W. Buxton, S. Patel, W. Reeves, and R. Baecker
Structured Sound Synthesis Project
Computer Systems Research Group
University of Toronto
Toronto, Ontario, Canada M5S 1A1

Scope in Interactive Score Editors

Introduction

In conversation, we typically constrain our comments to fall within the scope of the topic at hand. *Scope* can be used to define the sphere of action of any activity. When conductors request that players "play a little more staccato in the lower brass," they are imposing a certain scope (lower brass) on a specific action (play). Scope is an important concept in many contexts in which we desire to specify the precise range of some command. In this article we will consider the expression of scope in interactive score editors (Buxton et al. 1979).

Most existing score editors restrict the degree to which they permit users to impose scope upon operators, or commands (Smith 1972; Reeves et al. 1978; Wallraff 1978). For the most part, operators (such as "play" or "delete") can only be directed toward single notes or entire scores. This is unsatisfactory because ideally any operator should be able to be directed to any arbitrary grouping of notes the composer thinks of in the mind's ear. There are, however, considerable problems in implementing such a facility, most notably developing the semantics and syntax appropriate for such specification in a musical context. Addressing these issues is a current topic of research with the Structured Synthesis Sound Project (SSSP). Although this research is still in progress, we believe that the interim results presented here help to give some structure to the problem and will be of use to those currently writing or designing score-editing software.

S. Patel is currently with Human Computing Resources, Toronto, Canada. W. Reeves is now with Lucasfilm Corp., Novato, California. R. Baecker is with both the University of Toronto and Human Computing Resources.

Computer Music Journal, Vol. 5, No. 3, Fall 1981,
0148-9267/81/000050-07 $05.00/0

The Semantics of Scope

The objective in scope specification is to identify without ambiguity the notes that serve as the operand of a command. If this is to be done fluently, the grouping criteria must match the way that the notes may be grouped in the composer's mind. Such criteria form the basis for an unambiguous description of the notes. A first step in our research was to attempt to gain some understanding of these criteria through techniques of observation and interrogation. As a result, we have come to view grouping criteria as falling into five basic categories:

- **Simple grouping criterion** — scope is either the whole score or a single note (e.g., "play the third note").
- **Block-of-time grouping criterion** — scope is a set of notes contained in a particular time interval (e.g., "delete the notes of the third bar").
- **Local-attributes grouping criterion** — notes are encompassed in the scope on the basis of self-contained attributes such as pitch or duration (e.g., "raise the volume of all notes below C3").
- **Contextual-attributes grouping criterion** — a note's context (relationship to other notes and their attributes) determines whether the note is included in the scope of an operator (e.g., specifications such as "all notes followed by a leap of an octave upwards" or "sounding simultaneously with three or more other notes").
- **Named-structural-entities grouping criterion** — scope is specified by naming the musical structure to be affected (e.g., "motif A").

We see each successive category as growing in semantic power. Together, these categories form a foundation for the semantics of scope specification.

Simple Scope

When scope can only be expressed as a single note or an entire score, we refer to it as *simple scope*. If a program allows both single-note and whole-score scope to be expressed, the minimum basis for a score editor is provided. With regard to the underlying representation of the score data, simple scope presents no real problems since the operand of a command is either the entire data set or a single record. At this level, scope lends itself well to either a graphics or alphanumeric mode of specification. In graphics mode, for example, the entire score can be encompassed by activating the light-button corresponding to the desired operator. An operator can modify a single note by pointing at the desired note. (See the article by Buxton et al. [1979] for more details on such graphic techniques.) Alphanumeric-based interaction, on the other hand, can be modeled on line-oriented text editors. In such cases, scope can be specified as the line currently being edited or as the entire score file.

Scope by Block of Time

While simple scope can provide the basis for an elementary score editor, many compositionally important concepts are commonly encountered that cannot be handled by such an editor. For example, we might want to audition the last section of a long score or a sequence of notes in the middle, or we might want to transpose a particular chord. In each case, we want to address the set of notes falling within a particular *block of time*. The situation is commonplace in which such blocks are expressed with regard to note position by number (as in our first example), bars, rehearsal marks, seconds, or beats. If such concepts can be expressed in a congenial way, the editor's power is substantially increased, as is its usefulness to the composer. For example, notes of isolated "chunks" of the score can be heard in context, chords can be treated as single entities, and entire sections can be saved or copied in a single gesture for the purpose of repeats.

