The Development of Set-Theoretic Analysis Tools for Macintosh

Peter Castine
Wildganssteig 28
D-13503 Berlin, Germany
pcastine@mvax.kgw.tu-berlin.de

ABSTRACT*: The Contemporary Music Analysis Package (CMAP) is a set of programs to assist theorists and composers in using set theory for analysis and composition. Originally implemented for the Unix operating system, CMAP provides the experienced user with a comprehensive set of tools for calculating set class membership, row matrices and partitions, set relations, and other information. Interest in making these tools available for other computer systems has been great. However, developing a version of CMAP for Macintosh has proved to be a challenge, due to the inherent differences between character-based and graphic user interfaces. At the same time, the Macintosh system software has facilitated the implementation of several features into CMAP that were not included in the original Unix version.*

KEYWORDS: Set Theory, Computer, Programming, Unix, Macintosh.

Background

Set-theoretic musical analysis, as described by Babbitt, Forte, Rahn, and many other writers, is a promising field for the development of computer-based tools. Most of the methods described in the literature are time-consuming and error-prone when done by hand. The relationships discussed in set-theoretic analysis are difficult (at best) to distinguish by ear, so that the audition of a piece of music is a less reliable means of checking the accuracy of a given analysis than, for instance, analyses of functional harmony. In addition, the convention of using decimal or hexadecimal numbering of pitch classes, introduced to provide a notation that would better lend itself to the more-or-less mathematical language used in the literature of set theory, is an extra step in the analysis process that is not free from error.

There are a number of computer-based tools available to aid in set-theoretic analysis. An extensive collection of tools is provided in the “Contemporary Music Analysis Package” (CMAP). CMAP provides functions to analyze and manipulate pitch class sets and pitch class rows, including calculation of interval, invariance, and adjacent interval vectors, analysis of set complex relations, and calculation of invariance relations among set class and in pitch class rows.

CMAP under Unix

The original version of CMAP was developed by Craig Harris and Alexander Brinkman between 1985 and 1987. CMAP was originally developed for the Unix operating system, and many characteristics of the CMAP package reflect this heritage, in particular, dependence upon the piping of filter programs and the use of Unix shell scripts for providing a flexible set of analysis tools to the experienced Unix user. Furthermore, a “C” function library is included with the CMAP package. This provides users with access to the data structures used by CMAP and tools for manipulating CMAP’s internal database.

CMAP was soon ported to run under MS-DOS. The MS-DOS version is essentially identical to the Unix version and provides access to the same set of tools. The batch programming language provided under MS-DOS, although less powerful than the Unix scripting language, still provides sufficient flexibility for the experienced user to adapt CMAP to his purposes.

Functions of CMAP

The basic CMAP package consists of 29 separate programs used for the analysis of pitch class sets. Without cataloguing the entire repertoire of CMAP programs, I hope to give an impression of the range

```
% getset  439a4583a  9283  4-15  8-28  b5a9b0

SETNAME Z-SET  M-SET  IC VECTOR INV VECTOR      ORDER             CYCLE
------- -----  -----  --------- ----------      -----          ----------
 6-6    z38    m38     [421242] (11000011)      {012567}       <113115>
 4-9    z0     m 9     [200022] (22226666)      {0167}         <1515>
4-15    z29    m29     [111111] (10000331)      {0146}         <1326>
8-28    z0     m28     [448444] (44440000)      {0134679a}     <12121212>
 5-5    z0     m14     [321121] (10000111)      {01237}        <11145>

% prime

type set: 439a4583a
34589a       = t3p( 6-6)       {012567} 38 38 421242 77 11000011 113115

type pc set: 9283
2389         = t2p( 4-9)         {0167}  9  0 200022 43 22226666 1515

type pc set: 4-15
0146         = t0p(4-15)         {0146} 29 29 111111 77 10000331 1326
```

Figure 1: *CMAP output for the programs* getset *and* prime. *In this, and in all following displays of CMAP output, data entered by the user is shown in bold face. Both programs display the set class and its normal order, the Z-related and M-related sets, and the interval class, invariance, and adjacent interval vectors of the sets entered. Additionally, prime shows the twelve-tone operation used to map the normal order of the set class to the set entered.*

of different functions available to the user.

