PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1988,
edited by ICOT. © ICOT, 1988

HOW TO REALIZE JAZZ FEELINGS - A LOGIC PROGRAMMING APPROACH -

Keiji Hirata
NTT Software Laboratories
hirata@ntt-20.ntt.jp

Tatsuya Aoyagi
The University of Electro-Communications
aoyagi@cs.uec.ac.jp

Hiroki Konaka
The University of Tokyo
konaka@mtl.u-tokyo.ac.jp

ABSTRACT

This paper describes how to build a computer music system for generating jazz music. In particular, we consider a tool which generates fingering patterns for chord-voicing on real keyboards when a chord progression is given. This tool, called LPC88, produces solutions one by one, by communicating interactively with a user. Jazz chord theory, voicing knowledge and so on are declaratively expressed in a logic programming language. LPC88 is implemented on a PSI machine, written in ESP language. Based on experience in making the tool LPC88, we discuss a method of representing knowledge of jazz music, the user interface and the implementation. The paper also demonstrates how LPC88 takes advantage of the features of ESP.

1 INTRODUCTION

Computer music systems for sound synthesis, score editing, automated performance, education, analysis, experiment and so on, have recently become available. Popular or modern musicians already use a variety of computer systems. However, present systems fail to fully satisfy our requirements. There are several problems.

First, since present systems are developed independently, they are generally incompatible with each other. Therefore, we can not exchange or accumulate components to obtain an optimum combination. Second, almost all music systems are self-contained, i.e. they are closed in some way so that they lack expandability and flexibility. Finally, although computer music systems can not understand the musical meanings of tunes, users would let the machine do everything automatically. To overcome these difficulties, we are developing an advanced computer music system under the *ICOTone Project.* An ICOTone system is a workstation for musicians, especially for popular, jazz, and modern musicians.

This article presents LPC88, an ICOTone tool for generating fingering patterns for jazz chords on keyboards. LPC88 contains a lot of knowledge on jazz chord theory. When a chord sequence is given, LPC88 begins to construct the fingering patterns as a real jazz pianist does. Based on an experience in making the tool, we discuss a method for representing knowledge of jazz music, its user interface and its implementation.

LPC88 is implemented on a PSI machine (Yokota et al. 1984), written in ESP language (Chikayama 1984). PSI is a personal sequential inference machine developed by ICOT. ESP is a dialect of Prolog with object-oriented features. The operating system of PSI is also written in ESP, and provides rich window facilities.

2 JAZZ FEELINGS

Why does music played by jazz musicians sound jazzy ? Where do jazz feelings come from ? We think that the jazz sound originates in the following: a melody line including scale-out notes, the way of choosing chord

$$C_6 \rightarrow G_7 \rightarrow C_6$$

$$F_6 \rightarrow B^{\flat}_{maj7} \rightarrow C_7 \rightarrow F_6$$

Figure 1: Examples of Cadence

Chord Name	Chord Tones	Tensions
$G_{7(\flat 9)}$	G, B, D, F	$A^{\flat}$
$A^{\flat}_{7(9,\sharp 11,13)}$	$A^{\flat}, C, E^{\flat}, G^{\flat}$	$B^{\flat}, D, F$

Figure 2: Examples of Chord Tones and Tensions

tones, performing the chords, playing style of a bass part, so-called 4 beat rhythm etc. We will briefly describe such technical terms for jazz chord theory. In this article, we assume Be-Bop style jazz.

Keywords are cadence, tension, substitution chord and voicing. All the pieces are composed and performed, in a certain cadence structure. Cadence is a chord sequence which gives a feeling of termination or finish, and determines the tonality of the tune (Fig.1). Almost all standard jazz numbers have definite tonality.

Further, to increase the jazz feeling, the melody line and chord tones may include tensions. Tensions are notes other than normal chord tones (Fig.2). A chord containing tensions sounds *tense* and too many tensions may make the tonality ambiguous. Therefore, jazz musicians play tensions carefully.

