groupe
représentation
et traitement
des
connaissances

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
31, chemin Joseph Aiguier
F-13402 MARSEILLE CEDEX 20 (France)

téléphone: *(33) 91 16 40 64*
telex: *CNRSMAR 43 0225 F*
fax: *(33) 91 71 08 08*

Time-setting of sound-objects: a constraint-satisfaction approach

Bernard Bel

Abstract

This paper deals with the scheduling of "sound-objects", hereby meaning predefined sequences of elementary tasks in a sound processor, with each task mapped to a time-point. Given a structure of sound-objects completely ordered in a phase diagram, an "instance" of the structure may be obtained by computing the dates at which each task should be executed. Time-setting the structure amounts to solving a system of constraints depending on (1) metric and topological properties of sound-objects, (2) contexts in which they are found, and (3) parameters related to the performance itself ("smooth" or "striated" time, speed, etc.). This may require relocating/truncating objects or delaying part of the sound-object structure. A constraint-satisfaction algorithm is introduced, the time complexity of which is O(n.k) in most cases, where n is the number of sequences and k the maximum length of a sequence. In the worst case it remains better than $O(n^2.k^3)$. Other fields of applications are proposed, including multimedia performance and computer-aided video editing.

Keywords

Constraint satisfaction, task scheduling, synchronization, AI-music, multimedia performance, video editing.

Invited paper, *Workshop on Sonic Representations and Transforms*. INTERNATIONAL SCHOOL FOR ADVANCED STUDIES (ISAS), Trieste, 26-30 October 1992.

Time-setting of sound-objects: a constraint-satisfaction approach

Time-setting of sound-objects: a constraint-satisfaction approach

Bernard Bel
Groupe Représentation et Traitement des Connaissances (GRTC)
Centre National de la Recherche Scientifique
31, chemin Joseph Aiguier
F-13402 Marseille Cedex 9. E-mail: bel@grtc.cnrs-mrs.fr Tel. (033) 91 16 42 56

1. Introduction

Work presented in this paper originated from the need to design a "sonological component" in a computer environment for rule-based music composition, namely **Bol Processor BP2** [Bel 1993a].[1] The idea was to deal with **sound-objects** handled both at the symbolic level (where each object is represented by an arbitrary symbol, and symbols arranged in strings, tables, trees, etc…) and at the lower level of **elementary tasks** mapped to time-points. These tasks are messages dispatched to a sound processor to trigger and control sound synthesis processes [Bel 1990b, 1991a-b]. The format of messages depends on the hardware/software configuration of the sound processor and on the communication device used for its control. A sound-object may for instance be a "note", i.e. a NoteOn/NoteOff pair of "events" (elementary tasks) in the widely used MIDI environment (*Musical Instrument Digital Interface*).

In this context, music representation may be reduced to strings of symbols over a finite alphabet in which each symbol denotes a particular sound-object. These strings may be superimposed in various ways, yielding bi-dimensional **phase diagrams**. Consider for instance alphabet A = {a,b,c,d,e,f,g,h,i,j} and the three strings:

S1 = abcde S2 = afgh S3 = jiai

To define a structure in which S1, S2 and S3 are superimposed it is necessary to determine the time ordering of all sound-objects. String representation implicitly imposes a strict ordering of sound-objects contained therein, e.g. it is clear that in S1 "a" should precede "b", but additional information is needed to compare sound-objects belonging to different sequences, e.g. "c" in S1 with "g" in S2.

Strings actually represent strictly ordered sets (**sequences**) of sound-objects in which the precedence relation is an ordering on **symbolic time** [Bel 1990b:101-ff], an idea borrowed from Jaffes' [1985] *virtual time* (see also [Anderson & Kuivila 1989]).

[1] Bol Processor BP2 runs on Macintosh™ computers and is distributed as shareware by the author.

