Advanced Functional Programming for Fun and Profit

José Pedro Magalhães

Department of Computer Science, University of Oxford
http://dreixel.net

January 10, 2014
Amsterdam, The Netherlands

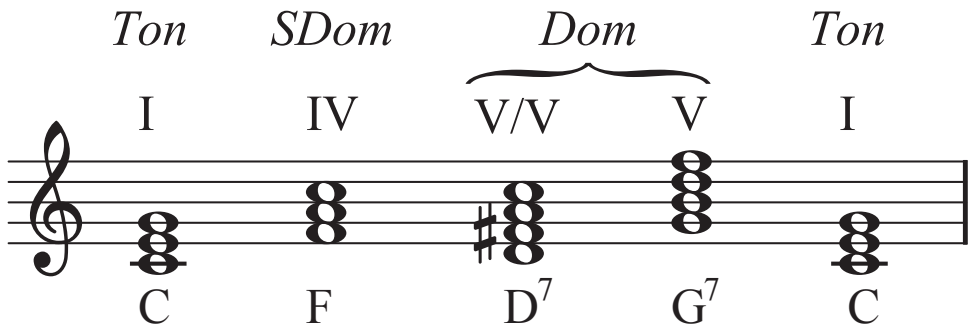
Three years ago...

@ Dutch Functional Programming day 2011, University of Twente

Introduction

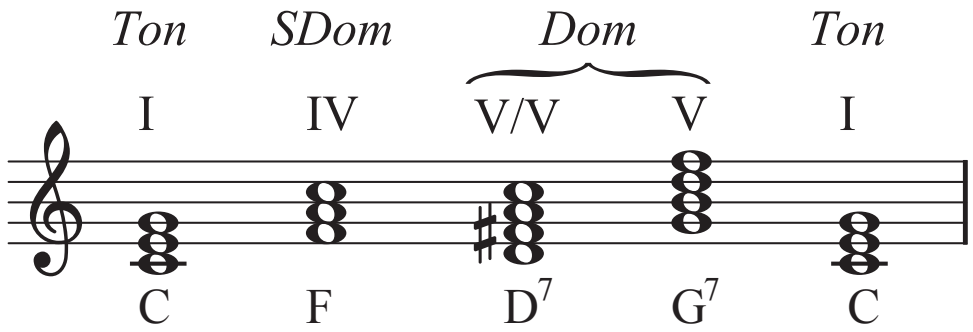
- Modelling musical harmony using Haskell
- Applications of a model of harmony:
  - Musical analysis
  - Finding cover songs
  - Generating chords for melodies
  - Correcting errors in chord extraction from audio sources
- Chordify—a web-based music player with chord recognition

What is harmony?



- Harmony arises when at least two notes sound at the same time
- Harmony induces tension and release patterns, that can be described by music theory and music cognition
- The internal structure of the chord has a large influence on the consonance or dissonance of a chord
- The surrounding context also has a large influence

What is harmony?



- Harmony arises when at least two notes sound at the same time
- Harmony induces tension and release patterns, that can be described by music theory and music cognition
- The internal structure of the chord has a large influence on the consonance or dissonance of a chord
- The surrounding context also has a large influence

Demo: how harmony affects melody

Why are harmony models useful?

Having a model for musical harmony allows us to automatically determine the functional meaning of chords in the tonal context.
The model determines which chords "fit" on a particular moment in a song.

