1D2-4

楽曲構造束とその上の演算系

Lattice for Musical Structure and Its Arithmetics

平田 圭二[*1] 東条 敏[*2]
Keiji Hirata Satoshi Tojo

[*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
Japan Advanced Institute of Science and Technology

In this paper, we describe and discuss problems in terms of the music formalization framework proposed by us last year. Currently, we aim at building a contents design framework for ordinary people in a formal approach. Based on the knowledge representation technology, an art piece and contents are represented by a feature structure and the high-level operations upon the feature structure are available. When we apply the framework to implementing a practical musical system, we face some problems related to algebra, knowledge representation, and/or implementation. These are the issues concerned with: a relative pseudo-complement introduced for increasing descriptive power, an abstraction level of the distance between musical fragments, and a formalization of musical operations in addition to that of musical relations.

1. はじめに

我々は，デザインの専門家でない一般の人々でもより積極的にデザイン活動に従事できるよう，事例を利用したデザインという枠組を提案している [Hirata 06]．この枠組を利用することで，一般ユーザでも既存のデザイン事例を利用して比較的容易にデザインに関する指示を出せることが期待され，既存のデザイン事例を出発点として修正・加工を行うことで比較的容易にデザイン作業を進められることが期待される．

我々は事例を利用したデザインに理論的基盤を与えるため，まず音楽を題材とし，楽曲を修正・加工する操作や関係の形式化を試みた．音楽理論 GTTM[Lerdahl 83] に基づいて音楽構造を抽出し，楽曲全体を木構造として表現する手法を提案した [Hirata 02, Hirata 03]．木構造中の各音には重要度が付与されているため，重要度を閾値として，楽曲の簡略な骨組みを取り出したり，より原曲に近い細かいレベルの音まで含む構造を取り出したりすることができる．このように，重要度を変えて一つの楽曲から取り出してきた音列を二つの異なる楽曲と考えると，その間には上位下位の半順序関係が存在する．こうして楽曲間に半順序関係を定義し導入することで，楽曲を要素とする束 (楽曲構造束) を構成できる．その束上の meet (最小上界, least upper bound) や join (最大下界, greatest lower bound) 等の基本演算を組み合わせることで，楽曲の修正・加工が実現できることを示した [Hirata 05]．ここで，楽曲は AVM (attribute-value matrices; 属性構造) で表わされるため，楽曲間の順序づけや meet, join と言った束演算は純粋に数学的なアルゴリズムとして実現可能である．

しかしこの枠組みに基づいて楽曲に演算を適用する音楽システムを構築しようとすると，以下のような課題に直面する:

- 包摂関係, meet, join だけでは演算の記述力が不足しているために導入した相対擬補元に関する課題,
- 楽曲 (束上の元) 間の距離の抽象化に関する課題,
- 楽曲間の関係だけでなく，楽曲を修正・加工する操作や手順の形式化に関する課題

である．

本稿ではこれら各課題とその解決法について議論する．

A: hirata@brl.ntt.co.jp, tojo@jaist.ac.jp

2. 相対擬補元の計算

相対擬補元は四則演算で言えば正の数に対する負の数に相当し，相対擬補元の導入は演算系の記述力を飛躍的に向上させる [小野 94, Hirata 05]．相対擬補元は以下のように定義される．ある代数系において，分配的な演算 $\sqcap$ (meet), $\sqcup$ (join) が定義されており，各要素の間に順序 $\sqsubseteq$ が存在すると仮定する．このとき，任意の二つの元 a, b に対して $a \sqcap x \sqsubseteq b$ となる x のうち最大のものが一意に存在するとき，これを $a \supset b$ と書き，a の b に対する相対擬補元 (relative pseudo-complement) と言う．任意の二元に対して相対擬補元が存在するとき，この代数系は相対擬補束であると言う．

2.1 効率的なアルゴリズム

相対擬補元の計算には，対象としている領域中から条件を満たす元を集め，その内の最大の元を探すという操作が含まれている．素朴な実装では，データベース中の要素 (オブジェクト) を全解探索することになり効率が良くない．そこで我々は，有限な分配束において $a \supset b$ を効率的に計算するアルゴリズムを提案する (ただし $a \not\sqsubseteq b$ かつ $a \not\supseteq b$).

