ヒューマンインタフェース 91－3
音 楽 情 報 科 学 37－3
（2000．10．12）

バービーブン: 音符レベルでユーザ意図を把握して編曲を行う事例ベースシステム

平田 圭二
NTT コミュニケーション科学基礎研究所
hirata@brl.ntt.co.jp

青柳 龍也
津田塾大学
aoyagi@tsuda.ac.jp

本稿では，現在開発中の音楽システム「バービーブン」の設計，動作機構，実装等について述べる．バービーブンには，あらかじめいくつかの編曲事例が蓄えられており，単純なモチーフが与えられると，それに類似した編曲事例を利用して編曲を行う．これらモチーフや編曲事例はすべて多声旋律として表現されており（音符レベル），バービーブンでは和音名（コード記号）は一切用いない．ユーザ意図は，入力のモチーフ，編曲事例にあらかじめ付与されたグルーピング構造と時間構造として表現される．また，そのグルーピング構造と時間構造に関する情報を付与するためのエディタを作成した．

Ba-Bi-Bun: Case-Based Reasoning Musical Arrangement System Understanding a User's Intention at Note Level

Keiji Hirata
NTT Communication Science Laboratories

Tatsuya Aoyagi
Tsuda College

This paper presents the design principle, the system organization and the implementation of a musical system "Ba-Bi-Bun" that we are developing. Ba-Bi-Bun stores several arrangement cases beforehand, and when it is given a simple motif, Ba-Bi-Bun arrange the motif by using a similar arrangement case. The motif and the arrangement case are all represented in polyphonic melodies (i.e. note level), and no chord symbols are used. A user's intention is realized as a special data structure representing grouping and temporal information. An editor dedicated to handling the special data structure is developed.

1 はじめに

我々は，これまで演繹オブジェクト指向データベース (Deductive Object-Oriented Database, DOOD[1][6]) の枠組を用いた音楽知識表現と事例ベース推論 [5] を採用した一連の音楽システムを研究開発して来た．これら音楽システムは，一般にハービー君族と呼ばれる (図 1)．本稿では，音楽システム「バービーブン」について述べる．

まず，ポップスやジャズのピアニストがソロピアノを弾く場合を考える．課題の楽曲は通常，単純な旋律 (melody) と，和声 (harmony) を指定するコード進行によって与えられる．ピアニストはそれらを元にフェイク (fake) や即興演奏を行う (本稿では編曲と総称する)．バービーブンの機能は，このピアニストのように課題楽曲をより複雑な楽曲に編曲することである．楽曲のジャンルは，ソロピアノであれば特に限定しない．バービーブンの特徴は，(1) 和声を指定するコード進行の記述に和音名 (コード記号) を一切使わない点と，(2) あるピアニストに単純な旋律と和声を与えた時，そのピアニストがどのように編曲したかという事例を元に，事例ベース推論を行う点である．(1) に関して，入力の旋律と和声及び出力もすべて音符レベルで表現され，システム内部でも音符レベルで計算が行われる．(2) に関して，事例ベース推論を用いることで，形容詞による指示やパラメータ空間におけるベクトル量を経由せずとも，ユー

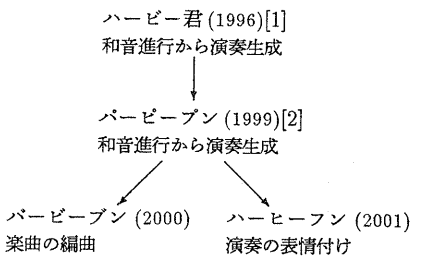

図 1: ハービー君族の系譜

[1]DOOD という用語はもともと国際会議や研究分野の名称として用いられていたが，本稿では知識表現手法の名称として用いる．

ザは「このような演奏が欲しい」という指示を出すことができる．

本稿の構成は次の通りである．まず第2章では，旋律や和声を表現するために用いた知識表現手法 (DOOD) について概説する．第3章では，DOOD をどのように応用して旋律や和声を表現したか，その表現手法はどのような条件を満たしているか，及びその特徴について述べる．第4章では，バービーブンの中心部である編曲アルゴリズムの設計や動作について述べる．第5章では，現在作成中のプロトタイプシステムについて述べる．最後に第6章では，まとめと今後の課題について述べる．