A possible superimposition of sequences S1, S2 and S3 would be for instance:

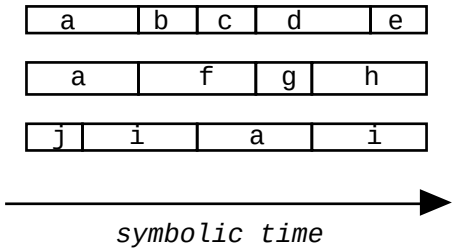


Fig.1 Superimposing sequences of sound-objects

This picture makes all precedence relations explicit. It is equivalent to the **phase diagram** shown Fig.2, in which symbol "_" is used to prolongate the preceding sound-object, thereby determining its **symbolic duration** [Bel 1991b:7,33]. NIL's mark the ends of sequences.

a	_	b	c	d	_	e	NIL
a	_	f	_	g	h	_	NIL
j	i	_	a	_	i	_	NIL

Fig.2 A phase diagram

The phase diagram may in turn be represented as a string of symbols that we call a **polymetric expression**. Using our convention [Bel 1990a-b, 1991a-b, 1993a-b] the preceding diagram would be notated:

{a_ b c d_ e , a_ f_ g h_ , j i_ a_ i_}

The use of commas and curled brackets is self-explanatory. Multileveled expressions may be constructed in a similar way. Bol Processor BP2 is able to deal with incomplete representations, i.e. to infer the "most evident" complete polymetric expression matching an incomplete expression. This process is explained in [Bel 1990a, 1990b:108-130, 1991a, 1991b:8-27] and illustrated by examples in conventional music notation.

Given (1) the phase diagram of a structure of sound-objects, (2) descriptions of sound-objects (**sound-object prototypes**) along with their metrical/topological properties (see §4), and (3) information about physical time (a list of dates called the **structure of time**), the **time-setting algorithm** introduced in this paper attempts to solve the system of constraints imposed by topological properties, thereby enabling the computation of the physical dates of all elementary tasks contained in sound-objects.

Since this algorithm does not rely on specific musical concepts it may be applied to various applications in which similar metrical/topological properties may be assigned to classes of "time-objects" similar to sound-objects:

— In **multiple-screen video editing**, time-objects could be predetermined picture sequences. A string of time-objects is the script of a "story" displayed on a single video monitor. This story may be predefined or edited at performance time. (Real-time editing is now possible thanks to laser-disc video storage.) Overlapping time-objects in a story may be displayed as dissolved sequences. The time-setting algorithm is needed when several stories on different screens need to be interrelated.

— **Multimedia performance** involving light and sound systems, laser shows, water devices, fire works, etc., may be controlled in a similar way.

2. The time-setting problem

When dealing with sound structures, Stroppa [1990] suggested that any sound-object could be assigned one (or several) time-points playing a particular role, e.g. the actual attack of a note (not necessarily at the very beginning of its sampled sound). These points (time **pivots**) are used for synchronizing sound-objects belonging to different sequences. To this effect, **anchoring points** are computed from pivots in each particular context and help defining sound-object locations unambiguously.

Here we use a simpler model in which the anchoring point is the *unique* pivot of a sound-object. To synchronize sound-objects, pivots will be located on particular time points that we call **streaks**. These streaks may be predefined (often the case in music, where they indicate regular or irregular pulses, see [Boulez 1960:107]) or partly determined by the sound-object structure itself. The list of time-streak dates may be called the **structure of time** [Bel 1990a, 1991a, 1991b:6], an term we borrowed from Xenakis [1963:190-191, 200].

Sometimes it is not possible (or irrelevant) to assign a pivot to a sound-object. We say that a sound-object is **striated** if it has a known pivot, and **smooth** otherwise. To simplify computation we consider that even smooth sound-objects have pivots (with arbitrary locations) but they are **relocatable**, thereby meaning that their pivots need not be precisely located on time streaks.

Consider the sound-object structure defined in §1 and suppose that all sound-objects labelled “a”, “b”, “c”, …, “j” have been fully defined. The definition of a sound-object contains the relative location of its pivot along with metrical properties allowing the computation of its **time-scale ratio** — informally, a factor adjusting the physical duration of the sound-object to the current speed of performance.

