Functional Generation of Harmony and Melody

José Pedro Magalhães   Hendrik Vincent Koops

Functional Art, Music, Modeling and Design 2014

6 September 2014

An extremely useful model of musical harmony

A couple of years ago I worked on a modelling musical harmony as a GADT in Haskell (together with W. Bas de Haas). From this model came:

An extremely useful model of musical harmony

A couple of years ago I worked on a modelling musical harmony as a GADT in Haskell (together with W. Bas de Haas). From this model came:

- A pretty-printer that shows the tree structure of a sequence of chords (ICFP 2011);

An extremely useful model of musical harmony

A couple of years ago I worked on a modelling musical harmony as a GADT in Haskell (together with W. Bas de Haas). From this model came:

- A pretty-printer that shows the tree structure of a sequence of chords (ICFP 2011);
- A system for finding cover songs based on harmonic similarity (ISMIR 2011);

An extremely useful model of musical harmony

A couple of years ago I worked on a modelling musical harmony as a GADT in Haskell (together with W. Bas de Haas). From this model came:

- A pretty-printer that shows the tree structure of a sequence of chords (ICFP 2011);
- A system for finding cover songs based on harmonic similarity (ISMIR 2011);
- An improved chord recogniser from audio that is aware of the rules of musical harmony (ISMIR 2012);

An extremely useful model of musical harmony

A couple of years ago I worked on a modelling musical harmony as a GADT in Haskell (together with W. Bas de Haas). From this model came:

- A pretty-printer that shows the tree structure of a sequence of chords (ICFP 2011);
- A system for finding cover songs based on harmonic similarity (ISMIR 2011);
- An improved chord recogniser from audio that is aware of the rules of musical harmony (ISMIR 2012);
- A company (`http://chordify.net`);

An extremely useful model of musical harmony

A couple of years ago I worked on a modelling musical harmony as a GADT in Haskell (together with W. Bas de Haas). From this model came:

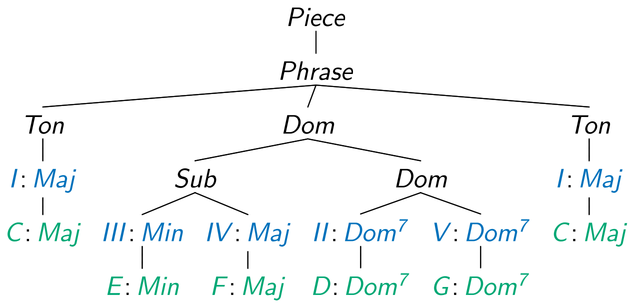
- A pretty-printer that shows the tree structure of a sequence of chords (ICFP 2011);
- A system for finding cover songs based on harmonic similarity (ISMIR 2011);
- An improved chord recogniser from audio that is aware of the rules of musical harmony (ISMIR 2012);
- A company (`http://chordify.net`);
- A system for automatic melody harmonisation (FARM 2013);

An extremely useful model of musical harmony

A couple of years ago I worked on a modelling musical harmony as a GADT in Haskell (together with W. Bas de Haas). From this model came:

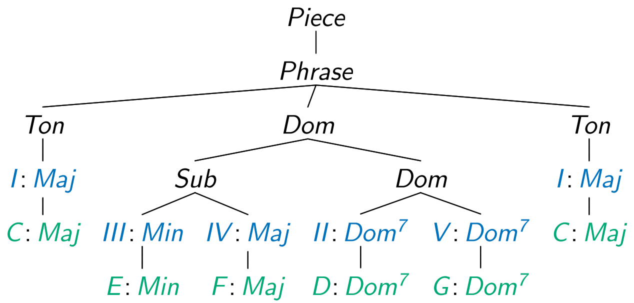
- A pretty-printer that shows the tree structure of a sequence of chords (ICFP 2011);
- A system for finding cover songs based on harmonic similarity (ISMIR 2011);
- An improved chord recogniser from audio that is aware of the rules of musical harmony (ISMIR 2012);
- A company (`http://chordify.net`);
- A system for automatic melody harmonisation (FARM 2013);
- FComp: a system for automatic generation of harmony and accompanying melody (this talk).

