互換性と数学的構造を持つ音楽プロジェクトファイルの提案

黒田 和暉[1]　今井 克暢[2]　北須賀 輝明[1]　中西 透[1]

概要：音楽制作において，作業ごとに処理やフォーマットが異なり，環境が分離し，ある作業の音楽情報が別作業では扱いづらい問題がある．後から情報の抽出・追加を行うことでこの問題に対処する方式では不完全で手間がかかるため効率が悪い．そのため音楽プログラミング言語や DAW のプロジェクトファイルなどで，各作業が有益な情報を次の作業に受け渡せるようにする取り組みがある．しかし現状は，特殊な記法，構文，意味論となっており，汎用的な用途には向いていない．本論文では，従来方式と比較しつつ，圏論や集合，有向グラフなどの数学的な表現に基づいたプロジェクトファイルを提案する．その利点としては，外部記述による可読性や再利用性が向上，類似データの関係性の記述が行える．加えて，データ構造をシンプルに保つことで拡張性に富み，共同作業やバージョン管理のしやすさにも繋がっている．

1. はじめに

現代のコンピュータを用いて作曲や演奏を行うコンピュータ音楽の分野では，デジタル・オーディオ・ワークステーション (DAW) での作業が中心となっている．音楽制作では複数人が作業ごとに分業することもあるが，DAW では録音やミキシングなどの一連の作業が行えるにもかかわらず，DAW ごとに互換性がないことやプラグインの非所持により，情報の受け渡しが困難であったり，複数人で同時に作業を行うことが難しい．つまり現状では，処理やフォーマットが作業ごとに独自なものとなって，環境が分離しており，データの受け渡しは WAV ファイルや MIDI ファイルなどの出力のみになっている．よって他の重要な情報が喪失・隠蔽されて，別作業の情報が扱いづらい問題がある．

また再生環境では，計算能力の飛躍的な向上によるオンライン処理能力の増加や，VR・AR 機器といった新たな環境の登場がある．そのためインタラクティブミュージックのように，作曲時だけでなく再生段階で音楽情報の一部を改変したり，別の指標として利用したいという需要も増えていると考えられる．

本論文では，従来方式や DAWproject[2] と比較しつつ，圏論や集合，有向グラフなどの数学的な表現に基づいたプロジェクトファイルを提案する．その利点としては，まず物事に対して外部から記述するような形で記述することができるようになり，可読性や再利用性が向上する．また似ているデータ同士の関係性の記述が行える．加えて，データ構造をシンプルに保つことで拡張性に富み，更に共同作業やバージョン管理のしやすさにも繋がっている．

[1] 広島大学大学院 先進理工系科学研究科
[2] 福山大学工学部

2. 関連研究

2.1 デジタル・オーディオ・ワークステーション（DAW）

現代の多くの音楽制作ではコンピュータを用いて作曲や演奏を行っている．特にデジタル・オーディオ・ワークステーション (DAW) での作業が中心となっている．DAW は，音楽制作における録音からマスタリングまでのプロセスを行うシステムである．例えば，レコーディングスタジオでよく使用されている Pro Tools[1] がある．但し全体でみると 25%以上の市場シェアを持つ DAW はなく，様々な DAW が使われている [11].

DAW を用いて曲を再生する際には，DAW の付属または外部のソフトウェアである楽器・エフェクトのプラグイン，WAV ファイルのような音声情報，MIDI のような演奏情報，使用している楽器・素材の追加情報やその音を鳴らすタイミングや音符の情報，エフェクトなどの設定した情報の集まりであるプロジェクトファイルが必要である．

2.1.1 音楽制作の作業と環境

現在のコンピュータを用いた作曲工程では，様々なソフトウェアや情報，人物が関わっている．現代の音楽制作のプロセスは，おおむね図 1 の枠内が白の四角形のように様々な作業を経て，曲を完成させている．またマスタリングで完成ではなく，リミックスやサンプリングのような曲を再利用して新たに曲を作ることもある．さらに，曲の情報を用いて，音楽ゲームやビジュアライズするなどのインタラクティブな曲の再生が考えられる．

具体的に図 1 は 3 人組で演奏を行うスリーピースのバン

ドの作業プロセスを示している．バンドメンバーの３人に加え，マスタリングエンジニアや PA エンジニアも携わっている．音楽制作では複数人が作業ごとに分業し，音楽処理の環境が変化することも少なくない．環境によって，使用する DAW が異なったり，外部プラグインを持っていなかったりする．

