Euterpea Quick Reference

Donya Quick, 27-Dec-2016

Note: type :i name into GHCi for more information on any type or function name.

Musical Types and Data Structures

type AbsPitch = Int  
MIDI pitch numbers. Middle C = 60.
type Duration = Rational  
1.0 = a whole note in 4/4 = wn (see duration values further down).
data PitchClass = C | Cs | Df | D | Ds | … | B  
f = flat, s = sharp
type Pitch = (PitchClass, Octave)  
Middle C = (C,4)
data Primitive a = Note Dur a | Rest Dur  
basic musical building blocks, notes and rests
data Music a = Prim (Primitive a)  
musical leaf node holding a Primitive (note or rest)
  | (Music a) :+: (Music a)  
  sequential composition
  | (Music a) :=: (Music a)  
  parallel composition
  |Modify Control (Music a)  
  modifier node (affects subtree interpretation)
data Control = Tempo Rational  
Interpret with a tempo multiplier (>1 is faster).
  | Transpose AbsPitch  
  Interpret transposed by some number of pitches (>0 is up).
  | Instrument InstrumentName  
  Interpret using the specified instrument
| ...

type Music1 = Music (Pitch, [NoteAttribute])

Musical Functions

Duration values: wn, hn, qn, en, sn, tn  
whole, half, quarter, etc. Add a “d” for “dotted” (ex: dqn)
Create notes using pitch classes: c, cs, df, d, ds, …, b  
examples: c 4 qn, ef 6 wn
Primitive constructor shorthands: note, rest  
examples: note qn (C,4), rest 4
Sequential composition over a list: line :: [Music a] -> Music a
Parallel composition over a list: chord :: [Music a] -> Music a
Create a percussion/drums note (MIDI channel 9): perc :: PercussionSound -> Dur -> Music Pitch
Modifier shorthands (simply adds a Modify node at the top level): instrument, tempo, transpose
Functions that alter the supplied Music tree: shiftPitches, shiftPitches1, scaleDurations,
changeInstrument, removeInstruments.

Retrograde (reverse): retro :: Music a -> Music a
Inversion (flip upsidedown) around first pitch: invert :: Music Pitch -> Music Pitch
Invert around a particular pitch: invertAt :: Pitch -> Music Pitch -> Music Pitch
Delay by some amount of time: offset :: Dur -> Music a -> Music a
Repeat a finite number of times: times :: Int -> Music a -> Music a
Repeat infinitely: forever :: Music a -> Music a
Remove some amount from the start: cut :: Dur -> Music a -> Music a
Remove some amount from the end: remove :: Dur -> Music a -> Music a
Apply a function to all Notes: mMap :: (a -> b) -> Music a -> Music b
Apply functions to all Music tree nodes: mFold

MIDI Playback Functions

Play to default MIDI output device: play :: (ToMusic1 a, NFData a) => Music a -> IO() Supported a types: AbsPitch, Pitch, (Pitch, Volume), Music1
List available MIDI devices: devices :: IO() Play to a specific MIDI device: playDev Timing-strict playback for finite values: playS, playDevS