Categorial Grammars for Automatic Generation of Music

Halley Young
ICFP 2017 FARM Workshop

Outline

- Introduction to music as math vs language
- Mathematical representations of musical objects
- Categorial grammars and linguistic/musical objects
- Use of categorial grammars to automatically generate music

Music as Math vs Language

- Pythagoras: Music as “sounding number”
- Fast forward 2000 years: “Musica poetica” (music as rhetoric)

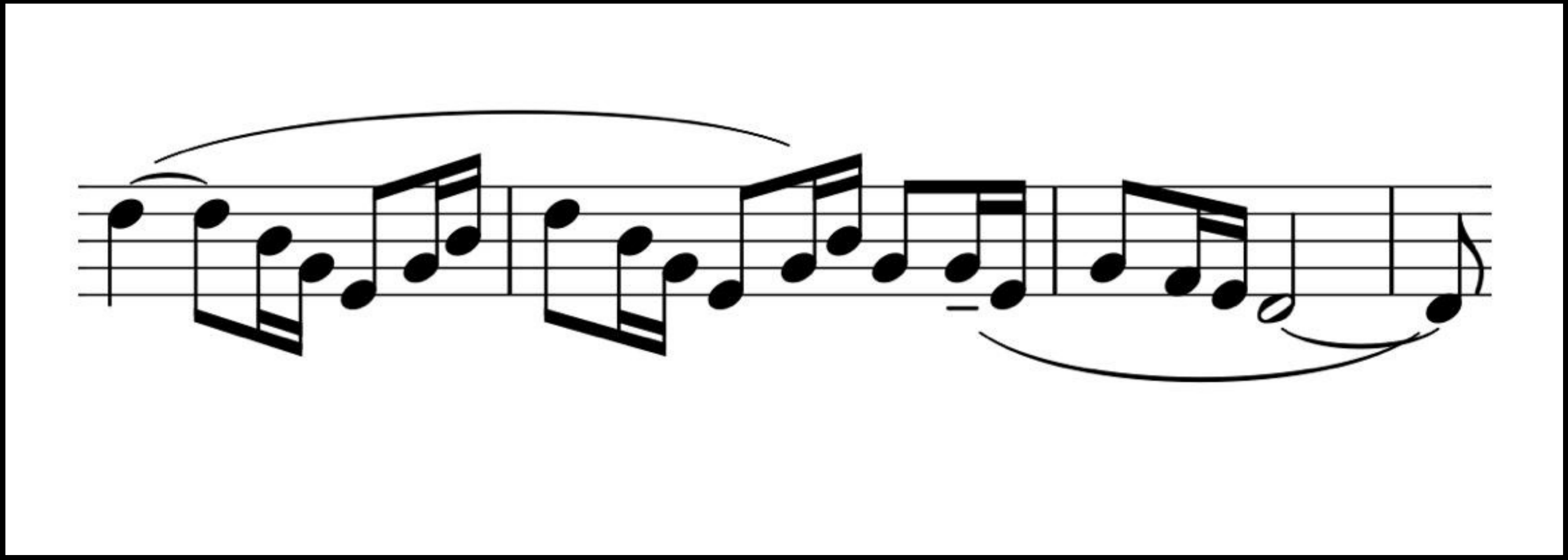
Music is Numbers

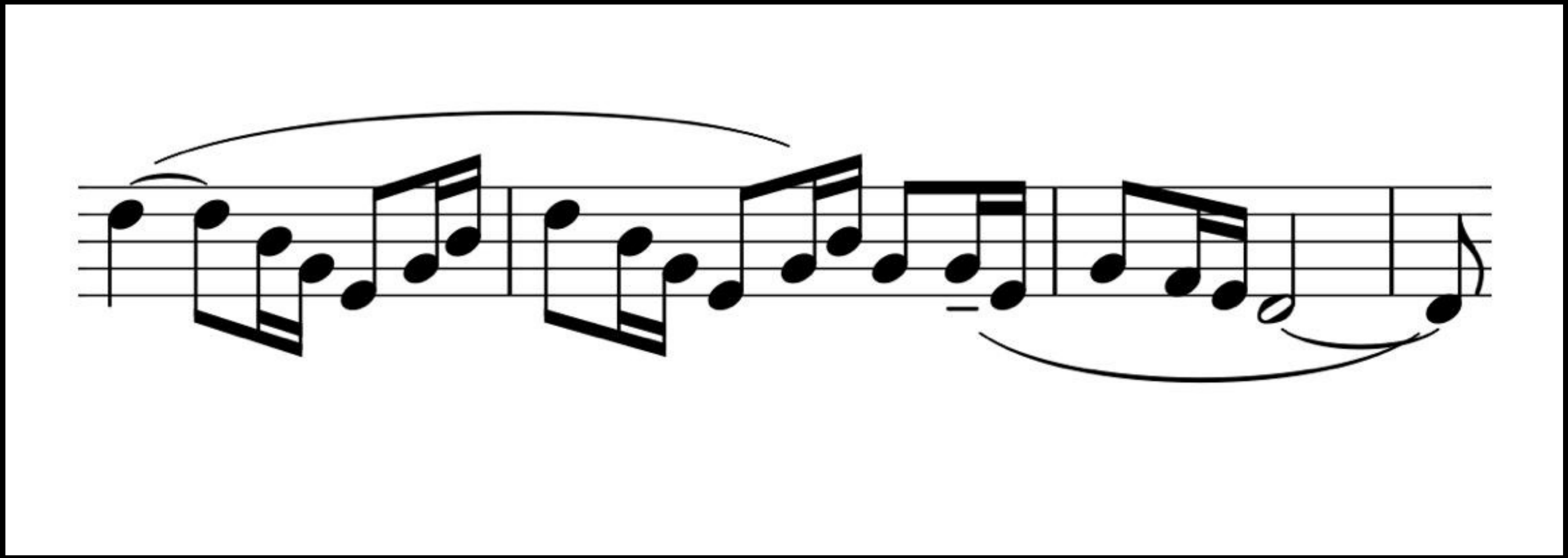
MIDI: (Pitch(Int), Duration(Float), Offset(Float), Instrument(Enum))

[(64,1.0), (62,1.0), (60,1.0), (62,1.0), (64,1.0), (64, 1.0), (64, 2.0), (64,1.0), (62, 1.0), (62,1.0), (62,2.0), (64,1.0), (67,1.0), (67,1.0), (62,1.0), (60,1.0), (62,1.0), (64,1.0), (64, 1.0), (64, 1.0), (64, 1.0), (62, 1.0) (62, 0.5) (62,0.5), (64,1.0), (62, 1.0), (60,1.0)]

Raw audio: long list of decimal numbers between -1 and 1

Music represents a complex mathematical object





- Pentatonic Scale
- Emphasis on pitch-class D
- Repetitive Eighth-Sixteenth-Sixteenth figure
- Inversion + Transposition of first two beats
- Repetition of first three beats

Generative Approach to Musical Complexity

- Music is complex because a complex process generated it
    - may be possible to describe generation of music in multiple ways
    - processes described as involving musical objects

Musical Objects

- dur :: Float
- pitch :: Int
- note :: (pitch, dur)
