Vol.2010-MUS-86 No.22
2010/7/30

タイムスパン木の単一化可能性

東条　敏[†1]　平田圭二[†2]　浜中雅俊[†3]

GTTM/ATTA によって生成されるタイムスパン木は編曲やモーフィングにおいて有用である．簡約 (reduction) というのは本来一つの曲に対して定義されている構造の簡素化の過程であるが，本稿ではこの概念を敷衍し，複数の木の間に簡約と同様な順序関係を仮定する．次に木の間で meet と join が定義されよって束 (lattice) を構成するように定式化を行う．これにより複数な木から新たな楽曲が創成できるための条件を考察する．

Unifiability in Time-span Trees

Satoshi Tojo,[†1] Keiji Hirata[†2]
and Masatoshi Hamanaka[†3]

Time-span tree, generated by GTTM/ATTA, is useful for arranging and morphing music pieces. The original notion of reduction is to retrieve a simplified skeleton from a piece; we generalize the notion and define a partial order between trees. Thereafter, we introduce 'meet' and 'join' operations to build a lattice, where we consider practical conditions that these operations produce meaningful pieces.

1. はじめに

タイムスパン木とは Generative Theory of Tonal Music (GTTM)[1] のグルーピングと拍節構造から見た音の重要度を階層的に表現する木である．図 1 は J. S. バッハ，マタイ受難曲のコラール “O Haupt, voll Blut und Wunden” から生成されるタイムスパン木である．図中，Level a,b,c,d はその階層的な簡約 (reduction) を表す.

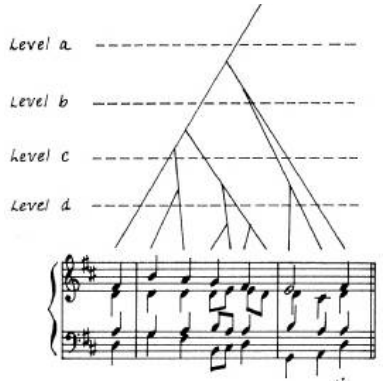

図 1　マタイのコラールによるタイムスパン木の簡約の階層

筆者らはこれまでタイムスパン木を自動生成し[2] それを用いて曲のモーフィングや外挿などを提案してきた[3),4)]．モーフィングをする際には，例えば二つのメロディから共通部分を探すような操作が必要であるが，このようすは図 2 に示す[5)]．

本稿では GTTM の本来の理論からこの木の階層構造と，さらには木の間の演算を考察して木の集まりが束を構成するような体系を考える[6)–8)]．これにより，木の間の演算の数学的基盤を与え，かつ演算結果が音楽として有意味であるような条件を考察する．

2. 一つの楽曲の木による束

2.1 水平簡約による全順序束

ある一つの楽曲のタイムスパン木 T においてその簡約の過程に現れる木，すなわち木を水平な線で切った上部の木構造の集合を $S_T^{\uparrow}$ とする．S_T は結合的 (connected) な強い半順序 (strict partial order) すなわち非反射的 (irreflexive) かつ推移的 (transitive) であることより全順序 (total order) 集合となる．すなわち $t_1, t_2, \cdots \in S_T^{\uparrow}$ に対して順序 '$\sqsubset$' が定義できる．ここで '$\sqcap$'(meet) および '$\sqcup$'(join) は trivial であり，$t_1 \sqsubset t_2$ ならば $t_1 \sqcap t_2 = t_1$, $t_1 \sqcup t_2 = t_2$ である．

†1 北陸先端科学技術大学院大学情報科学研究科
JAIST
†2 NTT コミュニケーション科学基礎研究所
NTT Communication Science Laboratories
†3 筑波大学システム情報工学研究科
Tsukuba University

