音楽情報科学 30−11
(1999. 5. 30)

複数流派への拡張が容易な尺八譜情報処理システム

松島 俊明* 坪井 邦明** 志村 哲***
*東邦大学 **千葉職業能力開発短期大学校 ***大阪芸術大学
matusima@is.sci.toho-u.ac.jp tsuboi@chiba-pc.ac.jp simura@osaka-geidai.ac.jp

尺八には様々な流派があり，尺八譜に用いられる譜字やその運指法／奏法は，流派毎に異なっている．筆者らは尺八譜の情報処理システムの研究を行っているが，このような違いがデータの共有化やシステム開発の効率化の妨げとなっていた．この問題を解決するために，昨年度，尺八譜のための標準データ形式COMSOを提案した．今回，COMSOを内部コードとして用いることにより，開発中の尺八譜情報処理システムを複数流派への拡張が容易に行えるようになったので報告する．

The Shakuhachi Tablature Information Processing System Easy to Extend Multiple Shakuhachi Schools

Toshiaki MATSUSHIMA*, Kuniharu TSUBOI** and Satosi SIMURA***
*TOHO University **Chiba Polytechnic College ***Osaka University of Arts

There exist some Shakuhachi schools. Fuji (notation character in Shakuhachi tablature) represents fingering and blowing techniques, and a set of Fuji used in each school is different among schools. Although we have been developing the Shakuhachi tablature information processing system, these differences tend to be an obstacle for the computer processing of the Shakuhachi tablature. In order to solve this problem, we proposed COMSO (COMmon representation language for Shakuhachi nOtation) last year. This paper reports the new Shakuhachi tablature information processing system that adapts COMSO as the internal data code and is easy to extend multiple Shakuhachi schools.

1 はじめに

筆者らは尺八譜を対象とした様々な情報処理システムの研究を行っている[2]．尺八譜は奏法を記述した奏法譜であり，奏法を示す譜字によって記述されている．しかし，尺八には都山流[7]，琴古流，上田流，竹保流[8]などのほか様々な流派があり[6]，流派により尺八譜の記述に用いる譜字が異なっている．また，同じ譜字を用いている場合にも，指遣い，メリ・カリの技法，複符運指法，譜字の形状(フォント)などが，全くあるいは微妙に異なっていることが多い．筆者らの研究に限っても，このような流派による譜字の違いが尺八譜データの共有化や尺八譜処理システム開発の効率化の妨げとなっていた．筆者らはこの問題を解決するために，様々な流派の尺八譜を統一的に記述できる譜字コードおよび記述言語の体系COMSO (COMmon representation language for Shakuhachi nOtaion) を提案した[1]．今回，尺

八譜の手書き入力編集システム[3,4,5]をCOMSOに基づくシステムとして設計し直すことにより，複数の流派への対応が容易に行えるようになったので，その具体的内容について報告する．

2　複数流派への対応方法

従来のシステムは都山流尺八譜のみを対象としていたが，そのため都山流専用のファイル入出力・認識・表示などの流派固有の処理もシステム本体の中に組み込まれていた(図 1)．新たに竹保流を対象として加えるにあたり，同様の方法によって竹保流の流派固有処理もシステム本体に組み込む形式を取ると，新しい流派を追加する度にプログラム全体を変更・再構築する必要が生じ，大変非効率的である．そこでシステム本体には流派によらない共通処理のみを入れ，流派固有の処理は流派毎に Windows の DLL (Dynamic Link Library) ファイルとして作成し，システム起動時に動的に呼び出すようにすることとした．すなわち，流派固有処理の Plug-In 化である(図 2)．このように流派固有の処理を Plug-In 化することによって

- 新しい流派をシステムに追加する場合は，その流派のフォントファイルとPlug-In DLLファイルを作成すれば良く，共通処理からなるシステム本体に手を加える必要がない．
- 対応中の流派の処理に変更を加える必要が生じても，他の流派用のプログラムへの影響なく修正が可能である．
- 流派毎に手書き入力の認識アルゴリズムを変えることができるので，流派毎に譜字の書体や筆記方法が異なる尺八譜にとって大いに有効である．

