HAL
open science

# ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music.

Arshia Cont

## To cite this version:

Arshia Cont. ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music.. International Computer Music Conference (ICMC), Aug 2008, Belfast, Ireland. pp.33-40. hal-00694803

**HAL Id: hal-00694803**

**https://inria.hal.science/hal-00694803v1**

Submitted on 6 May 2012

**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.

# *ANTESCOFO*: ANTICIPATORY SYNCHRONIZATION AND CONTROL OF INTERACTIVE PARAMETERS IN COMPUTER MUSIC

*Arshia Cont*
UCSD, Music Department, and
Ircam - UMR CNRS STMS.
cont@ircam.fr

## ABSTRACT

*Antescofo* is a modular anticipatory score following system that holds both instrumental and electronic scores together and is capable of executing electronic scores in synchronization with a live performance and using various controls over time. In its very basic use, it is a classical score following system, but in advanced use it enables concurrent representation and recognition of different audio descriptors (rather than pitch), control over various time scales used in music writing, and enables temporal interaction between the performance and the electronic score. *Antescofo* comes with a simple score language for flexible writing of time and interaction in computer music.

## 1. INTRODUCTION

The moment a composed computer music piece is concerned with mixed live instruments and electronics, the question of how to handle synchronization and interaction between the two becomes inevitable. During the early periods of mixed pieces, most composers (with exceptions) dealt with the problem as a secondary concern. In the early 80s and through the advent of interactive computer music environments and their popularity among musicians, the idea of dealing with interaction both during performance and composition became more apparent and softwares and technologies such as score followers began to rise the possibilities such control could bring into the world of computer music composition and performance. Today, celebrating 25 years of score following research and performance, the divide between 'compositional' and 'performative' aspects of computer music [1] and the lack of consideration for interaction in the writing of the piece are even more apparent than it was at the time. This is also accompanied by many composers' reluctance to consider such techniques in the compositional process. We believe this situation is partially due to the following constraints in available interactive computer music systems: (1) While the common *vocabulary* used in scoring instrumental music has extensively expanded, synchronization or score following applications are extremely limited to very simple vocabulary in western traditional notation (notes, trills etc.). (2) The notion of *interaction* is most often limited to mere triggering of a separate electronic score. (3) There has been limited or no consideration for different temporalities of musical events, the temporal interaction within and the writing of time involved [2].

This paper is concerned with interactive components of scored mixed instrumental and electronic music repertoire where the need for an explicit interaction between the live instrument and the electronics is evident. We present *Antescofo*, a tool that handles both scoring and live interaction of such computer music pieces. In its very basic use, *Antescofo* is a classical score following application that synchronizes real-time audio to a music score (such as MIDI). But it has been designed in its core to address the following extensions handling both flexible score scripting and live interactions: (1) to enable concurrent, flexible and user-defined representations of the audio stream in the score, (2) to concurrently represent and handle different time scales both in scoring and recognition and enable a flexible *writing of time*, (3) to provide a score language that handles interaction between the live performer(s) and electronics both time-wise and process-wise during runtime (i.e. music performance).

In this paper we present key ideas that form the foundation of *Antescofo* and discuss its use during production and performance. We begin the paper by providing some musical and technical background on the subject. This is followed by a general description of the underlying design architecture that defines the main concerns and components of the system. We outline our model in sections 7 to 10 with a contemplative emphasis on modeling time for music in section 5. The score language and syntax developed for *Antescofo* is defined in section 8. Finally, we direct the readers attention to development and production issues regarding the system as well as discussions and future directions. In this paper, we focus our attention on the musical aspects of the design in *Antescofo* and leave scientific derivations and proofs of methods introduced briefly for the sake of completeness for a later communication.

## 2. MUSICAL BACKGROUND

The consensus for interaction between a live music performance and electronics dates back to early experiments of Maderna, Stockhausen and Davidovsky among others through fixed or tape pieces where the live performer is responsible for synchronizing with the electronics using

click-tracks or active listening. Later on in mid 80s, the movement is joined by experiments and repertoire of the so called "real-time electronics", starting from experiments by Manoury and Boulez, where most often a score following application is responsible for synchronizing events of the live instruments to the pre-written score and triggering the appropriate electronic events (whether fixed or through live generation). In this latter group, until today, a lot of composers have leant on the idea of a score following application to automate the performance of the electronics score with synchronization to the live performer(s), while some others, with Manoury as a forerunner, immediately recognized and incorporated the possibilities in writing *interactive* electronic scores which are realized during the live performance.

