Constraining Temporal Sequences with Allen Relations

Pierre Roy[1], Alexandre Papadopoulos[1,2] and François Pachet[1,2]

[1] Sony CSL, 6 rue Amyot, 75005, Paris, France
[2] Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, France
alexandre.papadopoulos@lip6.fr
pachet@csl.sony.fr
roy@csl.sony.fr

Abstract. The declarative nature of constraint programming makes it well suited to the resolution of sequence generation problems. The generation of temporal sequences, i.e., the organization of timed events, involves the specification of relations between events based on their temporal position and duration. The typical constraint approach is to declare a variable for each event and to state constraints on these variables. However, in a temporal sequence, the *position* of an event is not determined by its *index* as it depends on the duration of the preceding events. This makes it difficult to express temporal properties with standard constraints or techniques from scheduling applications. To overcome this issue, we present ALLEN, a global constraint relating indexes and event positions. ALLEN maintains two set variables: the set of events and the set of indexes at a position defined by an Allen relation between time intervals. These variables enable the definition of constraints representing temporal synchronization properties. We propose an algorithm that achieves hybrid consistency for ALLEN in polynomial time. We apply our approach to two temporal sequence generation tasks: the synchronization of audio tracks and the composition of highly structured polyphony.

1 Introduction

Many difficult combinatorial problems, such as job-shop scheduling, or polyphonic musical composition, consist in arranging sequences of events in time, subject to *horizontal* and *vertical* constraints. Horizontal constraints relate events in the same sequence, but occurring at different positions – note that, in this article, *position* always refers to the *temporal* position – and vertical constraints relate events occurring simultaneously in different sequences. The combination of horizontal and vertical constraints make these problems very difficult to solve: the job-shop scheduling problem is notoriously among the hardest combinatorial problems.

The Constraint Satisfaction Problem approach (CSP) to sequence generation is to define a constrained variable for each item and to state sequence properties as constraints between item variables. Temporal sequences challenge this model, since the position of an event is independent of its index, but depends on the duration of all preceding events. It is therefore difficult, if not impossible, to express temporal properties using constraints on item variables.

A straightforward approach is to use a position-based model instead, in which variables represent events of *atomic duration*. For a given total duration, a fixed number of variables is defined and long objects are made up of several consecutive variables, therefore indexes are temporal too. This requires to discretize the time into a grid of equal-duration slices, small enough so that all events are aligned with the grid. In this model, the number of variables is considerably larger than the number of events in the generated sequences: if durations are expressed as fractions of the longest event, the atomic duration is the quotient of the *greatest common divisor* of the numerators of all fractions by the *least common multiple* (lcm) of the denominators. The growth of the lcm is exponential [13], hence the size of the grid may be exponentially smaller than the event durations. For instance, in a typical musical corpus with note values 1 (whole note), $^1/_2, ^1/_4, \ldots, ^1/_{32}, ^1/_3$ (triplet) and $^1/_5$ (quintuplet), the granularity of the grid is $^1/_{120}$ of a quarter note. The generation of an eight bar melody with a $^4/_4$ time signature, requires $8 \times 4 \times 120 = 3840$ variables. This approach is therefore not applicable, since the resulting CSP is too large.

Several frameworks use constraint propagation to reason about temporal relations from a qualitative [3] or quantitative standpoint [9]. The computational efficiency of these approaches is very limited in the general case, but they offer a precise and powerful representation of relations between times events. We propose to use Allen's algebra, not to make inferences about time in general, but rather as a language to express temporal position. We introduce the ALLEN global constraint, which defines variables corresponding to a given Allen relation. Technically, for a given time interval t and a given Allen relation $\mathcal{R}$, ALLEN maintains two set variables: the set of events and the set of variable indexes satisfying $\mathcal{R}$ for t. Temporal properties of the sequence are represented by constraints defined on these Allen variables. We present a filtering algorithm for ALLEN that runs in polynomial time.

Formally, ALLEN constructs temporal sequences consisting of non-overlapping, contiguous events. Such sequences are essential concepts in musical composition. More complex structures may be elaborated by combining several ALLEN constraints. We illustrate our approach on two musical applications: the synchronization of audio tracks and the generation of highly structured polyphony.

2 Constrained Temporal Sequences

We address the generation of sequences made of contiguous, non-overlapping events. By analogy with music, we call such sequences *monodies*. In a monody $S = (s_1, \ldots, s_n)$, each event s_i has a start time (or onset), denoted by s_i^-, an ending time denoted by s_i^+, and a duration $|s_i| = s_i^+ - s_i^-$. The time interval occupied by event s_i is denoted by $[s_i^-, s_i^+]$. S is a monody, thus, for all $i < n$, $s_i^+ = s_{i+1}^-$. The generation of multiple synchronized sequences is addressed by generating as many monodies and stating vertical constraints.

Here are some typical horizontal constraints on monodies:

- the sequence has an imposed total duration;
- any two consecutive events are different;
- the first and the last events are identical;
- events occurring between positions p and q belong to a specific subset of events;

– the subsequence starting at position p and ending at $p+d$ is the same as that starting at p' and ending at $p'+d$.

