A Constraint Based Approach to Musical Textures and Instrumental Writing

Mikael Laurson and Mika Kuuskankare
Center for Music and Technology, Sibelius Academy,
P.O.Box 86, 00251 Helsinki, Finland
email: laurson@siba.fi, mkuuskan@siba.fi

Abstract

PatchWork (PW), (Laurson, 1996a), is a visual programming language which has a direct correspondence to its base languages Common Lisp and CLOS. The user operates with graphical entities (boxes) and defines relations between them with the help of connections. PW has been extended by several user libraries that are designed to solve specific musical problems.

Our focus in this paper is a user library called PWConstraints (Laurson 1996a and 1996b). In PWConstraints the user does not formulate stepwise algorithms but describes the desired result with a set of rules. This approach has been used to solve concrete large-scale musical problems. During the last years PWConstraints has been extended to cover other areas such as generation of rhythms, textures and expressions. This paper offers also some new ideas how instrumental writing could be incorporated more deeply in the computer aided composition process.

1 Background

PWConstraints is a rule-based programming language where the user writes rules to describe the end result from many different points of view. A search-space is defined as a set of search-variables. Each search-variable has a domain containing a list of values. In a rule a pattern-matching language is used to extract relevant information from a potential solution. This information is given to a Lisp test function that either accepts or rejects the current choice made by the search-engine. The rules are compiled to efficient Lisp functions.

The pattern-matching part of a rule uses a fairly typical pattern-matching syntax. It can contain variables, anonymous-variables, wild card and index-variables. A variable extracts single values from a partial solution. By contrast, an anonymous-variable is never bound to a value, i.e. it only acts as a "place-holder" in the pattern. The wild card matches any continuous part of a partial solution. Finally, an index-variable extracts values from an absolute position. As an example a rule disallowing two adjacent equal values in a result is written as follows (this rule uses a wild card and two variables):

```
(* ?1 ?2 (?if (/= ?1 ?2)) "no equal adjacent values")
```

The user can also define preferences by heuristic rules. These are similar to the ordinary PWConstraints rules except the Lisp-code part of a heuristic rule returns a numerical value instead of a truth value (heuristic rules never reject candidates). This scheme allows the search to sort candidates that are accepted by the ordinary rules. The search tends to favour candidates with high numerical values. For instance a heuristic rule that prefers large intervals can be written as follows:

```
(* ?1 ?2 (?if (abs (- ?2 ?1))) "prefer large intervals")
```

Two classical constraint satisfaction tools (forward checking and back jumping) allow the user to optimise a difficult problem.

Besides these basic tools PWConstraints contains several extensions. The most important and complex one is used to solve polyphonic search problems. This is accomplished with a function called Score-PMC. Like in a traditional polyphonic score, the user operates with several layers (parts, voices) of events (notes). The rhythmic structure of a search problem is prepared in advance in a standard PW rhythm-editor. This input score (which can be arbitrary complex) is given as an argument to the search engine. The search, in turn, aims at filling the input score with pitch information according to the given rules. In a sense Score-PMC can be seen as a multi-layered search problem where each melodic line represents one queue structure similar to the one used by a simple search problem. For more details see Laurson (1996a) and Rueda et al. (1998).

2 Texture-PMC

Lately the multi-layered search idea has been extended to cover other parameters than pitch. A new function called Texture-PMC uses internally the Score-PMC function and is capable of producing complex rhythms and textures. The input-score consists of a dense train of pulses. Each input pulse can be interpreted either as an attack, tie, rest or some other texture type.

A Texture-PMC problem is solved in three steps (Figure 1): (1) the user enters the pulses (upper staff); (2) Texture-PMC is run - this results in an intermediate result where each pulse can take one of the following pitch-values: C4, C#4 or D4 - (middle staff); (3) the pitch information shown in the middle staff is interpreted as follows: all C4s become attacks, all C#4s become rests and finally all D4s are considered to be ties. The lowest staff of Figure 1 gives the resulting rhythm. This scheme can be extended easily to cover other textural types by adding more pitch-values to the input pulses. The intermediate result (step 2) is typically not shown to the user. All mappings are done automatically by the system. This means that the user sees only the first staff (the pulse input) and the result (the lowest staff). Furthermore, the user does not have to worry in Texture-PMC rules how the mappings work internally. The rhythmical and textural properties are always referred to symbolically (such as ‘attack’, ‘rest’, ‘tie’).

