Musical Ornaments

John Leo

Halfaya Research

May 14, 2018

Outline

- A look toward the future
- Music Tools
- Equivalences and Ornaments

Sources:

- `https://github.com/halfaya/MusicTools`

Robert Harper

> *Eventually all the arbitrary programming languages are going to be just swept away with the oceans, and we will have the permanence of constructive, intuistionistic type theory as the master theory of computation—without doubt, in my mind, no question. So, from my point of view—this is a personal statement—working in anything else is a waste of time.*

CMU Homotopy Type Theory lecture 1, 52:56–53:20.

What will programming look like in 50 years?

- Convergence of math and computer science
- Functional Programming, Algebra of Programming
- Dependent Types or a successor (Cubical?)
- Who does the programming?

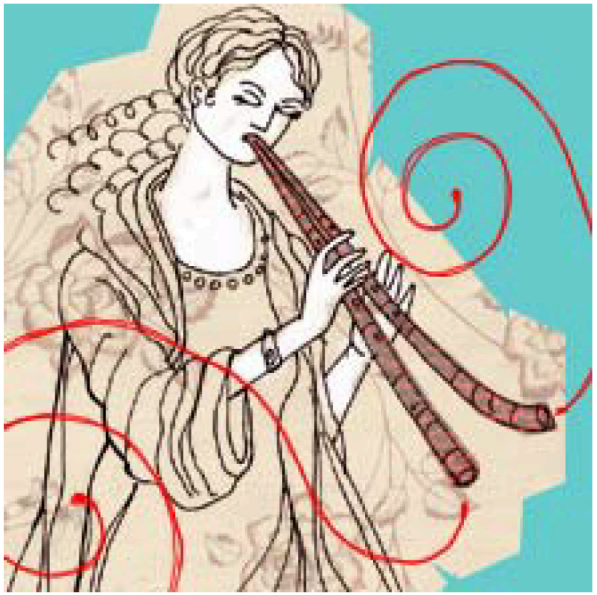
How do we get there from here?

- Add dependent types to an industrial-strength language (Haskell)
- Make a dependently typed language (Agda, Idris) practical to use
- Learn how to program using dependent types
- Many theoretical and practical advances are still needed

Euterpea

The Haskell School of Music

— From Signals to Symphonies —



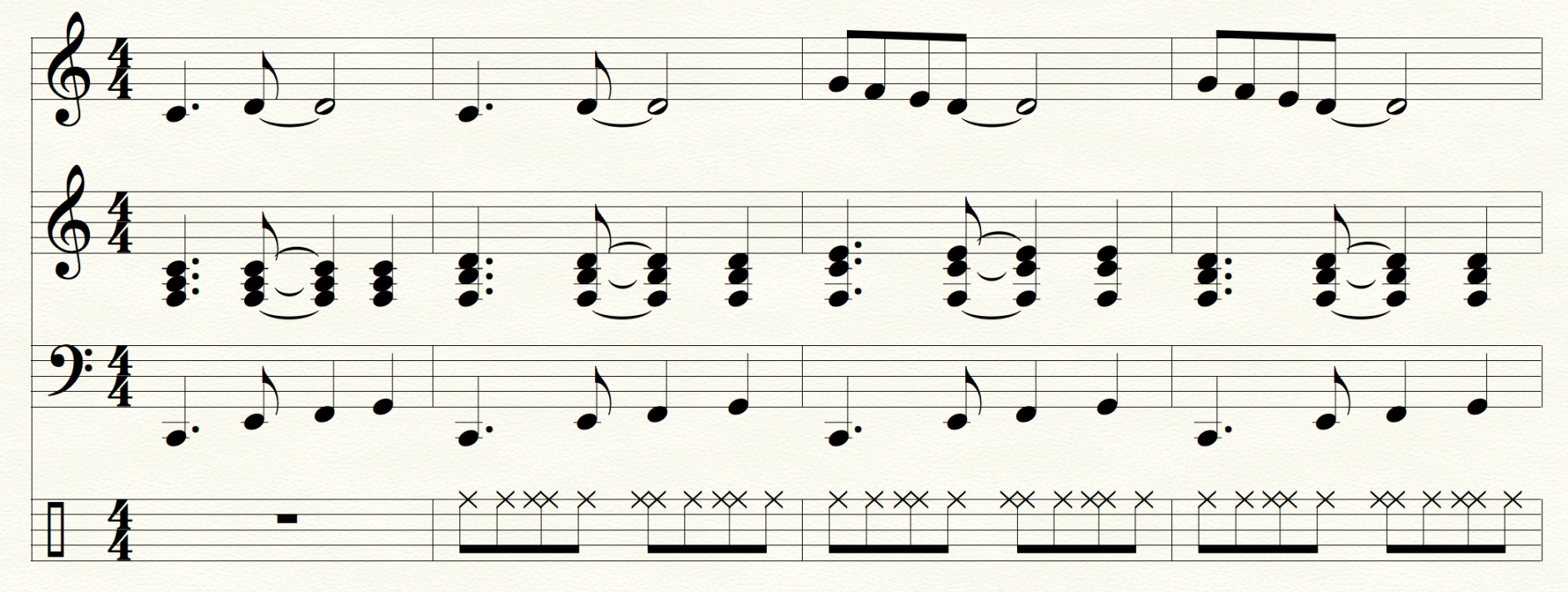
Paul Hudak

Yale University
Department of Computer Science

Music Tools

- Collection of composable tools for synthesis and analysis of music
- Originally written in Haskell
- Converted to Agda, using Haskell for MIDI interface
- Explore programming using dependent types in a circumscribed yet rich domain
- Use math, including transport of equivalences (from HoTT) and Ornaments

Look vs Time (1997)



Music Representation à la Euterpea

```
data Pitch : Set where
  pitch : ℤ → Pitch

data Duration : Set where
  duration : ℕ → Duration

data Note : Set where
  note : Duration → Pitch → Note
  rest : Duration → Note

data Music : Set where
  note : Note → Music
  _::_ : Music → Music → Music -- sequential
  _||_ : Music → Music → Music -- parallel
```

Equivalent Representations of Pitch

```
data Pitch : Set where
  pitch : ℤ → Pitch

chromaticScaleSize : ℕ
chromaticScaleSize = 12

data RelativePitch : Set where
  relativePitch : Fin chromaticScaleSize → RelativePitch

data Octave : Set where
  octave : ℤ → Octave

PitchOctave : Set
PitchOctave = RelativePitch × Octave
```

Equivalences

- Define an equivalence between `Pitch` and `PitchOctave`
- Using HoTT techniques, automatically lift this equivalence to functions defined using `Pitch`
- See *Equivalences for Free!* (Tabareau, Tanter, Sozeau)
- Challenge: Defining base equivalences. Can this be automated?

Ornaments

```
data Music a  = ...
  |  Modify Control (Music a)

data Control = ...
  |  Phrase [PhraseAttribute]

data PhraseAttribute = ...
  |  Orn Ornament

data Ornament =
  Trill | Mordent | InvMordent | DoubleMordent |
  Turn | TrilledTurn | ShortTrill ...
```

Ornaments

- Functions on a base `Music` structure can be automatically lifted to operate on `Music` ornamented with additional information
- See works on Ornaments by McBride, Dagand and others
- Challenge: Shallow embedding of ornaments in Agda

Conclusion

- Music is a good domain in which to explore practical application of dependent types
- Using math can be more work at first, but should be a big win in the long term
- Figure out how to minimize the work and maximize the reward