Vol.2010-MUS-88 No.3
2010/12/4

タイムスパン木の *join* と *meet* について

平田圭二[†1]　東条敏[†2]　浜中雅俊[†3]

単一化不可能な 2 つの旋律から 1 つの旋律を合成する問題を考える．そのような 2 つの旋律 (タイムスパン木) S_1，S_2 に対して $S_1 \sqsubseteq join(S_1, S_2) \wedge S_2 \sqsubseteq join(S_1, S_2)$ $(\star)$ という条件が成立するような *join* 演算を提案する．当該問題への対処法として，我々はこれまで単一化に幾つかの拡張を加えることを試みてきた．しかしこのアプローチでは上の条件 $(\star)$ を成立させることが難しい．そこで単一化アルゴリズムを拡張するのではなく，タイムスパン木の表現法を工夫するアプローチを検討し，タイムスパンを表現する構造化音長という型を導入した．これより，*join* 演算の適用範囲を広げその有用性を高めることができた．一方 *meet* 演算に関しても同様の議論を展開することができる．

Join and *meet* operators for timespan trees

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

We discuss the problem of constructing a melody from the two that are not unifiable. We propose the *join*operation such that for the two melodies (timespan trees) S_1, S_2 that are not unifiable, condition $S_1 \sqsubseteq join(S_1, S_2) \wedge S_2 \sqsubseteq join(S_1, S_2)$ $(\star)$ holds. Up to now, we have made attempts of extending the unification algorithm against the problem. However, it is difficult for this approach to satisfy the condition $(\star)$. Then, we do not extend the unification algorithm, but make an investigation into the representation method of a timespan tree, and we introduce a new type 'structural length' for representing the timespan of a note. As a result, the domain of the *join*operation is extended, and the utility improved. In parallel, we can make the same improvements to the *meet*operation.

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

1. はじめに

我々は文献 1) にて，タイムスパン木の素性構造による表現法と 2 つのタイムスパン木を単一化するアルゴリズムを与えたが，単一化の条件として，対応する 2 つの旋律をオーバレイした旋律がモノフォニ条件を満たす必要があった (必要条件)．ここで，2 つの旋律 M_1，M_2 のオーバレイとは，旋律をピッチ・イベントの集合と考えると，$M_1 \cup M_2$ である．つまり，M_1 と M_2 の楽譜を単純に重ね合わせて 1 つの楽譜としたものと考えればよい．モノフォニ条件とは，ある旋律に関して任意の時刻において 2 つ以上のピッチ・イベントが存在しないことである．つまり，1 つのピッチ・イベントが他のピッチ・イベントと時間的に排他であることを意味する．以下簡単のため「2 つの旋律をオーバレイした旋律がモノフォニ条件を満たす」ことを「2 つの旋律がモノフォニ条件を満たす」と記す．

モノフォニ条件を満たす 2 つの旋律は，直感的に互いに似通っていると考えることができよう．ここに 1 つの元となる旋律 C があり，C に異なる簡約操作列を適用して 2 つの旋律 A, B を作り出すとする．この時，A, B がモノフォニ条件を満たすまで，適切な簡約操作を A, B 各々に適用したとする．この時，$A \sqsubseteq C$ かつ $B \sqsubseteq C$ が成立している．つまり A, B はともに C の持つ特徴や雰囲気を引き継いだ旋律であり，従ってお互いに類似している．従って，現実的な状況では，任意の 2 つのタイムスパン木が与えられた時に，それらがモノフォニ条件を満たす場合はそれほど多くない．

ここで単一化と *join* 演算の違いについて簡単に触れる．まず，変数を含む 2 つの項 S_1，S_2 に対して，$S_1\theta = S_2\theta$ を満たすような変数代入 θ が存在する時単一化可能といい，もしそのような θ が存在しない時は単一化不可能という．最小の θ による具体化を $S_1\theta = S_2\theta = unify(S_1, S_2)$ と書く．この時，項間の包摂関係 $S_1 \sqsubseteq unify(S_1, S_2) \wedge S_2 \sqsubseteq unify(S_1, S_2)$ が成立する．これに対して，*join* の場合は，単一化不可能な場合も含む S_1, S_2 に対して，アトム間の包摂関係や構造を持つ項間の包摂関係に基づいて $join(S_1, S_2)$ が常に存在し同様に項間の包摂関係が成立する．