2 DOOD

DOOD では，対象をオブジェクトとその属性の集合から成るオブジェクト項として表現する．本稿では，オブジェクト項を $o(\cdots, l = v, \cdots)$ と記述する．ここで o は基本オブジェクト項，$l = v$ は属性，l は属性ラベル，v は属性値を表わす．

包摂関係 ($\sqsubseteq$) とは「情報量が多いオブジェクト $\sqsubseteq$ 情報量が少ないオブジェクト」という意味である．あるいは「具体的なオブジェクト $\sqsubseteq$ 抽象的なオブジェクト」，「特殊 $\sqsubseteq$ 一般」を意味する．オブジェクト項が表す集合の意味を考えて，集合の包含を表す記号 ($\subseteq$) と向きを揃えるために上記のような記法を採用している．

オブジェクト項間の包摂関係は以下の演繹規則によって定義される．今 $o_1 = p(\cdots, l_m = v, \cdots)$, $o_2 = q(\cdots, l_n = w, \cdots)$ とすると，o_1, o_2 間の包摂関係は，

$$o_1 \sqsubseteq o_2 \leftarrow p \sqsubseteq q \wedge \forall n \exists l_m \ (l_m = l_n \wedge v \sqsubseteq w)$$

と定義される．つまり基本項 p, q を比較し，o_2 の全ての属性 l_n について o_1 のそれが全て具体的ならば，o_1 の方が具体的 (あるいは o_2 の方が抽象的) という意味である．属性の少ないオブジェクト項は包摂関係に関してより抽象的である．

属性には固有と非固有の2種類がある．固有属性とはオブジェクトの識別に使用される属性であり，非固有属性はそうでない属性である．記法上は，$o(\cdots, l = v, \cdots)/(\cdots, m = w, \cdots)$ のように，'/' の左側に固有属性，右側に非固有属性を書く．例えば $o(a = 1)/(b = 2)$ と $o(a = 1)/(c = 3)$ は同一のオブジェクトである．

もし o_1, o_2 がオブジェクト項を要素に持つ集合の場合，o_1, o_2 間の包摂関係として様々な定義が考えられる．以下に代表的な2つを演繹規則として記述する．

$$o_1 \sqsubseteq_H o_2 \leftarrow \forall s \in o_1 \exists t \in o_2 \ s \sqsubseteq t$$
$$o_1 \sqsubseteq_S o_2 \leftarrow \forall t \in o_2 \exists s \in o_1 \ s \sqsubseteq t$$

$\sqsubseteq_H$ はいわゆる集合の Hoare 順序に，$\sqsubseteq_S$ は Smyth 順序に等しい．例えば，$\{b, d\} \sqsubseteq_H \{a, b, c, d\}$, $\{a, b, c, d\} \sqsubseteq_S \{b, d\}$ のようになる．Hoare 順序は集合要素間に選言の意味がある場合に，Smyth 順序は連言の意味がある場合に用いられる．

ドット記法の定義: $p(\cdots, l = v, \cdots)$ というオブジェクト項がある時，v の値に注目して $p.l = v$ と表記する．□

最小上界 (least upper bound, lub) の定義: オブジェクト項 x, y が与えられた時，x と y の最小上界とは $min(\{z | x \sqsubseteq z \wedge y \sqsubseteq z\})$ であり，$lub(a, b)$ と書く．□

lub の直観的な意味は，ある二つのオブジェクトに共通で抽象的なオブジェクトの内，最も具体的なものであり，積集合を計算するイメージである．最大下界 (greatest lower bound, glb) も同様に定義できる．DOOD では lub を join，glb を meet とも呼ぶ．上で定義したオブジェクト項の領域は $\sqsubseteq$ に関して完備束を構成する．この lub や glb が，編曲アルゴリズム (第4章) を構成する時の基本演算子となる．