- melody :: List<note>
- rhythm :: List<dur>
- scale :: List<pitch-class>
- retrograde :: melody -> melody
- transposition :: pitch -> pitch

Categorial Grammars

- Linguistic Formalism based on type theory and lambda calculus
- Used to relate various words to the composite meaning of the entire sentence
- Words inhabit different types, but the resulting type of the sentence is always a statement in predicate calculus

Categorial Grammars

Kim walked and fed the dog.

Kim: k

walked: λx[Walked(x)]

and: λxλyλz[x(z) & y(z)]

fed: λxλy[Fed(y,x)]

the: λx[x]

dog: d

Kim walked and fed the dog: λxλyλz[x(z) & y(z)] (λx[Walked(x)]) (λxλy[Fed(y,x)] (d)) (k) ==

Walked(k) & Fed(k,d)

Music as Language

- Music has “semantics”
- Music has structure akin to “syntax” which interacts with and produces the “semantics”
    - This syntax/semantics related to the relationships between musical objects

Categorial Grammars in Music

- Words have different types := Musical objects have different types
- The type of a composite sentence is the type of a predicate calculus statement := The final type of a composite piece of music is always type melody = List<note>
- Combining words := Combining musical objects to create other musical objects

Categorial Grammars in Music

Objects:

rhythm = [0.5,0.5,1.0]

start_pit = 60

contour = [1,3,2]

combine :: rhythm -> pitch -> contour -> melody

Lambda expression:

λx,y,z.combine(x,y,z)
(rhythm, [0.5, 0.5, 1.0])(start_pit, 60)),

(contour, [1, 3, 2])

