JFP **29**, e18, 02 pages, 2019.
doi:10.1017/S0956796819000091

Book review

Review of "The Haskell School of Music: from Signals to Symphonies," by Paul Hudak and Donya Quick, Cambridge University Press, 2018

Computer languages for music have a long and interesting history. Music signal processing in particular inspired some early work on dataflow computers, and functional programming concepts are frequently associated with music because of the importance of event sequences and sample sequences in music representation. This book offers a particular approach to music computation using the Haskell programming language and two libraries written in Haskell: Euterpea for representing musical structures and Haskell School of Music for both music performance and simple graphical interfaces for control.

This book is intended to teach computer music concepts along with Haskell. The many programming examples are almost exclusively drawn from music. For example, the first Haskell type expression that appears is *PitchClass*, the set of music note names. Similarly, functions are introduced for musical transposition, sequences are introduced through simple melodies, and so on. Some of the powerful features of Haskell are polymorphic and higher order functions, allowing music representations and operations on them to be parameterized by the details of music events. Music can be an organization of pitches, of chords, of sound qualities, etc., and abstraction can help to avoid being too rigid and limiting. On the other hand, readers are quickly forced to reckon with these abstractions. I expect beginners will struggle with recursive definitions and parametric types. For seasoned functional programmers, there is certainly some beauty in the terse yet flexible notation offered by Euterpea.

One of the attractions of computers for musicians is the ability to describe music as a dynamic process or computation rather than a static score. After introducing Euterpea, this book offers examples of algorithmic composition based on recursion, combinatorics, self-similar sequences, phasing, L-systems, and Markov chains. These are interleaved with more advanced Haskell topics such as type classes, monads, and induction proofs.

Most of this book is concerned with symbolic and parametric representations involving discrete notes with symbolic or numerical parameters such as pitch and loudness. Beginning with Chapter 18, this book turns to audio processing and music synthesis, which is also supported by the Euterpea library. Some basic synthesis methods are covered, including additive synthesis, subtractive synthesis, types of modulation, and simple physical models. These sound generation capabilities mean that one can write algorithms to compose music in terms of symbolic "note" events, describe the translation or "performance" of these events into detailed and nuanced control signals, and finally convert the parametric control into audio, all within one programming language framework. This is unusual, as most music programming systems focus on one representation or abstraction

layer, e.g., one might create a composition language that relegates sound generation to an external synthesizer written in a very different language.

While this book is full of technical details of Haskell as a language, it leaves many questions regarding design and implementation. For example, in a section on "Timing Correctness," it is noted that lazy evaluation can "throw off timing," but since music is so time-critical, most music systems compute and apply accurate timestamps to achieve precise timing. Functional languages almost demand explicit representations of time, but Euterpea seems to have a number of timing mechanisms including explicit time in *Music* data, timers in HSoM, asynchronous execution, and synchronous audio running at a fixed sample rate. This book presents proof techniques for functions and data, but how does one reason about timing or the implications of lazy evaluation? One of the historical driving forces in computer music systems has been signal processing efficiency. Since there are many existing sound synthesis systems, it would be interesting to give some idea of how Euterpea compares in performance.

Since there are at least a handful of functional programming languages for music and the concepts of this book are seen in virtually every computer music system, it would be nice to get a broader view of the landscape of computer music languages, concepts, and design concerns. This is probably also true for functional programming languages in general. Perhaps, the authors thought it was more than enough to describe *one* music system in *one* language, but current practice among many musicians is to integrate various systems. Systems and languages are often selected for inter-operability as much as raw capability.

The coverage of sound and signals in this book is good for beginners, but there are some notable errors. In the definition of *decibels* (dB), there is confusion between amplitude and power. Later, dB is used to describe a frequency ratio, but dB is only appropriate for describing power ratios. When describing equal-loudness contours, it should be stated that these are based on sine tones and not musical tones. Overtones, harmonics, and partials have specific technical meanings, but this book uses them informally without definitions. In connection with the FFT, $O(N^2)$ computations are referred to as "intractable," which is a poor choice in a computer science book. There is also some implied equivalence between frequency (repetition rate) and pitch (a percept), which should be clearly distinguished.

Putting these problems aside, The Haskell School of Music offers an in-depth look into a substantial programming system for music computation and synthesis. This book includes a very readable introduction to Haskell with good examples to illustrate both language and music concepts. Although not the central focus of this book, readers should get a good sense of the challenges that music brings to language design, including multiple representations, temporal structures, symbols and signals, concurrent events, and real-time control. I hope that this book, posthumous for the first author, leads to further exploration by Donya Quick and inspires others to join in this interesting work.

ROGER B. DANNENBERG
School of Computer Science,
Carnegie Mellon University,
Pittsburgh, PA, USA

https://doi.org/10.1017/S0956796819000091 Published online by Cambridge University Press