ヒューマンインタフェース 92－2
（2001．1．17）

実演奏の表情情報を用いた作曲のための他パートシミュレーション

中川 渉† 蔵川 圭† 中小路 久美代†,††,†††
e-mail: {wataru-n, kurakawa, kumiyo}@is.aist-nara.ac.jp

† 奈良先端科学技術大学院大学 情報科学研究科
†† (株) SRA ソフトウェア工学研究所
††† 科学技術振興事業団 TOREST

概要 計算機を用いることにより，複数の楽器で構成された演奏のシミュレーションを行いつつ作曲を行うことができる．しかし，作曲者にとって納得の行くシミュレーションを行うためにはそれぞれの音の発生時刻や大きさを細部にわたり調整する必要があり，音楽演奏データの処理に関する専門性と労力が要求される．本研究の目的は，そのような労力を軽減し，繰り返し演奏のシミュレーションを行うことで作曲を支援するシステムを構築することである．作曲支援インタラクティブシステム CAPADY(a Composition Assistance by Producing Agogics and DYnamics) は，音の発生時刻や大きさの変化を自由に表現できる実際の楽器演奏を用いて，計算機上に作成された異なる楽器の演奏シミュレーションにおける音の大きさ，発生時刻の調整を行う．

Musical Composition Support through the Extraction and Application of Expressive Features Based on Human Performance

Wataru Nakagawa† Kei Kurakawa† Kumiyo Nakakoji†,††,†††

† Graduate School of Information Science, Nara Institute of Science and Technology
†† Software Engineering Laboratory, Software Research Associates, Inc.
††† TOREST, Japan Science and Technology Corporation

Abstract This paper presents an approach to support the production of expressive music prototypes. With this approach, expressive features are automatically extracted from a piece of human musical performance and are applied to produce agogic and dynamic accents for another piece of musical notation in accordance with the original musical performance. CAPADY (a Composition Assistance by Producing Agogics and DYnamics) has been built in MAX based on the approach to help a musical composer construct music prototypes using multiple timbres.

1 はじめに

人間が知的創造作業を行う際に，計算機を利用することが多くなりつつある．本研究では，応用問題領域の一つとして音楽を取り上げ，音楽における代表的な知的創造作業である作曲を，計算機を用いて支援する場合について考察する．

人間の性質と計算機をはじめとする機械の性質とは大きく異なるものであり，それぞれの長所を生かして補完的に作業を行うことができるように機械を設計することが重要である [1]．本研究ではこの考えに基づき，演奏シミュレーション作成における表情付けの一手法を提案する．

ここで言う表情付けとは，計算機上で作成した音楽演奏データに対し，それぞれの音の大きさを変化させたり音の発生する時刻に数ミリから数十ミリ秒の変化をつけたりすることにより音楽演奏データを調整することを指す．表情付けを行うための情報を表情情報と呼ぶことにする．

提案手法に基づいて構築中のシステム CAPADY(a Composition Assistance by Producing Agogics and DYnamics)は，人間と計算機が協調して計算機上の音楽演奏データに対して表情付けを行うシステムである．利用者は任意の楽器を実際に演奏することによって表情情報を計算機に入力する．計算機は入力された表情情報を利用して，表情情報の入力に用いられた楽器とは異なる楽器の，あらかじめ計算機に入力済みの音楽演奏データを調整する．

CAPADY は作曲支援インタラクティブシステムである．Sloboda[2] は，作曲過程においては (a) 解候補を生成し，(b) それを評価する，という作業が繰り返し行われ，この (a) の過程は長年の経験によって獲得した作曲手法のレパートリに基づいて行われるとしている．

CAPADY は，調整された音楽演奏データを用いて利用者の創造性を喚起し，繰り返し利用することによって作曲の支援を行うことを目指している．すなわち，利用者の意図に基づいた様々な調整がなされた演奏を生成することにより，(a) における解候補の生成を支援しようとするものである．

本研究では作曲を行う主体は人間であり計算機はそのための道具であるという立場をとる．したがって，本研究と自動伴奏，自動表情付け等に関する研究 [4] [5] [6] [7] とは，技術的に共通する点は多く存在するが，目標が大きく異なる．