Naturally, the advent of score following techniques for synchronization of live performance with electronic score and control of interaction created a lot of momentum both in the research and music communities but not without criticism. Among many criticisms directed towards *real-time electronics* school, of particular interest are the ones by compsoers Jean-Claude Risset and Marco Stroppa. Risset argues that "Not only does the use of real-time systems bring limitations and difficulties for the durability of the music, but one may even argue that the concept of real-time concerns primarily performance and may be of little relevance to musical composition" [3]. A constructional read of Risset's paper would point to an important lack in the existing systems: the lack of a compositional interaction during performance. While this issue is in most parts aesthetical, it has also a great deal to do with a lack of explicit designs for such matter. Stroppa's extensive criticism of real-time electronics is accompanied by the composer's detailed proposal for a *Virtual Interpreter* system[2]. In this vast proposal (which we will not detail in this paper), the composer is mostly concerned with *temporality* of musical events and different temporal degrees of interaction in computer music spanning from explicit interaction of fixed or live electronic processes with real-time detected tempi to continuous representations of time to allow fine grain tuning of composed electronics to a live performance.

## 3. RESEARCH BACKGROUND

Since the inception of the first MIDI score followers in 1984 by Dannenberg [4] and Vercoe [5], the research community has made a lot of contributions and advancements to the field by incorporating more advanced recognition and alignment techniques such as probabilistic models [6], graphical modeling [7]and much more that we will not review for the scope of this paper and would refer the curious reader to [8] for a historical review. For this paper, we are particularly interested in approaches where explicit models of *time* are involved during recognition; mentioning that none of the available systems, to the extent of the author's knowledge, has explicitly integrated interaction with the electronic score within the recognition system and handling interaction is left to the user once events are recognized on-the-fly.

It is interesting to note that Vercoe's initial MIDI *Synthetic Performer* had explicit interaction with the deducted tempo of the live performer[5] [1] . With the move to audio systems using pitch detection algorithms *tempo* was temporarily forgotten focusing on string matching techniques and back again with Grubb and Dannenberg [6] where the pitch observations used in the probabilistic model can influence the running tempo by comparing the elapsed time and the idealized score tempo. Perhaps the most elaborate and explicit time model that has been used belongs to Raphael [7]. Raphael's design has two stages for decoding of score position and tempo. The first, comprising Hidden Markov Models deducted from the score responsible for decoding the position in the score and the second, an elaborate Bayesian network which takes this information to deduct the smooth tempo during the performance. Notably, Raphael uses this tempo in interaction with his accompaniment system to adapt the time-duration of the accompanying section using phase-vocoding techniques which has proven to be very effective.

*Antescofo* is an *anticipatory* score follower with an anticipatory design that will be outlied hereafter. It is capable of outputting the real-time tempo of the performance but unlike previous approaches, it comprises two *coupled* audio and tempo agents which work collaboratively and competitively in an anticipatory design to reduce computation and design complexity and increase robustness when there is uncertainty in one of the two agents. The core design of the probabilistic approach in *Antescofo*, enables direct access to temporal structures which can be easily accessed and tweaked in the provided score language.

## 4. GENERAL ARCHITECTURE

The proposed system in this paper is based on pair coupled audio and tempo agents. The music score is represented through a probabilistic graphical model constructed directly from a symbolic music score. The two audio and tempo agents collaborate at all times to map the real-time audio input to the most likely state sequence in the score model. The tempo agent computes on the *event* time-scale and is based on a cognitive model of musical metric structure introduced in [9] and provides continuous tempo predictions based on live audio input and the given music score. On the other hand, the audio agent undergoes computation on the *continuous* audio time-scale and assigns out-of-time probabilistic values to relevant states in the score state-space. The proposed model is an *anticipatory system* which implies "a system containing a predictive model of itself and/or of its environment, which allows it to change state at an instant in accord with the models predictions pertaining to a later instant"[10]. The audio agent is influenced dynamically by the predicted tempo, and in

[1] See the historical video at http://www.youtube.com/watch?v=vOYky8MmrEU

return the tempo agent is directly affected by the decisions obtained instantly by the system.

This model has the underlying hypothesis that the audio signal can be totally generated by the underlying state-space score model. Hence the problem of score following is the inverse of this hypothesis, or to find the most likely state-sequence associated with observed real-time audio sequence. Due to the nature of this inverse problem, the underlying state-sequence that generates the audio is not directly observable by the system. This process of finding the most likely state-sequence in a hidden process up to the present is referred to as the *inference* problem. The state-space generative model of the score proposed here is a *Hidden Hybrid Markov/semi-Markov chain* [11]. Figure 1 shows a general diagram of *Antescofo*'s design. Besides decoding the correct score position and tempo, the system is capable of undertaking (electronic) score actions if they are present within the pre-loaded score as a way to communicate with external generative or synthesis engines coupled with the musical time scales that are provided through *Antescofo*'s score language.