An example: visualising harmonic structure



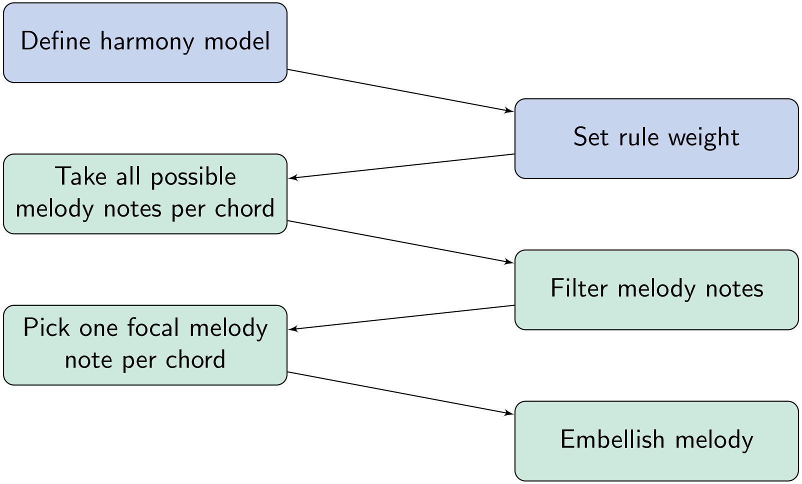
You can see this tree as having been produced by taking the chords in green as input. . .

An example: generating harmonic structure



You can see this tree as having been produced by taking the chords in green as input. . . or the chords might have been dictated by the structure!

System structure


Define harmony model
Set rule weight
Take all possible melody notes per chord
Filter melody notes
Pick one focal melody note per chord
Embellish melody

A functional model of harmony

$$\mathit{Piece}_{\mathfrak{M}} \rightarrow [\mathit{Phrase}_{\mathfrak{M}}] \qquad (\mathfrak{M} \in \{\mathsf{Maj}, \mathsf{Min}\})$$

A functional model of harmony

$$Piece_{\mathfrak{M}} \rightarrow [Phrase_{\mathfrak{M}}] \qquad (\mathfrak{M} \in \{\mathsf{Maj}, \mathsf{Min}\})$$

$$\begin{aligned} Phrase_{\mathfrak{M}} \rightarrow\ & Ton_{\mathfrak{M}}\ Dom_{\mathfrak{M}}\ Ton_{\mathfrak{M}} \\ \mid\ & Dom_{\mathfrak{M}}\ Ton_{\mathfrak{M}} \end{aligned}$$

A functional model of harmony

$$\mathit{Piece}_{\mathfrak{M}} \rightarrow [\mathit{Phrase}_{\mathfrak{M}}] \qquad (\mathfrak{M} \in \{\mathsf{Maj}, \mathsf{Min}\})$$

$$\begin{aligned}\mathit{Phrase}_{\mathfrak{M}} &\rightarrow \mathit{Ton}_{\mathfrak{M}}\ \mathit{Dom}_{\mathfrak{M}}\ \mathit{Ton}_{\mathfrak{M}} \\ &\mid \qquad\quad \mathit{Dom}_{\mathfrak{M}}\ \mathit{Ton}_{\mathfrak{M}}\end{aligned}$$

$$\begin{aligned}\mathit{Ton}_{\mathsf{Maj}} &\rightarrow I_{\mathsf{Maj}} \\ \mathit{Ton}_{\mathsf{Min}} &\rightarrow I^{m}_{\mathsf{Min}}\end{aligned}$$

