3-26-1987

Stylistic analysis and recognition of piano sonatas of four composers -- Mozart, Chopin, Debussy, Anton Webern

Emily Feng-Hwa Lin-Jeng

ROCHESTER INSTITUTE OF TECHNOLOGY

SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

Stylistic Analysis and Recognition of Piano Sonatas of Four Composers -- Mozart, Chopin, Debussy, Anton Webern

by

Emily Feng-Hwa Lin-Jeng

A thesis, submitted to
the Faculty of the School of Computer Science and Technology,
in partial fulfillment of the requirements for the degree of
M.S. in Computer Science

APPROVED BY : John A. Biles

Professor John Al Biles

: Stanislaw Radziszowski

Professor Stanislaw Radziszowski

: Peter G. Anderson

Professor Peter Anderson

Title of Thesis:

Stylistic Analysis and Recognition of Piano Sonantas of Four Composers -- Mozart, Chopin, Debussy, Anton Webern

I Emily Feng-Hwa Lin Jeng hereby deny permission to the Wallace Memorial Library, of R.I.T., to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date: March 16, 1987

Emily Feng-Hwa Lin Jeng

Emily Feng-Hwa Lin Jeng

0.2 Abstract

This thesis describes a system that incorporates techniques developed by musicologists to do stylistic analysis of music, an important applied field in music theory analysis. To do the analysis requires the knowledge of many musicological analysis methods and pattern recognition algorithms that are central issues to this project. In addition, AI techniques of learning were used to improve the whole system's skills.

The conclusions reached as a result of this project were that computers can perform musical tasks usually associated exclusively with naturally intelligent musicologists, and that learning techniques can expand and enrich the behavior of musically intelligent systems.

0.3 Key Words and Phrases

music theory, pattern recognition.

0.4 Computing Review Subject Codes

I.1.2 Artificial Intelligence

I.1.2.1 Applications and Expert System

I.1.2.4 Knowledge Representation Formalisms and Methods

I.1.2.6 Learning

0.5 Acknowledgement

I benefited from discussions with Professor Radziszowski. Special thanks are due to my advisor, Professor Biles, for his treasure of suggestions, encouragement and support.

I am grateful to my husband who helped reviewing the thesis and his many excellent suggestions for improving the system. It hardly needs saying that much of the value of this system is due to these individuals' assistance, but I alone bear responsibility for any error or omission that remain within these covers.

0.6 Table of Contents

1. Introduction & Overview

Many technological advances have been made over the last several years in the field of Artificial Intelligence (AI). In the area of applications to music, AI techniques have been used in four areas: composition, performance, music theory, and digital sound processing. This thesis fits into the third application area - music theory analysis. It is a system to analyze and compare special characteristics of music masterpieces composed by four composers:

Classical Period: Wolfgang Amadeus Mozart (1756-1791)

Romantic Period : Frederic Chopin (1810-1849)

Impressionism : Claude Debussy (1862 -1918)

Twentieth Century : Anton Webern (1883-1945)

The system tries to recognize different personal styles. It tests and classifies how a given work differs from those by other composers and hence identifies the composer for unknown pieces. In this thesis, only piano pieces were chosen for testing in order to limit the scope of the project. The problems of comparing different characteristics for different instruments, and in methods for reasonably representing full orchestral scores would be worth pursuing in another thesis.

The system written for this thesis did a stylistic analysis in five steps (Figure 4-1 in page 42). First, the music scores were encoded using a coding language called DARMS [Forte][Rothgeb][Erickson]. Though a number of viable coding schemes were available, most of these can represent more information than was needed for this project. It is obvious that more complex coding systems increase program complexity and magnify the task of coding the music. Consequently I decided to use DARMS code, a coding scheme that is concise,

mnemonic and adequate for the project analysis.

In the second step, the DARMS codes were fed into the characteristics analyzers, which are models for finding musical characteristics of styles. The analysis processes in the characteristics analyzers simulate the manner in which a human analyst might search for stylistic characteristics in the music. In the third step, the system derived and built a knowledge base (for music pieces used as sample) and a testing data base (for music pieces used for testing) from the results of the characteristics analyzers. These data bases represent the "knowledge" and the "meaning" of the music data and were used to organize the searching, parsing, matching, recognition and inferential processes involved in identification. In the fourth step, the system fed these data bases to the classifier to identify the composer of an unknown music work by doing comparisons between characteristics of the unknown music piece and those in the existing knowledge bases.

Finally, the system manipulated its internal classification tools based on their performance, such as the differences between the correct answer and the one those tools detected and the effects to the identification results those differences caused. The system then made a judgement about which tool should be emphasized and which should be discarded. This is where learning happens. The system gains knowledge about which characteristics are useful stylistic indicators and which characteristics can be ignored.

The first three chapters of this thesis provide a general overview of the system and some background concepts. In chapter 1, I describe what the goals of the thesis are, and I give a brief overview of how the computer system was built and how the system works. In chapter 2 and chapter 3, I discuss previous work that provides the conceptual background for the implementation of the

thesis. I also briefly introduce each chosen composer's musical preferences in their compositions and some typical musical styles of each period from the musicologist's point of view.

Chapter 4 describes the system specifications including a detailed description of each major module in the system. Chapter 5 describes the test plan and results. Chapter 6 gives the overall conclusions, a description of the problems encountered and a discussion of topics in this field for future research. Chapter 7 includes closing thoughts. The last three chapters are bibliography, appendices and program listing, respectively.

2. Previous Work and Background

As mentioned before, four areas of musical research employ AI techniques, Composition, Performance, Music Theory and Digital Sound Processing. Since this thesis is in the Music Theory category, I will discuss work in this field only. For readers who are interested in the other three areas, please refer to [Von Helmboltz][Plomp][Fletcher][Freedman][Risset][Luce] [Beauchamp][Silvian & Dunn & White][Divilbiss].

To begin a more detailed discussion, I'll define a music score as a representation of a musical composition, showing all parts for the instruments or voices. A generative approach to music theory involves the construction of a computer program or system, either to produce a music score that conforms to a known musical idiom, or to analyze a known score and check it against the existing theory. Some of the earliest work in this realm was reported in a survey by Hiller [Hiller 1970].

2.1 Representation of Music

In order to analyze a musical score by computer, the score must be encoded. In other words, the score must be translated to a machine readable representation. There are lots of representations for music. Some representation approaches are aimed at encoding musical data in fixed formats and only represent the surface structures of the musical data. Other approaches can represent the interpretation of the meaning (the active interconnections that glue together pieces into compositional or analytical concepts) of the musical data properly. Within those representational approaches, the grammatical approach is probably the most popular one.

2.1.1 Grammatical Approach

Just as a formal linguist describes the structures of a language by devising collections of rules called grammars that can be used to generate or recognize the sentences of language, a grammar is also a reasonable approach to represent music. It puts strong constraints on the musical patterns that are used in the score.

In linguistics studies or AI studies in language understanding, three major grammatical approaches can be distinguished: (1) formal (traditional) (2) generative (transformational) and (3) systemic [Winograd 1983][Chomsky 1958,1962][Halliday 1961] [Firth 1957].

Linguists and researchers explain that sentences and phrases of a language are made of "patterns." Those patterns can be as simple as a physical object (word) [Winograd 1983] whose form is identical, or a more complex structure like a transition network which is a formal structure combining a set of simple patterns.

A formal grammar puts constraints on the patterns and provides a way of describing the structure of a language and setting up a correspondence between the structures generated in producing or recognizing a sentence, and the process of recognition and production [Winograd 1983].

Many kinds of formal grammars have been developed and used in language understanding. Context-free grammars are probably the most widely used because of their simplicity and clarity. Context-free grammars are also known as immediate constituent grammars (by traditional linguists), Backus normal form (by programming language designers), and recursive patterns (in some computer applications) [Winograd 1983]. Context-free language is independent

of the context of the pattern when do a replacement of a pattern. Conversely, context-sensitive grammar is context-sensitive, in which replacements may be conditioned on the existence of an appropriate context [Lewis and Papadimitriou].

Generative (transformational) grammar [Hopcroft and Ullman][Chomsky 1965][Lewis and Papadimitriou] developed by Chomsky [Chomsky 1957,1965] and his followers has dominated theoretical linguistics over the last 25 years. According to Winograd, transformational grammar can be broadly explained as three related but distinct things: (i) The generative paradigm: Within this paradigm, the problem of linguistics is seen as that of understanding syntactic competence - the formal structure that underlies the intuitions of a native speaker about the grammatical structure of sentences. (ii) The transformational model, a particular model within the generative paradigm. It has served as the basis for much of the work in generative linguistics. However, it is not the only possible generative model, and much of the linguistic research is described as "non-transformational". As was true with the context-free grammars mentioned above, the basic ideas of the transformational are consistent with the generative paradigm but are not transformational. (iii) Within the transformational model therer is one mechanism called transformation rule and the transformational grammar was so named. But the current attempts seek to limit the importance of transformation while processing other parts of the overall transformation model.

In the development of transformational grammar, Chomsky [Chomsky 1965] proposed that linguistic theory should deal with relations between structures by looking both at surface structure, which catches the surface information and at deep structures, which capture the underlying similarities and

differences. This meant dealing with structures that could not be observed in the data but were postulated to have some kind of underlying meaning. The solution proposed by Chomsky for these problems is that, instead of treating grammar rules as constraints on the structure, treated them as rules for specifying the steps in a "derivation" (rewrite rules). Thinking of rules as steps in a derivation opened up possibilities for new kinds of rules that had no direct correlates in the phrase structure analysis.

Chomsky developed a formal theory of grammars in which he devised a classification of grammars into a hierarchy according to their power (table 2-1). There are four types of grammars, (type 0, 1, 2 and 3), each defined by the kind of rewrite rules it contains. Type 0 is the most powerful one, and each increasing number is more restricted. For each grammar type, there is a class of languages that can be defined only by a grammar of that power or greater power but not lower power [Hopcroft and Ullman][Chomsky 1965, 1958][Winograd 1983][Lewis and Papadimitriou].

Table 2-1 The hierarchy of grammars

Type	Restriction on Rule Form	Languages
0	Finite set of rules	Any language whose sentence can be generated by any deterministic or non-deterministic computational machine.
1	The right-hand side consissts of the left-hand side with a single symbol expanded.	Any language whose sentence can be recognized by any non-deterministic computational machine using an amount of storage proportional to the length of the input. (called a linear bounded automation)
2	The left-hand side consists of a single symbol.	Any context-free language
3	The right-hand side consists of a single terminal symbol or a terminal symbol followed by a single non-terminal symbol.	Any regular language

*** The terminology in the table is used under teh assumption that the reader has grammar analysis background. For those who don't, please refer to [Lewis and Papadimitriou].

The entries for rule restriction in this table are cumulative - the rules of a type 3 grammar obey the restriction for type 1 and type 2 as well. There is no standard name for type 0 grammars, although they are sometimes called unrestricted rewriting systems and are described as having Turing machine power [Hopcroft and Ullman][Lewis and Papadimitriou]. I included this table here because in my thesis, I use finite automata (computational machine) to recognize musical data (section 3.2.2).

In the same period during which transformational grammar emerged, a series of developments with AI followed a different approach. Instead of taking the process of derivation, they took the process of parsing, starting with a basic top-down left-to-right scheme. The clearest formulation was the Augmented Transition Network (ATN) [Woods 1970][Koplan 1972]. Because ATNs and their descendants are currently one of the most common methods of parsing natural language in computer science [Winograd 1983] [Lewis and Papadimitriou 1981][Woods 1970][Kaplan 1972] and they are clear enough to be grasped and followed easily, I do not include the detail explanation of it here.

Before we discuss systemic grammar, one thing that needs to be mentionsed is that in most grammatical approaches, language is looked at as a cognitive process [Winograd 1983], and the linguist is concerned with discovering the knowledge structure and processes of the individual language user. The linguists who developed systemic grammar have not taken cognitive processing as their fundamental starting point. The roots of their work were in anthropology and sociology [Halliday 1961] [Firth 1957]. Systemic grammar emphasizes the functional organization of language - how it presents speakers with systems of meaningful options as a basis for communication. It makes use of structural analysis, but it emphasizes the importance of context and situation in the analysis of each utterance or text.

One of the most important concepts in systemic grammar is the concept of realization, which provides the formal basis for integrating the components of a systemic grammar. One good example, case grammar, is a form of grammatical analysis dealing with the participants in a relation or action described by a verb. It is quite similar to systemic grammar, and we can relate systemic ideas to those of case grammar.

Currently, there are also new formalisms that have evolved outside of systemic grammar but share much of its orientation to structure and realization. Readers interested in knowing more about it should refer to [Winograd 1983][Gazdar, forthcoming].

In the musical domain, these grammatical approaches have also been used for analysis. Winograd's systemic grammars for the harmonic analysis of musical scores [Winograd 1968] can be looked upon as a pioneering work in analyzing music using a grammatical approach. In the 1970s and 1980s, representations based on generative grammars were also proposed [Chomsky 1957, 1965] and tested [Baroni and Jocoboni] [Baroni and Snell]. Other generative systems, not yet implemented in software, were proposed by [Lerdahl 1977,1983][Snell 1979]. Lerdahl used rules to assign generative structure to parts of existing pieces, while Snell proposed the use of grammars to generate musical fragments exemplifying a known style. These schemes were proposed or used in building knowledge representations for the deep structure of musical data.

As mentioned earlier in this section, some representation schemes that only consider the surface structure of music data (e.g. music printing) have also been used to represent musical scores [Forte][Rothgeb][Erickson]. DARMS is one of those, and it was used in my thesis. I used DARMS code to represent the first level (surface structure) of the musical data, but after feeding the fixed-format DARMS code to the characteristics analyzer and through all the characteristics analysis, a new knowledge base has been created that not only found instances of musical structures, but actually made connections between them and built internal knowledge structures on the basis of what it found. The following section covers a complete view of DARMS.

2.1.2 DARMS Codes

As we know, it is difficult to compare music by "watching" the notes, so we have to translate them into some code that can be analyzed. DARMS was chosen as the input language for this purpose. DARMS [Forte][Rothgeb] is an acronym for Digital Alternate Representation of Musical Scores and was originally conceived by Steven Bauer-Mengleberg in 1963. The intended purpose of DARMS is to provide a language of (computer readable) graphic symbols to represent music. It has been used in a number of theoretical studies by scholars such as Allen Forte and John Rothgeb[Forte][Rothgeb][Erickson]. I chose it as the input language for this project because of its readability, flexibility and completeness.

In figure 2-1, the basic DARMS codes are presented. DARMS encodes the graphic symbols used to represent music, not their meaning. It "knows" what symbols are in the score and where they are, but not what they represent. In that sense, uninterpreted DARMS is about as useful to an analysis program as the printed score is to a person who cannot read music. According to the figure, the lines and spaces on the staff are assigned integer numbers called space codes. Duration is specified by duration code: W for whole note, Q for quarter note, etc. Clefs are coded as an exclamation point followed by G, C, or F indicating the type of clef. Key signatures and meter signatures also follow the exclamation point. Beamed notes are coded with parentheses that indicate the number of beams on the note; a left parenthesis indicates the beginning of a beam, a right parenthesis the end. When more than one instrument is being coded, the current instrument is identified by an I followed by a numeric identifier.

Fig. 2-1 Original Basic DARMS codes

Space Codes:

```
            to 49

                36
          35 ---------
                34
          33 ---------
                32
          31 ---------
       30
29  ------------------------
       28
27  ------------------------
       26
25  ------------------------
       24
23  ------------------------
       22
21  ------------------------
       20
          19 --------
                18
          17 --------
                16
          15 --------
                14

             to 00
```

Key Signatures:

!K2# = key of two sharps

!K4- = key of four flats

nonstandard key signatures use accidental code followed by space code: !K-25#22 (flat on 3rd line and sharp 1st space)

Accidentals:

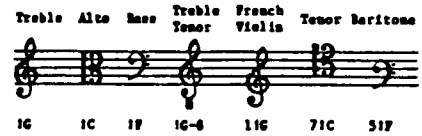
#	sharp
##	double sharp
-	flat
--	double flat
*	natural

Duration codes:

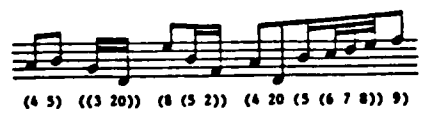
WW	Breve
W	Whole
H	Half
Q	Quarter
E	Eighth
S	Sixth
T	Thirty-Second
X	Sixty-fourth
Y	One hundred twenty-eighth
Z	Two hundred fifty-sixth
ZZ	Five hundred twelfth
ZZZ	One thousand twenty-fourth
.	(dot)
G	Single grace note

The duration code generally follows a note's space code, or R for a rest.

