音 楽 情 報 科 学 23－3
（1997．12．15）

Score following and real-time signal processing strategies in open-form compositions

Barry Moon

University at Buffalo-SUNY
Hiller Computer Music Studios
brmoon@acsu.buffalo.edu

In my most recent real-time computer music composition, entitled Interact I, I attempt to create a higher level of interaction between performer and computer by combining the formal openness of performance-driven systems with a score that enables the form to be interactively built of processes linked by an evolving logic. The performer is given a score with explicit instructions as to how they are to build larger forms using the fragmented materials in response to the computer's behavior. The computer follows an "electronic score" of the entire piece, and responds by making changes in processing according to predetermined algorithms and level of accuracy in following.

開かれた形式の作曲における スコア・フォローイングとリアルタイム信号処理戦略

バリー・ムーン

ニューヨーク州立大学バッファロー校
Hiller コンピュータ音楽スタジオ
brmoon@acsu.buffalo.edu

私の最も新しいリアルタイム・コンピュータ音楽作品「Interact I」の中で、私は発展しつづける論理により結びつけられたプロセスをインタラクティブに構築する形式を可能とする楽譜とともに、演奏により動かされるシステムの形式的開放性を結びつけることにより演奏者とコンピュータの高度な相互作用を生み出すことを試みた。演奏者はコンピュータの動作に応じて、いかにして断片的な素材をより大きな形式へ構築していくかを明確に指示されたスコアを与えらる。コンピュータは、作品全体の「電子スコア」を追いつつ、あらかじめ定められたアルゴリズムとフォローイングの正確さのレベルに応じて、プロセスの中で変更を加えながら反応していく。

Background

Real-time computer music places the composer in an historically unique relationship between composition and performance. The computer is both an instrument and performer, with almost boundless possibilities of sound production and transformation, whose role in any capacity is directed by the composer's choices. These choices are limited only by the technical ability and imagination of the composer. The performer, on the other hand, has technique and imagination which can be guided, but not fully controlled by the composer. Thus, the end result of the interaction between computer and performer is a consequence of these complex relationships.

Robert Rowe states; "Interactive (computer music) systems exhibit changing behavior in response to human input". (Rowe, 1993) The dictionary definition of the word interactive, "capable of acting on or influencing each other", implies that the interactive music system must also include a performer who completes the loop. Therefore, levels of interaction can be gauged by the potential for changes in the behaviors of computer and performer in their response to each other.

Rowe classifies two types of interactive systems; "Score-driven programs use predetermined event collections, or stored music fragments, to match against music arriving at the input... Performance-driven programs do not anticipate the realization of any particular score." (Rowe, 1993)

Each of the two systems classified by Rowe has levels of interaction inherent in the techniques associated with them. In performance-driven systems there is often great potential for changes in behaviors offset by an immediacy of response which can negate the building of more complex large scale forms. And, although the performer can practice with the computer to develop a repertoire of sounds that work well with certain processes, it can be difficult for performer or computer to change between those processes in a meaningful way. In score-driven systems, performers are often told that they need to play the score accurately to have the computer follow them. The performer, in this case, often has a high level of local interaction after triggering global processes via score following, but inaccurate triggering is often seen as a problem. Also, the large scale form is governed by the score and not susceptible to interaction.

Why Open-Form?

Through my experience with interactive systems I began to realize that the combination of score and performance-driven systems might provide a greater level of interaction than either system does alone. In my efforts to combine the two approaches, performers are given a score with explicit instructions as to how they are to build larger forms using the fragmented materials in response to the computer's behavior. The computer follows an "electronic score" of the entire piece, and responds by making changes in processing according to predetermined algorithms and levels of accuracy in following. Thus, inaccuracies in following become a determinate of interaction in a positive way. The formal openness of performance-driven systems is combined with a score that enables the form to be interactively built of processes linked by an evolving logic.