実応用の一例である旋律モーフィング[2] はそのアルゴリズム中で T_C と T_D の単一化を計算しているが，T_C と T_D が似通っている場合にのみ単一化可能でありモーフィングの解が得られる．しかし，似通っている旋律のモーフィングしか計算できないと，モーフィングの適用範囲は狭く有用性は低い．

そこで文献 1) で提案した単一化アルゴリズムでは，“単一化” の適用範囲を広げるために以下 3 点の拡張を加えた: (1) 単一化不可能な素性だけをマスクして単一化を続行する，

(2) 音高や音価の一致に柔軟性を持たせる，(3) 旋律 S_1, S_2 のタイムスパン木の形が単一化不可能な場合は S_1 に S_2 を埋めこむかその逆の操作を行う．この拡張により，単一化不可能な 2 つの旋律から可聴 (audible) な旋律を合成できるようになった．しかし特に上記 (3) の場合が生じると，拡張された単一化アルゴリズム中の “$unify(S_1, S_2)$ by (4)” の行が実行されるので，$S_1 \sqsubseteq unify(S_1, S_2) \wedge S_2 \sqsubseteq unify(S_1, S_2)$ が成立しなくなる．

本稿では，単一化不可能な 2 つのタイムスパン木に対して $S_1 \sqsubseteq join(S_1, S_2) \wedge S_2 \sqsubseteq join(S_1, S_2)$ が成立するような *join* 演算 (‘$\sqcup$’) を提案する．上述の旋律モーフィングアルゴリズムにおいて，単一化の代わりに *join* 演算を用いれば，より自由に旋律 C, D を選ぶことができ，その適用範囲は広くなり有用性は高くなるだろう．ただし，拡張された単一化によって具体化された旋律 (タイムスパン木) は可聴であったのに対し，本方式の *join* の結果は常に可聴であるとは限らない．*meet* 演算についても *join* と同様な提案と議論を展開する．

2. 準　　備

2.1 素 性 構 造

定義 (素性構造): 素性構造とは，素性 (feature) とその値 (value) 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出現可能である[6),7)]．

素性構造を見やすく表示するために，直接関係のない素性は省略する．また，連続する素性 F_i の表示は ‘$|'$ を挟んで ‘$F_1 \mid F_2 \mid F_3\ '$ のように表記し，間に挟まれた素性を省略する際は ‘$F_1 \mid\mid F_n\ '$，最初の部分を省略する際は ‘$\mid\mid F_n\ '$，最後の部分を省略する際は ‘$F_n \mid\mid'$ のように表記する．$[\![f\ v]\!]$ という素性構造 σ がある時，$\sigma.f$ と表記して v の値を指示する．

旋律 a，b に対するタイムスパン木を σ_a，σ_b のように表記する．一般に，1 つの旋律には複数のタイムスパン木が対応するが，本稿では 1 つの旋律に 1 つのタイムスパン木のみが対応すると仮定し，旋律 a とタイムスパン木 σ_a を同一視する．曖昧にならない限りタイムスパン木に対しても a，b のように表記する場合がある．

定義 (包摂関係): 素性構造 σ_1，σ_2 が与えられた時，以下 2 つの包摂関係を演繹規則によって与える．

$$\sigma_1 \sqsubseteq_H \sigma_2 \ \leftarrow\ \forall(f\ v_1) \in \sigma_1\ \exists(f\ v_2) \in \sigma_2\ \ v_1 \sqsubseteq v_2$$

$$\sigma_1 \sqsubseteq_S \sigma_2 \ \leftarrow\ \forall(f\ v_2) \in \sigma_2\ \exists(f\ v_1) \in \sigma_1\ \ v_1 \sqsubseteq v_2$$

$\sqsubseteq_H$ はいわゆる集合の Hoare 順序に，$\sqsubseteq_S$ は Smyth 順序を表す．Hoare 順序は集合要素間に連言の意味がある場合に，Smyth 順序は選言の意味がある場合に用いられる．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．素性構造の素性・値ペアは連言の意味なので，本稿では Hoare 順序を採用する．

2.2 *join* 演算と *meet* 演算

定義 (*join* と *meet*): 値 a, b が与えられた時，a と b の *join* (結び) とは，条件 $a \sqsubseteq c \wedge b \sqsubseteq c$ を満たす c の中で最小のものを指し，$a \sqcup b$ と書く．a と b の *meet* (交わり) とは，条件 $c \sqsubseteq a \wedge c \sqsubseteq b$ を満たす c の中で最大のものを指し，$a \sqcap b$ と書く．