Clefs:



Beamed Notes:



Meter Signatures:

!M3:4	3/4
!MC	4/4
!M2+3+2:8	2+3+2/8

Instrument Codes:

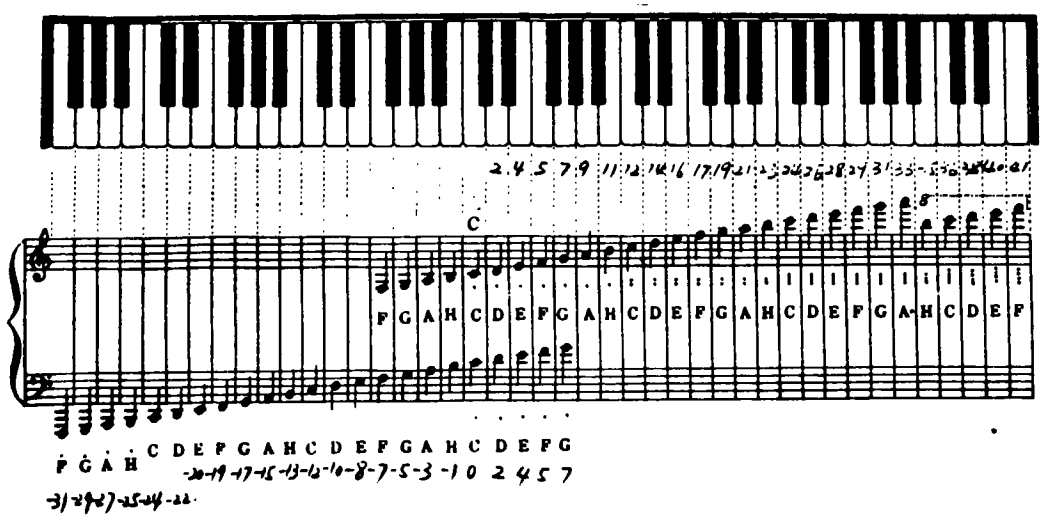
I1 (Instrument 1)
I2 (Instrument 2) ... I99

The subset of DARMS that was used is the following:

i. Space Codes (Figure 2-2)

Unlike many other music coding languages, DARMS does not name pitches, but instead encodes the location of notes on the staff. The lines and spaces on the staff, as well as ledger lines, are assigned integer numbers called space codes. Each note's position on the staff or piano keyboard is represented by one of these numbers. Note that the mapped integers are modified to cope with more convenient processing. For example, the original space codes for C to B are 19 through 24, which does not address the interval differences between pitches at all. The space codes used in my project are 0, 2, 4, 5, 7, 9 and 11, which werer chosen to reflect the interval differences. The only disadvantage for the DARMS space code format is that the data become less meaningful for users who are not familiar with the corresponding codes.

Figure 2-2 modified DARMS SPACE CODE



ii. Duration Codes (table 2-2)

Duration is specified by duration codes, which are for the most part mnemonic: W for Whole note, Q for Quarter note, etc. The system translates each DARMS duration code into a fractional value -- 1 for whole note (W), 1/2 for half note (H), etc. The value of dot is assigned half the duration of the note. If the duration code is followed by a dot, the fractional value for the dot is added to the duration, and the value of dot is halved. This process is repeated for any number of dots. Thus the duration for Q•• is 1/4 + 1/8 + 1/16, or 7/16. The duration code mapping used in the system is not complete. For example, it does not provide codes for eighth note triplets, half note triplets, and trills. However, it represents enough to meet the system's analysis requirement.

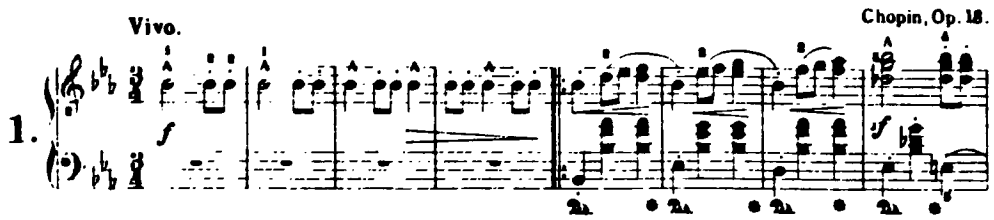
Table 2-2 DARMS Duration Codes

W	Whole	---- 1.0
I	Half with dot	---- 1/2 + 1/4 = 0.50 + 0.25 = 0.75
H	Half	---- 1/2 = 0.5
P	Quarter with dot	---- 1/4 + 1/8 = 0.25 + 0.125 = 0.375
Q	Quarter	---- 1/4 = 0.25
D	Eighth with dot	---- 1/8 + 1/16 = 0.125 + 0.0625 = 0.1875
E	Eighth	---- 1/8 = 0.125
S	Sixteenth	---- 1/16 = 0.0625
T	Thirty-second	---- 1/32 = 0.03125
X	Sixty-fourth	---- 1/64 = 0.015625
O	For triples	---- 1/3 = 0.33
M	For triples	---- 1/3 = 0.34 { e.g. 30, 50, 9M }
•	dot	---- e.g. Q•• = 1/4 + 1/8 + 1/16

Figure 2-3 is a simple example of the input code for Chopin's Op18. Each note is represented by a corresponding space code with a capital letter indicating the length of the note (duration code). Both comma and ampersand are used as note separators, but ampersand also indicates concurrent musical

notes, (i.e. the notes occur at that same time).

Figure 2-3 sample score input. (Chopin Op18)




```
{ Score name, Composer's name, key signature ... }
part1: 10H, 10E, 10E/ { 1st phrase with a whole note, two 1/8 notes}
        10H, 10E, 10E/ { 2nd phrase (same as the 1st phrase) }
        10Q, 10E, 10E, 10Q/ {3rd phrase}
        10E, 10E, 10Q, 10E, 10E, 10E/ {4th phrase}
        10Q, 14E, 15E, 17Q & 14Q/  {5th phrase}
        10Q, 15E, 17E, 19Q & 15Q/  {6th phrase}
        10Q, 17E, 19E, 21Q & 17Q/  {7th phrase}
        23H & 19H & 13H, 23E & 19E & 13E, 23E & 19E & 13E/ { 8th
phrase }

Part 2: RH, RQ/ RH, RQ/ RH, RQ/ RH, RQ/ { first 4 phrases}
        -14Q, 8Q & 5Q & -1Q , 8Q & 5Q & -1Q/   { 5th phrase }
        -9Q, 7Q & 4Q & -1Q, 7Q & 4Q & -1Q/   {6th phrase}
        -10Q, 8Q & 5Q & -1Q, 8Q & 5Q & -1Q/  {7th phrase}
        -9Q, 7Q & 2Q & -1Q, -8Q/ {8th phrase}
{The note following & happens concurrently with the preceding note }
```

As we can see, this scheme provides an economical representation, and in addition shows the notational groupings used by the composer. The basic code is easy to learn, since its designers have taken care to use mnemonic symbols whenever possible.

2.2 Different approaches in music theory analysis

More recently, with the computer's obvious utility as a model analysis and testing medium and under the influence of cognitive science, a handful of

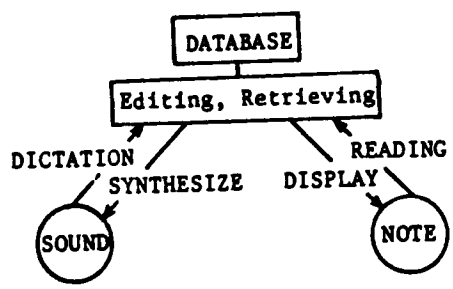
musicologists have used computers for theoretical work. Their research has been dominated by statistical techniques [Lincoln 1970], but in the area of AI research, the current focus is on knowledge based systems, in which searches for patterns in the input [Minsky 1975] are aided by domain-specific knowledge. In this kind of knowledge based system, the domain of discourse needs to be rich enough for studying the central issues, yet simple enough to facilitate progress. Because the knowledge is used to search patterns, parse sentences or infer concepts for analysis, the representation of the knowledge is the key to success.

From the standpoint of stylistic analysis, one can develop and apply techniques of comparative musicology to determine the type of a musical work and to compare the similarities among different types of music [Beauchamp]. Such programs are very limited. Some degree of success has been achieved in using the computer as an aid to forming or testing musical stylistic theories, while some attempts have addressed stylistic analysis of specific characteristics (e.g. Choral Melodies, [Baroni and Jocobini]) of a certain composer (e.g. Johann Sebastian Bach, [Brinkman]) or even one specific work of one composer (e.g., Bach's Orgelbuchlein, [Brinkman]). Few studies, however, have thoroughly examined distinct characteristics and derived any stylistic comparison or identification of different musical pieces by different composers. Yasuaki and Seiji [Yasuaki and Seiji] (hereafter Y&S) have developed a music information processing system especially for comparative musicology which has been used, for example, to determine the type of scale in folk music and to compare the similarities in melody and rhythm between different pieces. Because some concepts and analysis methods of the Y&S system were used in this thesis, a separate section is added to explain more detail of this system.

2.3 Yasuaki and Seiji's Information processing system

According to Y&S, musical information can be expressed by two types of patterns: one is an audible pattern - sound - and another is a visible pattern - music notes. though there is analogy with natural language, which has speech patterns and characters, it is more difficult to dictate the sound of melody and to read the musical notes. The music information processing system built by Y&S has two functions: dictation of melody sound (melody analyzer) and recognition of music note (music database). They used a synthesizer and a displayer for monitoring the music. Figure 2-4 shows the basic organization of the system. In the following, I will explain a little about the two functions and its applications to comparative musicology.

Figure 2-4 Yasuaki & Seiji's Music Information Processing System



(a) melody analyzer - to detect the pitch and the length of tone

Pitch interval can be identified by detecting the fundamental frequency of sound. Firstly, the sound signal is transformed into the power spectrum using an FFT algorithm [Yasuaki and Seiji][Longutet 1971]. In order to get the accurate frequency F_p, they compensate spectral peak frequency F_i with the

phase change as follows: (Figure 2-5)

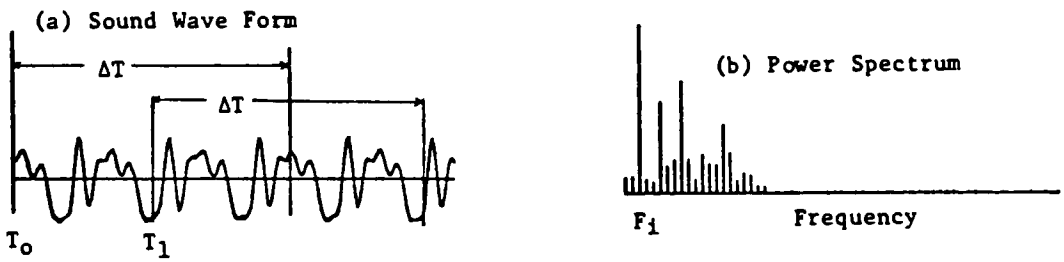
$$F_p = F_i + (\theta_1 - \theta_0) / 2\pi (T_1 - T_0)$$

where θ_1 and θ_0 are phases of F_i in the interval $[T_1, T_{1+}\Delta T]$ and $[T_0, T_0 + \Delta T]$, respectively, Pitch P is expressed as follows

$$P = 12 * \log_2 (F_p / F_r)$$

where F_r is reference frequency [Yasuaki & Seiji][Longutet 1971].

Figure 2-5 Frequency analysis



The pitch obtained has an unstable frequency at the transition between tones, but they used a median filter since the pitch of melody changes in a stepwise manner.

(b) Music Data Base

Y&S encoded musical data into a three-number code as shown in Figure 2-6. The M, N and L represent pitch name in octave, octave postion and note length, respectively. The length (L) for an eighth note is 4 and for quarter note 8, for half note 16, etc. The M and N are obtained as: (where P is pitch

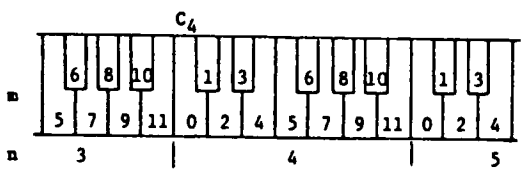
defined as above)

$M = \mathrm{mod}_{12}\,[P]$

$N = [P/12]$

The Notation of a music note is not unique, but the pitch code in the data base described above is unique [Yasuaki and Seiji][Longutet 1971].

Figure 2-6 Data Format



(c) Application to comparative musicology

(i) Tetrachord and Nuclear notes

Y&S determined the type of scale of Japanese folk music by the type of tetrachord [Dvidow] included in a melody and the position of nuclear notes [Lincoln 1970]. A scale is made up of a concatenation of two tetrachords. Nuclear notes are (i) Basic note of tetrachord, (ii) Median note of three adjacent notes, (iii) Very often final tone of melody and (iv) upper note in two-tone melody [Yusuaki & Seiji]. Table 2-3 is a list of scale type of Japanese folk music.

Table : 2-3 Possible scale type of Japanese Folk Music

(Underline indicate Tetrachord)

(1) Minyo Scale

+3+2 +2 +3+2

(2) Miyakobushi Scale

+1+4 +2 +1+4

(3) Ritsu Scale

+2+3 +2 +2+3

(4) Ryukyu Scale

+4+1 +2 +4+1

*These scales are shown in the form of pitch difference sequence.
*These scales are only for Japanese folk music.

(ii) Find out the similarity of melody and complexity of rhythm and melody

Melody may be considered as a finite Markov process [Y & S], in which the state k_i is the ith pitch in a melody. The pitch transition probability, p_{ij}, is the conditional probability that the note k_i is followed by k_j The weight of a note in the twelve-tone scale can be expressed by its steady state probability W, which can be solved by the following equation:

$$W = W * P = [w1\ w2\ \ldots wn] * \begin{array}{c} \\ k1 \\ k2 \\ \vdots \\ kn \end{array} \begin{array}{c} \begin{array}{cccc} k_1 & k_2 & \cdots & k_n \end{array} \\ \begin{bmatrix} p11, & p12, & & p1n \\ p21, & p22, & & p2n \\ \vdots & & & \\ pn1, & pn2, & & pnn \end{bmatrix} \end{array} \qquad \text{where } \sum_{i=1}^{n} w_i = 1$$

The weight of a pitch is effective in expressing the similarity between different melodies belonging to the same scale class. In order to solve the equation, a pattern matching Markov chain is built to get all pairs of notes (k_i k_j) in the musical data (Section 3.2.1).

I also referred to the analytical method Y&S proposed to calculate the complexity of melody and rhythm. They are defined as follows:

The root mean square of the note difference in melody indicates the complexity of melody:

$\delta_m = \mathrm{sqrt}[\ (1/n) * \sum_{i=1}^{n}(P_i - P_{i+1})^2\ ]$ where P_i is the pitch of note $(M + N * 12)$

The standard deviation δ_R of rhythm ratio R_i indicates the complexity of rhythm.

$$\delta_R = (1/n-1) * \mathrm{sqrt}(\sum_{i=1}^{n} R_i)$$

$$R_i = \log_2 (L_i / L_{i+1}) = \log_2 (L_i) - \log_2 (L_{i+1})$$

where L_i is the length (duration) of ith note

2.4. From the musicologist's point of view

A brief discussion of the musical backgrounds of the four composers is necessary because it would be presumptuous to attach any significance to musical characteristics without first developing some perspective on each composer's musical style or advancing the understanding of fundamental aspects in musical works. In this thesis, three basic elements in music are considered, rhythm, melody and harmonic structure. These are usually significant in a musical framework. Depending upon the emphasis intended by the composer, these elements either define or contribute to the organic whole. The study of melodic and harmonic rhythm may reveal the composer's pacing of musical events: melody may suggest the composer's attitudes toward expression and meaning of music, and harmony may provide important clues regarding musical architecture. The following is a summary of musical conclusions regarding this analysis from a musicologist's point of view.

2.4.1 Classical Era and Mozart

"Classical" is generally defined as a style that achieves an objectivity, a clarity of form, and a balanced emotional expression whether in literature, painting, or music [Grout] The period of 1750-1820 is commonly denoted as the Classical period.

The individual parameters of this style are very simple. Rhythms were straightforward, typically including only two or three different note lengths within a phrase. The melodic style was as simple as folk music, and it often assumed a singable quality, with phrase lengths about the duration of one breath. The harmonic structure relied heavily on the primary triads. The internal relationship of chord tonality and structure were always in perfect agreement. The determination of tonality was also simple. Haydn's Sonata No. 1 is a model of simplicity, a typical music from classical period. Besides Haydn, famous Composers from this period include Mozart and Beethoven. Haydn was fond of using the parallel minor within a major key context. Mozart was much less adventurous than Haydn in the matter of formal experiments. A theme of Mozart usually is complete in itself, and his invention is so profuse that sometimes he dispenses with a formal development section altogether and in its place, presents a complete new theme.

2.4.2 Romantic Music and Chopin

"Romantic music" started in the second quarter of 19th century and influenced the musical work ever since. Those musicians in this period enriched the gallery of music. In this period, character piano pieces [Thompson], symphonic poems and orchestrations had magnificent development and were the immortal achievements of music.

Romantic music is rich with the rhythms, harmonies, forms and melodic traits of national popular music. The music basically is inspired by national idioms. It seeks to express feelings or tell a story [Grout].

Famous Composers in this period include Schubert, Weber, Berlioz, Schumann, Chopin, Liszt, Wagner, Franck, Brahms, Bruckner, Tchaikovsky.

2.4.3 Impressionism Period and Debussy

"Impressionism", in relation to music, is defined in Webster's Dictionary as "A style of composition designed to create descriptive impressions by evoking moods through rich and varied harmonies and timbres." The Impressionistic period flourished at the end of 19th century and the beginning of the 20th century while Classicism and Romanticism were at their developing peak. Debussy started a new musical style to escape Beethoveneous [Thompson] pressure and Wagner's heroism, used color instead of pressure, and used "atmosphere" instead of the hero's self-story telling.

The general musical style in this period attempts to evoke a mood or an "atmosphere" with the help of sounds, rhythms and melody. It relies on allusion and understatement. One element in impressionistic music is "color": color is not only in the narrow sense of timbre, but in the broader sense as rising from harmonic, melodic, and rhythmic factors as well. Melodies are likely to be short motives of narrow range, freely combined to make a musical mosaic of irregular, varicolored pieces. Rhythm is "nonpulsatile, vague, and concealed by syncopations and irregular subdivisions of the beat" [Cooper].

Famous Composers in this period include Debussy and Ravel. Debussy achieved his color effects mainly by means of harmony. One basic factor in his harmonic idiom is the use of chords in a largely "nonfunctional" manner: that

is, chords are not used to shape a phrase, instead, each chord is conceived as a sonorous unit in a phrase whose structure is determined more by melodic shape or color value than by the movement of the harmony. Such a procedure does not negate tonality, which, indeed, Debussy sometimes maintains in ways that is too simple, for example, frequent returns to the primary chords of the key [Cooper]. One thing needs to note is that "impressionism" is only one aspect of Debussy's style; in many of his compositions there is little or no trace of it - for example (among piano music), the early Suite Bergamasque, the Suite Pour le Piano and the delightful children's corner.

