UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Massachusetts Institute of Technology
Project MAC

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP
None

3. REPORT TITLE
Syntax-Based Analytic Reading of Musical Scores

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Technical Report, Yale School of Music , August 1966

5. AUTHOR(S) *(Last name, first name, initial)*
Forte, Allen

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
April 1967	40	6

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
Office of Naval Research, Nonr-4102(01) b. PROJECT NO. NR 048-189 c. RR 003-09-01 d.	MAC-TR-39 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. AVAILABILITY/LIMITATION NOTICES
Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Advanced Research Projects Agency 3D-200 Pentagon Washington, D. C. 20301

13. ABSTRACT As part of a larger research project in musical structure, a program has been written which "reads" scores encoded in an input language isomorphic to music notation. The program is believed to be the first of its kind. From a small number of parsing rules the program derives complex configurations, each of which is associated with a set of reference points in a numerical representation of a time-continuum. The logical structure of the program is such that all and only the defined classes of events are represented in the output.

Because the basis of the program is syntactic (in the sense that parsing operations are performed on formal structures in the input string), many extensions and refinements can be made without excessive difficulty. The program can be applied to any music which can be represented in the input language. At present, however, it constitutes the first stage in the development of a set of analytic tools for the study of so-called atonal music.

The program and the approach to automatic data-structuring may be of interest to linguists and scholars in other fields concerned with basic studies of complex structures produced by human beings.

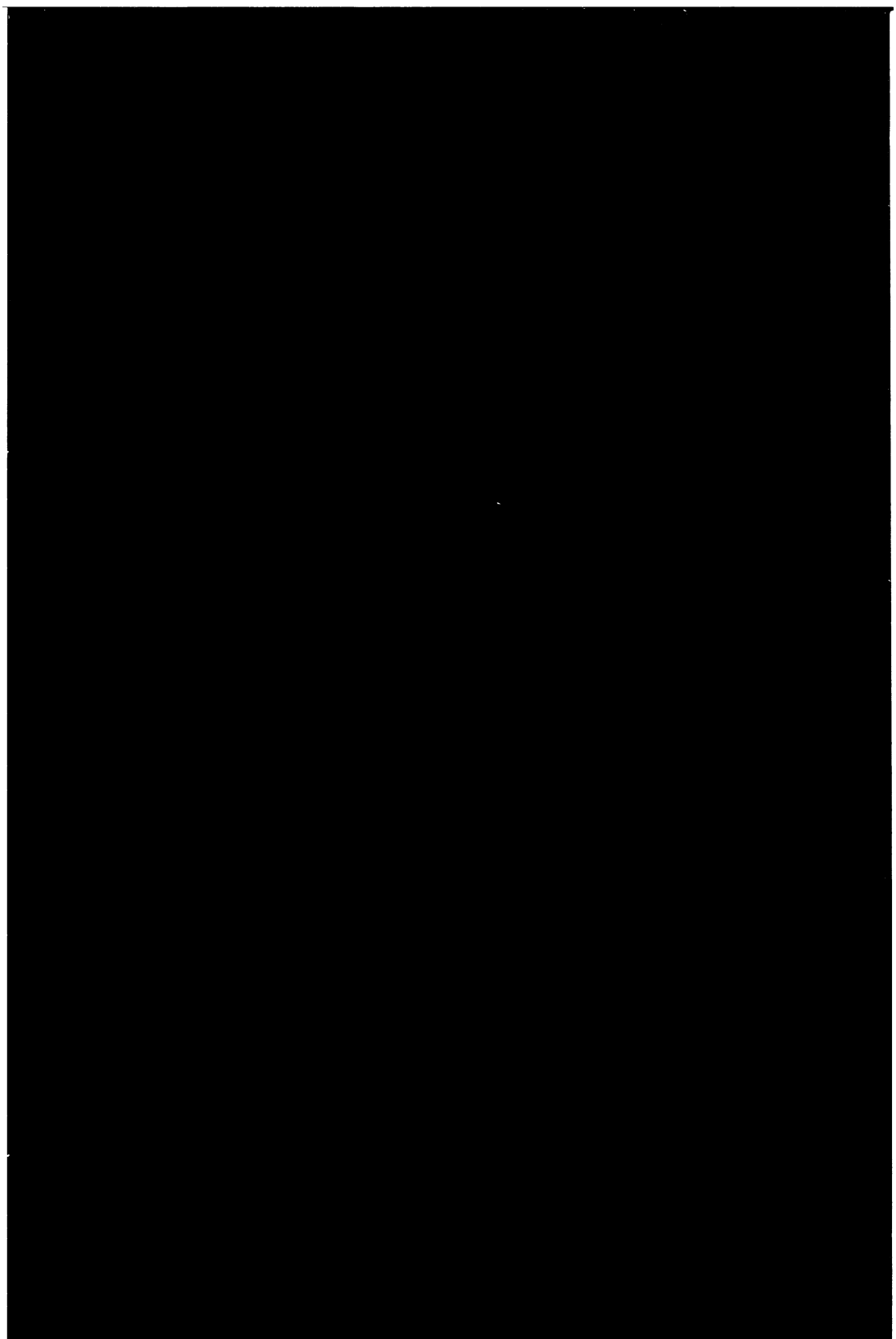
14. KEY WORDS

Automatic data-structuring	Multiple-access computers	Syntax-based analysis
Computers	On-line computers	Time-sharing
Machine-aided cognition	Real-time computers	Time-shared computers

DD , FORM NOV 65 1473 (M.I.T.)

UNCLASSIFIED

Security Classification



MAC-TR-39

SYNTAX-BASED ANALYTIC READING OF MUSICAL SCORES

by

Allen Forte

Project MAC

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ABSTRACT

As part of a larger research project in musical structure, a program has been written which "reads" scores encoded in an input language isomorphic to music notation. The program is believed to be the first of its kind.

From a small number of parsing rules the program derives complex configurations, each of which is associated with a set of reference points in a numerical representation of a time-continuum. The logical structure of the program is such that all and only the defined classes of events are represented in the output.

Because the basis of the program is syntactic (in the sense that parsing operations are performed on formal structures in the input string), many extensions and refinements can be made without excessive difficulty.

The program can be applied to any music which can be represented in the input language. At present, however, it constitutes the first stage in the development of a set of analytic tools for the study of so-called atonal music, the revolutionary and little understood music which has exerted a decisive influence upon contemporary practice of the art.

The program and the approach to automatic data-structuring may be of interest to linguists and scholars in other fields concerned with basic studies of complex structures produced by human beings.

ACKNOWLEDGMENTS

During the academic year 1965-66 the author held a fellowship in the American Council of Learned Societies, which had received a grant from the International Business Machines Corporation to enable a small number of humanist scholars to investigate the use of data-processing techniques in their fields of specialization. Grateful acknowledgement is made to these organizations for their essential assistance.

All work on the project described in the main part of this paper was carried out at Project MAC while the author was a guest of the Massachusetts Institute of Technology. Special thanks are due to Professor Robert M. Fano, Director of Project MAC, and to Professor Marvin L. Minsky, who heads the Artificial Intelligence Group with which the author was associated. Professors Joseph Weizenbaum and Robert McNaughton provided philosophical inspiration and words of encouragement, as did William H. Henneman, Stephen Smoliar, Daniel J. Edwards, Joel Moses, and A. Wayne Slawson.

The music input language was designed by Stefan Bauer-Mengelberg, President of the Mannes College of Music, musician, logician, and partially reformed computer expert. The author is indebted to Mr. Bauer-Mengelberg for his elegant language and for several of the notions which underlie the work described.

For his careful reading of the manuscript and for many helpful suggestions John Rothgeb deserves special mention and thanks.

TABLE OF CONTENTS

Section		Page
ABSTRACT		iii
ACKNOWLEDGEMENTS		iv
LIST OF ILLUSTRATIONS		vi
I	SCOPE OF THE RESEARCH PROJECT	1
II	APPROACH TO A SYNTAX-BASED METHOD	
2.1	The Score as a System of Graphic Signs	3
2.2	Advantages of Automatic Data-Structuring	3
2.3	The Music Input Language	4
2.4	The Programming Language	4
2.5	Output Forms	4
III	DESCRIPTION OF THE PARSING SYSTEM	
3.1	Definitions	7
3.1.1	Terms	7
3.1.2	Symbols	8
3.1.3	Abbreviations	9
3.1.4	Conventions	9
3.2	Preliminary Operations	10
3.3	Parsing Rules	13
3.3.1	General Comments	13
3.3.2	Primary Segments	13
3.3.3	Secondary Segments	16
3.3.4	BCP (SSGT)	19
3.3.5	BCP (PSGT)	21
3.3.6	SCP (SSGT)	23
3.4	Summary: Structure of the Parsing System	25
IV	FROM SCORE READING TO HIGHER ANALYTIC LEVELS	29
V	EXTENSIONS AND IMPLICATIONS	35
REFERENCES		36

LIST OF ILLUSTRATIONS

Figure		Page
1	Four Pieces for Violin and Piano, Op. 7/1—Anton Webern	5
2	Six Bagatelles for String Quartet, Op. 9/5—Anton Webern	11
3	The Terminal String of Primary Segments (PSGT)	15
4	Possible Interactions for Two Segments	17
5	Decision Structure for Secondary Segments	18
6	The Terminal String of Secondary Segments (SSGT)	20
7	Program Segment for Duplicates in SSGT	20
8	Block-Concatenation Parsing of SSGT	22
9	Block-Concatenation Parsing of PSGT	24
10	Segment-Concatenation Parsing of SSGT	26
11	Segment-Concatenation Parsing of PSGT	27
12	Pitch Structure from PV= 160 to PV= 276	30
13	Pitch Structure from PV= 312 to PV= 488	32
14	The Association Between Two Contexts	34
15	Extending the Parsing System	34

SECTION I

SCOPE OF THE RESEARCH PROJECT

During the years 1908-1922 musical composition underwent a remarkable and revolutionary transformation. This radical change was due almost entirely to the innovations of Arnold Schoenberg and his students, Anton Webern and Alban Berg. Among the major compositions of this time were Schoenberg's Pierrot Lunaire (1912), Webern's Six Orchestral Pieces (1910) and Berg's Wozzeck (1919). The term "atonal," commonly used to describe this music, is construed as "not exhibiting any of the structural characteristics of traditional, tonal music." However inadequate the term may be, it does indicate the fact that the structural determinants of this complex music are not well understood. No adequate structural description has yet been put forth which can be subjected to acceptable logical or pragmatic tests. The score-reading system described in this paper is part of a larger research project concerned with the development of such a general structural description.