Form is one aspect of composition which can sometimes be too obviously perceived as a contrivance which pushes or leads a listener's attention towards a composer's goals. I believe that adopting contrived, classical forms is contrary to the experimental nature of computer music. By using open-form, and putting some control over form in the hands of the performer, my aim is to direct more of the listener's attention towards the performer, where formal procedures become an integral part of the interaction between listener and performer. In the context of real-time computer music, where my choices direct the interaction between computer and performer, the control over open-form is only partially placed in the hands of the performer. This last sentiment can be seen as a response to Xenakis, and the numerous composers who seem to share his view that:
"The composer commits an act of resignation when he admits several possible and equivalent circuits. In the name of a scheme the problem of choice is betrayed, and it is the interpreter who is promoted to the rank of composer by the composer himself." (Xenakis, 1992)

INTERACT I

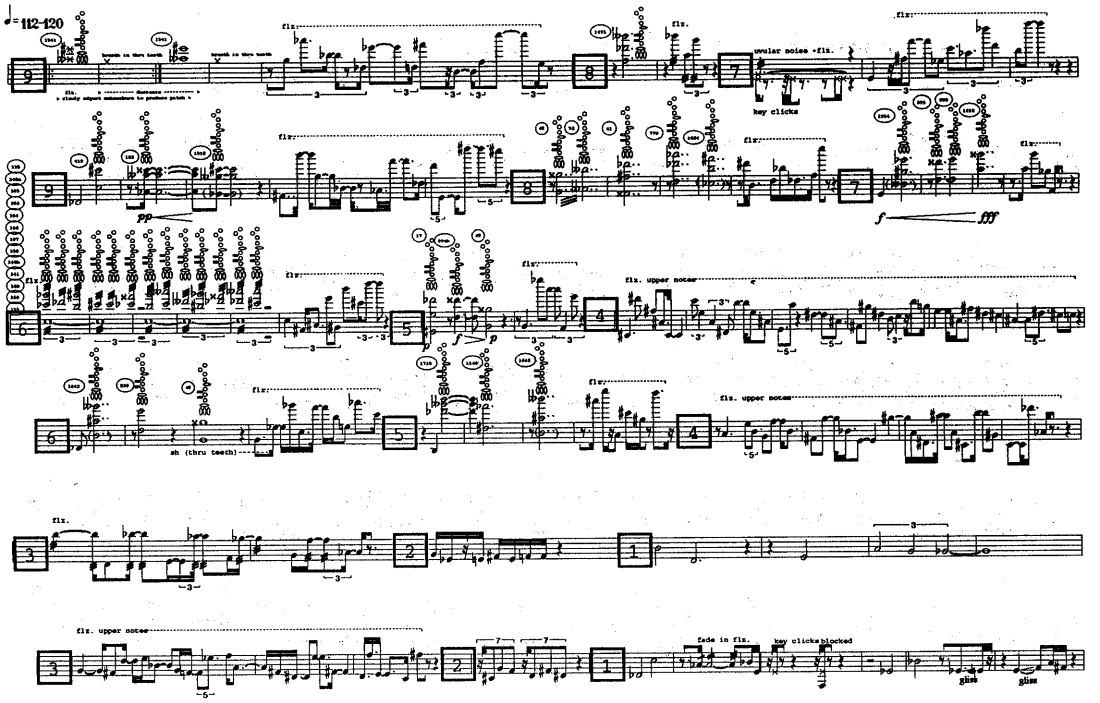
"Interact I", for Flute and ISPW (Ircam Signal Processing Workstation), written in 1996, is my first attempt at an open-form, real-time computer music composition.

The Score

The score to "Interact I" is divided into three parts. Each part broadly characterizes transitions in specific musical parameters; transitions in duration/speed in Part I, transitions in timbre in Part II, and transitions in dynamics in Part III. Each part has a sequence of materials which is recorded for score-following in the computer. I will refer to this order of materials as the "normal sequence".