2.4.4 Twentieth Century Music and Anton Webern

The first half of the twentieth century witnessed a progressive breakup of the system of music that had prevailed over the preceding two hundred years, roughly from Bach to Richard Strauss. Schoenberg, at first intuitively and later methodically with his twelve-tone rows, had introduced a new conception of musical structure and with his "emancipation of the dissonance" had in effect simply abolished the traditional distinction between consonance and dissonance [Cooper].

The main four factors in twentieth century music are: dissonance, the absence of a tonal center, the compression of style, and the feeling that this kind of music is cerebral, that is, mechanically constructed and without emotional significance [Grout].

Famous Composers in this period include Scheonberg, Alban Berg, and Anton Webern. The work of Schoenberg is on the boundary of the Western musical tradition. The continuity of his thought with that of Bach, Mozart, Beethoven, Brahms, and Wagner is genuine, but the radical break is equally a fact: what

Schoenberg did in music could not be repealed. Alban Berg is Schoenberg's famous pupil, who adopted most of his master's methods of construction, but he used them with freedom and often chose tone rows that allowed for tonal-sounding chords and progressions in the harmony. Moreover, Berg combined the technique with a warmth of Romantic feeling so that his music is more readily accessible than that of many twelve-tone composers. I originally chose Berg for testing my system but because I could find only one piece of his, I switched to Webern to represent this period for testing. Anton Webern is also a famous musician in Twentieth century, who stimulated a movement which came to be associated with a group of young composers for the "new music" [Cooper].

3. Background Concepts on Pattern Recognition

3.1 Pattern Recognition

Recognizing patterns occupies a large part of the daily activity of most humans. Pattern recognition, in this sense, may range from the straightforward identification of a familiar object in everyday use to the detection of trends in a set of data or to finding one's way about a strange environment [Kolers]. The computer scientist interested in developing pattern-recognizing machines always tries to make a machine that will emulate a human perceiver since humans are, by far, the most sophisticated pattern-recognizing devices. Besides, the knowledge of how humans perceive may aid the scientist in his research.

In the study of artificial intelligence, pattern-recognition research is one of the most difficult but important areas. It is concerned with character recognition, language recognition and voice recognition, etc. More recently, the studies of discrimination learning in random nets, perceptual learning in human beings, almost all intellectual tasks related to human's cognitive behavior has been touched and considered.

The study of pattern recognition can be delineated by distinguishing between the recognition of man-made patterns (patterns represented by a symbol in some conventional symbol system) and natural patterns (patterns represented by a classificational label applied to a set of natural objects [Eden]). In the first instance, the problem is to make a correct identification of a symbol such as a chemical symbol or musical notation; in the second instance, it is to make a correct identification of an object of the real world such as fingerprints, clouds and diseases [Eden]. In both kinds of pattern

recognition, the principle function may be characterized as reduction of complex environments. Neither the human nor the computer can afford to deal with each event as a special case. A small set of information structures and processes can be made powerful if there is a device available for recognizing when and where this set is applicable or relevant [Kolers].

NATURAL PATTERN CLASS

There is no common methodology for the recognition of natural patterns, but there appears to be a unifying objective. First, the pattern classes must be defined, and it is this problem that presents serious difficulties. When a satisfactory solution is found for it, the application of an assignment procedure to an actual set of objects may well become a logical triviality even though a number of technical difficulties remain. For example, if it is the clouds to be identified, the most difficult thing is to define "clouds", that is to give characteristical descriptions [Kolers] of the "clouds" pattern class. Because there are probably many characteristics of a single pattern class, the more distinctive characteristics are classified, the easier the pattern to be recognized. However, as mentioned above, in order to reduce complex environment, not all characteristics can be considered, we have to make some choices. The choices of characteristics will be discussed in 3.1.1.

CONVENTIONAL MAN-MADE PATTERN CLASS

Likewisely, in the conventional pattern classes recognition, the pattern class needs to be defined too, besides this, there are some other properties in this kind of systems. (i) The number of symbols is usually finite, (ii) the recognition rules are also finite, and there are probably redundancies in the pattern class because redundancy is essential to error correction [Eden].

For example, in musical notation, the lines that mark the measures can be said to be redundant because the time is specified at the beginning of the piece. However, there are many more subtle dependencies: the tonality, the melodic line, the dynamics, the tempo, etc.

Another property in which conventional symbol systems has is that the recognition rules can continue to grow and change to self-improve the system which is called learning process and will be discussed in 3.1.3.

3.1.1 Choice of Characteristics of Pattern Class

Because one can not consider all characteristics of a pattern class, choices need to be made to do recognition. The choice of appropriate characteristics will insure or at least serve to help solving problems [Kolers]. The choice of characteristics to be analyzed always is conditioned by the particular problem to be solved. The pattern recognition specialist depends heavily on explicit problem-related scientific theory, or sometimes, his knowledge to decide which of the features that are distinctive when judged by human perceptual skills can be used for recognition. For example, if musical notes is the pattern class to be recognized, the researcher can emphasize different characteristics according to different applications, such as problems in composition, contrapuntal method might be a choice, and for problems in some music theory analysis, rhythm and form are probably good choices.

3.1.2 The Problem of Weighting characteristics

After the pattern class was well defined and the characteristics of it were chosen, there is still weighting problem. Just as we do regression analysis, suppose one wishes to estimate the slope coefficient of a linear

regression function. If the error terms associated with these observations are from normal distributions with the same mean and different variances. A weighting scheme is used to get estimator of the slope coefficient. Basically, the higher the variance is, the lower the weight for that observation would be. Since higher variance implies the observation is less informative. For our case, we can weight all characteristics the same, that means, all characteristics chosen are equally important (useful). But this is futile sometimes. We have to weight those characteristics according to their effects to the problem solving. In other words, a rule must be prescribed to assign proper weight to the characteristic. Usually we can assume which characteristic are more important and set the weights in advance, and we can change them by detecting the performance, which is called self-adaptation or learning process.

3.1.3 Adaptation (Learning)

Learning in a computer system can be defined as the process by which the system increases its knowledge and improves its skills. Several categories of learning can be seen in AI studies: (i) Rote learning. The program memorizes all facts and data. (ii) Learning by example. The program learns from example and counterexamples of a concept. (iii) Learning by analogy. The program transforms or augments existing concepts to understand similar concepts. (iv) Learning from observation and discovery. The program classifies and builds new concepts in the process [Winston 1975, 1978,1980].

A lot of pattern recognition work has been carried out on self-adaptive or learning systems. A self-adaption system is closely related to a feedback one, since it improves its performance essentially by measuring how far it is from perfect (difference), and then making changes to improve its performance. To

do this, it must measure performance. Many self-learning systems start with a given fixed preprocessor (hence a given decision space)[Binnie and Simith and Batchalor] and work simply either by modifying the rules for associating the decision space to classes, or where the regions are implicit in a decision procedure rather than explicit, by altering the procedure. However, if this first stage of analysis is wrong the regions resulting are too complex for any reasonable adaptation procedure to operate. It is often best to use a self-adaptive system of a hierarchical kind. The use of high-level constraints to modify lower-level decisions to remark them is an example of "rapid self-adaptation".

According to [Binnie, Simith & Batchalor], it is possible to carry out the adaptation process at several levels. Every level of analysis is a decision process working on "features" detected at a lower level. Its purpose is to "recognize" the features which are the raw material of a higher-level recognition system. Thus, analysis at a higher level can indicate that feature recognitions carried out at a lower level were erroneous. As a response, the overall recognition process could set in motion a "rapid" adaptation. But the evidence also can suggest long-term modification to the decision rules, or that the actual features being looked for at the lower level are wrong and hence cause another level of adaptation. Even longer-term evidence can be accumulated to indicate that the rules for determining constraints are also wrong and are in need of modification.

Most systems that purport to use just one level of feature are in fact multileveled, since the more preprocessors, in effect, carry out several levels of feature analysis [Binnie, Simith and Batchalor]

In this thesis, the development of the basic concepts of learning is from [Uhr and Vossler]. In [Uhr and Vossler], in order to be capable of processing new classes of patterns, the system is designed to adjust its own classification tools when doing pattern classification (recognition). If relate to the preprocessor (decision space) described in [Binnie,Smith & Batchalor], quite a few preprocessors (one for each characteristic) in my system are built for carrying out the adaptation process.

In my system, a weight manipulator evaluates the performance of each characteristic analyzer so that the system gains knowledge about which characteristics are valuable indicators. According to the results of the performance evaluation, the weight manipulator adjusts the corresponding weights for each characteristic decision space.

Three weight manipulation methods were tested. The first method was similar to that used by Uhr and Vossler. The adjustment was made by taking each characteristic individually and comparing the term-value for the characteristic between the real composer and another composer. If the term-value gave a correct (positive) prediction to the identification, the weight for this characteristic for both composers were incremented, otherwise the weights were decremented. The weights were initialized to some arbitrary value for each composer and were allowed to climb arbitrarily high for those terms that consistently predicted the correct pattern; however, a weight for a characteristic could fall no lower than 0, since a negative weight would imply that a characteristic usually produced an incorrect judgement for the real composer relative to the incorrect composers. A weight of 0 essentially removes such characteristics from consideration [Biles 1973].

The other two weight manipulation methods are using the same concepts as the first method but the learning process is accelerated by different techniques.

3.2 Pattern Matching Algorithms

Two important pattern matching algorithms were used in the thesis. They are a Markov Chain and Deterministic Finite Automaton.

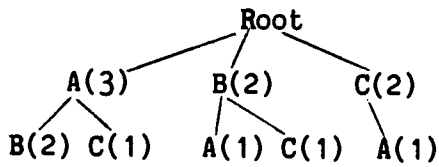
3.2.1 Markov Chain

As mentioned in Yasuaki and Seiji's system (Section 2.3), melody may be considered as a finite Markov process [Yasuaki & Seiji], in which the state k_i is the ith pitch in a melody. The pitch transition probability, p_{ij}, is the conditional probability that the note k_i is followed by k_j . The weight of a note in the twelve-tone scale can be expressed by its steady state probability W (it is like a weight vector). The weight of pitch is effective in expressing the similarity between different melodies belonging to the same scale class. Building a Markov chain can help to solve the equation. Figure 3-2 is a simple diagram that represents a sample Markov Chain for a small piece of music. The chain stores all notes occurring in the music input and all pairs of notes (k_i k_j) in the music input and their frequencies, so that we can use them to calculate the conditional probability (p_{ij}) and then solve the equation.

The chain is growing at the same time when it is doing pattern matching. Whenever a new pattern is matched, the pattern will be put into the chain and the frequency for the pattern will be updated. It is a very useful pattern matching algorithm.

Figure 3-1 A simple diagram to represent the Markov Chain

a piece of music string ABABCAC



A occurs 3 times,
B occurs 2 times,
C occurs 2 times,
AB pair occurs 2 times,
AC, BA, BC and CA occur once.
Total number of pairs occurred = 6
so P_{AB} = 2/6,
$P_{AC} = P_{BA} = P_{BC} = P_{CA}$ = 1/6

The Markov Chain can grow very wide (many children) and very deep (many levels) but I only used it to find the alphabet (Sigma I - please refer to next section) and two-code sequences of the score. Because searching a pattern involves traversing through the chain from the root, the Markov algorithm is less efficient than the algorithm to build a Deterministic Finite Automaton.

3.2.2 Deterministic Finite Automaton (DFA)

As mentioned in chapter 2, one can use grammatical approach to recognize language and analyze music and according to Chomsky [Chomsky 1957,1958][Hopcroft & Ullman], if we looked at a piece of music as a string (or a kind of language), then it should be able to be recognized or generated by a deterministic computational machine (section 2.1.1). In order to find the theme of a piece of music, a two-way push-down automaton was considered to be used to find the longest substring in a string, which longest substring, then, can be used as the theme of the music for analysis. However, I found out almost all music scores point out which portion of the music is the principle

theme or the second theme, so I do not need to design a two-way push-down automaton to search for the theme any more. But, unexpectedly, I found out a deterministic finite automaton (DFA) can be used to recognize musical string just as mentioned in the grammatical approach. So that a DFA was built to recognize the musical string (language).

I referred to the pattern matching algorithms in [Hopcroft & Ullman] to construct a Deterministic Finite Automaton (DFA) for recognizing a substring from a string. The algorithm is as follows:

-- Construction of a DFA for I*y where I = Sigma, y is a pattern string.

Input: A pattern string y = $b_1\ b_2\ \ldots\ b_g$ over alphabet I. For convenience, we take b_{g+1} to be a new symbol not equal to any symbol in I.

Output. A DFA M such that L(M) = I*y.

Method:

i. Construct the failure function f for y.

Failure function:

Input: Pattern = $b_1\ b_2\ \ldots\ b_g$, g >= 1.

Output: Failure function f for y.

Method:

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

    begin
         f(1) <-- 0;
         for j <-- 2 to g do
         begin
               i <-- f(j-1);
               while bj <> bi+1 and i > 0 do
                     i <-- f(i);
               if bj <> bi+1 and i = 0
                     then f(j) <-- 0
                     else f(j) <-- i + 1
         end
    end
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

ii. Let M = (S, I, δ, 0, {g }), where S = { 0,1,2, ..., g } and δ is constructed as follows.

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  begin
       for j = 1 to g
            do δ (j-1, b_j ) <-- j
       for each b in I, b <> b_1
            do δ (0, b) <-- 0
       for j = 1 to g
            do for each b in I, b <> b_j+1
                  do δ (j, b) <-- δ (f(j), b)
  end
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

The DFA continues operating in this fashion, until either it enters the final state g (pattern length), in which case we know that the last g input symbols scanned constitute an instance of the pattern $y = b_1\ b_2\ \ldots\ b_g$, or the DFA has processed the last input symbol input string x without entering state g, in which case we know that y is not a substring of x.

4. System Specification

The system was designed and implemented under the UNIX operating system, and a shell script was used to connect the main functional modules -- the Characteristics analyzers (written in C) and the Classifier (written in Prolog) to execute as one whole system.

To test the system, a total of twenty music pieces were encoded and used, and 51 musical characteristics were observed for each composer with 51 corresponding weights normalized and manipulated. The primary modules of the system are shown in Figure 4-1.

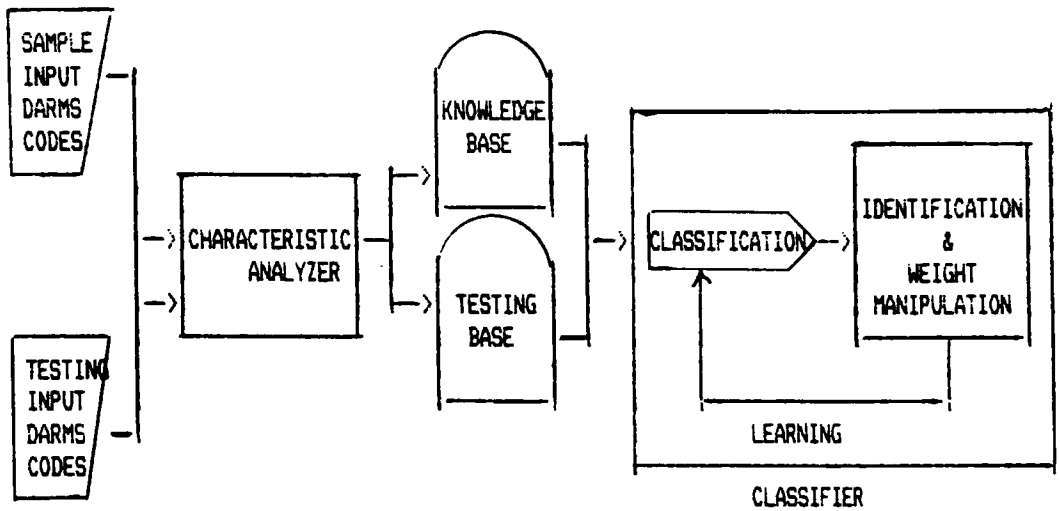
(4.1) Sample input DARMS codes and Testing input DARMS codes: I manually encoded sample musical scores and testing musical scores to system-understandable DARMS codes. To correct for different key signatures, separate program was written to transpose all pieces to the key of C, and then the transposed codes were fed into the Characteristics analyzer.

(4.2) Characteristics analyzers: The system consists of seven subanalyzers that determine musical characteristics and styles.

(4.3) Knowledge base and Testing data base: Each subanalyzer mentioned above creates a data base, and they are all used by the searching, parsing, matching, recognition and inferential processes involved in identification.

(4.4) Classifier: The classifier makes connections and does comparisons between an unknown music piece (from the testing data base) and known sample music pieces (from the knowledge bases) to give the rating scores for each composer. The system then identifies the composer for an unknown testing piece according to the classification results. Finally, a weight manipulator modifies the internal classification tools to hopefully enhance classification.

Figure 4-1 The simplified implementation processes



The characteristics analyzer and the classifier perform major functions. The purpose of the characteristics analyzer is to find musical characteristics that represent understandable analytical forms. From these characteristics, the classifier classifies and matches different characteristics and identifies the composer for an unknown piece. In order to give readers a complete view of the system, a more complete description of each module follows.

(4.1) DARMS Accepter

As mentioned above, all pieces were manually encoded into DARMS codes and transposed to the key of C. A DARMS accepter was designed to manage the DARMS codes.

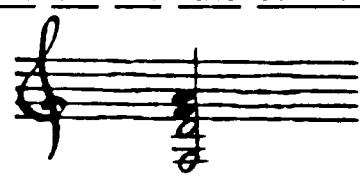
Because the left-hand playing (score2_part chords) in piano playing is heavily related to the rhythmic structure of the music, it was analyzed separately from the right-hand-part, and the right-hand and left-hand scores were constructed separately. The DARMS accepter uses two arrays of pointers to store the two parts. One array stores the right-hand-part score (score1_part), the other array stores the left-hand-part score (score2_part). Each pointer in the array points to a 5 X 3 array where the information on concurrent notes is stored.