Certain basic decisions which led to the development of the score-reader are intimately connected with the goal of the larger project outlined above. The choice of atonal music as the object of research brings with it the need for an extensive revision of outlook. For example, conventional descriptive language ("melody," "harmony," "counterpoint") is not very useful, and may even be a significant hindrance to new formulations, since it is oriented toward older music. When, in addition, problems of structural analysis are stated in terms accessible to computer programming, many conventions are set aside and many familiar concepts are rejected after serious scrutiny. These considerations and others, discussed below, led ultimately to the design of the basic tool which is described in this paper.

SECTION II

APPROACH TO A SYNTAX-BASED METHOD

2.1 THE SCORE AS A SYSTEM OF GRAPHIC SIGNS

One can obtain information about a particular composition in a number of ways: for example, by reading statements made by the composer which purport to describe his composing techniques or his aesthetic views, or by asking a trained listener to supply a description of some kind. Such information is always incomplete, however, and does not provide an adequate base for an analytic system. The musical score, on the other hand, constitutes a complete system of graphic signs and, properly represented for computer input, may be analyzed as a logical image of the unfolding musical events which make up the composition. This was the point of departure for the score-reading program: the analysis of formal characteristics of sequences of graphic signs in the music input language discussed below.

2.2 ADVANTAGES OF AUTOMATIC DATA-STRUCTURING

When the trained analyst examines a musical score he associates certain signs to form units and makes a series of basic decisions about the temporal spans of such units and their internal structuring. In a metaphorical sense he places a template over the score to frame patterns and show how they are interwoven to form local contexts.

Although the decisions which the analyst makes may rest upon years of practical experience, they are often unsystematic and subject to many influences which are not easily identified. Nevertheless, it is apparent that at least part of the time he makes decisions according to rules of some kind. This suggests the possibility that rules could be stated. Such rules, interpreted as algorithms and stated in a programming language, would yield a complete and precise structuring of the data presented by the score. A tool of this kind should produce results at least as good as those produced by the human analyst. It should not give provisional readings that require further hand-editing or interpretation of some kind. Indeed, it would be appropriate to aim for a reader which would

produce results superior to those obtained by the human analyst. The use of the expression "human analyst" in this context is not intended to imply that the programmed score reading process is based upon methods commonly used by human beings. The logical base of the program and its empirical relevance to a fundamental musical parameter, time, will become evident as we proceed.

2.3 THE MUSIC INPUT LANGUAGE

Before processing, scores are encoded by hand in a language designed by Stefan Bauer-Mengelberg [1] in connection with a computer-implemented project to print scores automatically. This input language has unique attributes:

(1) It is isomorphic to standard music notation.
(2) It is highly mnemonic, hence easy to learn.
(3) The responsibilities of the encoder are minimal. The encoding rules are unambiguous and do not require the encoder to make arbitrary decisions.
(4) It is economical. The amount of code for a complete representation is remarkably small. (See Figure 1.)

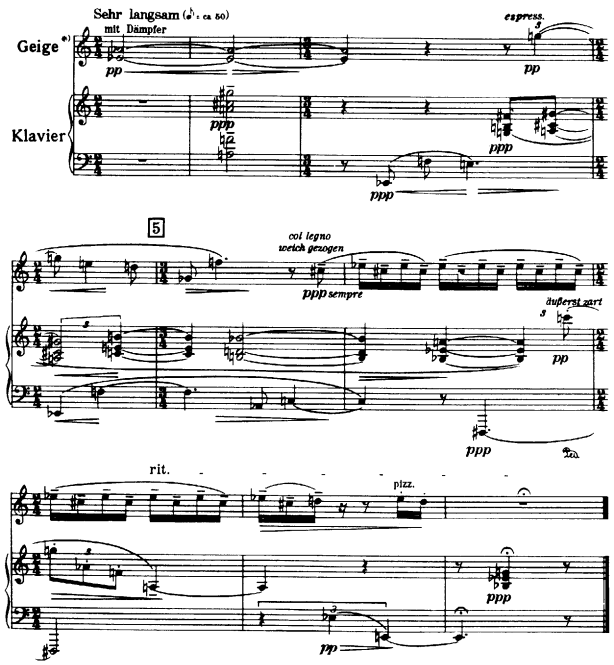
The language uses a currently non-standard character set of 64 graphics [2], but a set of equivalent codes is available when the expanded set cannot be obtained. Figure 1 shows a page of a modern score and the corresponding code in Bauer-Mengelberg's language, prepared for input to the score-reading program.

2.4 THE PROGRAMMING LANGUAGE

The high level programming language SNOBOL3, designed by Farber, Griswold, and Polonsky [3], is used for the program. This language handles free strings with facility, and its generality renders it most appropriate for the kind of experimentation which necessarily accompanies a research project of this kind. The score reading program itself consists of a number of SNOBOL-encoded functions. A small amount of basic information is stored in lists, but the power of the program resides in the functions. These cope with any eventuality in the input string, including encoding errors.

2.5 OUTPUT FORMS

The current forms are provisional. An appropriate syntactic structure for output will be developed as the reading process is tested and refined. In all



Copyright 1924 by Universal Edition
Renewed copyright 1952 by Anton Webern's Erben
Used by permission of Theodore Presser Company

INPUT LANGUAGE CODE

```
WEBERN,FOUR PIECES FOR VIOLIN AND PIANO, OP. 7 (1910)
2.101*VIOLIN,/102*PIANO,/                                              EOF
101 *XG,*XT*$SEHR LANGSAM (E=CA 50)$ *XM2*.4 VPP*L1,21-HJ,24-N4J,*AMIT *
$DAEMPFER*A / V*L1*G2,21HJ,24N4J *G2 / *XM3*.4 21Q RQ *X3E1*.2 RE1,*XVES
PRESS.$ VPP,30**Q1JL+1 / *XM2*.4 102 *XG *XM2*.4 RW / VPPP,24**H,26**H,3
0*=H*- / *XM3*.4 RQ RQ (VPPP,16**E,18**E,22*=EL+1 17**EJ,19*=EJ,23*=EJ)
/ *XM2*.4 ** *XF *XM2*.4 RW / 79**H,84**H*- / *XM3*.4 RE VPPP*L1,69-EL+1
 77**E V*L1*G2,76**Q.L+1 V*G2 / *XM2*.4 101 *XG *XM2*.4 V*L1,30**E 29**Q
 V*L1 27**E / *XM3*.4 V*G1,23-E 29**Q.L+1 V*G1 RE,*XVWEICH GEZOGEN$,*XVC
OL LEGNO, VPPPSEMPRE,26*=EL+1*- / ((28-S*- 26*=S*- 28S*- 26SL+1*-)) ((28
SL+1*- 26S*- 28S*- 26S*-)) ((28S*- 26S*- 28S*- 26S*-)) / *XM2*.4 102 *XG
 *XM2*.4 *X3Q2*.2 V*L2,17**H2,19**H2,23*=H2 V*L2,19**Q2J,21**Q2J,25**Q2J
 / *XM3*.4 19Q,21Q,25Q 18**HJ,20**HJ,25-HJ / V*G2,18Q,20Q,25Q V*G2,18-QJ
,21-QJ,24**QJ 18Q1,21Q1,24Q1L+1 VPP,33**E1L+1Q,*XVAEUSZERST ZART$ / *XM2
*.4 ** *XF *XM2*.4 V*L2,69-QL+1 V*L2,77**QJ / *XM3*.4 77Q. V*G2 72-E 74*
*QJ V*G2 / 74L+1 RE VPPP,63*=Q.L+1 / *XM2*.4 101 *XG *XM2*.4 ((28-SL+1*-
 26*=S*- 28S*- 26S*-)) ((28S*-,*XTRIT.$ 26S*- 28S*- 26SL+1*-)) / ((V*G1,
28-SL+1*- 26*=S*- 27**SL+1*-)) RS RE ((28SO V*G1,27SO)) / RW // 102 *XG
*XM2*.4 (V*G2,30**E10 24-E10 21**E10) V*G2,17**QJL+1*- / 17Q RQ RE VPPP,
18-Q,21-Q,23**Q RE // ** *XF *XM2*.4 63*=H / RQ2 VPP*G2,76-Q2L+1 V*G2,19
**Q2JL+1 / 19Q. RE //                                                  EOF
```

Figure 1. Four Pieces for Violin and Piano, Op. 7/1 – Anton Webern

likelihood the output forms will ultimately employ a subset of the input language, since this offers the additional advantage of automatic printing of the results of an analysis.

In the early phases of processing, the forms of the output (intermediate) strings are constrained by the syntax of the input language because of the relation between input and output strings. The nature of the relation will be made clear in the next section, which begins with a summary of terms, conventions, symbols, and abbreviations used to describe the parsing process.

SECTION III

DESCRIPTION OF THE PARSING SYSTEM

3.1 DEFINITIONS

The following definitions are adopted for use in describing the parsing system.

3.1.1 Terms

Character
: Any of the standard characters in six-bit mode. (The extra character ":" is used in the intermediate strings.)

Space-Code
: A sequence of characters in the input language which contributes to the specification of pitch.

Accidental-Code
: A sequence of characters in the input language which, like a space-code, contributes to the specification of pitch.

Pitch-Code
: A space-code followed by an accidental-code.

Element
: A sequence of characters in the input language representing a distinct symbol in music notation. For example, "." represents the dot of music notation.

String
: A sequence of characters which has markers to delimit beginning and end.

Data String
: The original data before processing: a string in the input language.

Input String
: A string of characters which is scanned to form a new string. The data string is the first input string.

Intermediate String
: A string, constructed by the program from an input string, which represents an intermediate stage in a parsing operation.

Output String
: A processed string.

Position-Value
An integer in an intermediate or output string which specifies the temporal location of an element or sequence of elements.

Segment
A distinct unit within an output string, delimited by "(" on the left and ")" on the right.

Block
A distinct unit within a segment, delimited in the output string by "P" on the left and a single blank on the right. The corresponding unit in the input language is delimited by blanks on left and right.

Parsing
The extraction, from an input string, of strings of segments representing classes of event-configurations in the score.

Terminal String
The final output form for a class of segments.

3.1.2 Symbols

In the definitions which follow, "A" and "B" name arbitrary intermediate strings, and the defined infixes denote relations between, or operations upon, pairs of such strings.

$A \rightarrow B$	A is replaced by B.
AB	Concatenation
$A \cap B$	The segment set shared by output strings A and B.
$A \equiv B$	A has the same unordered contents as B.
$A \subseteq B$	A is a substring (proper or improper) of B.
$A \supseteq B$	A is a superstring (proper or improper) of B.

3.1.3 Abbreviations

Special Classes of Strings

SG	Segment
PSGI	The string of primary segments for input.
PSGT	The terminal string of primary segments.
SSGI	The input string of secondary segments.
SSGT	The terminal string of secondary segments.
RSSGT	A replica of SSGT.
RPSGT	A replica of PSGT.