Fig.3 is a graphic representation of a possible **instance** of this sound-object structure:

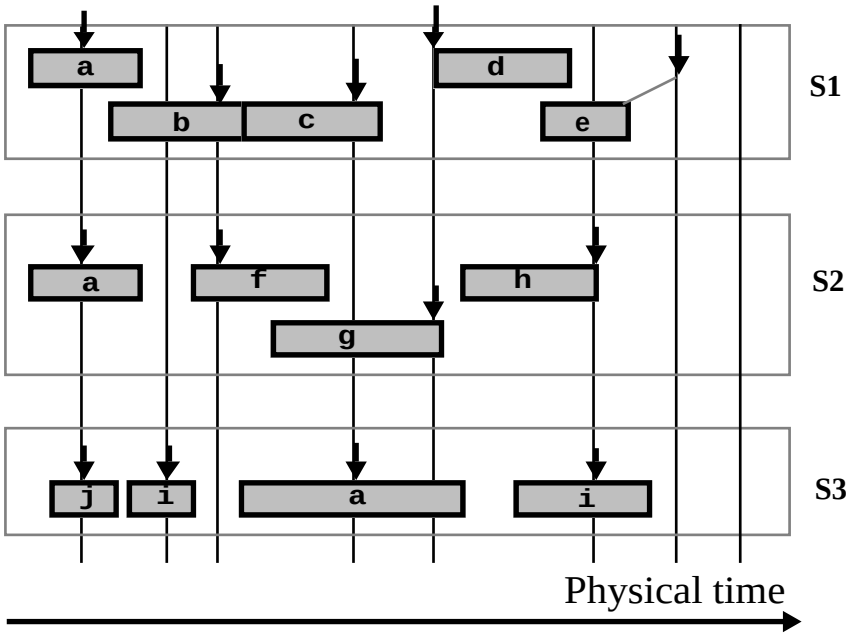


Fig.3 An instance of the sound-object structure

The structure of time is for instance an irregular pulsation represented with vertical lines (**time streaks**). The time-span interval of each sound-object is shown as a rectangle with arbitrary vertical width and vertical position. These positions have been chosen to separate objects on the graphic: it is clear for example that “c”, “f”, “g” and “a” have overlapping time-span intervals between the third and fourth streaks. Lengths of rectangles represent the physical durations of sound-objects. These durations depend on the structure of time. For example, although the two instances of “i” in S3 have identical symbolic durations, their physical durations (i.e. their time-scale ratios) are different because the “beat offsets” (i.e. the time interval from one streak to the next) are different.

Vertical arrows indicate time pivots. As shown with object “e”, the pivot is not necessarily a time point within the time-span interval of the sound-object.

This graphic represents the **default positioning** of objects with their pivots located exactly on time streaks. Although it is reasonable that instances of “c”, “f” and “a” are overlapping between the third and fourth streaks since they belong to distinct sequences which are performed simultaneously, it may not be acceptable that “f” overlaps “g” in a single sequence S2; the same with “d” and “e” in sequence S1. For similar reasons, it may not be acceptable that the time-span intervals of “j” and “i” are disjoint in sequence S3 while no silence is shown in the symbolic representation.

How could one deal with a constraint such as <<*the end of sound-object “f” may not overlap another sound-object in the same sequence*>> ? If object “g” is relocatable then it may be delayed (shifted to the right) until the constraint is satisfied. We call this a **local drift** of the object. However, the end of “g” will now overlap the beginning of “h”. Assume that this also is not acceptable and “h” is not relocatable. One should therefore look for another solution, for example truncate the beginning of “h”. If this and other solutions are not acceptable then one may try to shift “f” to the left or to truncate its end. In the first case it might be necessary to shift or truncate “a” as well.

In the time-setting algorithm the three sequences are considered in order S1, S2, S3. Suppose that the default positioning of objects in S1 satisfies all constraints and no solution has been found to avoid the overlapping of “f” and “g” in S2. Another option is to envisage a **global drift** to the right of all objects following “f” in S2. All time streaks following the third one are delayed (see dotted vertical lines):

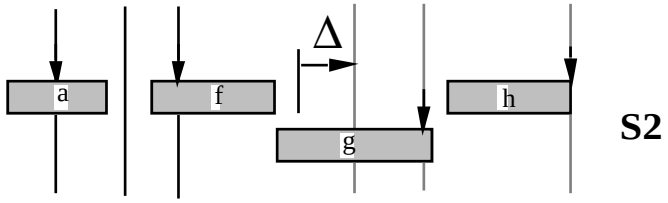


Fig.4 A structure with global drift Δ

Now the positioning of objects in S2 is acceptable but it might have become unacceptable in S1. For instance, there may be a property of “b” or “c” saying that their time-span intervals cannot be disjoint, so “c” should be shifted to the left, etc. Evidently, whenever a global drift is decided the algorithm must start again from the first sequence.