The first two constraints hold on events defined by their index while the last three hold on events defined by their position. To handle such constraints, it is therefore necessary to have a double representation of sequences relating the index of events to their position in time.

Permutation and scheduling problems raise similar issues. Smith [17] proposed a model with channeling constraints linking position variables and value variables in permutation problems. In a permutation problem with n variables $X_1, \ldots, X_n$ and n values $a_1, \ldots, a_n$, Smith [17] introduces *dual* variables, $\mathcal{I}_{a_j}$ representing the index of a_j in the sequence, and channeling constraints:

$$\mathcal{I}_{a_j} = i \iff X_i = a_j, \forall i, j \tag{1}$$

Zhou [18] solves job-shop problems with permutation constraints based on a similar system to relate task onsets and task indexes. These approaches are efficient for permutation problems, but they are not directly applicable to the generation of monodies. Monodies are not permutations: the same event may appear multiple times.

3 Model Based on Allen Relations

Generalizing equation (1) to the case where a value may be taken by several variables requires the introduction of a set variable $\mathcal{I}_{a_j}$ subject to the channeling constraint:

$$i \in \mathcal{I}_{a_j} \iff X_i = a_j$$

A further generalization is to consider a set A of values instead of a single value a_j, leading to:

$$i \in \mathcal{I}_A \iff X_i \in A$$

Adopting the *subset bound* representation of set variables [15], we have the following properties for the lower and upper bounds of $\mathcal{I}_A$:

$$lb(\mathcal{I}_A) = \{i \colon D_i \subseteq A\}, ub(\mathcal{I}_A) = \{i \colon D_i \cap A \neq \emptyset\}$$

where D_i denotes the domain of variable X_i.

A constraint involving set variables is *hybrid consistent* (HC) if: it is arc consistent for domain variables and for a set variables S, every support of the constraint is such that the value s of S in the support satisfies $lb(S) \subseteq s \subseteq ub(S)$.

For instance, for variables X_1, X_2, X_3 with $D_1 = \{a\}$, $D_2 = \{a, c\}$, and $D_3 = \{b, c\}$, for $A = \{a, b\}$, we have:

$$lb(I_A) = \{a\}, ub(I_A) = \{a, b\}$$

In the context of monodies, interesting sets are the sets of values satisfying temporal properties, such as relations between time intervals defined byn Allen [3]. Allen defined 13 atomic relations covering all the possible relative positions between the extreme values of intervals: any two time intervals are in exactly one of these relations, see Table 1. For a monody $S = (s_1, \ldots, s_n)$, a relation $\mathcal{R}$ and a time interval t, we define the set $\mathcal{R}_t(S) = \{s_i \colon [s_i^-, s_i^+]\mathcal{R}t\}$.

Relation	Symbol	Inverse	Example
t_1 before t_2	$<$	$>$	t_1 t_2
t_1 equal t_2	=	=	t_1 t_2
t_1 meets t_2	**m**	**mi**	t_1 t_2
t_1 overlaps t_2	**o**	**oi**	t_1 t_2
t_1 during t_2	**d**	**di**	t_1 t_2
t_1 starts t_2	**s**	**si**	t_1 t_2
t_1 finishes t_2	**f**	**fi**	t_1 t_2

Table 1. The 13 atomic relations between time intervals.

4 Temporal Sequence Generation as a CSP

We model the generation of a monody S of events with an imposed total duration D as a CSP with n sequence variables $X_1, \ldots, X_n$ taking values in E, a finite set of events. Each event $e \in E$ has a duration $|e| > 0$. The total duration is represented by linear constraint: $\sum_{i \leq n} |X_i| = D$.

4.1 Sequence Variables

In this approach, the number necessary sequence variables is optimal: $\lfloor D/d \rfloor \leq n \leq \lfloor D/d' \rfloor$ with d and d', the duration of the shortest and longest event. For the previous example, $n = 256$, instead of $n = 3840$ for the position-based model.

The number of variables actually used in a sequence is less than n and is unknown *a priori*. This aspect is taken into account by a simple *padding* strategy: a padding value p with $|p| = 0$ is added to each X_i, with the additional constraint that $X_i = p \Rightarrow X_{i+1} = p$ (the padding values are packed at the end of the sequence).

4.2 The ALLEN Constraint

For a given Allen relation $\mathcal{R}$ and a given time interval t, we define the ALLEN global constraint as follows:

$$\text{ALLEN}_{\mathcal{R},t}([X_1, \ldots, X_n], \mathcal{I}, \mathcal{E}) \Leftrightarrow \begin{cases} & \mathcal{I} = \{i \colon [\![X_i]\!] \mathcal{R} t\} \\ \wedge & \mathcal{E} = \{X_i \colon i \in \mathcal{I}\} \end{cases}$$

where $[\![X_i]\!]$ is the time interval $[|X_1| + \cdots + |X_{i-1}|, |X_1| + \cdots + |X_i|]$ and $\mathcal{I}$ and $\mathcal{E}$ are set variables. In this formulation, the set variable $\mathcal{I}$ contains the indices of the sequence variables $X_1, \ldots, X_n$ that satisfy the relation $\mathcal{R}$ for the time interval t in the sequence. The set variable $\mathcal{E}$ is the set of values of these sequence variables. Example 1 shows the values of these variables for a tiny problem. Note that $\mathcal{R}$ may be one of the 13 atomic relations or any logical combination of them, such as $\mathbf{s} \vee \mathbf{o}$, designating the events that start *or* overlap with a certain time interval.