In jazz music, one chord is often replaced another, which has the same function. This is called a substitution chord (Fig.3). Jazz musicians use substitution chords during performance as well as during composi-

$$C_6 \rightarrow \mathrm{G}_7 \rightarrow C_6 \approx C_6 \rightarrow \mathrm{D}^{\flat}_7 \rightarrow C_6$$

$$F_6 \rightarrow \mathrm{B}^{\flat}_{\mathrm{maj7}} \rightarrow C_7 \rightarrow F_6 \approx F_6 \rightarrow \mathrm{G}_{\mathrm{m7}} \rightarrow C_7 \rightarrow F_6$$

Figure 3: Examples of Substitution Chord

$F \rightarrow G \rightarrow C$ $\qquad$ $D_{m7} \rightarrow G_7 \rightarrow C_6$

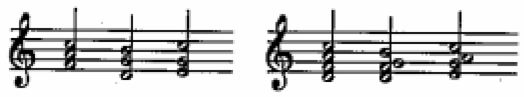


Figure 4: Examples of Voicing

tion.

In addition, a set of chord tones has various fingering patterns and a jazz musician usually performs a chord so that it is smoothly connected with the previous chord. Arranging the notes is called voicing (Fig.4). Wrong voicing weakens the jazz feeling.

There are many good text books on jazz music theory which describe these items. We can easily obtain the knowledge. Although jazz rhythm is also an important thing in jazz music, our system does not yet manipulate the jazz rhythm.

3 A TOOL FOR GENERATING JAZZ HARMONY: LPC88

3.1 Jazz Pianist and LPC88

A jazz pianist plays a tune according to a simple score which includes only the sequence of chord names (e.g. $G_{7(\flat 9)}$ C_{m7} F_{maj7} $\cdots$) and the melody line. Usually, he plays the melody with his right hand, and the chords with his left hand. He selects the chord notes, consistent with the sequence of chord names, and plays them in appropriate octaves. LPC88 generates such fingering patterns from the sequence of chord names as a jazz pianist does. Finally, LPC88 sends out performance data in the MIDI format (MIDI 1983) corresponding to the notes hit on the keyboard. Therefore, LPC88 contains much knowledge on jazz chord theory: cadence, chord functions, substitution chords, chord tones, tensions, and voicing.

