( \lor \) \( c_i \in \text{NotDur} \) holds; hence the applicability condition is satisfied and the rule can be applied. The same applicability condition prevents us to apply the PAR+BEG+END rule in case 4 when a statement \( c_i \) has a finite duration longer than \( c \).

The condition finite\( (k2) \) \( \implies \neg \text{Indef}(c) \land k2 \geq k1 \) states that in presence of a finite value of \( k2 \), the rule can be applied to the statement \( c \) only if it does not contain, at any level of nesting, an item with an indefinite duration. Moreover, the tag must end after its beginning, i.e. \( k2 \geq k1 \). The applicability condition Indef\( (c) \) \( \implies (\neg \text{defined}(k2) \lor \text{indefinite}(k2)) \) states that the attribute end must be equal to ‘‘indefinite’’, or ‘‘void’’ if the statement does not end.

As already discussed at the beginning of this section, we can argue the soundness of our rules according to an implicit operational semantics which formalizes the change in the state of a player described by the SMIL specifications. Consider for instance the rule PAR+BEG+END and assume \( \sigma^0 \) be an initial state where the
precondition $A'$ holds: $\sigma^0 \models A'$. Since $A' = \bigcup_i A_i \cup \{t_{cr}(c) = \text{init}\}$, then there exist $\gamma \in \Sigma$ such that $\gamma_c = \sigma^0 + k1$ and $\forall 1 \leq i \leq n, \gamma \models A_i \cup \{t_{cr}(c_i) = \text{init} + k1\}$. Hence we can assume that each child $c_i$ starts in the state $\gamma$ and, by the correctness of the premises of the rule, any intermediate final state $\gamma_i^f$ of $c_i$, satisfies the postcondition: $\forall 1 \leq i \leq n, \gamma_i^f \models B_i$. Since the presentation of the tag par ends only if all its children end, the presentation ends in a state $\sigma^f$ whose clock value is greater (or equal) than the clock values of all the intermediate final states $\gamma_i^f$. By monotonicity of $\sigma$, we get $\sigma^f \models \bigcup_{i=1}^n B_i^f$. $B_i^f$ is different from $B_i$ only if the parallel composition has a finite value for the attribute end, and $c_i$ has not. In this case, $c_i$ ends with $\text{end}$ and, by definition, $B_i = B_i^f \cup \{\text{end}(c_i) = \text{stop}\}$, therefore the rule is correct.

Let us illustrate how our rules find out temporal conflicts like the one described in case 4, due to an author’s error which can happen when the structure becomes more complex, including a lot of tags nested one into the other. Let us consider the following tag:

```xml
<par id="p" begin="0" end="5">
  <img id="i" begin="0" end="5" />
  <text id="tx" begin="0" end="7" />
</par>
```

Even if the temporal conflict is evident since the tag is simple, (text page tx lasts more then the tag in which it is contained), we try to check the semantic correctness of this statement to show how the system works.

We would like to prove that

$$\{t_{cr}(p) = 0\} <\text{par } p \ldots> \{Q\}$$

where $Q \equiv \{\text{begin}(i) = 0, \text{end}(i) \leq 5, \text{begin}(tx) = 0, \text{end}(tx) \leq 5, \text{begin}(p) = 0, \text{end}(p) = 5\}$ but statement $p$ is not correct since rule par+begin+end (see Table V) cannot be applied. In fact, since both tx and i do not belong to the set NotDur, in order to apply the rule we would have to prove the premises:

$$S_4 \equiv \{t_{cr}(i) = 0\} \land \{\text{begin}(i) = 0, \text{end}(i) = 5\}$$
$$S_{tx} \equiv \{t_{cr}(tx) = 0\} \land \{\text{begin}(tx) = 0, \text{end}(tx) = 5\}$$

The first triple $S_4$ is valid and we can prove it by the axiom media+begin+end +dur, but we cannot prove the triple $S_{tx}$ which is not valid. Therefore the par+begin+end rule cannot be applied since the premise $S_{tx}$ cannot be verified. In this case, the answer of our tool is that the presentation contains a semantic conflict since media item tx ends at time instant 7 while its father ends at time instant 5 (see Fig. 1).

The rule which describes the semantics of the sequential composition is very similar to the par+begin+end rule since the two tags can express the same synchronization if the values of the attributes are properly defined. There are only two differences: first, the current time instant of each child is equal to the end time instant of the previous child, and not to the current time instant of the seq tag. Second, the seq statement imposes a duration equal to zero to static media items which have not a defined duration, i.e., begin$_E(c_i) = h$ and end$_E(c_i) = h$ if $c_i$ is a static media contained in NotDur. This means that they are never played in the user screen. Its complete definition can be found in [Bossi and Gaggi 2007].
So far we consider only the use of the attribute `end`, but, as already discussed for media item definition, statements can also contain an attribute `dur` whose semantics is very similar to the `end` attribute and therefore an easily translation can be obtained with the rule `cmd+begin+end+dur` illustrated in Table VI.