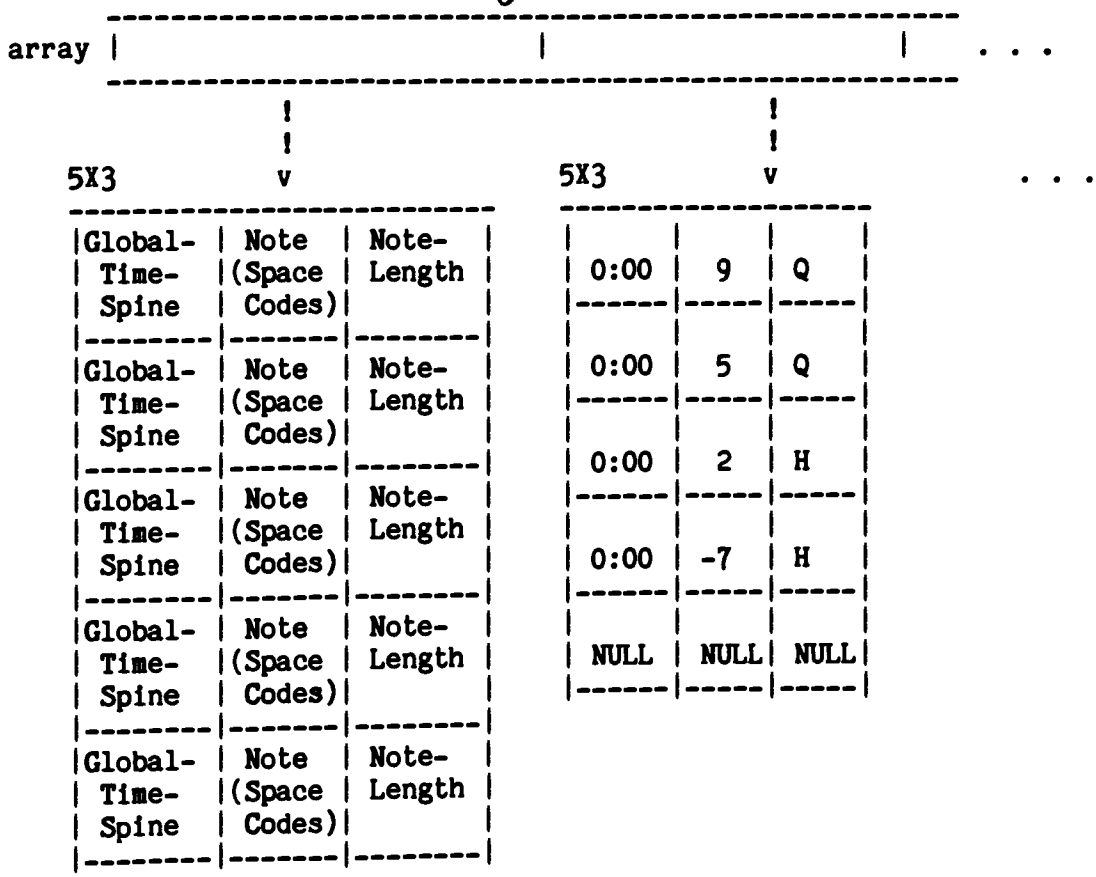
The whole structure is explained in more detail in Figure 4-2. The example says that at time spine 0:00 (starting time is set to be 0:00), note 9 (pitch A) with length Q (quarter note), note 5 (F with length Q), note 2 (D with length H (half note)) and note -7 (F with length H) occur in the score. More formally, the 5 X 3 two-dimensional array contains (i) The global-time-spine, which is a time-line for the score and tells the accumulated time spine for the score at the current time spot; (ii) the note field containing DARMS

codes that occur at this time spine, and (iii) the corresponding note-lengths of the DARMS code. This provides temporal information (i.e. the duration of notes).

Because the right hand part and the left hand part are stored separately, up to 10 concurrent notes can be stored in the array. When the system was doing harmonic analysis such as dissonance analysis, it traced two lists at the same time, merging them into a single time-line list.

Figure 4-2 The Data Structure of DARMS Accepter



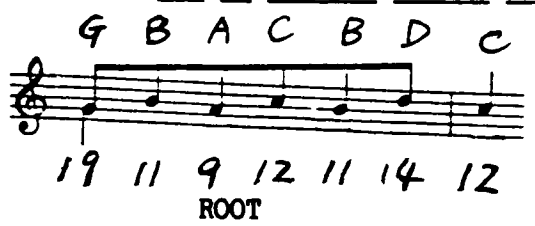


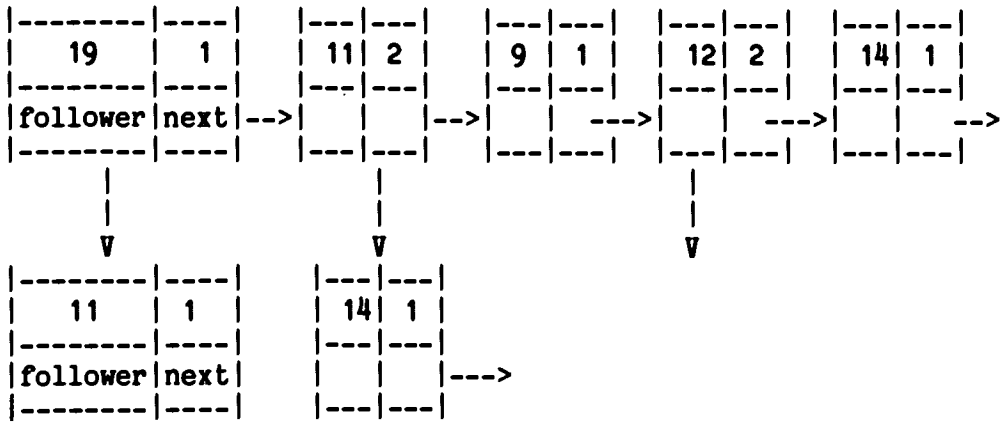
(4.2) Stylistic Analyzer

As mentioned in chapter 3, two pattern matching algorithms were used in this project. They are Markov Chain and Deterministic Finite Automaton.

The Markov Chain implemented is a tree-structured linked list with each node containing an information field, a frequency field, a next node pointer and a follower node pointer. The information field contains DARMS code. The frequency field represents the number of occurrences of the code. The next pointer points to the node which contains the information of next newly occurred code in the input string, while the follower pointer points to a linked list which contains information of the codes following the code in this node. Figure 4-3 is a formal example of Markov chain, for better understanding, please also refer to Figure 3-1.

Figure 4-3 An example Markov Chain





As shown in the Figure 4-3, the chain is built in a tree structure. Starting from a NULL root node, once the first symbol is read in, a child node is created (with information = symbol, frequency =1) and linked to the root's follower linked list. After this is done, whenever a symbol is encountered, the system traverses the linked list to see if this code is in the chain or not. If it is in the chain, then the system increments the frequency field of that note. If it is not in the chain, a new node is created and added to the end of the root's follower linked-list. The chain grows until the end of the input string is reached. This is the first level of the Markov Chain which contains all occurred symbols that is the alphabet (Sigma I) of the score string, and the occurrences of each symbol in the score.

The second pass of building the Markov Chain was to find two-code sequences in the score and store them in the chain. When a symbol (call it symbol A) was read in, the first-level-chain was traversed to find symbol A, then the next symbol in the score was read in (call it symbol B) and the symbol A's follower linked list is checked. If symbol B is not in A's follower linked list, then a new node is created and added to A's follower linked list. If B is in A's follower linked list, then the frequency field in that node is incremented by 1, which then, was the frequency of the two-code-sequence AB field.

Another algorithm used for pattern matching implements a deterministic finite automaton. Basically, I built the DFA according to the algorithm described in section 3.2.2, but I also modified the algorithm to be able to recognize many substrings simultaneously instead of only one substring. All patterns that needed to be recognized were stored in a two-dimensional pattern array, and the corresponding failure function and state transition function δ were built for each pattern and stored in the DFA. The recognition processes were the same as the original algorithm, only the DFA checks every pattern in the pattern array and gets the next state for every pattern so it can recognize many patterns at the same time.

The system uses seven analyzers to make specific observations on three fundamental aspects in a musical work: melody, rhythm and harmony. (Table 4-1)

Table 4-1 Seven Characteristic Analyzers

Melody :
(1) * Melodic Structure Analyzer
(2) * Contour Structure Analyzer
(3) * Contour Height Structure Analyzer
(4a)* Complexity Analyzer -- complexity of melody

Rhythm:
(4b)* Complexity Analyzer -- complexity of rhythm
(5) * Rhythmic Structure Analyzer

Harmony:
(6) * Triad Root-Movement Analyzer
(7) * Dissonance Analyzer

Melody tends to receive more analytical attention than any other component in music, possibly because it separates rather easily and naturally from other factors. The melodic structure analyzer examines repetition, figuration and sequence in small dimensions (e.g. one or a few phrases), and recurrent formulas and phrase structures in larger dimensions (e.g. pieces with a few movements). It is important to discover which recurrent structures have been used in other pieces by the same composer and which are merely idiomatic melodic patterns found frequently in some specific works. The contour structure analyzer abstracts various general curve types of melody by observing the movement by step and studying the location of peaks and underpoints (refer to Figure 4-5). The contour height structure analyzer observes the intensity resulting from rising line (curve) and heightened activity. One less tangible method than the above three methods is the complexity of melody [Yusuaki & Seiji], which expresses the effect of melodic transitions.

No other stylistic component offers more difficulties than rhythm, owing to a lack of adequate analytical methods to attack this infinitely varied aspect of music. As an example of these difficulties, consider two bars of common time, one consisting of two eighth-notes and three beats of rest, the other filled out by two half notes (Figure 4-4) Each bar has two impacts - should such bars be considered equal for analytical purpose? Satisfactory answers do not seem to exist yet.

Figure 4-4 An example to show the difficulties of rhythm

In this thesis, three accessible factors of rhythm were more or less considered: the details of rhythmic structure (most frequent rhythmic patterns), the dimension of rhythmic activity (e.g. frequencies of quarter notes, eighth notes, etc.) and the complexity of rhythmic setting.

In the analysis of harmony, this thesis concentrates on two things. One is the treatment of dissonances in chord alternation or in non-harmonic notes, which reflects a composer's individuality. The other is the dimension of triad movement, which is also valuable because it provides a fundamental view of chord movement.

(4.2.1) Melodic Structure Analyzer (Analysis Model)

Only pitch notes in the score (Space Codes in DARMS code) were used for melodic structure analysis, and they were called melodic strings. Pattern matching techniques were used to search for melodic patterns in these melodic strings. The search begins with the shortest melodic patterns (2-note patterns), since these shorter patterns can be used to derive longer patterns and the system can use them for further pattern recognition.

When doing this analysis, the right-hand melody and left-hand melody were retrieved from the score1_part and score2_part, respectively, and fed into markov chain builder to get the alphabet (Sigma I) and the five most frequent 2-note-patterns of the melody strings. With these most frequent 2-note-patterns and the alphabet, the system created all possible 3-note-patterns and fed them into the DFA for 4-note, 5-note, 6-note, 7-note and 8-note pattern recognition. These patterns, then, were printed out in a special format in an output file as data base for future classification use.

(4.2.2) Rhythmic Structure Analyzer (Analysis Model)

In piano music the left hand part score is heavily related to the rhythmic structure, and the lengths of notes affect the rhythmic setting even more, so that for this part of analysis, the note-length (Duration Codes in DARMS Code) were used for analysis. The rhythmic analysis processes are similar to melodic structure analysis. The only difference is that note lengths instead of pitches were retrieved for analysis.

(4.2.3) Contour Structure Analyzer (Analysis Model)

Melodic contour refers to shape and to the physical placement of pitches. By superimposing graphic or geometric designs over a staff, without the benefit of pitches and duration, one can create a basic outline that

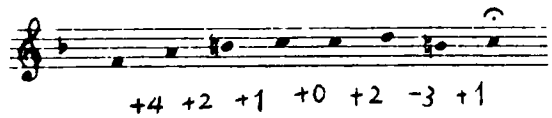
illustrates the concept of melodic contour. (Figure 4-5)

Figure 4-5 An example of the shape of contour

The analysis process is also similar to melodic analysis. Those notes (Space codes) in the main scores were retrieved and evaluated to get the contour strings (Please refer to Figure 4-6).

Figure 4-6 An example of contour evaluation

DARMS codes 5 9 11 12 12 14 11 12



C_i is contour code

$C_1 = 9 - 5 = +4$ $C_2 = 11 - 9 = +2$

$C_3 = 12 - 11 = +1$ $C_4 = 12 - 12 = +0$ $C_5 = 14 - 12 = +2$

$C_6 = 11 - 14 = -3$ $C_7 = 12 - 11 = +1$

Since the contour code does not differentiate the quality of intervals, all seconds are represented by the integer 2, all thirds, whether major, minor or diminished, by the integer 3, and all fourths by 4. Chromatic inflections of the same pitch are represented by the integer 1. A plus or minus sign

preceding the integer indicates the direction. Repeated notes are coded as '+0'. During the calculation process, compound intervals are reduced to simple ones. For notes occurring at the same time spot, the shortest one was used.

(4.2.4) Contour Height Structure Analyzer (Analysis Model)

This analyzer is a little complicated. The system gets the melodic contour string and partitions the phrases of the contour string into some possible subsets such as 4, 5, 6, 7, 8-codes patterns. (see Figure 4-7)

Table 4-7 An example of contour height evaluation

[+4 +2 +1 +0 +2 -3 +1]	8-notes contour, average height = 13/7
[+4 +2 +1 +0 +2 -3]	7-notes contour, average height = 2
[+2 +1 +0 +2 -3 +1]	average height = 9/6
[+4 +2 +1 +0 +2]	6-notes contour, average height = 9/5
[+2 +1 +0 +2 -3]	average height = 8/5
[+1 +0 +2 -3 +1]	average height = 8/5
.	
.	
.	

Then the system calculates the moving average height of these contour subsets, which is:

$$H_n = \sum_{i=1}^{n} |c_i| / (n-1)$$ where c_i is the melodic contour code

For example: the average height of 8-note contour subset is

$$H_8 = |4|+|2|+|1|+|0|+|2|+|-3|+|1| / (8-1) = 13/7$$

$$13/7 * 100 = 171$$

The actual values were multiplied by 100 to magnify the difference.

The absolute value of each contour code is used to calculate the average contour height because only the magnitude matters, not the direction. These moving average height values result in a moving-average contour height string. The system uses these contour-height strings to do pattern matching and gets the five most frequent average height for 4-code, 5-code, 6-code, 7-code and 8-code contour-height.

(4.2.5) Complexity Analyzer

I also referred to the analytical method used in Yasuaki and Seiji's music information processing system [7] to calculate the complexity of melody and rhythm. These are defined as in section 2.3 and in my project, the complexity of melody is defined as:

$$\delta_m = \mathrm{sqrt}[\ (1/n) * \sum_{i=1}^{n} (S_i - S_{i+1})^2\ ]$$

where S_i is the DARMS space code

The complexity of rhythm.

$$\delta_R = (1/n-1) * \mathrm{sqrt}(\ \sum_{i=1}^{n} R_i\ )$$

$$R_i = \log_2(\ D_i / D_{i+1}\ ) = \log_2(\ D_i\ ) - \log_2(\ D_{i+1}\ )$$

where D_i is the DARMS Duration code

(4.2.6) Triad Root-movement Analyzer

In this model, all triad root-movements were checked, classified and recorded in the class matrix of five classes : the seconds, thirds, fourths, fifths and all others (see 9.1 Glossary).

(4.2.7) Dissonance and Imperfect Consonance Analyzer

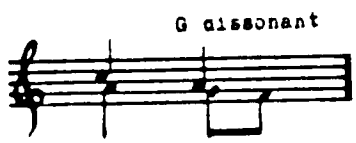
Consonance and dissonance are terms used to describe the effect of two or more simultaneous or contiguous sounds. Typically the distance (interval) between two pitches is defined in one of the following three classes (Figure 4-8):

1. Perfect consonances : unisons (primes), fourths, fifths, octaves
2. Imperfect consonances : third, sixths (tenths, etc)
3. Dissonances: seconds, sevenths (ninths, etc)

Figure 4-8 examples of Intervals

With these definitions, the system determines pitch-properties of voice-leading patterns resulting in dissonance and checks their occurrences in the musical score. The following is an example of dissonance. (Figure 4-9)

Figure 4-9 An example of dissonance



4.3 Analytical Knowledge Bases and Testing Bases

After each subanalyzer described in section 4.2 analyzed the fixed-format DARMS codes (musical data), a new data base is created from them. The data base keeps track of the "meaning" of the music data and performs inferences to determine what information is needed for a given task, even if this task has not been explicitly specified. It also contains the ability to view objects from multiple perspectives, for example, viewing the rhythmic structure and melodic structure of a melody in a certain transposition. These two characteristics - inference from partial description and multiple perspective - are accomplished through procedural specifications embedded in the data base, and make it possible for the system to reason about musical structure and the interrelations among instances of musical concepts. For distinguishing sample data from testing data, the data base created from a sample piece is called a knowledge base. Every pattern structure in the knowledge base has the composer's name and the sample mark with it. The data base created from a testing piece is called testing data base and the name of each pattern in it has a "test" marker preceding the original name to tell it is testing data.