Perhaps the most central programs are *getset* and *prime*. Both programs are used to enter either arbitrary collections of pitch classes or set class names. The programs display the set class name, the M-related and Z-related sets, the interval class, invariance, and adjacent interval vectors, and the prime forms of all sets entered. There are two differences between these programs. Prime is an interactive program used to analyze sets. Getset, on the other hand, will process all sets entered on the command line or, alternately, will generate all set classes of a given cardinality. In either case, getset also provides options for testing sets for properties of the interval vector and invariance. For instance, you can ask getset to list all set classes of cardinality four that contain all six interval classes exactly once.[1]

Further programs are available for processing sets. In every case, the sets to be processed are entered on the command line. The programs *complement*, *intersect*, *union*, *setdif*, and *symdif* do what one would expect. Several programs calculate twelve-tone operations: *transpose* can do more than what its name suggests—it can also apply inversion, M5 and M7 operations to operands specified on the command line; *rowop* performs the same operations (with the addition of retrogression) on ordered pc sets; and a third program, *pcmap*, has the primary function of calculating the mapping of each pitch class in the original set to its pendant in the set resulting from the twelve-tone operation invoked.

One of the more time-consuming activities in set theory is calculating row matrices. CMAP provides two functions for this activity. The first, *matrix*, will print out a standard twelve-tone row matrix and a matrix of order number specifying the position of each pitch class in the original row. The second, *imatrix*, generates invariance matrices as developed by Bo Alphonce [1974], which are useful for finding invariant properties of a pitch class row.

Four further programs operate on pitch class rows. *Rowstruct* will calculate all rows of a given cardinality imbricated in a given row, and *rotarray* will calculate rotational arrays as described by John Rogers [1968]. Rotarray will also calculate the set class for each vertical combination of pitch classes generated. The *rowcomb* program is similar in function to rotarray, however, the options provided are slightly different. Finally, the *partition* program will find all occurrences of a given set class in an ordered pc set. Set complex relationships[2] are even more tedious to calculate than row matrices. The program *kh* does all the dirty work. The user simply needs to type in all the set classes to analyze in the command line; the

```
% intersect 91827 ab90a92

set 1 = 12789
set 2 = 029ab

intersection = {29}

NORMAL ORDER   TTO NAME   PRIME FORM     M  Z IC VECTOR INV VECTOR CINT
============   === ====   ============  == == ========= ========== ======
92           = t9p( 2-5)   05            1  0  000010    11009988  57

% matrix -t ab6328715904

Row:    [a b 6 3 2 8 7 1 5 9 0 4 ]
      Pitch Class Numbers                Order Numbers

    a b 6 3 2 8 7 1 5 9 0 4      0 1 2 3 4 5 6 7 8 9 a b
    9 a 5 2 1 7 6 0 4 8 b 3      9 0 8 4 7 6 2 a b 5 1 3
    2 3 a 7 6 0 b 5 9 1 4 8      4 3 0 6 2 a 1 8 9 7 b 5
    5 6 1 a 9 3 2 8 0 4 7 b      8 2 7 0 9 3 4 5 a b 6 1
    6 7 2 b a 4 3 9 1 5 8 0      2 6 4 1 0 b 3 9 7 8 5 a
    0 1 8 5 4 a 9 3 7 b 2 6      a 7 5 8 b 0 9 3 6 1 4 2
    1 2 9 6 5 b a 4 8 0 3 7      7 4 9 2 8 1 0 b 5 a 3 6
    7 8 3 0 b 5 4 a 2 6 9 1      6 5 3 a 1 8 b 0 4 2 9 7
    3 4 b 8 7 1 0 6 a 2 5 9      3 b 1 5 6 7 a 2 0 4 8 9
    b 0 7 4 3 9 8 2 6 a 1 5      1 a 6 b 3 9 5 4 2 0 7 8
    8 9 4 1 0 6 5 b 3 7 a 2      5 9 b 7 a 2 8 1 3 6 0 4
    4 5 0 9 8 2 1 7 b 3 6 a      b 8 a 9 5 4 7 6 1 3 2 0
```

Figure 2: *CMAP output for the programs* intersect *and* matrix. *The output of the programs* union*,* setdif*, and* symdif *is similar to that of intersect. Matrix has an option for printing conventional note names, but input is limited to hexadecimal integers.*

program prints out a table of all K, Kh, and K* relations.