Figure 1. Mapping a pulse train into attacks, ties and rests.

The example in Figure 1 was realised using five rules. One of them states that all downbeat pulses (except in the first beat of a measure) should be rests:

```
(* ?1   (and (/= (beatnum ?1) 1) (downbeat? ?1))
  (?if (= (r ?1) rest)) "downbeat rests")
```

A complementary rule states that all downbeat pulses in the first beat of a measure should be attacks:

```
(* ?1   (and (= (beatnum ?1) 1) (downbeat? ?1))
  (?if  (= (r ?1) attack))  "downbeat attacks")
```

The next example (Figure 2) demonstrates how Texture-PMC works in a polyphonic context:

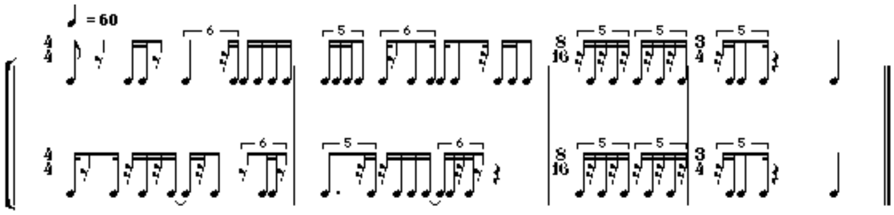

Figure 2. A polyphonic rhythmic result.

The two first measures of the second part in Figure 2 use similar rules than our previous example. The two first measures of part one, in turn, have always attacks at downbeats. The third measure uses a “hocket” rule that forces the parts to alternate attacks and rests. Finally, in the fourth measure the parts are synchronised.

Figure 3 shows an eight-part texture. Here the rules control the number of simultaneous notes in each measure: in measure 1 each attack consists of a single note, in measure 2 each attack has two notes, in measure 3 three notes and in measure 4 the number of notes is four. After this the process is reversed so that the example ends with a measure consisting of one-note attacks:

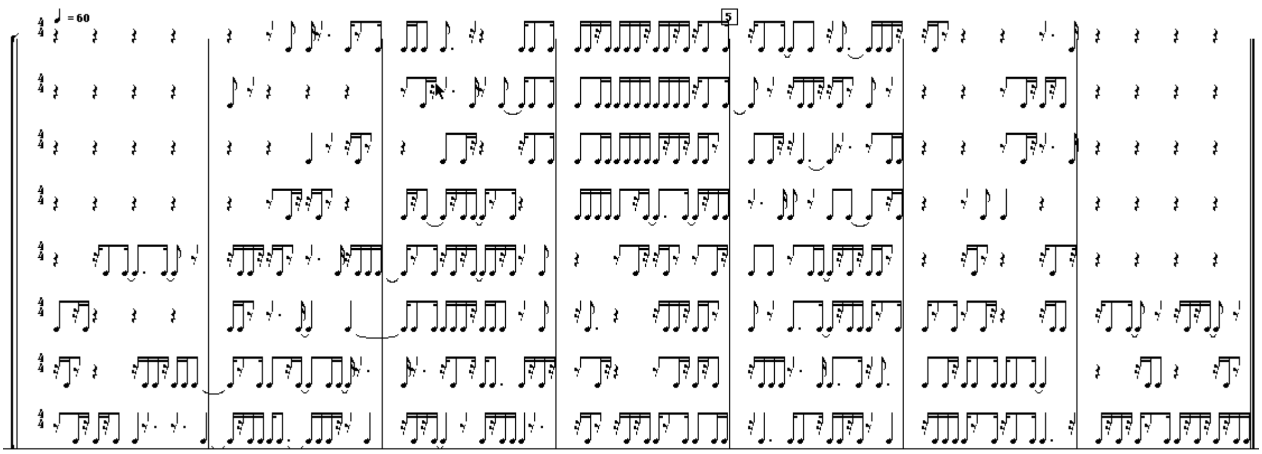

Figure 3. An eight-part example with varying density.