A functional model of harmony

$$Piece_{\mathfrak{M}} \rightarrow [Phrase_{\mathfrak{M}}] \qquad (\mathfrak{M} \in \{\mathsf{Maj}, \mathsf{Min}\})$$

$$\begin{aligned} Phrase_{\mathfrak{M}} &\rightarrow Ton_{\mathfrak{M}}\ Dom_{\mathfrak{M}}\ Ton_{\mathfrak{M}} \\ &\mid \phantom{Ton_{\mathfrak{M}}}\ Dom_{\mathfrak{M}}\ Ton_{\mathfrak{M}} \end{aligned}$$

$$\begin{aligned} Ton_{\mathsf{Maj}} &\rightarrow I_{\mathsf{Maj}} \\ Ton_{\mathsf{Min}} &\rightarrow I^{m}_{\mathsf{Min}} \end{aligned}$$

$$\begin{aligned} Dom_{\mathfrak{M}} &\rightarrow V^{7}_{\mathfrak{M}} \\ &\mid V_{\mathfrak{M}} \\ &\mid VII^{0}_{\mathfrak{M}} \\ &\mid Sub_{\mathfrak{M}}\ Dom_{\mathfrak{M}} \\ &\mid II^{7}_{\mathfrak{M}}\ V^{7}_{\mathfrak{M}} \end{aligned}$$

A functional model of harmony

$$Piece_{\mathfrak{M}} \rightarrow [Phrase_{\mathfrak{M}}] \qquad (\mathfrak{M} \in \{\mathsf{Maj}, \mathsf{Min}\})$$

$$\begin{aligned} Phrase_{\mathfrak{M}} &\rightarrow Ton_{\mathfrak{M}}\ Dom_{\mathfrak{M}}\ Ton_{\mathfrak{M}} \\ &\mid \quad Dom_{\mathfrak{M}}\ Ton_{\mathfrak{M}} \end{aligned}$$

$$\begin{aligned} Ton_{\mathsf{Maj}} &\rightarrow I_{\mathsf{Maj}} \\ Ton_{\mathsf{Min}} &\rightarrow I^{m}_{\mathsf{Min}} \end{aligned}$$

$$\begin{aligned} Dom_{\mathfrak{M}} &\rightarrow V^{7}_{\mathfrak{M}} \\ &\mid V_{\mathfrak{M}} \\ &\mid VII^{0}_{\mathfrak{M}} \\ &\mid Sub_{\mathfrak{M}}\ Dom_{\mathfrak{M}} \\ &\mid II^{7}_{\mathfrak{M}}\ V^{7}_{\mathfrak{M}} \end{aligned}$$

$$\begin{aligned} Sub_{\mathsf{Maj}} &\rightarrow II^{m}_{\mathsf{Maj}} \\ &\mid IV_{\mathsf{Maj}} \\ &\mid III^{m}_{\mathsf{Maj}}\ IV_{\mathsf{Maj}} \\ Sub_{\mathsf{Min}} &\rightarrow IV^{m}_{\mathsf{Min}} \end{aligned}$$

A functional model of harmony

$$Piece_{\mathfrak{M}} \to [Phrase_{\mathfrak{M}}] \qquad (\mathfrak{M} \in \{\mathsf{Maj}, \mathsf{Min}\})$$

$$\begin{aligned} Phrase_{\mathfrak{M}} &\to Ton_{\mathfrak{M}}\ Dom_{\mathfrak{M}}\ Ton_{\mathfrak{M}} \\ &\mid\ Dom_{\mathfrak{M}}\ Ton_{\mathfrak{M}} \end{aligned}$$

$$\begin{aligned} Ton_{\mathsf{Maj}} &\to I_{\mathsf{Maj}} \\ Ton_{\mathsf{Min}} &\to I^{m}_{\mathsf{Min}} \end{aligned}$$

$$\begin{aligned} Dom_{\mathfrak{M}} &\to V^{7}_{\mathfrak{M}} \\ &\mid\ V_{\mathfrak{M}} \\ &\mid\ VII^{0}_{\mathfrak{M}} \\ &\mid\ Sub_{\mathfrak{M}}\ Dom_{\mathfrak{M}} \\ &\mid\ II^{7}_{\mathfrak{M}}\ V^{7}_{\mathfrak{M}} \end{aligned}$$