以下，2 章では本研究で提案する利用者と計算機の協調による表情付けについて述べ，その実現手法について述べる．3 章では提案手法に基づいて構築中のシステム CAPADY(a Composition Assistance by Producing Agogics and DYnamics) が作曲過程においてどのように利用されるかについて概観する．4 章では現在のプロトタイプシステムにおける問題点とその対策について考察し，5 章で本論文を結ぶ．

2 人間と計算機の協調による表情付け

本章では人間と計算機の協調による表情付けについて述べ，提案手法をシステムとして実現する方法について述べる．

2.1 実演奏と計算機上での表情情報の表現方法の違い

数値によって表現された計算機上の音楽演奏データを調整する場合，数値を直接編集することが多い．そのため音の大きさや発生時刻の変化を数値として明確に表現する必要がある．

ところが，実際に楽器を演奏する際に音の大きさを変化させたり音の発生する時刻を変化させたりすることは実際には無意識のうちに行われている場合が多いため，音の大きさや発生時刻を計算機上で使用する数値として明確に把握することは困難である．

そこで本研究では，計算機上の音楽演奏データを調整する際には，実演奏の場合のように，音の大きさや発生時刻について明確に把握することなく行えることが望ましいという立場をとる．

2.2 アプローチ

利用者が任意の楽器を実際に演奏することによって表情情報を計算機に入力することは，表情情報を明確に把握することなしに入力することができる方法の一つである．そこで本研究では，次のようなアプローチをとる．

システムの利用者は，あらかじめ楽譜入力インタフェース等を用いて楽器 B の音楽演奏データを入力する．この楽譜入力インタフェース等で入力した音楽演奏データは音の大きさや発生時刻に変化がついていないものとする．次に楽器 B に適用する表情情報を入力するために，楽器 A を実際に演奏する．システムは，利用者が行った楽器 A の実演奏から表情情報を抽出し，その表情情報を用いて楽器 B の音楽演奏データを調整する．利用者は音の大きさや発生時刻を調整された計算機による楽器 B の演奏を聞き，利用者にとって納得の行くものであるかを判断する．さらに，納得が行くものではないと判断した場合には，再び楽器 A を実際に演奏して表情情報に変更を加えたり，基となる音楽演奏データそのものを変更したりする．

以上のように一連の作業を繰り返し行い，利用者にとって納得の行く楽器 B の音楽演奏データを作成する．インタラクションの概要を図 1に示す．

2.3 システム実現における課題

上記のインタラクションを実現する上で特に重要となるのは次の 2 点である．

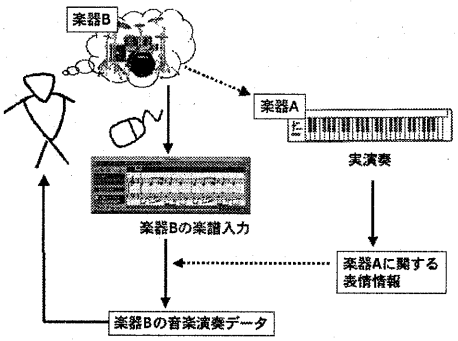

図 1: 目標とするインタラクション

1. 利用者の実演奏の内容に関しては，楽譜情報等をあらかじめシステムに入力しておく必要がないこと．
2. 利用者の実演奏の中では休符であるような部分に対しても，計算機上の音楽演奏データを調整できること．

以下では，楽器 A としてキーボード，楽器 B としてドラムをそれぞれ例にとり詳細を述べる．以下の議論においては，キーボードとドラムがそれぞれ持つ性質を利用して単純化している部分がある．そのため他の楽器に適用する場合には，今後検討が必要である．
提案するシステムは，(a) 入力されたキーボード実演奏から表情情報を抽出し，(b) ドラムの音楽演奏データに適用できるように加工し，(c) ドラムの音楽演奏データに適用する．さらに (d) 調整されたドラムの音楽演奏データを再生する．それぞれの段階についてシステムを実現する上で必要となる事項を述べる．