Example 1 *Consider events a and b, with $|a| = 1, |b| = 2$, and variables X_1, X_2, X_3, with $D_1 = D_2 = \{a, b\}$, $D_3 = \{b\}$, with $D_i = dom(X_i)$. The extension of constraint*

$$\text{ALLEN}_{\boldsymbol{d}[2,5]}([X_1, X_2, X_3], \mathcal{I}, \mathcal{E})$$

is the following:

X_1	X_2	X_3	$\mathcal{I}$	$\mathcal{E}$
a	a	$\underline{b}$	$\{3\}$	$\{b\}$
a	b	$\underline{b}$	$\{3\}$	$\{b\}$
b	$\underline{a}$	$\underline{b}$	$\{2, 3\}$	$\{a, b\}$
b	$\underline{b}$	b	$\{2\}$	$\{b\}$

We introduce the CONTIGUOUS global constraint, which will be useful to explain the filtering algorithm for ALLEN. Basically, CONTIGUOUS maps any event e to a couple $\langle s, e\rangle$, where s is the onset of e in the sequence, and is defined by:

$$\text{CONTIGUOUS}([X_1, \ldots, X_n], [Y_1, \ldots, Y_n]) \Leftrightarrow$$

$$s(Y_{i+1}) = s(Y_i) + |e(Y_i)| \wedge X_i = e(Y_i)$$

where $e(\langle s, e\rangle) = e$ and $s(\langle s, e\rangle) = s$. To filter the CONTIGUOUS constraint, we use a specific graph structure, the *onset graph*, which is defined as follows:

Definition 1 *The* onset graph *of an* CONTIGUOUS *constraint* CONTIGUOUS$([X_1, \ldots, X_n], [Y_1, \ldots, Y_n])$ *is the layered graph $G = (V, E)$, with $V = \{0\} \cup V_1 \cup \cdots \cup V_n$ and $V_i = \{\sum_{j=1}^{i} |x_j| \colon x_1, \ldots, x_i \in D(X_1) \times \cdots \times D(X_i)\}$. The edges are defined as follows:*

- $\forall x_1 \in D(X_1)$, *there is an edge, labeled x_1, between 0 and $|x_1|$;*
- $\forall i < n$, $\forall v_i \in V_i$, *and* $\forall x_{i+1} \in D(X_{i+1})$, *there is an edge, labeled x_{i+1}, between $v_i \in V_i$ and $v_i + |x_{i+1}| \in V_{i+1}$. We will use E_i to denote the set of edges between a node $v_i \in V_i$ and a node $v_{i+1} \in V_{i+1}$.*

Note that an edge $(v_i, v_{i+1}) \in E_i$, labeled x, corresponds to value $\langle v_i, x\rangle$ in the domain of Y_i. Each $e = (v_i, v_{i+1})$ in the onset graph corresponds to a unique interval $t(e) = [v_i, v_{i+1}]$.

Figure 1 shows the onset graph of Example 1. In the figure, red labels correspond to edges e such that $t(e)\mathbf{d}[2, 5]$. The domain of Y_i is obtained by reading all the edges

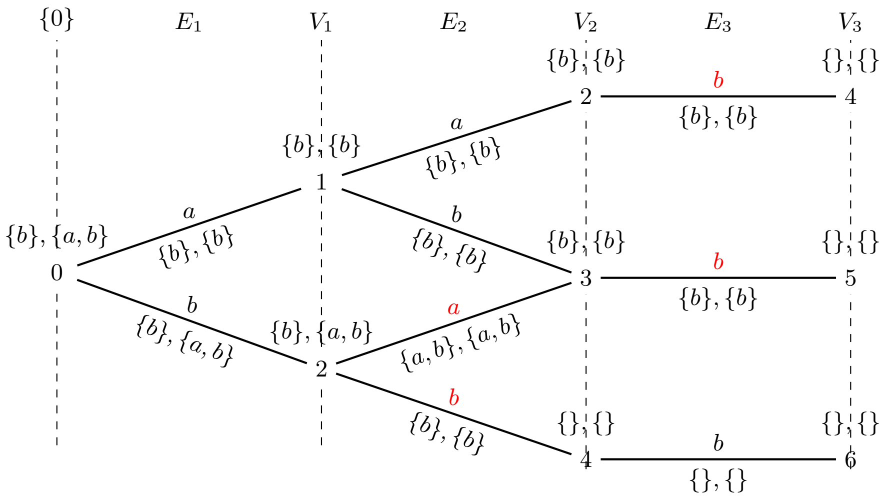

Fig. 1. Propagation of the bounds of $\mathcal{E}$ in the onset graph of an CONTIGUOUS constraint.

in E_i, the onset of an event is the value of the left node of the edge. The domain of X_i is the labels of the edges in E_i.