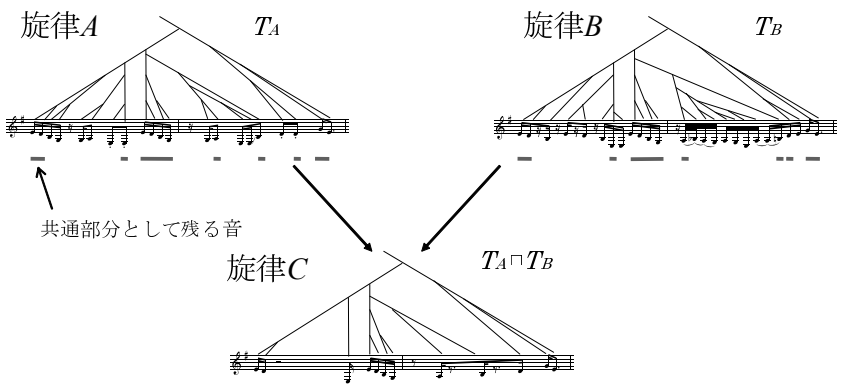

図 2 旋律 A と B の共通部分

ここでタイムスパン木に対する水平簡約はもとのグルーピング・拍節構造を反映するように行うものであり，枝のつく位置（高さ）に恣意性がないものとする．

2.2 部分木簡約による半順序束

一つの楽曲に対する**部分木簡約**とは最終的な解析木のうち，任意の枝からその枝を含んで下位の部分木を除去したものである．水平簡約と異なるのは，(i) ある部分だけ詳細な部分木を残して他の部分を除去できること（簡約の不均一性），および (ii) 下位の部分木を残したままそれとは disjoint な上位の枝が除かれる可能性があること（簡約の反階層性）である．図 3 はこのような簡約の例である．

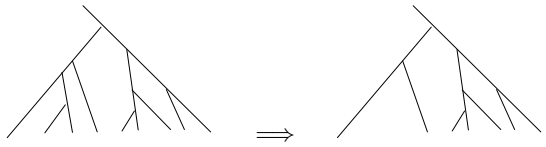

図 3 非水平な簡約

ある一つの楽曲の木に対する簡約全体の集合を S_T とすると，明らかに $S_T^{\uparrow} \subseteq S_T$ であり，S_T は包摂関係 '$\sqsubseteq$' に関して反射的 (reflexive)，遷移的 (transitive)，反対称的 (anti-symmetric) な半順序集合をなす．簡約による半順序集合は non-trivial な '$\sqcap$'(meet) および '$\sqcup$' を定義する．$t_1, t_2 \in S_T$ ならば $t_1 \sqcap t_2 \in S_T$ および $t_1 \sqcup t_2 \in S_T$ が一意に定まる．したがって S_T は**束**(lattice) をなす．以後この概念を敷衍し，異なる曲の間で木の間の演算を考察するが，その際にも擬似的な一つの楽曲の木から構成される束に還元されることを目標とする．

3. 素性構造と単一化

タイムスパン木の持つ枝の接合関係だけではもとの音楽を表現できない．木はあくまでもとの楽曲と結びついて，すなわち楽曲 M と木 T とのペア $\langle M, T\rangle$ でその解析結果として意味を持つ．前節までに論じた一つの曲に対する簡約においては常に簡約された楽曲 M' と枝を刈られた木 T' が対応付けられるため，$\langle M', T'\rangle$ は齟齬なく構成できる．

さて木にさらに枝を加えるという操作を考える．これはある楽曲に音を「足す」操作に相当し，例えば与えられた楽曲に対し，各音の装飾に相当するような操作に相当する．ここで足すべき音がもとの楽曲に入り込む余地があるかどうか，またそれによって既存のピッチ・イベント (pitch event) を書き替える必要があるかどうかなどの論点を生ずる．後節では二つの木から meet や join，さらにはモーフィングという操作まで定義できるかどうかを論じるが，例えば join で言えば $\langle M_1 \sqcup M_2, T_1 \sqcup T_2\rangle$ に相当する楽曲 $M_1 \sqcup M_2$ をどのように構成するかが問題となる．

楽曲に結び付かない木のみに対する操作は以下のように困難である．まず，異なる二つの木 T_1, T_2 は $T_1 \sqcap T_2$(meet) および $T_1 \sqcup T_2$(join) を一意に決定できるとき，一つの束に埋め込み可能 (embeddable) であるとしたい．図 4 上二行は直観的に考えた二つの木の meet と join のようすである． しかしながら，どの枝も葉に相当するピッチ・イベントを同定しないとこの操作は不毛である．すなわちトポロジカルに木の形状が同じであっても枝の指すピッチ・イベントが異なるかどうか判定しない限り有意味な操作には結びつかない．図 4 の最下行の絵はトポロジーの同一性と曲の同一性に関する曖昧性の例である．

