Enriched Score Access for Computer Assisted Composition in PWGL

Mika Kuuskankare

Sibelius Academy, Department for Doctoral Studies in Music and Research,
Helsinki, Finland
STMS : IRCAM/CNRS/UPMC, Paris, France
mika.kuuskankare@siba.fi

Abstract. PWGL is a visual composition environment that can be used to, among other things, solve musical constraints problems. The constraints system within PWGL, PWGLConstraints, allows us to write rules using a special pattern-matching language. Typically, the assignments use as a starting point a score prepared with the help of Expressive Notation Package (ENP). In this paper we present an extension to the PWGLConstraints pattern-matching language which allows us to access information from ENP to assist with the compositional process. ENP provides a rich library of standard and user-definable expressions called ENP-expressions. They range from standard articulation markings (such as staccatos and slurs) to fully interactive multi-purpose graphical expressions. A special syntax is developed which allows us to retrieve information about and contained by the expressions. In this paper, the syntax and the present state of the system are illustrated using a working example.

Keywords: constraint-based computer-assisted composition, visualizing musical constraints, computer-assisted music notation.

1 Introduction

There exists a wide range of general-purpose rule-based systems that have been used to solve musical constraint satisfaction problems, such as Situation [1], Arno [2], OMClouds [3], and Strasheela [4]. PWGLConstraints [8] is a rule-based pattern-matching language within PWGL [5] which can be used to solve musical problems. The main advantage of our system when compared to the aforementioned systems is that PWGLConstraints is closely integrated with a flexible music notation front-end, the Expressive Notation Package, or ENP [6]. Typically, the PWGLConstraints assignments use as a starting point a score skeleton prepared with the help of ENP. ENP allows us to enrich scores with additional attributes called ENP-expressions [7], which are Lisp-based multipurpose graphical objects that can be used to represent different kinds of information as part of a musical texture. The traditional expression markings, such as articulations and dynamic markings, form a subset of ENP-Expressions. The extension of

PWGLConstraints presented in this paper allows us to access the information about and contained by the ENP-expressions, using a simple but powerful syntax. The new syntax can be seen as a supplement to the existing accessor scheme reported in [8]. However, it allows us to write rules that access the score information in a completely new manner. Throughout this paper we will refer to the PWGLConstraints syntax which will not be presented here in detail. Instead, the reader is advised to study [8] for more information.

2 The New Expression Access Scheme

Access to the information contained by the expressions is provided through a new primitive called `e`. It takes as an argument a score object and a collection of keywords (for example, :pos, :first?, :last?, :sample, :at, :id) and finally returns either an object or a property or a list of objects or properties.

Example 1 shows the basic syntactic components of the system (the symbol `?1` represents a single note object in the score). The form `(e ?1 :accent)` is of interest here. It is a condition stating that the rule is executed only if there is an accent present in the note referenced by `?1`.

Example 1: A rule executed only for accented notes. The ellipsis denotes the rule part where the user defines the actual constraints.

```
(* ?1 (e ?1 :accent) (?if ...))
```

The e-syntax also allows us to "sample" an expression at any given point in time (obviously the ENP-expression in question has to contain time-varying information, such as a breakpoint function). In Example 2 an expression is sampled and the melodic movement is matched against the values retrieved from the expression using the combination of two keywords `:sample` and `:at`. The keyword `:at` defines the point in time that we want to sample. The argument can either be absolute time in seconds or a score object. Thus, `:at ?1` means that we want to access the value contained by the expression at exactly the onset-time of the note ?1.

Example 2: A rule forcing the melody to follow `:object3` exactly.

```
(* ?1 (?if (let ((ref-pitch (e ?1 :object3 :sample :at ?1)))
             (= (m ?1) ref-pitch))))
```

3 An Example of Basic Expression Access

In this section we provide a concise example using our new access scheme. Here, contrasting texture profiles are defined with the help of accents and slurs. Slurred texture uses small intervals and the accented texture, in turn, uses leaps larger than an octave. To make the example more interesting, we also use a collection of supplementary rules where, for example, the pitches inside each beat (i.e.,

non-interleaving four note groups) must result in a pitch-class set 4-7. However, due to space limitations these rules are not discussed here. The resulting score can be seen in Fig. 1.