Note that the onset graph is equivalent to the cost-sets defined by the METER constraint [16]. An edge $e = (v_i, v_{i+1}) \in E_i$, labeled x, of the onset graph corresponds to value v_{i+1} in the cost-set associated to variable X_i and value x. However, our graph representation is more compact (in Figure 1 node 3 in V_2 corresponds to two values in the cost-sets) and provides the basis for implementing Algorithm 1.

It is interesting to note that the ALLEN constraint is equivalent to the conjunction of three global constraints: a CONTIGUOUS, a RANGE, and a ROOTS constraints, defined by

$$\text{ALLEN}_{\mathcal{R},t}([X_1, \ldots, X_n], \mathcal{I}, \mathcal{E}) \Leftrightarrow \begin{cases} & \text{CONTIGUOUS}([X_1, \ldots, X_n], [Y_1, \ldots, Y_n]) \\ \wedge & \text{ROOTS}([Y_1, \ldots, Y_n], \mathcal{I}, \mathcal{R}_t) \\ \wedge & \text{RANGE}([X_1, \ldots, X_n], \mathcal{I}, \mathcal{E}) \end{cases}$$

ROOTS [6] states that S is the set of indexes of the variables taking a value in T:

$$\text{ROOTS}([X_1, \ldots, X_n], S, T) \Leftrightarrow S = \{i \colon X_i \in T\}$$

and RANGE [5] holds iff a set variable T is the set of values taken by variables whose indexes are in a second set variable S:

$$\text{RANGE}([X_1, \ldots, X_n], S, T) \Leftrightarrow T = \{X_i | i \in S\}$$

Achieving HC for ROOTS is NP-hard. However, when the second set variable T is ground, HC is tractable [7]. In this case, ROOTS is filtered by propagating binary constraints (using our variable names):

$$i \in \mathcal{I} \rightarrow Y_i \in \mathcal{R}_t \text{ and } Y_i \in \mathcal{R}_t \rightarrow i \in \mathcal{I} \tag{2}$$

By reformulating this with Boolean variables $\mathcal{I}_1, \ldots, \mathcal{I}_n$ such that $i \in \mathcal{I}$ iff $\mathcal{I}_i = true$, the hyper-graph formed by CONTIGUOUS and ROOTS is Berge-acyclic as shown in Figure 2. The HC of the conjunction of these two constraints is achieved by filtering the two constraints separately.

Although HC is polynomial for RANGE, it does not achieve HC for the whole decomposition. A counter example is given by Example 1, for which filtering the CONTIGUOUS constraint yields the following domains for the Y_i:

- $D(Y_1) = \{\langle 0, a\rangle, \langle 0, b\rangle\}$;
- $D(Y_2) = \{\langle 1, a\rangle, \langle 1, b\rangle, \langle 2, a\rangle, \langle 2, b\rangle\}$;
- $D(Y_3) = \{\langle 2, b\rangle, \langle 3, b\rangle, \langle 4, b\rangle\}$.

After filtering the ROOTS constraint, we have $lb(\mathcal{I}) = \emptyset$ and $ub(\mathcal{I}) = \{2, 3\}$ (no value in the domain of Y_1 is "during $[2, 5]$"). Finally, filtering RANGE, leads to the following bounds for $\mathcal{E}$: $lb(\mathcal{E}) = \emptyset$ and $ub(\mathcal{E}) = \{a, b\}$. The lower bound is not tight enough, as HC for ALLEN would give $lb(\mathcal{E}) = \{b\}$, as shown in Table 1: value b is underlined in every support of the constraint.

4.3 Filtering Algorithm for ALLEN

In this section, we present a filtering algorithm for ALLEN that computes the bounds for $\mathcal{E}$. We show that the resulting lower and upper bounds are tight. This algorithm is a breadth-first traversal of the onset graph, starting with the nodes in V_n and ending with the source node 0. The algorithm computes, for each node v, two sets $lb(v)$ and $ub(v)$. Let us prove that $lb(0) = lb(\mathcal{E})$ and $ub(0) = ub(\mathcal{E})$.

Consider the induction hypothesis H_i: $\forall v \in V_i$, $x \in lb(v)$ iff any path from v to a node in V_n contains an edge e such that $label(e) = x$ and $t(e)\mathcal{R}t$. Let $v \in V_{i-1}$ with $x \in lb(v)$, and let $(e_i, \ldots, e_n)$ be an edge-path from v to a node in V_n. By Algorithm 1, $x \in lb(v)$ can be because of e_i itself, that is, $label(e_i) = x$ and $t(e_i)\mathcal{R}t$, or because the ending node w of e_i is such that $x \in lb(w)$. In both cases, the induction property H_{i-1} holds, therefore H_0 holds: $lb(0) = lb(\mathcal{E})$. The very same argument, in which "any path" is replace by "there exists a path", shows that $ub(0) = ub(\mathcal{E})$.