Categorial Grammars in Music

λx,y,z.combine(x,y,z)
(rhythm, [0.5, 0.5, 1.0])(start_pit,60),
(contour, [1, 3, 2])

How combine works

```
Def combine(rhythm_z, start_pitch_y, contour_x):
    all_pitch_sequences = start_pitch_y + cartesian_product(all_pitches, product_n =
      length(contour_x) – 1)
    filter(all_pitch_sequences, function_to_filter = lambda y: has_contour(contour_x, y))
    good_melodies = []
    For pit_sequence in all_pitch_sequences:
              good_melodies.append( [ Note(pitch = pit_sequence[i], duration = rhythm[i]) f
                length(rhythm) ] )
    return good_melodies
```

Hierarchical Expressions

augment :: melody -> melody

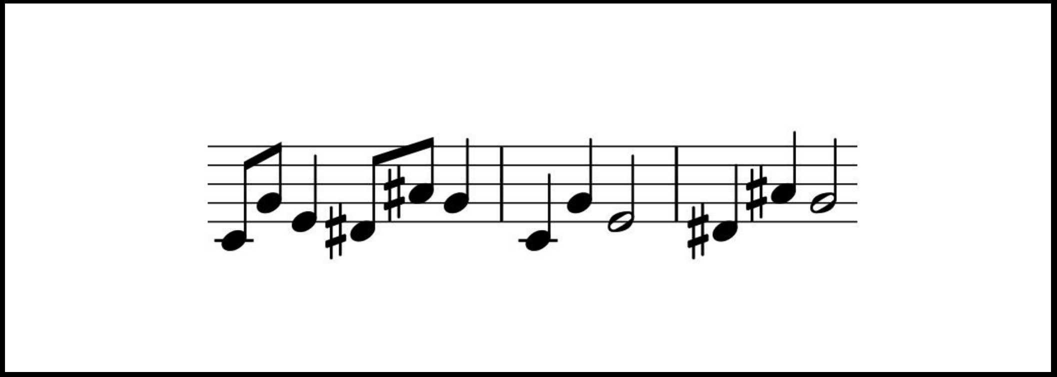
transpose :: melody -> melody

combine :: rhythm -> pitch -> contour -> melody

λx,d.[x,augment(x,d)]

λx,n.[x,transpose(x,n)](λx,y,z.combine(x,y,z) (rhythm, [0.5, 0.5, 1.0])(start_pit, (5, 0)),

(contour, [1, 3, 2]))(3)



Musical Semantics

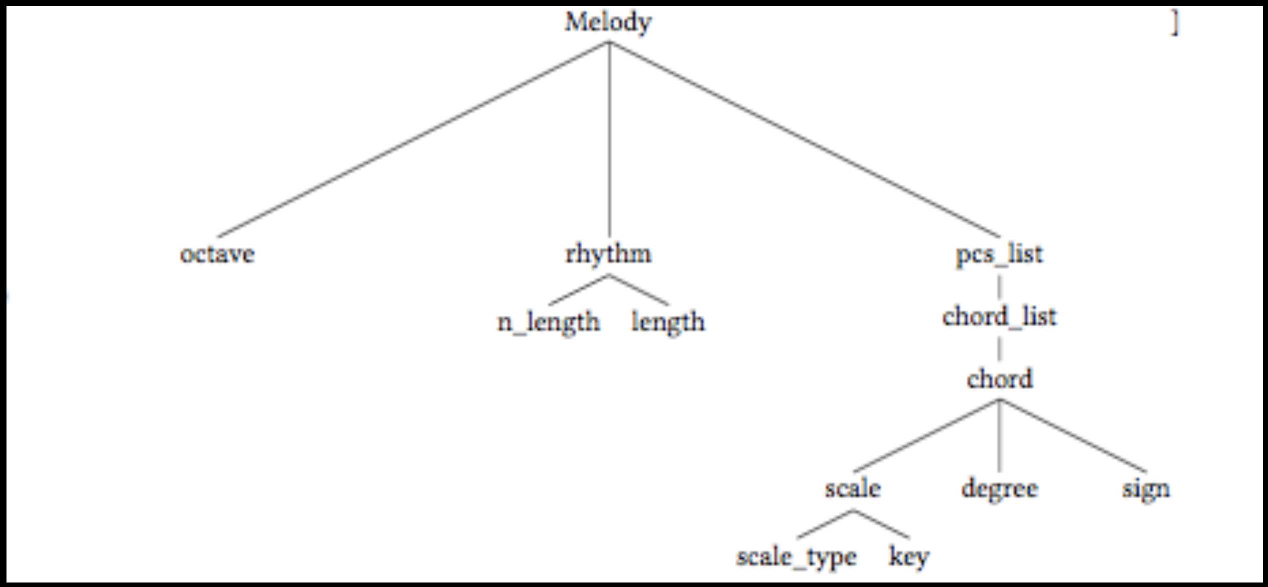
- Semantics of individual expressions?
  - the set of things in the (platonic?) universe of musical objects that they represent
- Semantics of a melody?
  - the semantics of all categorial analyses that could be used to generate the piece of music

