Chord Normalizer

A Type Checker for Music Design

Ziyue Yang

Outline

- Intuition & TODO
- Design Patterns
- Formal Definition
- Some Proofs
- Evaluation

Intuition

- Designing classic chord (for a melody) is difficult for…
  - Mode changing rules
  - Chord progression rules
  - Chord transformation rules
  - Limitation of the combination of melody and chord

- Manually check validity after catching inspiration is tedious!

TODO

- Design a programming language to
  - Generate music like other music programming languages (such as Nyquist by Roger B. Dannenberg)
  - Have some common control flows like the language presented in the textbook
  - However do chord checking using type system

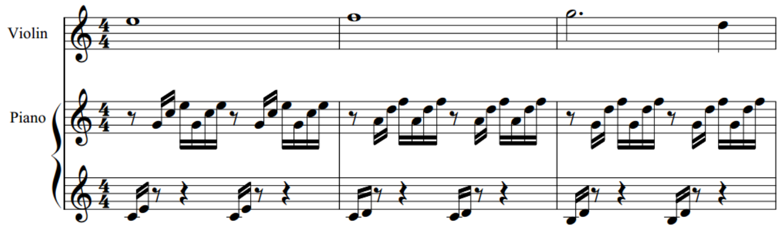
Outline

- Intuition & TODO
- Design Patterns
- Formal Definition
- Some Proofs
- Evaluation

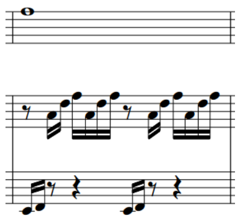
DP1 – Music Representation

- A piece of music has its own hierarchy structure
- We can describe it using trees
- A possible structure:
  - Passage
    - Segment
      - Phrase
        - Note Set
          - Note