\[
\begin{align*}
\text{CMD+BEGIN+END+DUR} & \\
\{A\} & <\text{cmd}\ c\ \text{begin}='k1'\ \text{end}='k2'\ \text{dur}=\text{void}'>\ c_1\ldots\ c_n\ </\text{cmd}>\ \{B\} \\
\{A\} & <\text{cmd}\ c\ \text{begin}='k1'\ \text{end}='k4'\ \text{dur}=\text{void}'>\ c_1\ldots\ c_n\ </\text{cmd}>\ \{B\} \\
\end{align*}
\]

**Applicability condition:**
- \(\text{finite}(k3) \implies (\text{finite}(k2) \land k3 = k2 - k1)\)
- \((\text{defined}(k4) \implies k4 = k2) \land (\text{infinite}(k2) \iff \text{infinite}(k3))\)

**SMIL Specifications [Butlerman et al 2008]**
The attribute `dur` specifies the simple duration. If the element does not have a (valid) `dur` attribute, the simple duration \([\ldots]\) is defined to be the implicit duration of the element.

Table VI. Proof rules for a general composition of tags when the attribute `dur` is defined

### 3.2 User interactions in the attributes begin and end

SMIL language permits also the use of events as possible values for the attributes `begin` and `end` of the tags (see Table I). Let us consider first the case in which the start (or the end) of a media, or a group of media items, occurs in correspondence of an user interaction, e. g. when the user keys in a character, say ‘s’, in the keyboard as described by the following tag:

\[
<\text{cmd}\ c\ \text{begin}=\text{accesskey(s)}+k'>\ldots</\text{cmd}>
\]

where `accesskey(s)` means that the user has to key in the character ‘s’ and \(k \geq 0\) represents a number of seconds.

The correctness of this statement can be proven only if we already know the instant in which the event `accesskey(s)` takes place. According to Section 2.1, the player recorded the last occurrence of an interactive event in the state through the function \(\sigma_\tau : ((E \times Id) \cup E) \rightarrow \mathbb{R}^*\) which records the time instant in which an event \(e \in E\) on the tag \(id \in Id\) occurs. Since the player enters a new state in response to an event, for each occurrence of an event, there exists a state such that the clock value is equal to the time instant recorded from the function \(\sigma_\tau\). An interactive event may involve a single media item, or the global environment, as in the case of a digit on the keyboard. In the precondition of a statement we use the functions `times(ev, id)`, `times(ev)`, `time(ev, id)`, and `time(ev)` to constraint the input events. For instance the preconditions \{`time(ev, id) = n`\} states that the initial state in which the tag is evaluated should enable the event `ev`, i. e., its clock should be \(\leq n\). In the rules we use an uniform notation and use the argument (event) to denote both a global event (`ev`) or a couple (`ev, id`) in case of non global events, being the different notation determined by the event itself. The occurrence of the event changes the current time instant of the tag, which is now equal to the time instant in which the event takes place.


BEGIN + USER-INTERACTION

\{A \cup \{tcr(c) = n\}\}<cmd c begin=''k1'' end=''k2''> \ldots </cmd>{B}

\{A'\}<cmd c begin='event+k1' end='k2'> \ldots </cmd>{B}

where $A' \overset{def}{=} A \cup \{tcr(c) = init\} \cup \{time(event) = n\}$

APPLICABILITY CONDITION:

\{A \cup \{tcr(c) = init\} \cup \{time(event) = n\}\} \implies \{n \geq init\}

END + USER-INTERACTION

\{A \cup \{tcr(c) = init\}\}<cmd c begin='k1'> \ldots </cmd>{B \setminus \{end(c) = n+k2\}}

\{A'\}<cmd c begin='k1' end='event+k2'> \ldots </cmd>{B}

where $A' \overset{def}{=} A \cup \{tcr(c) = init\} \cup \{time(event) = n\}$

APPLICABILITY CONDITION:

\{B \cup \{time(event) = n\}\} \implies \{n \geq begin_B(c) \cup \{end_B(c) = n+k2\}\}

SMIL SPECIFICATIONS [Bulterman et al 2008]

attribute begin: Describes an event and an optional offset that determine the element begin. The element begin is defined relative to the time that the event is raised. Events may be any event defined for the host language [...] and may include user-interface events, event-triggers transmitted via a network, etc.

attribute end: Describes an event and an optional offset that determine the end value. The end value is defined relative to the time that the event is raised. Events may be any event defined for the host language [...] and may include user-interface events, event-triggers transmitted via a network, etc.

Table VII. Proof rules for SMIL statements with an interactive event in the definition of the begin or the end attribute

Table VII shows the rules to deal with statements with a begin or an end attribute which is bound to an interactive event. Let as consider again the example of an input from the keyboard:

\{A'\}<cmd c begin=''accesskey(s)+k1'' end=''k2''> \ldots </cmd>{B}

where $A' = \{A \cup \{tcr(c) = init\} \cup \{time(accesskey(s)) = keyin\}\}$. These rules state that the tag must be evaluated with reference to the time instant in which the event occurs, i.e., if \{time(accesskey(s)) = keyin\} $\in A'$, we can prove the correctness of the tag c if we can prove that

\{A \cup \{tcr(c) = keyin\}\}<cmd c begin='k1'' end='k2''> \ldots </cmd>{B}

holds. The input from the keyboard, must occur after the evaluation of the statement, represented by the value init, or after its beginning if accesskey is defined in the end attribute of a statement.