Automatically Generating Musical Lambda Expressions

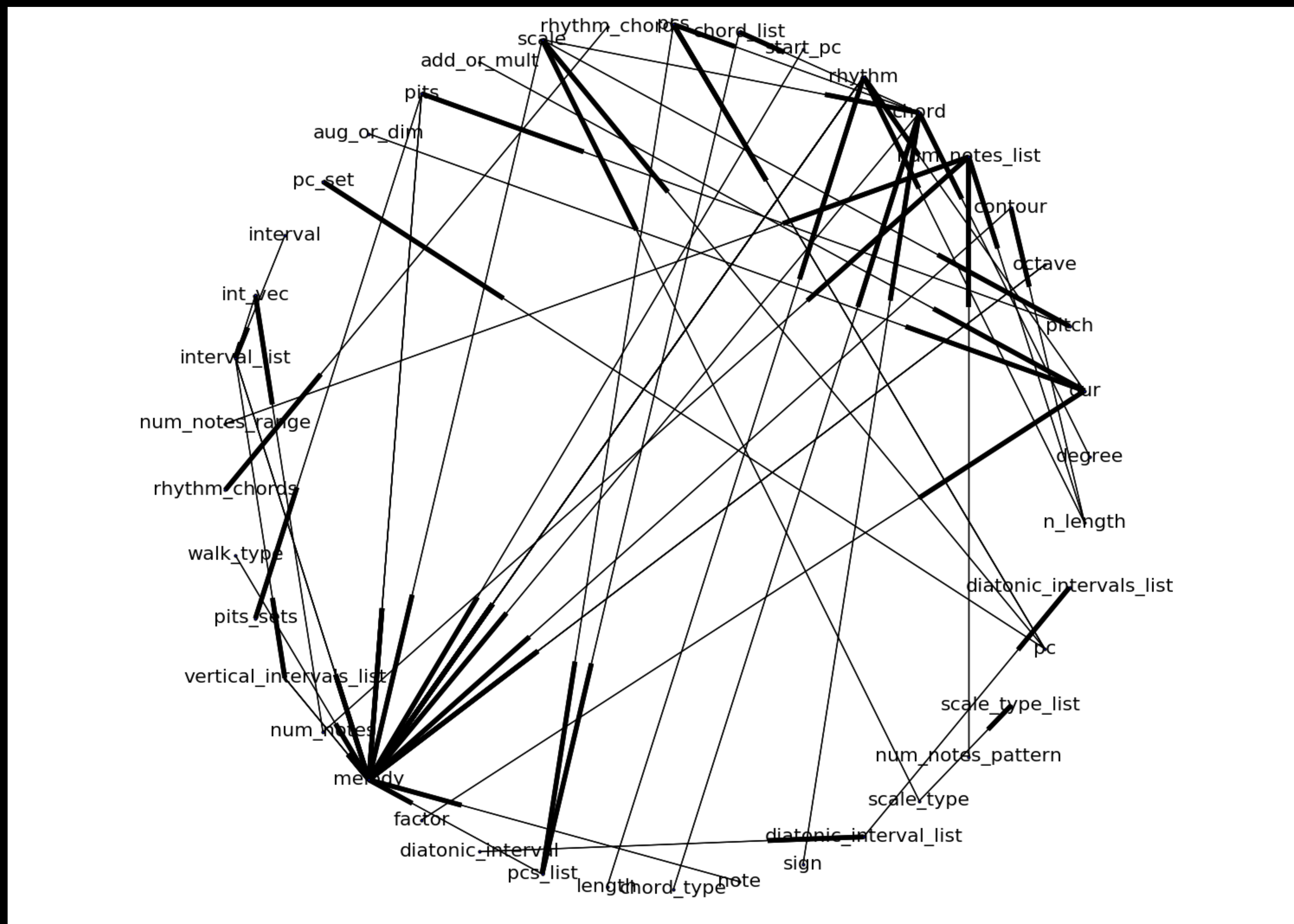
- Traversal of type-relationship graph
- Goal: find path from “primitive” types to melodies (including loops)