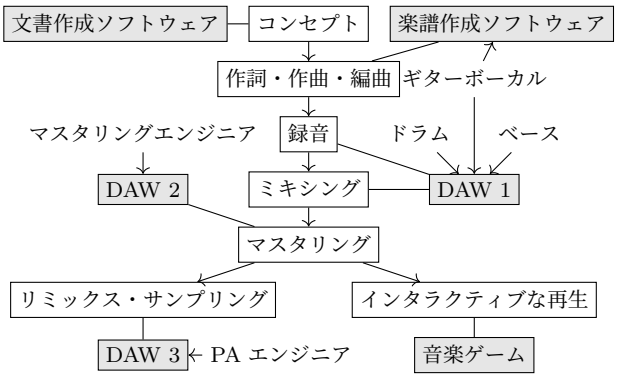

図 1　スリーピースによる音楽制作の分業の例

2.1.2　異なる音楽処理環境への受け渡し

1 で紹介したように，作業中に出現した重要な情報が喪失・隠蔽されて，別作業の情報が扱いづらい問題がある．

その従来方式の解決策は，楽曲ごとに「音楽抽出や分離」や「後から手動で情報の追加」である．例えば興味深い事例として，まず楽曲の中身を自動的に解析し，さらにユーザに共有して性能改善を図る Web サービス [16] がある．これらにより音から特徴的な情報を取得できるが，完全に抽出できなかったり，手間がかかったりして，作曲段階では存在した情報を再度追加することになって効率が悪い．

もう一つの解決策として，音楽制作に特化 [4] したプログラミング言語 [5], [8], [15] の採用がある．例えば Csound[5] では音楽のデジタル信号処理から音楽規則まで様々な記述ができ，多くの音楽を表現できるため，楽器や演奏，その楽譜などをコード記述として包括して扱うことができる．さらに近年，ソースコードの楽譜として利用を推進する研究 [17] や，楽譜の進化における新たな道としてライブ・コーディング [3] を論じている研究 [6] もあり，音楽プログラミング言語や専用のソフトウェアについての研究が多数見られるようになった．しかし音楽プログラミング言語は，音楽制作は工学というよりも創造的なプロセスであるため，迅速なプロトタイピングと実験をサポートする言語が必要とされる [8]．そのため，特殊な記法，構文，意味論となっており，様々な作業にも対応するための汎用的な言語には向いていない．

2.1.3　DAWproject

また，異なるベンダーの DAW 間でもユーザーデータを転送することができるフォーマットである DAWproject[2] も登場している．現在は，Bitwig Studio 5.0.9 と PreSonus Studio One 6.5 間でデータ交換ができる．形式としては XML を用いた木構造になっており，トラックやチャンネル，オートメーション，プラグインなどの要素が定義されている．記述例を Listing 1 に示す．

Listing 1　DAWproject の記述例（一部省略）

```
% Structure
<Track name="Piano" id="id1">
  <ClapPlugin deviceID="com.example.piano"
      deviceName="Piano" deviceRole="
      instrument">
    <State path="plugins/example.p-preset" />
  </ClapPlugin>
</Track>
<Track name="Vocal" id="id2">
  % 省略
</Track>
% Arrangement
<Clip time="0.0" duration="2.0" playStart="0.0"
     track="id1">
  <Note time="0.0" duration="0.25" key="65" />
  <Note time="1.0" duration="0.25" key="60" />
</Clip>
<Clip time="0.0" duration="4.0" track="id2">
  <Audio duration="2.0" sampleRate="48000">
    <File path="audio/vocal.wav" />
  </Audio>
</Clip>
```

2.2　圏論的アプローチ

圏論とは，数学的構造間のの関係を抽象的に扱う分野の１つである．音楽と圏のような数学的な表現を応用した先行研究としてはソフトウェア RUBATO[7] がある．RUBATO は分析やパーフォーマンスが行える．

ここでは本提案方式で応用する，データベースを圏論の概念を用いて表現した圏論的データベースと論理構造を使用するためのトポスについて紹介する．

2.2.1　圏論的データベース

Algebraic database[10] は，圏論の概念を用いてデータベースのスキーマやデータ操作を表現している．まずデータベースのスキーマは，一つの圏として，対象はテーブルやデータ型，射はカラムで表現する．またデータベースのインスタンスは，関手 $I : S \to Set$ として表現される．