以下，このような問題に対処するために楽曲と木を切り離した考え方を改め，両方の情報が統合された**素性構造**(feature structure)[9),11)] という考え方を導入する．素性構造とは素性 (属性) とその値のペアを縦に複数並べて角括弧でくくって表現したものである．値には他の素性構造が再帰的に入り込むことが可能である．

異なる二つの素性構造を一つの素性構造に統合する操作を**単一化** (unification) と呼び，属

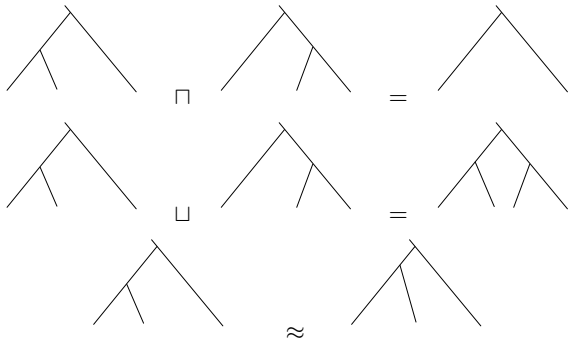

図 4　meet と join の定義可能性

性の値に矛盾のないときに限り可能である．いま単一化を θ と表せば，

$$\theta(\begin{bmatrix} f_1 & v_1 \\ f_2 & v_2 \end{bmatrix}, \begin{bmatrix} f_2 & v_2 \\ f_3 & v_3 \end{bmatrix}) = \begin{bmatrix} f_1 & v_1 \\ f_2 & v_2 \\ f_3 & v_3 \end{bmatrix}$$

と単一化可能 (unifiable) であるが，

$$\theta(\begin{bmatrix} f_1 & v_1 \\ f_2 & v_2 \end{bmatrix}, \begin{bmatrix} f_1 & v_1 \\ f_2 & v_4 \end{bmatrix}) = \bot$$

では f_2 の値が異なるため単一化の結果は論理矛盾 ($\bot$) と記されている．

まず単一のピッチ・イベントに関する素性構造は MIDI にならって以下のように定義することができる．

$$\begin{bmatrix} onset & v_1 \\ offset & v_2 \\ pitch & v_3 \end{bmatrix}$$

また楽譜にならって onset を楽譜上での位置情報として何小節め (bar) の何拍め (meter) という相対位置 (position) を，offset に替えて音価 (duration) を用いて図 5 左のように表示することができる．[⋆1]　ここで '~' に先行された文字列はその構造全体の型(sort) を表す．

複数のピッチ・イベントから構成される部分木は，ヘッド(head) という概念を用いて図

⋆1 さらに単音の素性構造にはアクセント（スタッカート，テヌートなど）などに関する素性を追加することも可能である．

$$\begin{bmatrix} \tilde{}event & \\ pos & \begin{bmatrix} bar & n_1 \\ meter & n_2 \end{bmatrix} \\ pitch & v_2 \\ duration & v_3 \\ beat & \{on, off\} \end{bmatrix} \qquad \begin{bmatrix} \tilde{}tree & \\ head & \{\boxed{\text{i}}, \boxed{\text{j}}\} \\ dtrs & \begin{bmatrix} left & \begin{bmatrix} \tilde{}tree & \\ head & \boxed{\text{i}}\begin{bmatrix}\tilde{}event\end{bmatrix} \end{bmatrix} \\ right & \begin{bmatrix} \tilde{}tree & \\ head & \boxed{\text{j}}\begin{bmatrix}\tilde{}event\end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

図 5　ピッチ・イベント（左）と木（右）の素性構造

5 右のように表現できる．構造全体はその構造に前置された四角囲みのインデックスを用いて参照することができる．図 5 右では *event* 型のものが素性 *head* の値になることが想定されており，構造全体の *head* 素性は再帰的に下位の左右の木構造の *head* から選択される．すなわち集合表現 { } はそのうちの値の一つを選択する．素性 *dtrs* はヘッドに対する子 (daughters) を意味する．

