Formal Timing Analysis Of Mixed Music Scores

Léa Fanchon, Florent Jacquemard

To cite this version:


HAL Id: hal-00829821
https://inria.hal.science/hal-00829821v2
Submitted on 12 Jun 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
FORMAL TIMING ANALYSIS OF MIXED MUSIC SCORES

Léa Fanchon, Florent Jacquemard

Ircam & INRIA, Paris, France
lea.fanchon@ircam.fr florent.jacquemard@inria.fr

ABSTRACT

Interactive music systems coordinate in real-time an artificial perception of dynamics of human musicians with timely execution of machine reactions. As every human performance will differ from another, it is a challenging task to be able to predict the behavior or such systems in response to any possible performance, and prevent unwanted outcomes.

We present here the application of formal models and methods from the real-time systems verification literature to the static analysis of interactive music systems. We consider in particular the good parameters problem, which consists in synthesizing a set of timing parameter valuations (representing performances here) guaranteeing a good behavior of the system analyzed. The methods presented have been applied to the system Antescofo, and are general enough to apply to other interactive music systems.

1. INTRODUCTION

There are two dimensions in authoring time and interaction for interactive music systems, corresponding to two main processes [23]. The first process is the composition, or programming i.e. the specification of mixed scores containing both instrumental and electronic parts and instructions for their synchronization. The second is the performance: The real-time evaluation and synchronization of the parts. We consider here the case of the score follower Antescofo [19] which aims at addressing these two processes, enabling live interactive performances of written music between computers and human musicians. It relies on two subsystems: on the one hand, a listening machine [11] tracks in real-time both the position of a musician in the score and his current tempo, and on the other hand a reactive engine, relying on a dedicated synchronous language, ensures the real-time coordination of the electronic part.

Such a system can be qualified as time critical. Like in other embedded systems e.g. for transportation or communication, time is not only a measure of efficiency but a real semantical issue: The objective is not to compute as fast as possible but to produce some actions at the right moment [18]. Moreover, such a system is also reactive and open: during a performance, it has to react in real-time to triggering events (notes) coming from an unpredictable environment (the musicians). Indeed, whereas the duration of every note is specified precisely in a music score, any two human interpretations of the same piece will differ, regarding timing values. Such a behavior is considered as a part of the definition of musical performance [19]. But from the computer scientist point of view, it can just be considered as unpredictable and must be dealt with. For example, some slight changes in the delays of the instrumental events may change the curse of electronic actions, e.g. by swapping two actions whose ordering ought to be preserved (see Example 3 below). In this context, it is a challenging task to be able to predict before a concert whether an interactive system such as Antescofo will react as expected to any human performance. Even intensive rehearsing with musicians cannot cover all cases and computer assistance is required for a complete timing analysis.

As part of the effort that has been made for the development of reliable embedded systems, several verification approaches have been developed in the past years for verifying statically that a system satisfies qualitative and quantitative temporal properties [5, 6]. Timed automata (TA) [1] are a central abstract model of real-time systems in these approaches. They are finite state automata extended with clocks: real-time variables increasing uniformly and which can be compared to fixed timing delays in guards of the automata’s transitions. Moreover, several TA can be composed in a network (NTA), each component automaton modeling a building block of a complex system. Synchronization of automata is then performed by rendezvous on the symbols of transitions.

In this paper, we present the application of some TA based formal methods to the timing analysis of mixed music scores in a language close to the one used in Antescofo (presented in Section 2). The similarity between augmented scores and NTA is clear: each part of a mixed score can be represented by a TA, and triggering events can be used for synchronization by rendezvous. A procedure of compilation of scores into NTA has been implemented and is presented in Section 3.

Uncertainty on the timing delays in a human performance can be modeled using parametric delays (i.e. unknown constants) in the transitions’ guards of the TA modeling the instrumental score. This corresponds to the model of parametric timed automata (PTA) [2]. In Section 4, we show how to apply to the analysis of mixed scores some techniques of parameter synthesis for PTA originally developed for the verification telecommunication protocols.
and asynchronous circuits, e.g. [3]. The good parameters problem, consists, given a FTA \( \mathcal{A} \), in inferring a set of parameters values for which \( \mathcal{A} \) behaves well [16]. Such a set is presented by a linear constraint on parameters.

Finally, we propose in Section 5 another approach which can be used in a computer assisted composition process in order to find appropriate timing values for the delays associated to electronic actions. Assuming that the delays for instrumental events have been fixed, and that a constraint expresses the expected ordering of events and actions, a valuation of the actions’ delays is chosen in a way that it maximizes a measure of the robustness of the mixed score to the timing variations in a musician’s performance.

2. MIXED SCORES SPECIFICATION LANGUAGE

Antescofo uses for the specification of mixed scores a textual language comparable to synchronous languages for programming real-time embedded systems such as Esterel [17]. Programs in this language describe both instrumental and electronic parts of the score and the instructions for their coordination during a performance. We present in the following grammar the syntax of a simplified fragment of this language, highlighting aspects relevant to the verification techniques presented below.

2.1. Mixed Score Syntax

A score is defined as an interleaving of instrumental events, to be recognized by Antescofo’s listening machine, and of the actions or groups of actions triggered by the former.

\[
\text{score} := (\text{event } d) \mid (\text{event } d) \text{ score} \mid (d \text{ comp}) \text{ score}
\]

\[
\text{comp} := \text{action} \mid \text{group} \mid \text{loop}
\]

\[
\text{group} := \text{group } \ell \{\text{list}\}
\]

\[
\text{loop} := \text{loop } \ell \{\text{list}\} \text{ until} (\text{event } \mid \text{action})
\]

\[
\text{list} := \text{comp} \mid \text{comp list}
\]

Antescofo can handle several kinds of events: notes, chords, trills, glissandi etc which are all abstracted here in symbols denoted \( e_0, e_1, \ldots \) in an an assumed predefined set \( E \).