The correctness of this rule derives from the following considerations: consider an initial state $\sigma^0$ such that the precondition $A'$ holds: $\sigma^0 \models \{A \cup \{tcr(c) = init\} \cup \{time(accesskey(s)) = keyin\}\}$. By definition of consistent state of the player, $\sigma^0 = init \leq keyin$ and it exists an intermediate state $\gamma$ such that $\gamma_c = keyin$ and $\gamma(accesskey(s)) = keyin$. Hence, by monotonicity, $\gamma \models \{A \cup \{tcr(c) = keyin\}\}$ and then, by the correctness of the premises, we have that the player...
BEGIN+MULTIPLE-PLAYBACK

∀i 0 ≤ i ≤ n \{A ∪ \{tcr(νi(c)) = starti}\}<cmd νi(c) begin='k1' end='k2'>...</cmd>{B_i'}

\{A'} \langle f\rangle<cmd c begin='event+k1' end='k2'>...</cmd><f> \{∪_{i=1}^{n} B_i\}

where f ∈ \{par, excl\}

A' \overset{def}{=} A ∪ \{tcr(c) = \text{init}\} ∪ \{sequence\}

sequence = \overset{def}{=} \text{times}(\text{event}) = (\text{start}_0, \ldots, \text{start}_n)

B_i' = \begin{cases} B_i \setminus (\text{end}(\nu^i(c)) = h) & \text{if } (\text{begin}_B(\nu^{i+1}(c)) = h) \\ B_i & \text{otherwise} \end{cases}

APPLICABILITY CONDITION:

\forall (i, j) i ≤ j \implies (\text{start}_i ≤ \text{start}_j)

\land \{A ∪ \{tcr(c) = \text{init}\} ∪ \{sequence\}\} \implies \forall i (\text{start}_i ≥ tcr(c))

\land \forall (i, j) (\text{begin}_B(\nu^i(c)) ≤ \text{begin}_B(\nu^j(c))) \implies (\text{end}_B(\nu^i(c)) ≤ \text{begin}_B(\nu^j(c)))

SMIL Specifications [Bulterman et al. 2008]

event-values, accesskey-values [..] do not yield an instance time unless and until the associated event happens. Each time the event happens, the condition yields a single instance time. The event time plus or minus offset is added to the list. If the event happens multiple times during a parent simple duration, there may be multiple instance times in the list associated with the event condition.

Table VIII. Proof rule for multiple execution of the same tag

reaches a final state σ^f which satisfies the postcondition B. In this particular case, \{\text{begin}(c) = \text{keyin} + k1\} ⊂ B, which means that c starts exactly k1 time instants after the occurrence of the event accesskey(s), i.e., our rule respects the standard specifications (see Table VII).

Note that all other interactive events supported by the SMIL specifications (a partial list can be find in Table I) could be addressed in the same way as soon as the player records the time instant in which the event occurs by the function σ_t. As an example, the $\text{activateEvent}$ represents the time instant in which an user clicks on a media item, and therefore, from our point of view, it is not different from the user clicking on the keyboard. Also in this case, the only constraint is that, in order to be useful, the event must occurs after the evaluation of the statement.

We may note here that an interactive event may occur more than once, e.g., the user may click several times on a button. This means that an object which binds its start to this event may play more than once. As discussed in Section 2.1, in order to record all the execution of an item, we consider multiple playback of the same item as new objects with new names. The rule BEGIN+MULTIPLE-PLAYBACK\(^1\) (see Table VIII) models this situation: multiple executions of the same item are correct as soon as each single playback is correct. We note here that, if an event occurs before the end of the media, it is immediately stopped and restarted\(^2\). In this case, we do not need to prove the end time instant of the playback in the premises as stated by the rule.

Let us consider the previous example where the user keys in the digit ‘s’ twice,

\(^1\)We consider only positive offsets since the rule deals only with interactive events.

\(^2\)We suppose that the attribute $\text{restart}$ assumes the default value always.
at time instants $\text{start}_1 < \text{start}_2$ and let $A' = \{A \cup \{\text{times}(\text{accesskey(s)}) = (\text{start}_1, \text{start}_2)\}\}$. We can apply the rule \text{BEGIN+MULTIPLE-PLAYBACK} to prove
\[
\{A'\}\text{<cmd c begin='\text{accesskey(s)}+k1' end='k2'>...<cmd>}{B}
\]
if we can prove the correctness of both the executions of the tag $c$
\[
\{A_1\}\text{<cmd c begin='\text{accesskey(s)}+k1' end='k2'>...<cmd>}{B'_1}
\{A_2\}\text{<cmd }v(c)\text{ begin='\text{accesskey(s)}+k1' end='k2'>...<cmd>}{B'_2}
\]
where $v(c)$ is a new name for the second execution of $c$, $A_1 = \{A \cup \{t_{cr}(c) = \text{start}_1\}\}$, $A_2 = \{A \cup \{t_{cr}(v(c)) = \text{start}_2\}\}$.