The following is part of a knowledge base:

```
complex(a, mozart, m_alpha, 80.41).
complex(a, mozart, r_alpha, 0.02).
```

The functor, "complex", indicates a pattern from the complexity analysis. The attribute "a" represents the sample music mark (There are marks "a" for the first test sample, "b" for the second sample and "c" for the third sample, see table 5-1). "mozart" indicates the real composer for this piece. "m_alpha" denotes complexity of melody, while "r_alpha" denotes complexity of

rhythm. Their values are 80.41 and 0.02, respectively, which means the complexity of melody of this piece is in the higher range in the possible range of 0 - 100 and the complexity of rhythm is in the lower range of 0.00 - 0.10.

```
struct_patt(a, mozart, contour, 2, [-2,-1],[2,-2],[1,2],[2,1],[2,2]).
struct_patt(a, mozart, contour, 3, [-2,-1,-2],[2,-2,2] . . . ).
struct_patt(a, mozart, contour, 4, [-2,-1,-2,-2] . . . . . ).
struct_patt(a, mozart, contour, 5, [2,-2,2,-4,-2]. . . . . ).
struct_patt(a, mozart, contour, 6, [2,-2,2,-4,-2,2]. . . . . ).
```

There are five different structure patterns: contour, contour height, right-hand-melody, left-hand-melody and rhythm. The Prolog rule name for structure patterns is "struct_patt", but there is a structure type to tell which type the pattern is. For example, the above structure patterns consist of a pattern from the "contour" structure analysis; the composer for the sample piece is "Mozart"; and the sample mark is "a". It is a "2"-code pattern, and the five most frequent contour patterns in this sample piece are [-2,-1], [2,-2],[1,2],[2,1] and [2,2]. We can see that for this sample piece, the most frequently used contour is in the range $\pm$ 2, meaning that it is rather smooth.

The testing base is identical to the knowledge base except that the knowledge base tells the real composer while the testing base does not. The following is part of a testing base.

```
test_complex(m_alpha, 79.02).
test_pattern(contour, 2, [0,0],[-4,-3],[-5,12],[-3,-5],[-2,-2]).
```

"test_complex" indicates a testing pattern from the complexity analysis. The pattern is complexity of melody (m_alpha), and its value is 79.02. "test_pattern" indicates a structure pattern; the pattern type is "contour" and a 2-code pattern with the patterns being [0,0], [-4,3], [-5,12],[-3,-5]

and [-2,-2].

4.4 Classifier

The classifier is written in Prolog, and it does the classification in the following steps: (Figure 4-10)

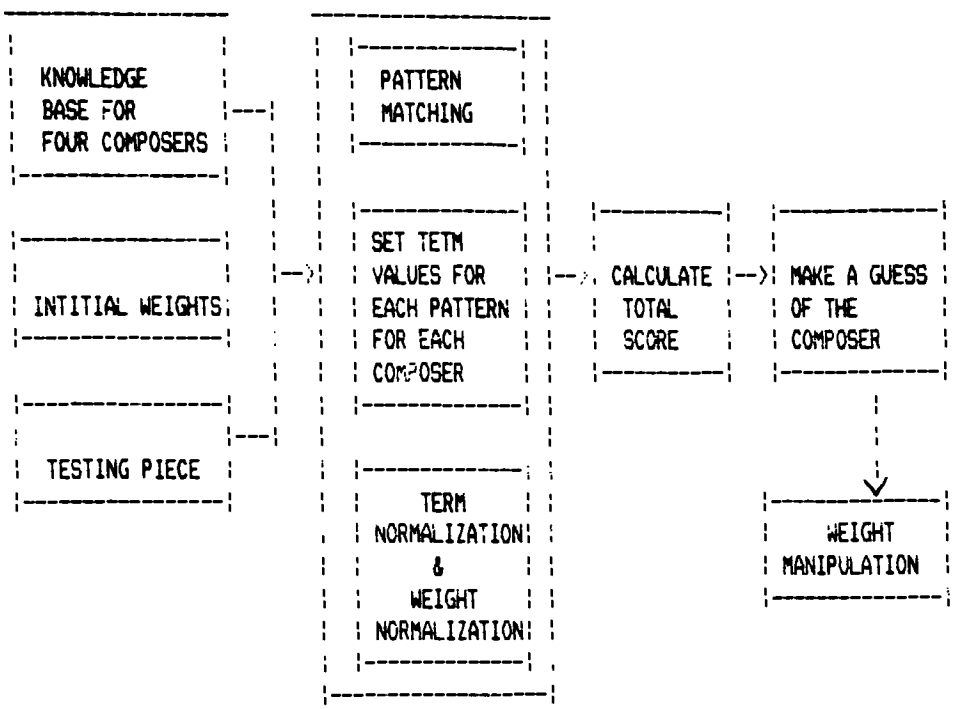
(1) In the initial run, the classifier has to load the knowledge bases for the four composers - Mozart, Chopin, Debussy and Anton Webern.

(2) The classifier asks for a testing piece to consult. If it gets a valid answer, the classifier will load the testing piece, otherwise it will wait for another answer.

(3) After the classifier has loaded the testing data base, it assigns initial weights for each pattern and starts to do the pattern matching, which includes melodic structure, rhythmic structure, contour structure, contour height, dissonance, triad root movement and complexity pattern matching.

(4) The classifier sets the term value for each pattern according to the following term determination rules.

Figure 4-10 Structure Chart of Classifier



For each pattern, the term values are set according to the significance of each characteristic and the pattern matching results. The evaluation standard is determined by experience. For melodic, rhythmic, contour and contour height pattern, the term value is set to be 1 for each matched pattern. For example, if testing piece A has the same 4-code rhythmic structure as Mozart's sample knowledge base, the term value for this pattern for Mozart is incremented by one. The same evaluation is executed for other composers. For complexity patterns, the term value is set by its range. For example, if the complexity of melody of testing piece A is within the range of $\pm$ 5% of the complexity of melody of Chopin's knowledge base, that means the complexity of melody of these two pieces are very similar, so that the term value for this pattern for Chopin is incremented by two. If it is in the range of $\pm$ 10%, that means the complexity is not close enough, so that the term value is incremented by one. If it is out of $\pm$ 10% range then the term value is unchanged. For triad pattern, the number of each class of triads root-movement in the testing score is compared with the corresponding number in the sample music to decide term value. If both numbers are 0, that means no such root-movement in both scores, the term value is set to be 1. If both numbers are equal, the term is set to be 2. If the difference between these two numbers is in the range of $\pm$ 5, then the term is set to be 1. For dissonance pattern, if the dissonance numbers in both the testing piece and knowledge piece are 0 or greater than 5 (this margin number is set because composers other than Webern do not use more than 5 dissonances in their music works) the term is set to be 2, but if one of them is 0 and the other is greater than 0, then the term value is set to be 0. For imperfect consonance pattern, the term is determined similarly to dissonance pattern, only the margin number is much bigger, it is set to be 40, i.e. if the imperfect consonance number in both pieces are greater than 40,

then the term value is set to be 2.

(5) After setting the term values, the classifier then has to do term and weight normalization.

A term-normalizer normalizes each term. The individual term for each characteristic was normalized to correct for heterogeneous ranges of values for different characteristics. This was done by dividing the difference for a characteristic for the composer by the mean difference for that characteristic across all composers [Biles 1973]. This yields a mean normalized value of 1 for each characteristic independent of the actual term value in the data base. The following formula gives the normalized j'th term for composer C.

$$\text{Normalized term} = \frac{\text{the jth term of composer C}}{\text{(the Sum of jth term of each Composer) / No. of composers}}$$

The range for each characteristic is further normalized to [0,2] by setting any normalized differences greater than 2 to be exactly 2.

There is also a weight-normalizer to normalize the weight for each term [18]. The weights were also normalized by dividing the weight for a characteristic by the total sum of the weights across all characteristics for the composer. The following formula gives the normalized weight for the j'th term of composer C.

$$\text{Normalized weight} = \frac{\text{the weight for the j'th term of Composer C}}{\text{the Sum of weights for the composer}}$$

The sum of the normalized weights for every composer is 1, and an individual normalized weight is the "propotion of importance" of that characteristic relative to the other characteristics of that composer. The sum of the products of the normalized terms and the normalized weights, then, yields an average normalized weighted score in the range [0,2].

For the entire system, there are 51 terms with 51 corresponding weights to be normalized.

(6) The classifier multiplies each normalized term-value by its corresponding normalized weight and adds all these products to get the total score for each composer.

(7) According to the total score, the classifier is able to make a guess about who is the composer for this unknown testing piece. After making the guess, the classifier passes its results to a weight manipulator.

(8) As indicated in chapter 3. the weight manipulator adjusts the weight of terms to reflect their performance. Three weight manipulation methods tested in the thesis are method1, method2 and method 3. (Table 4-2)

The initial weights were all set to 10 for each composer for each method. In method 1, if the term-value for a characteristics gave a correct prediction to the identification, the weights were incremented by 1, otherwise the weights were decremented by 1.

In order to accelerate the learning process, the second weight manipulation method is to divide the weight by two if its term value gives a incorrect prediction to the total score and the weight is multiplied by two if its term value gives a positive prediction.

The third weight manipulation method was a combination of the other two methods. The changes in weights were decided by previous experience. For each pattern, if, in the previous run, the term was a positive factor, then it is multiplied by two given that it is still a positive factor in the current run, otherwise it is incremented by 1. Similarly, if the pattern was a negative factor in the previous run, then its weight is divided by two given that it is still a negative factor in this run, otherwise it is decremented by 1. The three strategies are summarized in Table 4-2.

table 4-2 Brief Description of 3 Weight Manipulation Methods

weight manipulation methods	current run positive factor		current run negative factor	
	previous run positive	previous run negative	previous run positive	previous run negative
Method 1	weight + 1	weight + 1	weight 1	weight - 1
Method 2	weight * 2	weight * 2	weight / 2	weight / 2
Method 3	weight * 2	weight + 1	weight 1	weight / 2

After weight manipulation, the classifier closes the currently consulted testing data base, clears all useless terms, and asks for next testing piece. The whole processes continues until no testing pieces remain.

5. verification and validation

A total of twenty music pieces were encoded and tested in the system and the results came out fairly satisfactorily.

5.1 Test Plan

The testing was done in two phases. In the first phase, I randomly chose four piano pieces for each composer (Mozart, Chopin, Debussy and Webern), and one more piece by a different composer from each period (Classical, Romantic, Impressionism, and Twentieth Century) to test the system's performance. Haydn, Schumann, Ravel and Alban Berg were chosen against Mozart, Chopin, Debussy and Anton Webern, respectively. The pieces are listed in Table 4-1. The initial knowledge base (vectors of characteristic values) was derived from three samples by each composer.

In the second phase, I tested each piece in a different position using different weight manipulation schemes. For each manipulation scheme, the system was tested in the following ways: (i) Classify testing samples from every piece in the first run (with the initial weights, no learning); (ii) Classify testing samples from every piece in the second, the third or the fourth position in the testing run; (iii) Run all testing samples at once but in a different position; (in the 5th-10th position, 10th-15th position or 15th-20th position).

Table 5-1 Tested Music Pieces

```
Classic Period:
  Mozart (1756-1791)
      Sonata I                  *(1 test, 1 knowledge (a) )
      Sonata IV                 *(1 test, 1 knowledge (b) )
      Sonata VII                *(1 test, 1 knowledge (c) )
      Sonata XIV (K. No. 310)   *(1 test, 0 knowledge     )
  Haydn: Sonata No. 1           *(1 test, 0 knowledge      )

Romantic Period:
  Chopin (1810-1849)
      Grande Valse brillante Op. 18 *(1 test, 1 knowledge (a) )
      Valse brillante Op. 34 No. 1  *(1 test, 1 knowledge (b) )
      Grande Valse Op. 42           *(1 test, 1 knowledge (c) )
      Mazureka Op. 7. No. 2         *(1 test, 0 knowledge     )
  Robert Schumann: Klein Fuge Op.68*(1 test, 0 knowledge     )

Impressionism Period:
  Debussy (1862-1918)
      Suite bergamasque (Passepied) *(1 test, 1 knowledge (a) )
      Pour le Piano (Prelude)       *(1 test, 1 knowledge (b) )
      Deux Arabesques               *(1 test, 1 knowledge (c) )
      Reverie                       *(1 test, 0 knowledge     )
  Ravel: Miroirs (Noctuelles)      *(1 test, 0 knowledge      )

Twentieth Century:
  Anton Webern (1883-1945)
      Sonatenszatz (Rondo) fur Klavier *(1 test, 1 knowledge (a) )
      Satz fur Klavier (c. 1906)       *(1 test, 1 knowledge (b) )
      Klavierstuck (Op. post)          *(1 test, 1 knowledge (c) )
      Kinderstuck (Autumn 1924)        *(1 test, 0 knowledge     )
  Alban Berg: Variationen uber ein eigees thema*(1 test, 0 knowledge)

   *(n test, m knowledge(A)) means n test samples and m knowledge
sample
    samples were retrieved from that piece and the knowledge sample
was
    called as sample "A" in the knowledge base.
```

5.2 Test Results and Discussion

As mentioned in Section 2, three weight manipulation methods were tested in this thesis. For each manipulation scheme, many experimental works were

made for testing the system.

Generally speaking, the system worked satisfactorily. Table 5-2, 5-3 and 5-4 give the overall testing results.

(i) In the first run (no weight manipulation), the results were all the same for three schemes because they were using the same initial weights for each characteristic. (Table 5-2)

Table 5-2 Testing results (initial run)

testing orders	Mozart	Chopin	Debussy	Webern	Haydn	Schumann	Ravel	Berg
	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	1	1	1	1
a. each piece in the first run. (using method 1)	M M M M	C C C M	M D D M	W D W D	M	C	D	M
b. each piece in the first run. (using method 2)	M M M M	C C C M	M D D M	W D W D	M	C	D	M
c. each piece in the first run. (using method 3)	M M M M	C C C M	M D D M	W D W D	M	C	D	M

1. M --> Mozart, C --> Chopin, D --> Debussy, W --> Webern,
   H --> Haydn, S --> Schumann, R --> Ravel, B --> Berg.
2. method 1, method 2 and method 3 are weight manipulation method 1, 2, 3, respectively.

For Mozart's piano pieces, this system proved to be a very good identifier. Mozart was identified 100% correctly. For Debussy's pieces, on the other hand, this system gave a very frustrating performance. Debussy's two pieces were identified with Mozart. Nevertheless, musicology books [Grout] give a good reason for this. "Impressionism" is only one aspect of Debussy's style, and many of his compositions show little or no trace of it - for example, the Suite Bergamasque (1893), the Suite Pour le Piano (1901- I chose this piece). Also, Debussy's harmonic and coloristic traits were similar to classical forms and cyclic treatment of themes. This provides a good explanation for the wrong guess made by the system.

For testing Webern's piano works in the initial run, the second and the fourth pieces were also confusing. They were identified as Debussy's. With better weight manipulation scheme, the second piece can be correctly identified.

For testing Chopin's piano works, I also encountered the same problem. One of the testing pieces was confused with Mozart. I could not find any good reason for that from the viewpoint of musicology, but I think that Chopin's early music probably is not so distinguishable from classical music.

For those pieces chosen for testing the system's performance, the system did fairly good classification. Haydn was identified as Mozart; Schumann was identified as Chopin and Ravel as Debussy, which is exactly what I expected. However Berg was always identified as Mozart, which is not it supposed to be. The musicologist D. J. Grout [Grout] describes Berg's style: "Berg's music is unified partly by the use of a few leitmotifs but chiefly by being organized in closed forms adapted from those of Classical Music." This give some explanation for the unexpected identification.

(ii) For classifying each testing pieces in the second, the third or the fourth position in the testing run (Figure 5-3), for the simplest weight manipulation method (method 1) the result did not change too much. Only the total score for each composer was changed. For method 2 and method 3, however, the results were better. Webern's second testing piece was identified correctly, which is very encouraging. At least, the weight manipulation methods seemed to improve the system's performance.

Table 5-3 Testing results

testing orders	Mozart 1 2 3 4	Chopin 1 2 3 4	Debussy 1 2 3 4	Webern 1 2 3 4	Haydn 1	Schumann 1	Ravel 1	Berg 1
a. each piece in 2nd,3rd,4th order (using method 1)	M M M M	C C C M	M D D M	W D W D	--	--	--	--
b. each piece in 2nd,3rd,4th order (using method 2)	M M M M	C C C M	M D D M	W W W D D	--	--	--	--
c. each piece in 2nd,3rd,4th order (using method 3)	M M M M	C C C M	M D D M	W W W D	--	--	--	--

1. M --> Mozart, C --> Chopin, D --> Debussy, W --> Webern,
   H --> Haydn, S --> Schumann, R --> Ravel, B --> Berg,
2. method 1, method 2 and method 3 are weight manipulation method 1, 2, 3, respectively.

(iii) For testing all the testing pieces in one run but with each piece tested in different position. Since the classifier learned from weight manipulation, the order of testing the pieces could affect the classification results. For pieces with distinctive characteristics, there is no difference, but for some confusing pieces, the later it was tested, the closer to correct the classification turned to be. For example, using weight manipulation method 2, if Webern's second testing piece (piece (b) in table 5-1) was tested in the very beginning of run, then the testing results were;

```
grand_total(chopin 0.122962).
grand_total(mozart 0.16146).
grand_total(debussy 0.237502).
grand_total(webern 0.228793).
```

which shows Chopin's grand total score was 0.122962, Mozart's was 0.16146, Debussy's was 0.237502 and Webern's was 0.228793. The guessed composer for this result would be Debussy because he had the highest total score. If the same piece was tested at the fifteenth position or later of the testing run, then the results were:

```
grand_total(chopin 0.144463).
```

```
grand_total(mozart 0.174794).
grand_total(debussy 0.20972).
grand_total(webern 0.242619).
```

In this case, Webern received the highest total score and he would be correctly guessed as the composer.