3. Main data structures

Let the structure of time be encoded as a list of physical dates T(i) with i = 1,…, imax. Each T(i) is the date of the **reference streak** of sound-objects with **rank *i*** in the structure.

Consider for instance the phase diagram:

a _ b _ a _ _ b _ _ c _ d _ NIL

_ _ _ _ c _ d _ e _ NIL

Symbols “a”, “b”,… designate sound-objects E_k, where *k* is an arbitrary index ($k \geq -1$). Prolongation symbol “_” may be assigned the **null sound-object** E_0. Similarly, “NIL” is mapped to a **virtual sound-object** E_{-1} delimitating the end of each sequence.

Strictly positive values of *k* in this structure may be for instance:

1		**2**		**3**			**4**			**5**		**6**
a	_	b	_	a	_	_	b	_	_	c	_	d	_
				c	_	d	_	e	_				
				7		**8**		**9**					

All *k*'s are contained in a bidimensional array called the **phase table** Seq(nseq,i):

i =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
nseq = 1	1	0	2	0	3	0	0	4	0	0	5	0	6	0	-1
nseq = 2	0	0	0	0	7	0	8	0	9	0	-1	0	0	0	0

Thus, E_k is the k-th sound-object, given k = Seq(nseq,i). Column index *i* is the rank of E_k in its sequence. Let Obj(k) be another table yielding an index *j* pointing at the **symbol table**:

j =	0	1	2	3	4	5
symbol	_	a	b	c	d	e

Thus, each E_k (with $k \geq 0$) may be seen as an **instance** of a **sound-object prototype** Ep_j with j = Obj(k). If k = 0 the sound-object is null (labelled "_"), therefore Ep_0 is the **null sound-object prototype**.

Dates t1(k) and t2(k) are the physical **start/clip date**s of E_k, and $\alpha(k)$ its **time-scale ratio**. If $\alpha(k) > 1$ the sound-object is performed slower than its prototype. Rules used for calculating $\alpha(k)$ in various contexts are discussed in [Bel 1990b,1991a-b]. All time-scale ratios are computed before the constraint-satisfaction procedure is started.

Once an object E_k (with k = Seq(nseq,i)) has been instantiated by the time-setting algorithm, its pivot is located at physical date:

$$T(i) + \Delta(i) + \delta(k)$$

in which $\delta(k)$ is the **local drift** of E_k and $\Delta(i)$ the **global drift** of its reference streak. If $\delta(k) \neq 0$ the object has been **relocated**.

The beginning of an object may be truncated, i.e. elementary tasks contained in the truncated part may be delayed or deleted [Bel 1991b:37, 63-64]. We notate TrBeg(k) the physical duration of the truncated part. Similarly, TrEnd(k) represents the duration of the part truncated in the end of E_k.

4. Sound-object properties

Properties of sound-objects relevant to the time-setting algorithm are summarized. These are discussed in length in [Bel 1991a, 1991b:34-37].

Let Ep_j be the j-th sound-object prototype and E_k an instance of Ep_j with rank *i* in its sequence. **Reloc(j)** means that E_k is relocatable. **BrkTempo(j)** allows a global drift of all objects with rank i' > i. **OverBeg(j)** means that the beginning of E_k may overlap another object in the same sequence. **OverEnd(j)** means that its end may overlap another object in the same sequence. **TruncBeg(j)** means that the beginning of E_k may be truncated in case it overlaps another object in the same sequence. **TruncEnd(j)** means that the end of E_k may be truncated when overlapping another object in the same sequence. **ContBeg(j)** means that the beginning of E_k must join or overlap the end of a preceding object in the same sequence.