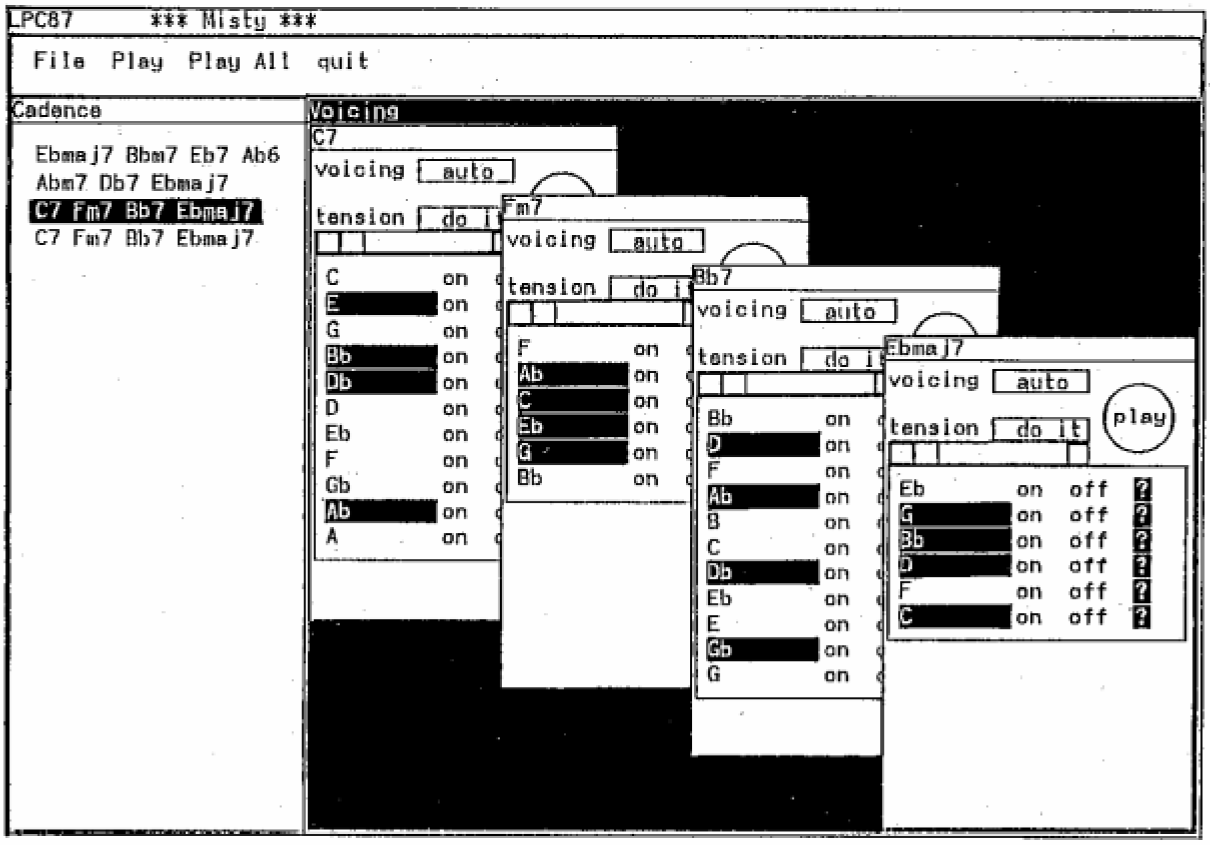


Figure 5: Example of Cadence Window

3.2 User Interface

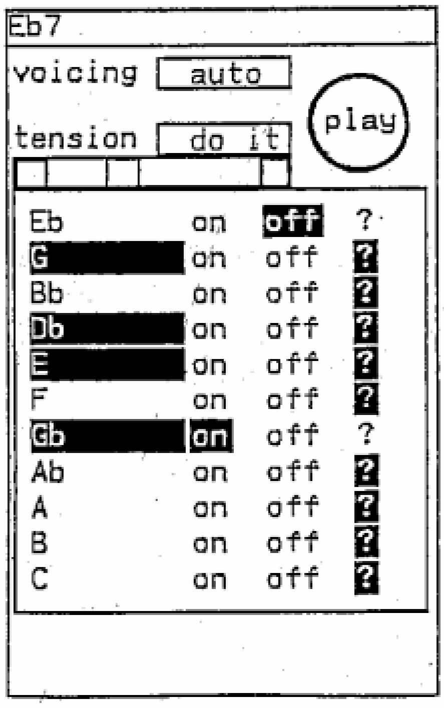
The following is an example of system's operation. First a user selects a target tune from a tune menu when LPC88 starts. LPC88 then shows chord windows corresponding to the chords in the cadence of the selected tune (Fig.5, Fig.6).

As shown in the Fig.5, the system displays a cadence window one at a time. The left side shows cadence selection, and the right side shows chord windows. As shown in Fig.6, the chord window contains the chord name, two scroll bars, the play button and the candidate notes, including tensions. A user can investigate alternative voicing patterns with the voicing scroll-bar and alternative tensions with the tension scroll-bar. The "on", "off" and "?" on the right of the note names indicates that, when voicing, each note is "included", "excluded" or "doesn't matter", respectively. A user sets these options as preferred and clicks the "do it" button. A user can also examine different solutions.

LPC88 has two voicing modes: "manual" and "auto" (manual mode in Fig.6). The voicing scroll-bar operates in the manual mode and the scroll bar determines the top note. In the auto-voicing mode, LPC88 does voicing appropriately.

At any time, LPC88 may play the whole or part of the tune (a selected cadence or a chord) to check an answer. If the user is not satisfied with the sound, he can try alternative four-note voicings using the scroll bar. Incidentally, the checking play is accompanied by a simple bass line and a cymbal rhythm.