Here are the propagators applied when the bounds of $\mathcal{E}$ are modified:

if x **is removed from** $ub(\mathcal{E})$: remove all edges $e \in E$ such that $label(e) = x$ and $t(e)\mathcal{R}t$. Propagate the removal of e to the rest of the graph.

if x **is added to** $lb(\mathcal{E})$: for each edge $e \in E$ such that $label(e) = x$ and $t(e)\mathcal{R}t$, mark the children edges of e up to V_n and the parent edges of e up to 0. Remove unmarked edges from the onset graph.

After both propagators, Algorithm 1 is applied to the onset graph. The domains of $Y_1, \ldots, Y_n$ are updated by a simple visit of the onset graph. The domains of $X_1, \ldots, X_n$ are updated by the CONTIGUOUS constraint. The domain of $\mathcal{I}$ is updated by a separate propagation of the ROOTS constraint.

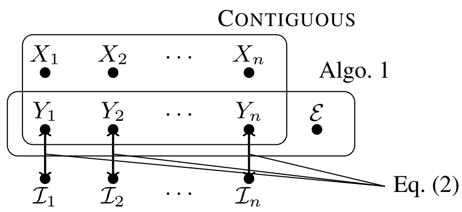

Fig. 2. Overview of the decomposition of the ALLEN constraint.

The worst-case complexity of Algorithm 1 is $O(|V|+|E|)$; $|V| = \sum_i |V_i|$ and $|V_i| \leq |\{id, \ldots, id'\}| = i(d'-d)$, where d' and d are the durations of the longest and shortest events, hence, $|V| \leq n^2(d'-d)$; $|E| = \sum_i |E_i|$ and $|E_i| \leq N|V_i|$, where N is the number of events in the domains. Finally, $|E| \leq n^2 N(d'-d)$. Overall, the HC for ALLEN is obtained in $O(n^2 N(d'-d))$. This can be significantly improved by taking into account the total duration T when it is imposed as in this case $|V_i| = |\{id, \ldots, \min(T, iD)\}|$.

Algorithm 1: Computing $lb(\mathcal{E})$ and $ub(\mathcal{E})$

// initializes nodes in V_n
foreach *node* $v \in V_n$ **do**
  $lb(v) \leftarrow \emptyset$; $ub(v) \leftarrow \emptyset$;
// backward BFS
for $i = n-1, \ldots, 0$ **do**
  // edges
  foreach *edge* $e = (v_i, v_{i+1}) \in E_i$ **do**
    $lb(e) \leftarrow lb(v_{i+1})$;
    $ub(e) \leftarrow ub(v_{i+1})$;
    if $t(e)\mathcal{R}t$ **then**
      $lb(e) \leftarrow lb(e) \cup \{label(e)\}$;
      $ub(e) \leftarrow ub(e) \cup \{label(e)\}$;
  // nodes
  foreach *node* $v_i \in V_i$ **do**
    $lb(v_i) \leftarrow \bigcap_{e \in E_i} lb(e)$;
    $ub(v_i) \leftarrow \bigcup_{e \in E_i} ub(e)$;

4.4 Domain Allen Variables

The Allen relation considered in an ALLEN constraint influences the nature of variables $\mathcal{I}$ and $\mathcal{E}$. The Allen variables defined by relations **=**, **m**, **o**, **s**, and **f** have at most one element. For instance, at most one event in a monody *meets* a given time interval. It is even possible to construct relations holding for exactly one event in a sequence: for instance $\mathbf{s} \vee \mathbf{o} \vee \mathbf{di}$ or $\mathbf{f} \vee \mathbf{oi} \vee \mathbf{di}$. The Allen variables defined by such relations are standard domain variables. In this case, the variables must take a value or the resolution fails, which may be a useful feature. Another consequence is that standard constraints may be defined straightaway on these variables.

4.5 Discussion

An open question is whether our approach could be extended to classical scheduling problems. There is some similarity between CONTIGUOUS and CUMULATIVE [2]: CONTIGUOUS may be seen as a special case of CUMULATIVE, where all task resources and the height limit are unitary. Even in this case, some differences remain (e.g., cumulative allows slacks). A possible technical formulation is: in the decomposition of ALLEN into CONTIGUOUS, RANGE, and ROOTS, what level of consistency can be achieved after replacing CONTIGUOUS by CUMULATIVE?

5 Audio Multitrack Synchronization

In illustrative music, such as video game or film music, timed events in a scene create synchronization points calling for a specific musical treatment (loud drum hit for the gunshot, fast repeated string pizzicato notes for the car chase, etc.) to enhance the emotional impact or fit the atmosphere. Such properties may be enforced by unary constraints on the corresponding Allen variables.