3 多声旋律の表現

ここでは，一般のピアノ譜程度の譜面を考える．旋律や和音を構成する複数の音が楽曲の一部あるいは全体を構成する．本稿ではそのような楽曲の一部あるいは全体を多声旋律と呼ぶ．図7 は多声旋律の例であり，この曲 (Nefertiti) は，和音名を使わずに多声旋律で和声が指定されているような楽曲である．

本章では，DOOD に基づいて多声旋律を表現するためのオブジェクト項を導入する．オブジェクト項の抽象化，具体化が，多声旋律の簡単化，複雑化にうまく対応するようにオブジェクト項を設計しなければならない．

3.1 グルーピング構造の表現

楽曲のグルーピング構造を表現するために，GTTM の Time-Span Reduction Tree (TS 簡約木) を利用する．TS 簡約木は，ボトムアップに楽曲構造的に重要な音 (和音) と隣接する重要でない音 (和音) をまとめて (グルーピングして) 行く様子を表現している (図2 に例を示す)．ここでは簡単のため TS 簡約木は二進木とし，重要な音を primary，そうでない音

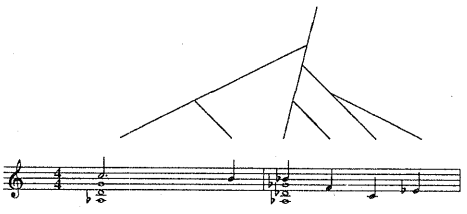

図 2: グルーピング構造の例

を secondary と呼ぶ．グルーピングされた primary と secondary はある 1 つの音 (和音) で代表され，それを head と呼ぶ．primary/secondary と head の関係には primary, fusion, transformational の 3 通りがある．TS 簡約木の末端のノード (葉) はある時刻に生起した音 (和音) を表し，primary/secondary を持たない．TS 簡約木では primary の生起時刻が 1 段上のグルーピング構造の生起時刻となる．ある TS 簡約木から primary や secondary の枝を削除すると，より簡単な TS 簡約木が得られる．

3.2 時間構造の表現

まず，楽曲における時間構造の抽象化，具体化の関係について考察する．通常，定量的な記述を抽象化すると定性的な記述が得られる．多声旋律中に現われる各音の生起時刻は定量的に決められるが，これを抽象化した定性的な記述とは「順序」であると考えられる．これは，厳密な時刻が分からずとも，先か後かの順番だけ記述されているという状況である．さらに順序は縮退によって条件の緩い抽象的で簡単な順序になる．例えば，「イベント a が先でイベント b が後」という順序を縮退すると，「イベント a はイベント b より先あるいは同時」という順序が得られる．以上の考察より，時間構造のモデルは時刻に関する定量的な情報と順序に関する情報を分離して記述できなければならないことが分かる．

本稿で提案する時間構造の例を図 3 に示す．図

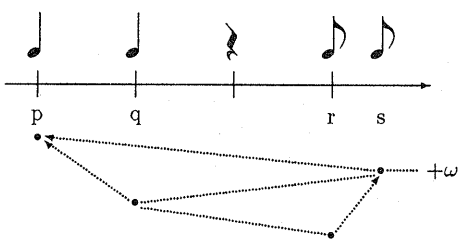

図 3: 時間構造の例

中 p, q, r, s はイベントである．まずイベント p の生起時刻を基準と仮定する．次にイベント s に着目すると，s は p と $+\omega$ の間に生起し (順序に関する情報)，p から 3 拍半の時刻に生起する (時刻に関する定量的な情報)．同様に q は p と s の間に生起し p から 1 拍め，r は q と s の間に生起し s から半拍めの場所に位置している．さらに，r の時刻が縮退する先は楽曲的により重要な拍であり (この場合は s)，それが ⇢ で示されている．

3.3 各オブジェクトの属性と属性値

バービーブンで実際に用いたオブジェクト項の定義を図 4 に示す．図中，objclass($name_1$: objclass_1, ...) の objclass はオブジェクトクラス名，$name_1$ は固有属性名，objclass_1 は属性値のオブジェクトクラス名あるいはデータ型という意味である．オブジェクトを構成する際に必須の属性と必須でない属性があり，必須でない属性名の左肩には & を付けた．小文

event(*head*: chord, *at*: temp | `Integer`, *key*: `C..B`,
  &*primary*: event, &*secondary*: event)

chord(*notes*: {note}, *duration*: `Integer`)

note(*noteNumber*: `Integer`)

temp(*after*: temp | `Integer` | $-\omega$,
  before: temp | `Integer` | $+\omega$,
  reference: `After` | `Before`,
  &*difference*: `Integer`)