(a) 抽出　本研究では，それぞれの音の発生時刻や大きさのパラメータを直接出力することができる，いわゆる MIDI(Musical Instrument Digital Interface) キーボードを利用することにする．ここで MIDI キーボードから出力されたそれぞれの音が何小節目，何拍目の音であるかをシステムが認識する必要がある．利用者のキーボード実演奏の内容について，楽譜等の情報をあらかじめ計算機に入力しておき，計算機がその楽譜を利用しながら利用者が行う実演奏を監視するという方法も考えられる．しかし，この方法では，利用者が演奏内容を変更するたびに楽譜を入力する必要があり，上記のインタラクションを妨げる要因となる．入力される実演奏の楽譜等を入力する必要なしに時間的に変化のあるそれぞれの音が何拍目にあたるかをシステムが認識する必要がある．

(b) 加工　キーボード実演奏におけるそれぞれの音が何拍目にあたるかを認識することができれば，同じ拍にあたる音について，キーボード実演奏から抽出した音の大きさ，発生時刻に関するパラメータをドラムの音楽演奏データに適用することができる．しかし，キーボードでは休符である部分においてもドラムでは音があるという場合については，注意を払う必要がある．
また，同じ拍にあたる音についてパラメータを適用する本手法では，利用者がキーボード演奏によって入力した表情情報が必要以上に誇張されたもので，ドラムに直接適用するには適さない場合も考えられる．この場合はパラメータの値を調整する機能も必要である．

(c) 適用　調整の対象となるドラムの音楽演奏データについても，あらかじめ楽譜入力インタフェース等を用いて入力した際に設定したテンポの値をもとに，それぞれの音が何拍目にあたるかをシステムが調べる必要がある．

(d) 再生　調整されたドラムの音楽演奏データが利用者にとって納得の行く音楽演奏データかどうかを利用者が判断するために実際に再生する．調整されたドラムの音楽演奏データを再生するだけでなく，利用者が行った実演奏と同時に再生する，あるいは比較のために異なるドラムの音楽演奏データを再生することができる必要がある．

なお，表情情報としてそれぞれの音の発生時刻，音の大きさおよび音の長さという 3 種類のパラメータをとることが多い [6] [7]．しかしここでは，調整の対象がドラムの音楽演奏データであるため，音の長さを示すパラメータの調整は行わず，音の発生時刻と大きさのみを処理の対象とする．

2.4 システム実現の手法

上記の要求を実現するため，ここでは次のような方法をとる．図 2は処理の手順を示したものである．

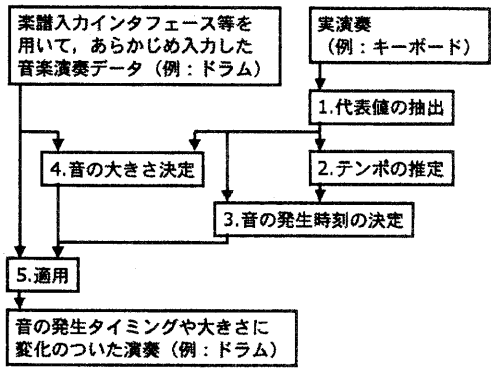

図 2: 提案システムにおいて行う処理の手順

ここでは処理の単位を 16 分音符とする．利用者はキーボードの実演奏とドラムの音楽演奏データ以外に，次の数値を入力するものとする．

- 時間窓サイズ：同じ拍とみなす時間幅 [ms]
- $T_{keyboard}$：実演奏のおおまかなテンポ [bpm (beats per minute)：一分間あたりの 4 分音符の個数]
- T_{drums}：計算機上の音楽演奏データをあらかじめ入力した際に設定したテンポ [bpm]
- w：重み係数 $(0 \leq w \leq 1)$

利用者が行ったキーボード実演奏から代表値を抽出し (1)，それを用いて実演奏のテンポを推定する (2)．実演奏から抽出した代表値の列において n 番目の音の大きさを $A_n^{(l)}$ とし，発生時刻を $A_n^{(t)}$ とする．また，楽譜入力インタフェース等を用いて，あらかじめ入力したドラムの音楽演奏データにおける n 番目の音の大きさ，発生時刻をそれぞれ $B_n^{(l)}$，$B_n^{(t)}$ とする．実演奏から抽出した代表値，推定テンポおよび重み係数から，音の発生時刻を示すパラメータの列を作る (3)．さらに，実演奏から抽出した代表値，あらかじめ入力済みのドラムの音楽演奏データおよび重み係数から音の大きさを示すパラメータの列を作成し (4)，これら 2 種類のパラメータ列をドラムの音楽演奏データに適用することによってドラムの音楽演奏データを調整する (5)．調整されたドラムの音楽演奏データにおける n 番目の音の大きさ，発生時刻をそれぞれ $C_n^{(l)}$，$C_n^{(t)}$ とする．