Table 5-4 Testing results

testing orders	Mozart 1 2 3 4	Chopin 1 2 3 4	Debussy 1 2 3 4	Webern 1 2 3 4	Haydn 1	Schumann 1	Ravel 1	Berg 1
1.1 each piece in 5th to 10th order	M M M M	C C C M	M D D M	W D W D	M	C	D	M
1.2 each piece in 10th to 15th order	M M M M	C C C M	M D D M	W D W D C	M	C	D	M
1.3 each piece in 15th to 20th order	M M M M	C C C M	M D D M	W W W D D	M	C	D	M
1.4 each piece in the last order.	M M M M	C C C M	M D D M	W W W D D	M	C	D	M
1.5 each piece in specific order and tested last	M M M M	C C C C	M D D M	W W W D	M	C	D	M
2.1 each piece in 5th to 10th order	M M M M	C C C M	M D D M	W W W D D C	M	C	D	M
2.2 each piece in 10th to 15th order	M M M M	C C C M	M D D M	W W W D D C	M	C	D	M
2.3 each piece in 15th to 20th order	M M M M	C C C M	M D D M D	W W W D	M	C	D	M
2.4 each piece in the last order.	M M M M	C C C M	M D D M D	W W W D	M	C	D	M
2.5 each piece in specific order and tested last	M M M M	C C C C M	M D D M D	W W W D	M	C	D	M
3.1 each piece in 5th to 10th order	M M M M	C C C M	M D D M	W W W D M	M	C	D	M
3.2 each piece in 10th to 15th order	M M M M	C C C M C	M D D D M	W W W D D	M	C	D	M
3.3 each piece in 15th to 20th order	M M M M	C C C M C	M D D D M	W W W D D	M	C	D	M
3.4 each piece in the last order.	M M M M	C C C M C	M D D D M	W W W D D	M	C	D	M
3.5 each piece in specific order and tested last	M M M M	C C C C	M D D D M	W W W D	M	C	D	M

1. M --> Mozart, C --> Chopin, D --> Debussy, W --> Webern,
   H --> Haydn, S --> Schumann, R --> Ravel, B --> Berg.
2. 1.1, 1.2, 1.3, 1.4, and 1.5 used weight manipulation method 1;
   2.1, 2.2, 2.3, 2.4, and 2.5 used weight manipulation method 2;
   3.1, 3.2, 3.3, 3.4, and 3.5 used weight manipulation method 3;

Another similar example, using weight manipulation scheme 3, Debussy's fourth testing piece was tested in the very beginning of the run, the testing results were;

```
grand_total(chopin 0.113474).
grand_total(mozart 0.223656).
grand_total(debussy 0.187486).
grand_total(webern 0.110973).
```

The guessed composer for this result would be Mozart because he had the highest total score. If the same piece was tested in the tenth order or later, the results were:

```
grand_total(chopin 0.114333).
grand_total(mozart 0.204328).
grand_total(debussy 0.255587).
grand_total(webern 0.115783).
```

Now Debussy got the highest total score and would be correctly guessed as the composer.

But, this is not always the case. The results might become worse as shown in the following: (either result A or B)

```
A: grand_total(chopin 0.111382).
   grand_total(mozart 0.254209).
   grand_total(debussy 0.190418).
   grand_total(webern 0.222009).

B: grand_total(chopin 0.235512).
   grand_total(mozart 0.212318).
   grand_total(debussy 0.220418).
   grand_total(webern 0.122009).
```

In this case, Mozart or Chopin received the highest score and the system would wrongly guess Mozart or Chopin as the composer.

Some piece like Debussy's fourth testing piece Reverie, sometimes were identified correctly; however, testing the piece in the later order with other pieces tested in different order yielded incorrect identification. It is possible that the testing order of the correctly-guessed pieces influenced

this result. If more correctly-guessed (very distinctive pieces) pieces were tested earlier, the answer tended to be much closer. But if it is that confusing pieces were tested earlier, then the answer could be worse.

6. Conclusion

6.1 A Summary of Conclusion

A total of twenty music pieces were encoded and tested in the system, and the results came out fairly satisfactorily. I think that this is because of the following four factors:

(i) Most of the chosen composers have distinguishable styles.

(ii) The musical aspects observed are straightforward and easy to implement. As mentioned before, melodic structure, contour structure, contour height structure, complexity of melody and rhythm, triad root-movement and dissonance are observed in the system. These useful indicators were easy to implement.

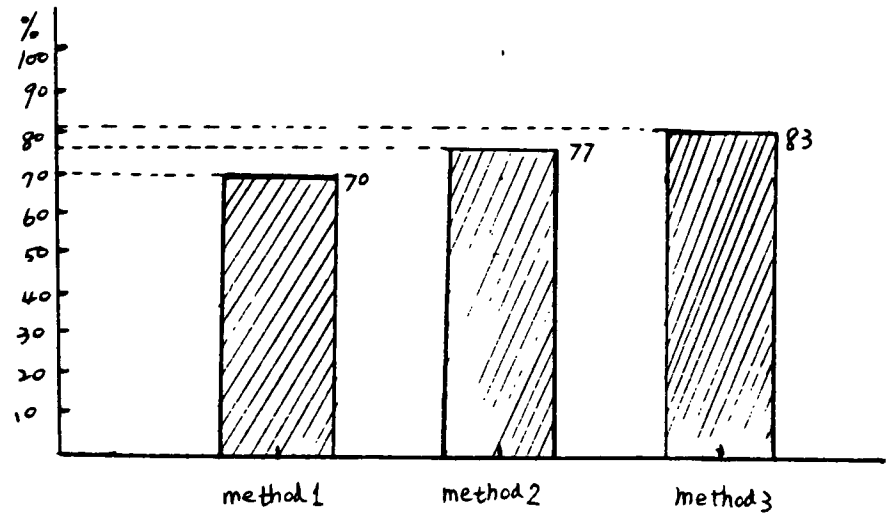
(iii) The weight manipulation techniques contributed to correct adjustment of the pattern classification. Weight manipulation techniques manipulate weights according to the classification results generated by the system. After weight manipulation, those "good" and "salient" characteristics are emphasized while "useless" or "weak" characteristics are discounted. This does improve the system's classification ability.

(iv) the rules used for determining the term values for each characteristic were general and practical. They are based on experiences, and are easy to understand and intuitively appealing.

The following are some performance charts to describe the overall performance rate of the system.

6.1.1 Performance Rate - By Weight Manipulation Methods

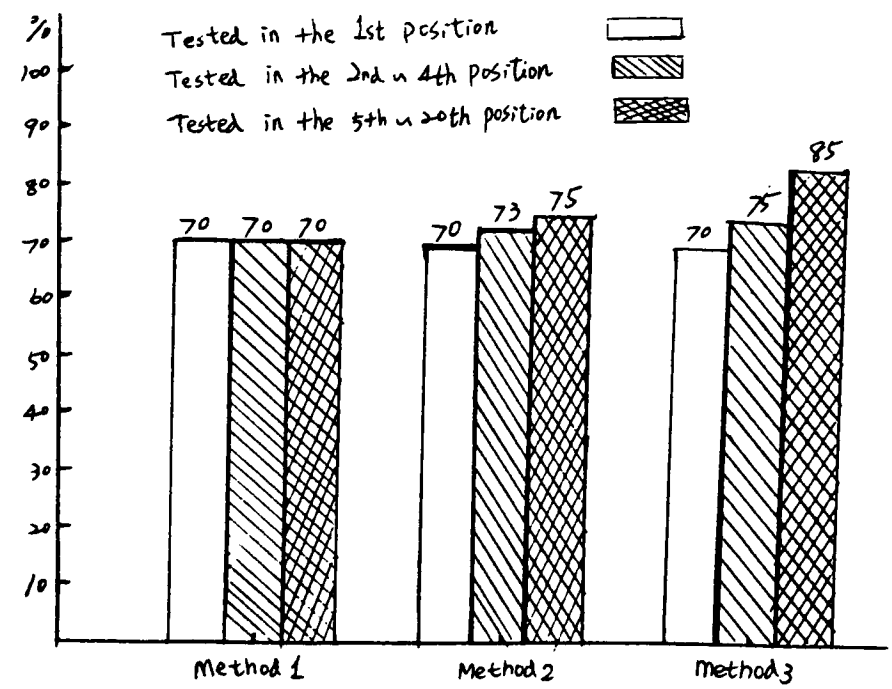
Figure 6-1 Performance Rate Chart - By Weight Manipulation Methods



The correct identification rate for weight manipulation method 1 is 70%, for method 2 is 77% and for method 3 is 83%. We can see that weight manipulation method 3 is the best method for improving the system's classification skill.

6.1.2 Performance Rate - By Order

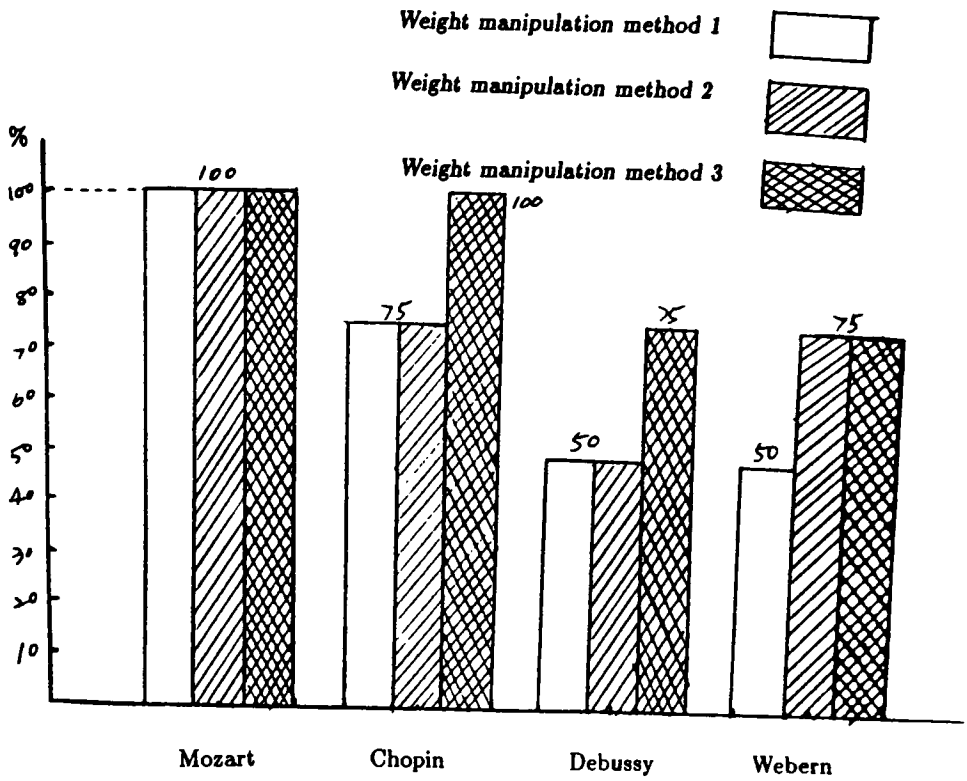
Figure 6-2 Performance Rate Chart - By Order



According to Figure 6-2, for each weight manipulation method, the latter the testing position, the better the performance becomes, and the effect is more obvious for the method 3.

6.1.3 Performance Rate - By Composer

Figure 6-3 Performance Rate Chart - By Composer



As shown in Figure 6-3, method 3 has still the best performance rate by composer.

6.1.4 The performance of each characteristic observed

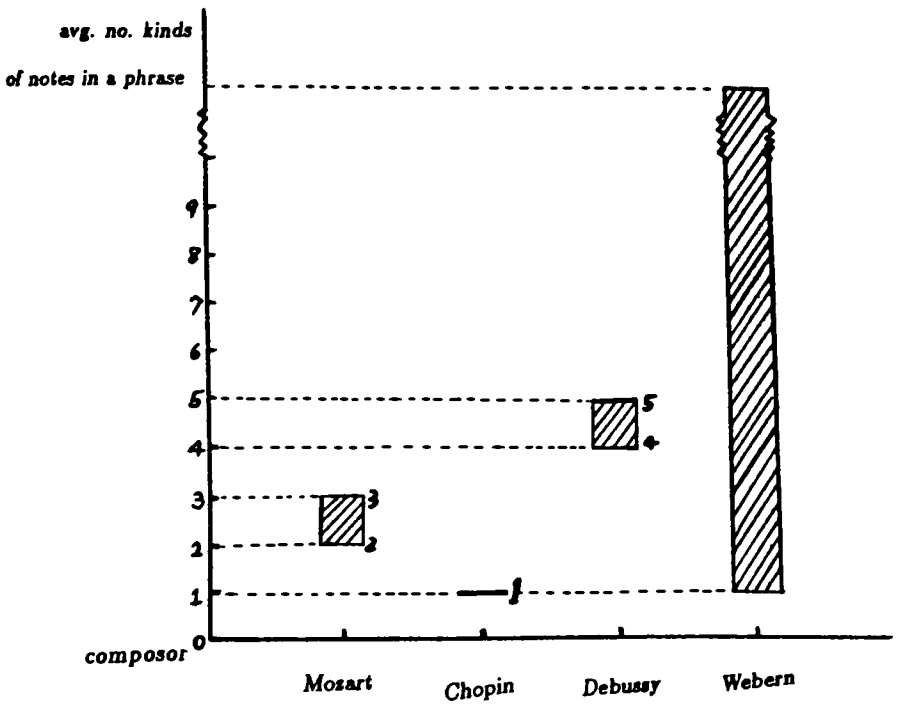
In the characteristics analyzer, seven characteristics were observed. The results of this project suggest several conclusions.

6.1.4.1 Melodic Structure Patterns

Melodic patterns are of primary importance in discovering which patterns have been used in other pieces by the same composer and which are merely idiomatic melodic patterns found frequently in specific works. Probably because of the different forms of piano pieces chosen (for example, Sonata, etude or Mazurka), the melodic patterns in this analysis were not very distinctive and sometimes led to the opposite judgement. Since melody may suggest the composer's attitudes toward expression and meaning of music, I believe that each composer had different preferences for melody in their compositions, but in this analysis this characteristic was weak and unclear.

6.1.4.2 Rhythmic Structure Patterns

Figure 6-4 Rhythmic Structure Patterns

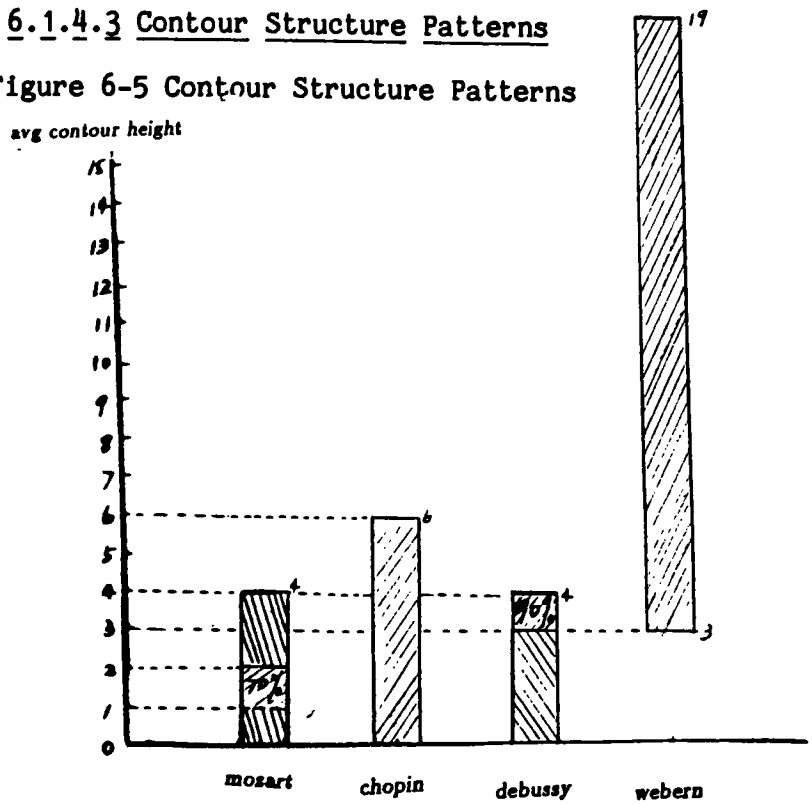


The rhythmic structure patterns were very useful for distinguishing different rhythmic characteristics in music pieces by different composers. Generally, in this analysis, rhythms in Mozart's, Chopin's or Debussy's pieces were straightforward, but Mozart's style typically included two or three

different note values within a phrase with shorter notes preferred (e.g. eighth note, sixteenth note). Chopin's rhythms were even simpler, sometimes only one note value (quarter note) was used in the left hand part score. Debussy's rhythmic structure was a little complicated, always containing four or five different note values within a phrase. Anton Webern's rhythmic settings consisted of interchangeable combinations of all possible note values.

6.1.4.3 Contour Structure Patterns

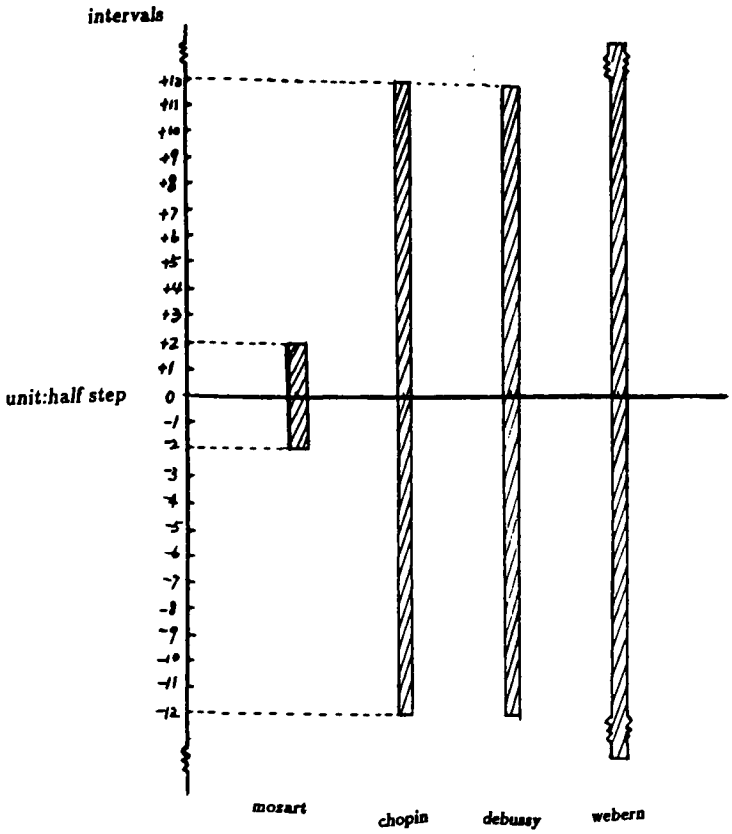
Figure 6-5 Contour Structure Patterns



The contour structure was an effective indicator for the analysis. The most frequently used contour structure in Mozart's pieces was very "smooth", with the intervals within $\pm$ 2 half steps. The contour structures in Chopin's and Debussy's pieces, however, did not have a consistent style; sometimes the intervals ranged up to 12 half steps. The contour intervals in Anton Webern's music pieces were always very high, but in only one piece were the intervals like Mozart's style, very smooth and stable.