Actions correspond to specific commands sent to the audio environment (a sound synthesis module or real-time audio processing system such as Max/MSP [21] or Pure Data [22]), and are represented as symbols \( a_0, a_1, \ldots \) in a set \( A \). We consider also a set \( L \) for group labels (\( \ell \) in the grammar).

Every event (resp. action or group) is associated with a duration (resp. a preceding delay) \( d \) specifying a number of beats (relative time). This value can be converted into physical time (i.e. in seconds) using the tempo detected in real-time by the listening machine as explained in introduction. For now, let us assume rational valued delays \( d \in \mathbb{Q}_{>0} \). The extension to delay expressions with variables is discussed later.

2.2. Idealized Performance

 Atomic actions can be sequentially composed into groups comparable to staves in traditional scores. Intuitively, a group \( g \) is triggered by the last event \( e \) occurring before \( g \) in the score \( \mathcal{S} \). If \( g \) occurs immediately after \( e \) in \( \mathcal{S} \), and is associated a delay \( d \) (denoted \( (d \ g) \) in the score) then after the detection of \( e \), the system will wait for \( d \) time units (relatively to the detected tempo) before launching the group \( g \), which consist in playing the sequence of elements contained in the group in the specified order. If \( g \) is followed by a group \( g' \) with delay \( d' \) in the score, then after launching \( g \), the system waits \( d' \) time units and launch \( g' \) concurrently. Groups can be nested, and the above interpretation of successive elements in the score is transposed recursively to nested groups.

There are several ways to define the termination of loops in Antescofo language. Here, we assume for simplifying that a loop is associated to a terminating event or action \( \kappa \). The loop’s body will be iterated until it is killed, as soon as \( \kappa \) is received. The semantics of the execution of groups and loops will be made formal in Section 5.4.

Let us first illustrate it in the following small example.

**Example 1.** We consider as a running example a trivial score containing 3 events \( e_1, e_2, e_3 \) and 3 groups \( g_1, g_2, g_3 \) (Figure 1). The groups \( g_1 \) and \( g_2 \) are triggered by the event \( e_1 \) and \( g_2 \) by \( e_2 \). Like \( g_1 \), \( g_2 \) is launched without delay, and contains one atomic action (an initialization message "init" for the former and a command for switching the lights on for the latter). The group \( g_3 \) is launched concurrently to \( g_1 \) after a delay of 0.25 time unit, and contains two elements: a message "msg" and a command for switching lights off.

This behavior is illustrated in Figure 2 assuming the musician plays the events \( e_1, e_2 \) and \( e_3 \) with delays conforming strictly to those specified in the score of Figure 1. \( \delta_1 = \delta_2 = 1.0 \) time units (we call this an idealized performance below). In this case, the lights are first switched off at time unit 1.25 (by group \( g_3 \)), and switched back on 0.25 time units later (by \( g_2 \)).
init \[ \begin{array}{ccc}
  e_1 & \delta_1=0.7 & \rightarrow & e_2 \\
  \delta_2=0.9 & \rightarrow & e_3 \\
  g_1 \quad 0.5 & \quad \text{init} & \quad g_2 \quad 0.5 & \quad \text{on} \\
  \quad 0.25 & \quad g_3 \quad 0.50 & \quad \text{msg} & \quad 0.5 & \quad \text{off}
\end{array} \]

Figure 3. Unexpected performance for the score of Fig. 1.

\begin{itemize}
  \item \( e_1 \quad 1.0 \)
  \item \( 0.0 \quad \text{group } g[\text{tight}] \quad \{ (0.5 \text{ init}) (1.0 \text{ on}) \} \)
  \item \( 0.25 \quad \text{group } g_3 \quad \{ (0.5 \text{ msg}) (0.5 \text{ off}) \} \)
  \item \( e_2 \quad 1.0 \)
  \item \( e_3 \quad 1.0 \)
\end{itemize}

Figure 4. Score equivalent to Fig. 1 with a tight group.

### 2.3. Group Attributes

In the language of Antescofo, the groups can be assigned attributes defining specific strategies of synchronization with events or for error handling, etc see [12]. For the sake of conciseness, we shall consider here one option of synchronization strategy: if a group has attribute "tight", each of its elements is automatically realigned to the latest event scheduled before it (instead of the latest event occurring before it in the score, like for groups without this attribute).

**Example 2.** The score of Figure 4 is almost equivalent semantically to the one of Example 1: the "tight" group \( g \) is played like it was split into the two groups \( g_1 \) and \( g_2 \) triggered respectively by \( e_1 \) and \( e_2 \). Note that the delay of 0.5 associated to the action "on" in group \( g_2 \) corresponds to the remaining of the original delay 1.0 associated to "on" in \( g \) after the occurrence of \( e_2 \).

Using a "tight" attribute permits to express that a large group is implicitly split into smaller groups synchronized to the nearest detected events. It can be useful \( e.g. \) in a context of automatic accompaniment, where the electronic (accompaniment) part has to follow closely the leading instrumental voice. This feature has been added to Antescofo in order to prevent the composer from the burden of having to manually segment the accompaniment part in small groups, keeping a high-level view of the grouping structure of the score [12].