In making scope specification available by block of time, the designer must pay close attention to delimiters and their effect on the underlying data structures. Bar lines or rehearsal marks cannot be used unless these concepts are kept in the data base. If time is represented by fractions of a beat with respect to a metronome marking (as in the SSSP system described by Buxton et al. [1978]), specification by units of real time (such as seconds) may be awkward to implement.

The mode of interaction also has an effect on the techniques of delimitation available in a system. Graphics-based systems that permit scrolling through and zooming in and out of the score lend themselves well to use of the current viewport as the scope delimiter. The specification of blocks of time by the use of markers on a time line is another technique better suited to a graphics-based approach. Alphanumeric systems are often more appropriate when numerical values are used as delimiters, such as when precise timings in seconds are required.

An ideal environment would provide a number of alternative methods for delimiting scope. For the purposes of the current discussion, however, we will present only one sample approach taken from the SSSP alphanumeric score editor *sced* (Buxton 1981*a*). *sced* is modeled on the line-oriented text editor *ed* (Kernighan 1974). *ed* is the most commonly used text editor at our facility. While from a purely musical perspective it is not the best model to use, *sced* was based on *ed* to permit text-editing skills to be applied to music and vice versa. However, we chose not to edit scores as text files. Rather, SSSP music editors operate directly on the musical data structures. This has several advantages: (1) scores need not be compiled into a lower-level representation to be performed, which results in fewer files and operations and permits real-time synthesis for purposes of verification; (2) the editor can be interpretative because it has knowledge about the various operations and can therefore provide error checking and other forms of help; (3) scores are edited in the common language of the SSSP system and therefore any score can be edited by any score editor whether or not it is graphics-based.

In *sced* note numbers rather than line numbers are the basis for navigating through the score. Thus note number is the basis for scope delimitation.

The following examples illustrate sample implementation of scope by block of time. First, the specification of scope at this semantic level encompasses the previous level, simple scope. The first example,

```
3d
```

means "delete the third note," while

```
*p
```

means "print the entire score" (by convention, * means "all notes"). Double delimiters are used to expand upon simple scope. The example

```
3,8l
```

means "let me listen (l) to notes three through eight," while the statement

```
10,$w fred
```

means "write (w) or save notes ten through to the end (specified by the special symbol $) as a new score called fred."

The specification of scope at this level has met with a very positive response from composers. Certain frustrations have arisen, however, which demonstrate that extraction of a block of time cannot enable us to handle all cases encountered in the editing of scores. Therefore the technique is insufficient, regardless of the units used for boundary delimitation.

Scope by Local Attributes

The problem with specifying scope using the techniques discussed thus far is that all notes in the indicated block of time are encompassed. Thus concepts such as "all quarter notes in the third bar" or "all brass playing *mf* below middle C in this section" cannot be accommodated. While time-block specification helps the composer focus in on one whole area of the score, it does not permit the identification of the specific notes in question. Such identification can be made if a note has some user-specified characteristic included in the scope. For the purposes of this discussion, let us say that the "legal" characteristics are the *local attributes* of the notes themselves. Besides block of time, we should be able to take a note's pitch, duration, loudness, instrument, and other properties into consideration before we include it in the scope.

The power of scope specification at this level depends on the composer's ability to express arbitrarily complex relations among these attributes. This implies that logical relations (e.g., equivalence, greater than, not equal to, etc.) and conjunctions (e.g., and, or, and exclusive or) must be expressible.

With implemented score-editing systems, the crucial limitation is that only attribute fields that form part of the data base can be considered as criteria for inclusion in the current scope. It is not possible to specify "all notes played by trumpets," for example, if no record is kept of orchestration. (The only exceptions to this are attributes that are implicit, such as note number or time. With this provision, it becomes clear that simple and block-of-time levels of scope specification are encompassed by the local-attributes level.) While it is easy for a musician to group notes mentally according to some relationship, it becomes more difficult when notes must be unambiguously identified to a computer. To get a feeling for some of the problems involved, let us look at some specific examples. Again, these are taken from the alphanumeric score editor *sced*.

```
{freq < C4}p
```

means "print all notes having a frequency less than C4." (In *sced*, the relational expression appears between braces and before the operator. In this and other examples, key words such as "frequency" indicate particular note attributes. All key words can be abbreviated in order to reduce typing.)

```
{freq = C4, G3}orch flute
```

will cause all notes having a pitch of C4 or G3 to be orchestrated with the "flute." An example of a more complex relation would be

{dur <= ¼ & obj != flute}d

which would delete all notes not orchestrated by "flute" that had a duration less than or equal to a quarter note.