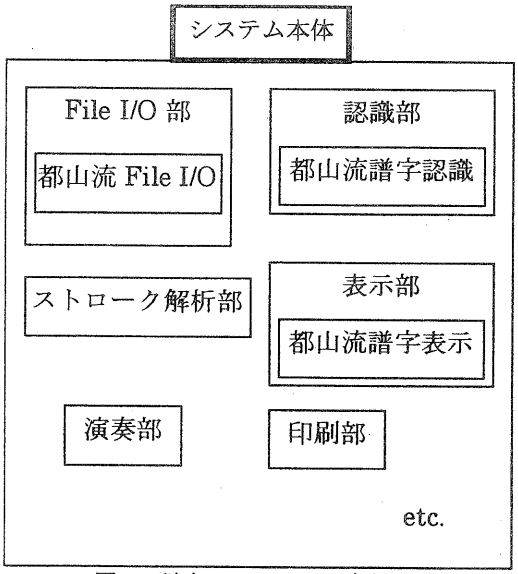

図 1　従来のシステムの内部構造

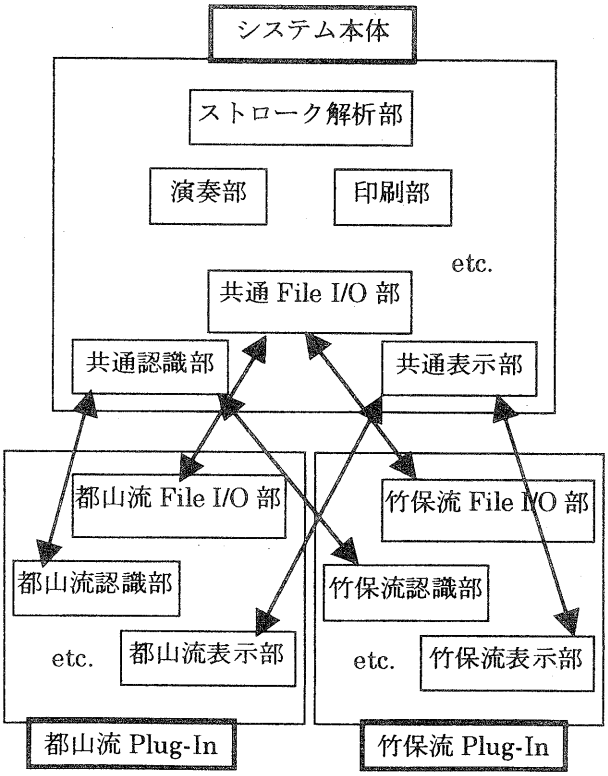

図 2　複数流派へ対応したシステム構成

など尺八譜の処理システムにとって非常にメリットが大きい．このように流派固有の処理を Plug-In 化するために，従来のシステムに対して(1) プログラム内部における尺八譜データのデータ構造および譜字コードを変更，(2) 流派毎の処理を DLL として分離，(3) 譜字表示に使用する TTF(True Type Font) ファイルを流派毎に作成，等の追加・変更を行った．