以下ではそれぞれのブロックでの処理について述べる．

(1) 代表値の抽出　利用者がキーボード実演奏を行う際，複数の音を同じ拍で演奏する（和音を演奏する）ことがある．システムは，利用者が時間窓サイズとして設定した大きさの時間幅に入っている複数の音を同じ拍の音とみなし，そこから代表値として 1 つの音を選択する．代表値の選択方法は多数存在するが，ここでは (i) 最も高い音 (top note) を選択するか，(ii) 最も大きく演奏された音 (max velocity note) を選択するかという 2 種類の方法から，利用者が決定することにした．

和音の構成音のすべてが弾かれるのは，もっとも音高が高い時が多いという理由から，もっとも音高が高い音を代表値として利用する [6]．さらに本研究では，最も大きく演奏された音は，実演奏時に発生するノイズである可能性が最も低いと考え，この音を代表値として利用するオプションも実装する．

(2) テンポの推定　利用者が実際に行ったキーボード演奏のテンポを推定する．ここで言うテンポとは，実演奏を行っている最中においては一定のものである．以下の処理では，16 分音符あたりのおおまかな時間 $t_{keyboard}$[ms] を用いる．これは利用者が入力した $T_{keyboard}$[bpm] から次のようにして変換する．

(一分間あたりの 4 分音符個数)
$$= T_{keyboard}$$
(4 分音符あたりの時間 [m])
$$= 1/T_{keyboard}$$
(16 分音符あたりの時間 [m])
$$= (4/16)(1/T_{keyboard})$$
(16 分音符あたりの時間 $t_{keyboard}$[ms])
$$= 60 \cdot 1000 \cdot (4/16)(1/T_{keyboard})$$
$$= 15000/T_{keyboard}$$

逆に 16 分音符あたりの時間 $t_{keyboard}$[ms] からテンポ $T_{keyboard}$[bpm] へは次のように変換できる．

$$T_{keyboard} = 15000/t_{keyboard}$$

実演奏における音の発生する時間間隔（以下ではオンセット間隔と呼ぶ）$A_{n+1}^{(t)} - A_n^{(t)}$ を $t_{keyboard}$ で割り，整数化した値はそれぞれのオンセット間隔が 16 分音符時間長の何倍に相当するかを示している．そこでオンセット間隔をこの値で割ることによって 16 分音符時間長を得ることができる．これにより，すべての音について 16 分音符時間長を求め，それらの平均値を求めることで，平均 16 分音符時間長 $\langle t \rangle$ を求める．ここから，テンポ $\langle T \rangle = 15000/\langle t \rangle$ を得

ることができる．以後の処理では $\langle t \rangle$ を用いる．以上をまとめると以下に示す式を得る．

$$\langle t \rangle = \left\langle \frac{A_{n+1}^{(t)} - A_n^{(t)}}{\left\lfloor \frac{A_{n+1}^{(t)} - A_n^{(t)}}{t_{key}} + 0.5 \right\rfloor} \right\rangle$$

ここで $\lfloor x \rfloor$ は x の整数部分を示すものとする．
$\lfloor x + 0.5 \rfloor$ は $x + 0.5$ の整数部分を示し，x を四捨五入して整数化したものである．

(3) 音の発生時刻の決定　利用者の実演奏においては音の発生が無い部分も含めて，一旦すべての 16 分音符について音の発生時刻を決定する．利用者がキーボードを用いて n 番目の音を演奏してから $n+1$ 番目の音を演奏するまでの時間が平均 16 分音符時間長 $\langle t \rangle$ の何倍に相当するかを求め，ここから $A_n^{(t)}$ と $A_{n+1}^{(t)}$ の間に補間する要素数 N を決定する．

$$N = \left\lfloor \frac{A_{n+1}^{(t)} - A_n^{(t)}}{\langle t \rangle} \right\rfloor - 1$$