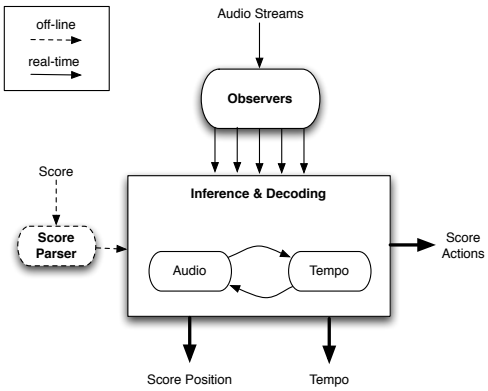



**Figure 1**. General Design Diagram of *Antescofo*

In what follows, we will look at more detailed design of each component of the system described above.

## 5. ON MODELING *MUSICAL TIME*

### 5.1. Musical Foundations

In most computer music systems that deal with models of time, modeling concepts are inherited from long studied models available for speech or biological sequences. In almost every field of research dealing with time sequences, the issue of belief propagation through time is a huge dilemma and subject to intense ongoing research. Choosing one approach over another brings in drawbacks and approximations according to the departing hypothesis lied in the model itself. When it comes down to music signals, the issue of time modeling is of more burden. On the other hand, musicians and artists, unlike researchers, have more freedom in contemplating on the issue of time modeling. The author believes that there is still a lot to learn from artists with regards to modeling time in every field of research dealing with the issue. In this section, we gather several important contemplations on time modeling as inputs from the artistic community that constitutes the core of *Antescofo*'s temporal modeling. There is no single reference regarding time models in the arts! Every composer or artist has dealt with the problem one way or another. We will not go through all aspects of time modeling in this paper and leave a broader analysis for future. Moreover, we do not claim to provide a universal model of time. Here, we present several views and categories that are widely accepted and are highly inspirational for the design of *Antescofo* and were partially presented in [12] .

#### *5.1.1. Temporal vs. Atemporal*

Formalized first by Xenakis[13], the *Atemporal* (or *out-of-time*) corresponds to an object that possess its own internal temporal structure independent of the overall temporal structures of the piece of music. He also emphasizes on the independence of the two time structures. To conform this distinction with our probabilistic design, we define an *Atemporal* object or event as one that posseses a physical space in the score but does not contribute to the physical time of the score. Typical examples of atemporal objects are grace notes and internal notes of a (free) trill in the western classical notation. While both have physical presence, the individual events do not contribute to the notion of tempo but their relative temporal appearance in the case of the first, or their overall *in-time* structure in the case of the second contribute to the notion of tempo. Other examples of atemporal objects are events with fermatas or free-improvisation boxes seen in various contemporary music repertoire.

#### *5.1.2. Striated-time vs. Smooth-time*

Striated time is one that is based on recurring temporal regularities while smooth time is a continuous notion of time as a flow of information [14]. The pulsed-time used in most western classical music notation is a regulated striated time-flow that uses an internal musical clock usually driven by a tempo parameter in beats-per-minute. In our terminology, we distinguish between a striated time-scale where the notion of time is driven relative to a constantly evolving tempo, and smooth time-scale where the information on the microscopic level consists of individual atemporal elements or is defined relative to a clock-time. Typical example of a smooth-time event in western traditional notation are free *glissandis*. This distinction is crucial in order to enable a flexible writing of time and also to enable the coexistence of traditional event notations with that of continuous audio-driven events.

## 6. PROBABILISTIC MODELS OF TIME

One of the main goals of probabilistic modeling is to decode temporal dynamics of an outside process. The main difficulty regarding this task is the temporal dynamics of the underlying model in use. In most problems, any state

of a given process occupies some duration that can be deterministic or not. In such tasks, we are interested in a probabilistic model of the macro-state duration and sojourn time. In a musical context, a macro-state can refer to a musical event (note, chord, silence, trills etc.) given an expected duration. A common way to model time series data in the literature is by the use of *state-space models*. A state-space model of a sequence is a time-indexed sequence of graphs (nodes and edges) where each node refers to a state of the system over time. Therefore each state has a time-occupancy that can be used to model sojourn time and duration of the events under consideration. In this paper, we limit ourself to two wide classes of graphical models and their duration models that cover most existing approaches: Hidden Markov models (HMM) and Hidden Semi-Markov chains. The major drawback in HMM is their inflexibility in describing the time spent in a given state due to their *implicit* nature of time occupancy. A semi-Markov chain is composed of an embedded Markov chain representing the transitions between distinct states, and discrete *explicit* state-occupancy distributions representing sojourn times. Our proposal is to use the benefits of the both worlds by introducing hybrid Markov/semi-Markov chains as first defined in [11]. In section 8 we will show how such a mixture can represent different time dependencies visited in the last section.