一つのピッチ・イベントは *dtrs* 素性を欠いた *tree* 型として以下のように表現できる．

$$\begin{bmatrix} \tilde{}tree & \\ head & \boxed{\text{i}}\begin{bmatrix}\tilde{}event\end{bmatrix} \end{bmatrix} \tag{1}$$

以上の素性構造を用いて前節に導入した部分木簡約を以下のように定式化することができる．まず入れ子になっている属性の値を言及するには属性名を '|' で連結し '⟨ 構造名 ⟩|f_n|f_m' のように記する．

前節の木の部分木簡約を素性構造のことばで再定義すると，**左部分木簡約**とは素性構造 S における '$S|dtrs|left|dtrs$' 素性を消去する操作となる．図 6 では '$\Longrightarrow$' の *left* 素性にある部分木が切り落とされ，(1) 式に基づき直接ピッチ・イベントに到達する木になっている．**右部分木簡約**は *right* 素性にある部分木を切り落とす操作として同様に定義できる．また木の水平簡約は，多くの場合左部分木簡約と右部分木簡約を同時に行う操作と一致する．

4. 異なる二曲間の階層と演算

前節で議論したとおり，一般に二つの木の間の演算 meet, join は，その演算結果に相当する楽曲が矛盾なく構成できるときにのみ有意味である．このとき meet, join はルートを同じくする二つの素性構造の単一化の結果であり，meet は二つの素性構造の無矛盾な共通部分と考えることができる．したがって二つの木に相当する楽曲は，

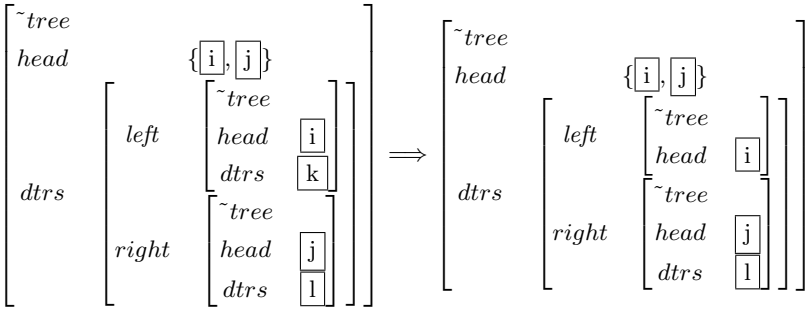

図 6　左部分木簡約

- 同じ長さ，すなわち同じ小節数を持つものとする．
- 同じ拍節構造を持つものとする．
- 単一化の対象となる個々のピッチ・イベントについてはピッチ・イベントに関するすべての素性について一致する．

ときに単一化可能であるとする．

ただし meet, join のための操作は音楽上の意味を解釈し直す必要がある．例えば休符は本来であれば「音がないこと」をポジティヴに述べているものであるが，ここで定義した木に相当する素性構造はピッチ・イベントに限定しているために，休符のところに音を置くことは整合的な操作となる．さらには，ある音が持続している状態に他の音が同時に鳴るような単一化を禁止/許可するためには各ピッチ・イベントの位置情報 (*pos* 素性) と音価 (*duration* 素性) を計算して整合性を取る操作を考慮しなければならない．本稿では以下の**モノフォニー条件**のみを考え，その数値的実現は実装に委ねる．

> “二曲に相当する素性構造がモノフォニック (monophonic) に単一化可能であるときは，単一化の結果において一つのピッチ・イベントの存在と持続時間が他のピッチ・イベントと時間的に排他的であることを要請する.”

