# Music Composition with LISP

Drew Krause
LispNYC
November 13, 2012

## Lisp Music Environments

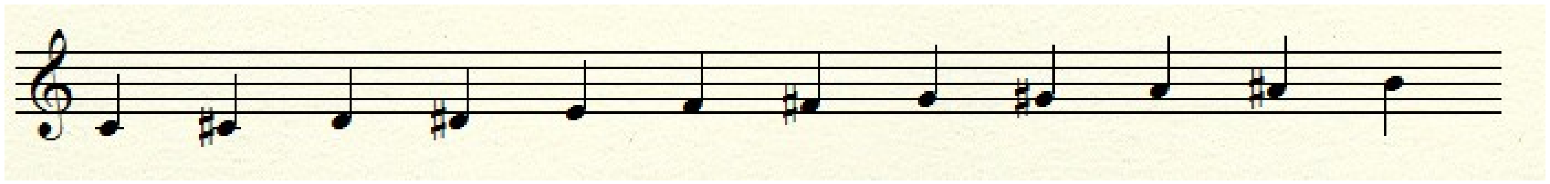
- Common Music
- Common Lisp Music (sound synthesis)
- Open Music (IRCAM gui)
- Symbolic Composer (commercial gui)
- Snd (sound editor w/ Scheme interpreter)
- Overtone (Clojure environment for Supercollider)

# My working environment

$$\begin{tikzcd}
& \text{Common Music (xemacs)} \arrow[d] & \\
& \text{Midi file} \arrow[dl] \arrow[dr] & \\
\text{Finale score} & & \text{Csound note list}
\end{tikzcd}$$