Programmed Operations on Strings

BCP	Block-concatenation parsing operation.
SCP	Segment-concatenation parsing operation.

Grammatical Categories

These are used for input, intermediate, or terminal strings. The abbreviations which follow will be used for describing the form of strings according to conventions given below:

AC	Accidental-Code
TC	Time-Code
SC	Space-Code
PC	Pitch-class, represented by an integer in the set $\{0,1,2, \ldots, 11\}$. The expression "pitch-class" or the abbreviation PC (after Babbitt [4]) is usually used in the sense "pitch-class representative". The meaning is clear from the context.
PV	Position-value
RC	Rest-code, indicating a silence.
ƀ	A (literal) blank.

3.1.4 Conventions

Names of strings, segments, blocks, functions, function arguments, and indices are in upper case alphabetic characters. Usage conforms to the rules of the programming language:

(1) Indices are separated from generic names by "."
(2) Literals are enclosed within single quotation marks (' ').
(3) Function arguments are enclosed within parentheses, and multiple arguments are separated by commas.

Examples:

A	String name A.
A.I	String name with (variable) index I.
A.'1'	String name A with index 1.
B(X,Y)	Function name B with arguments X and Y.

Variables used as indices, such as I and J, do not imply anything about sequence, but merely indicate distinctness.

The Grammatical Categories listed previously are used in this paper to specify the form of a string as follows: If X and Y are grammatical categories, then the expression

XY

designates the result of writing, in the input language, a representative of the grammatical category X followed immediately by a representative of Y, with no intervening space. Occasionally literals are used where the string form described contains invariant items; for example, the expression

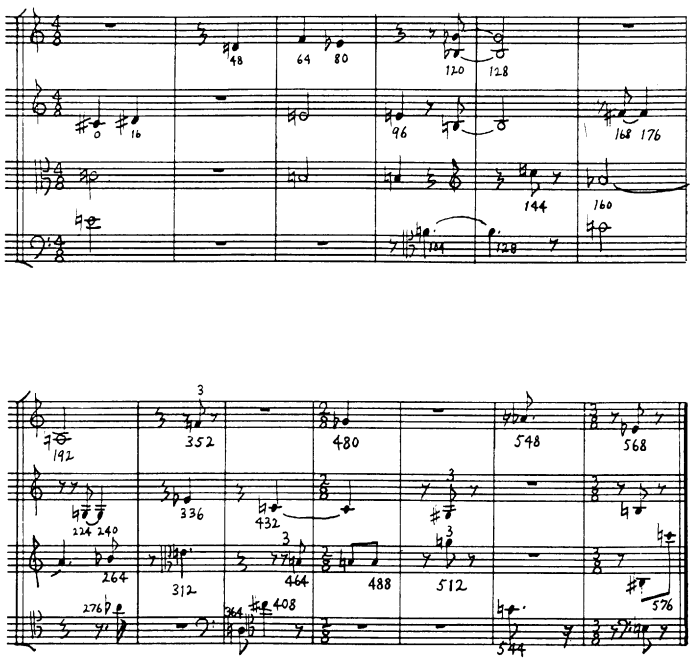
PV '$'

displays the form of a string consisting of some PV followed immediately by the literal dollar sign.

3.2 PRELIMINARY OPERATIONS

For the remainder of the paper a single composition will be used to illustrate the work done by the program. This is shown in Figure 2, together with the corresponding data string encoded in the input language. Two basic operations are performed on the data string before the main parsing begins.

(1) Since the encoding is basically a one-pass procedure, the encoder progressing from left to right across the page for each instrumental part, the characters which belong to a particular instrument are distributed throughout the data string. The program therefore begins by scanning the data string to assemble the complete code-sequence for each instrument. It must deal with such situations as the following: In an orchestral score of 200 measures the bass clarinet enters on the last beat of bar 75, plays a staccato thirty-second note, and is not heard from again during the remainder of the piece. In this and all such cases where the part is not present either at the beginning or end of the score, the program finds the complete instrumental part.

(2) The second operation establishes the basis for the derivation of complex event-configurations by the application of the parsing rules. This consists of the assignment of a position-value PV to each code which implies a temporal position. Two passes through the input string are required. On the first pass information relevant to the duration-value of time-codes is accumulated. In particular, a normative or list value is assumed for each time-code. This value is adjusted

Copyright 1922 by Universal Edition
Renewed copyright 1950 by Anton Webern's Erben
Used by permission of Theodore Presser Company

INPUT LANGUAGE CODE

```
WEBERN, SIX BAGATELLES FOR STRING QUARTET, OP.9/5 (1913)
4,I01*FIRST VIOLIN,/I02*SECOND VIOLIN,/I03*VIOLA,/I04*CELLO,/
I01 *XG *XM4*.8 RW / RQ 20**Q / 22**Q 21-Q / RQ RE 18-EJ,23-EJ / I02 *XG
 *XM4*.8 19*=Q 20*=Q / RW / 21**H / 21**Q RE 18**EJ / I03 25*XC *XM4*.8
25**H / RW / 24**H / 24**Q RQ *XG / I04 *XF *XM4*.8 33**H / RW / RW / RE
 27*XC 30**Q.J / I01 18-H,23-H / RW / 17**H / RQ *X3E1*.2,RE1 22**E1 RE1
 / I02 18**H / RE 22*=EJ 22Q / RE1 RE1 16**E1J 16Q / RQ 21-Q / I03 RQ 26
**E RE / 24-HJ / 24Q. 25-E / RE 25*XC 27**Q. / I04 30**Q. RE / 31**H / R
Q RE. 32-S / RW *XF / I01 RW / *XM2*.8 23-Q / RW / RS 24-E. / *XM3*.8 RE
 21-E RE // I02 RQ 19**QJ / *XM2*.8 19Q / RE1 16*=E1 RE1 / RW / *XM3*.8
RE 18**E RE // I03 RQ RE1 RE1 24**E1 / *XM2*.8 (24**E 24E) / RE1 30**E1
RE1 / RW / *XM3*.8 RE (18*=E 33**E) // I04 25**E 27*XC 34*=Q RE / *XM2*.
```

Figure 2. Six Bagatelles for String Quartet, Op. 9/5—Anton Webern

if a groupette occurs in the same bar. For example, if the meter signature is 2/4, an eighth note triplet is a groupette and indicates three eighth notes in the (normative) time of two. For each bar a number is computed which represents the product of all groupette divisors. The number is one if there are no groupettes in any part in a particular bar. On the second pass a duration-value is assigned to each time-code by consulting the list of normative values and the number which represents the product of the groupette divisors. The duration-value DURVL for groupette members is them computed by the SNOBOL statement

$$DURVL = (VL * GP2) * (Z/GP1)$$

where VL is the list value of the time-code, GP2 is the normative division of that value, Z is the groupette product, and GP1 is the groupette divisor. For all other time-codes in a bar which contains a groupette or groupettes,

$$DURVL = VL * Z$$

For bars in which groupettes do not occur, the list value is the duration-value assigned.

Once the position-values have been assigned — and this is straightforward after the duration values have been computed — position-value becomes the main parameter in the parsing since it indicates temporal position. The string can then be viewed as a succession of blocks within segments, each block containing the total of codes associated with the point in time represented by the position value. In this way it is possible to refer to any moment in a composition and to relate any event to any other event with respect to a time-continuum. In this regard as well as many others, the analytic technique differs fundamentally from those employed by researchers in other fields where the string is the basic data structure. The structural linguists, for example, are not concerned with time-placement in this extended sense, whereas for music research the temporal dimension is always prime [5].

The association of each event and each event-configuration with a set of position-values has the additional advantage that order characteristics are preserved. Thus, complexities of rhythmic interaction can be deduced from the set of numbers attached to particular configurations. For example, collation functions

can be written to search for categories which might be of special interest, such as all segments which have the same first position value.

3.3 PARSING RULES

3.3.1 General Comments

For the description of the parsings, information falls conveniently into these categories:

(1) Names of the strings being parsed.
(2) An informal description of the parsing technique and a characterization of the event-configurations which it extracts from the input string.
(3) A symbolic expression representing the form of the segment in the output string.
(4) A formal statement of the parsing rules.
(5) A description of the deletion conditions for segments in the output string.
(6) Illustrations of program-generated output, music notation, and scanning strategy.

3.3.2 Primary Segments

Underlying any segmentation or parsing process are certain assumptions regarding the discreteness of events. What conditions determine a unit? Which classes of signs serve as disconnectives? How can one speak with precision of components of larger event-configurations? Questions such as these are in the foreground of the description of parsing rules, and an effort is made to state the answers clearly.

For the first parsing of the input string, each substring representing a complete instrumental part is scanned to extract all primary segments. A primary segment is a continuous unit delimited by the graphic sign denoting silence: a rest in musical notation, RC in the input language. It begins with a PC which has been preceded by a rest (or by an understood [logical] rest if the pitch is the first in a part), contains no internal rests, and ends with a rest. Thus, it is set off from other substrings and is regarded as a disjunct sequence in a one-dimensional environment. The scan for primary segments in a instrumental part terminates when the input language code "//" (double bar) is encountered.

In the input language, a primary segment has the form

RC ∅ SC AC TC SG ∅ RC ∅ (1)

With respect to sequence in the string the block SC AC TC always follows the RC. Since there may be more than one RC in a substring of the input string without an intervening SC AC TC—for example

RC RC RC SC AC TC SG RC RC RC

—the first RC in the form given above is always the rightmost with respect to AC, while the second RC shown in the form is always the leftmost with respect to SG.

After parsing, the form of the segment (now in an intermediate string) is

'*' PC 'KA' PV '$' ... 'KZ' PV '$' (2)

where 'KA' and 'KZ' designate attack and release, respectively, and refer to the position-value which follows. In the case of 'KA' the position-value always refers to the first (attack) PC. In the case of 'KZ' the position-value refers either to the terminal delimiter (some RC) or, if the next code is '//', to some code which is an understood (logical) RC. See the viola part of Figure 2 for the latter.