Several programs are available for calculating other similarity relations between a set and itself (*setmap*) or between two different sets (*subset*). Setmap takes all transpositions of a set, as well as transposition under inversion, M_5, and M_7. It then calculates the number of pitch classes common to both versions of the set, and the set class of the intersection. Additionally, several special relations (invariance, invariance with the complement, or an R_p relation[3] between the two versions) are flagged. Subset provides essentially the same information for two different sets.

Strengths and Limitations

For all the richness of functionality described here, there are still some things the CMAP package cannot do, at least not without the intervention of the Unix operating system. One (relatively trivial) example would be to test set classes for, say, interval vectors with at least two minor second but no tritoni. Or, we might want to know which trichords included in set class 4-12 can be constructed from the intersection of the prime form of 4-12 and the set resulting from any twelve-tone operation applied to the prime form, and which contain at least one minor third and map into their complement under the M_5-relation at least three times.[4]

For these, and similar tasks, CMAP makes use of the Unix concept of filters and pipes, as well as the facility for writing shell scripts. As an example, to solve the second problem posed above, the following Unix script will provide the desired result:

```
setmap 4-12 |grep rp |no >pcfile
getset -t3g0 -xcmg2 'cat pcfile'
```

This script makes use of the standard Unix filter *grep* to find those lines produced by setmap that contain

```
                         GETSET

Usage: getset [-ivozmca] [-s] [-tx(q)e(#)] (pitch classes) (cardinality)

Options:

        i:      print interval class
        v:      print invariance vector
        o:      print normal order
        z:      print Z-set
        m:      print M-set
        c:      print adjacent interval array
        a:      print all information (default)
        s:      print sets of the following cardinality
        t:      test for interval class vector
                (-t)#r#:        interval class, relation(elg), frequency
        e#:     test ic vector for presence of ic # (a: all interval test)
        x:      test for invariance vector
                (-x)wxyz: invariance class ((s)elf, (c)omplement),
                operator (t,i,m,w), relation(elg), frequency
        xq:     test for invariance vector, query
```

Figure 3: *On screen help provided for the program* getset.

exactly three of the pitch classes in the prime form of set class 4-12. These are subsequently piped through the CMAP filter *no*, which removes all information from each line except for the pitch classes. This information is stored in a file named pcfile. Finally, getset is invoked with the appropriate options for testing the desired features of the interval vector and of invariance.

This flexibility has its price. The user who wants to use CMAP to its fullest must be comfortable with programming shell scripts, and with the Unix concept of filters. Although these are not particularly difficult to learn, they do go beyond the training that most musicians experience. And, CMAP resembles many other Unix-based software packages, in that many of the programs feature a long list of options abbreviated to one letter, which are difficult to memorize and tedious to look up in the documentation.

Some other limitations are set by the assumption that an ANSI terminal is being used. For instance, pitch class matrices are effectively limited to fifteen-tone rows: longer rows will not fit on an eighty-column line and wrap over onto the following line. Invariance matrices (which may be based on two different pc rows) are normally “limited” by the imatrix program to 300 pitch classes vertically. However, since the user cannot scroll, no row with more than about twenty pitch classes can be usefully displayed on screen.

Finally, there is the question of how many musicians have a Unix machine at home or in their office. Although this aspect has been alleviated by the fact that CMAP is also available for MS-DOS, the fact remains that the majority of musicians who own (or have access to) a computer, use Macintoshes.

CMAP for Macintosh

A version of CMAP is currently under development for the Apple Macintosh. There were two primary concerns during the design stage of the CMAP for Macintosh project: how to incorporate the functions provided by CMAP in a unified, “Macintosh-like” user interface, and how to provide the flexibility offered by Unix scripts under an operating system that did not provide a scripting language.