二つの楽曲に相当する素性構造がモノフォニー条件を充たして単一化可能であるとは，その二つの木構造をあるタイムスパン木を部分簡約した結果として含むような，ある複雑な曲を仮想することができるということである．このとき，前節で木について定義された簡約に関する束の構造と同様に，二つの素性構造は**束に埋め込み可能**であるという．

5. 緩めれらた条件での二曲間の演算

さて前節までに述べた二つの木の間の演算は，現実的なアプリケーションに対してはあまり興味を引く結果を得られない．楽曲の編曲の問題に還元するためには少なくとも次のような可能性を含ませることが自然に要請される．

(i)　二つの楽曲の長さが（多少なりとも）異なっていても単一化可能であること，あるいはピッチ・イベントの位置に柔軟性を求めること
(ii)　ピッチ・イベントの音価の一致に柔軟性を求めること
(iii)　ピッチ・イベントの音高の一致に柔軟性を求めること
(iv)　ピッチ・イベントの拍の一致に柔軟性を求めること

などなどである (図 7).

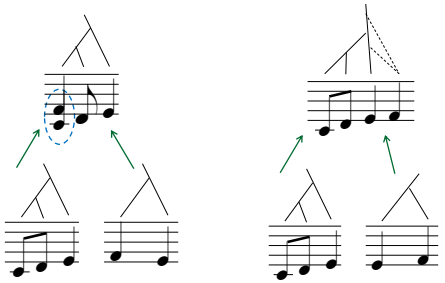

図 7　ピッチ・イベントの変更（左）と一致範囲（右）を緩める単一化

各項目については次のように演算条件を緩和することが考えられる．

(i)　木のトップに集まった左枝，右枝を基準の二音とみなし，その二つのピッチ・イベントが同じであれば演算可能であるとする．
(ii)　例えば 2 分音符と 4 分音符＋ 4 分休符を同一視できるとする．
(iii)　例えばオクターヴの差異は同一視できるとする．さらに発展的には異なる pitch の音も集合的に表現した上で，どちらか一方に単一化であるとする．
(iv)　拍の位置にあるピッチ・イベントが一致していれば拍子が異なっていてもそれらは同一であるとする．ただし音価は拍子に応じて補正されるべきである．またもしピッチ・イベントの情報に上拍・下拍の素性があるならばその値は保持されるべきである．

これにより 2 拍子の曲と 4 拍子の曲，あるいは 2 拍子の曲と 6/8 拍子の曲とを単一化できることが考えられる．

これらの緩和は，例えば音価・音高の変更に関してはそれがグルーピングの根拠になっていた可能性があるので，緩和することは木構造の信頼性を揺るがす問題になる．しかしそれでもなお，緩和によって新たな楽曲を創成できることの意義を考えることとする．

5.1 再帰的な単一化アルゴリズム

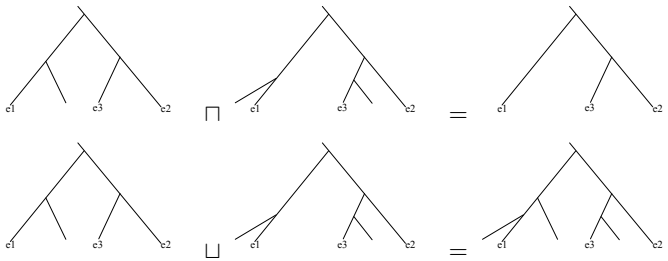

図 8 トップからの二音位置合わせの演算

図 8 はこの指針においては左右二つの枝がぶつかるところで両枝のピッチ・イベント e_1, e_2 を楽曲上で同定し，

- 左右の主従関係（「人」と「入」の関係）が一致し，
- e_1, e_2 が両木で同一のピッチ・イベントであるとき，

両方の木はこの**枝の接合箇所で単一化可能**であると考える．次に左右それぞれの枝について次の高さにある枝の接合場所において同様な操作を再帰的に繰り返す．図 8 では，両方の木で e_3 が e_2 上 e_1 の次に見つかるとき，e_2，e_3 について同じ検査を行っている．すべての枝の接合箇所において無矛盾に単一化可能であるとき**木は単一化可能**であるとする．この再帰的なアルゴリズムを以下に形式化する．

まず単一化を行う関数を *unify* とし，終端条件を次の三つの場合に分けて考える．

- 二つのピッチ・イベントを入力とするとき，素性の完全な一致を要請する．

$$unify(\boxed{\mathrm{i}}[\tilde{}event], \boxed{\mathrm{i}}[\tilde{}event]) = \boxed{\mathrm{i}}[\tilde{}event] \tag{2}$$

