Compositional Functional Programming with the Haskore Music

• Todays Topics
  – The Haskore System
  – The Music datatype
    – MIDI Instruments
    – Pitch & absolute Pitch
    – Composing Music
      » Delay
      » Repeating
      » Transposing
  – Manipulating Music
    » Duration
    » Cutting
    » Reversing
  – Percussion
  – Presentation and the MIDI file format
Haskore

• Haskore is a Haskell library for constructing digital music
  – It supports an abstract high-level description of musical concepts
  – Maps into the Midi (Musical Instrument Digital Interface) standard
    » a low-level binary bit based encoding of music
    » can be “played” by “Media-Players”
Musical Basics in Haskore

type Pitch      = (PitchClass, Octave)
data PitchClass =
    Cf | C  | Cs | Df | D  | Ds | Ef | E
    | Es | Ff | F  | Fs | Gf | G  | Gs | Af
    | A  | As | Bf | B  | Bs
    deriving (Eq, Show)
type Octave     = Int
data Music = Note Pitch Dur
| Rest Dur   
| Music :+:: Music
| Music :=: Music
| Tempo (Ratio Int) Music
| Trans Int Music
| Instr IName Music
Midi Standard supports lots of instruments

```haskell
data IName
  = AcousticGrandPiano | BrightAcousticPiano | ElectricGrandPiano | HonkyTonkPiano | RhodesPiano | ChorusedPiano | Harpsichord | Clavinet
  | Celesta | Glockenspiel | MusicBox | Vibraphone | Marimba | Xylophone | TubularBells | Dulcimer
  | HammondOrgan | PercussiveOrgan | RockOrgan | ChurchOrgan | ReedOrgan | Accordion | Harmonica | TangoAccordian
  | AcousticGuitarNylon | AcousticGuitarSteel | ElectricGuitarJazz | ElectricGuitarClean | ElectricGuitarMuted | OverdrivenGuitar | DistortionGuitar | GuitarHarmonics
  | Viola | Cello | Contrabass | TremoloStrings
  | PizzicatoStrings | OrchestralHarp | Timpani | StringEnsemble1 | StringEnsemble2 | SynthStrings1 | SynthStrings2 | ChoirAahs
  | VoiceOoohs | SynthVoice | OrchestraHit | Trumpet
  | Trombone | Tuba | MutedTrumpet | FrenchHorn | BrassSection
  | SynthBrass1 | SynthBrass2 | SopranoSax | AltoSax
  | TenorSax | BaritoneSax | Oboe | Bassoon | EnglishHorn
  | Clarinet | Piccolo | Flute | Recorder | PanFlute
  | BlownBottle | Shakuhachi | Whistle | Ocarina
  | Lead1Square | Lead2Sawtooth | Lead3Calliope | Lead4Chiff
  | Lead5Charang | Lead6Voice | Lead7Fifths | Lead8BassLead
  | Pad1NewAge | Pad2Warm | Pad3Polysynth | Pad4Choir
  | Pad5Bowed | Pad6Metallic | Pad7Halo | Pad8Sweep | FX1Train
  | FX2Soundtrack | FX3Crystal | FX4Atmosphere | FX5Brightness
  | FX6Goblins | FX7Echoes | FX8SciFi | Sitar | Banjo
  | Shamisen | Koto | Kalimba | Bagpipe | Fiddle
  | Shanai | TinkleBell | Agogo | SteelDrums | Woodblock
  | TaikoDrum | MelodicDrum | SynthDrum | ReverseCymbal
  | GuitarFretNoise | BreathNoise | Seashore | BirdTweet
  | TelephoneRing | Helicopter | Applause | Gunshot | Percussion
  | deriving (Show, Eq, Ord, Enum)
```
**Type Dur = Ratio Int**

- fractions of Integers such as 3/4. We write \((3 \% 4)\) in Haskell.