### 2.4. Musical Performances

By definition, in a musical performance (as opposed to the idealized performance presented in Example 1), the delays between events should always diverge from those specified in the score. This may lead to unexpected behaviors for the execution of actions, like \( e.g. \) unwanted inversions as illustrated in the next example.

\[ \delta = (L, l_0, F, \Sigma, inv, T) \]

where \( L \) is a finite set of locations, \( l_0 \in L \) is the initial location, \( F \subseteq L \) is the set of final (recognizing) locations, \( \Sigma \)

---

1. The difference is explained in Example 6.
is a finite alphabet of symbols, \( \text{inv} : L \rightarrow \mathcal{C}(X, \Delta) \) assigns an invariant to each location (condition on clocks to be allowed to stay in this location), and \( T \subseteq L \times \mathcal{C}(X, \Delta) \times \Sigma \times 2^X \times L \) is a finite set of transitions. The transition \( \langle \ell, g, a, R, \ell' \rangle \) from the location \( \ell \) to the location \( \ell' \) is guarded by the constraint \( g \) on clocks and parameters, and specifies an observed symbol \( a \) and the subset of clocks \( R \subseteq X \) to be reset when the transition is taken. Such a transition will also be denoted \( \ell \xrightarrow{g,a,R} \ell' \).

Examples of PTA can be found in Section 3.4. PTA without parameters correspond to standard timed automata (TA) 1. Given a PTA \( \mathcal{A} \) and a valuation \( \pi : \Delta \rightarrow \mathbb{Q}_+ \) defined on the parameters of \( \mathcal{A} \), the TA \( \mathcal{A}[\pi] \) is defined from \( \mathcal{A} \) by replacing every parameter \( \delta \) by its value \( \pi(\delta) \).

3.2. Timed Traces and Languages

A run of a TA \( \mathcal{A} = (L, \ell_0, F, \Sigma, \text{inv}, T) \) is a finite sequence of the form

\[
(\ell_0, v_0) \xrightarrow{d_1, a_1} (\ell_1, v_1) \xrightarrow{d_2, a_2} \ldots \xrightarrow{d_k, a_k} (\ell_k, v_k)
\]

where for every \( 1 \leq i \leq k \), \( \ell_i \) is a location in \( L, v_i : X \rightarrow \mathbb{Q}_+ \) is a clock valuation, \( d_i \in Q_+ \) is a delay and every \( a_i \in \Sigma \). Given \( d \in Q_+ \), let us denote \( v + d \) the clock valuation \( x \mapsto v(x) + d \). The conditions on the above run are that \( v_0 \) satisfies \( \text{inv}(\ell_0) \), and for all \( 1 \leq i \leq k \):

- for all \( 0 \leq d'_i \leq d_i, vi-1 + d'_i \) satisfies \( \text{inv}(\ell_{i-1}) \),
- there exists a transition \( \ell_{i-1} \xrightarrow{g_i,a_i,R_i} \ell_i \) in \( T \) such that:

\[
v_i-1 + d_i \text{ satisfies } g_i, \\
\text{for all } x \in X, v_i(x) = 0 \text{ if } x \in R_i \text{ and } v_i(x) = v_{i-1}(x) + d_i, \text{ otherwise}, \\
v_i \text{ satisfies } \text{inv}(\ell_i).
\]

The run is recognizing if \( \ell_k \in F \), and \( v_0(x) = 0 \) for all \( x \in X \). In this case, we say that the timed word \( \langle a_1, t_1, \ldots, a_k, t_k \rangle \in (\Sigma \times X)_+^k \) is recognized, where \( t_i = \sum_{j=1}^i d_j \) for all \( 1 \leq i \leq k \). Its projection \( a_1, \ldots, a_k \) over \( \Sigma \) is called the associated untimed word. The (un)timed language of \( \mathcal{A} \) is the set of recognized (un)timed words.

3.3. Automata Networks and Synchronization

It is convenient to use a parallel composition operator for PTA when modeling a system made of several subsystems. This approach is particularly relevant for modeling polyphonic mixed scores, where voices (in our case the instrumental part and each of the electronic groups) evolve concurrently and each of them can be represented by a PTA.

A network of parametric timed automata (NPTA) is a set of PTA, built with the operator \( \parallel \) of parallel composition. The automata of the network are supposed to have disjoint sets of clocks and locations, but they may share alphabet’s symbols which are used for synchronization. The synchronized product of a NPTA \( \mathcal{A}_0 \parallel \cdots \parallel \mathcal{A}_m \), with \( \mathcal{A}_i = (L_i, \ell_0, F_i, \Sigma_i, \text{inv}, T_i) \) for each \( 0 \leq i \leq m \), is the PTA

\[
\left( \bigwedge_{i=0}^m L_i, \langle \ell_0, \ldots, \ell_m \rangle, m \parallel \bigcup_{i=0}^m F_i, \bigcup_{i=0}^m \Sigma_i, \text{inv}, T \right)
\]

where \( \text{inv}(\langle \ell_0, \ldots, \ell_m \rangle) = \bigwedge_{i=0}^m \text{inv}(\ell_i) \) and \( T \) is the set of all transitions \( \langle \ell_0, \ldots, \ell_m \rangle \xrightarrow{g,a,R} \langle \ell_0', \ldots, \ell_m' \rangle \) such that for all \( 0 \leq i \leq m \), if \( a \in \Sigma_i \), then there exists \( g_i \) and \( R_i \) such that \( \ell_i \xrightarrow{g_i,a,R_i} \ell_i' \in F_i \), and otherwise \( \ell_i' = \ell_i \), and \( g = \bigwedge_{a \in \Sigma} g_i 

The notions defined above for PTA (language, \( [\pi] \)) are extended to NPTA using this synchronized product.

3.4. Compilation of Scores into Automata

The translation of the simplified mixed scores of Section 3 into NPTA is quite straightforward. Roughly, it returns one PTA representing the instrumental part of the score and one PTA for each group.

The events and actions become the symbols in transitions, and the guards are used to enforce the delays specified in the score. Instead of giving a complete definition of the translation process, we illustrate it in Example 4. A detailed translation procedure is described in [3] for a more complete fragment of the Antescofo language, in order to describe operational semantics for this language.