$$\begin{aligned} Sub_{\mathsf{Maj}} &\to II^{m}_{\mathsf{Maj}} \\ &\mid\ IV_{\mathsf{Maj}} \\ &\mid\ III^{m}_{\mathsf{Maj}}\ IV_{\mathsf{Maj}} \\ Sub_{\mathsf{Min}} &\to IV^{m}_{\mathsf{Min}} \end{aligned}$$

$$\begin{aligned} I_{\mathsf{Maj}} &\to C : Maj \\ I^{m}_{\mathsf{Min}} &\to C : Min \\ V^{7}_{\mathfrak{M}} &\to G : Dom^{7} \\ VII^{0}_{\mathfrak{M}} &\to B : Dim \end{aligned}$$

A functional model of harmony

$$Piece_{\mathfrak{M}} \rightarrow [Phrase_{\mathfrak{M}}] \qquad (\mathfrak{M} \in \{\mathsf{Maj}, \mathsf{Min}\})$$

$$\begin{aligned} Phrase_{\mathfrak{M}} &\rightarrow Ton_{\mathfrak{M}}\ Dom_{\mathfrak{M}}\ Ton_{\mathfrak{M}} \\ &\mid \quad Dom_{\mathfrak{M}}\ Ton_{\mathfrak{M}} \end{aligned}$$

$$\begin{aligned} Ton_{\mathsf{Maj}} &\rightarrow I_{\mathsf{Maj}} \\ Ton_{\mathsf{Min}} &\rightarrow I^{m}_{\mathsf{Min}} \end{aligned}$$

$$\begin{aligned} Dom_{\mathfrak{M}} &\rightarrow V^{7}_{\mathfrak{M}} \\ &\mid V_{\mathfrak{M}} \\ &\mid VII^{0}_{\mathfrak{M}} \\ &\mid Sub_{\mathfrak{M}}\ Dom_{\mathfrak{M}} \\ &\mid II^{7}_{\mathfrak{M}}\ V^{7}_{\mathfrak{M}} \end{aligned}$$

$$\begin{aligned} Sub_{\mathsf{Maj}} &\rightarrow II^{m}_{\mathsf{Maj}} \\ &\mid IV_{\mathsf{Maj}} \\ &\mid III^{m}_{\mathsf{Maj}}\ IV_{\mathsf{Maj}} \\ Sub_{\mathsf{Min}} &\rightarrow IV^{m}_{\mathsf{Min}} \end{aligned}$$

$$\begin{aligned} I_{\mathsf{Maj}} &\rightarrow C : Maj \\ I^{m}_{\mathsf{Min}} &\rightarrow C : Min \\ V^{7}_{\mathfrak{M}} &\rightarrow G : Dom^{7} \\ VII^{0}_{\mathfrak{M}} &\rightarrow B : Dim \end{aligned}$$

Simple, but enough for now, *and easy to extend*.

Now in Haskell—I

A GADT encoding musical harmony:

$\textbf{data}\ \mathit{Mode} = \mathit{Maj}_{\mathit{Mode}} \mid \mathit{Min}_{\mathit{Mode}}$

$\textbf{data}\ \mathit{Piece} = \forall \mu :: \mathit{Mode}.\mathit{Piece}\ [\ \mathit{Phrase}\ \mu\ ]$

Now in Haskell—I

A GADT encoding musical harmony:

$$\textbf{data}\ \mathit{Mode} = \mathit{Maj}_{\mathit{Mode}} \mid \mathit{Min}_{\mathit{Mode}}$$

$$\textbf{data}\ \mathit{Piece} = \forall \mu :: \mathit{Mode}.\mathit{Piece}\ [\,\mathit{Phrase}\ \mu\,]$$

$$\begin{aligned}
&\textbf{data}\ \mathit{Phrase}\ (\mu :: \mathit{Mode})\ \textbf{where}\\
&\quad \mathit{Phrase}_{IVI} :: \mathit{Ton}\ \mu \to \mathit{Dom}\ \mu \to \mathit{Ton}\ \mu \to \mathit{Phrase}\ \mu\\
&\quad \mathit{Phrase}_{VI}\ \ :: \qquad\qquad\ \ \mathit{Dom}\ \mu \to \mathit{Ton}\ \mu \to \mathit{Phrase}\ \mu
\end{aligned}$$