As previously mentioned, there may be a number of alternatives for a chord. Therefore, a user can inter-



Eb7

voicing auto

tension do it

Eb	on	off	?
G	on	off	?
Bb	on	off	?
Db	on	off	?
E	on	off	?
F	on	off	?
Gb	on	off	?
Ab	on	off	?
A	on	off	?
B	on	off	?
C	on	off	?

Figure 6: Example of Chord Window

actively select tensions and voicing patterns for each chord in a cadence.

4 SYSTEM DESIGN AND IMPLEMENTATION ISSUES

It is important to design an class configuration in object-oriented languages. Basically, LPC88 consists of three parts: (1) the musical knowledge part, (2) the score data structure and (3) the user interface.

4.1 Musical Knowledge Part

Fig.7 shows the execution flow of the musical knowledge part. When a chord sequence is given, LPC88 begins to analyze it. A user has written the chord sequences into a file in advance. First, it finds the cadences by using its knowledge of cadence and substitution chords, and determines the key from this cadence analysis. Then it decides the chord functions according to the key. Next, LPC88 makes a list of the chord tones and tensions for each chord, considering the chord function. Notice

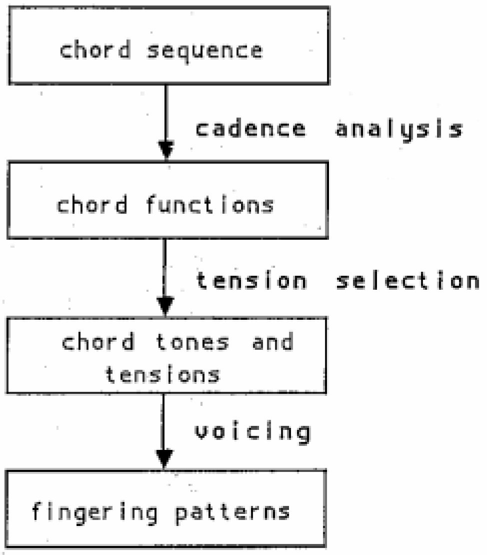


Figure 7: Execution Flow of Musical Knowledge Part

that the execution proceeds until this point without a user interaction. LPC88 does not throw away any alternatives until this stage. Then, LPC88 selects four appropriate notes from the list, following the tension-selection rules and user's directives. Finally, it plays the four notes as smoothly connected with the previous chord notes (voicing).

The cadence analysis algorithm checks if a new cadence, which begins with each candidate of key and chord function, starts, or a cadence finishes. At the same time, the algorithm tries to continue cadences which does not yet finish.

The below ESP program segment represents part of musical knowledge used for the cadence analysis.

```
cadence(['D','T']);
cadence(['SD','T']);
cadence(['SDm','T']);
cadence(['SD','D','T']);
cadence(['SD','SDm','T']);
cadence(['SDm','D','T']);
cadence(['SD','SDm','D','T']);
```

In the above program, 'T', 'D', 'SD' and 'SDm' represent chord functions.

When analyzing cadences, the system also uses musical knowledge about substitution chords as follows. Two chords which have the same function can be substitute each other.

```
function(('II',m7),'SD');
function(('VII',7),'SD');
function(('IV',7),'SD');

function(('II',m7b5),'SDm');
function(('bVII',7),'SDm');
function(('bII',maj7),'SDm');
function(('bVI',7),'SDm');
function(('bVI',maj7),'SDm');
```

Next, We show some of tension selection rules as follows:

- V_7 chord allows tension notes with degrees of ♭9, 9, ♯9, 11, ♯11, ♭13 and 13.
- there are a couple of notes which can not be included in chord notes simultaneously. For instance, ♭9 and 9, 11 and ♯11, ♭13 and 13.
- root note should be avoided and one of ♭9, 9 and ♯9 should be included, instead.

LPC88 computes all of the available chord notes and tensions. Moreover, all of the possible combinations of four notes selected from them are generated and saved in advance. Through interaction with a user, the system picks up suitable ones from the saved candidates. The system arranges a set of the four notes in proper order.