ContEnd(j) means that the end of E_k must join or overlap the beginning of another object following it in the sequence.

5. The time-setting algorithm

Instantiating (i.e. **setting in time**) a sound-object structure means computing $\alpha(k)$, $\delta(k)$, t1(k), t2(k), TrBeg(k), TrEnd(k) and $\Delta(i)$ for all objects E_k such that k = Seq(nseq,i) with i = 1,…,imax and nseq = 1,…,nmax.

Time-scale ratios $\alpha(k)$ are first calculated and the start/clip dates t1(k) and t2(k) of each object E_k are set to their default values, i.e. with its pivot located exactly on its reference streak. The resulting situation may not be acceptable because of topological constraints on sound-objects. A function called `Locate()` is therefore invoked to scan each sequence of the sound-object structure, modifying the start/clip dates of objects until all constraints are satisfied.

5.1 The `Locate()` function

Each sequence is scanned “from left to right” (increasing dates). The flowchart in Fig.5 shows the detailed operation.

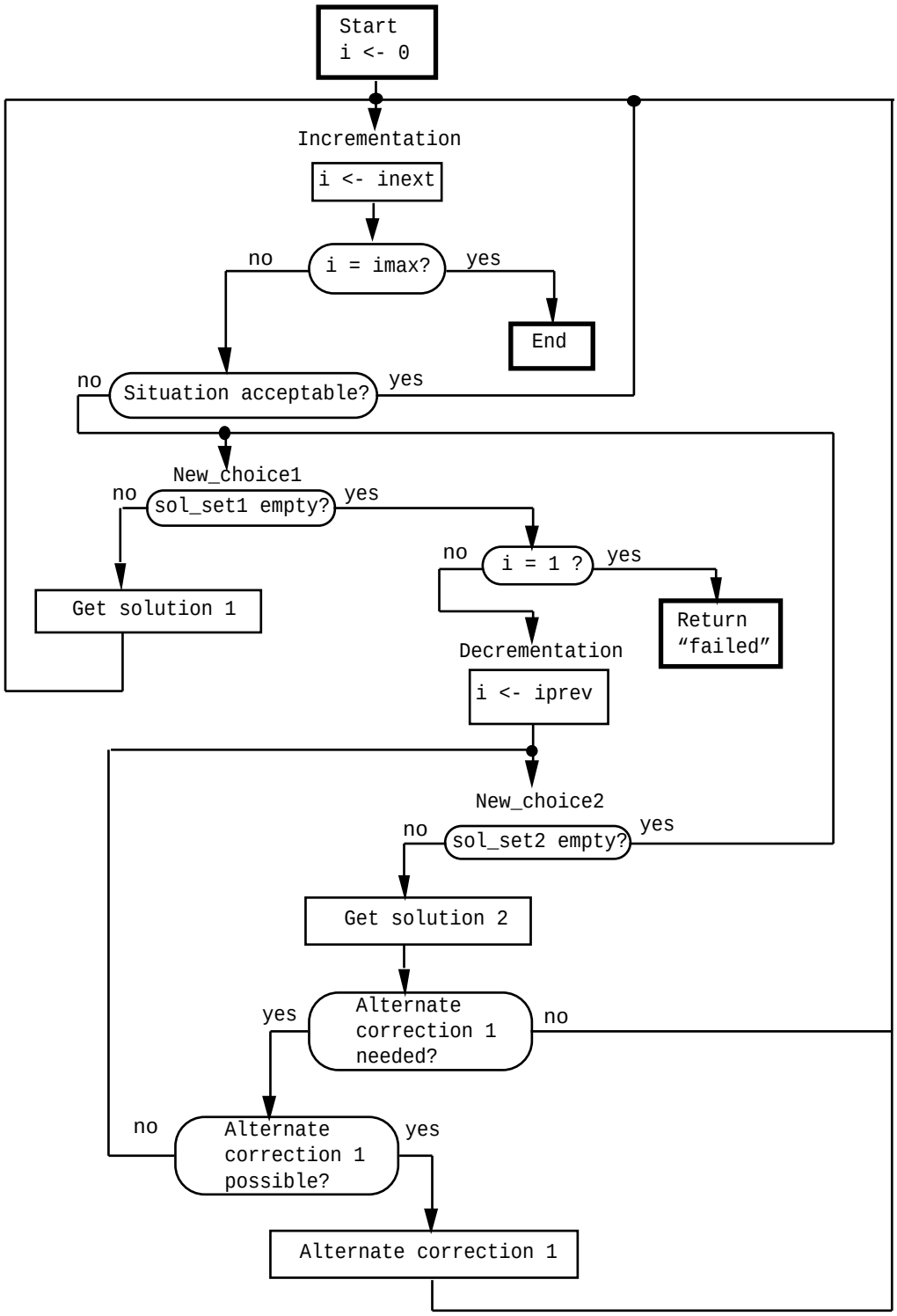


Fig.5 The `Locate()` flowchart

The main test in this function is "Situation acceptable?" (see §5.2). If the situation of E_k does not fulfill constraints then a first solution set `sol_set1` is calculated. This set contains possible corrections of E_k aimed at changing start date t1(k). Clip date t2(k) may also incidently be changed in this process. Each time the program