An attack in the final result can consist of any PW-chord object. Due to the flexible nature of the PW-chord object, an attack can thus have several textural interpretations: it can be a simple one-note chord with various dynamic levels, it can be a multi-note chord resulting in attacks with varying note density, it can consist of a multi-note chord where the notes have varying offset times (relative to the start time of the chord) resulting in "grace chords", "clouds of notes", etc. A chord can also be attached a label (such as "pizz", "harm", "gliss") allowing for instance to define different modes of playing when working with instrumental parts. Figure 4 gives an example with various PW-chord manifestations:

Figure 4. An imitative texture example with one-note chords, two-note chords, simple grace-notes and grace-note groups.

Our next example demonstrates how Texture-PMC can be used as a kind of textural "analysis/resynthesis" tool. A musical excerpt is analysed by the user and the rhythmical and textural features are converted to Texture-PMC rules. After this Texture-PMC is run in order to reproduce - or "resynthesise" - the original musical texture. Typically the result is not an exact replicate of the original. Figure 5 shows one result using seven textural rules (the original musical excerpt comes from the first part of György Ligeti's "Ten pieces for wind quintet"):

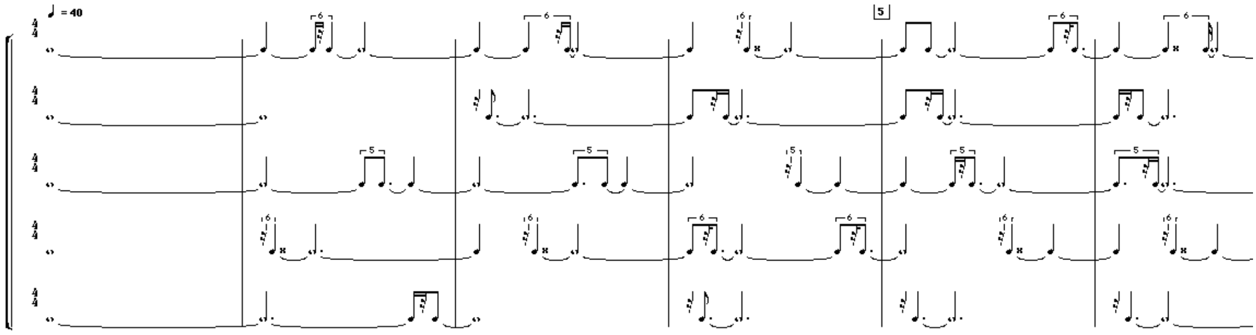

Figure 5. A "resynthesis" of the beginning of Ligeti's "Ten pieces for wind quintet".

We give below two rules which define some important characteristics of Figure 5. The first rule states that there should be no downbeat attacks after measure one. The second rule, in turn, states that only one-note attacks are allowed after measure one.

```
(* ?1 (and (> (measurenum ?1) 1) (downbeat?  ?1))
   (?if (or (= (r ?1) tie) (= (r ?1) rest))) "no downbeat attacks")

(* ?1  (> (measurenum ?1) 1)
   (?if (atmost-N-attacks? ?1 1)) "at most one attack")
```

Other rules control the exceptional status of measure one, the density of attacks and how rests are positioned in the resulting texture.

3 Instrumental Writing

One of the main difficulties in compositional systems is to keep track of the history of the decisions that have been made during the search. In PWConstraints the past history is stored in a polyphonic input-score which represents the rhythmic skeleton of the search problem. This scheme allows to write complex melodic, harmonic and voice leading rules. In order to extend the system with idiomatic instrumental writing each part in the input-score is given a "phantom" instrumental part. This extra part duplicates the rhythmic skeleton of the main part and contains the history - fingerings, bowings, pedals - of how the instrument is going to be played. The instrumental part has its own set of rules which are run in parallel with the rules of the main part.

In order to clarify our approach we next discuss two case studies. The first one - the classical guitar - is a good example of an instrument which is capable of playing a great number of different texture types: single melodic lines, multi-voiced chords and polyphonic textures consisting typically of 2 or 3 independent parts. One must, however, consider several constraints that are inherent in the instrument: it has six strings tuned in a specific way; the left hand of the player can use only four fingers; there is a limit of how much the left hand fingers can be stretched apart; the player must have enough time when changing from one left hand position to another, etc. Thus it remains a challenge to write for the guitar using its full potential without making the result unplayable.