Now in Haskell—I

A GADT encoding musical harmony:

$$\textbf{data}\ \mathit{Mode} = \mathit{Maj}_{\mathit{Mode}} \mid \mathit{Min}_{\mathit{Mode}}$$

$$\textbf{data}\ \mathit{Piece} = \forall \mu :: \mathit{Mode}.\mathit{Piece}\ [\,\mathit{Phrase}\ \mu\,]$$

$$\begin{array}{l}
\textbf{data}\ \mathit{Phrase}\ (\mu :: \mathit{Mode})\ \textbf{where} \\
\quad \mathit{Phrase}_{\mathit{IVI}} :: \mathit{Ton}\ \mu \to \mathit{Dom}\ \mu \to \mathit{Ton}\ \mu \to \mathit{Phrase}\ \mu \\
\quad \mathit{Phrase}_{\mathit{VI}}\ \ :: \qquad\qquad\ \mathit{Dom}\ \mu \to \mathit{Ton}\ \mu \to \mathit{Phrase}\ \mu
\end{array}$$

$$\begin{array}{l}
\textbf{data}\ \mathit{Ton}\ (\mu :: \mathit{Mode})\ \textbf{where} \\
\quad \mathit{Ton}_{\mathit{Maj}} :: \mathit{SD}\ \mathit{I}\ \mathit{Maj} \to \mathit{Ton}\ \mathit{Maj}_{\mathit{Mode}} \\
\quad \mathit{Ton}_{\mathit{Min}} :: \mathit{SD}\ \mathit{I}\ \mathit{Min} \to \mathit{Ton}\ \mathit{Min}_{\mathit{Mode}}
\end{array}$$

Now in Haskell—I

A GADT encoding musical harmony:

```
data Mode = Maj_Mode | Min_Mode
data Piece = ∀μ :: Mode.Piece [Phrase μ]

data Phrase (μ :: Mode) where
  Phrase_IVI :: Ton μ → Dom μ → Ton μ → Phrase μ
  Phrase_VI  ::         Dom μ → Ton μ → Phrase μ

data Ton (μ :: Mode) where
  Ton_Maj :: SD I Maj → Ton Maj_Mode
  Ton_Min :: SD I Min → Ton Min_Mode

data Dom (μ :: Mode) where
  Dom_1 :: SD V   Dom^7 → Dom μ
  Dom_2 :: SD V   Maj   → Dom μ
  Dom_3 :: SD VII Dim   → Dom μ
  Dom_4 :: SDom μ → Dom μ → Dom μ
  Dom_5 :: SD II Dom^7 → SD V Dom^7 → Dom μ
```

Now in Haskell—II

Scale degrees are the leaves of our hierarchical structure:

$$\begin{aligned}
&\textbf{data}\ \mathit{DiatonicDegree} = \mathit{I} \mid \mathit{II} \mid \mathit{III} \mid \mathit{IV} \mid \mathit{V} \mid \mathit{VI} \mid \mathit{VII}\\
&\textbf{data}\ \mathit{Quality} \qquad\quad\ = \mathit{Maj} \mid \mathit{Min} \mid \mathit{Dom}^{7} \mid \mathit{Dim}
\end{aligned}$$

$$\begin{aligned}
&\textbf{data}\ \mathit{SD}\ (\delta :: \mathit{DiatonicDegree})\ (\gamma :: \mathit{Quality})\ \textbf{where}\\
&\quad \mathit{SurfaceChord} :: \mathit{ChordDegree} \to \mathit{SD}\ \delta\ \gamma
\end{aligned}$$

Generating harmony

Now that we have a datatype representing harmony sequences, how do we generate a sequence of chords?

Generating harmony

Now that we have a datatype representing harmony sequences, how do we generate a sequence of chords?

QuickCheck! We give *Arbitrary* instances for each of the datatypes in our model.