{# >= 3 & # <= 12 & vol <100}setvol +20

can also be written as

3,12{vol < 100}sv +20,

which means "increment by 20 units the volume of all notes from the 3rd through the 12th whose volume is less than 100."

There are several points worth noting about these examples. First, it would be difficult to express such logical relations using graphical techniques. Although the concepts can be expressed easily in natural language, the only way to express them to a computer is with typed alphanumerics. The syntax used to express the relational concepts in the examples is rather arcane. From a human-factors point of view, some trade-off must be made between similarity to natural language and succinctness. (Some feeling for this conflict can be seen in the last example—compare the natural-language interpretation with the second version in *sced* notation.) Clearly, much research remains to be done. We hope that work such as that of Card, Moran, and Newell (1980) and Ledgard and coworkers (1980) will help pave the way. These researchers raise issues that must be dealt with in future score editors. The paper by Card, Moran, and Newell is a study quantifying the relationship between the verbosity of a message and the efficiency of the human-computer dialogue. The paper by the second group investigates the influence of natural-language-based constructs on the efficiency of text-editing tasks.

Second, while the simple and block-of-time levels of scope could be carried out using general-purpose text editors (if the score is represented as a text file), this is not the case with the local-attributes level. Here it is clear that the application-specific nature of the attributes (and the operators, e.g., "orch," or "orchestrate a note by attaching an instrument to it") makes the expression of complex logical relations impractical. We are no longer dealing with concepts that can be expressed simply as typographical operators. To enable scope specification at this level, it appears that a special-purpose editor that understands musical attributes must be provided.

Scope by Contextual Attributes

The previous section showed how a score editor's power can be increased by having a note's inclusion in the scope depend on the note's conformation to certain composer-defined characteristics. Often, however, more complex relations than those seen thus far are required. Simple relations based on pitch or duration are not enough. Rather, scope inclusion is sometimes best specified in terms of the notes' *contextual attributes*, or of the relationships among different notes and their attributes. We might want to specify, for example, a type of chord: "all notes that sound in combination with three or more others," or a motif: "all notes orchestrated with brass playing a dotted eighth, followed by a sixteenth," or an intervallic sequence: "all notes preceded by a step downward of a minor second and followed by an upward leap greater than a perfect fifth."

The ability to express such constraints is clearly of musical value, and musicians have no difficulty in grouping notes according to such criteria verbally or mentally. When it comes to identifying (unambiguously) such groups to a computer, however, problems arise. In contrast to the situation in which scope is defined by local attributes, the number of criteria for grouping in context is infinite. To understand context, the editor must have a far higher level of musical "knowledge" (1) to understand the various criteria and (2) to perform the consequent pattern recognition on the data base that will isolate the notes in question. Even if the appropriate concepts could be understood by the editor, the specification language used by the com-

poser would probably be too cumbersome to be useful. We have seen, for example, how specification of local attributes can approach the threshold of practical complexity.

As formulated, specification of scope at the contextual level would seem impractical without a great deal of research. We can either undertake this research or reformulate the problem. For the purposes at hand, we have chosen to reformulate the problem.

Musicians have no problems dealing with the concepts under consideration. An alternative approach, therefore, would be to make best use of the requisite knowledge of the composer. The task is then to provide an environment in which the composer can use this knowledge to identify manually the desired notes in an efficient way. The original problem involved the computer collecting the notes that fit a description supplied by the composer. We are now sidestepping the description problem and having the composer identify the notes. A descriptive approach is being replaced by a demonstrative one.

The success of this demonstrative approach depends on two conditions. First, the notation must highlight the relevant features of the musical data. Second, a straightforward means of interaction must be provided. As a result of both of these conditions, we have chosen to take a graphics-based approach in our experiments at the contextual level. With graphics, we have the notational flexibility to highlight most of the different features and relationships involved. We also have a far greater range of options as means of interaction (alphanumerics is one such option).