Voicing is performed on demand when a user pushes the "play" button. Since the selected four notes should be located within an octave. the system repeats inversions until the notes are appropriately placed on keyboards. That is, the system considers what fingering pattern is most smoothly connected with that of the previous chord.

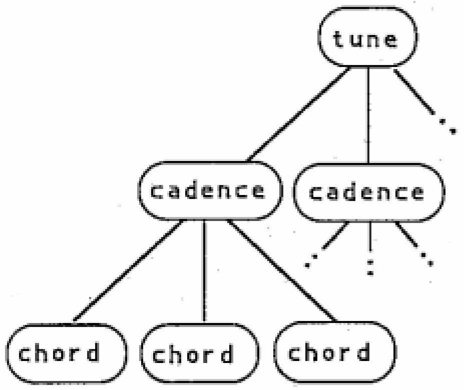


Figure 8: Score Data Structure

key	:	F
root note	:	C
type	:	$7th$
root degree	:	5
interval	:	$quarter$
chord tones	:	$C, E, G, \cdots$
tensions	:	$D^\flat, D^\sharp, F, \cdots$
four notes candidates	:	$(E, G, B^\flat, D), (B^\flat, D, E, G),$

Figure 9: Slots of Chord Object

4.2 Score Data Structure

The score data structure reflects the structure of a tune (Fig.8). In general, a tune includes more than one cadence, a cadence includes more than one chord and so on.

The slots and their values produced by the musical knowledge part are shown in Fig.9. The figure depicts an example for C_7 in key F. As mentioned above, LPC88 set these slots values before a user begins to give his directives.

4.3 User Interface

LPC88 is implemented with the well-known Model-View-Controller (MVC) mechanism (Xerox 1980) allowing modularity high enough for separating the mu-

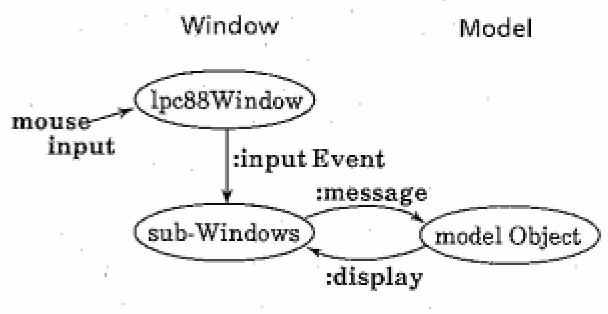


Figure 11: Event Propagation among Objects

sical knowledge part from the user-interface part [1]. Indeed, the windows manage user-interface, and each object in the score structure computes answers.

The windows and the score structure have the hierarchical structure corresponding to each other in our implementation (Fig.10). It is natural that a user manipulates the score data through the windows which reflects the structure of the score data. In Fig.10, each oval represents an instance of each class. Such an implementation technique enables us to realize the complicated user-interface systematically.

We show the way events propagate over objects in LPC88 (Fig.11). All the input events are received by `lpc88window` which is the top of the window objects. Then, `lpc88window` dispatches the events to the appropriate subwindows. The subwindow issues messages to the corresponding model. In this case, a model means an object in the score data structure. After the model finishes processing the messages, it sends the redraw message to the corresponding subwindow with a new state.

5 COMPARISON WITH RELATED WORK

One of the earliest works on harmony analysis by machines was done by T. Winograd (Winograd 1968). However, there are few systems for computer jazz music.

[1]Our implementation does not divide View and Controller. Both are included in Window.

5.1 Related Work

5.1.1 Ulrich's System

Ulrich has treated the jazz improvisation problem by using both analysis and synthesis components (Ulrich 1977). In his system, a left-to-right analysis is performed and he gleans the names of chords from chord inversions. Ulrich's program analyses the entire piece in terms of keys involving a mode more rigidly tied to its root. Then, the system fits the melody to a given harmonic background. At that time, an abstract motif is used, which is like a motif except that it is devoid of rhythmic and harmonic structure. This abstract motif can be a real melody after being instantiated with respect to harmony. The method of rhythmic and harmonic instantiation is specified before hand in an ad hoc manner. Ulrich said that such a simple technique is bound to produce less than interesting music although the music is consistent with the song's harmonic structure.