Generating harmony

Now that we have a datatype representing harmony sequences, how do we generate a sequence of chords?

QuickCheck! We give *Arbitrary* instances for each of the datatypes in our model.

. . . but we don't want to do this by hand, for every datatype, and to have to adapt the instances every time we change the model. . . so we use *generic programming*:

Generating harmony

Now that we have a datatype representing harmony sequences, how do we generate a sequence of chords?

QuickCheck! We give *Arbitrary* instances for each of the datatypes in our model.

. . . but we don't want to do this by hand, for every datatype, and to have to adapt the instances every time we change the model. . . so we use *generic programming*:

$$\mathit{gen} :: (\mathit{Representable}\ \alpha, \mathit{Generate}\ (\mathit{Rep}\ \alpha)) \Rightarrow \mathit{Gen}\ \alpha$$

Generating harmony

Now that we have a datatype representing harmony sequences, how do we generate a sequence of chords?

QuickCheck! We give *Arbitrary* instances for each of the datatypes in our model.

. . . but we don't want to do this by hand, for every datatype, and to have to adapt the instances every time we change the model. . . so we use *generic programming*:

$$\mathit{gen} :: (\mathit{Representable}\ \alpha, \mathit{Generate}\ (\mathit{Rep}\ \alpha)) \Rightarrow [(\mathit{String}, \mathit{Int})] \rightarrow \mathit{Gen}\ \alpha$$

Examples of harmony generation—I

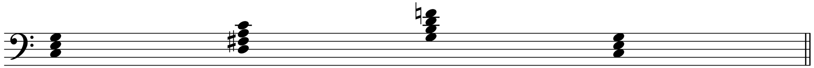
$$
\begin{aligned}
&\mathit{testGen} :: \mathit{Gen}\ (\mathit{Phrase}\ \mathit{Maj}_{\mathit{Mode}})\\
&\mathit{testGen} = \mathit{gen}\ [(\texttt{"Dom4"}, 3), (\texttt{"Dom5"}, 4)]\\
&\mathit{example} :: \mathit{IO}\ ()\\
&\mathit{example} = \mathbf{let}\ k = \mathit{Key}\ (\mathit{Note}\ \natural\ C)\ \mathit{Maj}_{\mathit{Mode}}\\
&\qquad\qquad\quad \mathbf{in}\ \mathit{sample}'\ \mathit{testGen} \ggg \mathit{mapM}\_\ (\mathit{printOnKey}\ k)\\
&\mathit{printOnKey} :: \mathit{Key} \rightarrow \mathit{Phrase}\ \mathit{Maj}_{\mathit{Mode}} \rightarrow \mathit{IO}\ \mathit{String}
\end{aligned}
$$

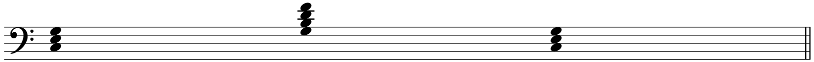
Examples of harmony generation—I

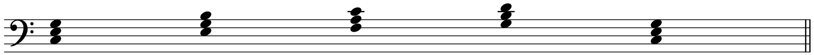
$$\begin{aligned}
&\mathit{testGen} :: \mathit{Gen}\ (\mathit{Phrase}\ \mathit{Maj}_{\mathit{Mode}})\\
&\mathit{testGen} = \mathit{gen}\ [(\texttt{"Dom4"},3),(\texttt{"Dom5"},4)]\\
&\mathit{example} :: \mathit{IO}\ ()\\
&\mathit{example} = \textbf{let}\ k = \mathit{Key}\ (\mathit{Note}\ \natural\ C)\ \mathit{Maj}_{\mathit{Mode}}\\
&\qquad\qquad \textbf{in}\ \mathit{sample}'\ \mathit{testGen} \ggg \mathit{mapM}_{\_}\ (\mathit{printOnKey}\ k)\\
&\mathit{printOnKey} :: \mathit{Key} \to \mathit{Phrase}\ \mathit{Maj}_{\mathit{Mode}} \to \mathit{IO}\ \mathit{String}
\end{aligned}$$