2.2.2　トポス

圏 C が初等トポスとは，カルテシアン閉で部分対象分類子を持つという条件を満たす．部分対象分類子は図 2 のように表される．例えば，圏 Set や圏 $FinSet$ は

$$\begin{array}{ccc} X & \xrightarrow{!} & 1 \\ \downarrow & \lrcorner & \downarrow{\scriptstyle true} \\ Y & \xrightarrow{\chi} & \Omega \end{array}$$

図 2　部分対象分類子

$\Omega = \{true, false\}$ となる初等トポスである．

2.3 Conflict-Free Replicated Data Types (CRDTs)

Conflict-Free Replicated Data Types (CRDTs) は，分散コンピューティングにおいて，強い結果整合性 (Strong Eventual Consistency (SEC)) を保証するデータ型の一つである [13]．SEC は，正確なレプリカは最終的に状態が収束し，かつ，順序不同の更新は同じ状態であることを保証する．具体的には，送られてくる情報の順番に依存しないために，情報を更新する操作が可換であることで解決している．

2.3.1 Grow-only Set

CRDT の Grow-only Set (G-Set)[12] は，追加のみ行える集合である．そのため，Listing 2 で示すように要素の追加は和集合を求めればよい．G-Set は，削除も行える集合の 2P-Set や有向グラフなどに応用される．本提案方式では，操作ベースの 2P2P-Graph [12] を用いる．

Listing 2 State-based grow-only Set (G-Set)[12]

```
payload set A
    initial φ
update add(element e)
    A := A ∪{e}
query lookup(element e) : boolean b
    let b = (e ∈A)
compare (S, T) : boolean b
    let b = (S.A ⊆T.A)
merge (S, T) : payload U
    let U.A = S.A ∪T.A
```

3. 提案形式

本プロジェクトファイルのフォーマットの提案の前に，まず 3.1 で本プロジェクトファイルの使用が想定されているシステムについて紹介する．そして 3.2 では本プロジェクトファイルの音楽情報の保存形式の定義や有用性について説明し，3.3 では共同編集や変更履歴の管理を行うためのデータ構造について説明する．

3.1 提案システム

図 3 は本プロジェクトファイルを用いた音楽制作や再生を行うシステムを示している．現在の多くの DAW や音楽プログラミング言語のツールが，各自で音声出力や操作，変更履歴などを実装している．一方で本システムでは，直接 DAW のような第三者のソフトウェアがプロジェクトファイルを編集するのではない．各ソフトウェアに内蔵された音楽管理機構という新たなサウンドミドルウェアを通して，音声出力だけでなく，音楽情報取得や著作物管理などの 4 つの音楽処理で頻出する処理も行う．これにより制作者は，複数の DAW と音楽プログラミング言語を同時に使用できるようになり，一つの環境に依存しない制作が可能になる．さらにそれらのツールの開発者は，UI や操作のようなユーザビリティを向上させることに専念できる．

つまりプロジェクトファイル駆動型では，プロジェクトファイルは中間表現としての役割を担い，それをベースに処理が行われている．音楽管理機構の処理を利用できるようにプロジェクトファイルは拡張されており，楽器・エフェクトのプラグインや音声・演奏データなども含まれている．

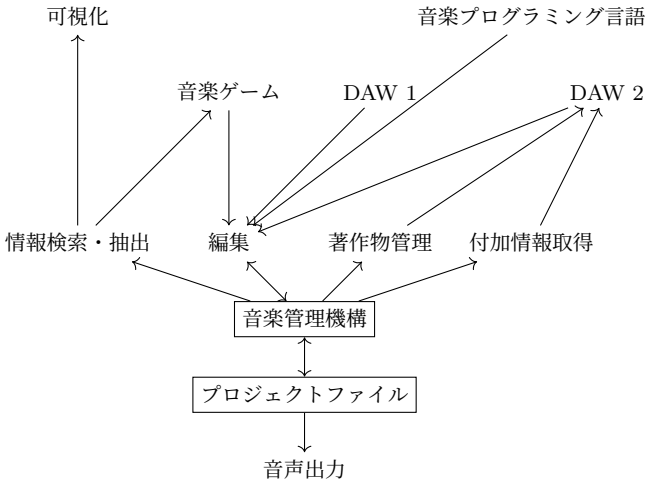

図 3 本システムによる統一された音楽処理環境の例

3.2 提案形式による音楽情報の扱い

本システムのように統一的にプロジェクトファイルを扱うには，完全性や再利用性，拡張性，可読性などの性質を満たすべきである．本プロジェクトファイルは圏論の概念を用いたデータの保持を行っている．