jumps at `New_choice1`, a solution is tried and deleted from the set. This correction is called **correction 1**. If `sol_set1` is empty, the situation of one of the preceding sound-objects must be revised (see `Decrementation`).

Another solution set `sol_set2` is calculated, yielding **correction 2**, i.e. mainly a modification of the clip date t2(k).

To enable backtracking, `sol_set1` and `sol_set2` are stored as arrays indexed on *i*.

5.2 Formalizing topological constraints

While scanning the sequence from left to right, the program stores the clip date of the rightmost object encountered. In Fig.6, E_{kk} is the rightmost object preceding E_k in the sequence. Let t2(kk) be the clip date of E_{kk}. We notate Ts(i) the maximum value of t2(kk) for all ii < i, with kk = Seq(nseq,ii).

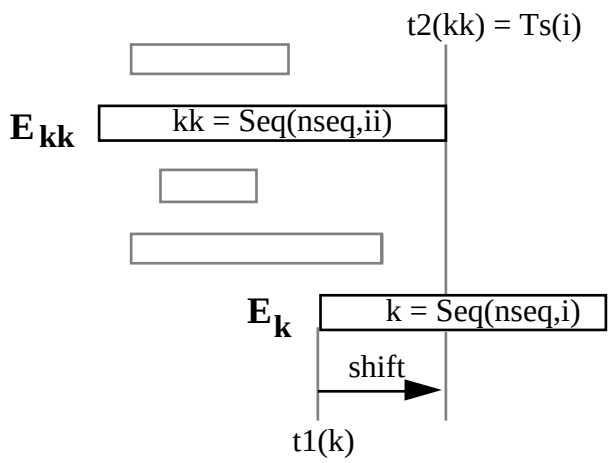


Fig.6 Finding Ts(i) and *shift*

Depending on the value of

$$\text{shift} = Ts(i) - t1(k)$$

and on topological properties of both E_k and E_{kk}, four situations may be envisaged. The only relevant properties of E_{kk}, in this context, are ContEnd(kk) and OverEnd(kk) that have been stored as ContEndPrev(i) and OverEndPrev(i) respectively. The four situations are listed in Fig.7 along with the conditions that make them acceptable. For instance, in situation 3 the rightmost preceding object E_{kk} overlaps E_k. This is acceptable if both the end of E_{kk} and the beginning of E_k can be overlapped.