図 4: オブジェクト項の定義

字で始まる語はオブジェクト名を表し，大文字で始まる語はそれ以外のデータやデータ型を表す．
`Integer` は整数の集合を表し，`C..B` はドからシまでの 11 の音名からなる集合を表し，`After`，`Before` は予約語であり，$+\omega$ は正の無限大の時刻，$-\omega$ は負の無限大の時刻の意味である．$x \mid y$ は x または y という意味である．$\{x\}$ は x から成る集合を表す．

オブジェクト項 chord の *notes* 属性は和音を構成する個々の音を表し，*duration* 属性は和音が鳴っている時間的長さを表す．*notes* 属性の値が singleton 集合の時，それは単音と等価である．note オブジェクトの *noteNumber* 属性では音高が MIDI ノートナンバによって表現されている．

この event オブジェクトは，時間的に重複するグルーピング構造も記述できるという点において，GTTM の TS 簡約木の拡張になっている．

3.4 event オブジェクトに関する包摂関係

event オブジェクトは，3.1 節と 3.2 節で述べたグルーピング構造と時間構造のモデルを実現している．つまり，モデルにおける抽象化，具体化の操作がオブジェクト項の包摂関係に対応付けられている．グルーピング構造に関する抽象化は，楽曲構造上，重要でない音を削除することであり，これは *primary* 属性や *secondary* 属性を削除することに対応する．時間構造に関する抽象化は 2 通りあり，1 つは定量的な時刻の情報を削除することであり，これは *difference* 属性を削除することに対応する．もう 1 つは，拍節構造的により重要な隣接する拍に時刻が縮退することであり，このために以下のような新しい包摂関係を導入する．

ある時刻を表現するオブジェクト項 t ($t \sqsubseteq \mathrm{temp}$) がある時，$t.reference$ によって指定される拍の方が拍節構造的により重要な拍である．従って，t が $t.(t.reference)$ に縮退すると時間構造はより抽象的 (簡略) になることが分かる．つまり，

$$t \sqsubseteq t.(t.reference)$$

である．ここで，$t.reference$ の指定はグルーピング構造と整合しなくてはならないことに注意されたい．図 3 の例では，時刻 t_r は $\mathrm{temp}(after = t_q, before = t_s, reference = \mathtt{Before}, difference = 8$ 分音符) というオブジェクト項で記述できるので，$t_r \sqsubseteq t_s$ となる．

4 編曲アルゴリズム

音楽家が M_c という簡単な楽曲を編曲して P_c を創作することを模倣し，未知の簡単な楽曲 M から新しい編曲 P を合成するような編曲アルゴリズムを設計する．

4.1 問題の形式化

本節では問題を形式化する．事例に基づく編曲の原理と編曲アルゴリズム $\mathcal{A}$ の仕様を図 5 に示す．

編曲を行う時の音楽家の内省に基づき，以下のステップで M から P を合成する．

S1. M に類似した M_c を持つ事例を検索する．
S2. M と M_c の共通部分を M_a とする．
S3. M_c から M_a への変換 $\mathcal{F}$ を模倣して，P_c を P_a に変換する ($\mathcal{F}'$)．
S4. M から M_a への変換 $\mathcal{G}$ を模倣して，さらに逆転させて P_a から P を作る ($\mathcal{G}'^{-1}$)．

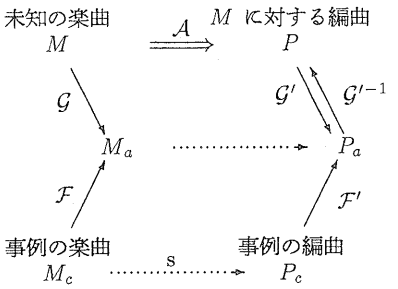

図 5: 事例に基づく編曲の形式化

S1. の最類似事例の検索は，パーピープン [2] で提案した方法 (相対的類似度) を援用する．図中，s の点線は，音楽家が M_c という簡単な譜面を見た時に P_c という演奏を行ったという対応関係を表す．