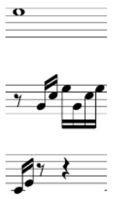
- Passage



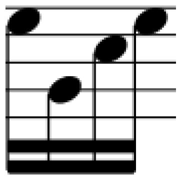
- Segment



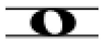
- Phrase



- Note Set

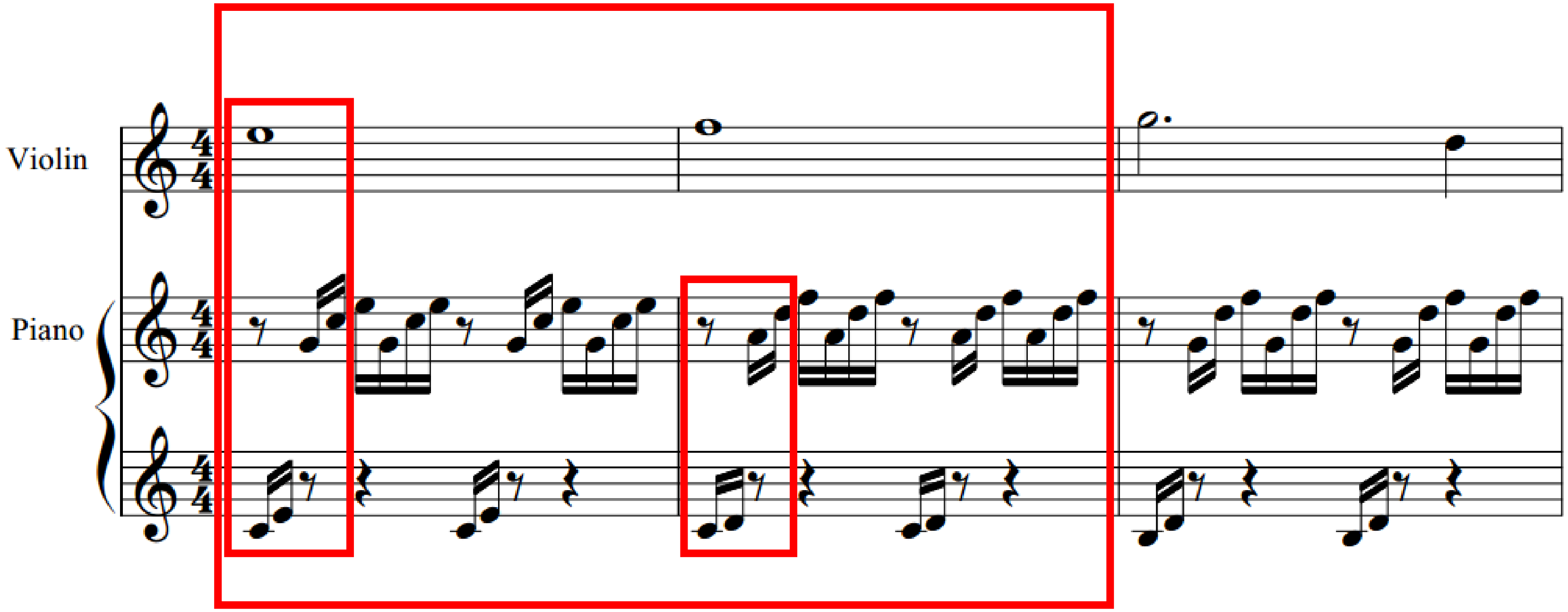


- Note



DP2 – Type Checking

- Chord construction and connection
- Mode connection



DP3 – Music Output

- OCaml
  - Traversing the music tree
  - Generating JSON
- Python
  - Serializing the notes
  - Generating MIDI

Outline

- Intuition & TODO
- Design Patterns
- Formal Definition
- Some Proofs
- Evaluation

Syntax

Terms

```
t ::=
        {#<STRINGV>|<STRINGV>|<STRINGV>|<INTV>#}
        MakeNoteset t t
        MakePhrase t t
        MakeSegment t(<INTV>,<STRINGV>)
        MakePassage t t
        ExportPsg t => <STRINGV>
```

Syntax

Values

```
v ::=
        {#<STRINGV>|<STRINGV>|<STRINGV>|<INTV>#}
        MakeNoteset v v
        MakePhrase v v
        MakeSegment v(<INTV>,<STRINGV>)
        MakePassage v v
        ExportPsg v => v
```

Syntax

Types

```
T ::=
        Note@<INTV>
        Noteset@<INTV>
        Phrase@(<INTV>,<INTV>)
        Segment@(<INTV>,<STRINGV>)
        Passage@(<INTV>,<STRINGV>,<INTV>,<STRINGV>)
        ExportPsg
```

Evaluation

TmNoteset:

$$\frac{\mathrm{t1} \rightarrow \mathrm{t1}'}{\mathrm{MakeNoteset\ t1\ t2} \rightarrow \mathrm{MakeNoteset\ t1}'\ \mathrm{t2}}$$

$$\frac{\mathrm{t2} \rightarrow \mathrm{t2}'}{\mathrm{MakeNoteset\ v1\ t2} \rightarrow \mathrm{MakeNoteset\ v1\ t2}'}$$

TmPhrase:

$$\frac{\mathrm{t1} \rightarrow \mathrm{t1}'}{\mathrm{MakePhrase\ t1\ t2} \rightarrow \mathrm{MakePhrase\ t1}'\ \mathrm{t2}}$$

$$\frac{\mathrm{t2} \rightarrow \mathrm{t2}'}{\mathrm{MakePhrase\ v1\ t2} \rightarrow \mathrm{MakePhrase\ v1\ t2}'}$$

TmSegment:

$$\frac{\mathrm{t1} \rightarrow \mathrm{t1}'}{\mathrm{MakeSegment\ t1}(\mathrm{A}, \mathrm{B}) \rightarrow \mathrm{MakeSegment\ t1}'(\mathrm{A}, \mathrm{B})}$$

Evaluation

TmPassage:

$$\frac{\text{t1} \rightarrow \text{t1}'}{\text{MakePassage t1 t2} \rightarrow \text{MakePassage t1}'\text{ t2}}$$

$$\frac{\text{t2} \rightarrow \text{t2}'}{\text{MakePassage v1 t2} \rightarrow \text{MakePassage v1 t2}'}$$

TmExportMsg:

$$\frac{\text{t1} \rightarrow \text{t1}'}{\text{ExportPsg t1 => t2} \rightarrow \text{ExportPsg t1}'\text{ => t2}}$$

$$\frac{\text{t2} \rightarrow \text{t2}'}{\text{ExportPsg v1 => t2} \rightarrow \text{ExportPsg v1 => t2}'}$$