The form given in (2) above is the input form for a primary segment. The terminal form is:

'(' 'P' 'V' '*' PC '*' PC . . . '*' PC ')' (3)

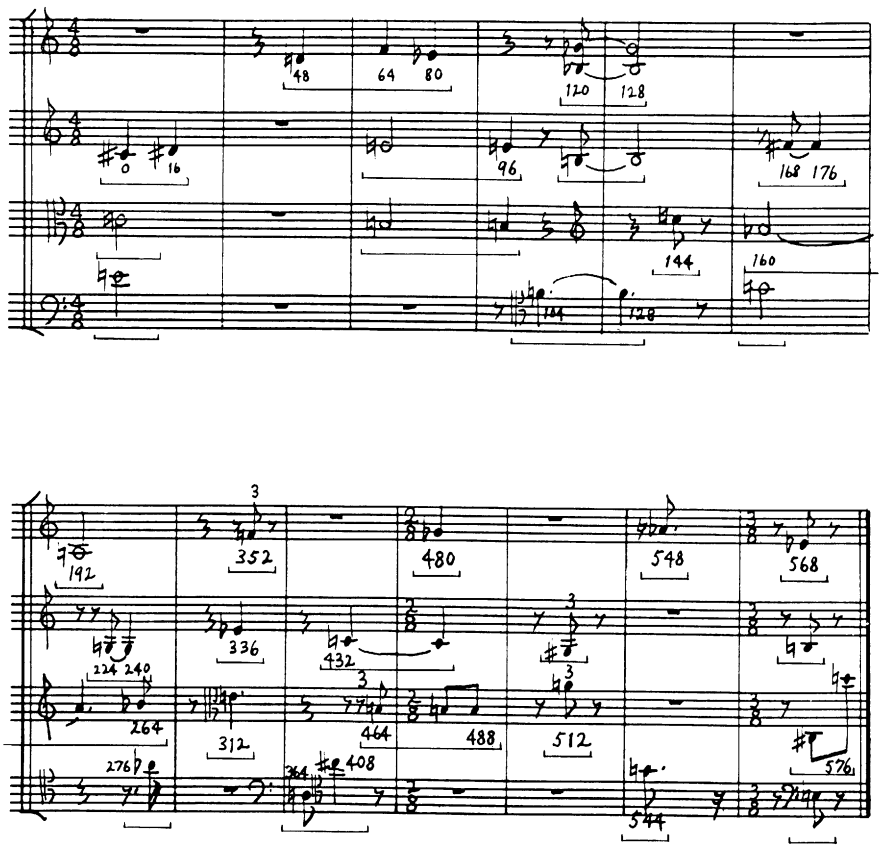
The transformation from input form to terminal form involves two kinds of change: deletions, and deletions with replacements. Characters which are deleted (without replacement) are 'KA', 'KZ', ∅, PV (when preceded by 'KZ'), groupette definers, and meter signatures. Each pitch-code is replaced by an integer representing a pitch-class PC. The appropriate mapping is executed by a function PCMAP, which utilizes modulo 7 arithmetic. A number of contextual details require attention here. First, if the instrument is a transposing instrument, the appropriate modulo 12 arithmetic must be performed. Second, a scan is made for space-codes which have implicit accidentals, under two familiar conventions of notation, and those accidentals are inserted. Third, the current clef sign must be made available for each pitch-class mapping. Figure 3 shows the

(P0*1 P16*3)(P0*0)(P0*4)(P48*2 P64*5 P80*3)(P64*4 P96*4)(P64*11 P96 *11)(P104*5 P128*5)(P120*6*10 P128*6*10)(P120*11 P128*11)(P144*0)(P 160*8 P192*8 P264*10)(P160*7)(P168*6 P176*6)(P192*9)(P224*7 P240*7) (P276*8)(P312*4)(P336*3)(P352*5)(P384*2 P408*1)(P432*0 P480*0)(P46 4*11 P480*11 P488*11)(P480*6)(P512*8)(P512*9)(P544*7)(P548*8)(P568 *3)(P568*11)(P568*1 P576*2)(P568*4)

Figure 3. The Terminal String of Primary Segments (PSGT)

complete string PSGT, together with the music notation where primary segments (31 in all) are indicated by parentheses.

3.3.3 Secondary Segments

The string of primary segments is made up of substrings, one for each instrumental part. Every (unordered) pair of these substrings is scanned for secondary segments. This is the first and most important step in the progression from the one-dimensional primary segments to a representation of the analyzed two-dimensional score. At this stage we are concerned with the ways in which events represented by pairs of primary segments interact in local contexts. Indeed, one of the important consequences of the parsing for secondary segments is to show how the primary segments are affected by motion in other parts of the context. The following combinatory situation provides the basis for a formal description of the parsing.

Given three temporal relations: less than (before), equal to (coinciding), and greater than (after), together with the position-values of the attack (A) and release (R) for each of two primary segments SG.I and SG.J (where SG.I $\subseteq$ PSGT.I and SG.J $\subseteq$ PSGT.J), there are $(3^2 + 4) - 2$ ways in which the two segments interact, excluding the non-contiguous cases. Figure 4 shows the 11 possible interactions reduced to 6 classes by virtue of the symmetry of the defined relations. Each class is assigned a mnemonic name in quasi-Polish notation; for example, EAR means that the relation equal to obtains between both A's and both R's, while LAER means that the first A is less than (before) the second, while the two R's coincide. The same name is applied to the second (symmetrical) relation in each class and a subscript is used to distinguish the two.

As indicated above, a given pair of primary segments is scanned (left to right) for A1, R1, A2, and R2. Figure 5 is a graph of the decision structure which is traversed to determine the particular relation between the pair.

When the relation has been determined, one of four functions is called and secondary segments are formed as shown by the arrows in Figure 4. Observe that the basis of segmentation is attack and release operating reciprocally on the constituent primary segments. Since the disconnectives (attack and release

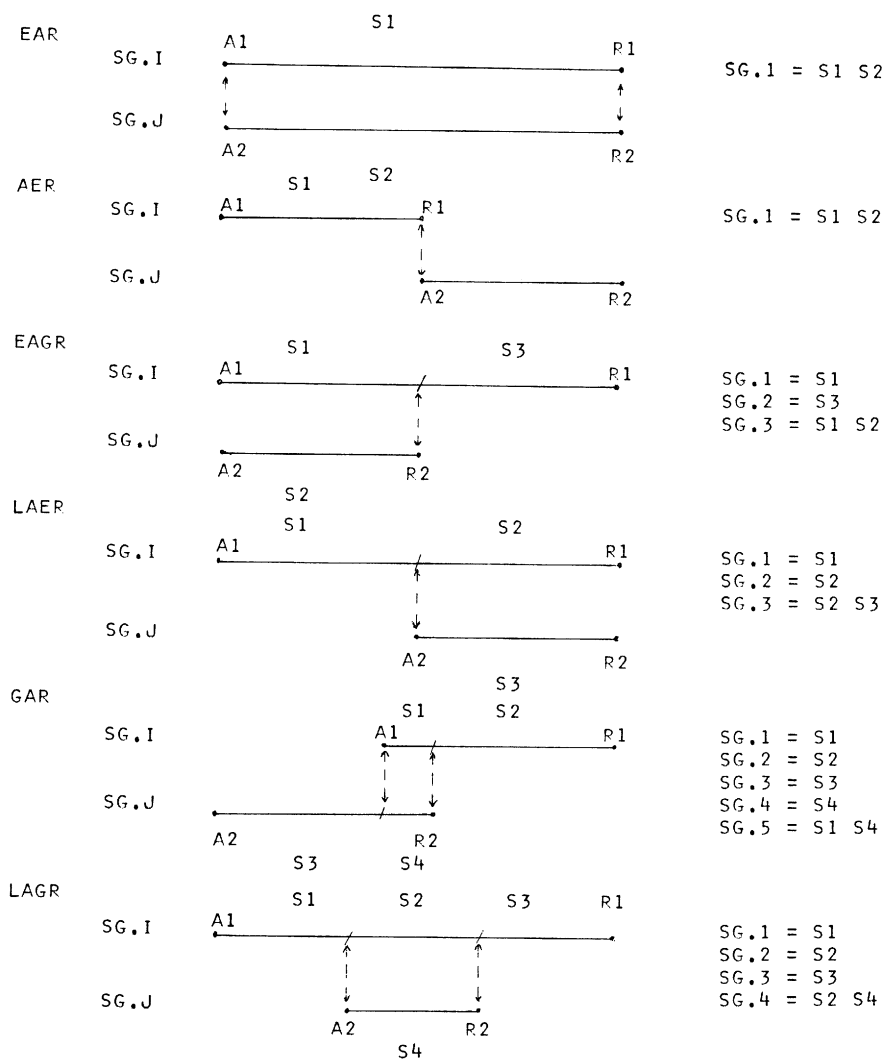


Figure 4. Possible Interactions for Two Segments

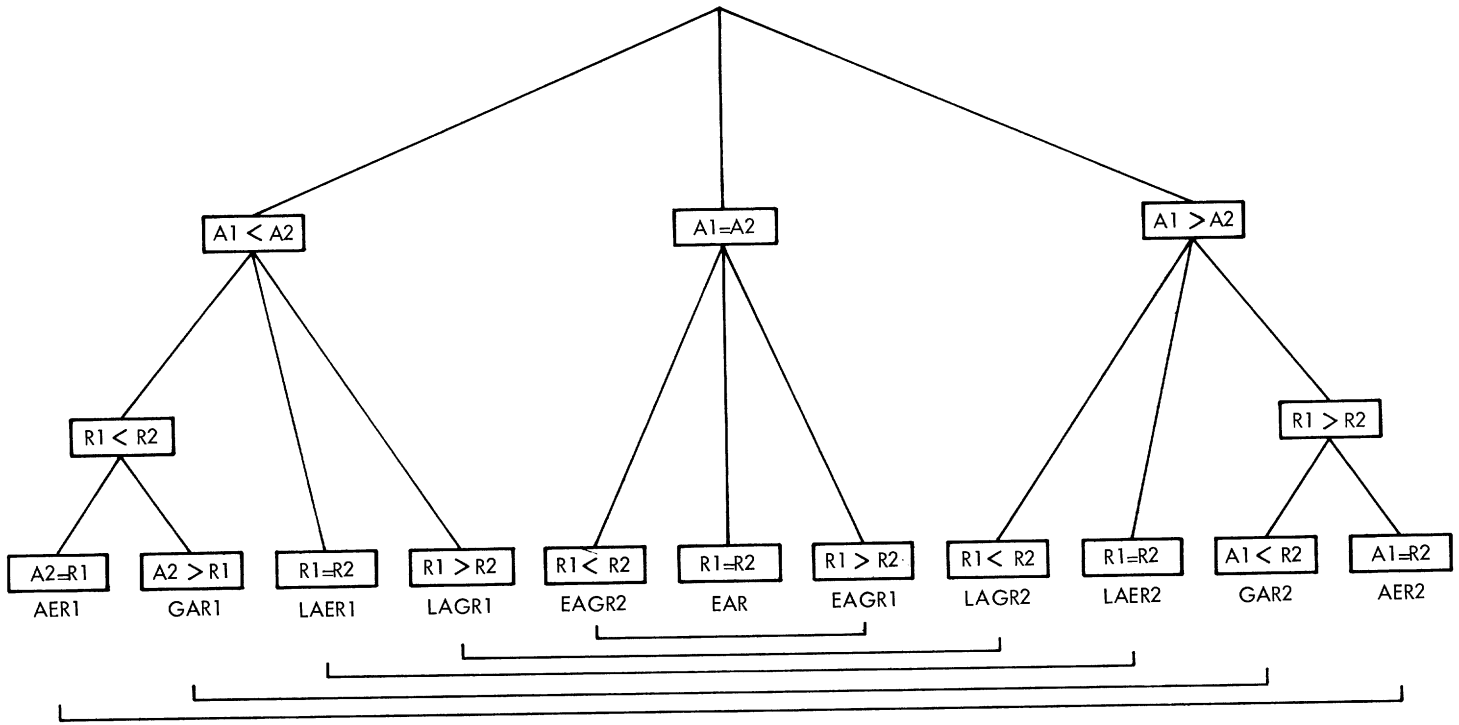


Figure 5. Decision Structure for Secondary Segments

markers) of the original primary segments now themselves point to other disconnectives, it is evident that the parsing is recursive in the limited respect.