If the interactive event appears in the definition of the end attribute, we need different considerations. While multiple occurrences of an event defined in the begin attribute cause multiple starts, therefore multiple playbacks of a tag, this is not true for multiple occurrences of an event defined in the end attribute. If an item starts only once it must also end only once. Therefore, the player will end the item as soon as the event occurs for the first time, subsequent occurrences of the same events are ignored. The rule \text{END+USER-INTERACTION} (Table VII) can be used in this case, since any state which satisfies the first element of a sequence of event times satisfies, by definition, also the sequence. The correctness of the rule derives from the fact that subsequent occurrences of the same event must be ignored.

A particular case regards media items which contain an interactive events both in the begin and in the end attributes. Rule \text{BEGIN+END+MULTIPLE-PLAYBACK} (see Table IX) is very similar to the case with a unique event in the begin attribute and could be used to prove the correctness of each single execution of the item. The only relevant differences are contained in the applicability conditions: the two sequences must be ordered and the sequence of starts must have a length equal or greater than the sequence of end events. Moreover, the last event must be an occurrence of the stop event.

### 3.3 Management of non interactive events

Another possibility offered by the SMIL standard is to bind the begin (or end) event of a (group of) media item $m$ with the begin (or end) event of another (group of) media item $n$. As already discussed in Section 2.1, non interactive events are recorded differently in the state of the player, since they are not traced by the function $\sigma_\tau$, but by function $\sigma_\delta$. As an example, consider the tags

\[
\text{(1)}\quad \text{<par id="p" end="au.end">}
\quad \text{<audio id="au" />}
\quad \text{<text id="tx" />}
\quad \text{</par>}
\]

\[
\text{(2)}\quad \text{<cmd id="m" begin="n.begin+5"/>}
\]

In case (1), the whole par statement ends when media item au ends; in case (2) media item m begins 5 seconds after the beginning of n.
Automatic validation of SMIL documents

\begin{align*}
\{A\}' & <\text{cmd c begin='k1'}> \ldots </\text{cmd}\{B\}' \quad n > 1 \implies \{A\}'' \nu(c) \{B\}'
\end{align*}

where

\begin{align*}
A' & \overset{\text{def}}{=} A \cup \{t_{\nu}(c) = \text{start}_1\} \\
A'' & \overset{\text{def}}{=} A \cup \{t_{\nu}(c) = \text{init}\} \cup \{\text{starts}\} \cup \{\text{stops}\} \\
\text{starts} & \overset{\text{def}}{=} \text{times(event}_1) = (\text{start}_1, \ldots, \text{start}_n) \\
\text{stops} & \overset{\text{def}}{=} \text{times(event}_2) = (\text{stop}_1, \ldots, \text{stop}_m) \\
B & \overset{\text{def}}{=} B_1 \cup B'' \\
B_1 & = \begin{cases} 
B_1 \setminus \{\text{end}(c) = h\} & \text{if } (\text{begin}_B(\nu(c)) = h) \\
B_1 & \text{otherwise}
\end{cases} \\
\text{starts'} & \overset{\text{def}}{=} \text{times(event}_1) = (\text{start}_2, \ldots, \text{start}_n) \\
\text{stops'} & \overset{\text{def}}{=} \text{times(event}_2) = (\text{stop}_2, \ldots, \text{stop}_m) & \text{if } (\text{end}_B(c) = \text{stop}_1 + k2) \\
& \text{if otherwise} \\
A'' & = A \cup \{t_{\nu}(\nu(c)) = \text{init}\} \cup \{\text{starts'}\} \cup \{\text{stops'}\}
\end{align*}

\textbf{Applicability condition:}

\begin{align*}
m \leq n & \land (\forall(i, j), i \leq j \implies \text{start}_i \leq \text{start}_j) \land (\forall(i, j), i \leq j \implies \text{stop}_i \leq \text{stop}_j) \\
\land \ A' \cup B & \implies \ (\text{stop}_i \geq \text{begin}_B(c) \land (\text{end}_B(c) = \text{stop}_i + k2) \lor (\text{end}_B(c) = \text{begin}_B(\nu(c))))
\end{align*}

\textbf{SMIL specifications [Bulterman et al 2008]}

Event-values, accesskey-values \ldots do not yield an instance time unless and until the associated event happens. Each time the event happens, the condition yields a single instance time. The event time plus or minus any offset is added to the list. If the event happens multiple times during a parent simple duration, there may be multiple instance times in the list associated with the event condition.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\textbf{Tag} & \\
\hline
\text{<par id="p" end="stop">} & \text{Therefore the following rule can be applied to tag \text{(1)}.} \\
\hline
\end{tabular}
\caption{Table IX. Proof rule for media items which contain an interactive events both in the \text{begin} and \text{end} attributes}
\end{table}

Tag (1) can be considered similarly to the case of interactive events: if we already know (from the premise) the end point of \text{au}, i. e. \text{end(au) = stop}, we can then analyze the tag \text{<par id="p" end="stop">}\ldots</\text{par}>. Therefore the following rule can be applied to tag (1).