We have implemented this technique in the program *scriva* (Buxton 1981*b*). In *scriva*, an intuitive gesture—circling—can be combined with spatially distributed data as a means of isolating desired groups. More than one circle can be drawn on the screen (Buxton et al. 1979). Doughnutlike circles within a circle can be drawn, with the effect that all notes within the larger circle, except those contained in the inner one, are included in the current scope. There can be as many "holes in the doughnut" as desired and a circle within the hole is again a circle of inclusion.

Often the notes we want to group into the contextual-attributes level of scope are distributed throughout the score, making circling impractical. In such cases, the desired notes can be "collected" if we point at them one by one with the graphics cursor (Buxton et al. 1979).

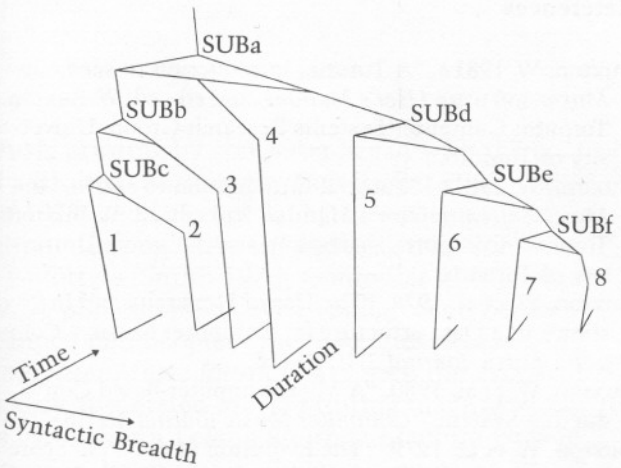
With the techniques seen in the examples we can make some headway in the specification of scope according to context. The techniques are still rather primitive, however, and much work remains to be done.

Scope by Named Structural Entities

At the contextual-attributes level of scope, we encountered problems owing to our inability to describe adequately the notes constituting the scope of an operator. As a result, we resorted to a demonstrative approach. Another alternative is to identify the notes in question by name. In cases where these notes constitute a musically significant structure in the score, this is common practice among musicians. Simple examples would be "motif A," "the second theme of the first movement," or "the ostinato in the second section." Names such as "the second theme" are too vague for automatic isolation of the intended notes by a computer program. This problem can often be circumvented if (1) we are more rigorous as to what constitutes a name and (2) if, when composed, musically significant structures are named by the composer.

What does this mean musically? Our approach to scope at the *named-structural-entity* level is based on the assumption that the score is more than a collection of notes. It is assumed that the score is made up of sections, motifs, and parts, each of which has an explicit name. Our approach is also based on the assumption that the composer has assembled the score so that there are relationships among these structures. Finally, our approach assumes that the internal representation reflects this structural "recipe." If these three conditions are met, there is no reason why any of these structural entities cannot be identified directly by name. At the contextual-attributes level, we failed to perform

Fig. 1. Snapshot of an image generated by a three-dimensional score display and editor developed at the SSSP.



the analysis that would extract the referenced part of the score. At the named-structural-entities level, the equivalent to such an analysis is given: it is the explicit structure as pieced together by the composer. This structure is reflected in the data structures.

Rendering such an approach practical has one other property. Whereas up to this point scope has been based on purely physical or acoustical properties of the data (time, pitch, timbre, etc.), we now have a means of scope specification with a compositional foundation!

There are, however, practical limitations to the named-entity approach. Fundamental among these is the need for an appropriate model for the underlying structure of scores. A list structure is too simple musically and a general relational network too complex to handle (due to space/time trade-offs that would result in poor performance). We have chosen a hierarchical "tree" structure (Buxton et al. 1978). While this clearly has musical limitations, it is the most general structure we feel to be manageable at this moment.

The technique is impractical except for larger-scale structures. It would not be reasonable to expect the composer to name every single entity. Yet it is exactly these larger structures that are the most poorly handled by the demonstrative technique. Smaller structures that are not well suited to the naming technique are generally easily handled by the demonstrative approach.

Our first experiments with structuring data by named entities had to do with notational techniques. How could the external representation of the data reflect the underlying structure so as to aid the composer in exploiting the information? The example shown in Fig. 1 is an illustration of this work. Illustrated is an example score called SUBa taken from the program *treed* (Kwan 1978) (contrived for the purposes of demonstrating the system). The feet at the bottom of the display represent note events whose durations are proportional to the length of the line that is the foot, along the time axis. The triangles labeled SUBb, SUBc, and so on represent large syntactic categories, that is, subscores. These subscores encapsulate two or more notes. Thus SUBa is the root of the tree, and SUBf is the lowest nonterminal. The numbers 1, 2, . . . 8 next to the terminal branches of the tree are labels used in editing. Since this is a snapshot of a dynamic program display that can be rotated in various ways, certain features that emerge in movement cannot easily be seen in this static image. For example, frequency information is embedded in this representation that comes out more clearly in the dynamic image. A clear structure is provided (in this example, at least) that facilitates the expression of concepts such as "play SUBb," or "transpose SUBd."