この N を用いて補間を行う．ここでは，線形補間を用いる．次の要素が $A_n^{(t)}$ と $A_{n+1}^{(t)}$ の間に補間される．

$$A_n^{(t)} + k \frac{A_{n+1}^{(t)} - A_n^{(t)}}{N+1} \quad (k = 1, 2, \ldots, N)$$

すべての n について $A_n^{(t)}$ と $A_{n+1}^{(t)}$ の間を補間して得た全要素を新たに $A_m^{(t),(interpolated)}$ とする．
さらに，利用者が入力した重み係数 w を用いて，$A_m^{(t),(interpolated)}$ と $(m-1)\langle t \rangle$ との間で重み付き平均を求める．これは，利用者がキーボード演奏によって提供した表情情報が必要以上に誇張されたもので，ドラムに直接適用するには適さないものであった場合に調整を行うためのものである．

$$A_m^{(t),(w)} = \frac{w A_m^{(t),(interpolated)} + (1-w)(m-1)\langle t \rangle}{2}$$

以上によりすべての 16 分音符について，音の発生時刻を決定する．

(4) 音の大きさの決定　音の発生時刻の場合と同様に N を決定し，$A_n^{(l)}$ と $A_{n+1}^{(l)}$ の間に次の要素を補間する．

$$A_n^{(l)} + k \frac{A_{n+1}^{(l)} - A_n^{(l)}}{N+1} \quad (k = 1, 2, \ldots, N)$$

すべての n について補間を行うことで得た音の大きさを示すパラメータの列を新たに $A_m^{(l),(interpolated)}$ とする．

音の発生時刻の場合と同様に，次のようにしてすべての 16 分音符について音の大きさを示すパラメータの値を決定する．

$$A_m^{(l),(w)} = \frac{w A_m^{(l),(interpolated)} + (1-w)\langle B^{(l)} \rangle}{2}$$

ここで $\langle B^{(l)} \rangle$ は楽譜入力インタフェース等を用いてドラムの音楽演奏データを入力した際に設定した，ドラムの音の大きさの初期値である．

(4) 発生時刻，大きさの適用　利用者が入力した T_{drums} [bpm] から，楽譜入力インタフェース等を用いてあらかじめ入力したドラムの音楽演奏データにおける 16 分音符時間長 $t_{drums}(= 15000/T_{drums})$ [ms] を得ることができる．ドラムの音楽演奏データにおける n 番目の音は，すべての 16 分音符についてのパラメータ列 $A_m^{(t),(w)}$，$A_m^{(l),(w)}$ における $B_n^{(t)}/t_{drums}$ 番目の 16 分音符に対応する．したがって，$C_n^{(t)}$，$C_n^{(l)}$ の値を次のように定めることができる．

$$C_n^{(t)} = A_m^{(t),(w)}, C_n^{(l)} = A_m^{(l),(w)} \left(m = \frac{B_n^{(t)}}{t_{drums}} \right)$$

なお，重み係数は音の大きさと発生時刻について，個別の値を適用することができるようにする．さらにドラムセットに含まれる楽器を「シンバル類」「バスドラム，スネアドラム」「その他」の 3 つのカテゴリに分け，それぞれに個別の値を適用できるようにする．従って，重み係数として入力する値は 6 種類である．
これは「シンバル類の音の大きさや発生時刻に与える変化を，バスドラム，スネアドラム対して行うものに比べ誇張したものにする」あるいは，「音の大きさに与える変化を，音の発生時刻に与える変化に比べ誇張したものにする」ということが経験的に知られているためである．
さらに，奇数番目の 16 分音符について，発生時刻を付加的に数ミリ秒遅らせる機能も実装する．人間が実際にドラムを演奏する場合には，「奇数番目の 16 分音符は一般に左手で演奏され，わずかに遅れる傾向にある」ことも経験的に知られているためである．

2.5　実用上の制限

上述の処理手順は処理の単位として 16 分音符を用いているため，3 連符等の 16 分音符の整数倍でない音符，および 32 分音符等の 16 分音符より細かい音符は正しく認識できない．また，利用者が平均 16