ステップ 1: $a \sqcap (b \sqcup x) \sqsubseteq b$ を満たす x をすべて集める $(x_1, \cdots, x_n)$.
ステップ 2: $a \supset b = b \sqcup x_1 \sqcup \cdots \sqcup x_n$

単なる x でなく $b \sqcup x$ に限定して探索しても同じ相対擬補元を求めることができる．

2.2 最大要素に対する制限

上の相対擬補元 $a \supset b$ の定義中には，最大の x を探す手続きが含まれている．素朴な実装では，この最大の x の値を計算するため，まず a や b と無関係な要素 y をデータベース中から探し ($a \not\sqsubseteq y$ かつ $a \not\supseteq y$. $b \not\sqsubseteq y$ かつ $b \not\supseteq y$)，y との join 演算を繰り返すことになるであろう．最大の x の値は，データベース中の a や b と無関係な要素 y の影響を受ける．その結果，相対擬補元の計算結果は限りなく上方，すなわち音が稠密になる方向に行ってしまい，いずれの計算結果も「ほとんどすべての音を含んだ構造」になってしまう[*1].

[*1] 実際には，1 回の join で楽曲を統合した結果の構造も音楽的に不必要な音を多く含んだ構造になってしまうことが多い．

ここで「ほとんどすべての音を含んだ構造」とは，数学的な定義に従えば束における $\top$ (top) に近い要素に対応する．例えば 8 小節という短い楽曲の世界でのコンテンツを考える．すると $\top$ (top) に近い要素とは，その 8 小節中すべての拍が 32 分音符で埋め尽くされ，各発音時刻にはあらゆる音高の音が縦に連なり，あらゆる木構造が重畳しているような状態を指す．このような音列に音楽的な意味はほとんど無いであろう．

そこでコンテンツとして意味のある制限を考える:

(1) a や b と無関係な要素 y をデータベース中から探す手続きを制限する
(2) $\top$ の木構造を制限する

(1) の制限は第 3. 章に関連している．元の間の距離に敏感でない抽象化をした場合，join の結果は音楽的に関係の薄い 2 つの音でも統合できてしまうからである．

次に (2) について議論する．まず，制限の一つとして木の深さを考える．$\top_n$ は深さ n までの木で最大の元とする．$\top_0$ の場合は深さゼロ，すなわち音を一つも含まない構造であり，$\bot$ (bottom) に等しい．この時，以下の関係が成り立つ:

$$\bot \equiv \top_0 \leq \top_1 \leq \cdots \top_n \leq \cdots \leq \top_\infty \equiv \top.$$

深さ以外にも，例えば楽曲の時間長 (タイムスパン長)，音の数等様々な尺度が考えられるので，$\top_n$ はより一般化して考えるべきであろう．現実には様々な尺度の $\top_n$ を状況によって使い分けたり併用したりすることになるであろう．従って $\top_n$ という制限には多峰性，すなわち同じ n で限定された top が数多く存在するという問題がいつも隣接することになる．我々は純粋に数学的な意味で束ではない代数構造の中でそれに似たものを代数的な手法で解く方法を考える必要があろう．

3. 楽曲間の距離の抽象化

3.1 meet/join 演算の定義

我々の枠組を利用して 2 つの音 C3, C4 を属性構造で各々表現したとすると，$C3 \sqcap C4$ (meet, 共通部分) 及び $C3 \sqcup C4$ (join, 統合) の結果としていくつかの値が考えられる (表 1)．ここで抽象化レベルとは，meet や join の定義を与える時に楽

抽象化レベル	1	2	3	4
meet	$\bot$	C	C3	[34]
join	$\top$	C$\top^\dagger$	C4	3|4

† さらに $\top$ を部分に含む項はその全体も $\top$ となると解釈する場合もある．

表 1: C3 と C4 の meet/join 演算結果の定義

曲間の距離にどの程度敏感な抽象化を行うかを表している (抽象化レベル 1 に近づくほど敏感)．まずレベル 1 は C3, C4 を全く別々の音と解釈する．レベル 2 は C3 をピッチクラス C とオクターブ位置 3 の組だと解釈する．レベル 3 はレベル 2 に加え，もしオクターブ位置に関して $3 \sqsubseteq 4$ という関係を加えた場合である．レベル 4 は，$3 \sqcap 4 = [34]$ のような「3 または 4」を意味する特殊な値と，$3 \sqcup 4 = 3|4$ のような「3 同時に 4」を意味する特殊な値を導入した場合である．meet/join 演算に関し，いずれの抽象化レベルの定義でも数学的には問題はない．