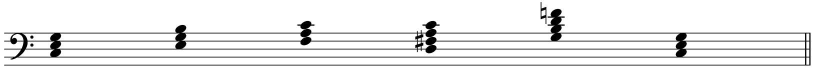
$$\begin{aligned}
&> \mathit{example}\\
&[C{:}\ \mathit{Maj}, D{:}\ \mathit{Dom}^7, G{:}\ \mathit{Dom}^7, C{:}\ \mathit{Maj}]\\
&[C{:}\ \mathit{Maj}, G{:}\ \mathit{Dom}^7, C{:}\ \mathit{Maj}]\\
&[C{:}\ \mathit{Maj}, E{:}\ \mathit{Min}, F{:}\ \mathit{Maj}, G{:}\ \mathit{Maj}, C{:}\ \mathit{Maj}]\\
&[C{:}\ \mathit{Maj}, E{:}\ \mathit{Min}, F{:}\ \mathit{Maj}, D{:}\ \mathit{Dom}^7, G{:}\ \mathit{Dom}^7, C{:}\ \mathit{Maj}]\\
&[C{:}\ \mathit{Maj}, D{:}\ \mathit{Min}, E{:}\ \mathit{Min}, F{:}\ \mathit{Maj}, D{:}\ \mathit{Dom}^7, G{:}\ \mathit{Dom}^7, C{:}\ \mathit{Maj}]
\end{aligned}$$

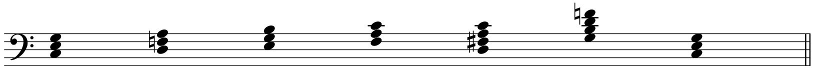
Examples of harmony generation—II











Generating a melody for a given harmony

We then generate a melody in 4 steps:

Generating a melody for a given harmony

We then generate a melody in 4 steps:

1. Generate a list of candidate melody notes per chord;

Generating a melody for a given harmony

We then generate a melody in 4 steps:

1. Generate a list of candidate melody notes per chord;
2. Refine the candidates by filtering out obviously bad candidates;

Generating a melody for a given harmony

We then generate a melody in 4 steps:

1. Generate a list of candidate melody notes per chord;
2. Refine the candidates by filtering out obviously bad candidates;
3. Pick one focal candidate melody note per chord;

Generating a melody for a given harmony

We then generate a melody in 4 steps:

1. Generate a list of candidate melody notes per chord;
2. Refine the candidates by filtering out obviously bad candidates;
3. Pick one focal candidate melody note per chord;
4. Embellish the candidate notes to produce a final melody.

Generating a melody for a given harmony

We then generate a melody in 4 steps:

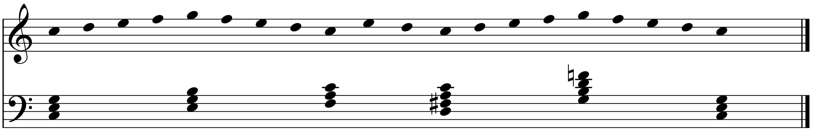
1. Generate a list of candidate melody notes per chord;
2. Refine the candidates by filtering out obviously bad candidates;
3. Pick one focal candidate melody note per chord;
4. Embellish the candidate notes to produce a final melody.

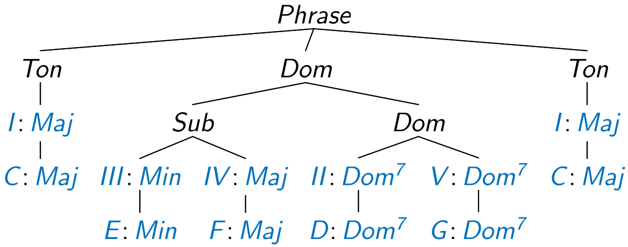
These four steps combine naturally using plain monadic bind:

$$
\begin{aligned}
&\mathit{melody} :: \mathit{Key} \rightarrow \mathit{State\ MyState\ Song}\\
&\mathit{melody}\ k\ =\ \mathit{genCandidates} \ggg\!= \mathit{refine} \ggg\!= \mathit{pickOne} \ggg\!= \mathit{embellish}\\
&\qquad\qquad \ggg\!= \mathit{return} \circ \mathit{Song}\ k
\end{aligned}
$$

More details in the paper!