定義より $a \sqcup c = c$，$a \sqcap c = c$ (吸収則) が成立する．

3. タイムスパン木の表現

本章では，単一化不可能な 2 つの旋律から 1 つの旋律を合成するため，単一化アルゴリズムの方ではなく，タイムスパン木の表現法を工夫する手法を提案する．

3.1 タイムスパン木の持つ意味

タイムスパン木は，音楽学的には次のように説明されることが多い．旋律に含まれるピッチ・イベントどうしの音高と時間の近さによるグルーピングと拍節の強さによるグルーピングに基づいて，隣接する音との相対的な構造的重要度を決めながら構成された二分木である[3)]．この二分木を構成する際にピッチ・イベントが支配する時間幅 (タイムスパン) を導入する．五線譜には音符と休符があって音の鳴っていない時間帯が明記されているが，音の鳴っていない時間帯もいずれかのタイムスパンに含まれると考える[⋆1]．直感的には，タイムスパン木は，動機や楽節に相当する数小節から数十小節の長さの旋律の静的な構造を表現している．

一方，情報学的には，タイムスパン木は旋律の静的な構造を表現するために，発音時刻と休符を捨象し，旋律を正規化したと考えることができる．タイムスパン木により旋律の静的な構造を容易に比較できるようになったことで，旋律間の共通な特徴あるいはその旋律だけが持つ特徴等の検出や判定がより容易になる．

3.2 第 0 型表現

まず先行する文献 1) にて提案したタイムスパン木の表現法を第 0 型表現とする (図 1)[⋆2]．GTTM が規定するタイムスパン木の意味や性質を素朴に素性構造で表現する方法と言える

⋆1 GTTM のタイムスパン簡約において，最下層の楽譜レベルでは音符と休符が存在しているが，それより上のレベル (1 回以上の簡約を適用したレベル) では休符の概念が無い．
⋆2 元は，文献 4) で提案された旋律モーフィングアルゴリズム中で暗黙的に採用された表現方法である．

だろう．ここで $\tilde{}tree$ と $\tilde{}event$ はその素性構造全体の型 (sort) を表し，集合表現 $\{x, y\}$ は

$$\begin{bmatrix} \tilde{}tree & \\ head & \{\boxed{\text{i}}, \boxed{\text{j}}\} \\ dtrs & \{\begin{bmatrix} left & \begin{bmatrix} \tilde{}tree & \\ head & \boxed{\text{i}}\begin{bmatrix} \tilde{}event \end{bmatrix} \end{bmatrix} \\ right & \begin{bmatrix} \tilde{}tree & \\ head & \boxed{\text{j}}\begin{bmatrix} \tilde{}event \end{bmatrix} \end{bmatrix} \end{bmatrix}, \bot\} \end{bmatrix}$$

$$\begin{bmatrix} \tilde{}event & \\ pitch & Pitch \\ pos & \begin{bmatrix} bar & Integer \\ meter & Length \end{bmatrix} \\ duration & Length \end{bmatrix}$$

図 1 タイムスパン木の第 0 型表現における木 (上) と ピッチ・イベント (下)

x あるいは y の選択を表し，$\bot$ は空を表す．素性 dtrs (daughter) が空の素性構造はタイムスパン木の葉に対応する．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 1 下では，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いずれかの head から選択される．つまり，タイムスパン木の形が $\diagup\!\diagdown$ の時は $\sigma.head = \sigma.dtrs.left.head$ の制約が成立し，$\diagdown\!\diagup$ の時は $\sigma.head = \sigma.right.head$ の制約が成立する．Pitch の具体例は $C4$, $B\flat6$, $F\sharp3$ などである．pos 素性は発音時刻 (onset) を表し，楽譜上の位置情報として先頭から何小節め (bar) の何拍め (meter) という形式で表現する．本稿では素性 meter の単位を四分音符 1 つの音長とし，その型を Length とおく．素性 duration の値も Length 型である．

第 0 型表現ではピッチ・イベントに限定して記述したので，2 つのタイムスパン木の *join* (あるいは単一化) を考えた時，一方の休符の位置に他方の音を置くことは整合的な操作である．つまりそれらがモノフォニ条件を満たす場合は，一般に素性をマスクするだけでその 2 つの旋律は単一化可能である．この時，文献 1) の単一化アルゴリズム (図 $9^{1)}$) 中の "$unify(S_1, S_2)$ by (4)" の行は実行しない．

