AlgoRhythm

A Library for Algorithmic Music Composition

Joris ten Tusscher, Cas van der Rest, Orestis Melkonian

April 5, 2018

Universiteit Utrecht

Music DSL

music representation (Music, MusicCore, Scale, Chord, etc...
music manipulation (transpose, retrograde, time-scale, etc...

Focus on Generation, Ignore Analysis

Generation

genState, selectors, diatonic improv, etc...

Dynamic Performance

k-means, etc...

Grammars: Properties

(Generative) *context-free grammars*, with a few extra features:

- **Temporal**: Rules are parametric to duration
- **Probabilistic**: Rules can be assigned weights
- **Graph**: Allow node sharing (using *let*-expressions)

Grammars: Definition

```
data Grammar meta a =
    a |: [Rule meta a]
data Rule meta a =
    (a, Weight, Dur -> Bool) :-> (Dur -> Term meta a)
data Term meta a =
    a %: Dur
    | Term meta a :-: Term meta a
    | Aux Bool meta (Term meta a)
    | Let (Term meta a) (λb. Term () b -> Term () b)

class Expand meta a b | meta a -> b where
    expand :: Term meta a -> IO (Term () b)

(a, w) -| f = (a, w, f) :-> (a %:)
a |->  b = a :-> const b
a |--> b = (a, 1, always) |-> b
($:)  = Aux False
(|$:) = Aux True
```

Grammars: Tabla Rhythm

```
tabla :: Grammar () Syllable
tabla = S |:
  [ S  |--> TE1 :-: XI
  , XI |--> TA7 :-: XD
  , XD |--> TA8
  , XG |--> TB2 :-: XA
    ...
  , TE4 |--> Ti :-: Rest :-: Dha :-: Ti
  , TC2 |--> Tira :-: Kita
  , TB3 |--> Dha :-: Tira :-: Kita
  , TD1 |--> Rest
    ...
  ]
instance ToMusicCore Syllable where
    ...
```

Grammars: Tonal Harmony

```haskell
harmony :: Grammar Modulation Degree
harmony = I |:
  [ -- Turn-arounds
    (I,  8, (> wn)) :-> \t ->
      Let (I%:t/2) (\x -> x :-: x)
  , (I,  6, (> hn) /\ (<= wn)) :-> \t ->
      II%:t/4 :-: V%:t/4 :-: I%:t/2
  , (I,  2, (> hn) /\ (<= wn)) :-> \t ->
      V%:t/2 :-: I%:t/2
  , (I,  2) -| (<= wn)
    -- Modulations
  , (V,  5, (> hn)) :-> \t -> Modulation P5 $: I%:t
  , (V,  3) -| always
  , (II, 2, (> hn)) :-> \t -> Modulation M2 |$: I%:t
  , (II, 8) -| always
  ]

instance Expand Degree Modulation SemiChord where
    ...

voiceLead :: Music SemiChord -> IO (Music Chord)
```

Grammars: Jazz Improvisation

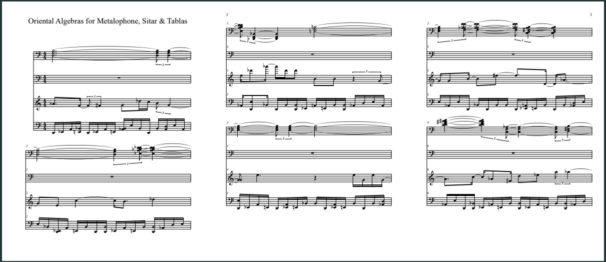
```haskell
melody :: Grammar () NT
melody = MQ |:
  [ -- Abstract Rhythm { MQ ~> Q }
    (MQ,  1, (== qn)) |-> Q%:qn
  , (MQ, 25, (> (hn^.))) :-> \t -> Q%:hn :-: MQ%:(t - hn)
    ...
    -- Concrete Rhythm { Q ~> MN }
  , (Q, 47, (== wn)) |-> MN%:qn :-: Q%:hn :-: MN%:qn
  , (Q,  6, (== hn)) |->
      MN%:(qn^^^) :-: MN%:(qn^^^) :-: MN%:(qn^^^)
    ...
    -- Abstract Melody { MN ~> N }
  , (MN, 1, (== wn)) |-> N%:qn :-: N%:qn :-: MN%:hn
  , (MN, 1, (== qn)) |->
      N%:(en^^^) :-: N%:(en^^^) :-: N%:(en^^^)
    ...
    -- Concrete Melody { N ~> NT }
  , (N, 50, (== qn)) |-> ColorTone%:qn
  , (N, 45, (== qn)) |-> Rest%:qn
  , (N,  1, (== en)) |-> ApproachTone%:en
    ...
  ]

mkSolo :: Music SemiChord -> Music NT -> IO Melody
```

Demo: Code

```
orientalAlgebras = do
  let ?config = MusicConfig
    { basePc    = A
    , baseOct = Oct3
    , baseScale = arabian
    , chords    = equally allChords
    , scales    = equally allScales
    , octaves = [(20, Oct4), (15, Oct5), (5, Oct6)]
    , colorWeight = 0, ...
    , tempo = 6%5
    , instruments = [Piano, Sitar, Tabla]
    , beat = sn
    }
  let t = 12 * wn
  har <- voiceLead  <$> runGrammar harmony t
  mel <- mkSolo har <$> runGrammar melody  t
  rhy <- runGrammar tabla t
  writeToMidiFile "out.mid" (dyn (har :=: mel :=: rhy))
```

Music score


Oriental Algebras for Metalophone, Sitar & Tablas