3.2.1 圏論の概念を用いる利点

圏論を用いる他の理由としては，いくつかの利点が挙げられる．圏論では，集合論のように物事を内部から記述するのではなく，外部から記述しており，圏の条件を満たしているその他の数学的構造に拡張できる．つまり，再利用性や拡張性の高いプログラムを組むことができるようになる利点がある．

また木構造のようなデータ構造による表現では，人によってデータの捉え方が変わることがある．そのため，厳格に決められた数学の概念からアプローチを試みることで，フォーマット独自の特殊な表現法や例外処理が発生せず，可読性が向上する．

さらに圏論では，双対性を明確にすることができる利点がある．例えば，集合論における直和と直積は，双対の関係になっている．このように異なる見方を与えることに

よって音楽制作においては想像力をかき立てる重要な役割を果たす利点がある．

3.2.2 音楽情報のデータ構造

提案形式では，圏の圏 Cat を用いる．本論文内で使用する Cat の対象は，圏 $FinSet$ と小さい圏 C であり，Cat の射は，関手 $F : C \to FinSet$ を使用する．

圏 $FinSet$ は，すべての有限集合が対象，それらのすべての写像を射とする圏であり，カルテシアン閉で部分対象分類子を持つため，トポスである．圏 C は，プロジェクトファイルが含む音楽情報の構造を定義するための圏であり，C から $FinSet$ への関手 F を通して，レコードの値を取得する．圏論的データベースの観点からすると，圏 C はデータベースのスキーマとして，関手 F はデータベースのインスタンスとして捉えることができる．

3.2.3 関手による音楽情報の記述

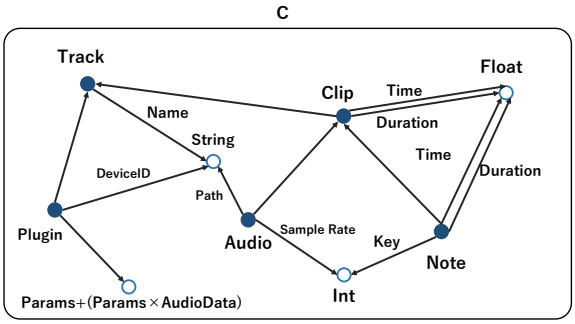

図 4 圏 C による音楽情報スキーマ

2.1.3 で紹介した DAW のプロジェクトファイル (Listing 1) に類似したスキーマとなる圏 C は，図 4 のようになる．各図では，恒等射は自明なものとして省略する．例えば各トラックは，楽器名称や演奏時間，出力ソースの情報が含まれている．そして圏 C を $FinSet$ への関手 $F : C \to FinSet$ を用意し，それによってインスタンスが与えられる．Track や Clip，Note などの対象はテーブルの ID 列としての役割を果たし，Time や Duration，Name などの射はそのコドメインの Float や String などの対象の型の値の列として捉えられる．

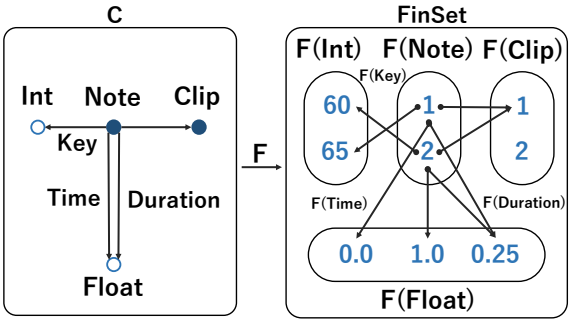

図 5 Note のインスタンス

例として，特別に Note 周りを抽出した圏で考えると，図 5 のように，F(Note) の要素が ID であり，時間やノートナンバーの値を持ったレコードとして考えることができる．