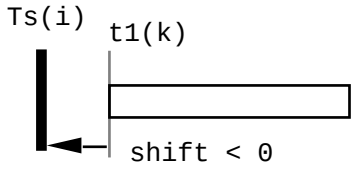
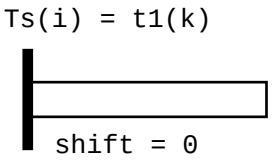
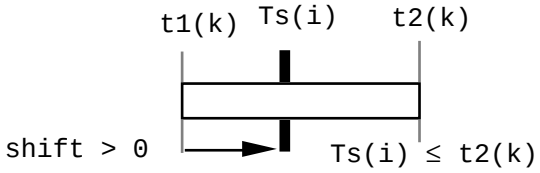
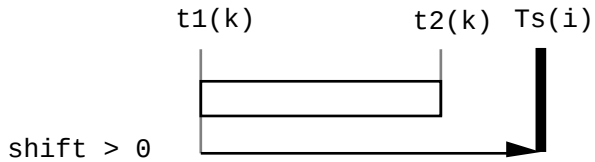
Nr	Configuration	Condition
1		(not ContBeg(j)) and (not ContEndPrev(i))
2		none
3		OverBeg(j) and OverEndPrev(i)
4		OverBeg(j) and OverEnd(j) and (not ContEnd(j)) and OverEndPrev(i)

Fig.7 The four situations

5.3 Corrections 1-2

We call **canonic correction 1** the smallest modification of t1(k) yielding an acceptable solution. It is easy to prove that the canonic correction 1 is:

```
t1[k]  <-  t1[k] + shift
```

There are three possible transformations yielding this canonic correction:

— Local drift: both start and clip dates are incremented.

— Global drift (only if shift > 0): same as above, but the reference streak is also relocated.

— Truncating the beginning (only if shift > 0): only the start date is changed.

Correction 2 is similar to correction 1 except that it is aimed at modifying t2(k). It may truncate the end of the object in the same way correction 1 may truncate its beginning [Bel 1991b:49-50].

5.4 Alternate correction 1

Changing the clip date t2(k) of object E_k modifies topological constraints on objects following E_k in the sequence. These constraints will be evaluated and taken into account while further scanning the sequence. Once a solution has been chosen in `sol_set1`(i), correction 1 is performed and the following objects in the sequence are examined. (See `incrementation` on Fig.5)

On the other hand, whenever correction 2 modifies the start date of E_k it is necessary to check that the constraints on all objects preceding E_k in the sequence are still satisfied. Let E_{kk} and *shift* be defined as in §5.2. The function `Alternate_correction1()` checks the topological situation between E_k and E_{kk} (as per Fig.7). In situation 3 it attempts to truncate the beginning of E_k. If the correction was successful then '0' is returned and incrementation starts again (see flowchart). Otherwise another solution is selected for correction 2. If `sol_set2`(i) is empty, then another solution is tried for correction 1, etc.

It is proved [Bel 1990b:164] that correction 2 always improves the situation and that if there is a solution it will be found in a finite number of steps. Further, if no solution is found it is easy to relax constraints by ignoring part or all topological properties of sound-objects; this yields an approximate solution.

6. Complexity of the time-setting algorithm

The following results have been proved in [Bel 1990b:165-168]:

— In the worst case the time complexity of the `Locate()` function is $O(imax^3)$, where *imax* is the number of sound-objects in the sequence.

— If no global drift is created, the time complexity of the time-setting algorithm is $O(nmax.imax^3)$, where *nmax* is the number of sequences and *imax* the maximum length of a sequence. In the worst case, time complexity is $O(nmax^2.imax^3)$.

Further details about this algorithm, proofs of its correctness and complexity may be found in [Bel 1990b:chapter IX] and [Bel 1991b:39-53].

7. Other features

7.1 Quantization

The phase diagram of a sound-object structure (see §1) contains complete information about precedence relations. It is convenient to represent it as a bidimensional array for computing the dates of all elementary tasks contained in sound-objects. However, in many cases such arrays may have unwieldingly many columns representing time delays smaller than the current time resolution. To save both memory and computation time, a simplification procedure has been implemented. Given a **time quantization**, an acceptable delay below which two events may be seen as simultaneous (i.e. belong to the same column), BP2 is able to simplify the polymetric expression before it creates the phase diagram [Bel 1993a]. However, this simplification process does not change precedence relations in sequences. This is an important feature because any pair of sound-objects might be used to set a process on and off.

7.2 Smooth time

Computing dates of sound-objects in **smooth time** [Boulez 1960:107] makes it possible to combine "plasticity" with accuracy. In striated time, dates of sound-objects were determined by predefined time streaks (e.g. metronom beats) and the pivots of sound-objects. In smooth time, both dates and durations depend on the hierarchy of sound-objects. Objects appearing on the first line of the phase diagram are performed in a strict sequence. Their durations depend on the

durations of corresponding sound-object prototypes. Durations and dates of objects of the second line are calculated to match those of the first line, and so on.