2.1 内部データ構造の変更

従来のシステムでは尺八譜を小節の集合，小節を音符の集合と捉え，二重のリスト構造で表していた．しかし，一般的に尺八譜では拍や小節の概念が明確であるとは限らない．そこでシステム内部における尺八譜データを，図 3 に示したように NoteC 型構造体の双方向リストとして保持するように変更を行った．各々の譜字情報は，譜字の流派(ryh)，種類(type)，譜字コード(fuji)，音価(onka)等のメンバで構成されている．譜字と譜字以外のデータは構造体の種類で区別する．譜字コードは COMSO 譜字コードで表しているため，流派に依存することなく譜字を表すことができる．また，流派情報も各譜字毎に指定できるため，複数の流派の譜字が混在するような尺八譜を表すことも可能である．

```
typedef struct _ScoreC{
        int     len;            /* 管長[一尺八寸=18] */
        int     bpm;            /* テンポ*/
        int     tsg;            /* 拍子 */
        char    tit[80];        /* タイトル */
        NoteC   *note;          /* 最初の譜字へのポインタ */
} ScoreC;
```

```
typedef struct _NoteC{
        int             ryh;            /* 流派番号 */
        int             type;           /* 構造体の種類 */
        struct _NoteC   *next_note;     /* 次の小節構造体へのポインタ */
        struct _NoteC   *prev_note;     /* 前の小節構造体へのポインタ */
        int             fuji;           /* 譜字コード */
        int             onka;           /* 音価 */
        int             oct;            /* 甲乙 */
        int     x0,y0,x1,y1;            /* 譜字枠小節線枠表示位置 */
} NoteC;
```

図 3 内部データ構造

2.2 流派固有の処理の DLL 化

流派固有の処理を DLL 化した例として，譜字認識処理を例に説明する．例えば，譜字入力枠内に"ロ"を表す"｜"，"┐"，"－"の3つのストロークが入力された場合，これらのデータはまず共通処理部であるストローク解析部により冗長なデータの除去やストロークデータ構造への格納などの処理が行われる．次に，これらのストロークデータを引数としてストローク認識関数 S8C_DLL_Reco_RecoStrokes( )が呼び出されるが，その際，図 4 に示したように流派情報 ryh により呼び出されるストローク認識関数が切り替えられる．認識が

正しく行われたとすると，流派が都山流の場合(ryh = 0)は譜字コード fuji = 6334 が設定され，流派が竹保流の場合(ryh = 1)は譜字コード fuji = 6333 が設定される．このように流派情報により呼び出す関数を切り替えることにより複数流派の譜字認識が可能となる．

2.3 フォント切り替えによる譜字表示

ファイル入出力部や譜字認識部は呼び出す DLL 関数の切り替えによって複数の流派を扱っているが，譜字表示は使用するフォントの切り替えによって行う．例えば図 5 のように全く同じ譜字コードを持つ2つの NoteC 構造体: nt1,nt2 があったとする．nt1 の譜字は流派情報 ryh = 0 であるので都山流フォントで文字コードが 6334 の文字「ロ」が x0,y0,x1,y1 で囲まれた矩形に表示され，nt2 の場合は ryh = 1 であるので竹保流フォントで文字コードが 6334 の文字「フ」が表示される．このように複数流派の譜字表示はメンバ ryh を参照して使用するフォントを切り替える事によって行った．