To formalize the problem, we assume that the audio stream through time $\tau$ or $x_0^\tau$ (as short for $x_0, \ldots, x_\tau$) is a stochastic process represented by the random variable $\{X_t\}$, which is generated by a sequence of states $s_0^\tau$ through the random variable $\{S_t\}$ corresponding to (hidden) states in a hybrid markov/semi-Markov chain generated from the score. The solution to the inference problem then determines the most-likely state-sequence $S_0^\tau$ that would generate $X_0^\tau$ and in return the score position and real-time decoded tempi. Beyond this point, we use $P(S_t = j)$ as a short for $P(S_t = s_j)$ denoting the probability that state $j$ is emitted at time $t$.

Let $S_t$ be a $J$-state hybrid Markov/semi-Markov chain. It can be then defined by:

- *Initial probabilities:*
  $\pi_j = P(S_0 = j)$ with $\sum_j \pi_j = 1$, which correspond to the starting point in the score of the synchronization application during performance.

- *Transition Probabilities:*

  - semi-Markovian state $j$:
    for each $k \neq j$, $p_{jk} = P(S_{t+1} = k | S_{t+1} \neq j, S_t = j)$ where $\sum_{k \neq j} p_{jk} = 1$ and $p_{jj} = 0$.
  - Markovian state $j$:
    $\tilde{p_{jk}} = P(S_{t+1} = k | S_t = j)$ with $\sum_k \tilde{p_{jk}} = 1$.

- An *explicit* occupancy (or sojourn time) distribution attached to each semi-Markovian state:

$$d_j(u) = P(S_{t+u+1} \neq j, S_{t+u-v} = j, \quad v = 0, \ldots, u-2 | S_{t+1} = j, S_t \neq j) \tag{1}$$

  where $u = 1, \ldots, M_j$ and $M_j$ denotes the upper bound to the time spent in state $j$. Hence, we assume that the state occupancy distributions are concentrated on finite sets of time points.

- An *implicit* sojourn distribution attached to each Markovian state $j$ where
  $P(S_{t+1} = k | S_{t+1} \neq j, S_t = j) = \tilde{p_{jk}}/(1 - \tilde{p_{jk}})$
  defines an implicit state occupancy distribution as the geometric distribution with parameter $1 - \tilde{p_{jk}}$:
  $d_j(u) = (1 - \tilde{p_{jk}})\tilde{p_{jk}}^{u-1}$

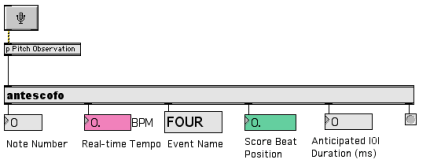
## 7. THE OBSERVER

The audio process $\{X_t\}$ is related to the hybrid Markov / semi-Markov chain $\{S_t\}$ (or the score) by the *observation probabilities*:

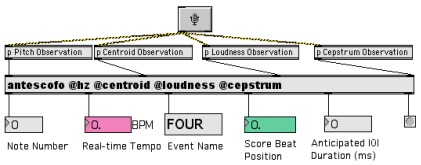
$$b_j(y) = P(X_t = y | S_t = j) \quad \text{where} \quad \sum_y b_j(y) = 1.$$

Observations in our context correspond to the instantaneous belief about the expected score events given the latest (or real-time) inputs to the system. Most score following applications comes either with their internal observation mechanisms or are tuned towards a specific observation module living outside the score follower itself. For the design of *Antescofo*, we have decided to make it modular by both providing an internal observation mechanism and also enabling user-defined observation inputs. In its basic use, *Antescofo* is a classical score following application that accepts a list of pitches (in Hz or MIDI) as input to map it to the score position and the tempo variable. But for more curious users, *Antescofo* is able to accept concurrent observations of different nature. The number of concurrent observations to the system (which are user-defined and calculated outside *Antescofo*) and their code names are defined by the user during object instantiation in Max or Pd. Figure 2 shows the classical and user-defined appearances of the *Antescofo* on a Max window. Here, the creative user of figure 2(b) has attempted to provide four different concurrent observation to the module. Consequently, the score that the user creates would normally make use of these different sources to follow different aspects of audio streams during a performance. We will get back to this point later in section 8.2 and here focus on how these observations are prepared by the observer for the inference and decoding modules.

*Antescofo* comes with several standard built-in observations for classical score following which are denoted by: `hz`, `midi` and `KL` used respectively for pitch observation using an external pitch tracker, MIDI observation and polyphonic audio observations. To obtain instantaneous observation probabilities for *Antescofo*'s modular observer, we simply center Normal distributions over the expected value indicated in the score and obtain probabilities in run-time. Obviously if the input consists of a vector of values multivariate normal distributions are used instead. The variance of these normal distributions (or the




(a) Classical


<antescofo @hz @centroid @loudness @cepstrum


(b) User-defined

**Figure 2**. Modular Observation in *Antescofo*

diagonal covariance matrix for vector inputs) are set to $1\%$ of the expected value (or a semi-tone for pitch), also controllable by the user through the score (section 8.2). The above holds for all observation modes except the `KL` mode which is designed for polyphonic tracking and is left out here due to space considerations.