The scanning strategy is straightforward: If k is the number of substrings in the string of primary segments—each substring representing an instrumental part—then the number of scans required to find all secondary segments is the same as the number of distinct pairs in the unordered cross-product of a set of k elements

$$(k(k+1)/2)-k = k(k-1)/2$$

Each scan of a pair of strings of primary segments yields a string of secondary segments (possibly null). The name of this string is SSGI.JK, where the variable index J is the index of one member of the pair of primary segments, and K is the index of the other. Figure 6 shows the complete output string SSGT.

The scanning strategy for the deletion of duplicate secondary segments utilizes the fact that for any pair of strings of secondary segments SSGI.J and SSGI.K, where J and K represent concatenated indices of primary segments as described above, duplicate segments will occur only if one element of the compound index J is the same as an element of the compound index K. This strategy greatly reduces the number of scans required and permits the operation to be controlled by a non-arithmetic program segment in a fashion idiomatic to SNOBOL. The program segment which performs this operation is shown in Figure 7.

3.3.4 BCP (SSGT)

A secondary segment results from the interaction in time of the attacks and releases of two primary segments. Possibly a particular secondary segment is a constituent of a still larger event-configuration; that is, it may be associated with other events in the local context. The parsing function BCP (block concatenation parsing) finds such situations and derives a new class of segments.

BCP operates on SSGT and its replica RSSGT as follows. Each block in each segment of SSGT and RSSGT has the form

'P' PV '*' PC. '1' '*' PC. '2' '*' . . . PC.N ∅

```
(P0*0*1 P16*3 )(P0*1*4 P16*3 )(P0*0*4 )(P48*2 )(P64*5 P80*3 )(P64*4 )(P6
4*4*5 P80*3 )(P64*11 )(P64*5*11 P80*3 )(P64*4*11 P96*4*11 )(P64*4 P96*4
P104*5 )(P64*11 P96*11 P104*5 )(P96*4 )(P96*11 )(P104*5 )(P104*5 P128*5
P144*0 )(P120*6*10*11 P128*6*10*11 )(P120*6*10 P128*6*10 P144*0 )(P120*6
*10 P128*6*10 P160*8 P192*8 P264*10 )(P120*6*10 P128*5*6*10 )(P120*6*10
P128*6*10 P160*7 )(P120*11 P128*11 P144*0 )(P120*11 P128*11 P160*8 P192*
8 P264*10 )(P120*11 P128*5*11 )(P120*11 P128*11 P160*7 )(P128*5 )(P160*8
 )(P160*7 P192*9 )(P160*8 P168*6 P176*6 )(P160*8 P192*8 )(P160*7 P168*6
P176*6 )(P160*7*8 )(P160*8 P192*8 P264*10 P276*8 )(P168*6 P176*6 P192*9
)(P192*9 P224*7 P240*7 )(P192*8 P264*10 )(P192*8*9 P264*10 )(P192*9 P276
*8 )(P224*7 P240*7 P264*10 )(P224*7 P240*7 P276*8 )(P264*10 )(P312*4 P35
2*5 )(P312*4 P336*3 )(P312*4 P384*2 P408*1 )(P336*3 P352*5 )(P336*3 P384
*2 P408*1 )(P384*2 P408*1 P432*0 )(P432*0 )(P464*11 )(P464*11 P480*0*11
P488*11 )(P480*0 )(P480*0*6 )(P480*11 P488*11 )(P480*6*11 P488*11 )(P512
*8*9 )(P544*7 P548*8 )(P568*3*11 )(P568*1 )(P568*1*3 )(P568*3*4 )(P568*1
*11 )(P568*4*11 )(P568*1*4 )(P576*2 )
```

Figure 6. The Terminal String of Secondary Segments (SSGT)

```
VV      SNM   *N1/-1-* *N2/-1-* -,- =       /F(WRTE)
        HNM = SNM
V0      HNM   N1 *N3/-1-* -,- =             /S(V1)
        HNM   N2 *N3/-1-* -,- =             /S(V2)
        HNM   *N3/-1-* N2 -,- =             /F(V4)S(V3)
V1      $(-SG- N1 N3) = DP2($(-SG- N1 N2),$(-SG- N1 N3))  /(V0)
V2      $(-SG- N2 N3) = DP2($(-SG- N1 N2),$(-SG- N2 N3))  /(V0)
V3      $(-SG- N3 N2) = DP2($(-SG- N1 N2),$(-SG- N3 N2))  /(V0)
V4      $(-SG- N1 N2) = DP2(PSGT,$(-SG- N1 N2))
        SSGT = SSGT $(-SG- N1 N2)           /(VV)
```

*The character ' is represented by -
in the example.

Figure 7. Program Segment for Duplicates in SSGT

For every block B.I in a segment SG.I (SG.I $\subseteq$ SSGT), with PV.I, and every block B.J in SG.J (SG.J $\subseteq$ RSSGT), with PV.J,

$$\begin{array}{l}\text{if PV.I} = \text{PV.J} \\ \text{then B.I} \rightarrow \text{B.I B.J}\end{array}$$

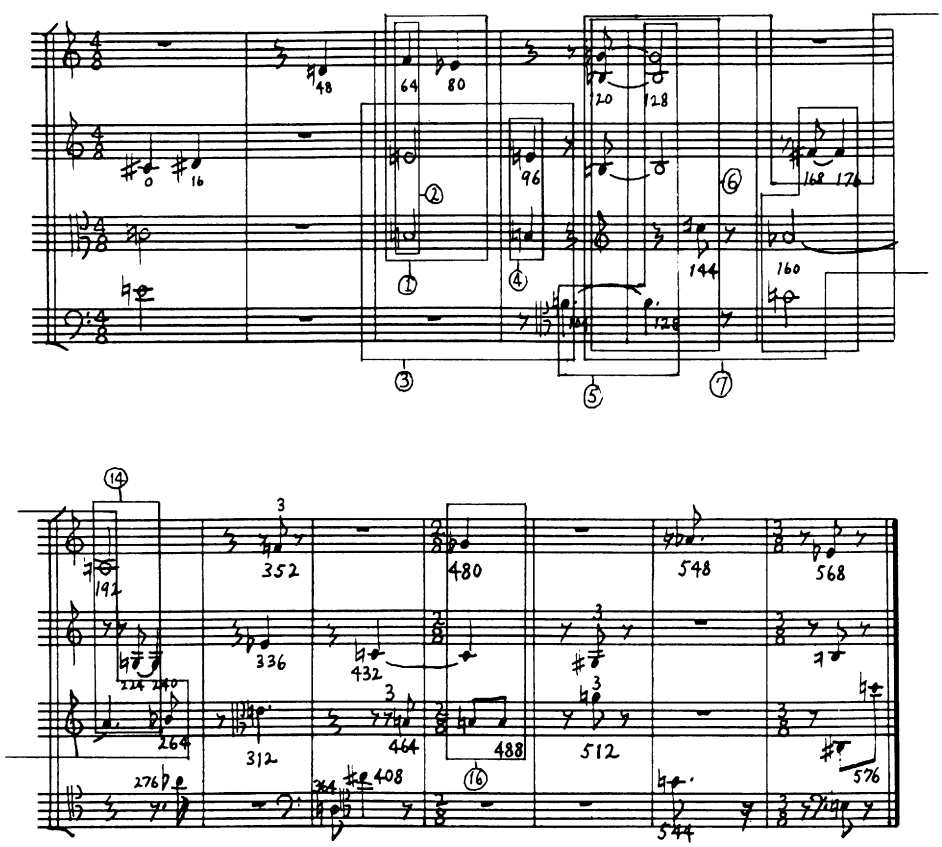
In this way, a secondary segment SG.I is expanded by infixing all blocks in all other segments which intersect in time with some block in SG.I. Two additional aspects should be noted. First, concatenation is followed by the deletion of duplicate PC's. Second, the process is not recursive, in the sense that only the original blocks in a segment are operated on; concatenation does not extend to blocks derived from the parsing.

Possibly some secondary segment does not contain any PV block which intersects with the PV of some other block in the way defined. In this case the segment is complete in itself and does not belong to a larger context. Such residual segments are deleted when the execution of the function has been completed. It is also possible that a segment in the parsed string will have one or more duplicates. These internal duplicates are deleted after BCP has been executed.

Because the concatenation does not take into consideration the order of position-values or the order of pitch-class representatives, it is necessary to call collation functions within BCP. Pitch-classes are placed in ascending order at present. However, the set of indices representing original ordering could be retained, wherever that ordering is significant. Figure 8 shows the output string BCP(SSGT) and the corresponding notation, with segments delimited by frames.

3.3.5 BCP(PSGT)

When BCP operates on PSGT the result is a string each component of which is anchored to a position-value which occurs in two or more instruments. It should be remarked here that the "instrumental part" is merely a device. Once the position-values have been assigned, the location of an element within a particular part is of no fundamental concern. The reading thus is independent of the notions of "part," "voice," "voice-leading" and other interpretive formulations which belong to a higher analytic level.