Example 4. The NPTA \( \mathcal{N} = \mathcal{A}_0 \parallel \mathcal{A}_1 \parallel \mathcal{A}_2 \parallel \mathcal{A}_3 \) associated to the mixed score of Example 1 is presented in Figure 5. It has an alphabet \( \Sigma = \{e1, e2, e3, init, msg, on, off\} \). The guards that involve equalities constraints only. All the bounds of guards in the instrumental automaton \( \mathcal{A}_0 \) are parameters, and they are real values for the other PTAs \( \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3 \) (which are hence all standard TA).

Each of the three automata \( \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3 \) for electronic parts starts with an unguarded transition labeled by the event triggering the group. According to the semantics of the synchronized product of this NPTA presented in Section 3.3, this permits to synchronize the start of each group with a transition of the instrumental automaton. Let us consider for instance the PTA \( \mathcal{A}_1 \) associated to the first group \( g_1 \). It has three locations \( g_0^1, g_1^1, g_2^1 \). Starting in the initial location \( g_0^1 \), it waits for the detection of \( e_1 \). Note that the guard of the transition between \( g_0^1 \) and \( g_1^1 \) is empty (i.e. equal to true): there is no condition on clock values to fire this transition. Once \( e_1 \) is detected, \( \mathcal{A}_1 \) enters \( g_1^1 \) and waits for 0.5 time units (as expressed by the guard \( y_1 = 0.5 \)), sends the message "init", before entering the final state \( g_2^1 \).

The translation of a group \( g_n \) nested in a group \( g \) follows the same principle, using the group label \( \ell_n \) of \( g_n \), instead of an event \( e_n \), as triggering symbol. In the outer group \( g \), a transition labelled with \( \ell_n \) will trigger the nested
group \( g_n \). The nested group \( g_n \) is compiled in a separate PTA \( \mathcal{A}_n \), which starts with a transition labelled with \( \ell_n \) and without guard, exactly like \( \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3 \) in Ex. 4.

For a loop, we use just the same construction as for a group, except for an additional unguarded backward transition from the final location to the first one (actually the target location just after the triggering transition). Moreover, for stopping the loop when the terminating event or action \( \kappa \) is received, we add a transition labelled with \( \kappa \) from every non-final state to the final state.

Example 5. Figure 6 presents the PTA corresponding to the following loop, with triggering event \( e_1 \) and terminating event or action \( \kappa \).

0.0 loop \( \ell \) \{ (0.1 a) (0.2 b) \} until \( \kappa \)

Note the guards \( y_2 < 0.1 \) and \( y_2 < 0.2 \) in the transitions labelled with the terminating \( \kappa \), aiming at the stop state.

The construction for groups with an attribute "tight" is illustrated in the next example.

Example 6. The translation of the score of Example 2 (Figure 4) is presented in Figure 7. The group \( g \) with attribute "tight" is compiled into one PTA with 5 locations \( g^0 \) to \( g^4 \). Note the transition labelled with \( e_2 \) between the transitions labelled with "init" and "on". The behavior of this PTA is be the following: after the detection of event \( e_1 \), it enters location \( g^1 \), waits for 0.5 time units, sends the message init, enters \( g^2 \) and waits for the detection of the event \( e_2 \). After the detection of \( e_2 \), it enters \( g^3 \). However, if \( e_2 \) arrives before the expiration of the delay of 0.5, then init is skip and the PTA jumps to \( g^4 \) (transition with the guard \( y < 0.5 \)). This is the difference between the "tight"

\[
\begin{align*}
&\langle e_0, g^0 \rangle, t=x_1=x_2=0 \\
&\langle e_1, g^1 \rangle, y_1=0.5, \text{msg}. \\
&\langle e_2, g^2 \rangle, y_2=0.5, \text{on}. \\
&\langle e_3, g^3 \rangle, y_3=0.5, \text{off}.
\end{align*}
\]

The NPTA obtained from a mixed score is the basis of verification procedures presented in the next section.

4. VERIFICATION OF MIXED SCORES

Given a NPTA model \( \mathcal{N} \) of a mixed score, we will show how to infer a linear constraint \( \mathcal{K} \) on the parameters of \( \mathcal{N} \).
guaranteeing an acceptable behavior of the automata network (referred to as the good parameters problem in [16]). In other words, \( K \) is such that if a valuation \( \pi : \Delta \rightarrow \mathbb{Q}_\geq \) defined on the parameters of \( \mathcal{N} \) satisfies \( K \), then all the untimed words recognized by \( \mathcal{A}[\pi] \) belongs to an acceptable set \( \text{Acc} \) (where \( \mathcal{A} \) is the synchronized product of \( \mathcal{N} \)). The set \( \text{Acc} \) defines the acceptable discrete behavior of \( \mathcal{N} \), restricting e.g. the ordering on which some actions should be played.

The constraint \( K \) hence represents a range of musical performances \( \pi \) for which the system is guaranteed to behave as expected, from a discrete point of view. We shall present below two notions of acceptable sets of behavior, and define first the crux tool for inferring the expected constraint \( K \): Symbolic semantics of PTA.