Typing

TyNote:

$$\frac{\{\#\mathrm{A}|\mathrm{B}|\mathrm{C}|\mathrm{D}\#\}\ \textit{type checks and is rank N}}{\{\#\mathrm{A}|\mathrm{B}|\mathrm{C}|\mathrm{D}\#\}\text{: Note@N}}$$

TyNoteset:

$$\frac{\Gamma \vdash \text{t1: (Note@M} \ \ \| \ \ \text{Noteset@M), t2: (Note@N} \ \ \| \ \ \text{Noteset@N) and rank matches}}{\text{MakeNoteset t1 t2: Noteset@M}}$$

TyPhrase:

$$\frac{\begin{array}{c}\Gamma \vdash \text{t1:}\left(\text{Note@M1} \ \ \| \ \ \text{Noteset@M1} \ \ \| \ \text{Phrase@(M1, M2)}\right), \\ \text{t2:}\left(\text{Note@N2} \ \ \| \ \ \text{Noteset@N2} \ \ \| \ \text{Phrase@(N1, N2)}\right) \\ \text{and rank matches}\end{array}}{\text{MakePhrase t1 t2: Phrase@(M1, N2)}}$$

Typing

TySegment:

$$\frac{\Gamma \vdash \text{t: Phrase@(1,1)}}{\text{MakeSegment t(P, Q): Segment(P, Q)}}$$

TyPassage:

$$\frac{\begin{array}{c}\Gamma \vdash \text{t1: (Segment@(P11, Q11) } \parallel \text{ Passage(P11, Q11, P12, Q12),} \\ \text{t2: }\big(\text{Segment@(P22, Q22) } \parallel \text{ Passage(P21, Q21, P22, Q22)}\big) \\ \text{and mode matches}\end{array}}{\text{MakePhrase t1 t2: Passage(P11, Q11, P22, Q22)}}$$

TyExportMsg:

$$\frac{\Gamma \vdash \text{t: Passage, f: String}}{\text{ExportMsg t => f: ExportMsg}}$$

Outline

- Intuition & TODO
- Design Patterns
- Formal Definition
- Some Proofs
- Evaluation

Type Checking Termination

- A music tree is a binary tree
- Type checking for any known term terminates
- **There are recursive types, so we can build an infinite binary tree, and the type checking can diverge**
- The algorithm terminates when the number of nodes is limited
- where the tree represents valid music :)

Type Safety - Progress

`TmNoteset:`

$$\frac{\mathrm{t1} \rightarrow \mathrm{t1}'}{\mathrm{MakeNoteset\ t1\ t2} \rightarrow \mathrm{MakeNoteset\ t1'\ t2}}$$

$$\frac{\mathrm{t2} \rightarrow \mathrm{t2}'}{\mathrm{MakeNoteset\ v1\ t2} \rightarrow \mathrm{MakeNoteset\ v1\ t2'}}$$

1. t1 is a ((term and t1 -> t1’, 1 step) || (value -> case 2))
2. t2 is a ((term and t2 -> t2’, 1 step) || (value, a value))

Other cases are similar

Type Safety - Preservation

TmNoteset:

$$\frac{\mathrm{t1} \rightarrow \mathrm{t1}'}{\mathrm{MakeNoteset\ t1\ t2} \rightarrow \mathrm{MakeNoteset\ t1'\ t2}}$$

$$\frac{\mathrm{t2} \rightarrow \mathrm{t2}'}{\mathrm{MakeNoteset\ v1\ t2} \rightarrow \mathrm{MakeNoteset\ v1\ t2'}}$$

TyNoteset:

$$\frac{\Gamma \vdash \mathrm{t1}: (\mathrm{Note@M} \ \| \ \mathrm{Noteset@M}), \mathrm{t2}: (\mathrm{Note@N} \ \| \ \mathrm{Noteset@N}) \text{ and rank matches}}{\mathrm{MakeNoteset\ t1\ t2}: \mathrm{Noteset@M}}$$

1. t1 and t2 is TyNote or TyNoteset. TyNoteset applies.
2. v1 and t2 is TyNote or TyNoteset. TyNoteset applies.

Other cases are similar

Outline

- Intuition & TODO
- Design Patterns
- Formal Definition
- Some Proofs
- Evaluation