分音符時間長に比べて非常に大きな時間的変化を加えようとすると正しく認識できない．これらの点については今後検討が必要である．

3 プロトタイプシステム：CAPADY

本章では，提案手法に基づき構築中のシステム CAPADY(a Composition Assistance by Producing Agogics and DYnamics) について述べる．

3.1 システムの概要

CAPADY は 2 つの部分から構成される．一つは Setting Window(図 3) であり，もう一つは CapadyPlayer Window(図 4) である．
利用者は Setting Window を用いて 2.4 節に示した，キーボードの実演奏とドラムの音楽演奏データ以外に必要となる数値を入力する．

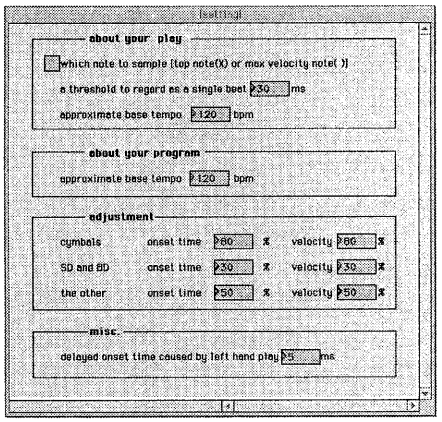

図 3: Setting Window

CapadyPlayer Window は調整されたドラムの音楽演奏データを再生する際に用いる．ここでは利用者が行った実演奏，調整の対象となる基のドラムの音楽演奏データ，および CAPADY によって調整されたドラムの音楽演奏データにそれぞれ対応するチェックボックスが用意されている．CapadyPlayer Window が再生状態になった時にチェックがついているものが再生される．複数のチェックボックスにチェックがついている場合は，チェックがついている音楽演奏データが同時に再生される．さらに利用者がドラムの音楽演奏データを比較検討するために，ファイルとして保存された音楽演奏データを同様にして再生できる一時読み込み領域を 3 か所設けた．

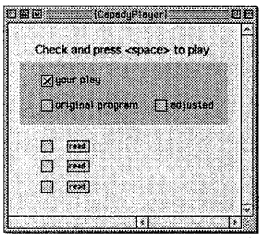

図 4: CapadyPlayer Window

3.2 システムの利用例

以下では，本研究の提案手法に基づくシステム CAPADY の作曲過程における利用についてシナリオを記述する．

[アクターの設定] ここでは以下の様にアクター (利用者) を設定する．

> タロウは，ほぼ毎日パソコンを利用して，友人とのメールやレポート作成，web ページ作成などを行っている．ジャズ・フュージョンを演奏するアマチュアバンドでキーボードを担当して 6 年目になる．これまでに MIDI シーケンサを利用して，彼が所属するバンドで演奏するための曲を約 10 曲作った．今，彼が所属するバンドで演奏するための新曲を作っているところである．楽譜入力インタフェースを用いて一般的な 16 ビートのドラムパターンを 4 小節入力し保存した．CAPADY を利用して，この 4 小節に表情付けを行おうと思った．

[目的 1] タロウはキーボードを実際に演奏して，あらかじめ楽譜入力インタフェース等を用いて入力したドラムの音楽演奏データを調整する．

> 彼は CAPADY を起動しメニューバーから import your program を選択して先ほど楽譜入力ツールで作成した 4 小節のドラムのデータを CAPADY に読み込んだ．次にメニューバーから CapadyPlayer Window(図 4) を開き your program にチェックを付けて読み込んだドラムのデータを再生した．
> 彼はメニューバーから record your play を選択して CAPADY を録音状態にした．そして先週の所属バンドの練習の際に行ったジャムセッションでメンバーから好評だったフレーズを思い出し，4 小節のフレーズをキーボードで演奏した．CapadyPlayer Window を your play だけにチェックが付いている状態にして

録音された彼の演奏を再生した．彼は演奏の再生を聞き終るとメニューバーから adjust を選択した．
CapadyPlayer Window を your play と adjusted にチェックがついた状態にし先ほど録音した彼の演奏と調整されたドラムのデータを同時に再生した．CapadyPlayer Window の your play のチェックをはずし adjusted のみにチェックがついた状態にして調整されたドラムのデータだけを再生した．彼は「ちょっとやりすぎかな」と思った．