## 8. SCORE TOPOLOGY AND LANGUAGE

The state-space topology of $\{S_t\}$ is determined by the score, where each score element is mapped into a hybrid Markov/semi-Markov equivalent chain using *Antescofo*'s parser. These built-in maps describe different event types and time models and can be directly described using a simple text-based score language. The score language of *Antescofo* has been carefully designed to enable importing of common score formats such as MIDI and to be able to easily describe common classical music repertoire as well as user-defined events coming from different audio observations and with different temporalities. In this section we describe the basics of *Antescofo*'s score language with their real-world equivalences and show how the same syntaxes can define complex unconventional score events. Moreover, *Antescofo*'s score language enables the coexistence of the instrumental score and the electronic music score altogether. As a convention, in the figures that follow, a Markov state is demonstrated by a regular circle and a Semi-Markov state by a double-lined circle. Also in defining command syntaxes, the plus sign (+) next to each type should be interpreted as "one or more of". As a last convention, a `<float>` indicates a floating number representing the notated observations in the score. For pitched events and as a convention, events would be represented by either MIDI or MIDIcent note numbers.

### 8.1. Basic commands

#### *8.1.1. `BPM` command*

The initial tempo and any tempo change in the score can be encoded by the `BPM` command in Beats-Per-Minute.

#### *8.1.2. Single Event*

A single event can be a single pitch, silence or grace note if the observation under consideration is pitch. These events can be either temporal or atemporal (see section 5.1.1). The usual syntax for a single event is as follows:

```
<float> <duration> <optional name>
```

where the duration is expressed as the number of beats relative to the initial score tempo. Figure 3 shows a sample graphical score, the *Antescofo* equivalent and the state-transition diagram created after parsing. If the duration associated with a single event is set to $0.0$, it is a sign that the associated event is *atemporal*. In this example, pitches are encoded using MIDIcent format and a left-right state diagram is created that is in one-to-one correspondence with the score. Note that in this example, a *dummy* atemporal silence is created in the middle state. *Antescofo*'s parser automatically puts dummy silences between events where appropriate to better model the incoming audio.

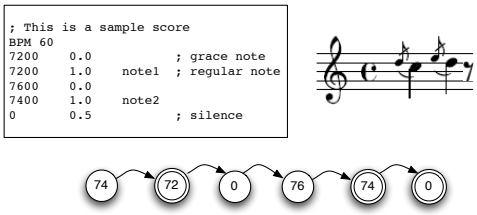

; This is a sample score
BPM 60
7200 0.0 ; grace note
7200 1.0 note1 ; regular note
7600 0.0
7400 1.0 note2
0 0.5 ; silence


**Figure 3**. *Antescofo*'s score sample and state transition diagram for single events

#### *8.1.3. `TRILL` Class*

As the name suggests, the `TRILL` class of *Antescofo* is a way to imitate classical music trill notation. In terms of modeling, *Antescofo*'s `TRILL` is *one in-time* event that has several *out-of-time* events within. Moreover, the order in which these sub-states appear is not important. Figure 4 shows two examples for Antesofo's `TRILL` syntax where the second is taken out of the first measure in Marco Stroppa's *Little-I* for flute and electronics and demonstrates a free glissandi which can be successfully encoded using the `TRILL` class in *Antescofo*. The `TRILL` class syntax is as follows:

```
TRILL ( +<float> ) <duration> <name>
```

#### *8.1.4. `CHORD` Class*

As the name suggests, a chord class denotes a single semi-markov (or markov if duration is set to zero) state that models polyphonic events. The regular syntax for the `CHORD` class is similar to the `TRILL` class but translates to only one state:

```
CHORD ( +<float> ) <duration> <name>
```

#### *8.1.5. `MULTI` Class*

Using the above commands, any classical music piece can be easily parsed or rewritten in *Antescofo*'s format. We

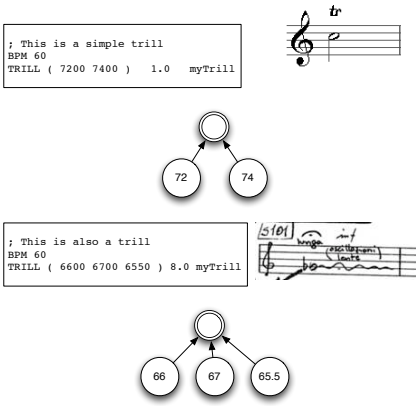



**Figure 4**. *Antescofo*'s score sample and state transition diagram for `TRILL` class

add one more class to allow more complex object and temporal encoding. The `MULTI` class is similar to the `TRILL` class with the exception that the symbols defined within it are *ordered in time*. This new addition to *Antescofo*'s score syntax allows decoding of continuous time events such as glissandis (in western notation) and even audio streams associated with a pre-defined observation for audio-matching. The `MULTI` syntax is as follows:

```
MULTI ( +<float> ) <duration> <name>
```

In this new topology, a high-level semi-markov state represents the overall temporal structure of the whole object that is mapped to a series of sequential left-right Markov chains. Figure 5 shows a `MULTI` example for two consecutive notated glissandis.