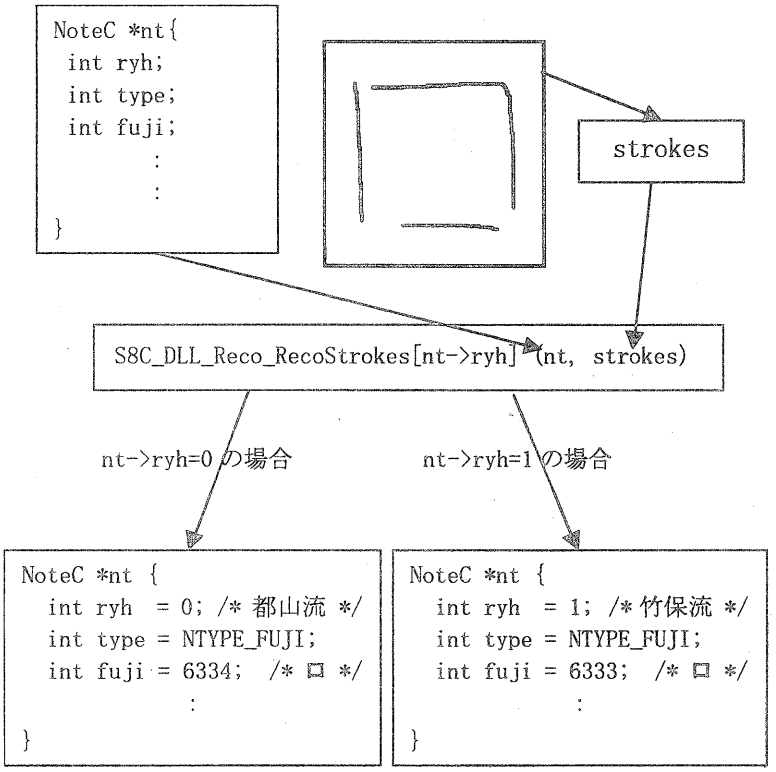

図 4　譜字認識処理例

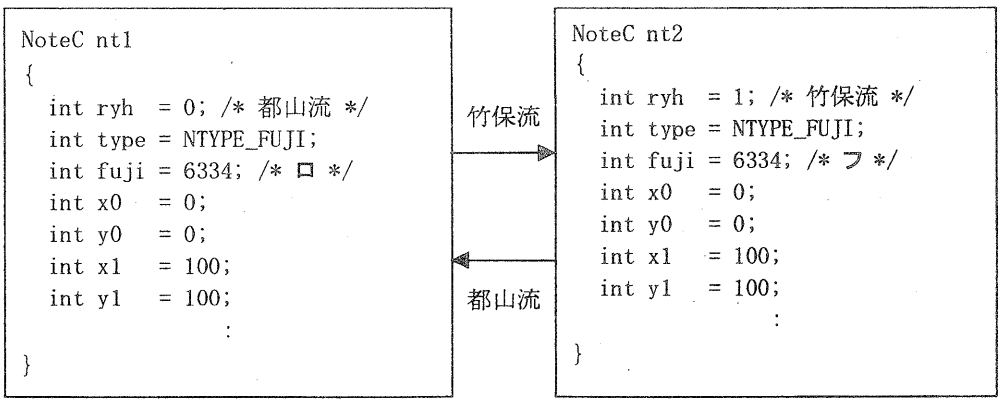

図 5　譜字の流派変更

3　COMSO 譜字コードの追加

純粋な譜字ではないが，COMSO Stage.1 では定義していなかった休符・拍子線にも譜字コードの規則に従い新たに譜字コードを割り当てた．これらの指遣いとは関係ない譜字等については，尺八では全部の孔を開いた音は演奏には用いられないことを考慮して，上位 7 ビットのうち指遣いに当たる 5 ビットを 00000 とした．また，下位7ビットのうち2ビットを休符・拍子線の区別に，5ビットを時価および付点を表すこととした．表 1 および表 2 に休符および拍子線の譜字コードを示す．

4　まとめ

図 6 は本システムによる尺八譜の表示例である．都山流で作成した尺八譜の後半を竹保流に変更した例である．このように，ある流派で作成した尺八譜を他の流派の尺八譜に変更することや，複数流派が混在した尺八譜も作成することができ，流派によらない尺八譜の情報処理システムとして非常に有用と考えている．現在対応しているのは，都山流と竹保流の2流派であるので，その他の流派への拡張と合わせて今後システムの改良を行っていく予定である．