In this section, we describe a system in which Allen variables are used to synchronize audio tracks and to enforce external synchronization constraints. The system relies on a database of fixed-tempo multi-track recordings [4]. We segment each track into chunks using a standard onset detection procedure. Chunks are clustered according to harmonic similarity (for pitched instruments, such as the bass) and timbre similarity for drums. The timbre similarity, using Mel Frequency Cepstral Coefficients (MFCC) with 13 coefficients; the harmonic similarity is computed using the Harmonic Pitch-Class Profile (HPCP) feature with 36 divisions of the octave [11]. For each instrument, we organize chunks into ten clusters $C_0^b, \ldots, C_9^b$ for the bass and $C_0^d, \ldots, C_9^d$ for drums.

We call *multi-chunk* a group of chunks that are played simultaneously and the clusters of a multi-chunk (one cluster per chunk) is called the *cluster signature*.

We write $c \in C_i^d$ to indicate that drum chunk c belongs to cluster C_i^d and $c_1 \rightarrow c_2$ if chunk c_2 follows c_1, either in the corpus or in the generated track. We generate new multitracks with the following horizontal constraints:

- the total duration is imposed, e.g., 16 bars;
- the first last chunks of the bass track are user-defined;
- a chunk transition $c_1 \rightarrow c_2$, with $c_1 \in C_1$ and $c_2 \in C_2$ is allowed if, in the training tracks, there is a transition $c_1' \rightarrow c_2'$ such that $c_1' \in C_1$ and $c_2' \in C_2$.

The only vertical constraint enforces the vertical patterns of the training corpus, which would be lost otherwise, by synchronizing the tracks at every bar onset. The constraint imposes that at the onset of a bar in the generated multitrack, the cluster

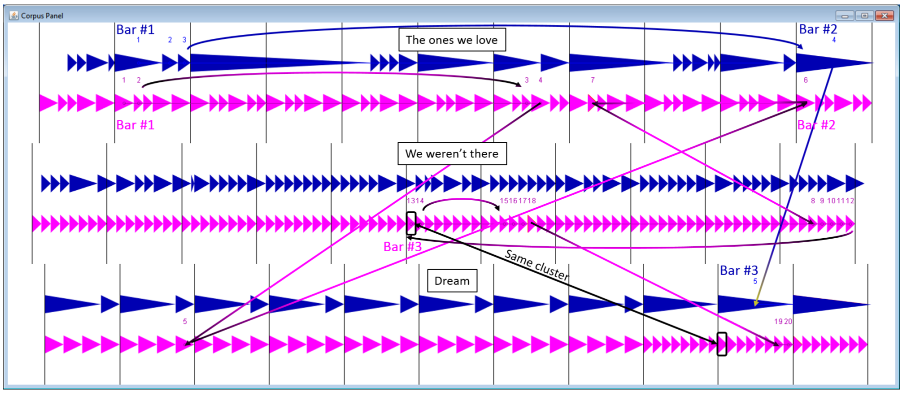

Fig. 3. A screen capture of an execution of the system on three multi-tracks of the corpus. Each blue (resp. pink) triangle represents a bass (resp. drum) chunk. Vertical lines indicate bars. The generated multi-track is not shown: it consists in playing the chunks in the order indicated by numbers (arrows indicate jumps in each track).

signature of the multi-chunk formed by the bass and drum chunks is the same as the cluster signature of an existing multi-chunk in the training multitracks.

Figure 3 shows a screen capture of very short execution of the system on three multi-tracks of the database: *The ones we love*, *We weren't there*, and *Dreams*. The figure shows the first eleven bars of these three multi-tracks. The generated multi-track is not explicitly shown, but it can be reconstructed by following, for each instrument, the numbers above chunks (in the order). Jumps within a track are indicated by colored arrows. Note that bars #1 and #2 start with multi-chunks of the original multi-tracks. On the contrary, bar #3 plays a bass chunk from the third song (number 5 in blue) and a drum chunk from the second song (number 13 in pink). This is allowed as chunk 13 is in the same cluster as the chunk played with the bass chunk in the original song (the two chunks are in black rounded corner rectangles in the figure).

We state this problem as the generation of two sequences of chunk variables: $(B_1, \ldots, B_n)$, for the bass, and $(D_1, \ldots, D_n)$, for drums, subject to:

- linear constraints $\sum_i |B_i| = \sum_i |D_i| = 16 \times 4 = 64$;
- B_1 and B_n are assigned to user-defined chunks;
- binary constraints on chunk cluster transitions.

To represent the vertical synchronization constraints, we state two ALLEN constraint for each bar i:

$$\text{ALLEN}_{\mathbf{s}[4i,64]}([B_1, \ldots, B_n], \mathcal{I}_{b,i}, \mathcal{E}_{b,i}), \text{ for the bass,}$$

$$\text{ALLEN}_{\mathbf{s}[4i,64]}([D_1, \ldots, D_n], \mathcal{I}_{d,i}, \mathcal{E}_{d,i}), \text{ for drums.}$$

Note that for Allen relation **s**, $\mathcal{E}$ and $\mathcal{I}$ are domain variables (see Section 4.4). A binary constraint imposes that the cluster signature of the multi-chunk $(\mathcal{E}_{b,i}, \mathcal{E}_{d,i})$ is the same as that of an existing multi-chunk in the database.