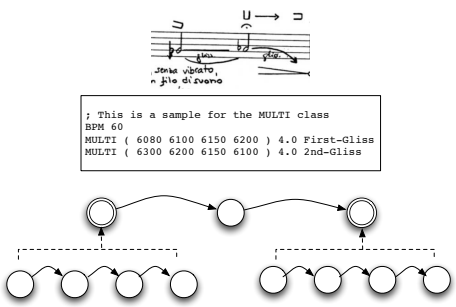



**Figure 5**. *Antescofo*'s score sample and state transition diagram for `MULTI` class

### 8.2. Advanced commands

#### *8.2.1. VARIANCE*

The variance associated with the observer (see section 7) can also be controlled in the score and takes as unit, semitones (for pitched events) or percentage values of the expected score event (for other observations). This is quite useful in several circumstances. For example, when following audio signals from a flute in a live performance, the tuning of high pitches might become different than the expected tuning considered in the score due to warming up of the flute's body. An increase of the observer's variance in such case could save the system and the performance! Also, when dealing with different information sources of audio, one might want to adapt this percentage to the nature of the incoming process.

#### *8.2.2. The @ Operator*

As mentioned in section 7, *Antescofo* is capable of handling concurrent representations of audio streams for recognition and interaction. By default, *Antescofo* uses the leftmost inlet for all recognition work unless specified beforehand by the `@` operator. The string following the `@` operator should conform to the code names created by the user during the object's instantiation (section 7) otherwise it would be neglected during score parsing and an error message would be sent. Using this, the user can easily switch between various representations of audio and follow the desired aspects simultaneously in a single score.

### 8.3. Computer Music Score Actions

One of the important features of *Antescofo* scores is the coexistence of the instrumental part with the electronics part at the same place. Traditionally, in most interactive environments that make use of score following systems, the electronic score lives separately in additional *qlists* and the event numbers (or names in our case) associated with specific score events would trigger the messages in the *qlist* for further processing. This can be easily done inside *Antescofo*'s scores. In addition to that, *Antescofo* has an internal timer that is coupled with the real-time decoded tempo that can be used to do sequencing over the messages to be sent. Note that while traditional *qlist*s also allow sequencing through delay values in milli-seconds, the simple fact that the *sequencer* in *Antescofo* can accept *relative* musical values (in beats) that are realized in run-time (during performance) adds a new dimension and flexibility in the writing of temporal processes for mixed instrumental and computer music repertoire.

An interactive (or fixed) electronic music event might be bound to a single event in the score. In this case, a sequence of `FWD` commands with corresponding messages following the event in *Antescofo*'s score would do the work. The simple syntax of `FWD` is as follows:

```
FWD <symbol> +<message>
FWD <delay> <symbol> +<message>
```

where `<symbol>` is the string corresponding to the receiving symbol and `+<message>` correspond to *atom(s)* that would be sent to the symbol at the desired position. The `<delay>` option if exists, is a float number indicating the delayed time value in beats which would delay the clock-time for sending the message to outside processes using an internal scheduler coupled to the tempo agent.

The additional command `LFWD` (or loop forward) enables periodic messaging with the following syntax:

```
LFWD <name> <period> <symbol> +<message>
```

which upon triggering sends the indicated messages to a symbol in a periodic manner. The period is given in beats

and is coupled with the decoded tempo inside *Antescofo*. This simply means that the period of the looped message would change appropriately with the tempo of the musician(s). The `KILLFWD` command can stop a loop forward by calling the process' name anywhere in the score: `KILLFWD <name>`.

This feature of *Antescofo*'s score language is constantly growing in accordance with the needs of the computer music community using the developed software. For a comprehensive list of score actions and future plans of these features we invite the curious reader to try and follow the development in section 11.

## 9. TEMPO DECODING

The perception of time in music synchronization tasks is not merely an analysis of rhythmic content, and rather it shapes an *active listening* strategy in which the listener's expectations about future events can play a role as important as the musical events themselves. Therefore, any model for timing synchronisation of musical events should consider the hypothesis that the temporal structure of listeners' expectations is a dynamic structure. A primary function of such structure is *attentional* which allows *anticipation* of future events, enabling perceptual targeting, and coordination of action with musical events. These considerations led Large et al. [9] to design a cognitive model of tempo based on internal entrained oscillations.