Relationships Between Musical Objects

Determined by what functions exist to combine them



Relationships Between Musical Objects



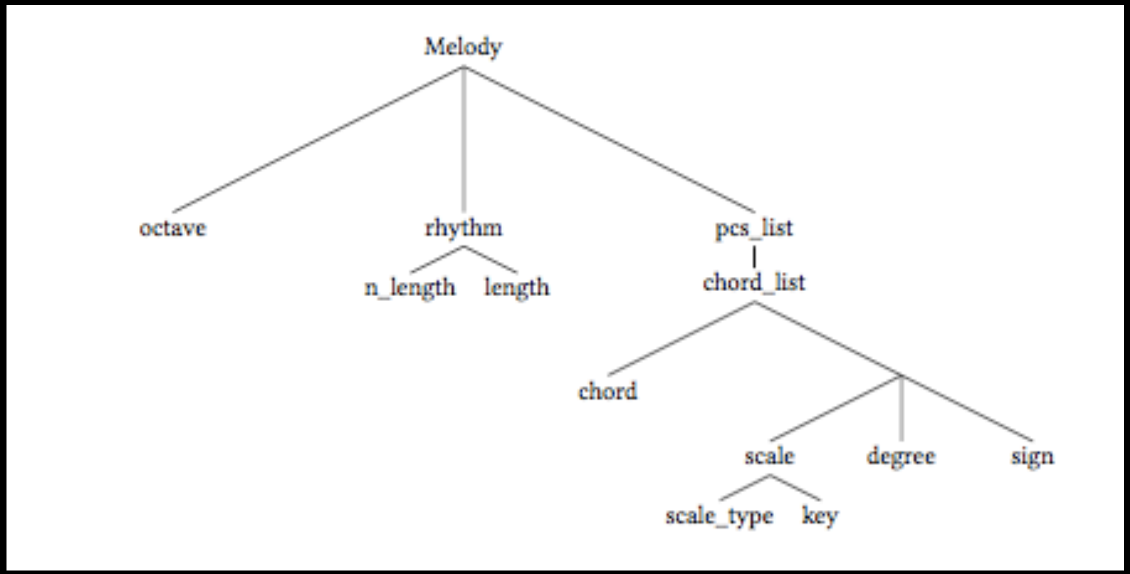
The Graph Traversal Algorithm

```
def genPath(desired_final_node = melody)

  main_path = path in graph from base type-nodes to the desired final node (such that each
  edge in the path represents a function that takes the source node and returns the root
  node

  for each (edge, source_node, target_node) in path:

    other_source_nodes = other arguments to the function besides the specified source
    node

    for each source_node in other_source_nodes:

      new_sub_graph = genPath(desired_final_node = other_source_nodes)

      connect new_sub_graph to main_path

  return main_path
```

Results

```
mel = ( (lambda i1, j1: i1(j1))
((lambda i2, j2: i2(j2))(
applyAllTo , [id, augDimRepeatMelody,
addAppogiaturasMelody ,
chromaticInvertMelody] ) ,
(lambda i2, j2: i2(j2))
(combine10 ,
([("pcs_list", (lambda i4, j4: i4(j4))
(combine11 , ([("chord_list",
(lambda i6, j6: i6(j6))
((lambda i7, j7: i7(j7))
(applyAllTo ,
[ id , fourOf , fiveOf , ] ) ,
(lambda i7, j7: i7(j7))
(combine15 , ([("degree", -1 ),
("scale", (lambda i9, j9: i9(j9))
(combine17 , ([("scale_type", "diatonic"),
("pc", [6,11,5] ), ]) ) ),
("sign", 0 ),
("chord_type",
["triad","ninth", "seventh","eleventh"])
,])))),]))),
("rhythm", (lambda i4, j4: i4(j4))
(combine7 , ([("length", 2.0 ),
("n_length", 3),]))),
 ("octave",6),]))))[1]
writeScore(mel)
```

Results


Melody
octave
rhythm
pcs_list
n_length
length
chord_list
chord
scale
degree
sign
scale_type
key


3
5
8
9
10

Questions?