### 4.1. Symbolic States and Traces

A constrained PTA is a pair, denoted as \( \mathcal{A}(K) \), made of a PTA \( \mathcal{A} = (L, \ell_0, F, \Sigma, \text{inv}, T) \) and a constraint over parameters \( K \in \mathcal{E}(\Delta) \). The symbolic semantics of a constrained PTA \( \mathcal{A}(K) \) with \( p \) clocks is a transition system over \( \Sigma \) denoted by \( \mathcal{A}^s \). It has a set of states of the form \( (\ell, C) \), with \( \ell \in L \) and \( C \in \mathcal{E}(X, \Delta) \), initial state \( (\ell_0, C_0) \) with \( C_0 = K \land \text{inv}(\ell_0) \land \bigwedge_{i=1}^{p-1} x_{i+1} = x_i \), and transitions \( (\ell, C) \xrightarrow{a} (\ell', C') \) such that there exists a transition \( \ell \xrightarrow{R} \ell' \) of \( \mathcal{A} \) and \( C'(x') = \exists x \exists d \ C(x) \land g(x) \land \bigwedge_{s \in R} x_s' = x_s \land \text{inv}(\ell')(x') \land \text{inv}(\ell')(x' + d) \) where \( x = x_1, \ldots, x_p \), and \( x' = x_1', \ldots, x_p' \).

The formula \( C'(x') \) can be transformed into a constraint of \( \mathcal{E}(X, \Delta) \) using for instance Fourier-Motzkin elimination of existentially quantified variables. It expresses the condition in the definition of runs in Section 3.2 applied to a TA \( \mathcal{A}[\pi] \) such that the parameter valuation \( \pi \) satisfies \( K \). A symbolic trace is a sequence of symbolic states corresponding to a run of \( \mathcal{A}^s \).

### 4.2. Depth First Symbolic State Exploration

In this first approach, the set of acceptable untimed words \( \text{Acc} \) is defined as the language of a given finite state automaton \( \mathcal{B} \) over \( \Sigma \). The automaton \( \mathcal{B} \) can be seen as a special case of TA whose guards and invariants are all set to true. Therefore, it can be composed with \( \mathcal{N} \), to restrict (i.e. control) the language of the network.

The procedure for the inference of a constraint \( K \) as expected consists in a depth-first exploration of the symbolic states reachable by \( \mathcal{A}^s \), where \( \mathcal{A} \) is the synchronized product of \( \mathcal{N} \parallel \mathcal{B} \). The synchronized product is not computed a priori but on the fly, at each step of the exploration. Note that the tree \( T \) of partial symbolic traces of \( \mathcal{A}^s \) is finite in the case of mixed scores without loops, or with loops terminated by an event or action occurring in the score. In this case, the depth of \( T \) is the length of the instrumental score. Its size can be exponential. In order to avoid state space explosion, on can prune a search branch as soon as the constraint of the current symbolic state is unsatisfiable. It is correct since the sequence of symbolic constraints in a symbolic trace is monotonic. The expected constraint \( K \) is then defined as the disjunction of the constraints on the leaves of \( T \) whose state contains a final location.

**Example 8.** Let us come back to the NPTA \( \mathcal{N} \) described in Example 4 and let \( \Sigma_0 = \Sigma \setminus \{ \text{on, off} \} \). The automaton \( \mathcal{B} \) of Figure 8 recognizes the words in which both “off” and “on” occur exactly once, and in this order. The complete exploration of the symbolic traces of \( (\mathcal{N} \parallel \mathcal{B})^s \) is presented in Appendix. Let us see two examples of traces.

The untimed word \( e_1, e_2 \), init, msg, off leads to an unsatisfiable constraint (hence this branch is pruned). Indeed, the constraint reached after \( e_1, e_2 \), init, msg, off is \( \delta_1 \leq 0.5 \), because of the invariant \( y_1 \leq 0.5 \) associated to the location \( g_1 \) in \( \mathcal{B} \). Furthermore, it also contains \( 1.25 - \delta_1 \leq 0.5 \), because of the invariant \( y_2 \leq 0.5 \) for location \( g_2 \) and the guards in the transition of \( \mathcal{B} \) with msg and off. Hence, we have a contradiction.

On the other hand, the branch corresponding to the idealized untimed word \( e_1, \text{init}, \text{msg}, e_2, \text{off}, \text{on} \), leads to the constraint \( 0.75 \leq \delta_1 \leq 1.25 \land 0.5 \leq \delta_2 \). This word can be obtained only for these values of parameters.

Despite the pruning, the state space remain large, and this naive approach (not implemented) should be tractable only for small excerpts of scores. It makes sense however for problems like the inversion of “off” and “on” in Ex. 4.

The verification can be narrowed to a window delimited by the triggering events (\( e_1 \) and \( e_2 \) in the example) and the end of the groups \( (g_3 \text{ and } g_2) \) containing these actions.

### 4.3. Inverse Method

A more efficient approach consist in focusing on the original score for the definition of the acceptable set \( \text{Acc} \) as the untimed language of \( \mathcal{A}[\pi_0] \), where \( \pi_0 \) is the parameter instantiation corresponding to the delays specified in the score. In our case, this language is reduced to the (unique) untimed word corresponding to an idealized performance.

This corresponds to the goal of the inverse method (IM), implemented in the system Imitator: Given a NPTA \( \mathcal{N} \) (with synchronized product \( \mathcal{A} \)), it starts with a good parameter instantiation \( \pi_0 \), and generalizes it, synthesizing a constraint \( K_0 \in \mathcal{E}(\Delta) \) such that \( \pi_0 \) satisfies \( K_0 \) and for all parameter instantiation \( \pi \) satisfying \( K_0 \), the untimed languages of \( \mathcal{A}[\pi] \) and \( \mathcal{A}[\pi_0] \) coincide.