[目的 2] ドラムの音楽演奏データを再調整するために重み係数を変更する．
メニューバーから Setting を選択し，彼は Setting Window(図 3) を開いた．そして Setting Window における adjustment の SD and BD onset time 欄の数値を 30 から 5 に変更した後，再びメニューバーから adjust を選択した．CapadyPlayer Window を your play と adjusted にチェックがついている状態にして録音された彼の演奏と調整されたドラムのデータを同時に再生した．CapadyPlayer Window の your play のチェックをはずし，adjusted のみにチェックがついた状態にして調整されたドラムのデータを再生した．

[目的 3] 楽譜入力インタフェース等を利用して，ドラムの音楽演奏データを変更する．
メニューバーから save as... を選択し，調整されたドラムのデータをファイルとして保存した．彼は楽譜入力ツールを起動して先ほど楽譜入力ツールで作成したドラムのデータを開くとバスドラム，スネアドラムの位置を変更して保存した．
楽譜入力ツールで変更を加えたドラムのデータをメニューバーの your program を選択して読み込み，メニューバーから adjust を選択した．CapadyPlayer Window を adjusted のみにチェックがついた状態にして調整されたドラムのデータを再生した．
CapadyPlayer Window に 3 つある read ボタンの 1 つをクリックして先ほど CAPADY で保存したドラムのデータのファイルを指定した．読み込んだ read ボタンのチェックボックスにチェックをつけ再生した．

[目的 4] キーボードを演奏し直すことで，ドラムの音楽演奏データを再調整する．
メニューバーから record your play を選択し CAPADY を録音状態にしてから，先ほど彼が演奏したフレーズに少し変更を加えて演奏し直した．次に CapadyPlayer Window の your play にチェックがついている状態にして，新たに録音した彼の演奏を聞いた．その後，彼はメニューバーから adjust を選択した．
CapadyPlayer Window を your play と adjusted にチェックがついている状態にして新たに録音した彼の演奏と調整されたドラムを同時に聞いた．さらに CapadyPlayer Window の adjusted のみにチェックがついた状態にして調整されたドラムを聞いた．

以上のような操作を繰り返し行うことによりタロウは納得の行くドラムの音楽演奏データを作成することができる．また，演奏内容を任意に変更しながら繰り返しキーボードを実際に演奏することで，キーボードについても (a) 解候補の生成，および (b) 評価が行われる．

システムの問題点を発見する方法の一つとして失敗シナリオを作成することがある．以下では CAPADY の利用における失敗シナリオの例を示す．

[目的 5] 過去に行った調整と同じように現在対象としているドラムの音楽演奏データを調整する．
彼は新たにドラムのデータを楽譜入力ツールで作成した．再び CAPADY を起動し，新しいドラムパターンに表情付けを行うことにした．昨日 CAPADY を利用したときの調整と同じように新しいドラムパターンも調整したいと思ったが，どのような実演奏でどのような Setting で調整したのか思い出せなかった．

[目的 6] 様々な調整を行った多数の解候補からもっとも満足の行く解を選択する．
実演奏を何度もやり直したり Setting を何度も変更したりすることによって何通りもの調整を行い，多数のドラムのデータをファイルとして保存した．その中にはある程度納得の行く調整ができたものもあったが，より良いと思えるものを作成するために，データを保存しておいて，さらに作業を続けた．作業を終了しようと思った時，彼は今までで一番満足の行くものがどれであったか思い出せなかった．

4 考察

以下では，上述の失敗シナリオ例への対応策を述べる．また実演奏を行う楽器としてキーボード，調整の対象としてドラム演奏データという本研究で例としてとりあげた以外の楽器を用いる場合の課題について考察する．

実演奏と Setting を対にした保存 CAPADY を利用して音楽演奏データに表情付けを行う際，ファイルとして保存することができるのは CAPADY が調整した音楽演奏データのみである．一旦作業を中断し，システムを終了すると過去に行った作業のコンテキストがすべて失われてしまう．実演奏と Setting を対にして保存する機能を実装することは，この問題の対応策の一つである．