編曲アルゴリズムの設計に際し以下の仮定を置く：

A1. M と M_c は似ている
A2. $\mathcal{F}$ と $\mathcal{F}'$ は似ている．$\mathcal{G}$ と $\mathcal{G}'$ も似ている．

ここで，**A1.** は **S1.** で保証される．**A2.** は音楽家の内省より得られる．M, M_c, M_a, P, P_c, P_a は DOOD のオブジェクト項として表現されるので，これらオブジェクト項間には包摂関係が成り立つ．

4.2 編曲アルゴリズムの設計

まず，$\mathcal{F}$ を実現する具体的な関数について考える．$\mathcal{F}$ を特徴付ける関係は，

$$\mathcal{F}(M_c) = M_a \quad \text{(F1)}$$
$$\mathcal{F}(M) = M \quad \text{(F2)}$$

である．$\mathcal{F}$ には様々な関数が考えられるが，M と M_c の共通部分 M_a を計算するには最小上界を用いるのが自然なので，

$$\mathcal{F}(X) = lub(X, M)$$

という定義が考えられる．P_a は $P_a = \mathcal{F}'(P_c)$ として求められるが，**A2.** より

$$P_a = \mathcal{F}(P_c) = lub(P_c, M) \quad (1)$$

を得る．

次に，関数 $\mathcal{G}$ を特徴付ける関係は

$$\mathcal{G}(M) = M_a \quad \text{(G1)}$$
$$\mathcal{G}(M_c) = M_c \quad \text{(G2)}$$

なので，同様に

$$\mathcal{G}(X) = lub(X, M_c)$$

とすることができる．ここで $\mathcal{G}$ を特徴付ける関係 G1, G2 の重要度について考える．**A1.** から，G1 において M を M_c で置き換えた場合 1 と，G2 において M_c を M で置き換えた場合 2 を比較する．場

合 1 からは，$\mathcal{G}(M_c) = M_a$ かつ $\mathcal{G}(M_c) = M_c$ より $M_a = M_c$ が得られる．一方，場合 2 からは，$\mathcal{G}(M) = M_a$ かつ $\mathcal{G}(M) = M$ より $M_a = M$ が得られるが，G1 も考慮すると $\mathcal{G}$ は恒等写像になり，$\mathcal{G}$ の特徴が消失する．よって，G2 の方がより重要な関係であることが分かる．

逆写像 $\mathcal{G}^{-1}$ を特徴付ける関係は，G1, G2 から，$\mathcal{G}^{-1}(M_a) = M$ 及び $\mathcal{G}^{-1}(M_c) = M_c$ となる．上の G1 と G2 の重要度に関する考察と **A1.** から，前者の関係を $\mathcal{G}^{-1}(M_a) = M_c$ と置き換えて，$\mathcal{G}^{-1}$ を実現する関数を考える．単純なものとして，

$$\mathcal{G}^{-1}(X) = glb(X, M_c)$$

が考えられる．ゆえに **A2.** より，

$$P = \mathcal{G}^{-1}(P_a) = glb(P_a, M_c) \qquad (2)$$

である．

以上の考察をまとめると，M から P を合成する編曲アルゴリズム $\mathcal{A}$ は，式 1，2 より

$$\mathcal{A}(X) = glb(lub(P_c, X), M_c) \qquad (3)$$

と表すことができる．

関数 $\mathcal{A}$ に注釈を加える．$\mathcal{A}$ はオブジェクト項の作る完備束上の連続関数である．この編曲アルゴリズムは，和音名 (コード記号) を全く使わずに音符レベルで計算を行う．M_c より抽象的な楽曲 M に対しては $(M_c \sqsubseteq M)$，$\mathcal{A}(M) = M_c$ となる．本アルゴリズムはソフトウェア発展の原理 [3, 4] の応用である．