The Analysis Scratchpad

Most functions of CMAP on the Macintosh are available through an *Analysis Scratchpad*. This is a window in which pitch class combinations can be entered, analyzed, and transformed. Set information is displayed in *analysis lines*, which show the set class name, prime form, interval vector, invariance vec-

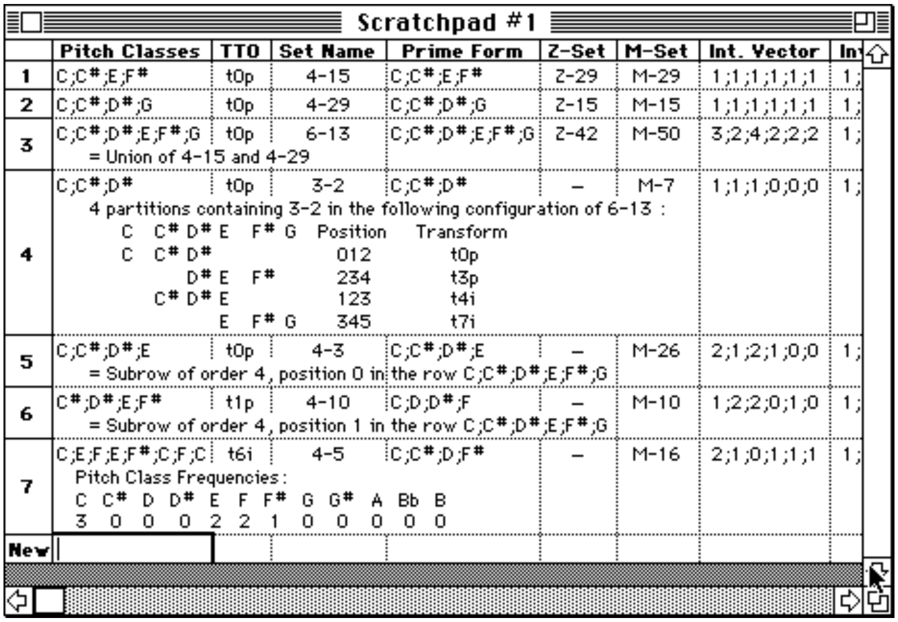


Figure 4: *The Analysis Scratchpad. Notice that data is entered using an integer notation, exactly like the Unix version. However, once data is entered, it is displayed using traditional note names.*

tor, and other related information for the pitch classes entered. Commands are provided to generate sets; to test set classes for properties of invariance and interval content; to perform set transformations such as complement, union, intersection, or difference to perform twelve-tone transformations; and to find inclusion relations between sets.

However, many of CMAP's functions simply do not fit into the Analysis Scratchpad. Those functions involving row matrices are obviously intransigent, but set complex tables, subset relations, and pc mapping are also out of place. The solution found was quite simple: define other display formats which are appropriate for the functions that do not fit in the Scratchpad. These display formats are implemented as individual document types.

Unfortunately, there are a dozen programs in the category "don't fit in the Scratchpad." A program design the requires the user to switch between such a large number of different windows types would be, simply, a poor design. Furthermore, the demands made on the programmer who must implement the design are distressing, to say the least.

One way to reduce the number of additional document types that must be used was to attach comments to analysis lines in the Scratchpad. The idea is quite simple: certain commands simply append a comment on to an analysis line. For instance, the Count Pitch Classes command appends a table with the twelve pc's and the frequency with which they occur in a row to each selected analysis line. Once the decision to append comments on to an analysis line has been taken, it seemed only logical to add comments to analysis lines generated by commands such as Intersect and Union. For these commands, a comment is not necessary to display the resulting set. Nevertheless, a comment reminding the user of how the set was generated can be a great convenience. And, as an extension of this idea, the user is also able to append comments to lines in the Scratchpad. This can be an aid in referencing sets to a musical work under analysis ("This is bar 20 of Klavierstück IX").

Still, there were a number of commands that did not fit well into the set-oriented tabular model of the Scratchpad, and that number was still more than the number of additional document windows I wanted to implement. A closer look at collection of programs in the original CMAP revealed a high degree of similarity between certain programs. Accordingly, during the design process, an analysis was made to see which of the original CMAP programs could be amalgamated into a single document window.