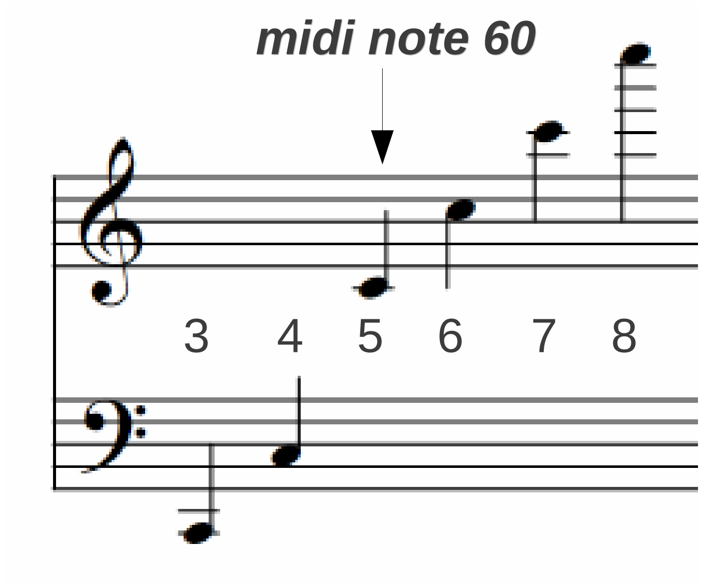
## Preliminaries - pitch

Pitch Class



0 1 2 3 4 5 6 7 8 9 10 11

Register

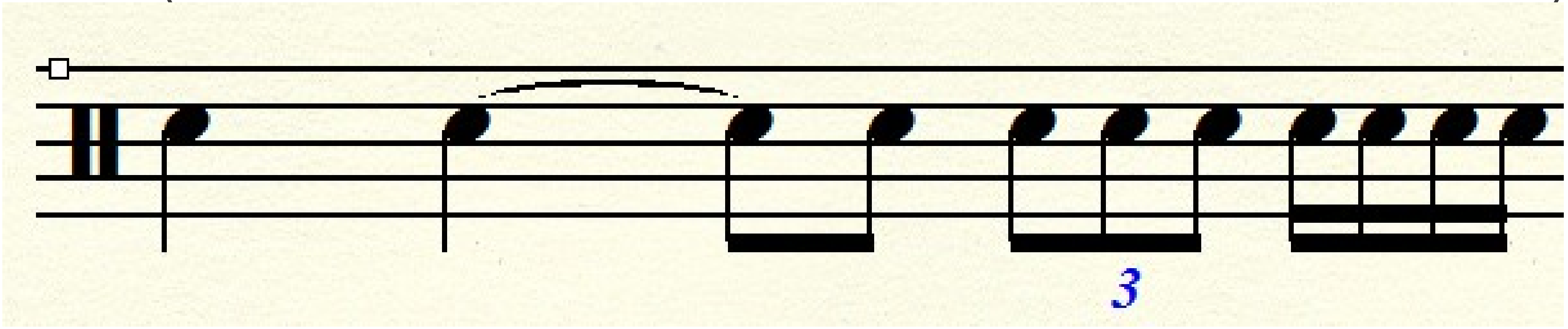


# Preliminaries - rhythm

Duration is conventionally expressed as

1 = quarter note

'(1 1.5 .5 1/3 1/3 1/3 .25 .25 .25 .25)



# Randomness – with or w/o replacement

## pattern 'weighting' – with replacement

***Uniform weights***

```
(new weighting of '(60 62 64 65 67))
```

***Favoring highest & lowest pitches with 10:1 probability***

```
(new weighting of '((60 :weight 10) 62 64 65 (67 :weight 10)))
```

## pattern 'heap' – without replacement

```
(next (new heap of '(60 62 64 65 67)) 20)
```

# Markov chains & analysis

*first- and higher-order transition probabilities*

- created in a transition matrix

***to*** →

***from*** ↓

|   | C | E | G |
|---|---|---|---|
| C | 1/2 | 1/4 | 1/4 |
| E | 1/2 | 0 | 1/2 |
| G | 0 | 1/2 | 1/2 |

- extracted from music

```
(next (markov-analyze birthday) 50)
```

***the tune***

***the 'markov' tune***

# Rewriting Systems

## Morse-Thue

```
(define mtrules '((0 -> (0 1))
                  (1 -> (1 0))))
```

Fourth generation rewrite, with initial condition 1

```
(rwgen mtrules '(1) 4)  ------>  (1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1)
```

rules        init        gen #

Mapped to: 1 = note, 0 = rest

# Spectral Music

*... frequency instead of pitch class*

**Expwarp** = raising frequencies to exponent

```
(loop for n from 1.0 to 3.0 by .1 collect
   (expwarp '(36 55 64) n))
```

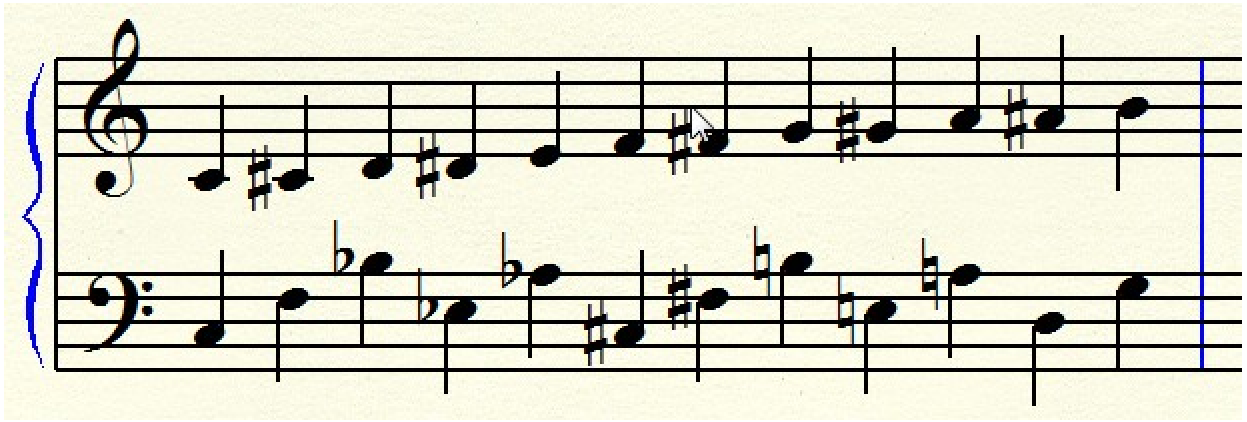
**Scale-spectrum-low** = scales frequency differences (intervals) by new bass note

```
(mapcar (lambda (x) (scale-spectrum-low '(36 55 64) x))
     (placereg tonerow 3))
```