5 プロトタイプシステム

5.1 事例としての event オブジェクト

バービーブンにおける事例は，和声 (H)，課題楽曲あるいは旋律パート (M)，それに対する編曲あるいは演奏例 (P) の組から構成される．H は event オブジェクトとして表現され，M と P の組は H を表現する event オブジェクトの非固有属性として表現される (図 6)．

```
event(head: chord, at: t | Integer, key: C..B,
      &primary: event, &secondary: event)
     /(plays:{play})

play(melody: event,
     &perform: {event})
```

図 6: 非固有属性 plays を持つ event オブジェクト項の定義

H は事例楽曲の和音進行に関するグルーピング構造や時間構造を表す．和声の上方は，固有属性 ('/' の左辺) を用いて記述されているので event オブジェクトの識別子として機能する．*plays* は非固有属性であり，固有属性で識別された和声の雰囲気や文脈における編曲を表している．1 つの和声に対して複数の課題楽曲とその編曲が考えられるので，*plays* 属性値は集合となっている．*plays* は非固有属性なので，システム動作中にその属性値が変化してもオブジェクトの同一性が保たれる．play オブジェクトの *melody* 属性で M を記述し，*perform* 属性で P を記述する．1 つの課題楽曲に対して複数の編曲が考えられるので，*perform* 属性値は集合となっている．また，H, M, P 全てが event オブジェクトによって表現される．

H を表現する全 event オブジェクトが必ず *plays* 属性を持つとは限らないが，事例である event オブジェクトは *plays* 属性を持つ．図 7 の例を用いて説明する．図の上方には和声 H の TS 簡約木を描いた．e_1 は C_2 を primary な部分木として，C_1 を secondary な部分木として持つ event オブジェクトを表している．同様に，e_2 は C_3 を primary な部分木として，e_1 を secondary な部分木として持つような event オブジェクトを表している．

例えば e_1 を事例とする場合を考える．この事例には先頭 2 小節分の H, M, P が含まれる．それぞれを表現する event オブジェクトを e_H, e_M, e_P とすると，事例 e_1 は $e_H/(plays = \{play(melody = e_M, perform = \{e_P\})\})$ と書ける．この時，e_H の *primary* と *secondary* の属性値として C_1 と C_2 を表現する event オブジェクト (それぞれ e_{C_1}, e_{C_2} とする) が出現している．しかし，e_{C_1}, e_{C_2} は *plays* 属性を持つ必要がない．つまり，事例を構成するトップの e_H オブジェクトのみ *plays* 属性を持つ．

event オブジェクトは，その定義から明らかなように再帰的な構造をしている．従って，事例の粒度は単和音から楽曲全体まで変化し得る．つまり，e_1 以外にも e_2, e_3 を事例とすることもできるし，逆に C_1, C_2 の個々の和音を事例とすることもできる．

事例の粒度は推論を行う範囲 (拍数，小節数) を規定する．事例の粒度が小さいと 1 回の推論で生成する編曲の範囲は小さくなる．この時，類似した事例を多く検索することができるが，大域的な曲の流れを考慮した編曲は難しくなる．一方，粒度が大きいと 1 回の推論で生成する編曲の範囲は大きくなり，粒度が小さい場合と逆の特徴を示す．プロトタイプシステムでは，簡単のため，1 つの和音でカバーされた範囲の楽曲部分と編曲部分を 1 つの事例としている．