タイムスパン木の形は単一化可能でも，2 つの旋律がモノフォニ条件を満たさない場合，この時，音高 (pitch)，発音時刻 (pos)，音価 (duration) どうしの *join* や *meet* の定義を与えなければならない．各属性ごとにどのような抽象レベルと表現を導入すべきかについては詳細な議論を要する[5]．最も単純な解決法は，項モデル (term model) で意味を与えることであり，例えば演算 $C4 \sqcup G5$ の値を '$C4 \sqcup G5$' という項で表現する．しかし，この方式では，音楽的な意味[⋆1]を反映した抽象化や構造同値性を表現することが難しい．

3.3 第 I 型表現

図 2 に示すように，タイムスパン木自体を表現する tree 素性構造からインデックスの参照を削除し，その代わり dtrs 素性の下に新に primary 素性を導入し，head の値を左右どちらの下位の head から選択するかを指定する．primary 素性の値は 'left' か 'right' のいずれかである．同図下左のように新に timespan 素性を導入し，下右のように Timespan 型を定義する．

$$\begin{bmatrix} \tilde{}tree & \\ head & \begin{bmatrix} \tilde{}event \end{bmatrix} \\ dtrs & \{\begin{bmatrix} primary & \{`left', `right'\} \\ left & \begin{bmatrix} \tilde{}tree \end{bmatrix} \\ right & \begin{bmatrix} \tilde{}tree \end{bmatrix} \end{bmatrix}, \bot\} \end{bmatrix}$$

$$\begin{bmatrix} \tilde{}event & \\ pitch & Pitch \\ timespan & Timespan \end{bmatrix}$$

$$\begin{aligned} Timespan ::= \; & Length \mid Length \; `\sqcup' \; Timespan \\ & \mid Length \; `\sqcap' \; Timespan \\ & \mid Timespan \; `\sqcup' \; Timespan \\ & \mid Timespan \; `\sqcap' \; Timespan \end{aligned}$$

図 2 第 I 型における tree 型 (上)，event 型 (下左) と Timespan 型 (下右)

⋆1 例えば，音高の間にはオクターブの関係，半音の関係，五度の関係などがある．pos に関しては，例えば，各小節の先頭の拍は強拍と呼ばれ特別な意味を持っていたり，旋律が弱拍から始まる場合を auftakt と呼んで区別する．

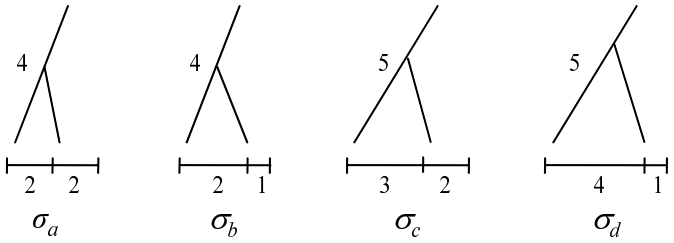

図 3 第 I 型タイムスパン木の例

例えば，図 3 中 σ_a は 2 音から成る旋律で，2 つの音とも timespan 素性の値が 2 であるようなタイムスパン木を表している．ピッチ・イベントより 1 段上のレベルでは 1 番目の音の方が head となり，その head は長さ 4 のタイムスパンを支配する．

図 3 に示すように *join* 演算の実行は，対応する各素性の値をトップダウンに再帰的に *join* していく (*meet* についても同様である)． 図中 σ_b と σ_c のタイムスパン木を *join* した結果

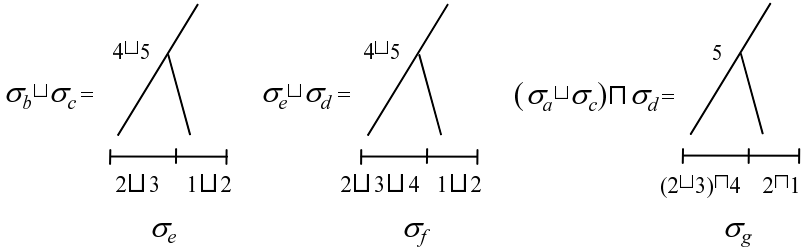

図 4 第 I 型タイムスパン木間の *join* 演算の例 (1)