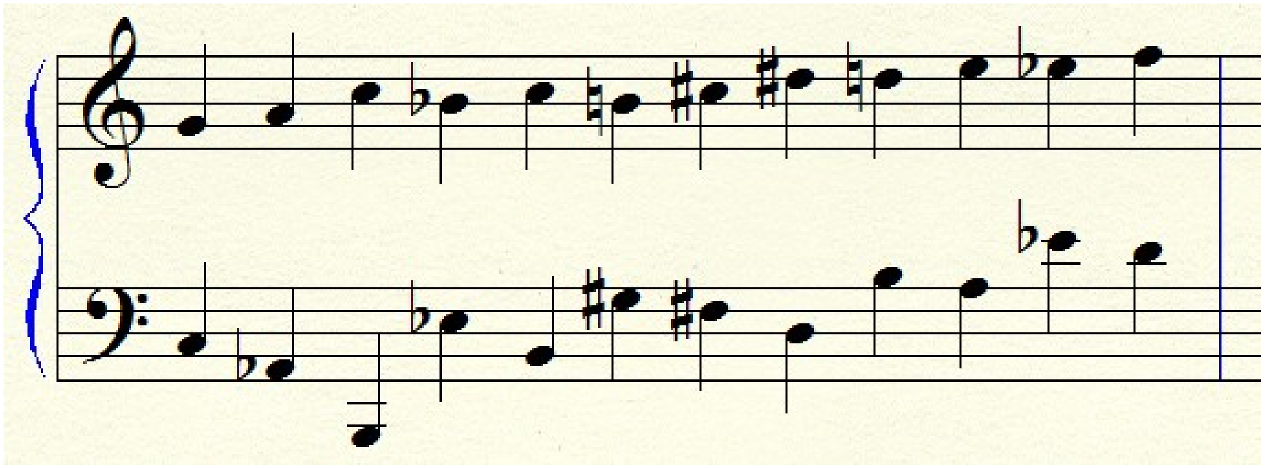
# Spectral Music (II)