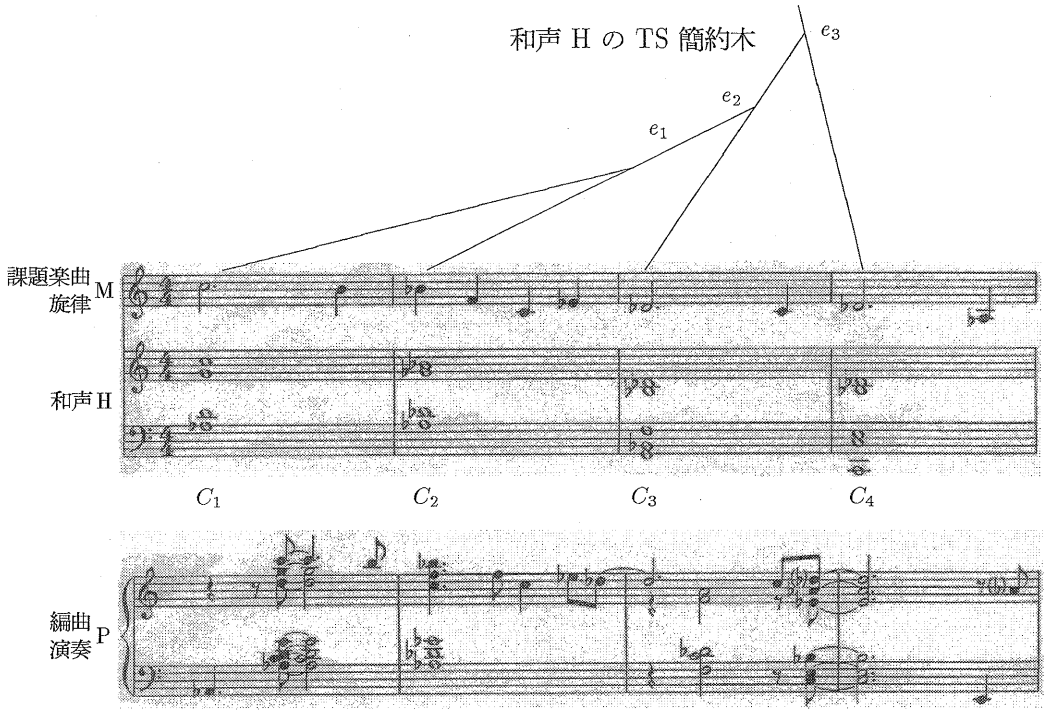

図 7: event オブジェクトによる事例の表現

5.2 事例ベース推論

バービーブンの事例は，一般に $e_H/(plays = \{play(melody = e_M, perform = \{e_P\})\})$ のように書ける．ここで e_H は楽曲部分の和声を，e_M は対応する旋律を，e_P は対応する編曲 (演奏例) を表している．

バービーブンのプロトタイプシステムに未知曲が与えられた時，その和音列を $C_1, C_2, C_3 \cdots$ とすると，以下のように処理が進む

1. 事例ベース中から C_i に最も類似した e_H を持つ事例を選ぶ．この時，バービーブン [2] で提案された相対的類似度を用いる．
2. さらに，未知曲の旋律 m に最も類似した e_M を持つ事例に絞り込む．
3. 選ばれた事例を用いて，C_i のカバーする範囲の編曲 $\mathcal{A}(m)$ を計算する．

ステップ 1. において先に和声の類似性で事例を選択するのは，編曲という応用の性質を考慮し，和声の類似性の方が旋律の類似性よりも重要であると考えたからである．

本プロトタイプシステムには，実世界的な状況でも頑健に動作するような工夫が幾つか加えられている．1 つは，音どうしの類似性をモデル化するための新しいデータ型の導入である．音楽では音高の近い音どうしやオクターブ離れた音どうしは類似していると言われている．この類似性を持つような特殊なデータ型を設計しプロトタイプシステムに実装した．また，編曲アルゴリズムにおいて，仮定 **A1.**, **A2.** があまり成立しないような状況が起こり得る．そのような状況で本編曲アルゴリズムを適用しても，trivial な編曲しか得られない．そこで，オブジェクト項の解釈を拡張して lub 演算を行うような改良を加えた．詳細は別稿に譲る．

5.3 グルーピングエディタ

プロトタイプシステムの一部として，グルーピングエディタを開発している (図 8)．ユーザはグルーピングエディタを用いて TS 簡約木を組み上げる．この時，*head*, *primary*, *secondary* の関係，時間構造における *reference*, *difference*, 各 event オブジェクトの *key*, どのレベルの event オブジェクトを事例とするかの情報を指定する．図中，ユーザは内省により α の枝が β の枝に従属していると考え，α の