を σ_e に示し，葉とその 1 段上の head の各音の timespan 素性値を表示した．一般に *join* や *meet* 演算を経たタイムスパン木に現れる timespan 素性値には '⊔' や '⊓' が含まれているので可聴ではない．次に σ_e と σ_d の *join* 演算の結果を σ_f とすると，$\sigma_b \sqcup \sigma_c \sqcup \sigma_d = \sigma_f$ にもかかわらず $\sigma_a \sqsubseteq \sigma_f$ が成立する[⋆1](図 4)．そして σ_g のような演算を行うと，head の timespan 値はスカラ値になるが，葉の音の timespan 値は Timespan 型の値である．

第 I 型タイムスパン木表現の特徴は以下の通り．第 I 型タイムスパン木 σ_1，σ_2 の *join* 演算は，定義により $\sigma_1 \sqsubseteq join(\sigma_1, \sigma_2) \wedge \sigma_2 \sqsubseteq join(\sigma_1, \sigma_2)$ を満たす[⋆2]．head の位置は異なっていてもタイムスパンが等しいような旋律の *join* 演算の結果は可聴になる．その時，各タイムスパンの開始時刻は，旋律の先頭から timespan 素性の値を積算することで算出する．head の実際の発音時刻 (onset) は，各 head が支配しているタイムスパン内のどこかに適切に具体化することで得られる．

図 5 は深さの異なるタイムスパン木の *join* を示す． 素性構造 σ に含まれる timespan 素性の値に関する制約として，素朴には $\sigma.head.timespan = \sigma.left.head.timespan + \sigma.right.head.timespan$ という制約式 (1) の成立が期待されるが (ここで '+' は Timespan 型の値の加算を意味する)，上で見たように第 I 型では一般にこの制約は成立しない．

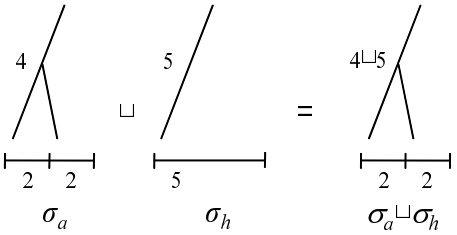

図 5 第 I 型タイムスパン木間の *join* 演算の例 (2)

3.4 第 II 型表現

ここでは前節の制約式 (1) が成立するようなより正確なタイムスパン木表現を検討する．あるタイムスパンの値がその直下レイヤの 2 つタイムスパンの値の和から成ることを表現する構造化音長 (structural length，以下 StrLen と記す) を導入する (図 6 右)．構造化音長の例には，$4,\ 4\langle 2,2\rangle,\ 8\langle 4\langle 2,2\rangle, 4\langle 3,1\rangle\rangle$ などがある．例えば $4\langle 3,1\rangle$ は，タイムスパン 4 の head ノードとその左右の子ノードのタイムスパンが各々 3, 1 であることを表わしている．タイムスパン木自体を表現する tree 素性構造は第 I 型表現と同じ構造を用い (図 2 上)，Length の代わりに StrLen を用いて新しいタイムスパンの型 StrTS を定義する．

⋆1 このような spurious が出現しても，*join* の定義である「条件 $a \sqsubseteq c \wedge b \sqsubseteq c$ を満たす c の中で最小のもの」は満たしている．

⋆2 timespan 素性以外の，pitch, pos, duration 素性の *join* 演算に関しては，第 0 型 (3.2 節) と同様に項モデルで意味を与えるとする．

$$\left[\begin{array}{ll} \tilde{}event & \\ pitch & Pitch \\ timespan & StrTS \end{array}\right]$$

$$\begin{aligned} StrLen ::= \;& Length \\ & \mid Length`\langle'StrLen`,'StrLen`\rangle' \\ StrTS ::= \;& StrLen \mid StrLen`\sqcup'StrTS \\ & \mid StrLen`\sqcap'StrTS \\ & \mid StrTS`\sqcup'StrTS \\ & \mid StrTS`\sqcap'StrTS \end{aligned}$$

図 6 第 II 型における event 型 (左) とタイムスパン型 (右)

新しいタイムスパンの型 StrTS によって表現されたタイムスパン木の例を図 7 に示す. 図中 σ_i の右半分の部分木において，左枝のタイムスパン 2，右枝のタイムスパン