\begin{align*}
\text{PAR+END+EVENT} \\
\{A\} & <\text{par c end='stop \pm k'>c_1 \ldots c_n</par}\{B\} \\
\{A\} & <\text{par c end='ci.end \pm k'>c_1 \ldots c_n</par}\{B\}
\end{align*}

\textbf{Applicability condition:}

\begin{align*}
\text{end}_B(c) & = \text{stop} \land \text{begin}_B(c) \leq \text{stop} \pm k
\end{align*}

The situation is more complex in case (2), which cannot be analyzed singularly since its evaluation needs information about the \text{begin} of media item \text{n}. For this reason we must consider a set of media items as shown by the following rule:
BEGIN+EVENT

\{A\}<cmd c>\ldots c_i \ldots c_j'\ldots</cmd>\{B\}
\{A\}<cmd c>\ldots c_i \ldots c_j\ldots</cmd>\{B\}

where \(c_i, c_j\) and \(c_j'\) are related as follows: there exist \(n, m\) such that \(n \in \text{Closure}(c_i), \ m \overset{\text{def}}{=} <cmd id="m" begin="n.begin+k">\ldots</cmd> \in \text{Closure}(c_j), \ c_j'\) is obtained from \(c_j\) by replacing \(m\) with \(<cmd m begin="begin_B(n)+k">\ldots</cmd>\).

A particular attention is required if there are multiple executions of the media item \(n\) due to user interactions. Let us consider as example, a video \(v\) activated by a click on an image \(i\), and its associated soundtrack \(a\):

<par>
<img id = "i" />
<video id = "v" begin="i.activateEvent"/>
<audio id = "a" begin="v.begin"/>
</par>

In this case, even the soundtrack \(a\) must be played several times. If we try to prove the correctness of this parallel composition, we need to analyze any single component. In particular, by applying the rule BEGIN+MULTIPLE-PLAYBACK to video \(v\), we obtain the set \(\{v, \nu(v), \ldots \nu^h(v)\}\) of activations of the same video but we have to consider also the set of associated executions of \(a\), \(\{a, \ldots, \nu^h(a)\}\).
To obtain that, we must refine the BEGIN+EVENT rule by requiring that \(c_j'\) is obtained from \(c_j\) by expanding \(m\) with all its executions \(\nu_i(m)\). Each \(\nu_i(m) \overset{\text{def}}{=} <cmd \nu_i(m) begin="begin_B(\nu_i(n))+k">\ldots</cmd>\) is associated to a playback of \(n\), as stated in the postcondition: \(\forall i \{\text{begin}(\nu_i(n)) = \text{value}_i\} \in B\).

3.4 Multiple values for the attributes begin or end

The SMIL standard allows to define an unordered list of value for the attribute begin and end. This list may contain, separated by a semicolon, a list of events, or a time instant. In this case, the tag, respectively, starts, or ends, as soon as one of the events, contained in the list, occurs, or the player’s clock reaches the time instant if defined. Then, each time an event in the list occurs (or the time instant is reached), it is restarted. Rule CMD-BEGIN-VALUE-LIST-END (see Table X) describes the behavior of a tag with a list of values for the attributes begin.
This rule simply states that, if the tag \(c\) can be formally proved to be correct for each possible occurred event in the list, or for the time instant, if defined, it is correct also for the entire list. The obtained set of postcondition \(B\) is the sum of all the executions.

3.5 The excl tag

SMIL language provides also a tag for the exclusive composition of media items, i.e., the tag excl, whose semantics states that only one of its children is active at any given time instant. Therefore, this tag is very similar to the sequential composition since, even in this case only one child is active at a time, but excl does not impose any order in the visualization of the children. This means that each child may
Automatic validation of SMIL documents

\[ \text{finite}(k_1) \Rightarrow \{ A' \} c_{k_1} \{ B'_{k_1} \} \quad \forall i \{ A \cup \{ \text{trigger}(c_i) = \text{init} \} \} \nu^i(c) \{ B' \} \]

\{ A' \} <f><cmd c begin='event-list' end='k_2' dur='void'>... </cmd></f> \{ B \}

where \( f \in \{ \text{par}, \text{excl} \} \)

\[ \forall i \nu^i(c) \begin{array}{c} \begin{array}{c} \text{def} \end{array} \end{array} <cmd \nu^i(c) \begin{array}{c} \begin{array}{c} \text{begin}=' \text{event}_i' \end{array} \end{array} \text{end}='k_2' \text{dur}='void'>... </cmd> \]

\[ c_{k_1} \begin{array}{c} \begin{array}{c} \text{def} \end{array} \end{array} <cmd c_{k_1} \begin{array}{c} \begin{array}{c} \text{begin}='k_1' \end{array} \end{array} \text{end}='k_2' \text{dur}='void'>... </cmd> \]

\[ A' = A \cup \{ \text{trigger}(c_{k_1}) = \text{init} \} \]

\[ B = \begin{cases} \bigcup_i B_i \cup B_{k_1} & \text{if } \text{finite}(k_1) \\ \bigcup_i B_i & \text{otherwise} \end{cases} \]

\[ B'_{k_1} = \begin{cases} B_{k_1} \setminus \{ \text{end}(\nu^i(c)) = h \} & \text{if } \exists j (\text{begin}_B(\nu^j(c)) = h) \\ B_{k_1} & \text{otherwise} \end{cases} \]

\[ B'_{i} = \begin{cases} B_i \setminus \{ \text{end}(\nu^i(c)) = h \} & \text{if } \exists j (\text{begin}_B(\nu^j(c)) = h) \\ B_i & \text{otherwise} \end{cases} \]