Our second case study is concerned with brass instrument fingerings, the trumpet in particular, in a virtuosic context. The trumpet is a brass instrument with three valves resulting in eight different fingerings. With each fingering a different overtone series can be played. When no valves are pressed the resulting overtone series in a C trumpet starts from C3 (we use here the convention where 'middle C' = C4). The first playable tone is always, however, the second partial. Thus, the lowest note achieved with an open fingering is C4. When all valves are pressed down (the maximum pipe length) this results in F#3 which is also the lowest natural tone of the trumpet. So called pedal tones are playable down to C3 and even lower. These principals can be applied also to other brass instruments with valves.

3.1 First Case Study: Classical Guitar

The starting point in our case studies is a database of all the main instruments of the standard western orchestra. For string instruments this means that we can access various instrument specific information concerning the number of strings, their tuning, etc. The user can create new instances out of the standard instruments through inheritance. This allows, for instance, to have instrument instances with different tunings (scordatura).

In order to demonstrate the effect of instrument specific rules we give first a short texture example without fingering rules (Figure 6). (We refer in the following to the written pitch.) This example was produced in Score-PMC using various melodic and harmonic rules (due to space limitations we give here only some rough ideas about the rules that were used to produce the result): all five-note chords have the set-class identity 5-28a or 5-28b; one and two-note textures avoid pitch-class duplicates and belong to a harmonic series starting from F2; a heuristic rule aims to produce a pitch contour that resembles a triangle-shaped wave form, etc. It is obvious that the musical texture of Figure 6 is not playable with the guitar:

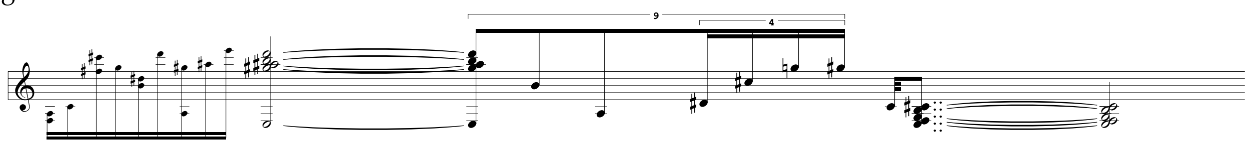

Figure 6. A texture example without fingering rules.

Figure 7, however, gives an example that utilises besides the melodic and harmonic rules discussed above also instrument specific fingering rules. The search is prepared as any other Score-PMC search except that for each instrumental part one extra part - this part is rhythmically an exact duplicate of the instrumental part - is created which will contain information on how the guitar part will be fingered. The lower part in Figure 7 shows the actual musical result containing the rhythms and pitches that are intended to be realised by a guitarist. The upper part, in turn, gives the fingerings. The fingerings are given as pitches of the open strings of the guitar (therefore this part can only contain six pitches: E3, A3, D4, G4, H4, and E5). Thus, for instance in the first chord the lowest pitch in part 2, F3, is to be realised with the sixth string (= E3 in part 1), the highest pitch in the first chord, A3, is using the fifth string (= A3 in part 1), the second note, C4, is using the fifth string (= A3 in part 1), and so on. (We use here a somewhat simplified fingering system as we do not give information which left hand fingers should be used. Our approach - using only open strings - was mainly motivated by the need to keep the system as simple and manageable as possible).

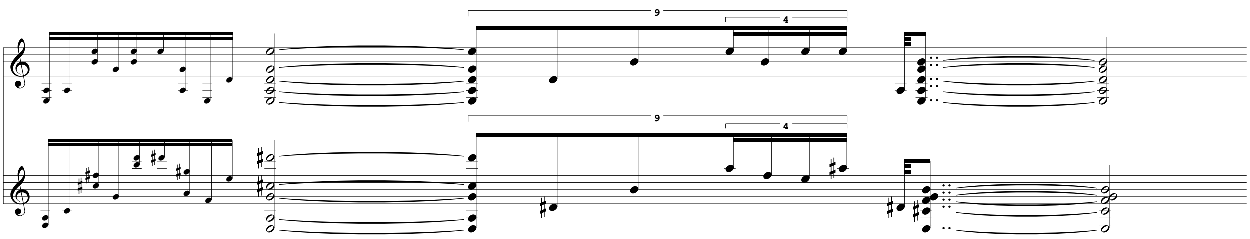

Figure 7. A guitar texture example with fingering rules. Upper part gives the "phantom" part. Lower part gives the musical result to be played by a guitarist.