6 Synchronization of Several Sequences

In polyphonic music, or counterpoint, two or more melodic lines are played together. Each part has its own rhythm and contour, but they harmonize with one another. The generation of counterpoint is a well-studied task in algorithmic composition. It was addressed in many ways, using expert systems [10], grammars [1], evolutionary methods [14], or machine-learning [8]. Some forms of counterpoint, such as canons or fugues, are based on strict structures involving (possibly transformed) imitation of one part by the others with some delay. Bach's famous "Cancrizan Canon" (see Figure 4) of the *Musical Offering* is an enigmatic composition: although it is entitled "Canon a 2", it consists of a *single* part. The "trick" is that the melody is to be played together with a reversed version of itself. A combinatorial approach is well-suited to the generation of such highly-structured pieces. In this section, we show how we can generate cancrizan canons using a combination of ALLEN and other, simpler constraints.

Fig. 4. The manuscript of Bach's Canon a 2 of the Musical Offering BWV 1079.

We address the generation of a 10-bar cancrizan canon, subject to:

scale: all bars use notes from the scale of C major;
stylistic imitation: all melodic transitions (ordered pairs of adjacent notes) are present in a a reference corpus;
harmonic intervals: some harmonic intervals are not allowed at the onset of each beat, either to avoid dissonances or unexpressive intervals: unisons, 2^{nd}, tritones, 7^{th}, and 8^{ves} are not admitted;
harmony: the notes played at the onset of each bar are the tonic or the third of the imposed chords: C major (bars 1 and 2), G (3, 4), F (5, 6), G (7, 8), and C to the end.

We model cancrizan canon generation by defining note variables: $P_1, \ldots, P_n$ for Part 1 and $Q_1, \ldots, Q_n$ for Part 2. The domain of each P_i and Q_i is the set of notes whose pitch belongs to the scale of C major and that are present in the reference corpus. The reference corpus consists of the 42 songs (in $^4/_4$) from the Michel Legrand Songbook [12]. The maximum number of variables in each part is $^{10\times4}/_d$, where d is the duration, in beats, of the shortest note in the corpus: $d = 1/8$ for thirty-second notes, thus $n = 320$. In practice, we declare only $n = 60$ variables, as we do not want melodies with only very short notes.

Fig. 5. A 10-bar cancrizan canon in the melodic style of Michel Legrand.

The total duration of 10 bars, that is, 40 beats, is imposed by linear constraints:

$$\sum_{i=1}^{n} |P_i| = 40, \sum_{i=1}^{n} |Q_i| = 40$$

We state binary constraints that each melodic note transition from P_i to P_{i+1} (and from Q_i to Q_{i+1}) exists in the reference corpus.

We state an ALLEN constraint for each beat and each part:

$$\text{ALLEN}_{\mathbf{s}\vee\mathbf{o}\vee\mathbf{di}\ b_i}([P_1, \ldots, P_n], \mathcal{I}^P_{b_i}, \mathcal{E}^P_{b_i}), \forall i = 1, \ldots, 40$$

$$\text{ALLEN}_{\mathbf{s}\vee\mathbf{o}\vee\mathbf{di}\ b_i}([Q_1, \ldots, Q_n], \mathcal{I}^Q_{b_i}, \mathcal{E}^Q_{b_i}), \forall i = 1, \ldots, 40$$

where b_i denotes the time interval corresponding to the i-th beat: $[i, i+1]$. We use the overlaps relation **s** $\vee$ **o** $\vee$ **di**, which defines the unique note playing at the onset of beat i (see Section 4.4). The harmonic interval constraint is stated by binary constraints:

$$\text{INTERVAL}(\mathcal{E}^P_{b_i}, \mathcal{E}^Q_{b_i}) \notin \{unison, 2^{nd}, 7^{th}, 8^{ve}\}$$

where INTERVAL yields the interval between two notes.

For each melody, the note playing at the onset of each bar is the tonic or third of a given chord:

$$\text{PITCH}(\mathcal{E}^P_{\mathbf{s}\vee\mathbf{o}\vee\mathbf{di}\ b1}) \in \{C, E\}, \text{PITCH}(\mathcal{E}^P_{\mathbf{s}\vee\mathbf{o}\vee\mathbf{di}\ b5}) \in \{C, E\}$$

$$\text{PITCH}(\mathcal{E}^P_{\mathbf{s}\vee\mathbf{o}\vee\mathbf{di}\ b9}) \in \{G, B\}, \text{ etc.}$$

where constraint PITCH maps a note onto its pitch-class.