解候補の構造化　比較対象となる解候補が多くなれば，解候補を管理し，それぞれの解候補の間にある関係を利用者に提示する手段が重要となる．構築中のシステムではファイルとして保存された音楽演奏データを読み込み，調整中のデータと比較検討するために一時的に音楽演奏データを読み込む領域を3か所設けたにすぎない．網谷らの研究では，(a) 解候補を2次元空間に配置し，(b) 解候補に言葉を対応付ける，という2つの手法で解候補を管理している[3]．多数ある解候補を管理することの重要性は上述の実演奏と Setting を対にして保存する場合にも当てはまる．

例に挙げた以外の楽器を用いる場合の課題　本研究においては実演奏に用いる楽器および調整の対象となる楽器としてそれぞれキーボード，ドラムを例にとり，それぞれの楽器が持つ性質を利用して議論の一部を単純化した．以下では実演奏に用いる楽器や調整の対象となる楽器として他の楽器を用いる場合について考える．

実演奏に MIDI キーボードのように MIDI 信号を出力できる楽器ではなく，音響信号のみを生成する楽器を利用する場合には実演奏の音響信号から音の大きさや発生時刻といった表情情報を抽出することが課題となる．

2.3節で述べたように，本研究では音の長さについては考慮しなかったが，調整の対象となる楽器をドラムに限定しないのであれば音の長さについても考慮する必要がある．さらに管楽器では音が発生した後に音の大きさが変化するような場合もある．表情情報として扱うパラメータは楽器によって異なる．実演奏に使用する楽器が表情情報として扱うパラメータに，調整の対象となる楽器が表情情報として扱うパラメータが含まれている場合には2章で述べた処理手順を拡張することで対応できる．そうでない場合については今後検討が必要である．

5　おわりに

本研究では，計算機上の音楽演奏データに対する表情付けの一手法として人間と計算機の協調による表情付けを提案した．システムの利用者は楽器を実際に演奏することによって表情情報をシステムに入力する．システムは入力された表情情報を利用して，表情情報の入力に用いられた楽器とは異なる楽器の，あらかじめ入力済みの音楽演奏データを調整する．本手法においてはシステムの利用者は表情情報を明確に把握することなく，システムに表情情報を入力することができる．提案手法に基づいて構築中のシステム CAPADY が作曲過程の中でどのように利用されるかについてシナリオを用いて述べた．今後は4章で述べた問題点および対応策に考慮し，さらに実用的なシステムを構築したいと考えている．

謝辞

本研究を行うにあたって有益な御助言を頂いた北陸先端科学技術大学院大学の西本一志氏，奈良先端科学技術大学院大学の大平雅雄氏に感謝いたします．

参考文献

[1] D. A. Norman, “Things That Make Us Smart,” Addison-Wesley Publishing Company, 1993.

[2] J. A. Sloboda, “The Musical Mind,” Oxford university press, 1985.

[3] 網谷重紀，堀浩一，“作曲者のメンタルスペースの外在化による作曲支援環境の研究”，情報処理学会音楽情報科学研究会研究報告，MUS37-5，2000.

[4] 坂本圭司，堀内靖雄，市川熹，“計算機との合奏データによる人間の演奏モデルの推定”，情報処理学会音楽情報科学研究会研究報告，MUS37-9，2000.

[5] 石毛大吾，堀内靖雄，市川熹，“相互作用を考慮した人間の協調演奏モデルの推定”，情報処理学会音楽情報科学研究会研究報告，MUS37-9，2000.

[6] 野池賢二，乾伸雄，野瀬隆，小谷善行，“演奏情報と楽譜情報の対からの演奏表情規則の獲得とその応用”，情報処理学会音楽情報科学研究会研究報告，MUS26-16，1998.

[7] 鈴木泰山，徳永健伸，田中 穂積，“Kagurame Phase-I 事例ベースの演奏表情生成システム-”，情報処理学会音楽情報科学研究会研究報告，MUS-24-11，1998.

[8] 中川渉，高嶋章雄，山本恭裕，蔵川圭，中小路久美代，“実演奏における演奏表情情報の抽出と適用”，情報処理学会ヒューマンインタフェース研究会研究報告，HI89-10，2000.