**Type AbsPitch = Int**

**absPitch :: Pitch -> AbsPitch**

**absPitch \((pc,\text{oct})\) = 12*oct + \text{pcToInt pc}**
Pitch to integer

pcToInt :: PitchClass -> Int
pcToInt pc = case pc of
  Cf -> -1  -- should Cf be 11?
  C  -> 0  ;  Cs  -> 1
  Df -> 1  ;  D   -> 2  ;  Ds  -> 3
  Ef -> 3  ;  E   -> 4  ;  Es  -> 5
  Ff -> 4  ;  F   -> 5  ;  Fs  -> 6
  Gf -> 6  ;  G   -> 7  ;  Gs  -> 8
  Af -> 8  ;  A   -> 9  ;  As  -> 10
  Bf -> 10 ;  B   -> 11 ;  Bs  -> 12 -- maybe 0?

Note how several different pitches have the same absolute pitch. This is because the “flat” of some notes is the “sharp” of another.
From AbsPitch to Pitch

\[
\text{pitch12} = [\text{C}, \text{Cs}, \text{D}, \text{Ds}, \text{E}, \text{F}, \text{Fs}, \text{G}, \text{Gs}, \text{A}, \text{As}, \text{B}]
\]

\[
\text{pitch} :: \text{AbsPitch} \rightarrow \text{Pitch}
\]

\[
\text{pitch} \ a = (\text{pitch12} !! \ \text{mod} \ a \ 12, \ \text{quot} \ a \ 12)
\]

\[
(\text{C},0) \ (\text{C},1) \ (\text{C},2) \ (\text{C},3)
\]

\[
\text{trans} :: \text{Int} \rightarrow \text{Pitch} \rightarrow \text{Pitch}
\]

\[
\text{trans} \ i \ p = \text{pitch} \ (\text{absPitch} \ p \ + \ i)
\]
Generic Music - Notes

cf, c, cs, df, d, ds, ef, e, es, ff, f, fs, gf, g, gs, af, a, as, bf, b, bs
:: Octave -> Dur -> [NoteAttribute] -> Music

cf o = Note(Cf, o);  c o = Note(C, o);  cs o = Note(Cs, o)
df o = Note(Df, o);  d o = Note(D, o);  ds o = Note(Ds, o)
ef o = Note(Ef, o);  e o = Note(E, o);  es o = Note(Es, o)
ff o = Note(Ff, o);  f o = Note(F, o);  fs o = Note(Fs, o)
gf o = Note(Gf, o);  g o = Note(G, o);  gs o = Note(Gs, o)
af o = Note(Af, o);  a o = Note(A, o);  as o = Note(As, o)
bf o = Note(Bf, o);  b o = Note(B, o);  bs o = Note(Bs, o)

Given an Octave creates a function from Dur to Music in that octave. Note that Note :: Pitch -> Dur -> Music

These functions have the same names as the constructors of the PitchClass but they're not capitalized.
Generic Music - Rests

wn,  hn,  qn,  en,  sn,  tn :: Dur

wnr,  hnr,  qnr,  enr,  snr,  tnr :: Music

wn  = 1     ; wnr  = Rest wn      -- whole
hn  = 1%2   ; hnr  = Rest hn      -- half
qn  = 1%4   ; qnr  = Rest qn      -- quarter
en  = 1%8   ; enr  = Rest en      -- eight
sn  = 1%16  ; snr  = Rest sn      -- sixteenth
tn  = 1%32  ; tnr  = Rest tn      -- thirty-second

dhn = 3%4   ; dhnr = Rest dhn     -- dotted half
dqn = 3%8   ; dqnr = Rest dqn     -- dotted quarter
den = 3%16  ; denr = Rest den     -- dotted eighth
dsn = 3%32  ; dsnr = Rest dsn     -- dotted sixteenth
Let's Write Some **Music**!

line, chord :: [Music] -> Music

line = foldr (:+:) (Rest 0)

chord = foldr (:=:) (Rest 0)