The model used in *Antescofo* for decoding of the continuous tempo variable is highly inspired by [9]. Internal tempo is represented through a random variable $s_k$ revealing how fast the music is flowing with regards to the physical time. After Large, we model the behavior of such random variable as an internal oscillator entraining to the musician's performance. Such internal oscillation can be represented and modeled easily using sine circle maps. These models have been well-studied in the literature and can be considered as non-linear models of oscillations that entrain to a periodic signal and using discrete-time formalism. Using this framework, we represent the tempo random variable in $seconds/beat$ and note onset positions as phase values $\phi_k$ on the sine circle. This way, given a local tempo $s_k$, the onset time $t_n$ can be represented as $\phi_n = \frac{t_n}{s_k} + 2k\pi$ where $k$ is the number of tempo cycles to reach $t_n$. For our model, a phase advance would be the portion of the oscillator's period corresponding to note *Inter-Onset-Intervals* (IOI).

In order to compensate for temporal fluctuations during live music performance, we would need a function of $\phi$ that would *correct* the phase during live synchronization and at the same time model the *attentional* effect discussed previously, thus enabling perceptual targeting, and coordination of action with musical event. The attentional pulse can be modeled using a periodic probability density function, the *von Mises distribution* which is the circle map version of the Gaussian distribution. Since we are interested in the tempo update and not directly the phase, the corresponding attentional factor for the tempo variable would be a derivative of the von Mises distribution as depicted below [9]:

$$F(\phi, \kappa) = \frac{1}{2\pi \exp \kappa} \, e^{\kappa \cos(2\pi\phi)} \sin 2\pi\phi$$

Here, the parameter $\kappa$ plays an important role as smaller values spread the correction all over the phase domain. To take this fact into account, we accumulate attentional factors that occur during note IOIs and numerically solve for the best $\kappa$ that should be used during the next update.

It can be shown that the above considerations correspond to the following tempo update upon arrival of each new decoded score position from the audio agent:

$$s_{n+1} = s_n(1 + F(\phi_n, \kappa))$$

## 10. ANTICIPATORY INFERENCE FORMULATION

As mentioned earlier, the inference problem addresses decoding of the most likely state sequence in time associated with the observed audio sequence $x_0^{\tau-1}$. In a non-realtime context, an exact inference can be obtained using the Viterbi algorithm [15] that for each time $t$ uses both beliefs from time 0 through $t$ (referred to as *forward* or $\alpha$ probability) and future knowledge from present ($t$) to a terminal state at time $T$ (referred to as *backward* or $\beta$ probability). In a score following system that necessitates on-the-fly synchronization of audio with the music score, using the $\beta$ or backward probability of an exact inference framework is either impossible or would introduce considerable delays in the system. In the proposed system, we hope to compensate for this absence of future beliefs through our anticipatory model of audio/tempo coupled agents and an adaptive $\alpha$ calculation procedure. Here, we formulate a dynamic programming approach for an adaptive $\alpha$ calculation for a hidden hybrid Markov/semi-Markov process. The inference techniques shown here can be deducted and proofed mathematically from the framework discussed in section 6, which are left here out due to space considerations.

For a semi-Markovian state $j$, the Viterbi recursion of the forward variable is provided by the following dynamic programming formulation:

$$\begin{aligned} \alpha_j(t) &= b_j(x_t) \max\Big(\max_{1 \le u \le t} \Big(\Big\{ \prod_{v=1}^{u-1} b_j(x_{t-v}) \Big\} \\ &\cdot \; d_j(u) \max_{i \neq j}(p_{ij}\alpha_i(t-u))\Big)\Big) \end{aligned} \tag{2}$$

For a Markovian state $j$, the same objective amounts to:

$$\begin{aligned} \tilde{\alpha}_j(t) &= \max_{s_0,\ldots,s_{t-1}} P(S_t = j, S_0^{t-1} = s_0^{t-1}, X_0^t = x_0^t) \\ &= b_j(x_t) \max_i (\tilde{p_{ij}} \tilde{\alpha}_i(t-1)) \end{aligned} \tag{3}$$

Within this formulation, the probability of the observed sequence $x_0^{\tau-1}$ jointly with the most probable state sequence is $\operatorname*{argmax}_j [\alpha_j(\tau - 1)]$.

Equations 3 and 2 provide us with a recursive framework to decodes the real-time score position given that we know the *survival function* $d_j(u)$ for each event in the score in equation 2. In return, the tempo can be decoded correctly if we have the score positions of the current note and the previous notes as seen in section 9. To solve both issues, we couple both agents together where both are running in parallel and collaborating to decode the best score position and tempo altogether. While the influence of score position decoder on the tempo agent is evident, we need to refine the other way round mostly because the two agents run on different time scales.