3.2.4 プラグインとソースの選択

複数人で作業を行う場合は DAW の違いだけでなく，所持しているプラグインの差異によっても互換性が損なわれる．そのため，プラグインのための圏 C の対象は，積と余積を用いて様々なデータを入れられるようにしている．具体的には，図 6 のように Plugin のソースを $Params + (Params \times AudioData)$ として持たせる．タプルの始めの要素は，二番目の要素の型が $Params(0)$ であるか $Params \times AudioData(1)$ であるか表している．そして二番目の要素で実際のパラメータの値またはパラメータの値と音声データが保存されている．DAW であるプラグインを使用する場合，初めはそのプラグインのパラメータの値を取得することができ，Params で保存される．その後プラグインを使用して再生されると，そのプラグインの LPCM などの音声データが取得できる．そのタイミングで，パラメータと音声データの両方を保存することで，プラグインを持っていない状態でも再生が行えるようになる．

このように積や余積を使用することで，簡潔な圏の記述のまま様々な構造の表現や追加が行える．またプラグインだけでなく WAV のような他のファイルやソフトウェアについても，複数の構造からソースの確保を行えるようになる．

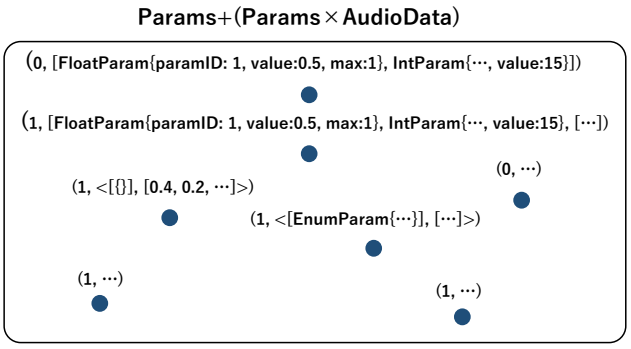

図 6 $Params + (Params \times AudioData)$ の集合

3.2.5 追加情報を与えられるためのトポス

上述の内容では，圏論や集合の概念を用いることで再利用性や拡張性，可読性を向上させつつ，従来のプロジェクトファイルの表現が行えることを示した．3.2.5 では，圏 $FinSet$ がトポスを成すことを利用して，論理構造を構築して，追加情報を与えたり，情報を抽出できることを紹介する．

まず FinSet の要素内で新たな情報を付加することを説明する．例えば，簡約されたトラックの情報として $(id, name) = [(1, guitar), (2, drum), (3, vocal)]$ があると

する．p =「弦楽器である」ということに対応する特性写像を用意する．この写像は，$id\ (F(Track))$ が 1 の時に $true$ に写され，そのほかの場合は $false$ に写すように定義する．部分対称分類子の性質により，$\{Track|p\} = \{t \in F(Track)|p(t) = true\} = 1$ が導かれる．このように，同じデータとして扱っていたものを要素ごとに，特徴を新たに付加することができる．

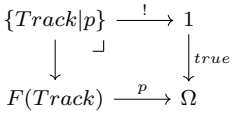

図 7　部分対象分類子の例

続いて，ある条件を満たす F(Track) の要素を探すことを考える．例えば，ギターボーカルの担当のトラックを把握したいため「弦楽器または人間である F(Track)」を探したい．上述と同様に q =「人間である」ということに対応する特性写像を用意する．$\cup = \Omega \times \Omega \to \Omega$ を用いることで，部分対称分類子の性質により $\{1, 3\}$ が得られる．このような操作が行えることで，再利用性や可読性が向上する．

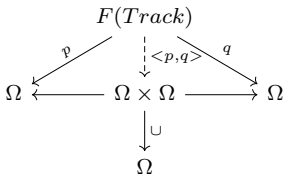

図 8　弦楽器または人間である F(Track) の部分集合の抽出

3.3　提案形式のデータ構造

本プロジェクトファイルは 3.2 の情報がそのまま保存されているのではなく，2.3 で紹介した CRDT のデータ型の状態で保存される．これにより，完全性に関わる共同編集と変更履歴の管理が行えるようになる．

3.3.1　詳細な変更履歴による利点

詳細な変更履歴を用いることで，著作物の管理における利点がある．CRDT のような詳細な変更履歴では操作の情報が蓄積されるため，誰がいつどの部分を編集したかを細かく把握できる．

まず一つ目の利点として，著作者を明確化できる．電子音楽では，音を出すためのプログラムとなる楽器やエフェクトとそのパラメータとなるメロディーや音色，曲の構成などが含まれる．プログラムとパラメータで著作者が違うことが多い．また楽器の一部を改変することも考えられる．そのような状況では著作者が曖昧になる可能性があるが，詳細な変更履歴によって明確に著作者を管理できる．