The algorithm IM can be described roughly as follows. Starting with \( K := \text{true} \), it iteratively computes a set \( S \) of reachable states of \( \mathcal{A}(K)^s \). When \( \pi_0 \)-incompatible states \( (\ell, C) \) is found in \( S \) (i.e. such that \( \pi_0 \) does not satisfy \( \exists x C(x) \)) then \( K \) is refined as follows: A \( \pi_0 \)-incompatible inequality on parameters \( J \) is selected in (the normal form

![Figure 8. Control automaton \( \mathcal{B} \) (Example 8)](image-url)
of) $\exists x C(x)$ and $\sim J$ is added to $K$. This way, in the next iteration, $\mathcal{A}(K)^2$ will avoid the counterexample. The algorithm stops when the state set $S$ reaches a fix point, and returns the intersection of the constraints of the states in the final set $S$. Note there are 2 non-deterministic choices in IM: the selection of the $p_{\pi C}$-incompatible state $(\ell, C)$ and of $J$, and in general IM is non-confluent: several applications may lead to different results.

**Example 9.** When run on the NPTA of Example 2, Figure 5. IMITATOR reaches a fixpoint after 6 iterations and returns the constraint $K = 0.75 < \delta_1 \leq 1.0 \land 1.25 < \delta_2$, which defines a smaller domain than the naive approach. Moreover, the application of the algorithm is deterministic in this case. \hfill $\Diamond$

### 4.4. Discussion

Reachability is undecidable for PTA. Hence in general, the algorithms presented in the two previous sections can diverge. The problem however becomes decidable on NPTA associated to scores without loops.

IMITATOR has given good results for the verification of case studies including circuits and communication protocols [4]. The associated models typically contained few (less than a dozen) parameters and locations. As we associate one parameter to every instrumental event in a mixed score, our models can be much larger, in term of number of parameters and locations. In counterpart, they have also a simpler structure. A traditional bottleneck for the verification of automata models equipped with clocks or registers is the occurrences of some control locations in more than one simple cycle [9]. This situation is generally avoided in mixed scores and the experiment conducted so far have given satisfying results. Another originality of our approach is that the PTA models are not constructed manually but obtained by compilation of mixed scores.

In the literature on the verification of embedded or cyber-physical systems, the uncontrollable environment is generally related to physical phenomena: stabilization time in circuits, temperature in a nuclear core, linear acceleration and angular velocity returned by sensors in a plane, GPS... The case of interactive music systems brings a human in the computation loop, possibly opening new application perspectives in this research field, see [13].

### 5. COMPOSITION ASSISTANCE

We briefly describe in this last section another application of the parametric approach presented above. The objective is to provide assistance to a composer by proposing appropriate delays for some actions, according to a given ordering constraint on actions and events. More precisely, let us consider an unfinished mixed score $\mathcal{S}$ where all the delays of the instrument events $e_1, \ldots, e_n$ are set values $d_1, \ldots, d_n$, but some actions’ delays are still parameters $\alpha_1, \ldots, \alpha_m \in \Delta$ (we note $\alpha = \{ \alpha_1, \ldots, \alpha_m \}$). Moreover, we assume that the expected behavior of $\mathcal{S}$ is specified by a linear constraint $K \in \mathcal{G}(\alpha \cup \delta)$, where $\delta = \{ \delta_1, \ldots, \delta_j \} \subset \Delta$ is a set of parameters associated to the events’ delays. We show below that we can propose a valuation $\pi$ of $\alpha$ satisfying $K[\delta_i := d_i, 1 \leq i \leq n]$ and maximizing the robustness of $\mathcal{S}$ to the timing variations in the instrumental performance ($K[\pi]$ denotes the replacement in $K$ of parameters by their value defined by $\pi$).

**Definition of Robustness.** For $1 \leq i \leq n$, let $\sigma_i$ be the valuation defined on $\delta \setminus \{ \delta_i \}$ by $\sigma_i(\delta_i) = d_i$, for all $j \neq i$, and for $\pi$ defined on $\alpha$, let $\inf[\pi]_i$ and $\sup[\pi]_i$ be the lower and upper bounds for $\delta_i$ defined by $K[\pi \cup \sigma_i]$. We let $\text{rob}(K, \pi, e_i) = \min(|d_i - \inf[\pi]_i|, |d_i - \sup[\pi]_i|)$ and $\text{rob}(K, \pi) = \min_{1 \leq i \leq n}(\text{rob}(K, \pi, e_i))$. Intuitively, the smaller this robustness value is, the closer to the delays $d_i$ the musician must play in order to satisfy $K$.

**Total Ordering Constraint.** When $K$ is defined by a total ordering $<$ on the events and actions of $\mathcal{S}$ (i.e. the expected untimed word), then a valuation $\pi$ of $\alpha$ maximizing the above robustness can be defined, after some variable renaming, from the following instantiation of the delays between successive actions $wrt <$ (see [13]):

\[ 1 \leq i \leq n, \text{ and assume } e_i < a_1^i < \ldots < a_n^i < e_{i+1}, \text{ where there are no actions or events between successive elements in this sequence (wrt $<$).} \]

For all $1 \leq j < n_i$, if the actions $a_j^i$ and $a_{j+1}^i$ are triggered by the same event, then the delay between them is set to the smallest delay supported by the system (let us call $\epsilon$ this value). Otherwise, the delay between $a_j^i$ and $a_{j+1}^i$ is set to $\frac{p_{j+1} - p_j}{n_{j+1} - n_j}$, where $p_i$ is the number of pairs of successive actions in $a = \{ a_1^1 < \ldots < a_n^1 \}$ triggered by the same event, and $c_i$ is the number of sets of successive actions in $a$ triggered by the same event. Moreover, the delay between $e_i$ and $a_1^i$ and between $a_n^i$ and $e_{j+1}$ is also set to the same value. Intuitively, successive actions triggered by the same event must be as close as possible and other successive actions must be as far as possible.

**Regular Constraint.** The constraint $K$ can also defined from a set of acceptable untimed words, e.g. presented by a FSA $B$ as in Section 4.2. In this case, we can pick an untimed word $w$ recognized by $B$ and propose one optimal $\pi$ for the total ordering defined by $w$, using the above technique. Finding one word $w$ recognized by $B$ such that the associated $\pi$ maximizes robustness is an open question.