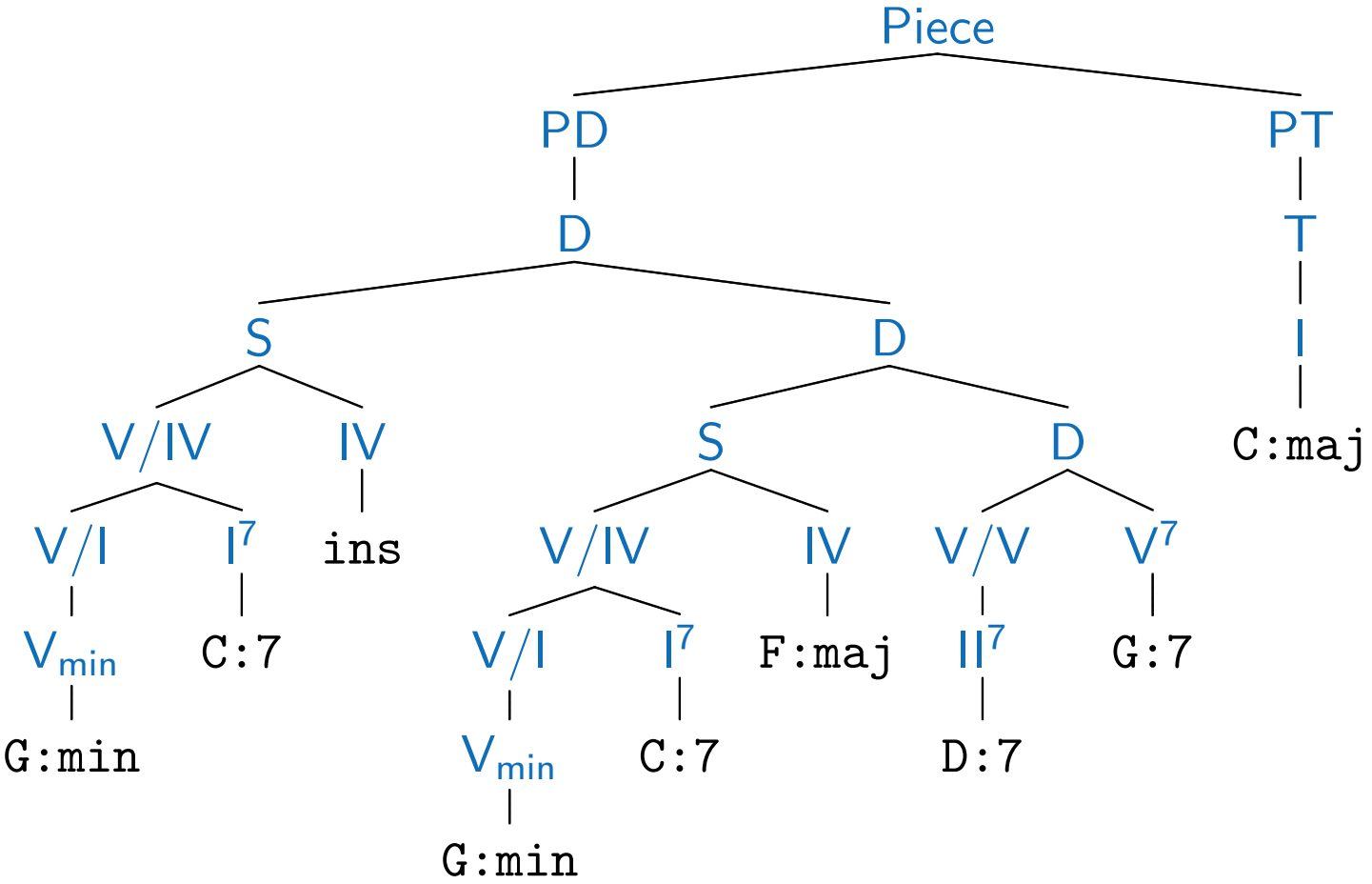
Why are harmony models useful?

Having a model for musical harmony allows us to automatically determine the functional meaning of chords in the tonal context. The model determines which chords "fit" on a particular moment in a song. This is useful for:

- Musical information retrieval (find songs similar to a given song)
- Audio and score recognition (improving recognition by knowing which chords are more likely to appear)
- Music generation (create sequences of chords that conform to the model)

Application: harmony analysis

Parsing the sequence G_{min} C^7 G_{min} C^7 F_{Maj} D^7 G^7 C_{Maj}:

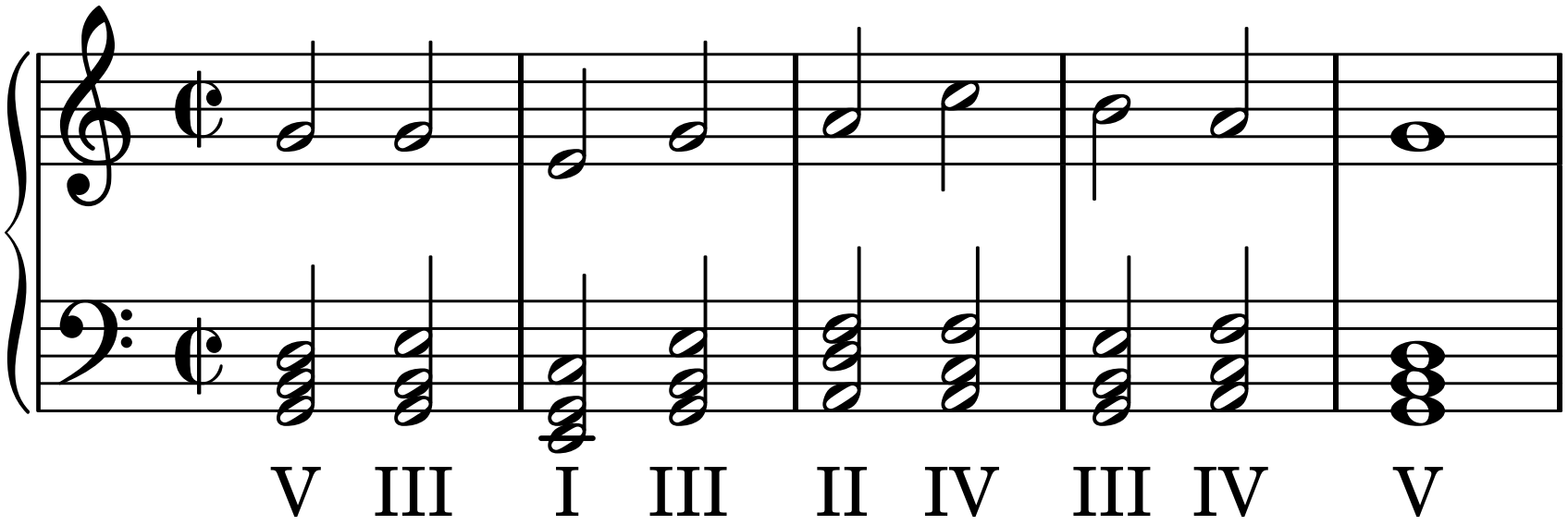


Application: harmonic similarity

- A practical application of a harmony model is to estimate harmonic similarity between songs
- The more similar the trees, the more similar the harmony
- We don't want to write a diff algorithm for our complicated model; we get it automatically by using a *generic diff*
- The generic diff is a type-safe tree-diff algorithm, part of a student's MSc work at Utrecht University
- Generic, thus working for any model, and independent of changes to the model

Application: automatic harmonisation of melodies

Another practical application of a harmony model is to help selecting good harmonisations (chord sequences) for a given melody:



We generate candidate chord sequences, parse them with the harmony model, and select the one with the least errors.

Application: chord recognition

Yet another practical application of a harmony model is to improve chord recognition from audio sources.

Chord candidates		0.92 C	0.96 Em
		0.94 Gm	0.97 C
	1.00 C	1.00 G	1.00 Em
Beat number	1	2	3

How to pick the right chord from the chord candidate list? Ask the harmony model which one fits best.

Demo: Chordify

Demo:

chordify®

`http://chordify.net`

Chordify: architecture

- Frontend
  - Reads user input, such as YouTube/Soundcloud links, or files
  - Extracts audio
  - Calls the backend to obtain the chords for the audio
  - Displays the result to the user
  - Implements a queueing system, and library functionality
  - Uses PHP, JavaScript, MongoDB

Chordify: architecture

- Frontend
  - Reads user input, such as YouTube/Soundcloud links, or files
  - Extracts audio
  - Calls the backend to obtain the chords for the audio
  - Displays the result to the user
  - Implements a queueing system, and library functionality
  - Uses PHP, JavaScript, MongoDB
- Backend
  - Takes an audio file as input, analyses it, extracts the chords
  - The chord extraction code uses GADTs, type families, generic programming (see the harmtrace package on Hackage)
  - Performs PDF and MIDI export (using LilyPond)
  - Uses Haskell, SoX, sonic annotator, and is mostly open source

Chordify: architecture

- Frontend
  - Reads user input, such as YouTube/Soundcloud links, or files
  - Extracts audio
  - Calls the backend to obtain the chords for the audio
  - Displays the result to the user
  - Implements a queueing system, and library functionality
  - Uses PHP, JavaScript, MongoDB
- Backend
  - Takes an audio file as input, analyses it, extracts the chords
  - The chord extraction code uses GADTs, type families, generic programming (see the harmtrace package on Hackage)
  - Performs PDF and MIDI export (using LilyPond)
  - Uses Haskell, SoX, sonic annotator, and is mostly open source
  - Will replace the frontend queuing system (using Happstack)

Summary

Musical modelling with Haskell:

- A model for musical harmony as a Haskell datatype
- Makes use of several advanced functional programming techniques, such as generic programming, GADTs, and type families
- When chords do not fit the model: error correction
- Harmonising melodies
- Recognising harmony from audio sources

Play with it!

`http://hackage.haskell.org/package/HarmTrace`

chordify®

`http://chordify.net`