次に二つ目の利点として，一定の盗作を防ぐことができる．いつどの部分を編集したか把握できるため，どのような手順で制作したかわかる．そのため，試行錯誤することなくプログラムが完成しているなどの不自然な制作を発見できる．

また CRDT では，Eventual Consistency (EC) により，各レプリカは最終的に収束し，一貫性があることを目的にしているが，可換であることによる別の利点も得られる．

プロジェクトの全てを共有するために，全ての操作を他のレプリカに送信するのではなく，あえて自分しか使用しないことが分かっている部分の操作は他のレプリカに送信しなくても，追加の処理が不要なまま，共有する部分の更新を受け入れることができる．つまり可用性がある．例えばユーザの環境設定などのユーザ固有の処理に，仕様の異なる特別な設定ファイルを用意することなく一元に管理できるようになる．

さらに CRDT は，外部のライブラリをインポートする際にも有用である．外部のライブラリを考えていく際には，各操作にユニークな識別子を持たせることで，様々なプロジェクトであるライブラリの様々なバージョンをインポートしているときの最適化に役立つと考えられる．

3.3.2　シンプルなデータ構造

ここでは，具体的なデータ構造について説明する．対象や射は，点と矢印を持つ有向グラフのデータ構造で表される．そのため通常のグラフでは対象や射，圏などをそれぞれ別の構造体で表す必要がある．さらに本プロジェクトファイルは実際には圏論的な表現だけではなく，音声データなどの情報も含む．他にも，VR・AR の発展に伴い，AR 機器で電子楽器を演奏することが想定される．その際には，周辺の空間構造のデータを扱うことが必要になるだろう．これらが全て別の表現方法を用いていると，新たな概念を追加する度に新たな実装や変更を行う必要が出てくるため，再利用性が下がるだけでなく，実装が複雑になりバグや実装工程数が増え，第三者ソフトウェアでの利用が憚られるようになる．加えて，3.3.1 で CRDT を使用する利点を紹介したが，CRDT の性質上，複雑なデータ構造を持つ CRDT を作成することは難しい．以上から統一的に扱えるようなデータ構造や規則が必要になる．

従って本プロジェクトファイルでは，有向グラフで表現

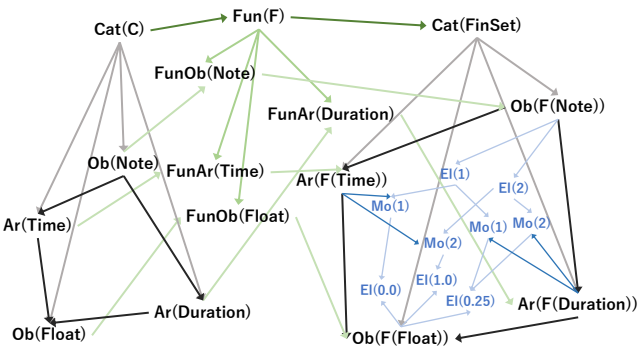

図 9　圏と関手，集合の有向グラフによる表現

される．圏や関手，集合などの各概念はノードとして表現され，近接するノードを知ることで，意味が与えられる．例えば，図 5 の 2 つの対象 Note と Float，そして 2 つの射 Time と Duration に限定した圏 C' を考えると，図 9 のような構造となる．注意として，エッジに色が付いているのは視認性のためであって，構造の差異ではない．圏 C' の対象 Note は，Ob を型とする Note という名前のノードとして表現する．射や関手なども同様にノードとして表す．

このような構造にすることの利点として，拡張性に優れていることが挙げられる．例えば，圏 C の射 Time から射 Duration への射，つまり射の射を考える必要が出てきた時に 2Ar を型とするノード $2Ar(TimeDur)$ を用意することで，$Ar(Time) \to 2Ar(TimeDur) \to Ar(Duration)$ のように，データ構造を変更することなく新たな概念の表現ができる．他の例として，関手と各対象の移り変わりの関係や，圏 $FinSet$ 内の射，つまり集合間の写像とその要素の各マッピングの関係のような上位の概念と下位の概念の関係を表現できる．例えば関手のノード $Fun(F)$ から $FunOb(Note)$ のノードへのエッジが存在することで，$FunOb(Note)$ は関手 F による Note の別の圏への射であることがわかる．