(P64*4*5*11 P80*3)(P64*4*5*11)(P64*4*5*11 P96*4*11 P104*5)(P96*4*11)
(P104*5 P128*5*6*10*11 P144*0)(P120*6*10*11 P128*5*6*10*11 P144*0)(P12
0*6*10*11 P128*5*6*10*11 P160*7*8 P192*8*9 P264*10)(P120*6*10*11 P128*5
*6*10*11 P160*7*8)(P128*5*6*10*11)(P160*7*8 P192*8*9)(P160*7*8 P168*6
P176*6)(P160*7*8 P192*8*9 P264*10 P276*8)(P168*6 P176*6 P192*8*9)(P1
92*8*9 P224*7 P240*7)(P192*8*9 P276*8)(P480*0*6*11 P488*11)

Figure 8. Block-Concatenation Parsing of SSGT

Deletion conditions for BCP(PSGT) are as follows: residual primary segments are deleted, and internal duplicates are deleted. Since the rules for BCP when applied to PSGT may form duplicates of secondary segments, a scan must be made to delete such duplicates. Figure 9 shows the complete output string BCP(PSGT) and corresponding notation.

3.3.6 SCP(SSGT)

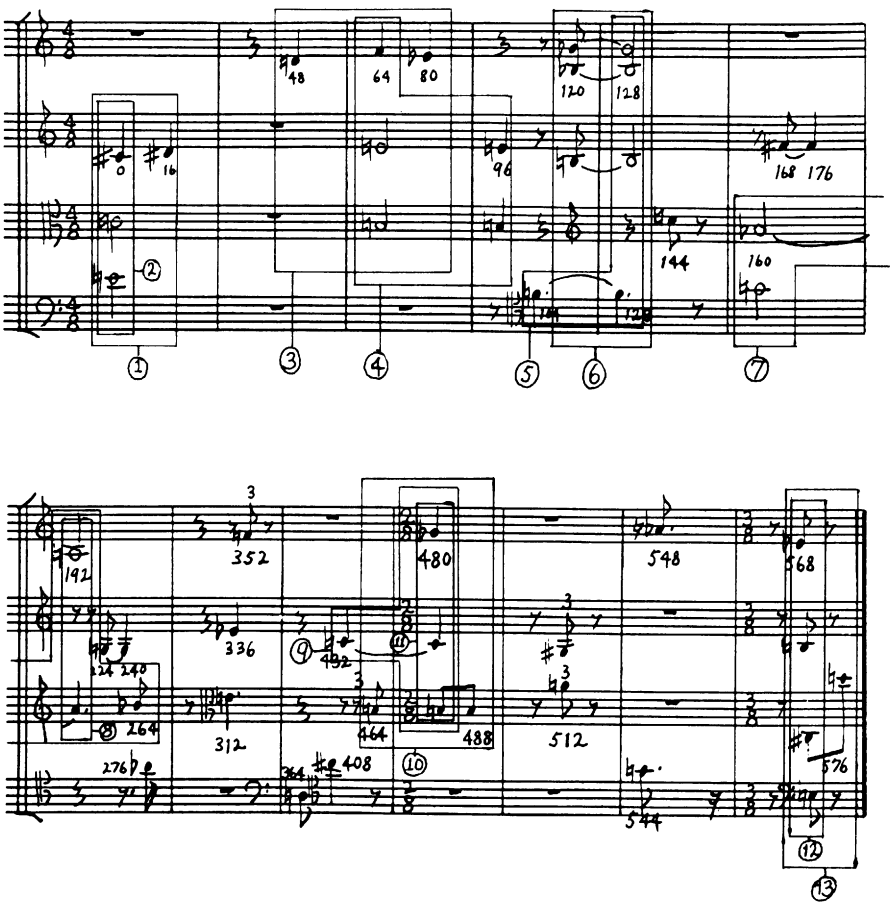
The content of segments derived by BCP is limited with respect to span since the concatenation is restricted to units of block to event-configurations of greater span since it defines the segment as the concatenation unit.

For every PV.I in SG.I (SG.I $\subseteq$ SSGT) and every PV.J in SG.J (SG.J $\subseteq$ RSSGT),

if PV.I = PV.J
then SG.I $\rightarrow$ SG.I SG.J

As remarked above, SCP finds event-configurations which are connected by (at least) one common PV. The SCP-derived segments thus represent continuity over longer spans, even though disjunctions may occur in some subparts of the musical context. As a consequence SCP yields very intricate and subtle readings.

The scanning procedure for SCP differs significantly from the scan for BCP. In the case of BCP the scan for a new segment always begins at the head of the string. This is not true for SCP. Here the first PV in each segment is a potential prime generator of a segment in SCP. If a particular first PV has not previously been a prime generator it is qualified to become one. Let us assume that some first PV, PV.'1' is a prime generator. The scan to find segments for concatenation begins with the first segment to the right of SG.'1', SG.'2' in RSSGT. If the scan finds a match for PV.'1' in SG.'2' the entire segment is concatenated with SG.'1' after duplicate blocks have been removed. The PV.'2' in SG.'1' then becomes a secondary generator; the scan begins again with the first segment to the right of SG.'2' in RSSGT, and so on. Possibly the segment which is concatenated contains position-values which are not in the original SG.'1'. Such position-values become secondary generators.



(P0*0*1*4 P16*3)(P0*0*1*4)(P48*2 P64*4*5*11 P80*3)(P64*4*5*11 P96*4*1
1)(P104*5 P128*5*6*10*11)(P120*6*10*11 P128*5*6*10*11)(P160*7*8 P192*
8*9 P264*10)(P192*8*9)(P432*0 P480*0*6*11)(P464*11 P480*0*6*11 P488*1
1)(P480*0*6*11)(P568*1*3*4*11)(P568*1*3*4*11 P576*2)

Figure 9. Block-Concatenation Parsing of PSGT

In this way, links are formed to other segments and the new segment in SCP is expanded until there are no position-values for which a complete scan has not been made. Thus, unlike BCP, the scan for SCP is ordered: there is no retracing and a prime generator heads only one segment. The characteristic nesting of segments generated by SCP is evident in Figure 10, which gives the complete output string and a partial representation of the corresponding notation for SCP(SSGT). This output string contains eleven segments in all. Because of density, however, it is not feasible to give all output information in notation. Figure 11 gives SCP(PSGT) in its entirety.

3.4 SUMMARY: STRUCTURE OF THE PARSING SYSTEM

From the preceding descriptions of the parsings which make up the system it should be evident that the system is hierarchic, in the initial sense that primary segments are formed before secondary segments. There is also a hierarchic relation between the function BCP and the function SCP. Specifically, if we regard the function calls as string names: (1) Every SG in BCP(PSGT) is a subsegment (possibly improper) of some segment in SCP(PSGT); (2) Every SG in BCP(SSGT) is a subsegment (possibly improper) of some SG in SCP(SSGT). Moreover, the nesting of BCP or SCP, or BCP and SCP does not produce additional distinct strings of segments. For example:

$$\text{SCP}(\text{PSGT}) \supseteq \text{SCP}(\text{BCP}(\text{PSGT}))$$
$$\text{SCP}(\text{SSGT}) \supseteq \text{SCP}(\text{BCP}(\text{SSGT}))$$
$$\text{BCP}(\text{PSGT}) \supseteq \text{BCP}(\text{SCP}(\text{PSGT}))$$
$$\text{BCP}(\text{SSGT}) \supseteq \text{BCP}(\text{SCP}(\text{SSGT}))$$

Also,

$$\text{BCP}(\text{SSGT}, \text{PSGT}) \supseteq \text{BCP}(\text{PSGT}) \cap \text{BCP}(\text{SSGT})$$

and

$$\text{SCP}(\text{SSGT}, \text{PSGT}) \supseteq \text{BCP}(\text{SSGT}) \cap \text{SCP}(\text{SSGT})$$

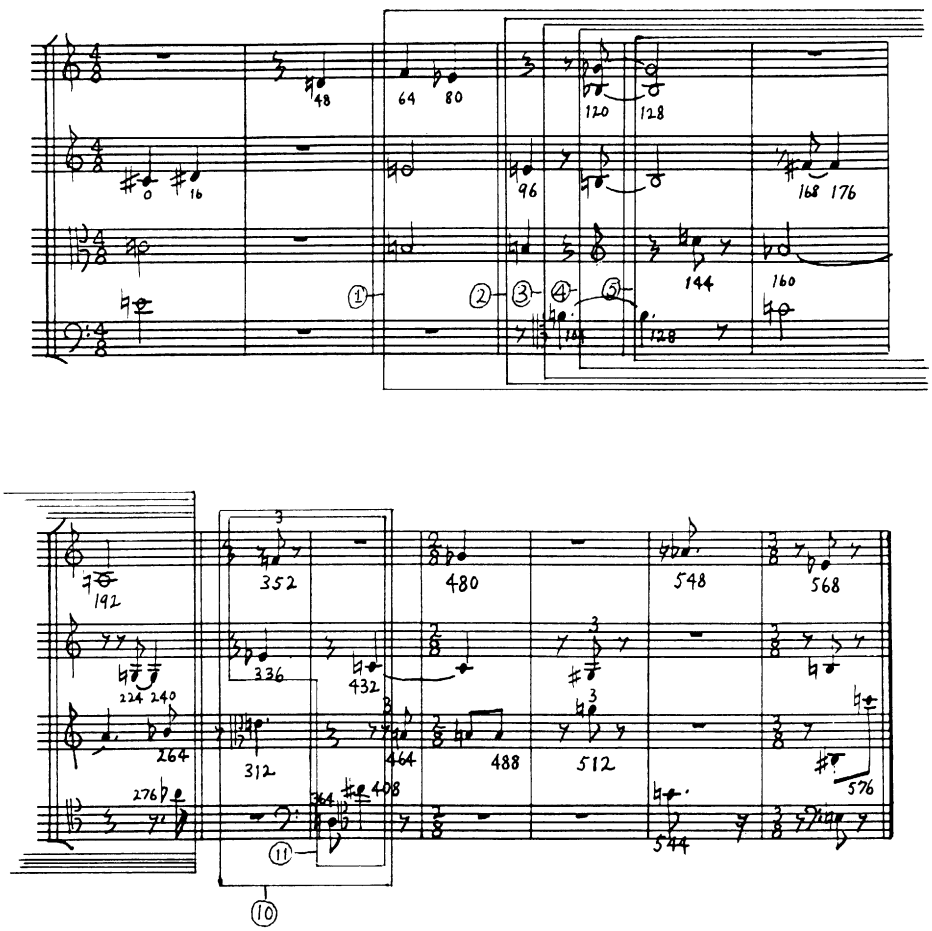
Finally, every SG in BCP(SSGT, PSGT) is a segment in BCP(PSGT) or BCP(SSGT), and every SG in SCP(SSGT, PSGT) is a segment in BCP(SSGT) or SCP(SSGT).