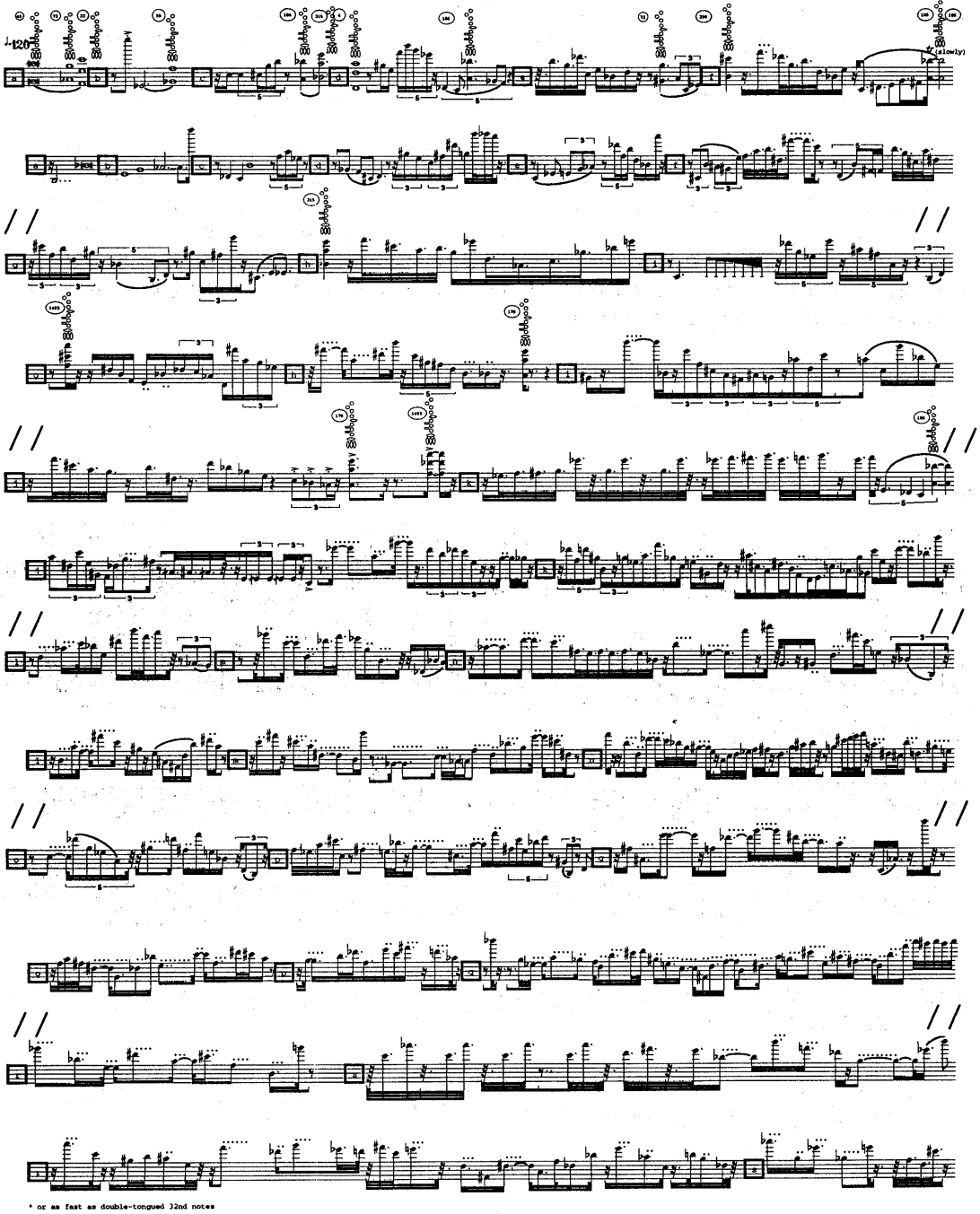
The normal sequence for Part I, (Ex. 1), is standard alphabetical order, (a-s), for the upper staff, and reverse alphabetical order, (s-a), for the lower staff. In performance, a transition can be made by going in either standard or reverse alphabetical sequence. Thus, the player can direct a transition from long duration, (slow), to short duration, (fast), by playing through the upper staff in standard alphabetical order, or from short duration to long duration by playing through the lower staff in reverse alphabetical sequence. Additionally, the player can choose to keep the duration steady by swapping back and forth between staves within a single lettered block.