### 6. CONCLUSION

In general, the purpose of verification procedures such as model checking [5] is to ensure (e.g. with a formal proof destined to a certification authority), that a critical embedded system satisfies some correctness property in every possible execution. The approaches presented in Section 4 for the timing analysis of mixed scores aim rather at characterizing precisely the range of instrumental executions (performances) for which the reactive engine of Antescofo will behave as expected. This seems more appropriate in the context of interactive music systems. To our knowledge, it is the first time that such methods are applied to
of interactive music systems. The outcome of the verification procedures can be exploited in several ways, for instance for warning performers about sensible parts of the score, or to assist the composer in adjusting the delays of electronic actions in these parts. The CAC approach proposed in Section 3 follows the same idea. A user-friendly presentation of the outcome (currently linear constraints), directly in the score, is under study.

As possible extensions, one could use (linear) expressions over parameters for expressing delays in transitions’ guards of PTA (instead of a single parameter), e.g. for specifying mixed scores with variables [24, 20]. The generalization of IM to this case could be interesting. Another application target could be non-linear scores (aka open scores), in which the instrumental part contains branches, the direction in these branches being chosen by the performer at real-time. Our framework could be extended to this case with a straightforward translation of the branches as non-deterministic transitions in the instrumental PTA (automaton $\phi_6$) in Example 4.

Other applications of the TA model of Section 3 could also be worth investigating. For instance reachability analysis or timed model checking [17] over the unparametric TA model could be used for verifying temporal properties of the idealized performance. We are also planning to use approaches based on timed games theory [8] for giving a quantitative estimate of the robustness (upper bound on the time deviation guaranteeing expected behaviour) of a mixed score to timing variations in performances.

Acknowledgements.

The authors wish to thank Romain Soulat and Laurent Fribourg for valuable discussions and their precious assistance on using the tool MITATOR.

7. REFERENCES


Appendix

We present in Figure 9 a depth-first-search of the trace tree for the NPTA of Example 4, Figure 5 (score of Example 1, Fig. 1).

In this exploration, we give for each (partial) trace the constraint of the symbolic state reached, decomposed in four columns corresponding to the four clocks of the automaton: $x$, $y_1$, $y_2$ and $y_3$. Note that the constraints reached at the different prefix of a trace are accumulated (one should consider a conjunction of all the constraints reached at prefixes). The symbol $\land$ indicates that the constraint reached is unsatisfiable and that a backtrack is necessary. The column “delay” presents the delay elapsed for performing the last step of the trace.

(1–4) these partial traces lead to unsatisfiable constraints containing both $\delta_1 \leq 0.5$ and $\delta_1 \geq 0.75$.

(5–6) these traces are possible only if $\delta_1 = 0.75$, $\delta_2 = 0$ (i.e. $e_2$ and $e_3$ are simultaneous) and off and on are simultaneous. Hence it is neither a realistic nor a desirable solution.

(7) this trace is possible under the constraint $0.75 \leq \delta_1 \leq 0.75$, $\delta_2 \leq 0.5$ and $\delta_1 + \delta_2 \geq 1.25$, i.e. only for $\delta_1 = 0.75$ and $\delta_2 = 0.5$.

(8) this trace is possible under the constraint $0.75 \leq \delta_1 \leq 0.75$, $0.25 \leq \delta_2 \leq 0.5$ and $\delta_1 + \delta_2 \geq 1.25$, i.e. only for $\delta_1 = 0.75$ and $\delta_2 = 0.5$.

(9) this trace is possible under the constraints $0.75 \leq \delta_1 \leq 1.25$, $\delta_2 \leq 0.5$, $\delta_1 + \delta_2 \leq 1.25$, and $0.25 \leq \delta_1 - 2\delta_2$.

(10) this traces is possible under the constraints $0.75 \leq \delta_1 \leq 1.25$, and $\delta_2 \leq 0.5$, $1.25 \leq \delta_1 + \delta_2$.