5.1.2 Levitt's System

Given an initial melody and harmonic outline, the program produces a single voice of improvisation, which is intended to resemble a New Orleans jazz solo (Levitt 1981). First, Levitt's system divides the initial melody into 2-measure phrases, and then exploits some interesting features from them: chord-consonant pitch, intervallic leap i.e. large/small or ascending/descending, repeated notes, and rhythmic syncopation. Second, the system chooses a proper 2-measure source phrase according to the degree of interest and generates candidates by a variation-making algorithm. This algorithm is a procedure for creating phrases which inherit from the source phrase some features which are fixed in advance. The algorithm is a constraint satisfier. Third, the system tries by heuristics to successively eliminate the candidates until only

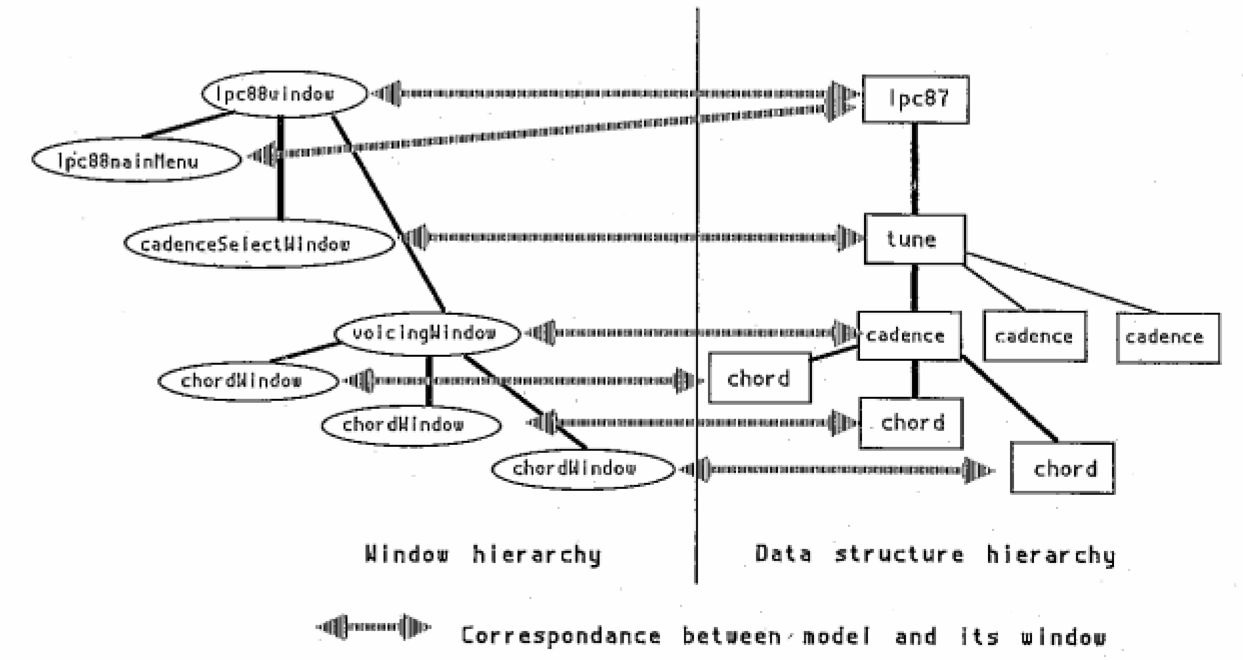


Figure 10: Relation Between Score Structure and Windows

one remains. Levitt said that human soloists have a much larger vocabulary of features to imitate, *and* use them more judiciously.

5.2 Problems

It is a major problem that the previous systems are not flexible. They lack flexibilities in some aspects as below.