Example 1



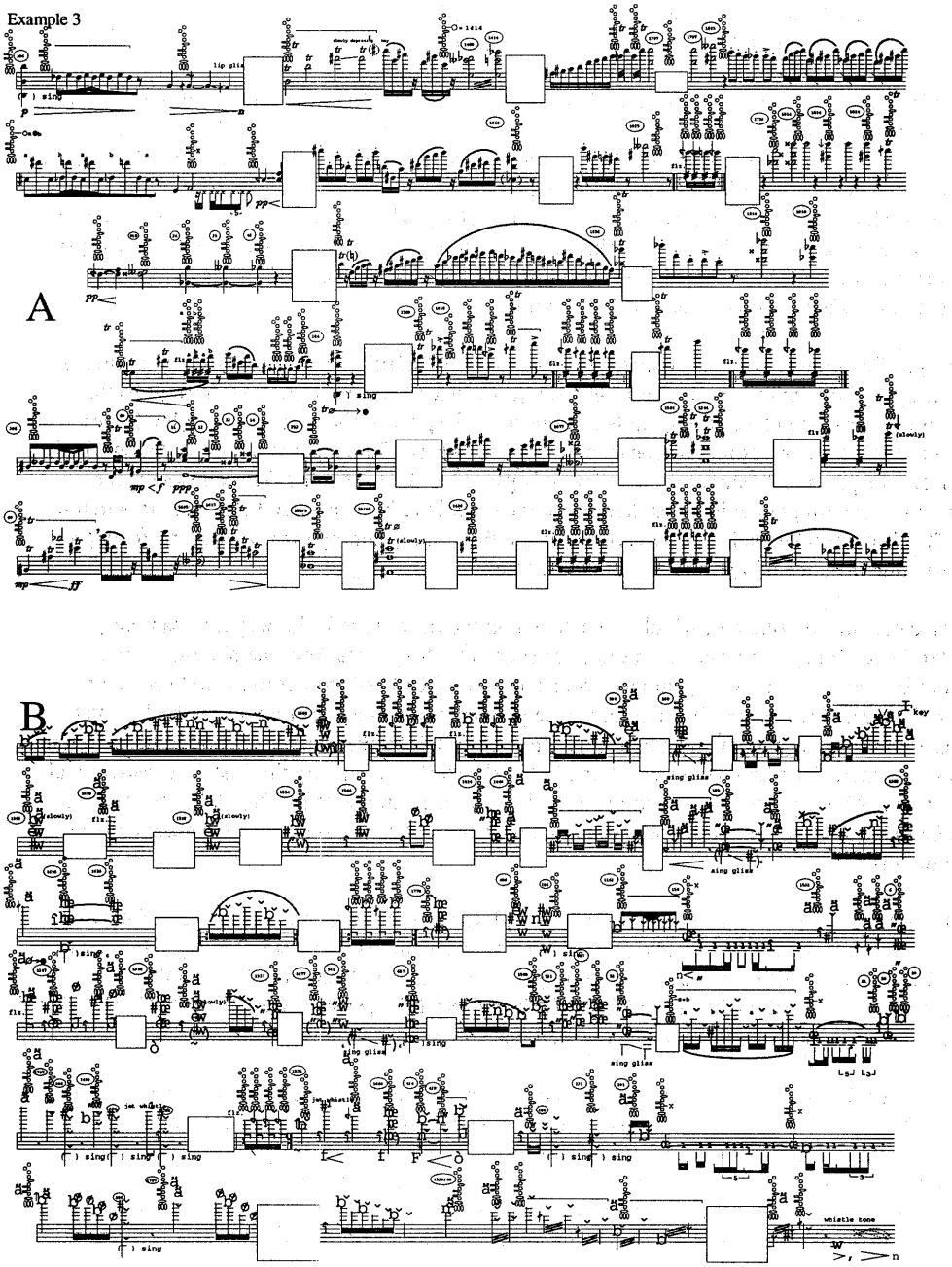
In Part II, (Ex. 2), the player can direct a transition from noisy to pitched materials by playing the upper staff in descending numerical order, a transition from pitched to noisy materials by playing the lower staff in ascending numerical order, or remain in a static timbral area by alternating between staves within a single numbered block.