(P64*4*5*11 P80*3 P96*4*11 P104*5 P120*6*10*11 P128*5*6*10*11 P144*0 P160*7*8 P168*6 P176*6 P192*8*9 P224*7 P240*7 P264*10 P276*8)(P96*4*11 P104*5 P120*6*10*11 P128*5*6*10*11 P144*0 P160*7*8 P168*6 P176*6 P192*8*9 P224*7 P240*7 P264*10 P276*8)(P104*5 P120*6*10*11 P128*5*6*10*11 P144*0 P160*7*8 P168*6 P176*6 P192*8*9 P224*7 P240*7 P264*10 P276*8)(P120*6*10 P128*5*6*10*11 P144*0 P160*7*8 P168*6 P176*6 P192*8*9 P224*7 P240*7 P264*10 P276*8)(P128*5*6*10*11 P160*7*8 P168*6 P176*6 P192*8*9 P224*7 P240*7 P264*10 P276*8)(P160*7*8 P168*6 P176*6 P192*8*9 P224*7 P240*7 P264*10 P276*8)(P168*6 P176*6 P192*8*9 P224*7 P240*7 P264*10 P276*8)(P192*8*9 P224*7 P240*7 P264*10 P276*8)(P224*7 P240*7 P264*10 P276*8)(P312*4 P336*3 P352*5 P384*2 P408*1 P432*0)(P336*3 P352*5 P384*2 P408*1 P432*0)

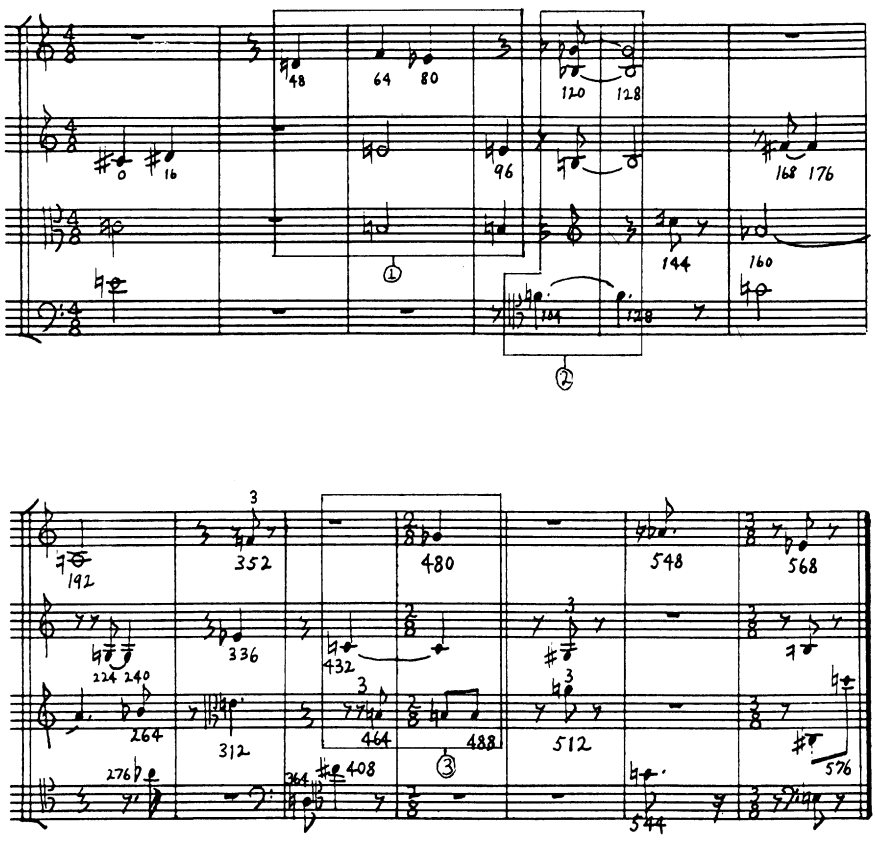
Figure 10. Segment-Concatenation Parsing of SSGT

```
(P48*2 P64*4*5*11 P80*3 P96*4*11 )(P104*5 P120*6*10*11 P128*5*6*10*11 )(
P432*0 P464*11 P480*0*6*11 P488*11 )
```

Figure 11. Segment-Concatenation Parsing of PSGT

Thus, the most efficient (in the sense of shortest scan) and complete set of distinct parsings possible within the system is represented by the following six names:

PSGT
SSGT
BCP(PSGT)
SCP(PSGT)
BCP(SSGT)
SCP(SSGT)

As the output strings are generated in the above order, the schema of string relations for the deletion of all duplicate segments follows accordingly:

IN	DELETE DUPLICATE SEGMENTS OF
PSGT	PSGT
SSGT	SSGT PSGT
BCP(PSGT)	BCP(PSGT) SSGT PSGT
SCP(PSGT)	SCP(PSGT) BCP(PSGT) PSGT
BCP(SSGT)	BCP(SSGT) SSGT
SCP(SSGT)	SCP(SSGT) BCP(SSGT) SSGT

SECTION IV

FROM SCORE READING TO HIGHER ANALYTIC LEVELS

After a reading has been obtained, the question arises: How is the output to be interpreted at a higher level? A comprehensive answer to this question would exceed the scope of the present report. Nevertheless, some indication of the nature of further analytic operations can be given.

A sequence of linked programs in the MAD programming language has been written to process the output from the score reading program in order to yield increasingly higher representations of structural relations. If the collection of pitch-class representatives in each segment of the output strings is scanned to remove duplicates, the resulting collection may be called a compositional set. Then, from a listing of all the compositional sets, the analysis programs: (1) determine the class to which each set belongs; (2) list and count all occurrences of each set-class represented; (3) compute, for each pair of set-class representatives, an index of order similarity; (4) determine the transposition-inversion relation for each pair of set-class representatives; (5) list, for each set-class represented, those classes which are in one of three defined similarity relations to it and which occur in the work being examined; (6) summarize in matrix format the set-complex structure of the totality of classes represented in the work; (7) accumulate and retrieve historical and other informal comments in natural language.

These programs thus permit environments to be examined for similarities and differences and for distributional characteristics which might suggest generalizations about the structure of the particular work or about the structural determinants of the entire class of compositions to which the work belongs. To illustrate this, let us consider certain aspects of the pitch-structure of two contiguous contexts in the Webern work parsed by the reading program.

The first context, extending from PV = 160 to PV = 276, is represented in Figure 12. The position-values are arranged from left to right at the top of the illustration. Each row below then gives the pitch-class representatives for one

PV	160	168	176	192	224	240	264	276	CLASS
PSGT									
1	8			8			10		2-2
2	7								
3		6	6						
4				9					
5					7	7			
6								8	
SSGT									
7	8								
8	7			9					2-2
9	8	6	6						2-2
10	8			8					
11	7	6	6						2-1
12	7,8								2-1
13	8			8			10	8	2-2
14		6	6	9					2-3
15				9	7	7			2-2
16				8			10		2-2
17				8,9			10		3-1
18				9				8	2-1
19					7	7	10		2-3
20					7	7		8	2-1
21							10		
BCP(PSGT)									
22	7,8			8,9			10		4-1
23				8,9					2-1
BCP(SSGT)									
24	7,8			8,9			10	8	4-1
25		6	6	8,9					3-2
26				8,9	7	7			3-1
27				8,9				8	2-1
SCP(SSGT)									
28	7,8	6	6	8,9	7	7	10	8	5-1
29		6	6	8,9	7	7	10	8	4-1
30				8,9	7	7	10	8	4-1
31					7	7	10	8	3-2

Figure 12. Pitch Structure from PV = 160 to PV = 276

segment. The name of the string from which the segment derives is indicated by the headings at the left margin. The name of the set-class to which the compositional set corresponding to the segment belongs is given in the rightmost column. The first integer gives the cardinality of the set, the second gives the ordinal position on a stored list. Criteria of class membership are based upon interval content and will not be explained here [6]. (Since a set containing one element does not form an interval, it is not distinctive with respect to interval content.)

The largest set in Figure 12 is the set numbered 28: class 5-1. It is evident by inspection that all the other sets are subsets of this one and therefore all components of the context belong to a single set-complex. It should be said that the determination of set-complex structure is not as easy as this in other cases. In all, seven set-classes are represented: 5-1, 4-1, 3-1, 3-2, 2-1, 2-2, and 2-3. The theoretical set-complex about 5-1 contains 15 set-classes.

The second context is displayed in Figure 13. There the largest set is numbered 26: set class 6-1. The number of set-classes represented is nine: 6-1, 5-1, 3-1, 3-2, 3-5, 2-1, 2-2, 2-5, and 2-6. All but two of these belong to the set-complex about class 6-1. The two referred to are 3-5 and 2-6. Set 2-6 is a subset of 3-5, which first occurs in the output string BCP(PSGT) and can be seen in the score as the simultaneity at bar 10. Thus, within the context there is a distinct division into two parts on the basis of set-complex structure, a division which is signalled by an entrance of pitch-name C at PV = 432.

A comparison of the two environments shown in Figures 12 and 13 reveals that all the sets in the first are transformed subsets of set number 26; in the second, set class 6-1. This provides a basic and audible measure of change since the event-configuration which begins at PV = 312 is a continuation of the intervallic complex of the preceding context. The next distinctive event, from the standpoint of pitch-relations, is the formation discussed above: the pitch-class collection 0, 6, 11, which is a member of class 3-5.

Many detailed contextual associations are revealed by interpretation beyond the reading stage. In Figure 14, (A) shows a secondary segment distributed between viola and cello. When this is compared with the primary segment at (B) the

PV	312	336	352	384	408	432	464	480	488	CLASS
PSGT										
1	4									
2		3								
3			5							
4				2	1					2-1
5						0		0		
6							11	11	11	
7								6		
SSGT										
8	4		5							2-1
9	4	3								2-1
10	4			2	1					3-2
11		3	5							2-2
12		3		2	1					3-1
13				2	1	0				3-1
14						0				
15							11			
16							11	0,11	11	2-1
17								0		
18								0,6		2-6
19								11	11	
20								6,11	11	2-5
BCP(PSGT)										
21						0		0,6,11		3-5
22							11	0,6,11	11	3-5
23								0,6,11		3-5
SCP(PSGT)										
24						0	11	0,6,11	11	3-5
BCP(SSGT)										
25								0,6,11	11	3-5
SCP(SSGT)										
26	4	3	5	2	1	0				6-1
27		3	5	2	1	0				5-1

Figure 13. Pitch Structure from PV = 312 to PV = 488

two are seen to be related: they are representatives of the same set-class. The first (X) is a transposition of the second (Y) at the interval of eleven half-steps. The relation is nontrivial, for the two notes F and Eb that sound with the E in X are elements of Y. The association of the two contexts is further strengthened by the simultaneities V and W. Again, these are members of the same class: V is a transposition of W at the interval of seven half-steps.

It is hoped that this necessarily incomplete treatment demonstrates the way in which a basis for a still higher level of analysis can be constructed in terms of a system of pitch and interval relations interpreted with the aid of theoretical concepts.