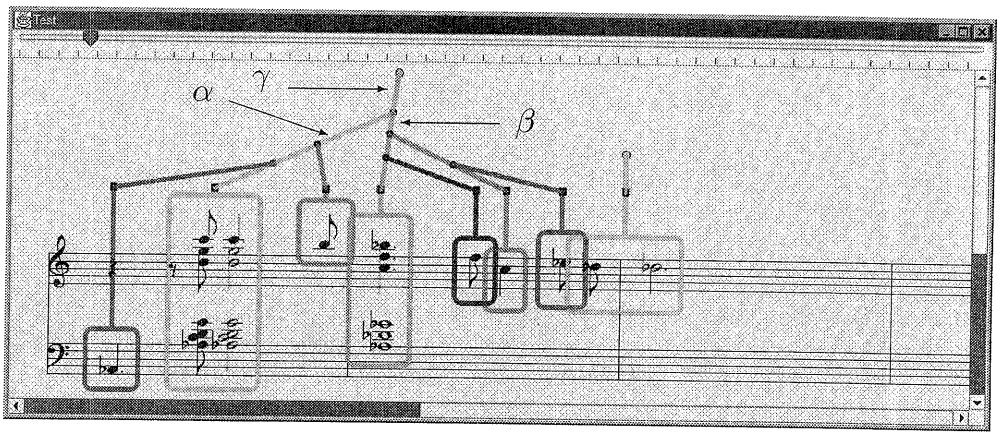

図 8: 開発中のグルーピングエディタのウィンドウ

枝を β に貼付けている．この時，β が *primary* に，α が *secondary* に，γ が *head* に対応する．このようにして，ユーザは，その楽曲の解釈を TS 簡約木として容易に表現することが可能となった．また，将来入出力を XML 経由で行うことと，移植性を考え，本エディタは Java で実装を行っている．

6 まとめにかえて

まず，今後のシステム改良に関する課題について述べる．大きく，知識表現手法の改良と編曲アルゴリズムの改良に分けられる．

知識表現手法の改良については，さらに表現力を上げることと，より簡潔な表現手法を開発することが必要である．表現力を上げるためには，元となっている音楽知識のモデルを拡張する方法があり，GTTM の延長的還元 (Prolongational Reduction) を採り入れたり，Narmour の暗意 - 実現 (Implication-Realization) モデルを採り入れることが考えられる．簡潔な表現手法に関して，現在の event オブジェクトと temp オブジェクトでは一部重複して表現されている情報があるので，その部分をできるだけ少なくすることが考えられる．

編曲アルゴリズムに関して，現在，次のような改良を考えている．1 つの未知曲に対して複数の事例を同時に用いて編曲を行うアルゴリズム，編曲結果に含まれる和音の協和性，非協和性の制御，未知曲と事例の楽曲が同じになってしまったような場合への対処などである．

最後に，本システムをどのように評価すべきかということも大きな課題である．少なくとも次の 4 通りの観点から評価を行う必要があると考えている．(1) 観察に基づく一般ユーザに対するシステムの全体評価，(2) ユーザ意図の把握と反映に関する評価 (3) 開発当事者がシステムを改良するための評価，(4) 知識表現手法や事例ベース推論等の個々の要素技術に対する評価．評価に関しては，稿を改めて報告したいと思う．

参考文献

[1] 後藤真孝, 平田 圭二, ハービー君: 演繹オブジェクト指向に基づいてジャズらしいコードにリハーモナイズするシステム, 情報処理学会 音楽情報科学研究会 研究報告 96-MUS-16, pp.33–38 (1996).

[2] 平田圭二, 青柳龍也, パーピープン: 誰でもどこでもインタラクティブに使える知的ジャズ和音生成システム, 情報処理学会 音楽情報科学研究会 研究報告 99-MUS-31, pp.7–12 (1999).

[3] 片山卓也, 進化発展するソフトウェアの原理, 情報処理学会誌 Vol.40, No.2, pp.153–156 (1999).

[4] 片山卓也, ソフトウェア発展の原理とメカニズム, bit Vol.32, No.3, pp.18-22, 共立出版社 (2000).

[5] Janet Kolodner, Case-Based Reasoning, Morgan Kaufmann (1993).

[6] 横田一正, 演繹オブジェクト指向データベースについて, コンピュータソフトウェア Vol.9, No.4, pp.3–18 (1992).