• Example 1

\[
\text{cScale} = \\
\begin{align*}
\text{line} & = [c \ 4 \ \text{qn} \ [], \ d \ 4 \ \text{qn} \ [], \ e \ 4 \ \text{qn} \ [], \\
& \quad f \ 4 \ \text{qn} \ [], \ g \ 4 \ \text{qn} \ [], \ a \ 4 \ \text{qn} \ [], \\
& \quad b \ 4 \ \text{qn} \ [], \ c \ 5 \ \text{qn} \ []]
\end{align*}
\]

Note the change in Octave
More Examples

\[
\begin{align*}
\text{cMaj} &= \{ n \, 4 \, \text{hn} \mid n \leftarrow [c, e, g] \} \\
\text{cMin} &= \{ n \, 4 \, \text{wn} \mid n \leftarrow [c, ef, g] \}
\end{align*}
\]

• Example 2

\[
\text{cMajArp} = \text{line cMaj}
\]

• Example 3

\[
\text{cMajChd} = \text{chord cMaj}
\]

• Example 4

\[
\text{ex4} = \text{line } \{ \text{chord cMaj, chord cMin} \}
\]
Time Delaying Music

delay :: Dur -> Music -> Music

delay d m = Rest d :+: m

ex5 = cScale :+: (delay dhn cScale)
Transposing Music

ex6 = line [line cMajor
,Trans 12 (line cMajor)]
Repeating Music

repeatM :: Music -> Music
repeatM m = m :+: repeatM m

nBeatsRest n note =
  line ((take n (repeat note)) ++ [qnr])

ex7 =
  line [e 4 qn [], d 4 qn [], c 4 qn [], d 4 qn [],
        line [ nBeatsRest 3 (n 4 qn []) | n <- [e,d] ],
        e 4 qn [], nBeatsRest 2 (g 4 qn []) ]
Cse536  Functional Programming

Fancy Stuff

pr1, pr2 :: Pitch -> Music
pr1 p = Tempo (5%6)
    (Tempo (4%3) (mkLn 1 p qn :+:)
        Tempo (3%2) (mkLn 3 p en :+:
            mkLn 2 p sn :+:
            mkLn 1 p qn :+:)
    mkLn 1 p qn :+:
    Tempo (3%2) (mkLn 6 p en))

pr2 p = Tempo (7%6)
    (m1 :+:)
        Tempo (5%4) (mkLn 5 p en :+:)
        m1 :+:)
    Tempo (3%2) m2)

where m1 = Tempo (5%4) (Tempo (3%2) m2 :+: m2)
m2 = mkLn 3 p en

mkLn n p d = line (take n (repeat (Note p d)))

pr12 :: Music
pr12 = pr1 (C,5) :+: pr2 (G,5)
How long is a piece of music?

```
dur :: Music -> Dur

dur (Note _ d)    = d

dur (Rest d)      = d

dur (m1 :+: m2)   = dur m1   +   dur m2

dur (m1 :=: m2)   = dur m1 `max` dur m2

dur (Tempo  a  m) = dur m / a

dur (Trans  _  m) = dur m

dur (Instr  _  m) = dur m
```
Reversing a piece of music

revM :: Music -> Music

revM n@(Note _ _) = n
revM r@(Rest _) = r
revM (Tempo a  m) = Tempo a    (revM m)
revM (Trans i  m) = Trans i    (revM m)
revM (Instr i  m) = Instr i    (revM m)
revM (m1 :+: m2)  = revM m2 :+: revM m1
revM (m1 :=: m2)
    = let d1 = dur m1
       d2 = dur m2
       in if d1>d2
           then revM m1 :=: (Rest (d1-d2) :+: revM m2)
           else (Rest (d2-d1) :+: revM m1) :=: revM m2
Cutting a piece of music short

cut :: Dur -> Music -> Music

cut d m | d <= 0  = Rest 0
cut d (Note x d0) = Note x (min d0 d)
cut d (Rest d0)   = Rest (min d0 d)
cut d (m1 :=: m2) = cut d m1 :=: cut d m2
cut d (Tempo a m) = Tempo a (cut (d*a) m)
cut d (Trans a m) = Trans a (cut d m)
cut d (Instr a m) = Instr a (cut d m)
cut d (m1 :+: m2) =
  let m1' = cut d m1
       m2' = cut (d - dur m1') m2
  in m1' :+: m2'
Comments

• **Music** is a high level abstract representation of music.