- 一方が木，他方がピッチ・イベントであるとき，単一化を可能にするためには木のヘッドがピッチ・イベントと一致することを要請する．単一化の結果は式 (1) に基づいて入力の木そのものとなる．

$$unify(\begin{bmatrix}\tilde{}tree \\ head & \boxed{\mathrm{i}} \\ dtrs & \boxed{\mathrm{j}}\end{bmatrix}, \begin{bmatrix}\tilde{}event \\ head & \boxed{\mathrm{i}}\end{bmatrix}) = \begin{bmatrix}\tilde{}tree \\ head & \boxed{\mathrm{i}} \\ dtrs & \boxed{\mathrm{j}}\end{bmatrix} \tag{3}$$

- 両方が木でありヘッドが一致しているが，一方の木には右側に枝，他方の木には左側に枝が残る場合（cf. 図 7 右），単一化においてどちらの枝をより高い位置に接合するかで恣意性を生ずる．すなわち枝の接合位置に高低のオプションを残して二種類の単一化を得る．

$$\begin{aligned} &unify(\boxed{\mathrm{l}}\begin{bmatrix}\tilde{}tree \\ head & \boxed{\mathrm{i}} \\ dtrs & \begin{bmatrix}left & \boxed{\mathrm{i}} \\ right & \boxed{\mathrm{j}}\end{bmatrix}\end{bmatrix}, \boxed{\mathrm{m}}\begin{bmatrix}\tilde{}tree \\ head & \boxed{\mathrm{i}} \\ dtrs & \begin{bmatrix}left & \boxed{\mathrm{k}} \\ right & \boxed{\mathrm{i}}\end{bmatrix}\end{bmatrix}) \\ &= \left\{\begin{bmatrix}\tilde{}tree \\ head & \boxed{\mathrm{i}} \\ dtrs & \begin{bmatrix}left & \boxed{\mathrm{k}} \\ right & \boxed{\mathrm{l}}\end{bmatrix}\end{bmatrix}, \begin{bmatrix}\tilde{}tree \\ head & \boxed{\mathrm{i}} \\ dtrs & \begin{bmatrix}left & \boxed{\mathrm{m}} \\ right & \boxed{\mathrm{j}}\end{bmatrix}\end{bmatrix}\right\} \end{aligned} \tag{4}$$

(4) の最初の解は部分木 $\boxed{\mathrm{j}}$ を右に低く，部分木 $\boxed{\mathrm{k}}$ を左に高く接合した結果，二番目の解は部分木 $\boxed{\mathrm{j}}$ を右に高く，部分木 $\boxed{\mathrm{k}}$ を左に低く接合した結果である．関数 *unify* は第一引数と第二引数の順序が逆であっても同様であることに注意．

このアルゴリズムを図 9 に示す．このアルゴリズムは二つの素性構造を入力とするものであるが，上式 (2), (3), (4) を終端条件とする．

5.2 素性をマスクしたピッチ・イベントの単一化

式 (2) においてはピッチ・イベントの素性の完全な一致が求められていた．ここで図 7 左の議論に基づいて，単一のピッチ・イベントに関して音高や音価，拍の制限を緩める関数

$$unify^{-}_{(pitch)},\ unify^{-}_{(duration)},\ unify^{-}_{(beat)},\ \cdots$$

を (2) に代えて用いることを考える．右肩の '−' は緩和された関数であることを示す．また suffix として素性構造のどの素性を無視した単一化関数であるかが示されている．これにより，それぞれ音価，音高，拍などの情報をマスクして一致を求めることができる．特に一致に関しては二つのピッチ・イベントがどちらもが理論的に（単一化の手続き上）可能である場合は，(i) パラメータによるスイッチ (ii) 可能な候補のセット (iii) ユーザ選択などさまざまなオプションが考えられよう．