6.1.4.4 Contour Height Structure Patterns

Figure 6-6 Contour Height Structure Patterns



For this part of the analysis, the system partitions the melodic contour string derived from the contour analysis into a few ordered subsets - from eight-code patterns down to three-code patterns, and calculates the moving average height of each contour subset. The moving average heights themselves constitute contour height strings, and these were used to do contour height pattern recognition.

The absolute value of each contour code was used to calculate the average contour height because only the magnitude matters, not the direction. Each composer showed different average height of contour subsets. In general, the longer the subset was, the lower the average height was. Mozart's average

contour height was in the range 0 - 4, but 70% of them were 1 or 2. Chopin's average contour height ranged from 0 - 6, but the distribution was nearly uniform. Debussy's average contour height ranged from 0 - 4, with most of them 3 or 4. Anton Webern's average contour height ranged from 3 - 19.

This structure was very effective in expressing the melodic activities from different musical works.

6.1.4.5 Triad Root-movement Patterns

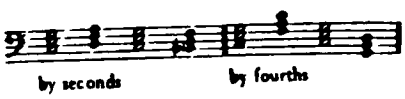
Figure 6-7 Triad Root-movement Patterns

Mozart None.

Chopin

Debussy None.

Webern



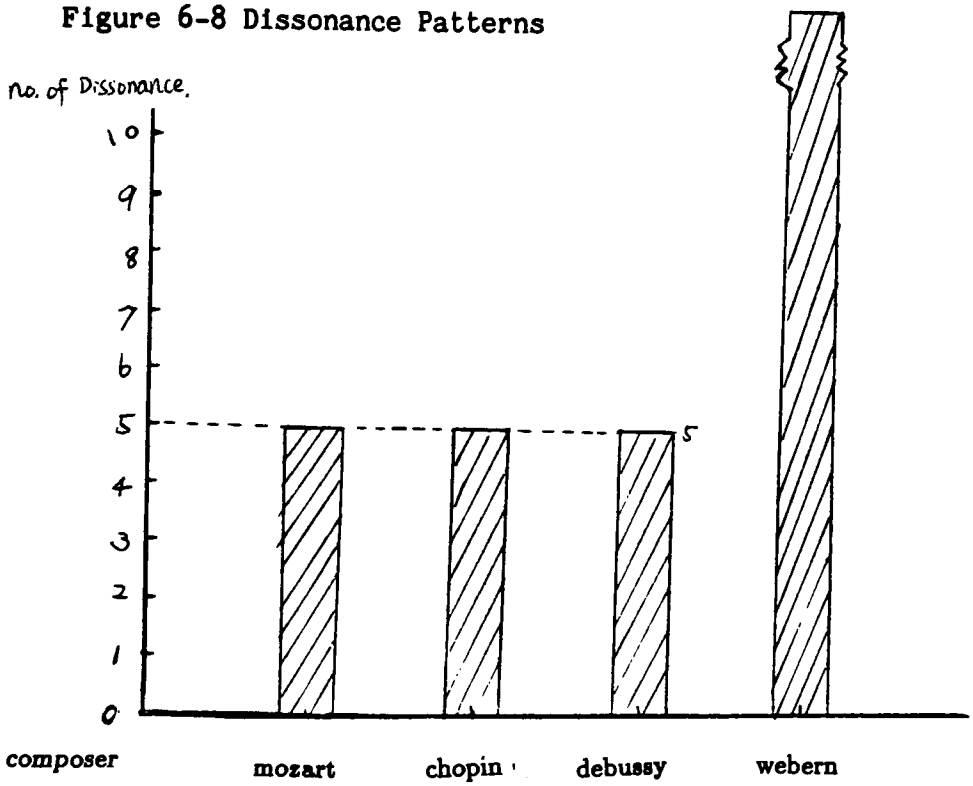
Another method for characterizing music was to classify the triad root-movements used in the score and record the frequency counts for each class, that is by seconds, thirds, fourths, fifths and others. As defined before and in the glossary (Section 9-2) triad is a term for three tones sounding simultaneously, and root movement is a term used to describe the scale distance between vertical sonorities. Often the style of an individual composer will reveal a marked preference for certain kinds of root movement.

Since the analysis is done vertically, the full scores need to be retrieved. The system searches for the adjacent triads, classifies their root-movements, and then records them in the database.

The relationships of triads were an important feature for every period. In this analysis, Mozart and Debussy rarely used triad pairs in their pieces. Chopin used some triads, and the root movement was by seconds only. Webern used lots of triads, and most of the root movements were by seconds and by fourths.

6.1.4.6 Dissonance Patterns

Figure 6-8 Dissonance Patterns



Consonance and dissonance are terms used to described the effect of two or more simultaneous sounds. In this project, I found that almost every composer in each period used some dissonance, which was unexpected. Fortunately, the pieces by Mozart, Chopin and Debussy have very few dissonances (zero to five), but most of Anton Webern's piano pieces have many dissonances.

6.1.4.7. Complexity of Melody and Rhythm

Figure 6-9 Complexity of Melody

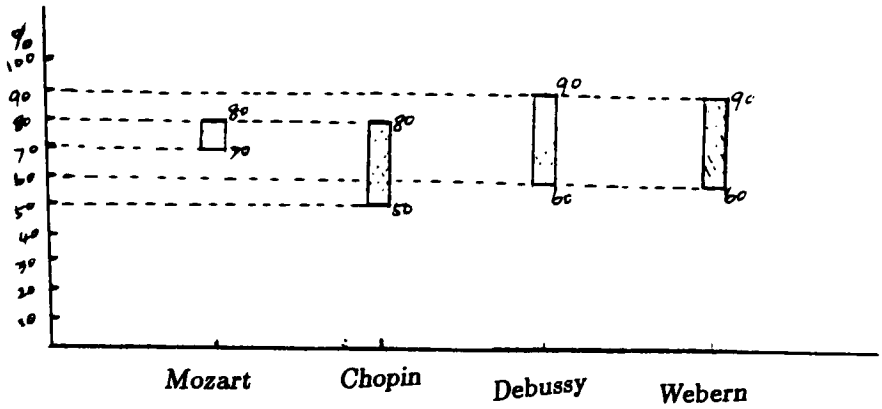
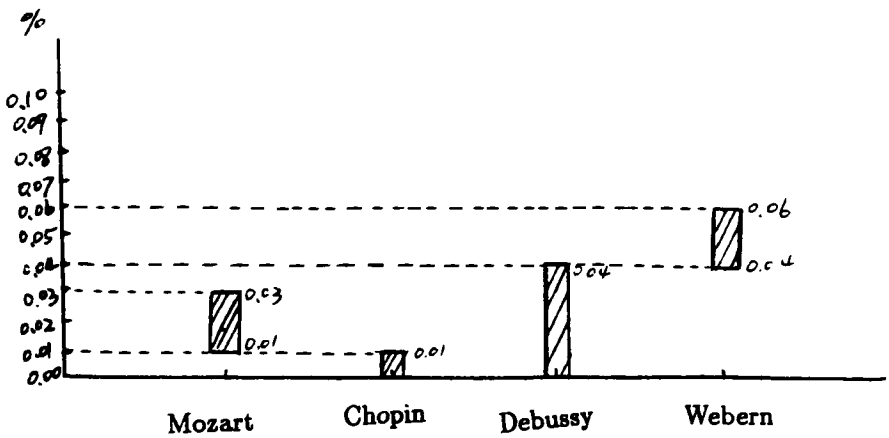


Figure 6-10 Complexity of Rhythm



This method is a useful measure of the complexity of music. The possible range for the complexity of melody is 0 - 100, and the range for the complexity of rhythm is 0.00 - 0.10. In my thesis, the complexity of melody in Mozart's pieces ranges from 70's to 80's. In Chopin's pieces, it ranges from 50's to 80's. Most of them were 50's. In Debussy's and Webern's pieces, it ranges from 60's to 90's, so we can see the complexity of Chopin's melody is lowest. Similarly, the complexity of rhythm in Chopin's pieces is the also the lowest, ranging from 0.00 to 0.01. While it ranges 0.01 - 0.03 in Mozart's pieces, 0.00 - 0.04 in Debussy's pieces and 0.04 - 0.06 in Webern's pieces.

6.2 Problems encountered and Shortcomings of the system

Problems that I encountered while working on this project and their solutions include:

(i) Encoding the musical scores in DARMS codes

It took me very long time to encode the musical scores in DARMS code, especially when I was not familiar with the space codes. I had to look up the corresponding codes for almost every note. Sometimes a few notes occurred at the same time spot but with different note lengths. I had to record the shortest next-event (next note) first in order to keep the right meter and the consistency of the global time spine. Sometimes a note had a sharp or flat or dotted length. This was a very error-prone, especially when the score was very long and complicated, so I wrote some checking routines to check if the DARMS codes were correct or not. If there were unknown codes, mismatched meters or a wrong global time spine, the program gave an error message to tell where the problem was. Then I could check with the original score and correct the DARMS codes. Even though this program does not know any musicology rules at all, it was still helpful for checking the translated DARMS codes.

(ii) Stack and Heap size limit in C-Prolog

Because the knowledge base and testing base in the system supplied music characteristics information, I tried to use literal terms as much as possible. One drawback for this is that the pattern matching process in the classifier requires a great deal of time to backtrack and a fairly large amount of computer memory to store global variables and backtrack points. In order to save time and memory space, all useless backtrack points were "cut" by "!" (cut in C-Prolog). This saves time and memory, but because the system has to

match 51 music patterns and does the normalization, manipulations, and score calculation, I still needed to increase the Heap size and Global stack size in C-Prolog while I ran the classifier.

Shortcomings of the system

(1) For convenience, I transposed all scores to the key of C. That means that the changes of key signature in the scores were not represented and considered. Furthermore, major versus minor key was not distinguished at all.

(2) Many other important characteristics were not considered in the system such as Dynamics, Texture (e.g. tempi) and color. Those are all very valuable characteristics.

(3) The testing pieces were chosen randomly so that a few pieces were compared to pieces that were in a different form, for example, Chopin's Valse Brilliante and Mazurka. More pieces should be tested.

(4) The classifier adjusts its own analysis tools (e.g. weight) by weight manipulation, but it learns on one level only. The relation between characteristics is determined by experience. Everything else is fixed by the designer. It would certainly be desirable for a characteristics recognizer to use experience for more general improvements: to refine its decision process and the ways weight are manipulated. My project does not have such an interaction of several levels of learning.

(5) A random number generator was considered to generate a random testing pattern to serve as a control but was not implemented, because of lack of time.

6.3 Lessons Learned

I used C language to implement the characteristics analysis and Prolog to do pattern matching. This makes the system work more efficiently and powerfully.

(1) One of the primary attributes that makes the music stylistic analysis system work successfully is the encoding language used - DARMS. DARMS encodes the music score efficiently and it can bring the right information whenever it is needed during the course of analysis. As mentioned before, DARMS has been used in a number of theoretical studies by scholars such as Allen Forte and John Rothgeb [Forte][Rothgeb]. A group of scholars involved in the DARMS project has been working on a set of programs to help with this task. The principle programs under development are a DARMS syntax checker and an Input-DARMS to Canonical-DARMS translator (or "Canonizer") that will translate any of many syntactically correct Input-DARMS representations into a fully resolved standard form [Erickson].

(2) One challenge this system faced was determining the pattern recognition algorithms. Because the music score was encoded to DARMS code and that code can be treated like a string, a substring recognition machine (Deterministic Finite Automata-DFA) was built. The algorithm used to construct the DFA was referred from [Hopcroft and Ullman]. According to the book, for a machine M that recognizes substring Y in string X, in $O(|X| + |Y|)$ steps, the result can be determined, where $|X|$ is the length of string X, $|Y|$ is the length of substring Y.

The machine I built in the system recognizes some Y^i substrings in a string X. In $O(\ell + | X |)$ steps, we can determine whether some Y_i (i=1...k)

is a substring of X, where $\ell = |Y_1| + |Y_2| + \ldots + |Y_k|$. This is rather efficient.

(3) Another algorithm used in the system is the algorithm to build Markov chain. This is used for mainly computer memory considerations. "malloc()" or "calloc()" in C language was used to allocate memory for storing the chain which is a linked list storing alphabet of a string and all pairs of adjacent alphabet and their occurrences. This algorithm is also efficient because I only used it to find the alphabet (Sigma of each string) and the two-code patterns in the string. In only $O(|X|)$ steps. we got the Sigma, and in another $O(|X|)$ steps, we got the most frequent 2-code pattern.

(3) Another challenge I faced was determining the suitability of building a classifier for pattern matching and characteristics classification.

Two solutions using C failed because the level of complexity was too high, and the concepts are too abstract for modeling with a procedural language, especially because I want to make the pattern be written as human-understandable type. Finally, a classifier in Prolog was used.

In Prolog, it is much easier to set a well-bounded domain. It is also easier to express music terms in literal form. Besides, pattern matching is a fundamental characteristic of Prolog. Using Prolog also played an important role in making the system work successfully.

6.4 Future Development

A possible future development of this project could be to add more composers in the same period. This would involve adding some case statements in the stylistic analyzer (main.c) and adding some matching clauses in the

classifier. For better classification ability, however, more analysis methods would have to be added and probably more pattern recognition algorithms would be needed. This would involve rewriting the analyzer (main.c) and the classifier.

For this thesis, I took the advantage of choosing four composers in four different periods, most of their works, supposingly, are different from each other. But for the composers in the same period, more characteristics are definitely need to be considered. Selecting the features to be observed has to take into account the relative complexity of the various musical elements. Generally, five basic elements can be observed: sound, form, harmony, rhythm and melody. I did not consider sound and form at all, and the other three are only partially detected.

Sound is basic to all the other elements. For analyzing sound, we can detect the fundamental frequency of sound using a band-pass filter and then transforming the sound signal into power spectrum for analysis [Yusuaki and Seiji].

In most cases, considerations of form should proceed differently. For pieces in more than one movement we should note the balances and relationships of tempos, tonalities, meters, choices of timbres and textures, rhythmic structure, modulatory goals, and similar stylistic details. While pieces have individual movements, we may often identify them broadly with one of the conventional plans such as sonata, rondo, or variation form.

The remaining three elements still have many characteristics can be analyzed, such as key-schemes, chromaticism, alterations, progressions, and rhythmic density of harmonic change, repetitive, sequential, or continuously

developed structure.

Another possible future thesis topic could be to extend the classification options to include learning and generating new characteristics for its own and so that it can use them in identifying new composer's work. This would be looked as an interaction of several levels of learning and probably learning by analogy would be a good choice.

Learning by analogy is another potentially fruitful area for music. The groundwork for learning by analogy was laid by Evans [Evans] and Winston [Winston 1979,1980]. Learning by analogy uses procedures to compare the structureal descriptions of two different phenomena to extract their pertinent similarities.

Levitt's recent work is based on Evans's analogy program [Levitt]. It generates a structural model of the relationships between a pair of musical objects or their structural descriptions. The system works in the following way. The experimenter gives the program two things: a musical concept description (e.g. a semantic network for describing a melody or a chord progression), and a specific realization of that concept. The realization could be a particular rhythmic setting for the specified melody, or a musical pattern. The program then determines the relationships between the concept and the realization. When presented with a second concept description, the program is able to generate its own realization of it according to the first example, because the concept descriptions can be used for generative as well as analytical purposes. Given samples of a particular purposes and a particular style (Levitt's domain is traditional New Orleans-style jazz), the program should be able to learn (i.e., be able to recognize and generate) new or different scale structures. In this way, the system can initially recognize

a limited set of characteristics and gradually absorb more high-level concepts from samples presented to it.

7. Closing Thoughts

The research for this thesis is concerned with doing musical stylistic analysis by a computer system that mimics human intelligence. The major aim is to perform analysis tasks the same way that persons perform these tasks.

Musical stylistic analysis is part of music theory analysis. To summarize the ideas of music theory analysis, we can say that it consists of a set of observations of the nature of musical sound and of its continuity in time, of the elements of music and their multiple combinations, and of actual compositions from about 800 AD to the present. An understanding of these observations is essential to the music theory researcher.

In the thesis, the computer system developed a few musical characteristic analysis models that observed and analyzed features of music works, and it derived a set of well-defined classification rules for classifying those features in the music pieces. Doing stylistic analysis (or more broadly, music theory analysis) by computer has many benefits, the most significant one probably being that the resultant system can be tested repeatedly. Other advantages are higher reliability, due to a decrease of human error and faster and more accurate results.

The thesis explains that cognitive science and AI techniques can be used to help do many human tasks. For further extensions of this thesis, the same basic idea can be used in more stylistic analysis and other kinds of pattern recognition.

8. Bibliography

[1] Alphonce, Bo H. "Music Analysis by computer - A field for theory formation." paper presented at the 1978 International Computer Music Conference, Nov. 1-5, North Western University, Evanston, Illinois. Also in Computer Music Journal 4 (summer 1980)

[2] Baroni, M. and Jacoboni, C. "Analysis and generation of Bach's Choral Melodies"

[3] Baroni, M. and Jacoboni, C. 1978, "A proposal for a grammer of melody". Les Presses de i'Universite 'de Montreal, Montreal, Canada

[4] Batchalor, B.G. "Classification and data analysis in vector spaces". Pattern Recognition edited by Batchalor, B. G. Plenum press, NY, London

[5] Beauchamp, J. W. "A Computer System for Time-variant Harmonic Analysis and Synthesis of Musical Tones"

[6] Beauchamp, J. W. "Electronic instrumentation for the synthesis, control, and Analysis of Harmonic Musical Tones", Ph. D Thesis, Univ. of Illinois, Urbana(1965).