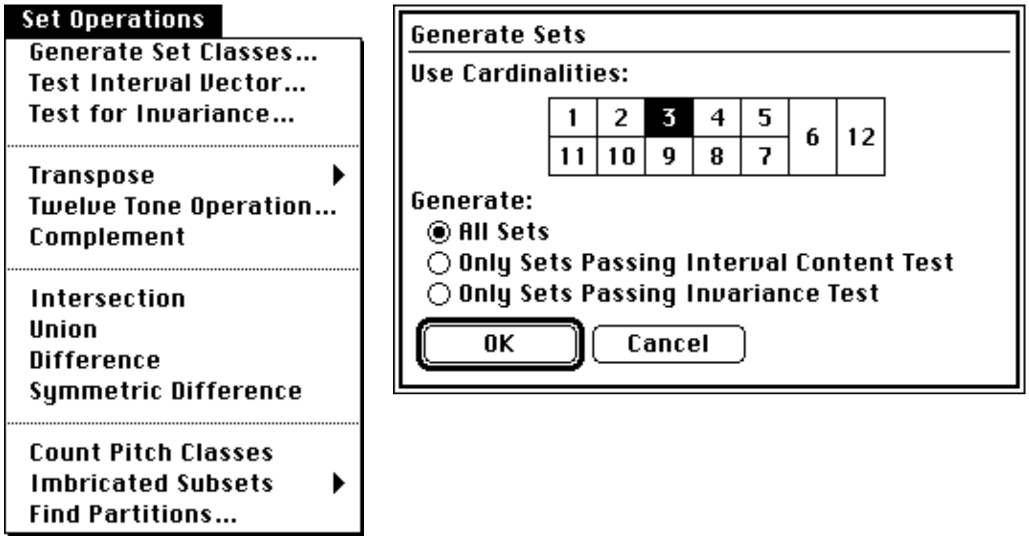


Figure 5: *Set operations menu of commands for manipulating ordered and unordered pc sets. Transpositions and imbrications are generated by means of hierarchical submenus. The dialog invoked by the Generate Set Classes command is displayed at the right.*

The result of this analysis was a division of the remaining functions among three further types of document windows. The Row Matrix window generates pitch class and pitch order matrices for one (or two) arbitrary rows entered by the user. It can also search for the occurrence of specific sets in the matrix. Commands are provided to transform rows (rotation and permutation) and to generate rotational arrays. The Set Relations window generates tables showing set complex and R-relations between set classes.[5] Finally, the Set mapping windows is used to show operations under which a set maps into itself, its complement, or into an arbitrary superset.

Data Entry

The Unix version of CMAP requires the user to enter pitch class data using hexadecimal notation with no spaces or other delimiters between pc's. Although this is an extremely efficient way of entering data, many users would prefer to view data in other forms, either in decimal notation, or note names.[6] There are a number of ways for the user to enter data into CMAP documents: both hexadecimal and decimal integers are supported, as well as note names. MIDI keyboards are also supported—a considerable improvement over typing in pitch class data as integers. The notation used during data entry may be converted to another format if the user so wishes.

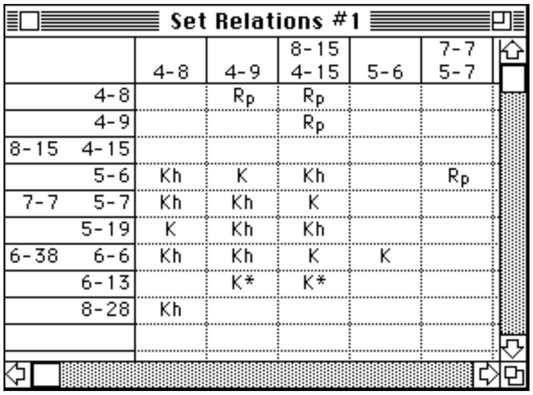
Combining Commands

If you want a function not directly provided by CMAP under Unix, you can always write a program to calculate it—the access to the set class database provided by the programming libraries makes such calculations much easier to program than if you had to start from scratch.

There are two levels of programming with the original CMAP version: use of a conventional programming language and shell scripts. Both are powerful tools for extending and customizing CMAP to the user's needs. This paper will focus on the second of these two forms of programming.

At the time work began on CMAP for Macintosh, the available system software did not provide any facilities for communication between programs, nor was there a facility for shell scripts. The only facilities available for automating program interaction were journaling utilities such as QuicKeys or MacroMaker.

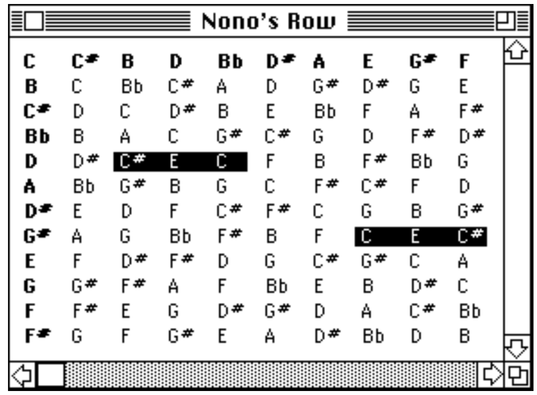
Examination of shell scripts written for the Unix version of CMAP reveals one common characteristic.