例えば，楽曲間の差異に最も敏感な場合 (レベル 1)，ある程度の大きさを持つ楽曲どうしの meet を計算すると，一般には，かなり類似した楽曲どうしでない限り meet の結果は空疎 $\bot$ あるいはそれに近い値になってしまう．この場合，実装は簡単であるが，正比例アルゴリズムの適用可能条件 (3.2 節, $b \sqcap x \neq \bot$) を成立させることは難しく，したがって正比例アルゴリズムが適切に機能する条件は限られる．

しかし，楽曲間の距離に敏感でない定義を用いると (例えばレベル 4)，以下に挙げるような別の問題が生じる:

(1) 上で導入した [34] や 3|4 のような特殊な値を含む項の meet や join の定義が曖昧
(2) システム内部表現の実世界コンテンツへの変換が曖昧

(1) について，例えば，meet 演算 "[34] $\sqcap$ 3|4" の値を他の演算と矛盾なく定義するのは困難である．この定義にも要素間の差異に関するレベルが考えられる．(2) について，meet の計算結果の項中には未定義部分が残ったり [34] のような特殊な値が現れる場合がある．そのような項を実世界コンテンツとして具体化する場合，未定義部分には何らかの実世界に存在する具体的な項を割当てなければならないし，3|4 のような項に対しては，逆に，多重部分から必要な情報を取捨選択する必要がある．

3.2 正比例アルゴリズムの適用範囲

相対擬補元 (第 2. 章) を利用した有用な演算の一例として正比例のアルゴリズムがある．これは，元 a, b が与えられた時，未知の x に対して正比例の関係 $a : b = x : y$ にあるような y を求めるアルゴリズムであり，

$$\mathcal{A}_{ab}(x) = a \supset (b \sqcap x)$$

と定義される．例えば，元 a, b は 2 つの楽曲を表現しているとすると，a を b に変換するように未知の入力 x を変換すると解釈できるので，既存の編曲を模倣して未知の楽曲を編曲するアルゴリズムとして利用できる．

この正比例のアルゴリズムは次のようなステップで設計された．正比例の関係 $a : b = x : y$ における積を meet に対応付けて $a \sqcap y = b \sqcap x$ を得る．この式を満たす y の値を求めるため，まず適当に小さな初期値を y に与えてから $a \sqcap y \sqsubseteq b \sqcap x$ を満たしながら y の値を大きくしていくと考える．これは相対擬補元 $a \supset (b \sqcap x)$ の計算に等しい．

しかし $\mathcal{A}_{ab}(x)$ の適用範囲は広くない．なぜなら，式 $a \sqcap y = b \sqcap x$ が意味を持つためには

- $b \sqcap x \neq \bot$
- $a \geq b \sqcap x$

という制約を満たす必要があるからである．実際の楽曲はある程度の大きさや多様さを持っているので，元の間の距離の抽象化レベルを 4 に設定したとしても，上の制約を満たす場合は多くないと思われる．一方，サイズの小さい楽曲に関しては相対的に上の制約が満たされる場合は多く，正比例のアルゴリズムも機能する．

4. 操作の形式化

4.1 アレンジ操作

上述した半順序関係に基づく楽曲の修正・加工の枠組は，修正や加工する前後の関係を形式化したものであり，大きな木構造のどの局所的な部分を修正したり加工したかという情報は十分に形式化されていない．例えば $a \sqsubseteq b$ という式は，楽曲 a と b の間の関係を述べているだけで，楽曲 a のどの部分が

変更を受けてより大きな値を持つ b となったかに関する情報は記述されていない．

そこで，小さい部分木構造を修正・加工するという情報と大きな木構造のどの局所的な部分に操作を加えるかという情報を区別し，本節では，後者の操作に関する情報を形式化することを考える．議論を簡単にするため，楽曲の修正・加工を，単純な主題旋律とその変奏の編曲 (アレンジ操作) に限る．

単音あるいは旋律全体に対するアレンジ操作は写像と見なせる．特に楽曲を属性構造で表わし，かつその属性構造がこの写像によって全体構造の中での位置を変移しない場合，そのようなアレンジ操作は射 (morphism) と考えることができる [Barwise 97]．すなわち，そのドメインとコドメインが元の属性構造において同じ位置に据えられるような時にのみアレンジ操作は射であるとする．