We define the REVERSE constraint, which creates the reversed sequence of an Allen index variable $\mathcal{I}_{\mathcal{R}\,t} = \{i_1, \ldots, i_k\}$, where the indexes are sorted, i.e., $i_j < i_{j+1}$:

$$\text{REVERSED}([X_1, \ldots, X_n], [Y_1, \ldots, Y_n], \mathcal{I}_{\mathcal{R}\,t}) \Leftrightarrow$$

$$Y_1 = X_{i_k}, \ldots, Y_k = X_{i_1} \text{ and } Y_j = p, \forall j > k$$

where p is the padding value. We do not give here the propagators for REVERSED. Finally, the "cancrizan" feature is simply represented by:

$$\text{REVERSE}([P_1, \ldots, P_n], [Q_1, \ldots, Q_n], \mathcal{I}^P_{\mathbf{d}[1,40]})$$

Figure 5 shows a cancrizan canon.

7 Conclusion

We have presented the ALLEN global constraint. ALLEN maintains set variables representing events in a temporal sequence in two ways: one variable is the set of events occurring at a given position, defined by an Allen relation with a reference time interval; the other variable is the set of indexes of these events. In practice, ALLEN offers the possibility to control the generation of temporal sequences by constraining events defined by their index *and* by their position.

We have proposed a propagation algorithm for ALLEN, based on a decomposition using other global constraints: CONTIGUOUS, which we introduced, ensures that a sequence of events are contiguous with one another, ROOTS, which computes variables in set that map onto a subset of predefined values, and RANGE, which computes the set of values taken by a set of variables. Our filtering algorithm achieves HC for ALLEN in polynomial time.

ALLEN makes it possible to model and solve new types of problems involving structural constrains on patterns, represented by subsequences. We illustrated the expressive power of ALLEN on the task of composing highly structured compositions, including virtuoso canons in the style of J.-S. Bach. Such tasks could hardly be addressed using standard global constraints. Another illustration shows that ALLEN may be used to synchronize several musical parts with one another and with external timed events.

References

1. Acevedo, A.G.: Fugue composition with counterpoint melody generation using genetic algorithms. In: Computer Music Modeling and Retrieval, pp. 96–106. Springer (2005)
2. Aggoun, A., Beldiceanu, N.: Extending chip in order to solve complex scheduling and placement problems. Mathematical and Computer Modelling 17(7), 57–73 (1993)
3. Allen, J.F.: Maintaining Knowledge about Temporal Intervals. Commun. ACM 26(11), 832–843 (1983)
4. BASS-dB: BASS-dB: the Blind Audio Source Separation evaluation database (2015), http://bass-db.gforge.inria.fr/BASS-dB/
5. Bessière, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: The range constraint: Algorithms and implementation. In: Beck, J.C., Smith, B.M. (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Third International Conference, CPAIOR 2006, Cork, Ireland, May 31 - June 2, 2006, Proceedings. Lecture Notes in Computer Science, vol. 3990, pp. 59–73. Springer (2006), http://dx.doi.org/10.1007/11757375_7
6. Bessière, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: The ROOTS constraint. In: Benhamou, F. (ed.) Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4204, pp. 75–90. Springer (2006), http://dx.doi.org/10.1007/11889205_8
7. Bessiere, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: Range and roots: Two common patterns for specifying and propagating counting and occurrence constraints. Artif. Intell. 173(11), 1054–1078 (2009), http://dx.doi.org/10.1016/j.artint.2009.03.001

8. Boulanger-Lewandowski, N., Bengio, Y., Vincent, P.: Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. pp. 1159–1166 (2012), `http://icml.cc/discuss/2012/590.html`, bibtex: Boulanger-Lewandowski2012
9. Dechter, R., Meiri, I., Pearl, J.: Temporal Constraint Networks. Artif. Intell. 49(1-3), 61–95 (1991)
10. Ebcioglu, K.: An expert system for harmonizing chorales in the style of j. s. bach. J. Log. Program. 8(1), 145–185 (1990), `http://dx.doi.org/10.1016/0743-1066(90)90055-A`
11. Gómez, E.: Tonal Description of Music Audio Signals. Ph.D. thesis, Universitat Pompeu Fabra (2006), `files/publications/emilia-PhD-2006.pdf`
12. Legrand, M.: The Michel Legrand Songbook. Warner Bros. Publications (1997)
13. Nair, M.: On chebyshev-type inequalities for primes. AMM 89, 126–129 (1982)
14. Phon-Amnuaisuk, S., Wiggins, G.: The four-part harmonisation problem: a comparison between genetic algorithms and a rule-based system. In: Proceedings of the AISB99 Symposium on Musical Creativity. pp. 28–34 (1999)
15. Puget, J.F.: PECOS: a High Level Constraint Programming Language. In: Proceedings of Singapore International Conference on Intelligent Systems. pp. 137–142. SPICIS'92 (1992)
16. Roy, P., Pachet, F.: Enforcing Meter in Finite-Length Markov Sequences. In: desJardins, M., Littman, M.L. (eds.) AAAI. AAAI Press (2013)
17. Smith, B.M.: Modelling a permutation problem. In: Proceedings of ECAI2000 Workshop on Modelling and Solving Problems with Constraints. pp. 615–619. Springer (2000)
18. Zhou, J.: A Permutation-Based Approach for Solving the Job-Shop Problem. Constraints 2(2), 185–213 (1997)