$unify(S_1, S_2)$
% **input**: 素性構造 S_1, S_2 でルートの位置を揃える.
begin
if $S_1|head = S_2|head$ **then**
% ヘッドのピッチ・イベントが一致すれば,
if $S_1,\ S_2 \in event$ **then**
return $unify(S_1, S_2)$ by (2);
else if $S_1 \in event$ & $S_2 \in tree$ **then**
return $unify(S_1, S_2)$ by (3);
else if $(S_1|dtrs|left|head = S_2|dtrs|left|head$ &
$S_1|dtrs|right|head = S_2|dtrs|right|head)$ **then**
% 子の左右の部分木のヘッドが一致すれば,
return $\begin{bmatrix} \tilde{}tree & \\ head & S_1|head \\ dtrs & \begin{bmatrix} left & unify(S_1|dtrs|left, S_2|dtrs|left) \\ right & unify(S_1|dtrs|right, S_2|dtrs|right) \end{bmatrix} \end{bmatrix}$;
% 子の左右の枝において再帰的に操作を継続.
else % $S_1,\ S_2$ は木であるが左右別に枝を残すとき,
return $unify(S_1, S_2)$ by (4);
else quit & fail;
end

図 **9** 単一化のアルゴリズム

6. おわりに

与えられた二つの素性構造が単一化可能であるか，すなわち，その二曲の join に相当する楽曲が存在するかどうかを決定する問題は (i) グラフ構造の単一化問題と (ii) 各ピッチ・イベントの独立性（位置・音価を侵害し合わない）を保証する数値的な計算によって解くことができる．しかしながら，緩和された条件での単一化と可能な解空間の広がりは容易に計算量を同定できるものではない．さらには二つの楽曲から単一化可能な曲の解空間において，もっとも妥当な結果を選ぶアルゴリズムにも考察が必要である．このように計算論的な困難は伴うが，その応用可能性は広い．

まず筆者らは楽曲のモーフィングアルゴリズム[3]，外挿アルゴリズム[4] などを提示してきたが，単一化条件を緩和することによってさらにさまざまな条件下でのモーフィング・外挿が可能になる．また単一化の操作が純粋に素性構造の一致問題と素性のマスキングによることから，純粋に計算機科学の応用問題として装飾・編曲のアルゴリズムが提案できる．このことは音楽情報処理の手法のさらなる発展を期待できるものである．

さらにまた素性構造の単一化手法によるメディア操作は音楽だけにとどまるものではない．さまざまなメディアを素性構造によって表現し，その素性を緩和する単一化アルゴリズムによって計算機によるメディアの取り扱い手法を大きく開拓する可能性がある．

参 考 文 献

1) Fred Lerdahl and Ray Jackendoff, *A Generative Theory of Tonal Music*, The MIT Press (1983).
2) M. Hamanaka, K. Hirata, and S. Tojo. Implementing 'A Generative Theory of Tonal Music,' Journal of New Music Research, vol.35, no.4, pp. 249-277 (2007).
3) Masatoshi Hamanaka, Keiji Hirata and Satoshi Tojo, Melody Morphing Method Based on GTTM, In *Proc. of ICMC 2008*, pp.155–158.
4) Masatoshi Hamanaka, Keiji Hirata and Satoshi Tojo, Melody Extrapolation in GTTM Approach, In *Proc. of ICMC 2009*, pp.89–92.
5) 平田圭二, 東条敏, 浜中雅俊．旋律モーフィングアルゴリズムの形式的検証，第 85 回音楽情報処理研究会, 2010.
6) 平田圭二, 東条敏, 相対擬補元を用いたメディアデザイン操作の形式化について, 第 19 回 人工知能学会 全国大会, 2B3-08 (2005).
7) 平田圭二, 東条敏, 楽曲構造束とその上の演算系, 第 20 回 人工知能学会 全国大会, 1D2-4 (2006).
8) Keiji Hirata and Satoshi Tojo, Lattice for Musical Structures and Its Arithmetics, *LNAI 4384* (Selected Papers from JSAI 2006, T. Washio et al. (Eds)), Springer-Verlag, pp.54–64 (2007).
9) Bob Carpenter, *The Logic of Typed Feature Structure*, Cambridge University Press (1992).
10) 東条敏, 計算論的音楽理論の応用, 情報処理学会誌, 道しるべ : 音楽と言語の構造認知. 計算の視点から音楽の構造を眺めてみると (3), Vol.49, No.9 (2008).
11) 東条 敏，素性から組み上げられる文の論理構造．人工知能学会誌 vol.22 (2007).