Set Relations #1

	4-8	4-9	8-15 4-15	5-6	7-7 5-7
4-8		Rp	Rp		
4-9			Rp		
8-15 4-15					
5-6	Kh	K	Kh		Rp
7-7 5-7	Kh	Kh	K		
5-19	K	Kh	Kh		
6-38 6-6	Kh	Kh	K	K	
6-13		K*	K*		
8-28	Kh				

Figure 6: *Set Relations window for calculating K/Kh, R_p, R_0, R_1, and R_2 relations. In this example, the Set Complex relations are shown in the lower diagonal, similarity relations are in the upper diagonal. Other display combinations are possible, including the triangular displays used in* The Structure of Atonal Music.



Nono's Row

C	**C#**	**B**	**D**	**Bb**	**D#**	**A**	**E**	**G#**	**F**
B	C	Bb	C#	A	D	G#	D#	G	E
C#	D	C	D#	B	E	Bb	F	A	F#
Bb	B	A	C	G#	C#	G	D	F#	D#
D	D#	C#	E	C	F	B	F#	Bb	G
A	Bb	G#	B	G	C	F#	C#	F	D
D#	E	D	F	C#	F#	C	G	B	G#
G#	A	G	Bb	F#	B	F	C	E	C#
E	F	D#	F#	D	G	C#	G#	C	A
G	G#	F#	A	F	Bb	E	B	D#	C
F	F#	E	G	D#	G#	D	A	C#	Bb
F#	G	F	G#	E	A	D#	Bb	D	B

Figure 7: *The Row Matrix window. To generate a matrix, the user enters pc's into the top row and, optionally, into the first column. The contents of the matrix are automatically updated after every change. CMAP has found all occurrences of segments containing the set {0, 1, 4}. The window is too small to display the entire table, so the user can scroll horizontally.*

Typically, a program, such as getset, will be used to generate sets with a given property. The resulting data will be piped into another program, which processes the data. Its output is then, in turn, processed by a further program, and so on.

This form of data processing can be managed by Macintosh journaling utilities if one simple convention is consistently followed. The convention is that every command automatically selects all the data in its output. This is a common convention, particularly in programs that manipulate tabular data, but it has important repercussions regarding the writing of macros. Quite simply, this convention means that macros can follow the model described for shell scripts. The macro simply has to record the commands processed, pipelines are emulated by virtue of the fact that the data needed by every command used in the macro will have been selected by the previous command.

This solution seems too simple to be effective, and not all shell scripts written for the Unix version of CMAP can be easily emulated by the method sketched above. Too many scripts rely on the vast array of filters provided as utilities with Unix, above and beyond those specific to CMAP. In the end, to be certain that every shell script written for CMAP under Unix could be executed on Macintosh, it would be necessary to write software to emulate all the programs contained in the standard Unix distribution. The possible gains do not justify the effort involved in such a project.

Provision could be made for one group of tasks that shell scripts are commonly used for: looping through all set classes of a given cardinality and looping through all cardinalities. Care was taken in the design of all dialogs where a set class must be specified to offer the user an option to select "All Set Classes of Cardinality *x*." Looping through cardinalities needed a little more flexibility, since the user might want all cardinalities from 1 to 12, or from 3 to 9, or any other range. This led to the invention of a new interface element called the Cardinality Chooser.

The goal is to provide the user with a visible means of selecting any combination of values from 1 to 12. There are already two standard interface elements that could serve this purpose. The user could be presented with a series of twelve checkboxes, or a list of items. However, selecting (or deselecting) multiple cardinalities would be tedious with checkboxes. There is no standard mechanism for selecting multiple checkboxes with one mouse gesture, each one must be individually clicked. There are several standard techniques for selecting multiple objects in a list: clicking with the shift key depressed is used to extend the current selection; dragging the mouse with the command key depressed will add or remove items from the current selection; a mouse click with no modifier key begins a new selection. The repertoire of gestures is reasonably rich, but a long list in the middle of a dialog box looked just ugly.[7]

A horizontal orientation for the cardinalities seemed to fit better into dialog boxes, since these are usually wider than they are high. Placing the cardinalities in two rows provides a more compact and attractive display, but selecting ranges of cardinalities with standard list selection techniques is not as convenient as one might wish. As one example, selecting cardinalities three to nine requires the use of at least two separate mouse gestures: dragging from three to six with the shift key depressed, then dragging from seven to nine with the command key depressed. (Other combinations are possible, but each requires at least two mouse actions.)