Applicability Condition:

\[ \text{finite}(k_1) \lor (\exists i \text{time}(<\text{event}_i> = n_i \land \text{finite}(n_i)) \]

SMIL Specifications [Bulterman et al 2008]

A semi-colon separated list of begin values. [...] In general, the earliest time in the list determines the begin time of the element. [...] Each element can have a begin attribute that defines one or more conditions that can begin the element. [...] In order to calculate the times that should be used for a given interval of the element, we must convert the begin times and the end times into parent simple time, sort each list of times (independently), and then find an appropriate pair of times to define an interval.

Table X. Proof rule for SMIL tags containing multiple values in the begin attribute

contain the attribute begin in the definition, or may be activated by the user, e.g. following a link. Let us consider the following example:

\[
\text{<par>}
\text{\hspace{1cm} <img id = "a" /> <img id = "b" /> <img id = "c" />}
\text{\hspace{1cm} <excl id="e" dur="10">}
\text{\hspace{1cm} \hspace{1cm} <video id = "video_a" begin="a.activateEvent"/>}
\text{\hspace{1cm} \hspace{1cm} <video id = "video_b" begin="b.activateEvent"/>}
\text{\hspace{1cm} \hspace{1cm} <video id = "video_c" begin="c.activateEvent"/>}
\text{\hspace{1cm} </excl>}
\text{\hspace{1cm} </par>}
\]

in this case, the user chooses a video clip by clicking on an image button chosen between media items a, b and c. The corresponding video is activated by the proper activateEvent. The excl tag simply states that only one video clip plays at a time: in fact, the video currently playing is stopped when the user clicks on another image, choosing another video clip.

The example shows how the excl command does not deal with the activation of its children but with their deactivation; in fact, the playback order of the video clips completely depends on the user choices and not on the tags’ definitions.
EXCL-BEGIN-END

\[ A \cup \{ t_{cr}(c_i) = init + k1 \} \} \cap \{ B'_i \} \quad \forall i 1 \leq i \leq n \]

\( \{ A' \} <excl c begin='k1' end='k2'dur='void'> c_1 \ldots c_n </excl> \{ B \} \)

where

\[
A' = A \cup \{ t_{cr}(c) = init \} \]

\[
B = \bigcup_{i=1}^{n} B_i \cup \{ begin(c) = init + k1 \} \cup \text{End} 
\]

\[
\text{stop} = \begin{cases} 
\text{init} + k2 & \text{if finite(k2)} \\
\max_{i \in \text{stopB}(c_i)} & \text{if } \text{Indef(k2)} 
\end{cases} 
\]

\[
\text{End} = \begin{cases} 
\text{end(c) = stop} & \text{if } \neg \text{Indef(c)} \\
\emptyset & \text{otherwise} 
\end{cases} 
\]

\[
B'_i = \begin{cases} 
B_i \setminus \{ \text{end}(c_i) = t_i \} & \text{if } \exists j \begin{cases} \text{beginB}(c_i) = t_i \cup \text{(finite(k2)} \land c_i \in \text{NotDur}) \\
B_i & \text{otherwise} 
\end{cases} 
\end{cases} 
\]

APPLICABILITY CONDITION:

\[
\text{finite(k2)} \implies \neg \text{Indef(c)} \land \text{k2} \leq k1 \\
\land \text{Indef(c)} \implies (\neg \text{defined(k2)} \lor \text{Indefinite(k2)}) \\
\land \text{finite(k2)} \implies \forall i \begin{cases} \text{endB}(c_i) \leq \text{notDUR} \lor c_i \in \text{NotDur} \\
\land \forall (i) \begin{cases} \text{beginB}(c_i) \leq \text{begin} + k1 \\
\land \forall (t) \begin{cases} \text{beginB}(c_i) \leq \text{beginB}(c_i) \implies \text{endB}(c_i) \leq \text{beginB}(c_i) \end{cases} 
\end{cases} 
\end{cases} \]

SMIL SPECIFICATIONS [BULTERMAN ET AL 2008]

SMIL 3.0 defines a time container with semantics based upon par, but with the additional constraint that only one child element may play at any given time. If any element begins playing while another is already playing, the element that was playing is stopped. [...] The implicit synchase of the child elements of the excl is the begin of the excl. The default value of begin for children of excl is "indefinite". This means that the excl has 0 duration unless a child of the excl has been added to the timegraph.