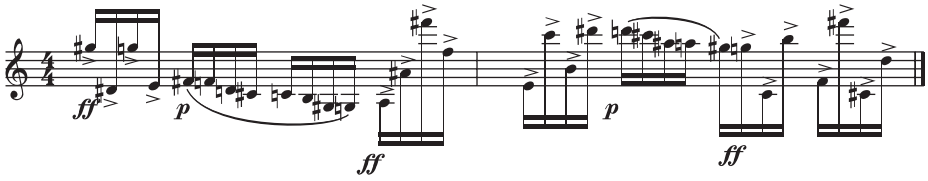

Fig. 1. contrasting textures created with the help of ENP-expressions: the slurred notes use small intervals and the accented ones use large intervals.

The two expression access rules (Examples 3 and 4) are used to define the two contrasting textures. The variables `?1` and `?2` in the pattern-matching part (line 1) give access to every two consecutive note objects in the score. The form `(and (e ?1 :accent) (e ?2 :accent))`, in turn, is a condition indicating that the rule is executed only if there is an accent on both of the notes. Finally, the rule simply states that the absolute value of the interval between the notes must be greater than an octave (12 semitones).

Example 3: Use big leaps on accented notes.

```
(* ?1 ?2
   (and (e ?1 :accent) (e ?2 :accent))
   (?if (> (abs (- (m ?1) (m ?2))) 12)))
```

Our second rule (Example 4) is almost identical to the first one but, instead of the `:accent` keyword, uses the `:slur` keyword to indicate that the rule is to be executed only on slurred passages. In the rule part, we specify that all of the intervals inside the slurred passages must be between the minor second and the major third intervals (the intervals must also be descending).

Example 4: Small descending intervals on slurred notes.

```
(* ?1 ?2
   (and (e ?1 :slur) (e ?2 :slur))
   (?if (<= 1 (- (m ?1) (m ?2)) 4)))
```

Finally, Example 5 defines the transition from one texture to another. Here, we indicate (lines 2-3) that this rule is active only when we are going from an accented note to a slurred one (line 2), or vice versa (line 3). In both cases, the interval between the two sections is constrained in such a way that it is either a minor or a major second (see line 4).

Note, that without this rule, the interval between the two contrasting textures would be random, since, by default, there is no rule that would be applied in this case.

Example 5: When going from one texture to another use a stepwise movement.

```
(* ?1 ?2
   (or (and (e ?1 :accent) (e ?2 :slur))
       (and (e ?1 :slur) (e ?2 :accent)))
   (?if (<= 1 (abs (- (m ?1) (m ?2))) 2)))
```

4 Conclusions

This paper presents a new scheme which allows us to access score information for the purposes of computer-assisted composition in the visual programming language PWGL. ENP, the music notation system of PWGL, is used to prepare the starting point of the search. The scores can be enriched with standard or user-definable markings called ENP-expressions. The information about and contained by the ENP-expressions can then be used by our rule-based compositional system to guide the search process. The underlying expression system itself is quite powerful and also user-extendable. Basically, any expression can be given an arbitrary meaning by the user. The scheme described in this paper is capable of expressing and solving highly sophisticated musical problems.

Acknowledgments

The work of Mika Kuuskankare has been supported by the Academy of Finland (SA137619).

References

1. Rueda, C., Lindberg, M., Laurson, M., Bloch, G., Assayag, G.: Integrating constraint programming in visual musical composition languages. In: ECAI 98 Workshop on Constraints for Artistic Applications. Brighton (1998)
2. Anders, T.: Arno: Constraints programming in common music. In: Proceedings of the International Computer Music Conference (2000)
3. Truchet, C., Assayag, G., Codognet, Ph.: OMClouds, a heuristic solver for musical constraints. In: MIC03 Metaheuristics International Conference, Kyoto, Japan (2003)
4. Anders, T.: Composing Music by Composing Rules: Design and Usage of a Generic Music Constraint System. Ph.D. thesis, Queen's University, Belfast (2007)
5. Laurson, M., Kuuskankare, M., Norilo, V.: An Overview of PWGL, a Visual Programming Environment for Music. Computer Music Journal 33(1), 19–31 (2009)
6. Kuuskankare, M., Laurson, M.: Expressive Notation Package. Computer Music Journal 30(4), 67–79 (2006)
7. Kuuskankare, M., Laurson, M.: ENP-Expressions, Score-BPF as a Case Study. In: Proceedings of International Computer Music Conference. pp. 103–106. Singapore (2003)
8. Laurson, M., Kuuskankare, M.: Extensible Constraint Syntax Through Score Accessors. In: Journées d'Informatique Musicale. pp. 27–32. Paris, France (2005)