Smooth time is often combined with **time patterns**, i.e. sequences of **time-objects** that do not contain any elementary task. The time-span intervals of these time-objects are structured as a **join semi-lattice** (often a simple tree structure) [Bel 1993a]. For instance, given time-objects t1, t2, t3, t4 defined with

$$t1 = 1/1 \quad t2 = 3/2 \quad t3 = 4/3 \quad t4 = 1/2$$

BP2 may perform the sound-object sequence

do5 re5 mi5 fa5 - la5 si5 do6_ mi6

(in which "-" is a silence and "_" is the prolongation of do6) in the time-setting shown Fig.8.

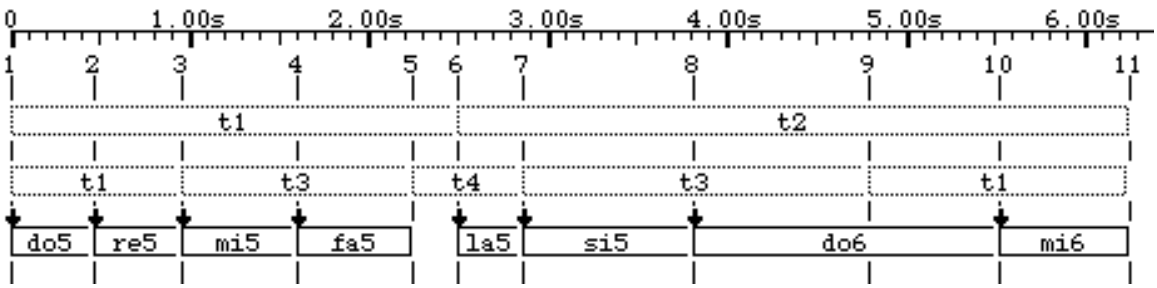


Fig.8 Sound-object structure in smooth time

Musical examples given in [Bel 1990b, 1991a-b] point at the variety of solutions proposed by the time-setting algorithm for the performance of the same musical item in different structures of time. This approach widely contributed to compensate the rigidity of the timing of computer-generated musical pieces.

References

Allen, J.F. 1983
Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26, 11:832-843

Anderson, D.P., and Kuivila, R. 1989
Continuous abstractions for discrete event languages. *Computer Music Journal*, 13, 3:11-23.

Bel, B. 1990a
Time and musical structures. *Interface*, 19, 2-3:107-135.

1990b
Acquisition et Représentation de Connaissances en Musique. PhD dissertation, Faculté des Sciences de St-Jérôme, Université Aix-Marseille III.

1991a
Symbolic and sonic representations of sound-object structures. In *Understanding Music with AI*, eds. M. Balaban, K. Ebcioglu & O. Laske, AAAI Press.

1991b
Two algorithms for the instantiation of structures of musical objects. Internal report GRTC/458. Groupe Représentation et Traitement des Connaissances, CNRS, Marseille.

1993a
BOL PROCESSOR BP2: "QuickStart" and reference manual. Available from author as electronic file.

1993b
Rationalizing musical time: syntactic and constraint-satisfaction approaches. *The Ratio Symposium*, Den Haag (The Netherlands). (Forthcoming)

Boulez, P. 1963
Penser la musique aujourd'hui. Paris (France): Gonthier.

Duthen, J., and Stroppa, M. 1990
Une représentation de structures temporelles par synchronisation de pivots. In *"Le Fait Musical: Sciences, Technologies, Pratiques"*, eds. B. Vecchione and B. Bel, Proceedings of Conference "Music and Information Technology" (MAI 90), Marseille: 471-479.

Jaffe, D. 1985
Ensemble timing in computer music. *Computer Music Journal*, 9, 4:38-48.

Xenakis, I. 1963
Musiques formelles. Paris: La Revue Musicale. Augmented translation: *Formalized music*, 1971. Bloomington: Indiana University Press.