The scheme presented in Figure 7 has many interesting features. The system knows the complete history of the found pitch information up to the current point in the search (lower part). Also there is the complete history available of the fingerings (upper part). Each time a new pitch value is chosen it must simultaneously also fulfil the fingering rules. The fingering rules can be time and texture sensitive. Different rules can be applied for musical textures consisting of fast notes, slow notes, single notes, chords or some other mixed type of textures.

3.2 Second Case Study: Brass Instrument Fingerings

When considering brass instrument fingerings, and those of the trumpet in particular, it can be seen that this case study is somewhat simpler than the previous one. The trumpet is a monophonic instrument, therefore no rules for polyphonic behaviour need to be defined. The basic fingering scheme of the valve instruments is quite self explanatory. By selecting a tube length with different valve combinations, it is possible to play in any of the seven overtone series available. This allows a brass instrument with valves to produce all the notes of the chromatic scale.

However, the brass instrument fingering scheme poses also problems. It may be possible to play a pitch with several different fingerings, but some of those fingerings may result in a pitch that is out of tune or difficult to produce. These are both instrument and player specific problems and they have been taken into consideration by using a trumpet fingering database. The database consists of information of which pitches are possible to produce using a given fingering. Special cases, such as partials that are out of tune, were also collected into the database.

We will demonstrate two different applications on how trumpet fingerings can be used in a compositional context. First, it is helpful to use "static fingerings" when managing rapid passages. Static fingerings may allow the player to maintain the same fingering in a sequence consisting of alternating pitches. Second, fingering patterns can be learned and then repeated several times within a musical passage. Both approaches allow rapid passages to be executed with relative ease.

The examples below use a similar set of melodic rules. Some rules control the pitch-class contents of the examples: tonal implications are avoided by discarding major and minor triads, diminished chords and certain subsets of major and minor scales. Heuristic rules are also applied to control the contour of the two last examples.

3.2.1 Static fingerings

Static fingerings are especially useful when using higher partials because these partials lie close to each other. The higher we go in range, the larger will be the amount of different pitches that can be played with only one fingering.

A typical technique for the brass instruments is called “lip trill”. When lip trilling a brass player does not use different fingerings to produce a trilled note, but uses instead the embouchure and air pressure (Figure 8). This technique is useful for intervals below minor third for classical trumpet applications. Intervals larger than this - “a shake” - can be executed in the highest register and these are considered to be part of the jazz trumpet vocabulary.

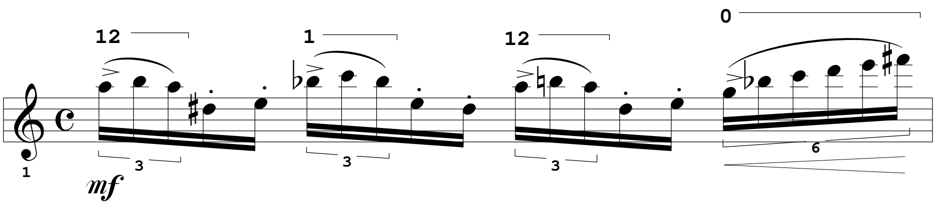

Figure 8. The use of lip trilling technique. The fastest rhythms use only one fingering at a time (the fingerings are shown above the respective notes). The passage is intended to be performed in a tempo around 144 per quarter note.