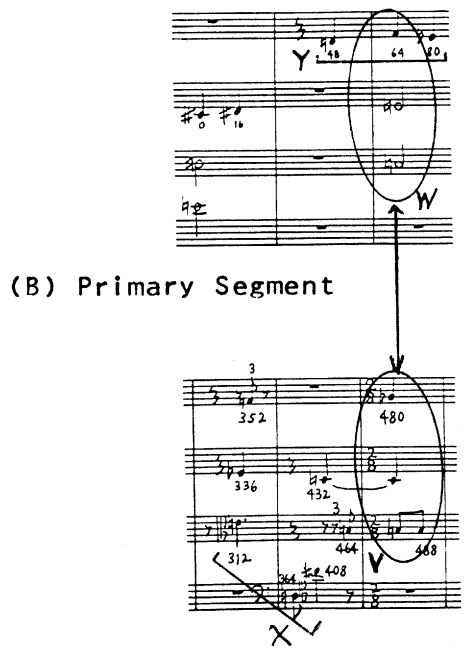
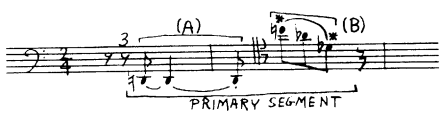


Figure 14. The Association Between Two Contexts.




```
INPUT LANGUAGE CODE FOR (B):  (34**L1 32- 28-L1)
```

Figure 15. Extending the Parsing System Detail

SECTION V

EXTENSIONS AND IMPLICATIONS

A simplified input string was used in order to facilitate development of the parsing program. This limitation in no way affects the generality of the system. Any syntactic class of interest to the analyst can be associated with a position-value, and an appropriate output format can be designed to accommodate as many properties of event-configurations as desired.

Since the parsing rules are based on the occurrence of syntactic structures in the input language (which is capable of representing every syntactic class), it is possible to extend the system in many ways. It could be extended, for example, with respect to depth of detail, as explained below.

In Figure 15, the primary segment, delimited by the bracket below the staff, contains two subsegments of interest: (A) and (B). In particular, (B) is delimited by the slur, with end points marked by "*". In the corresponding input language statement, given below the staff, these end points are represented by "L1". From the standpoint of programming, it would not be difficult to write a parsing function, modelled on the parsing for primary segments, to isolate all such subsegments marked by slurs within primary segments. Any syntactic class or combination of classes could be defined as delimiters in a similar way, thus creating new analytic strata. It might be pointed out in this connection that aspects of individual composition "style" could easily be investigated to any depth, provided, of course, that the researcher can specify the syntactic conditions with sufficient precision.

The second possibility for extending the program is more sophisticated and correspondingly more difficult: that of providing an inductive capability such that scans would be modified under certain conditions to consider other possible parsings or to scan for certain structures conditionally. This heuristic facility would greatly increase the power of the program, and would make it possible to investigate context-sensitivity in greater depth.

In conclusion it should be noted that the score-reading program suggests that a feasible long-range project would be the development of a precise descriptive language for event-configurations. Such a language would be valuable for the study of musical statements in general and would be especially suitable for computer-implemented studies. Much work remains to be done.

REFERENCES

[1] An abstract of the project, being carried out by S. Bauer-Mendelberg and M. Ferentz, is published in: Bowles, Edmund (comp.), "Computerized Research in the Humanities", ACLS Newsletter, Special Supplement (June 1966) p. 38

[2] "Proposed Revised American Standard Code for Information Exchange", Communications of the ACM (April 1965), pp. 107-114

[3] Farber, D.J., R.E. Griswold, and I.P. Polonsky, "The SNOBOL3 Programming Language", Bell System Technical Journal (July/August 1966), pp. 875-944

[4] Babbitt, Milton, "Set Structure as a Compositional Determinant", Journal of Music Theory (April 1961), pp. 72-94

[5] Babbitt, Milton, "Twelve-Tone Rhythmic Structure and the Electronic Medium", Perspectives of New Music (Fall 1962), pp. 49-79

[6] Forte, Allen, "A Theory of Set Complexes for Music", Journal of Music Theory (Winter 1964), pp. 136-183

Massachusetts Institute of Technology
Project MAC
545 Technology Square
Cambridge, Massachusetts
02139

Work reported herein was supported in part by Project MAC, an M.I.T. research project sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Nonr-4102(01). Reproduction of this report, in whole or in part, is permitted for any purpose of the United States Government.

Government contractors may obtain copies of this report from the Defense Documentation Center, Defense Supply Agency, Cameron Station, Alexandria, Virginia 22314. Response will be expedited if requests are submitted on DDC Form 1, copies of which are available from the office in your activity which has been established as the focal point for requesting DDC services.

Other U.S. citizens and organizations may obtain copies of this report from the Clearinghouse for Federal Scientific and Technical Information (CFSTI), Sills Building, 5285 Port Royal Road, Springfield, Virginia 22151.

EDITOR'S NOTE

This paper has also appeared in the Winter 1966 issue of the Journal of Music Theory. At that time, when it was submitted for publication, Allen Forte was Associate Professor of the Theory of Music at the Yale University School of Music. He will shortly be joining the faculty of M. I. T. as Professor of Music in the Department of Humanities.

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 12/28/1995

M.I.T. Libraries
Document Services

CS-TR Scanning Project
Document Control Form

Date : 12/11/95

Report # LCS-TR-39

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 42(49-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☐ Single-sided or
☒ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☒ Unknown ☐ Other:_____

Check each if included with document:
☒ DOD Form ☒ Funding Agent Form ☒ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number):_____

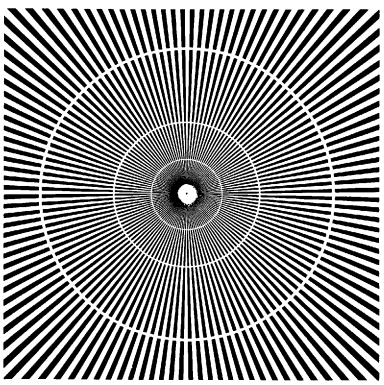
Photographs/Tonal Material (by page number):_____

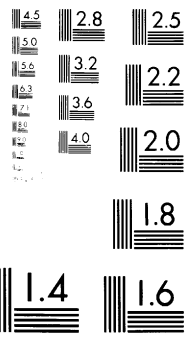
Other (note description/page number):

Description : Page Number:
IMAGE MAP: (1-42) UNNUMBERED TITLE & BLANK PAGES, III-VI,
1, UNNUMBERED BLANK, 3-36
(43-49) SCANNING TARGET, COVER, FUNDING AGENT, DOD, TRGT'S(3)

Scanning Agent Signoff:
Date Received: 12/11/95 Date Scanned: 12/28/95 Date Returned: 12/28/95

Scanning Agent Signature: Michael N. Cook



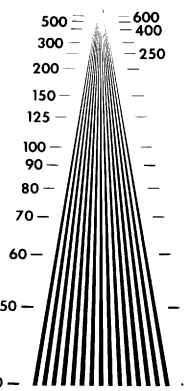


NMA MICROFONT QJKLPYZ
6BS12GH5D4X7U3W8V9E
PQR45DE9UV670FG8STHIJNOWXABYZ
3KLM12C IJK90FGHBSTJ67AB45CDEZ123VWNOPXYQRLM

ABCDEFGHIJKLMNOPQRSTUVW
XYZabcdefghijklmnopqrst
uvwxyz0123456789 OCR-B

ABCDEFGHIJKLMNOPQRSTUV
WXYZabcdefghijklmnopqr
stuvwxyz1234567890PICA

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Elite



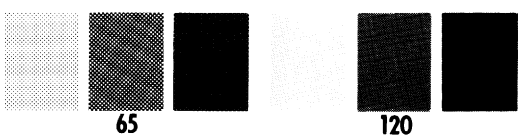
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 6 pt

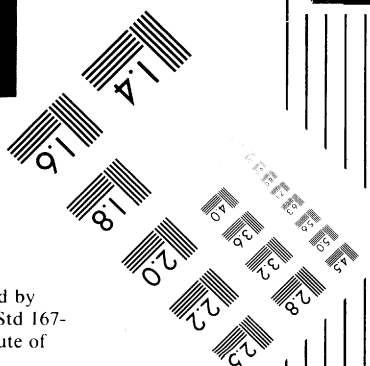
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 4 pt

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 8 pt

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 10 pt

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 12 pt





IEEE Std 167A-1987

FACSIMILE TEST CHART

Prepared by the IEEE Facsimile Subcommittee and printed by Eastman Kodak Company. Use in accordance with IEEE Std 167-1966, Test Procedure for Facsimile. Copyright 1987, Institute of Electrical and Electronics Engineers.

AIIM SCANNER TEST CHART #2

Spectra

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789

Times Roman

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789

Century Schoolbook Bold

4 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789**

6 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789**

8 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789**

10 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789**

News Gothic Bold Reversed

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789

Bodoni Italic

4 PT *ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789*

6 PT *ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789*

8 PT *ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789*

10 PT *ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',../?$0123456789*

Greek and Math Symbols

4 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',../≦±=≠°><⋫⋪⋬≤≡

6 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',../≦±=≠°><⋫⋪⋬≤≡

8 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',../≦±=≠°><⋫⋪⋬≤≡

10 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',../≦±=≠°><⋫⋪⋬≤≡

White

Black

Isolated Characters

e	m	1	2	3	a
4	5	6	7	o	°
8	9	0	h	I	B

A4 Page 6543210

MESH HALFTONE WEDGES

65

85

100

110

133

150

A4 Page 6543210

Massachusetts Institute of Technology
Project MAC
545 Technology Square
Cambridge, Massachusetts
02139

Work reported herein was supported in part by Project MAC, an M.I.T. research project sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Nonr-4102(01). Reproduction of this report, in whole or in part, is permitted for any purpose of the United States Government.

Government contractors may obtain copies of this report from the Defense Documentation Center, Defense Supply Agency, Cameron Station, Alexandria, Virginia 22314. Response will be expedited if requests are submitted on DDC Form 1, copies of which are available from the office in your activity which has been established as the focal point for requesting DDC services.

Other U.S. citizens and organizations may obtain copies of this report from the Clearinghouse for Federal Scientific and Technical Information (CFSTI), Sills Building, 5285 Port Royal Road, Springfield, Virginia 22151.

EDITOR'S NOTE

This paper has also appeared in the Winter 1966 issue of the Journal of Music Theory. At that time, when it was submitted for publication, Allen Forte was Associate Professor of the Theory of Music at the Yale University School of Music. He will shortly be joining the faculty of M. I. T. as Professor of Music in the Department of Humanities.