- Amount of information given by a user: In some case, a user may want to let the system generate almost part of his performance. In other case, he may want to examine every note to play. It is natural that the amount of information which a user would give to the system is varied, as the song position changes and as the stage of his music creating process proceeds.

  The previous systems require the constant amount of information; for example, they need only chord progression. Even if a user wants to give more precise or other information, the systems cannot accept it.

- User interface: One style of user interface is not always the most suitable for all users. A user may prefer a window-menu-based user interface, and another user may would use a command-interpreter interface.

  The previous systems provide only one kind of user interface, and cannot extend their own interface. Moreover, it is not easy to modify them to provide other styles of user interface.

- Musical knowledge in system: We do not have a music theory which is applicable in any situation. Thus, the musical knowledge in a system must be modifyable.

  The previous systems cannot modify their knowledge or the algorithms flexibly as needed.

5.3 Our Approach

The logic programming is very helpful for representing the musical knowledge declaratively. This makes each musical knowledge independent from others. We wrote the program of LPC88 with the high modularity, which comes from one the object-oriented features of ESP. Thus, we can divide the program into some independent parts to increase the flexibilities of our system.

We solved the problems described in the previous section as follows:

- Amount of information given by a user: The system receives user's input in two ways: chord progression in a file and other directives through windows. The former information is written in a file in advance. With an interaction between a user and the system the user gives the latter information. The amount of the latter information to be provided depends on the user's intention.

  The processes of LPC88 are also categorized into two part. One is a process which runs only once at the beginning of the system execution such as the cadence analysis process. Another ones perform whenever a user inputs additional information and presses the "do it" button.

- User interface: Since LPC88 is implemented with the MVC mechanism, the user interface is completely independent from the other processing parts. Therefore, programmer can easily exchange for other kinds of interfaces, such as a command-interpreter interface.

- Musical knowledge in system: We wrote the musical knowledge as declaratively as possible. This simplifies adding or deleting musical knowledge to or from the knowledge base. This enables an answer to be deduced from musical axioms and rules. We can investigate the musical meanings of musical axioms and rules formally, not heuristically (Ebcioglu 1987).

6 CONCLUSION

We have successfully formalized the major part of jazz chord theory. Owing to the logic programming feature of ESP, we can represent the musical knowledge declaratively. In addition, the object-oriented feature of ESP allows high enough modularity to isolate various knowledge from each other.

In addition to this formalization, we think that the user interface of LPC88 suggests an advanced style of tool for generating computer music. Indeed, we can obtain the preferred answers immediately and have proved interaction between users and a system to be important, especially in computer music systems.

REFERENCES

(Chikayama 1984) T. Chikayama: Unique Features of ESP, Proc. of The International Conference on Fifth Generation Computer System '84 Japan (1984).

(Ebcioglu 1987) K.Ebcioğlu: An Efficient Logic Programming Language and its Application to Music, in Logic Programming : Proc. 4th Int'l. Conf., J.-L.Lassez (ed), MIT Press (1987).

(Levitt 1981) D. Levitt: A Melody Description System for Jazz Improvisation, M.S. thesis, MIT Dept. of Electrical Engineering and Computer Science (1981).

(MIDI 1983) International MIDI Association: MIDI Specification 1.0, International MIDI Association, North Hollywood, CA (1983).

(Pennycock 1985) B.W. Pennycock: Computer-Music Interfaces : A Survey, ACM Computing Surveys, Vol.17, No.2, pp267-289 (1985).

(Ulrich 1977) W. Ulrich: The Analysis and Synthesis of Jazz by Computer, In Proceedings of the 5th IJ-CAI (1977).

(Winograd 1968) T. Winograd: Linguistics and the Computer Analysis of Tonal Harmony, Journal of Music Theory, Vol.12, No.1, pp2-49 (1968).

(Yokota et al. 1984) M. Yokota et al.: A Microprogrammed Interpreter for The Personal Sequential Inference Machine, Proc. of The International Conference on Fifth Generation Computer System '84 Japan (1984).

(Xerox 1980) Xerox: StandardSystemView, StandardSystemController etc., The source code of Smalltalk-80 system.