Table XI. Proof rule for the exclusive composition when the attribute \text{dur} is equal to \text{void}

The semantics of the excl tag is described in Table XI by rule EXCL-BEGIN-END. Like tags \text{par} and \text{seq}, the excl tag begins at its current time instant, or after \text{k1} time instants if the attribute \text{begin} is finite, and ends, when there are no children playing. This means that, it can have an instantaneous duration if no child starts together with it. For this reason, the attribute \text{end} of this statement usually does not contain the special value ‘‘\text{void}’’.

The EXCL-BEGIN-END rule is very similar to the rule which describes the semantics of parallel composition, therefore we do not repeat here the problem of the termination of the tag. Even in this case, to prove the correctness of the statement \text{<excl> c}_1 \ldots c_n \text{</excl>}, each \text{c}_i must be proven to be correct, assuming as its current time instant the current time instant of its father. Since the exclusive tag may impose a premature stop of the playback of its children, in some cases, we do not require to know, in the premises, the time instant in which the child \text{c}_i ends:

(1) when \text{c}_i ends together with excl if it does not contain the attribute \text{end} or \text{dur} in its definition (i.e k2 is finite and \text{c}_i \in \text{NotDur});

(2) when the playback of \text{c}_i is stopped before its termination due to the user interaction or some other external event (i.e. \exists j \begin{cases} \text{beginB}(c_i) = t_i \end{cases}.

The applicability condition prevents the application of the rule in presence of temporal conflicts. Among the conditions already discussed for the parallel composition, the condition \forall \text{c}_i, \text{c}_j \begin{cases} \text{beginB}(c_i) \leq \text{beginB}(c_j) \end{cases} \implies \text{endB}(c_i) \leq
Fig. 1. Screenshot of the tool

\begin{itemize}
\item \textit{begin}$_B$(\textit{cj})) states that only one child plays at any given time instant, i. e., if child \textit{ci} begins before \textit{cj}, it also ends before \textit{cj}'s beginning.
\item We note here that each child can be played several times, since their executions are usually driven by user interaction, e. g., in the previous example the user may click more than once on the images \textit{a}, \textit{b} and \textit{c}. This situation is solved by applying the rule \textit{begin+multiple-playback} (see Table VIII) to the repeating media item.
\end{itemize}

4. DESCRIPTION OF THE TOOL

Based on the formal semantics described in the previous section, we have implemented a tool for the semantic validation of a SMIL document. Since an uncorrect multimedia presentation cannot be rendered properly, our tool allows us to check its consistency during all the authoring phases, when the author asks it or saves her/his work. We do not consider dynamic checking a good solution since temporary inconsistencies, due to the \textit{work-in-progress}, should be allowed, while we must guarantee the correctness of the final result. This choice is also cost-effective.

The Semantic Validator implemented has two goals: to assist the user in the complex task of authoring a multimedia presentation, automatically finding temporal inconsistencies and helping their correction, and to produce the sequence of playbacks of media items contained in a SMIL file to be used for the presentation playback. For this reason, our implementation keeps separated the Semantic Validator Module, which is the engine that can be used both by a player and an authoring
system, from the interface for the automatic verification of SMIL file. We note again that we do not want to realize a new authoring system (a number of which has been implemented offering different useful facilities like visual editors, preview window, etc.) but a new tool which can be used in conjunction with an authoring system to help the user to design correct SMIL documents, since this facility is still absent.

The interface has been implemented using the Java language with the goal to test the engine, realized with the Prolog language, and to support multimedia authoring by pointing out conflicting values in the document. Figure 1 shows a screenshot of the tool. The user selects a SMIL file to evaluate and the Semantic Validator checks its syntactic correctness and then displays it, empathizing tags and attributes. As second step\(^3\), the tool asks to the user to input the preconditions, i.e., the natural duration of the continuous objects, the user interactions and the tag to evaluate. The user can ask for the validation of the entire document, by selecting the first tag of the document, or only a single tag; in any case, he/she must give as input the time instant in which the tag should be evaluated. Then, the validation process can be started and returns as output the postconditions of the selected tag. If any temporal inconsistency is found, the tool prompts a message containing the tag which contains the error and its motivation (see Fig. 1). This information allows the user to easily detect and correct the error; if it is not sufficient, the tool provides also a "step by step" modality by clicking on the “Step Into” button, which shows, for each interaction, a single step of the process, displaying in the source code (respectively before and after the tag itself) the preconditions and the postconditions of each analyzed tag. This second modality allows a better understanding of the overall process and of the context in which a single tag is evaluated. This is particularly useful in the case of SMIL tags which refer to user interaction in their definition: in this case only events that occurs after their evaluation should be considered.

A panel, positioned below the SMIL code, contains the tool’s messages to the user, i.e., errors or warnings (e.g., a tag with a duration equal to zero), and the last used rule. Expert users can visualize the last used rule by clicking on the

\(^3\)The interface buttons are activated/disactivated in order to guide the user to a correct sequence of interactions.
button “Show Rule” which shows the corresponding table of this paper (Fig. 2).

If the document is correct, the validation process returns as output the correct sequence of start and stop events of all media items involved in the presentation (Fig. 3). This information can be used obviously by a player for the playback of the presentation, but is also useful for the implementation of a preview window in an authoring system. We note here that the use of our Semantic Validator in conjunction with an authoring system is particularly important since the composition of a SMIL document driven by the rules is correct by construction. The analysis of a tag finds out a temporal conflict if the construction of the proof fails because one of the needed premises cannot be proved or the applicability conditions are not satisfied. The compositionality of our approach helps the user to correct the error and to incrementally continue the analysis.