[7] Bent, Ian, and J. Morehen, "computers in the Analysis of Music." Proceedings of the Royal Musical Association 104 (1977-78) : 30-46

[8] Biles, J. A. "Adaptive Recognition and synthesis of transcribed Jazz Solos by Computer" University of Kansas, 1973.

[9] Binnie, C. D., G. F. Simith and B. G. Batchalor, "Pattern Recognition in Electornecephalography". Pattern Recognition edited by Batchalor, B. G. Plenum press, NY, London

[10] Brinkman, Alexander R. "Johann Sebastian Bach's Orgelbuchlein: A computer Assisted analysis of the influence of the cantus firmus on the contrapuntal voice", The Journal of the Society for Music Theory Vol. 2 1980

[11] Chomsky, N., "Syntactic Structures", Mouton, The Hague, 1957

[12] Chomsky, N. and G. A. Miller "Finite-State Languages", Information and Control, 1(1958),

[13] Chomsky, N. "Context-free Grammar and pushdown storage", Quarterly progress Report, 65, (1962) M.I.T. Research Lab. in Electronics, Cambridge, Mass.

[14] Chomsky, N, "Aspects of the Theory of syntax". MIT Press, Cambridge, Mass. (1965)

[15] Cooper, P. "Perspectives in Music Theory", Dodd Meade, New York, 1973.

[16] Divilbiss, J. L. "The real-time generation of music with a digital computer", J. Music theory, 18, 99 (1964).

[17] Dvidow, D. H. Ed. "Music by Computer" from 1966-EFCC

[18] Eden, Murray, "Other pattern recognition problems and some generalizations". Recognizing Patterns Studies in living and automatic systems edited by Paul A. Kolers and Murray Eden , The MIL Press, Cambridge, Mass.

[19] Erickson, R. F. "MUSICOMP 76 and the state of DARMS", College Music Symposium 17/1(1977):90-101.

[20] Evans, T. "A program for a solution of a class of geometric analogy intelligence test questions". In Semantic Information Processing, M. Minsky, Ed, MIT Press, Cambridge, Mass.

[21] Firth, J.R. "Papers in Linguistics", London Oxford Univ. Press, 1957

[22] Fletcher, H. "The Pitch, Loudness and Quality of Musical Tones", Bell telephone system Monograph No. 1397(1946).

[23] Forte, A. "Schenker's conception of musical structure". Journal of Music Theory 3,1(April, 1959), 1-30.

[24] Forte, A. "A program for the Analytic Reading of Scores", Journal of music theory 10/2 (1966):330-364

[25] Freedman, M. D. "On-line Generation of Sound".

[26] Freedman, M. D. "Analysis of Musical Instrument Tones", J. Acoust. Soc. Am., 41(4), 793-806 (1967).

[27] Fu, K. S. "Syntactic pattern Recognition: Applications", Springer-Verlag (1977)

[28] Fucks, W. "Mathematical analysis of formal structure of music"

[29] Gazdar, G. "phrase structure grammar", to appear in The Nature of Syntactic Representation. edited by Pauline Jocobson and Geoffrey K. Pullum. Dordrecht:Reidel, forthcoming

[30] Gazdar, G. Geoffrey, K. Pullum, and Ivan A. Sag, A phrase structure grammar of the English auxiliary system. To appear in the Nature of Syntactic Representation. edited by Pauline Jocobson and Geoffrey K. Pullum. Dordrecht:Reidel, forthcoming

[31] Gross, Dorothy S. "A Project in Computer-Assisted Harmonic Analysis." paper presented at the 1980 International Computer Music Conference, November 13-15, 1980, Queens College of the City Univ. of New York.

[32] Grout, D. J.,"History of Western Music", Revised ed., W. W. Norton, New York, 1973.

[33] Halliday, M.A. Categories of the theory of grammar. (1961)

[34] Hiller, L.A. and L. M. Isaacson, "Experimental Music: Composition with an Electronic Computer". McGraw-Hill, New York, 1959

[35] Hiller, L.A. "Music Composed with computers: a historical survey". In the computer and Music. (H.S. Lincoln, Ed.) Cornell University Press, Ithica, N.Y., 1970, 42-96.

[36] Hopcroft, A. and Ullman, "The Design and Analysis of Computing Algorithms", Addison Wilsley Publish Co.

[37] Kaplan, Ronald M. "Augmented transition networks as psychological models of sentence comprehension". Artificial Intelligence (1972)

[38] Kmenta J., "Elements of Econometrics"

[39] Kolers, Paul A. "Some psychological Aspects of Pattern Recognition". recognizing Patterns Studies in living and automatic systems Recognizing Patterns Studies in living and automatic systems edited by Paul A. Kolers and Murray Eden , The MIL Press, CAmbridge, Mass.

[40] Lerdahl, F., and R. Jackendoff, "Toward a formal theory MIT Press, Cambridge, Mass. (1977)

[41] Lerdahl, F., and R. Jackendoff, "A Generative Theory of Tonal Harmony". MIT Press, Cambridge, Mass. (1983)

[42] Levitt D. "Learning music by imitating", Unpublished manuscript. (1983)

[43] Lewis N. R. and C. H. Papadimitriou, "Elements of the theory of Computation". Prentice-Hall, Inc. N.J. (1981)

[44] Lincoln, H.S. (Editor). "The computer and Music". Cornell University Press, Ithica, N.Y., 1970.

[45] Lincoln, H.S. "Uses of computer in composition and research". Advances in Computers 12 (1972), 73-114.

[46] Lincoln, H. B. Ed. "The Computer and Music", Cornell University Press, 1970

[47] Longutet Higgins, H. C. "On Interpreting back", Machine Intelligence 6, Endingburgh, 1971.

[48] Luce, P. A. "Physical Correlates of Non-percussive Musical Instrument Tones", Ph. D Thesis, M.I.T. Cambridge, Mass. (1963).

[49] Mathews, M. V. "An Acoustical compiler for music and psychological stimuli", Bell system Tech. J., 40, 377 (1961).

[50] Mathews, M. V., J.E. Miller, J. R. Pierce, and J. Tenney, "Computer Study of Violin Tones", Bell Telephone Lab Internal Report, Murray Hill, N.J. (1966).

[51] Minsky, M., "A Framework for representing knowledge". MIT AI Lab., Cambridge, Mass. (1975)

[52] Parsley, W. J. "Identifying the unknown communicator" Journal of Communication Vol. 14 /1964

[53] Patrick, P. H. and K. Struckler, "A computer-Assisted Study of dissonance in the masses of Josquin Desprez".

[54] Pinker, S. "Formal models of language learning". Cognition 7:3 (September 1979)

[55] Plomp, R. "The Ear As a Frequency Analyzer", J. Acoust. Soc. Am., 36(9), 1628-1636 (1964).

[56] Risset, J. C. "Computer Study of Trumpet Tones", Bell Telephone Laboratories Internal Report, Murray Hill, N.J. (1966).

[57] Rothgeb, J. E. "Harmonizing the Unfigured Bass: A computational Study" (Ph.D. diss., Yale Univ, 1968)

[58] Rue, Jan La "FORUM (on style analysis)" Journal of Music Theory Vo. 6 #1 (spring , 1962)

[59] Silvian, L. J., H. K. Dunn and S. D. White, "Absolute Amplitude and spectra of certain Musical Instruments and Orchestras" Bell Telephone System Monograph No. 3381 (1959).

[60] Snell, J. "Design for a formal system for deriving tonal music". M.A. thesis. Dept. of Music, State Univ. of NY at Binghamton, Binghamton, N.Y. (1979)

[61] Thompson, M. G. "Syntactic Pattern Recognition: An Introduction", Addison-Wesley (1978)

[62] Thompson O. "The international cyclopedia of music and musicians"

[63] Uhr, L. and Vossler, C. "A pattern-recognition program that generates, evaluates, and adjusts its own operators" Computers and Thought edited by E. A. Feigenbaum & J. Feldman (1963)

[64] Von Helmholtz, H. L. F. (translated by A. J. Ellis), "On the sensation of Tone as a physiological basis for the theory of music", Dover, New York(1954).

[65] Winograd, T., "Linguistics and the computer analysis of tonal harmony. J. Music theory. (1968)

[66] Winograd, T. "Language as a Cognitive Process". Vol I, Addison-Wesley Publishing Co. (1983)

[67] Winston, P. "Learning by creating and justifying transfer frames". In Artificial Intelligence: An MIT Perspective, Vol. 1, P. Winston, Ed. MIT

Press, Cambridge, Mass. (1979)

[68] Winston, P. "Artificial Intelligence", Addison-Wesley Publishing Co. (1984)

[69] Winston, P. "Learning structural descriptions from examples." In the psychology of Computer Vision, (1975) McGrow-Hill, New York

[70] Winston, P. "Learning and reasoning by analogy". Commun. ACM 23, Dec. (1980)

[71] Woods, William A. "Transition Network Grammars for Natural Language analysis". CACM 13:10 (1970)

[72] Xenakis, I. "Formalized Music", Indiana University Press, Bllomington, (1971)

[73] Yasuaki, N. and Seiji Inokuchi, "Music Information Processing System and Its Applications to Comparative Musicology". Proceeding of the sixth International Joint Conferences on Artificial Intelligence, Tokyo, Aug. 1979.

9. Appendices

9.1 Glossary

some music notations(terms) are explained for those who are not familiar with this field.

(1) Staff:
Staff refers to the five horizontal lines upon or between which the notes and rests are placed.

(2) Clefs:
Clefs indicate the location of a specific pitch. Most common ones are the treble, or G, clef and the Base, or F, clef. Different clefs assign different pitches to lines on the staff.

(3) Key signatures:
A key signature gives a preinventory of pitches that by default require either a sharp or a flat; it traditionally indicates one of two keys: the major tonic or the minor tonic. For example, a key signature of three flats is $E^{\flat}$ major or C minor.

(4) Meter signature:
Meter signature is employed in notated music to indicate the number and kind of durational units (beats) contained within a measure. The most widely used are perhaps 4/4,3/2,2/4,3/4,6/8,9/8. The upper number specifies the number of principal beats (e.g. 3 in 3/2); the lower number indicates the kind of note that represents one beat (e.g., the half note receives one beat in 3/2, or in 3/4 the quarter note receives one beat).

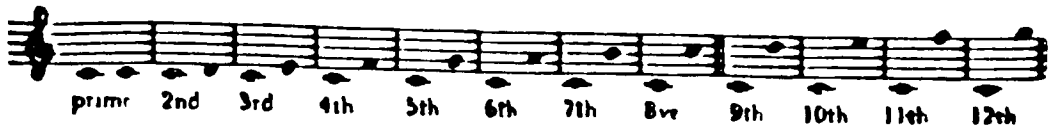
(5) Scale:
Scale is a term that denotes an ordered arrangement of pitches. Strictly interpreted, the term implies a sequence of rising pitches-- usually defined within the octave. The basis of traditional Western music is the diatonic scale(c d e f g a b c) that consists of both whole tones and semitones. A scale comprising only semitones results in a chromatic scale.

(6) intervals:
The distance between two pitches is called an interval. In calculating and describing an interval, at least four different aspects are considered: (Figure 9-1)

i. The theoretical description of consonance or dissonance.
ii. The arithmetic distance(or difference) between two notes.
iii The quality (harmonic color, tonal color)
iv. The indication of whether the interval is contained within an octave(simple) or is greater than an octave(compound)

Intervals are typically described as being in one of three classes:
Perfect consonances: unisons(primes),fourths,fifths, octaves
Imperfect consonances: thirds, sixths(tenths, etc)
Dissonance: seconds, sevenths(ninths, etc)

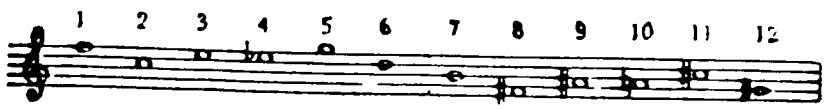
Figure 9-1



(7) twelve-tone system: (Figure 9-2)

Twelve-tone system is primarily a technique of pitch arrangement and utilization. The ordering of the twelve pitches contained withing the octave may serve as the basis of vertical and horizontal formations in a composition.

Figure 9-2



(8) Rhythm is a term used to describe the duration of sound.

(9) Melody is a succession of pitches; by its nature it can not be separated from rhythm.

(10) Harmony is the resultant of the simultaneous combination of two or more musical sounds.

(11) A triad is three tones sounding simultaneously. It is described by

the base note:(See Figure 9-3)

Base note	Description	Name
Root of Chord	Root position	Chord of the fifth
Third of chord	First inversion	Chord of the sixth
Fifth of chord	Second inversion	Six-four chord

Figure 9-3 an explanation of Root position

Root movement is a term used to describe the scale distance between vertical sonorities. Often the style of an individual composer will reveal a marked preference for certain kinds of root movement.

9.2 Proposal

The original proposal of this thesis describing the goal and the functions to be performed is on file with the Computer Science Department. What follows is the description of changes to the project from the original functional specification and system specification.

In the whole system a number of minor modifications and some major changes were made. One minor change was that an array of pointers was used to store

the musical scores instead of a doubly-linked list, because the array was easier to access. However, when I designed the stylistic analysis models, I used a linked list to build up my Markov Chain for pattern matching.

A second minor modification was that the meter signature, score name and key signature were not used. Instead, all scores were manually translated to fixed-C-keyed DARMS codes and then transformed to movable-C-keyed codes using a separate sub-program. In addition, the duration codes of the DARMS codes were also modified (increased) to cover all possible durations (note-lengths) in the scores, such as triplets and dotted notes.

Another minor difference from the original specification was that a few analysis methods were added to the stylistic analyzer and a few methods mentioned in the proposal were dropped.

One major modification was that Anton Webern's piano pieces were used for testing instead of Alban Berg's because of a lack of available works by Berg.

The other major change to the system specification was that only C or PASCAL language was proposed to be used as implemention tools so that no interface work was needed. However, PROLOG's pattern matching was extremely useful for the classification processes, so I wrote the stylistic analyzer in C and the classifier in PROLOG.

As the implementation tools were changed, the system data flow chart was consequently different from the proposal. The input DARMS codes were fed into the stylistic analyzer (written in C) which generated analytical stylistic patterns which were put into output files (data bases). The classifier (written in PROLOG) concatenated these data bases and did the classification and identification. The main flow of the system was retained, but the

execution flow was changed.

9.3 How to run the system

The following is the procedures used to run the system:

First, to compile the characteristics analyzer:

```
% cc -o main.c -lm      /* -o --> optimize the compilation        */
                              /* -lm --> math library routines included */
```

The following is the SHELL SCRIPT used to run the system:

Then create a file and change the file mode to be executable by using the following command:

```
% chmod "+x" filename
```

The content in the file is as following:

```
-----------------------------------------------------------------

% a.out < chopin1_piece > chopin1_base /* 1st music piece for knowledge base
*/
% a.out < chopin2_piece > chopin2_base /* 2nd music piece for knowledge base
*/
% a.out < chopin3_piece > chopin3_base /* 3rd music piece for knowledge base
*/
% cp chopin1_base chopinbase           /* concatenate them to one file*/
% cat chopin2_base >> chopinbase
% cat chopin3_base >> chopinbase
% a.out < chopin1_test > chopin1_testbase /* 1st testing piece */
% a.out < chopin2_test > chopin2_testbase /* 2nd testing piece */
% a.out < chopin3_test > chopin3_testbase /* 3rd testing piece */
% a.out < chopin4_test > chopin4_testbase /* 4th testing piece */
```

The same processes for Mozart, Debussy and Webern. Then

```
% prolog -G 1500 -H 1500 < one_run > one_output
-----------------------------------------------------------------
```

In the file one_run (should be built previously):

```
[classifier,chopinbase, mozartbase, debussybase, webernbase].
classify.      /* do the classification */
chopin1_testbase.      /* the first testing piece */
```

```
no.                     /* no. do not want to stop testing */
chopin2_testbase.       /* here is the next testing piece */
no.                     /* no. do not want to stop testing */
chopin3_testbase.       /* here is the next testing piece */
no.                     /* no. do not want to stop testing */
chopin4_testbase.       /* here is the next testing piece */
no.                     /* no. do not want to stop testing */
mozart1_testbase.       /* the first testing piece */
no.                     /* no. do not want to stop testing */
mozart2_testbase.       /* here is the next testing piece */
no.                     /* no. do not want to stop testing */
mozart3_testbase.       /* here is the next testing piece */
no.                     /* no. do not want to stop testing */
mozart4_testbase.       /* here is the next testing piece */
   .
   .
Yes.                    /* stop classifying  */
```

All the classification results then will be in the output file one_output.

To do the classification in real time, it is like the following:

```
%prolog -H 1500 -G 1500
C-Prolog version 1.4
| ?- [classifier, chopinbase,mozartbase,debussybase,webernbase].
classifier consulted 38600 bytes 9.649994 sec.
chopinbase consulted 41796 bytes 8.53 sec.
mozartbase consulted 41796 bytes 8.23 sec.
debussybase consulted 41796 bytes 8.76 sec.
webernbase consulted 41796 bytes 8.23 sec.

yes.
| ?- Which testing file to be reconsulted?
chopin1_testbase.
|: chopin1_testbase reconsulted 13500 bytes 2.566673 sec.
    grand_total(chopin, 0.298316).
    grand_total(mozart, 0.182775).
    grand_total(debussy, 0.154587).
    grand_total(webern, 0.094566).
    The composer for this piece is chopin!
| ?- Do you want to stop testing -- answer yes or no.
 no.
| ?- Which testing file to be reconsulted?
chopin2_testbase.
    .
    .
    .
```

Then keep testing like this till all testing pieces were classified.

10. Program Listings

The listings are on file with the Rochester Institute of Technology Graduate School of Computer Science and Technology.