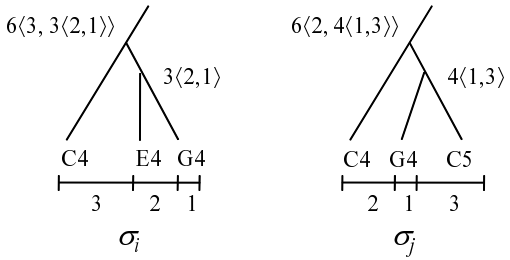

図 7 第 II 型タイムスパン木の例

1 より，その head のタイムスパンは $3\langle 2,1\rangle$ となる．また，σ_i と σ_j の *join* 演算の結果のタイムスパン $(\sigma_i \sqcup \sigma_j).head.timespan$ の値は $6\langle 3, 3\langle 2,1\rangle\rangle \sqcup 6\langle 2, 4\langle 1,3\rangle\rangle$ となり，$(\sigma_i \sqcup \sigma_j).dtrs.right.head.timespan$ の値は $3\langle 2,1\rangle \sqcup 4\langle 1,3\rangle$ となる．

第 II 型タイムスパン木表現の特徴は以下の通り．タイムスパン木途中の head においても重複して timespan 素性に StrTS 型の値を持たせるのは，部分木を取り出した時にもその構造に一貫性があるようにするためと，タイムスパン木が再帰的な構造を持っている方が何らかの性質を証明する場合に好都合だからである．第 II 型タイムスパン木ではタイムスパンの構造をより正確に表現できるので，第 I 型のような spurious は生じない．

4. おわりに

今後の課題は以下の通り:

- 第 I, II 型の表現に対する *join* 演算, *meet* 演算が，*join*, *meet* の定義 (2 節) を満たすことを確認する．
- 第 II 型の表現において以下の制約が成立することを証明する．
  $\sigma.head.timespan = \sigma.left.head.timespan + \sigma.right.head.timespan$
- *join* 演算式 $C = A \sqcup B$ の手続的解釈には次の 3 通りが考えられる:
  (a) タイムスパン木 A, B が与えられた時，$C = A \sqcup B$ なる C を計算する，
  (b) タイムスパン木 C が与えられた時，$C = A \sqcup B$ を満たすような A と B の組を見つける，
  (c) タイムスパン木 A, C が与えられた時，$C = A \sqcup B$ を満たすような B を見つける，
  (c) は (b) の特殊な場合であるが，与えられる A, C の組によって B の存在しない場合がある．一方 (b) には必ず解が存在する．*meet* に関しても同様に 3 通りの手続的解釈が考えられる．本稿では (a) の解釈に従って表現とアルゴリズムを議論したが，実用的には，表現はそのままで (b)，(c) のアルゴリズムを考案することが望まれるだろう．

さらに現在，複数の旋律から 1 つの旋律を合成する関数として，単一化，*join* に続き，ある旋律 B をもう一方の旋律 A の中に埋めこむ 2 引数の関数 embed に有用性があるのではないかと考えている．embed は，A については $A \sqsubseteq embed(A,B)$ が成立するが，B については成立するとは限らない．そのような embed の仕様とアルゴリズムが何通りか考えられるので，理論的に良い性質と有用性のトレードオフに関して検討を加えたい．

謝辞: 本研究は文部科学省科学研究費補助金 (研究課題名：生成的音楽理論に基づく自動楽曲分析器の構築，課題番号：20300035) の補助を受けて実施しました．

参 考 文 献

1) 東条敏, 平田圭二, 浜中雅俊, タイムスパン木の単一化可能性, (社) 情報処理学会 音楽情報科学研究会, 2010-MUS-86, No.22 (2010).
2) 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, (社) 情報処理学会 音楽情報科学研究会, 2010-MUS-85, No.4 (2010).
3) Fred Lerdahl and Ray Jackendoff, *A Generative Theory of Tonal Music*, The MIT Press (1983).
4) Masatoshi Hamanaka, Keiji Hirata and Satoshi Tojo, Melody Morphing Method Based on GTTM, In *Proc. of ICMC 2008*, pp.155–158.
5) 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).
6) Bob Carpenter, The logic of Typed Feature Structure, Cambridge University Press

Vol.2010-MUS-88 No.3
2010/12/4

(1992).
7) 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, No.5, pp.605–612 (2007).