Considering the importance of complementary set cardinalities led to the idea of placing complementary values vertically next to each other. However, hexachords are their own complements, and the aggregate's complement, the empty set, is not a valid selection. For this reason, cardinalities six and twelve are displayed at the right of the list, and fill both rows of the Cardinality Chooser. The configuration displayed in the Generate Sets dialog (bottom right of Figure 5) was the result of these deliberations. To select cardinalities three to nine with this configuration, drag with the command key depressed from the three to the six and continue along the lower half of the Cardinality Chooser to the nine. The display looks essentially like a list, and the behavior and repertoire of mouse actions are the same as those provided by standard lists.

The only sure way to evaluate the effectiveness of a new interface element is through user testing. Preliminary user testing has not turned up any difficulty with this variation on list selection. A final evaluation, however, of the Cardinality Chooser will have to await more thorough testing.

A Note on Implementation

All versions of CMAP are conceived as tools for interactive analysis, and this is particularly true of the Macintosh version. To provide a fast and efficient implementation of the various software tools, a database of set class information is stored in an extremely compact form that remains resident in memory. This data base is called the Set Class Table. All sets are stored as bit vectors. This limits memory usage to a minimum and allows for the implementation of particularly efficient algorithms for all operations. The use of bit vectors and a memory resident Set Class Table is common to all CMAP implementations and was first documented in [Harris and Brinkman 1986].[8] In the Macintosh version, a further speed optimization was implemented by building a compact index into the Set Class Table. The cost, in terms of additional memory requirements, is extremely modest, and was balanced by a slightly more space efficient representation of the Set Class Table than the implementation used in the Unix version.

Conclusions

The Macintosh version of CMAP unifies a collection of about thirty separate programs into one coherent package, providing a comprehensive set of tools for set-theoretic analysis. Priority has been given to implementing fast algorithms using compact data representations. A further goal was to provide the same flexibility found in the Unix version under an operation system that did not support scripting.

Bibliography

Alphonce, Bo Harry (1974)
"The Invariance Matrix." Ph.D. diss. Yale University.

Apple Computer Inc. (1987)
Human Interface Guidelines: The Apple Desktop Interface. Reading, MA, Addison-Wesley Publishing Co.

Forte, Allen (1964)
"A Theory of Set-Complexes for Music." *JMT* **8**(2): pp. 136-183.

Forte, Allen (1973)
The Structure of Atonal Music. New Haven, Conn., Yale University Press.

Harris, Craig R. and Aleander R. Brinkman (1986)
"A unified set of software tools for computer-assisted set-theoretic and serial analysis of contemporary music." In *Proc. International Computer Music Conference 1986*, P. Berg, Ed. pp. 331-336. International Computer Music Association.

Rogers, John (1968)
"Some properties of non-duplicating rotational arrays." *PNM* **7**(1): pp. 80-102.

Notes

[1] Strictly speaking, getset will look for all sets that contain each of the six interval classes at lest one time, but with tetrachords this amounts to the same thing.

[2] The Set Complex relations K, Kh, and K* are defined in [Forte 1964; 1973].

[3] The R_p relation, as defined in [Forte 1973], is actually only defined between set classes, whereas it is applied in the setmap program to different representatives of one set class. The extension of the definition made in CMAP matches Forte's adjectival phrase "strongly represented," which is used frequently in his discussion of the R-relations.

[4] This, admittedly extreme, example is taken from the tutorial included in the CMAP documentation.

[5] The name "Set Class Relations" would be a more precise title, but was rejected as too long-winded.

[6] Conventional staff notation will be supported in a future version of CMAP for Macintosh.

[7] Cf. [Apple 1987] for a discussion of the relevance of "Aesthetic Integrity" to software design.

[8] Note that similar techniques had already been used at least as soon as the early seventies. (Cf. [Alphonce 1974])