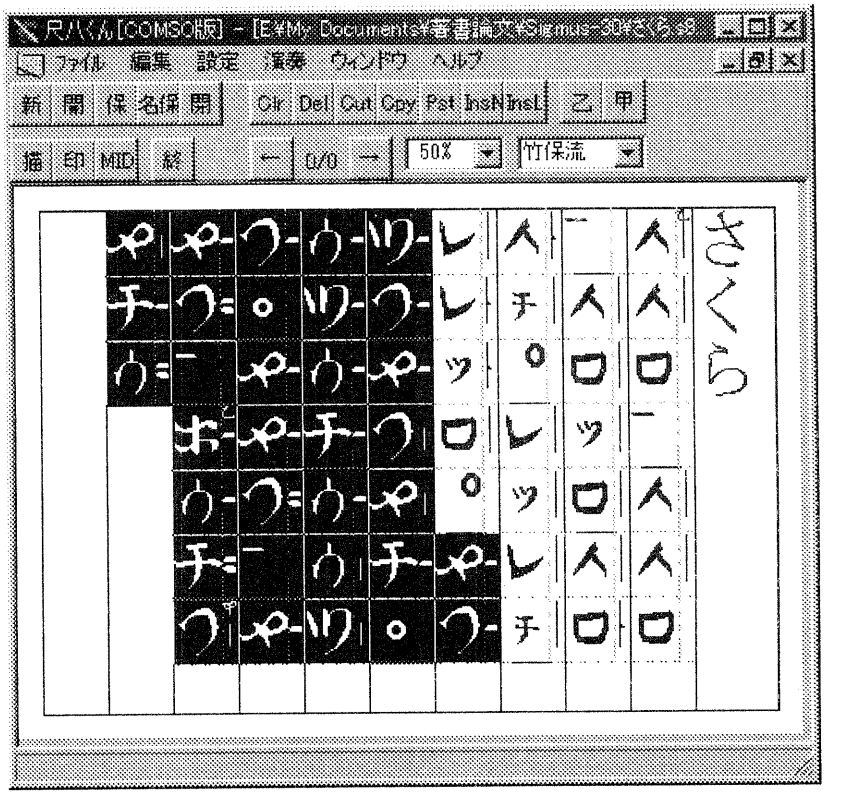

図 6 新システムによる表示例

参考文献

[1] 松島俊明，坪井邦明，志村哲: COMSO: 尺八譜のための標準データ形式，情処研報 MUS-26-2, 1998.

[2] 志村哲，坪井邦明，松島俊明: 日本音楽の情報処理 -尺八の場合-，情処研報 MUS-2-3, 1993.

[3] 長沢理恵，坪井邦明，松島俊明: 尺八くん -尺八譜の手書き入力・編集マルチメディアシステム-，情処研報 MUS-8-8, 1994.

[4] T. Matsushima and R. Nagasawa: "Multimedia System for Shakuhachi Tablature," Proc. ICMC'95, pp. 487-488, 1995.

[5] T.Matsushima: "The Recognition and Editing System for Shakuhachi Score," Proc. ICMC'93, pp.405-407, 1993.
[6] 尺八, 音楽大辞典, 第3巻, pp.1052-1063, 平凡社, 1982.
[7] 上野堅實: 尺八音楽のための楽典, 島田音楽出版, 1986.
[8] 酒井竹保: 竹保流尺八の手引, 竹保流尺八宗家出版部, 1971.

表1 休符コード表

都山流	竹保流	区点コード	識別	指遣	休符	時価	付点
∨		3232	01	00000	01	000	00
∨		3233	01	00000	01	000	01
∧		3236	01	00000	01	001	00
∧		3237	01	00000	01	001	01
○		3240	01	00000	01	010	00
𝄽		3244	01	00000	01	011	00
𝄾		3248	01	00000	01	100	00
𝄿		3252	01	00000	01	101	00

表2 拍子線コード表

都山流	竹保流	区点コード	識別	指遣	拍子線	時価	付点
		3264	01	00000	10	000	00
		3265	01	00000	10	000	01
		3268	01	00000	10	001	00
		3269	01	00000	10	001	01
		3272	01	00000	10	010	00
		3273	01	00000	10	010	01
		3274	01	00000	10	010	10
		3276	01	00000	10	011	00
		3277	01	00000	10	011	01
		3280	01	00000	10	100	00
		3281	01	00000	10	100	01
		3284	01	00000	10	101	00
		3285	01	00000	10	101	01