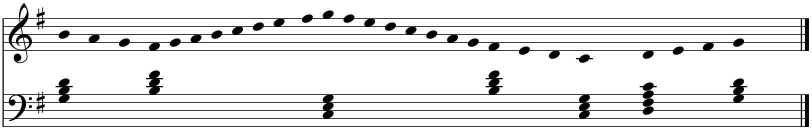
Example I

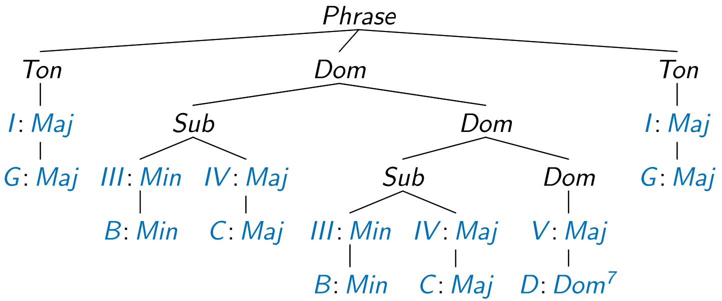




Phrase
Ton
Dom
Ton
I: Maj
Sub
Dom
I: Maj
C: Maj
III: Min
IV: Maj
II: Dom7
V: Dom7
C: Maj
E: Min
F: Maj
D: Dom7
G: Dom7

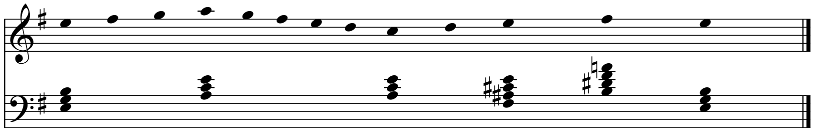
Example II

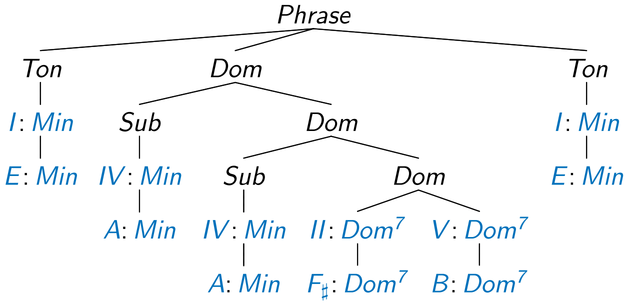




Phrase
Ton
Dom
Ton
I: Maj
Sub
Dom
I: Maj
G: Maj
III: Min
IV: Maj
Sub
Dom
G: Maj
B: Min
C: Maj
III: Min
IV: Maj
V: Maj
B: Min
C: Maj
D: Dom7

Example III





Phrase
Ton
Dom
Ton
I: Min
Sub
Dom
I: Min
E: Min
IV: Min
Sub
Dom
E: Min
A: Min
IV: Min
II: Dom7
V: Dom7
A: Min
F♯: Dom7
B: Dom7

Conclusion

FComp: a *simple* and *easy to understand and improve* functional system for automatic generation of harmony and accompanying melody.

Conclusion

FComp: a *simple* and *easy to understand and improve* functional system for automatic generation of harmony and accompanying melody.

Lots of room for improvement:

- Voice leading and counterpoint
- Handle repetition
- Improve embellishment
- Rhythm, form, instrumentation, dynamics

Conclusion

FComp: a *simple* and *easy to understand and improve* functional system for automatic generation of harmony and accompanying melody.

Lots of room for improvement:

- Voice leading and counterpoint
- Handle repetition
- Improve embellishment
- Rhythm, form, instrumentation, dynamics

Thank you for your attention!