3.2.2 Fingering patterns

Fingering patterns, kind of “fingering canons”, can also be utilised. This means that the player can repeat certain fingering sequences over and over again. The musical texture can reflect the fact that the fingering patterns remain the same by for instance repeating pitch patterns in the register space. Another possibility is to do just the opposite, trying to keep the fingering and pitch patterns as independent of each other as possible, as can be seen in the Figure 9:

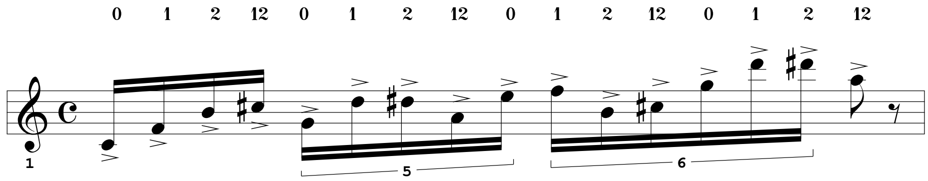

Figure 9. An example of a fingering pattern being repeated through a rapid passage of music.

As a final example we give a musical excerpt from a larger context. Figure 10 demonstrates the use of a simple fingering pattern (1, 12, 1, 12, etc.). This kind of a pattern is very easy to play as it involves the movement of one finger only. The same pattern is repeated throughout the example. This kind of passage can be executed very rapidly to give an impression of great virtuosity.

The rules used here are somewhat different than the ones described above. Apart from the rules that define the use of the fingering patterns there are a few rules that affect the contour of the individual figures. The change of direction in the rapid runs is not allowed except for the second figure from the end. Furthermore, the contour of the longer notes is controlled by restricting the interval between two consecutive eighth-notes to be smaller than fifth. Pitch repetitions among the eighth-notes are also disallowed.

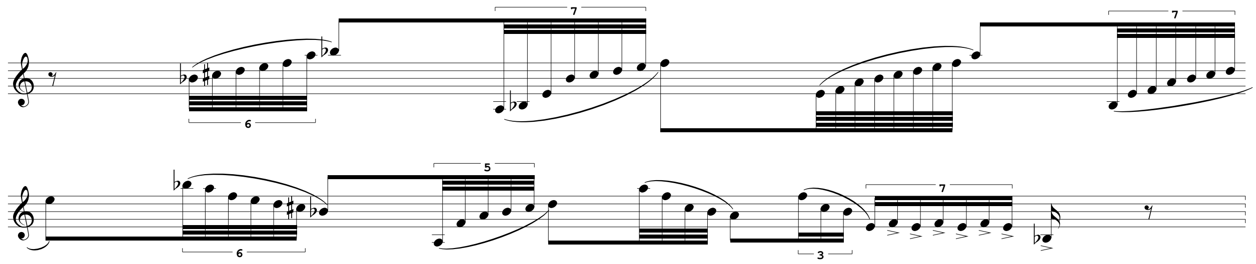

Figure 10. An example of the use of fingering patterns in a cadenza written for trumpet (Konzertstück for trumpet and wind orchestra by Mika Kuuskankare).

Conclusions

We have discussed some background information concerning PWConstraints. After this we showed how PWConstraints can be used to solve several musical problems. First we extended the system so that it is able to produce musical textures. After this we discussed how the user can specify instrumental rules within a constraint based context. In a typical case our approach allows to produce complex results by using only a relatively small amount of rules.

Acknowledgements

This research has been conducted within the project "Sounding Score—Modelling of Musical Instruments, Virtual Musical Instruments and their Control" financed by the Academy of Finland.

References

Laurson, M. and Duthen J. (1989); PatchWork, a graphical language in PreForm; Proc. Int. Computer Music Conf., San Francisco (pp. 172-175)

Laurson, M. (1996a); PATCHWORK: A Visual Programming Language and Some Musical Applications; Doctoral dissertation, Sibelius Academy, Helsinki, Finland

Laurson, M. (1996b); PWConstraints Reference Manual; IRCAM, Paris, France

Rueda C., Lindberg M., Laurson M., Bloch G. and Assayag G. (1998); Integrating Constraint Programming in Visual Musical Composition Languages; in ECAI 98 Workshop on Constraints for Artistic Applications, Brighton.

Laurson, M. (1999); Recent Developments in PatchWork: PWConstraints - a Rule Based Approach to Complex Musical Problems. In Systems Research in the Arts: Volume I, Musicology. Editors: George

Lasker and James Rhodes. The International Institute for Advanced Studies in Systems Research and Cybernetics, Windsor, Ontario, 1999.