To solve this burden, we consider a stochastic process $P(T_k)$ which models the estimated arrival time (in terms of number of analysis frames) for the $k^{th}$ event in the score as a *poisson process*. Since the *survival function* $d_j(u)$ is the inter-arrival time, using basic stochastic theory principles it can be easily shown that,

$$d_j(t_n - t_{n-1}) \quad = \quad e^{-\lambda(t_n - t_{n-1})}$$

where $\lambda$ is the expected number of occurrences that one would expect to happen during the given time interval. During the inference framework of equation 2, $t_n$ being the real-time and $t_{n-1}$ the last decoded position, we can easily obtain $\lambda$ by using the latest decoded tempo and the event duration (in beats) from the score.

## 11. DEVELOPMENT AND DISCUSSION

*Antescofo* has been developped using advanced C++ template libraries and is available for free from the author's website for Max/MSP and PureData environments on their respective operating systems. *Antescofo*'s described score language is also exportable through the *NoteAbility Pro* commercial music notation software [2]. Figure 6 shows a snapshot of the help file that comes with the downloadable package with demonstrative examples and tutorials:
http://cosmal.ucsd.edu/arshia/antescofo/

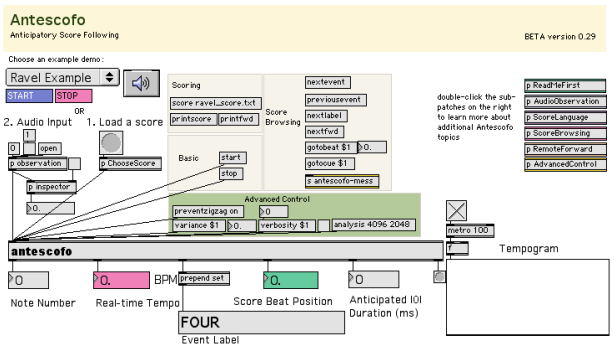



**Figure 6**. *Antescofo*'s Help snapshot in Max/MSP

*Antescofo* was primarily conceived for Marco Stroppa's "... *of Silence*" for saxophone and chamber electronics, premiered on Nov. 23rd 2007 in Shizuoka, Japan. Since then, it has been used in several concerts including Boulez' "... *Explosante-Fixe...*" with Los Angeles Philharmonic in January 2008 and several new productions are underway. In this paper we demonstrated basic foundations and concepts of *Antescofo*. We leave a more rigorous discussion on the design, evaluation and exploitation of the system to a future communication.

[2] http://debussy.music.ubc.ca/NoteAbility/

## 12. ACKNOWLEDGMENTS

The author would like to strongly acknowledge his collaboration with Marco Stroppa on this project, without whose strong musical intuition and perseverance this project would never reach its current state.

## 13. REFERENCES

[1] Miller Puckette. A divide between 'compositional' and 'performative' aspects of pd. In *First International Pd Convention*, Graz, Austria., 2004.

[2] Marco Stroppa. Live electronics or live music? towards a critique of interaction. *Contemporary Music Review*, 18(3):41–77, 1999.

[3] Jean-Claude Risset. Composing in real-time? *Contemporary Music Review*, 18(3):31–39, 1999.

[4] Roger B. Dannenberg. An on-line algorithm for real-time accompaniment. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 193–198, 1984.

[5] Barry Vercoe. The synthetic performer in the context of live performance. In *Proceedings of the ICMC*, pages 199–200, 1984.

[6] Lorin Grubb and Roger B. Dannenberg. A Stochastic Method of Tracking a Vocal Performer. In *Proceedings of the ICMC*, pages 301–308, 1997.

[7] Christopher Raphael. Aligning music audio with symbolic scores using a hybrid graphical model. *Machine Learning*, 65(2-3):389–409, 2006.

[8] Arshia Cont. Improvement of observation modeling for score following. Dea atiam, University of Paris 6, IRCAM, Paris, 2004.

[9] Edward W. Large. Periodicity, pattern formation, and metric structure. *Journal of New Music Research*, 22:173–185, 2001.

[10] Robert Rosen. *Anticipatory Systems*, volume 1 of *IFSR International Series on Systems Science and Engineering*. Pergamon Press, Oxford, 1985.

[11] Yann Guédon. Hidden hybrid markov/semi-markov chains. *Computational Statistics and Data Analysis*, 49:663–688, 2005.

[12] Ircam. Colloque international écritures du temps et de l'interaction. In *Agora Festival*, Paris, France., June 2006. Ircam-Centre Pompidou.

[13] I. Xenakis. *Formalized Music*. University of Indiana Press, 1971.

[14] Pierre Boulez. *Penser la Musique Aujourd'hui*. Gallimard, 1964.

[15] Kevin P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science Division, 2002.