最終的にこの有向グラフは，CRDT の操作ベースの有向グラフ 2P2P-Graph を用いて表現される．そのため，フォーマットとしては各操作が保存されている．

4. おわりに

本論文では，圏論や集合，有向グラフなどの数学的な表現を用いることで，可読性や再利用性を向上させつつ，簡潔に情報を表すことができた．これにより DAWproject のような従来のプロジェクトファイルで使用される音楽情報が表せるだけでなく，共同作業や変更履歴の管理，より多くのデータ表現や抽出が行えるようになった．

4.1 今後の課題

本論文では圏の圏 Cat において，小さい圏 C と有限集合の圏 $FinSet$ のみを用いていたが，圏の条件を満たしているその他の数学的構造への移行や他の圏の登場，圏の拡張を行うことなど想定される．例えば，インスタンスの圏 $C \to Set$ を拡張した前層トポス [14] を使用したり，時間を扱いたい場合には，期間にわたる推論ができる時間に関する層 [9] を用いたりするとよいだろう．また本提案方式は，CRDT から有向グラフ，有向グラフから圏の圏というようなデータ変換が行われているが，各データ構造が適切であるか検証できておらず改善の余地がある．

今後は音楽制作や再生向けの構造を持つ方式に改善しつつ，応用ソフトウェアとの連携を試みて，本提案方式の課題の発見と改良を目指す．

参考文献

[1] Avid Technology, I.: Pro Tools, `https://www.avid.com/pro-tools`.
[2] Bitwig: DAWproject, `https://github.com/bitwig/dawproject`.
[3] Cycles, T.: Tidal Cycles, `http://tidalcycles.org/`.
[4] Dannenberg, R. B.: Languages for computer music, *Frontiers in Digital Humanities*, Vol. 5, p. 26 (2018).
[5] Lazzarini, V., Yi, S., Heintz, J., Brandtsegg, Ø., McCurdy, I. et al.: *Csound: a sound and music computing system*, Springer (2016).
[6] Magnusson, T.: Algorithms as Scores: Coding Live Music, *Leonardo Music Journal*, Vol. 21, pp. 19–23 (online), DOI: 10.1162/LMJ_a_00056 (2011).
[7] Mazzola, G.: *The Topos of Music II: Performance: Theory, Software, and Case Studies*, Springer (2018).
[8] McCartney, J.: Rethinking the computer music language: Super collider, *Computer Music Journal*, Vol. 26, No. 4, pp. 61–68 (2002).
[9] Schultz, P. and Spivak, D. I.: Temporal Type Theory: A topos-theoretic approach to systems and behavior, *arXiv preprint arXiv:1710.10258* (2017).
[10] Schultz, P., Spivak, D. I., Vasilakopoulou, C. and Wisnesky, R.: Algebraic databases, *arXiv preprint arXiv:1602.03501* (2016).
[11] Sethi, R.: Top 12 Most Popular DAWs (You Voted For) - Ask.Audio, `https://ask.audio/articles/top-12-most-popular-daws-you-voted-for`.
[12] Shapiro, M., Preguiça, N., Baquero, C. and Zawirski, M.: A comprehensive study of convergent and commutative replicated data types, PhD Thesis, Inria–Centre Paris-Rocquencourt; INRIA (2011).
[13] Shapiro, M., Preguiça, N., Baquero, C. and Zawirski, M.: Conflict-free replicated data types, *Stabilization, Safety, and Security of Distributed Systems: 13th International Symposium, SSS 2011, Grenoble, France, October 10–12, 2011. Proceedings 13*, Springer, pp. 386–400 (2011).
[14] Vigna, S.: A guided tour in the topos of graphs, *arXiv preprint math/0306394* (2003).
[15] Wang, G., Cook, P. R. and Salazar, S.: Chuck: A strongly timed computer music language, *Computer Music Journal*, Vol. 39, No. 4, pp. 10–29 (2015).
[16] 後藤真孝，吉井和佳，藤原弘将，中野倫靖ほか：Songle: 音楽音響信号理解技術とユーザによる誤り訂正に基づく能動的音楽鑑賞サービス，情報処理学会論文誌， Vol. 54, No. 4, pp. 1363–1372 (2013).
[17] 松浦知也， 城一裕ほか：音楽プログラミング言語のソースコードを楽譜と捉え，それを編集するツールの構想，研究報告音声言語情報処理 (SLP)， Vol. 2019, No. 17, pp. 1–4 (2019).