ドメインが単音であるような射には，以下のようなものが考えらる:

f_1　付点をともなったリズムへの変更,
f_2　トリルをつける,
f_3　三度と五度を重ねる,
f_4　三連符にする．

これらは，もともとの単音を属性構造の中での位置を変更せずに，複数の音からなる音列・和音に加工する写像である．実際に，パッサカリア (passacaglia; あるいはシャコンヌ) と呼ばれる変奏曲では，与えられた三拍子のバスの主題旋律に対してそれ自身を装飾したり，その上に別の旋律が重ねられる [音楽中辞典 79]．作品としてはバッハの BWV582，ブラームスの交響曲第 4 番の第 4 楽章，ウェーベルンの op.1 などが有名である．パッサカリアでは，単純な旋律が与えられそれに装飾が施されるので，主題旋律と変奏の間には明らかな複雑さの序列が存在し，この序列に沿って複数の変奏が半順序関係を構成し，束を成す．

4.2　テンソルによる表現

楽曲を表わす属性構造 A に対してその中のドメインを指定して加工をするような射を f とする．このドメインが単音であれば，$f(A)$ は A の中のすべての単音に一律に f を適用して得られる構造を意味する．またアレンジ操作の射は可換 (commutative) ではない．すなわち一般に $f_i \circ f_j(T)$ と $f_j \circ f_i(T)$ は異なる．

これらの性質はドメインの異なる写像を合成した時に，その作用の及ぶスコープを決定するために重要である．例えば，旋律に対するアレンジ操作は旋律にならなければならない．旋律自体に対する加工としてよく知られたセリー (音列) の操作:

g_1　逆行
g_2　反行
g_3　拡大

を考えると，$g_i(A)$ は再び単旋律を生み出さなければならない．

属性構造 A に対して，その個々のイベントを統一的に B に書き換えた結果の構造を $A \otimes B$ と表現しよう．また，前記の f (単音に装飾を施すアレンジ操作) と g (旋律全体に対し旋律を生み出すアレンジ操作) の合成も，同様にアレンジ操作の積として $g \otimes f$ と表現しよう．各々のドメインが正しく反映されるように楽曲をアレンジ (修正・加工) できるとしたら，

$$(g \otimes f)(A \otimes B) = g(A) \otimes f(B)$$

のような関係が充たされることが好ましい．実は，この式はテンソル積の定義に等しいので，ドメインを指定したアレンジ操作の適用に関する形式化の候補として，テンソル積を用いることが考えられる [佐藤 05]．

5.　まとめ

本稿で挙げた課題を一般的に解決する方法を見い出すのは難しいであろう．実際に，楽曲や画像などのコンテンツをデザインするシステム毎に異なる解決策を必要とするかも知れない．しかし，我々が提案している枠組は，一般ユーザにとって有用な事例に基づくデザイン手法に共通した理解と応用可能性をもたらすものである．

今後も，事例に基づくデザインに，理論と実際の両面から取り組んでいきたい．

参考文献

[Barwise 97] Barwise, J. and Seligman, J.: Information Flow - The Logic of Distributed Systems -, Cambridge University Press (1997).

[Hirata 02] 平田, 青柳: 音楽理論 GTTM に基づく多声音楽の表現手法と基本演算, 情報処理学会論文誌 Vol.43, No.2 (2002).

[Hirata 03] Hirata, K. and Aoyagi, T.: Computational Music Representation based on the Generative Theory of Tonal Music and the Deductive Object-Oriented Database, *Computer Music Journal* Vol.27 (3), pp.73–89, The MIT Press (2003).

[Hirata 05] 平田, 東条: 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).

[Hirata 06] 平田, 片寄, 笠尾, 宮田, 原田: コンテンツのデザイン支援技術による社会貢献を目指して, 人工知能学会論文誌 Vol.21, No.2 SP-B, pp.215–218 (2006).

[Lerdahl 83] Lerdahl, F. and Jackendoff, R.: *A Generative Theory of Tonal Music*, The MIT Press (1983).

[音楽中辞典 79] 音楽中辞典，音楽の友社 (1979).

[小野 94] 小野寛晰:　情報科学における論理，日本評論社 (1994).

[佐藤 05] 佐藤，房岡: マルチエージェント環境におけるテンソル積を用いた知識構造, 第 19 回人工知能学会全国大会 (2005).