Example 2



Part III, (Ex. 3), works somewhat differently from the other two parts. Only the top four staves, which the performer plays to direct a transition from quiet to loud dynamics, and bottom two staves, which the performer plays to direct the opposite transition, are recorded in the computer for following, where as the other six staves are intended for improvisation on materials with static dynamics. The normal sequence for both the top four and bottom two staves is visually left to right, delineated by blocks or measures. To remain within one dynamic area, the player traverses the six staves in the center of the part towards points which are symmetrically opposite, such that top left and bottom right are equivalent, as are top right and bottom left.

Example 3



The Score-Following Algorithm

A detailed flow chart of my score-following algorithm can be seen in Example 4. The two inputs to the algorithm are pitch-tracker derived MIDI note numbers and control values coming from MIDI foot-pedals. The player steps on one of three foot-pedals which corresponds to Part I, II or III of the score. This directs the pitch-tracker output to search through smaller sections of the electronic score. It was necessary to break up the electronic score into smaller sections because the score-following algorithm is computationally very expensive. The other step taken to reduce the amount of computation is a data reduction system which filters out notes from the pitch-tracker that are repeated in close proximity, such as those that occur in trills and short sequences, for both the recording and subsequent following of the electronic score.

One fundamental way in which this score-following algorithm differs from those commonly adopted is that a history of the input is kept. Score-following algorithms such as "follow" in Max, ignore input if it doesn't match with the score. There are essentially two sides to my algorithm; one, which I have labeled "past", keeps track of how out-of-date the input has become since a correct note was found, and the other labeled "future" which, like the "follow" object, allows the computer to look ahead in the score in case input from the pitch-tracker reflects "wrong" or "skipped" notes by the performer. (One must be cautious when describing the concepts of "wrong" or "skipped" notes to a performer, when it is more often the computer which is at fault!)

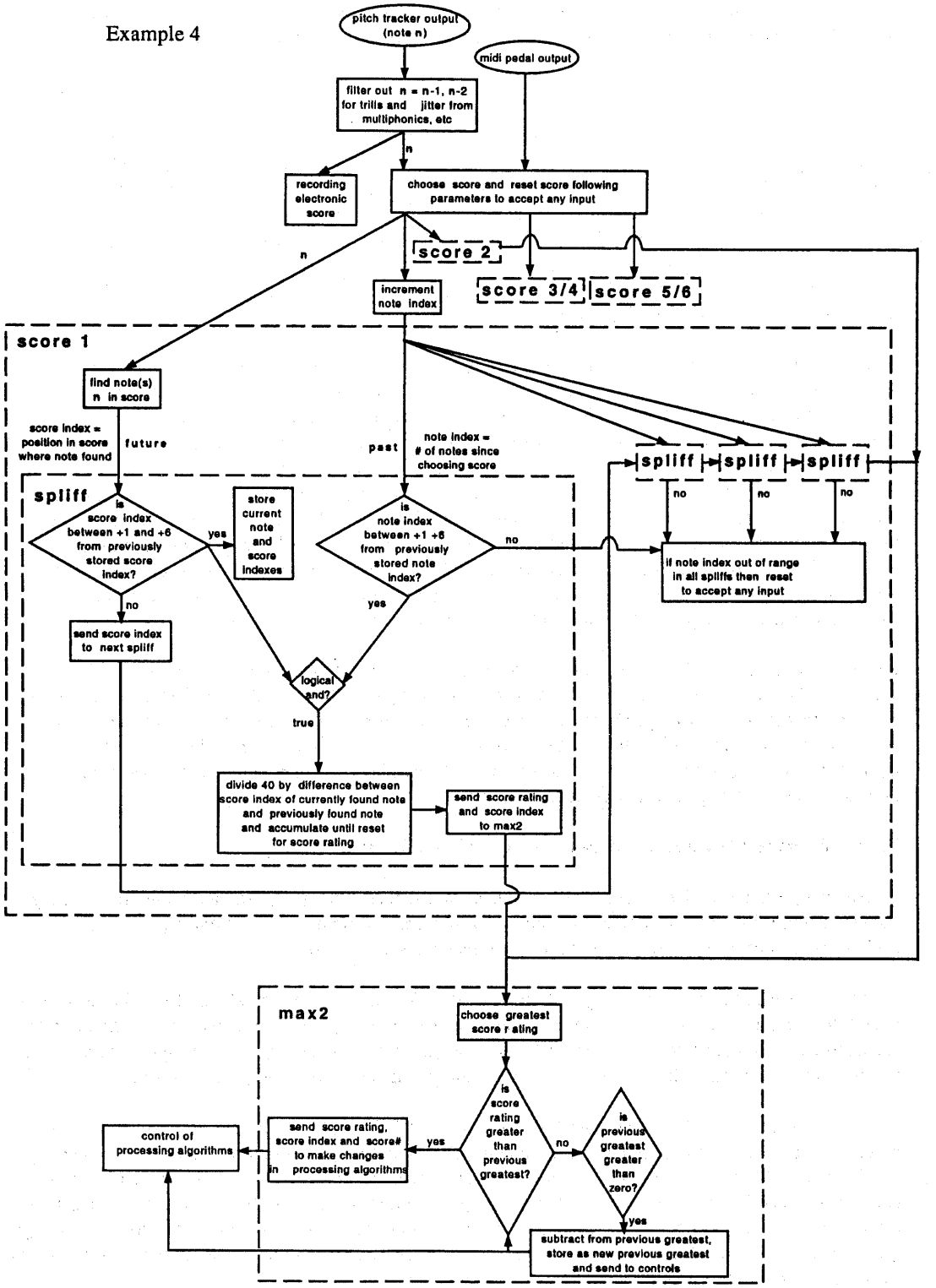
Referring to Example 4, we can see that the only way for a "spliff" to act as though it is following successfully is if the input is both "fresh" enough (note index between +1 and +6 from previously stored note index), and close enough in the future from the previously stored score index (score index between +1 and +6 from previously stored score index). If the input is not close enough in the future from the previously stored score index, it is passed to the next spliff. This allows for the follower to be looking at the input as though the input could refer to several different places in the score. This is especially useful in the case (of which there are generally many) where there exists more than one instance of a particular note in a score.