We do not yet have usable score editors that work on scores structured in this way. The number of problems encountered make progress very slow. One key challenge has been to use this complex level of internal representation while retaining our ability to play the scores in real time without any compilation or preprocessing. Our recent efforts to meet this challenge have focused on the performance of such structures. These efforts resulted in the *conduct* system (Buxton et al. 1980). One of the key concepts reflected in the *conduct* system is the ability to address and transform named structural entities in real time. *conduct* is a convincing example of the power and practicality of scope specification at this level. We are now trying to bring this power into the domain of our score-editing tools.

Conclusion

In this article we have described how the notion of scope can be developed within the context of interactive score editors. Our objective is to extend the means by which composers can express scope according to musically relevant constraints. There are three different syntactic approaches to scope definition: the descriptive approach, the demonstrative approach, and the naming approach. Differing combinations of syntactic approach and semantic level sometimes make conflicting demands on both the underlying internal representation and the mode of the musician-machine dialogue. With the descriptive approach, greater semantic power can be achieved by use of alphanumerics than by use of graphics-based techniques. On the other hand, graphics appear to be the most appropriate technique for the demonstrative approach. Both graphics and alphanumerics can be used in addressing named syntactic entities.

Acknowledgments

The work reported in this paper has benefited from discussions, comments, and other types of input from many people associated with the SSSP. In particular, we would like to acknowledge the contributions of James Montgomery, Martin Lamb, Wesley Lowe, William Matthews, Otto Laske, Leslie Gondor, K. C. Smith, Susan Frykberg, and Curtis Roads. The research has been funded by the Social Sciences and Humanities Research Council of Canada, whose support we gratefully acknowledge.

References

Buxton, W. 1981*a*. "A Tutorial Introduction to *sced*." In *Music Software User's Manual*, 2nd ed., ed. W. Buxton. Toronto: Computer Systems Research Group, University of Toronto.

Buxton, W. 1981*b*. "Tutorial Introduction to *scriva*." In *Music Software User's Manual*, 2nd ed., ed. W. Buxton. Toronto: Computer Systems Research Group, University of Toronto.

Buxton, W. et al. 1978. "The Use of Hierarchy and Instance in a Data Structure for Computer Music." *Computer Music Journal* 2(4):10–20.

Buxton, W. et al. 1980. "A Microcomputer-Based Conducting System." *Computer Music Journal* 4(1):8–21.

Buxton, W. et al. 1979. "The Evolution of the SSSP Score Editing Tools." *Computer Music Journal* 3(4):14–26.

Card, S. K., T. P. Moran, and A. Newell. 1980. "The Keystroke-Level Model for User Performance Time with Interactive Systems." *Communications of the Association for Computing Machinery* 23(7):396–410.

Kernighan, B. W. 1974. "A Tutorial Introduction to the UNIX Text Editor." Murray Hill, New Jersey: Bell Laboratories *Technical Memorandum 74-1273-17*.

Kwan, A. 1978. "Treed Project Report." Unpublished manuscript. Toronto: C.S.R.G., University of Toronto.

Ledgard, H. et al. 1980. "The Natural Language of Interactive Systems." *Communications of the Association for Computing Machinery* 23(10):556–563.

Reeves, W. et al. 1978. "Ludwig: An Example of Interactive Computer Graphics in a Score Editor." In *Proceedings of the 1978 International Computer Music Conference*, vol. 2, ed. C. Roads. Evanston, Illinois: Northwestern University Press, pp. 392–409.

Smith, L. 1972. "SCORE—A Musician's Approach to Computer Music." *Journal of the Audio Engineering Society* 20(1):7–14.

Walraff, D. 1978. "Nedit—A Graphical Editor for Musical Scores." In *Proceedings of the 1978 International Computer Music Conference*, vol. 2, ed. C. Roads. Evanston, Illinois: Northwestern University Press, pp. 410–450.