5. CONCLUSIONS AND RELATED WORK

In this paper we consider the problem of automatic verification of SMIL documents and present a tool which can assist the user in the complex task of authoring a multimedia presentation. The tool is based on a formal semantics defining the temporal aspects of SMIL tags by mean of a set of inference rules inspired by the Hoare’s semantics, which describe how the execution of a piece of code changes the state of the computation of a player.

The paper mainly focuses on SMIL 2.1 features but, since only temporal aspects are taking into account and SMIL 3.0 specification [Bulterman et al 2008] leaves the basic syntax and semantics of the SMIL 2.1 timing model unchanged, it also applies to the latest version.

We remark that we do not aim at implementing a new authoring system, but a module to assist the user in finding and fixing temporal conflicts in SMIL documents: although many tools have been implemented since SMIL first definition, this facilities is usually still absent. The main advantages of this work are the following:

— it assists multimedia authoring by pointing out conflicting temporal values in the document;
— it allows for a modular evaluation of the tags nested in a SMIL document and helps the context adaption process;
— it minimizes the set of preconditions needed to evaluate a SMIL tag;
— the compositionality of the approach allows for an easy extension of SMIL features actually considered.

It’s worth noting that all the rules of the proof system can be used both for a top-down construction of a correct playback sequence of the media items involved in the multimedia presentation and for a bottom-up analysis of the SMIL document. This second feature is particularly useful during the context adaption of a document to find out a suitable candidate for substitution or, more in general, during the authoring of the document by composition of tags. Moreover, our rules help in discovering the weakest precondition, i.e., the minimal set of requirements needed to evaluate a tag. In our system this set contains the natural duration of continuous media, the syncbase of the tag, which is equal to zero by standard convention for the outer-most tag, and information about user interactions.
The choice of a semantics inspired by the Hoare logic as basis for the formalism allows us to incrementally extend the subset of SMIL features implemented. New features are added by defining new rules to describe the semantics of a particular tag or attribute, or by defining a translation to a more simple situation, e.g. the \texttt{cmd+begin+end+dur} translates a tag containing all the attributes \texttt{begin}, \texttt{end} and \texttt{dur} into an equivalent tag without the attribute \texttt{dur}.

The soundness and completeness of our approach has been discussed according to an operational semantics which formalizes the changes in the state of a player described informally in the SMIL recommendation. Indeed the tool passed the test provided by W3C in the SMIL Testsuite [Chang and Michel].

The problem of finding out temporal conflicts into SMIL documents has been already considered in literature. In [Sampaio et al. 2000; Sampaio and Courtiat 2004; Valente and Sampaio 2007], Sampaio et al describe RT-LOTOS, a formal description of SMIL tags which enables the generation of a valid scheduling for its rendering, considering QoS problem. The authors do not aim at defining a semantics for SMIL language, but compare different players’ behaviors which are still implementation-dependent.

Yang [Yang 2000] and Yu [Yu et al. 2002] proposes the use of Petri Nets to describe the temporal evolution of a SMIL document. Yang translates the SMIL synchronization tags into transitions and places of the \texttt{Real Time Synchronization Model (RTSM)} and tries to detect possible temporal conflict, but this work is limited to the features of SMIL 1.0. Yu defines a formalism based on Petri Nets named \texttt{SAM (Software Architecture Model)} which aims to check if QoS properties, expressed through logical formulas, are satisfied, and not to the verification of the semantic correctness of the SMIL document.

The only real attempt to define a formal semantics for SMIL is presented in [Jourdan 2001]. This approach is based on the use of timed automata and has been used during the design of SMIL 2.0 to improve specification, since the author was a co-editor of the document which describes timing and synchronization features of this language. The work presented in this paper mainly focuses on SMIL 1.0 and take into account only two new features of SMIL 2.0.

Other works address the problem of temporal consistency of multimedia documents not described with SMIL language. Among others, Elias [Elias et al. 2006] presents an algorithm, based on the graph theory, which is able to dynamically maintain a consistent and complete set of constraints during the authoring phase. Other works address the same problem with constraints solver techniques. Differently from our approach, all the works addressed here require to translate the SMIL document into another formalism, e.g., a set of temporal constraints or a Petri Net, in order to check its temporal consistency. This operation is not always cost-effective, especially when the complexity of the input file increases and a non compositional approach is used.

Finally, since most available authoring systems adopt “... SMIL language for building the final representation scheme” [E.Bertino et al. 2005], we argue that a formal semantics for this language is needed.
ACKNOWLEDGMENTS

The authors would like to thank Mattia Boldrin for his support in the development of the tool and the helpful discussions.

REFERENCES


Chang, W. and Michel, T. SMIL 2.0 Testsuite.


Patrick Schmitz and Jeff Ayars and Bridie Saccocio and Muriel Jourdan. 2001. The SMIL 2.0 Timing and Synchronization Module.