Another aspect of my follower, which is not shown in the flow chart, is the way the searching algorithm looks through the electronic score for each note that comes from the pitch-tracker. It alternates between searching sequentially from start to end, and end to start to avoid bias being given to the start of the score. This bias can be explained if one imagines that the spliffs had just been reset to accept any input and a note came from the pitch-tracker that was found in ten different places in the score. Reading from start to end in the electronic score, only the first four instances of that note would be registered by the follower (limited by the number of spliffs). A better solution would be to search through the score in random order, but I have yet to find a computationally cheap way to do this.

If the input from the pitch-tracker is not fresh enough for any of the spliffs to register as following, they are all reset to accept any input again. When a single spliff registers as following (close and fresh), the difference between the score index of the previously and currently found notes is calculated as an indication of how well it followed at that particular instance (score-rating). This score-rating is accumulated to indicate how well that spliff is following generally.

The next stage, labeled max2 in Example 4, determines the greatest score-rating coming from all eight spliffs in the two parts being followed (there may be any number of score-ratings between 0 and 8, since the ratings don't come from a spliff that did not register successful following for a given note). If this score-rating is greater than that previously stored, it is stored, and sent, along with the score-index and the score number, to make changes in the processing algorithms. If it is not greater, some value is subtracted from the previous greatest, stored as a new previous greatest and sent to

Example 4


pitch tracker output (note n)
midi pedal output
filter out n = n-1, n-2 for trills and jitter from multiphonics, etc
n
recording electronic score
choose score and reset score following parameters to accept any input
n
score 2
score 3/4
score 5/6
increment note index
score 1
find note(s) n in score
score index = position in score where note found
future
past
note index = # of notes since choosing score
spliff
spliff
spliff
spliff
no
no
no
spliff
is score index between +1 and +6 from previously stored score index?
yes
store current note and score indexes
is note index between +1 +6 from previously stored note index?
no
if note index out of range in all spliffs then reset to accept any input
no
send score index to next spliff
yes
logical and?
true
divide 40 by difference between score index of currently found note and previously found note and accumulate until reset for score rating
send score rating and score index to max2
max2
choose greatest score r ating
is score rating greater than previous greatest?
yes
no
is previous greatest greater than zero?
yes
send score rating, score index and score# to make changes in processing algorithms
control of processing algorithms
subtract from previous greatest, store as new previous greatest and send to controls

the processing algorithms. This allows for the score-follower to become less critical if it is not following successfully and for the poor following to be reflected in the processing algorithms.