(11) this corresponds to the ideal trace, defined by $\delta_1 = \delta_2 = 1.0$. This trace is possible under the constraints $0.75 \leq \delta_1 \leq 1.25$, $0.5 \leq \delta_2$ and $1.25 \leq \delta_1 + \delta_2$. 
<table>
<thead>
<tr>
<th>trace</th>
<th>delay</th>
<th>(x =)</th>
<th>(y_1 =)</th>
<th>(y_2 =)</th>
<th>(y_3 =)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e_1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(e_1 e_2)</td>
<td>(\delta_1)</td>
<td>0</td>
<td>(\delta_1 \leq 0.5)</td>
<td>0</td>
<td>(\delta_1 \leq 0.75)</td>
</tr>
<tr>
<td>(e_1 e_2 e_3)</td>
<td>(\delta_2)</td>
<td>(\delta_1 + \delta_2 \leq 0.5)</td>
<td>(\delta_2 \leq 0.5)</td>
<td>(\delta_1 + \delta_2 \leq 0.75)</td>
<td></td>
</tr>
<tr>
<td>(e_1 e_2 e_3) init</td>
<td>(0.5 - \delta_1 - \delta_2)</td>
<td>0.5</td>
<td>(0.5 - \delta_1 \leq 0.5)</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>(e_1 e_2 e_3) init msg</td>
<td>0.25</td>
<td>0.75 - (\delta_1 \leq 0.5)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) (e_1 e_2 e_3) init msg off (\odot)</td>
<td>0.5</td>
<td>1.25 - (\delta_1 \leq 0.5)</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e_1 e_2) init</td>
<td>(0.5 - \delta_1)</td>
<td>(0.5 - \delta_1 \leq \delta_2)</td>
<td>0.5</td>
<td>(0.5 - \delta_1 \leq 0.5)</td>
<td></td>
</tr>
<tr>
<td>(e_1 e_2) init (e_3)</td>
<td>(\delta_2 - 0.5 + \delta_1)</td>
<td>(\delta_2 \leq 0.5)</td>
<td>(\delta_1 + \delta_2 \leq 0.75)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e_1 e_2) init (e_3) msg</td>
<td>(0.75 - \delta_1 - \delta_2)</td>
<td>0</td>
<td>(0.75 - \delta_1 \leq 0.5)</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>(2) (e_1 e_2) init (e_3) msg off (\odot)</td>
<td>0.5</td>
<td>1.25 - (\delta_1 \leq 0.5)</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e_1 e_2) init (e_3) msg</td>
<td>0.25</td>
<td>0.75 - (\delta_1 \leq \delta_2)</td>
<td>0</td>
<td>(\delta_1 \leq 0.75)</td>
<td></td>
</tr>
<tr>
<td>(e_1 e_2) init (e_3) msg off (\odot)</td>
<td>(\delta_2 - 0.75 + \delta_1)</td>
<td>0</td>
<td>(\delta_2 \leq 0.5)</td>
<td>(\delta_1 \leq 0.75)</td>
<td></td>
</tr>
<tr>
<td>(3) (e_1 e_2) init (e_3) msg (\odot)</td>
<td>(1.25 - \delta_1 - \delta_2)</td>
<td>(1.25 - \delta_1 \leq 0.5)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4) (e_1 e_2) init (e_3) msg off (\odot)</td>
<td>(0.5)</td>
<td>(1.25 - \delta_1 \leq \delta_2)</td>
<td>(1.25 - \delta_1 \leq 0.5)</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>(5) (e_1) init</td>
<td>(0.5)</td>
<td>(0.5 \leq \delta_1)</td>
<td>0</td>
<td>(\delta_1 \leq 0.75)</td>
<td></td>
</tr>
<tr>
<td>(e_1) init (e_2)</td>
<td>(\delta_1 - 0.5)</td>
<td>0</td>
<td>0</td>
<td>(\delta_1 \leq 0.75)</td>
<td></td>
</tr>
<tr>
<td>(e_1) init (e_2) (e_3)</td>
<td>(\delta_2)</td>
<td>0</td>
<td>(\delta_1 \leq 0.5)</td>
<td>(\delta_1 + \delta_2 \leq 0.75)</td>
<td></td>
</tr>
<tr>
<td>(e_1) init (e_2) (e_3) msg</td>
<td>(0.75 - \delta_1 - \delta_2)</td>
<td>(0.75 - \delta_1 \leq 0.5)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e_1) init (e_2) (e_3) msg off (\odot)</td>
<td>(0.5)</td>
<td>(1.25 - \delta_1 \leq 0.5)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(6) (e_1) init (e_2) (e_3) msg off (\odot)</td>
<td>(\delta_1 - 0.75)</td>
<td>(\delta_1 \leq 0.75)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(7) (e_1) init (e_2) (e_3) msg (\odot)</td>
<td>(\delta_2 - 0.75 + \delta_1)</td>
<td>(\delta_2 \leq 0.5)</td>
<td>(\delta_1 \leq 0.75)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(8) (e_1) init (e_2) (e_3) msg off (\odot)</td>
<td>(\delta_2 - 1.25 + \delta_1)</td>
<td>(\delta_2 \leq 0.5)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(9) (e_1) init (e_2) (e_3) msg off</td>
<td>(\delta_1 - 0.75)</td>
<td>(\delta_1 \leq 0.75 \leq 0.5)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e_1) init (e_2) (e_3) msg (\odot)</td>
<td>(\delta_2)</td>
<td>(\delta_2 \leq 0.5)</td>
<td>(\delta_1 - 0.75 \leq 0.5)</td>
<td>(\delta_1 - 0.75 \leq 0.5)</td>
<td></td>
</tr>
<tr>
<td>(10) (e_1) init (e_2) (e_3) msg (\odot)</td>
<td>(\delta_2 - \delta_1 + 0.75)</td>
<td>(0.25 \leq \delta_2)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e_1) init (e_2) (e_3) msg off</td>
<td>(\delta_2 - \delta_1 + 0.75)</td>
<td>(\delta_2 \leq 0.5)</td>
<td>(\delta_1 - 0.75 \leq 0.5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(11) (e_1) init (e_2) (e_3) msg off</td>
<td>(\delta_1 - 0.75)</td>
<td>(\delta_1 \leq 0.75 \leq 0.5)</td>
<td>(\delta_1 - 0.75 \leq 0.5)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 9.** Symbolic traces (Example 5). Delay is from previous parent in the trace tree.
**Model: ex-onoff.imi**

**Mode: inverse method.**

Parsing completed after 0.00300000000001 second.
Memory for abstract program: 256.36328125 KB (i.e., 65629. words)

Computing post\(^1\) from 1 state.
Computing post\(^2\) from 1 state.
    Adding the following inequality:
        1 < 2*d1
Computing post\(^3\) from 2 states.
    Adding the following inequality:
        3 < 4*d1
Computing post\(^4\) from 2 states.
Computing post\(^5\) from 2 states.
    Adding the following inequality:
        5 < 4*d1 + 4*d2

Fixpoint reached after 6 iterations: 8 reachable states with 7 transitions.
Analysis has been fully deterministic.

Final constraint K0 (3 inequalities):
1 >= d1
& 4*d1 > 3
& 4*d1 + 4*d2 > 5

Inverse method successfully finished after 0.0220000000001 second.

IMITATOR successfully terminated (after 0.0220000000001 second)

**Figure 10.** Execution of IMITATOR on Example 9.