• It's analyzable so we can do many things with it
  
  – First, we can play it
  – But we can also
    » compute its duration (without playing it)
    » reverse it
    » scale it’s Tempo
    » truncate it to a specific duration
    » transpose it into another key
Percussion

data PercussionSound

= AcousticBassDrum -- MIDI Key 35
| BassDrum1 -- MIDI Key 36
| SideStick -- ...
| AcousticSnare | HandClap | ElectricSnare | LowFloorTom
| ClosedHiHat | HighFloorTom | PedalHiHat | LowTom
| OpenHiHat | LowMidTom | HiMidTom | CrashCymbal1
| HighTom | RideCymbal1 | ChineseCymbal | RideBell
| Tambourine | SplashCymbal | Cowbell | CrashCymbal2
| Vibraslap | RideCymbal2 | HiBongo | LowBongo
| MuteHiConga | OpenHiConga | LowConga | HighTimbale
| LowTimbale | HighAgogo | LowAgogo | Cabasa
| Maracas | ShortWhistle | LongWhistle | ShortGuiro
| LongGuiro | Claves | HiWoodBlock | LowWoodBlock
| MuteCuica | OpenCuica | MuteTriangle | 
| OpenTriangle -- MIDI Key 82

  deriving (Show,Eq,Ord,Ix,Enum)
Let’s beat the drums

perc :: PercussionSound -> Dur -> Music
perc ps = Note (pitch (fromEnum ps + 35))

funkGroove
  = let p1 = perc LowTom qn
     p2 = perc AcousticSnare en
  in Tempo 3 (Instr Percussion (cut 8 (repeatM
    :=: roll en (perc ClosedHiHat 2) )
  )))
Music Presentation

- Music is a highlevel, abstract representation
- We call the playing of Music its Presentation
- Presentation requires “flattening” the Music representation into a list of low level events.
  - Events contain information about
    - pitch
    - start-time
    - end-time
    - loudness
    - duration
    - instrument etc.
- The MIDI standard is a file format to represent this low level information.
- Presentation is the subject of the next lecture.
MIDI Event List

Hours, Minutes, Seconds, Frames

Measure, Beats, Ticks

Pitch, Volume, Duration

<table>
<thead>
<tr>
<th>Trk.</th>
<th>HMSF</th>
<th>MBT</th>
<th>Ch</th>
<th>Kind</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00:00:00:00:00</td>
<td>1:01:000</td>
<td>1</td>
<td>Note</td>
<td>C 4</td>
</tr>
<tr>
<td>1</td>
<td>00:00:01:00:00</td>
<td>1:03:000</td>
<td>1</td>
<td>Note</td>
<td>D 4</td>
</tr>
<tr>
<td>1</td>
<td>00:00:02:00:00</td>
<td>2:01:000</td>
<td>1</td>
<td>Note</td>
<td>E 4</td>
</tr>
<tr>
<td>1</td>
<td>00:00:03:00:00</td>
<td>2:03:000</td>
<td>1</td>
<td>Note</td>
<td>F 4</td>
</tr>
<tr>
<td>1</td>
<td>00:00:04:00:00</td>
<td>3:01:000</td>
<td>1</td>
<td>Note</td>
<td>G 4</td>
</tr>
<tr>
<td>1</td>
<td>00:00:05:00:00</td>
<td>3:03:000</td>
<td>1</td>
<td>Note</td>
<td>A 4</td>
</tr>
<tr>
<td>1</td>
<td>00:00:06:00:00</td>
<td>4:01:000</td>
<td>1</td>
<td>Note</td>
<td>B 4</td>
</tr>
<tr>
<td>1</td>
<td>00:00:07:00:00</td>
<td>4:03:000</td>
<td>1</td>
<td>Note</td>
<td>C 5</td>
</tr>
</tbody>
</table>

Time in 2 formats

track

channel