The Processing

There are three types of processing used, each of which somehow relates to one of the three parts of the score. The processing which relates to Part I (duration transitions), is a real-time sampler which records and plays back lengths of sounds based on the input crossing amplitude thresholds. The sample playback is then sent through ring modulation. The processing which relates to Part II (timbral transitions) is a stereo harmonizer. The two intervals of transposition of this harmonizer are taken from determining the three loudest spectral peaks of the input using the jack~ object (see below), and then determining whether those frequencies are even multiples of one another in the attempt to base the intervals on multiphonic complexes rather than pure tones.

The third type of processing, relating to Part III (dynamic transitions), is a mixture of amplitude modulation (AM) and frequency modulation (AM) of sine tones generated using oscillators based on output of the jack~ object. Briefly, the jack~ object is an analysis object which outputs a list of frequency/amplitude pairs for every analysis window (in this case, a list of the 10 loudest spectral peaks every 32 milliseconds). This analysis information being sent to oscillators will re-synthesize the input sound. I use two different methods of driving the oscillators; one which uses the analysis amplitudes to drive the amplitudes of the oscillators, and another, which splits analysis amplitudes within a certain range to make notes of statistically calculated durations to drive the oscillators. The output of each oscillator is delayed using a variable delay and some amount of that delayed signal is sent back to the oscillator to create AM and/or FM. The combined output of the oscillators is then fed through a recirculating stereo delay which is used to sustain the sound when the flute input stops.

I have discussed some of the ways in which processing is based on the immediate input sound. Other, more global controls, are determined by the score-following algorithm. On the simplest level, the order of the performer's choices, and the subsequent score-rating coming from the score-follower, control the levels of a virtual mixer. After the first two choices have been made, there are twelve different algorithms used to control this mixer. Twelve being the number of possible choices of three parts in three samples, allowing only non-adjacent repetition.

The ten faders on the virtual mixer can be described as follows:

flute to sampler, harmonizer to sampler, resynthesis oscillators to sampler, sampler to digital-to-analog converter (dac), flute to harmonizer, sampler to harmonizer, harmonizer to harmonizer, resynthesis oscillators to harmonizer, harmonizer to dac, and resynthesis oscillators to dac

There are generally two or three stages of mixer control for the score-rating to be broken up into. This is where the receding score-rating due to poor following becomes apparent to the performer, who hears the processing returning to a sound similar to that which was previously heard. Another aspect of the virtual mixer is that it can remember its settings to avoid abrupt changes where they are not desired.

The score index coming from the score-follower makes global changes to processing algorithms. These global changes control the sampler's amplitude thresholds and ring modulation frequency ranges (range for choice of random numbers), the harmonizer's preference for upwards or downwards transposition and window size (window size relating to a panning effect due to the separation of windows to right and left channels), and a number of parameters having to do with the resynthesis using jack~ data, including; type of oscillator control (amplitude from input or note made from amplitude split), range of AM and FM depths (a range of random numbers), delay times and delay algorithms.

Future Directions

There are many alternatives to the techniques I have used in "Interact I" for following and processing open-form compositions. A piece I am planning for solo cello and computer will have a group of notes assigned to certain phonetic symbols, and glissando speeds assigned to others. The score will have these notes blocked out spatially on the score with separate directions for the durational/rhythmic, dynamic and timbral information. The computer will determine which of the phonetic symbols the performer is trying to invoke by simply matching notes coming for the pitch-tracker and measuring any glissando present. It can then search through a dictionary of phonetic translations of the English language (a dictionary available for downloading from Carnegie Mellon University) and make a choice of a corresponding phonetic sound based on grammatical formations. This process of response can continue until words and sentences are formed. The logic behind this piece is the dichotomy of a speaking cello and a listening computer, neither of which are immediately obvious paradigms.

References

Rowe, R. 1993. "Interactive Music Systems" The MIT Press, Cambridge, Massachusetts.

Xenakis, I. 1992. "Formalized Music: Thought and Mathematics in Composition" Pendragon Press, Styvesant, New York.