## Ring modulation: sum & difference frequencies

Two-voice texture



With ring modulation

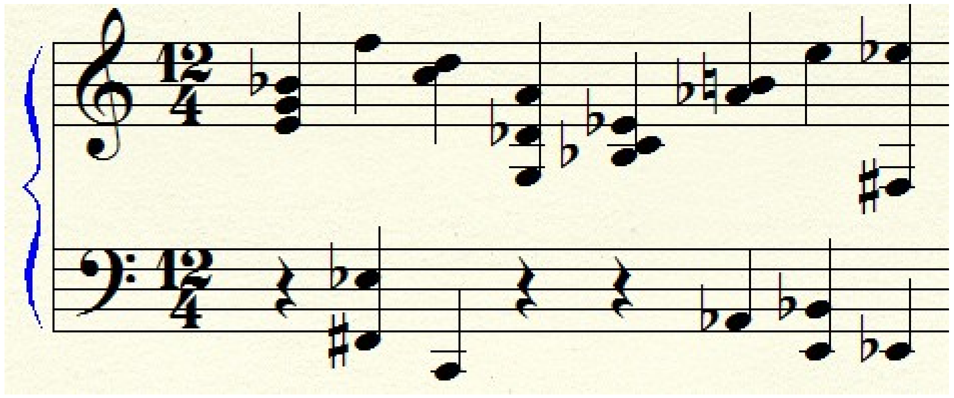


# Optimization

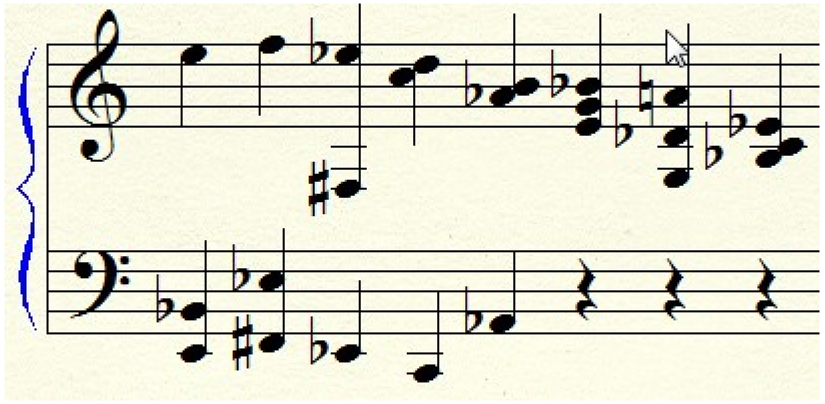
*“Traveling Salesman” problem: given distances between cities, in what order should the salesman visit cities in order to minimize total distance traveled?*

- Cities = trichords
- Distance = total of semitone distance between corresponding members

Random three-note chords



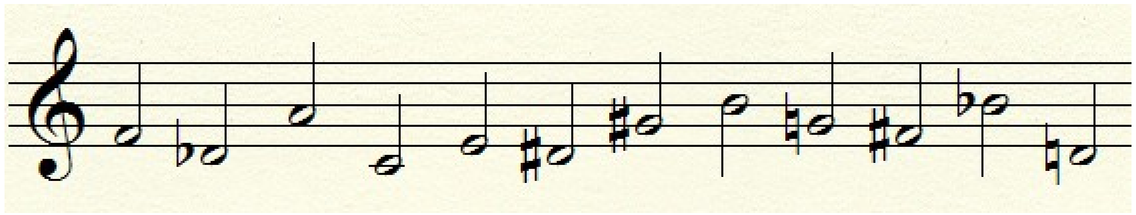
Chords arranged with shortest path



# Constraints

'Wiggle' – get from one pitch to another using only stipulated melodic intervals

tonerow



(wigline tonerow 8 '(5 -2))

Input melody

max # steps

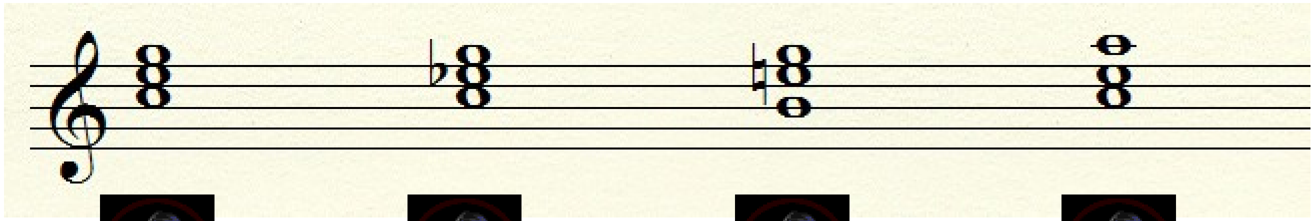
using only +P4, -M2

# Transformations

Neo-Reimannnian “Tonnetz”:

A major triad can go to three minor triads by moving a member of the triad stepwise (and vice-versa)

Source triad  Transform 1  Transform 2  Transform 3



## … a path between any two triads can be made using these three operations

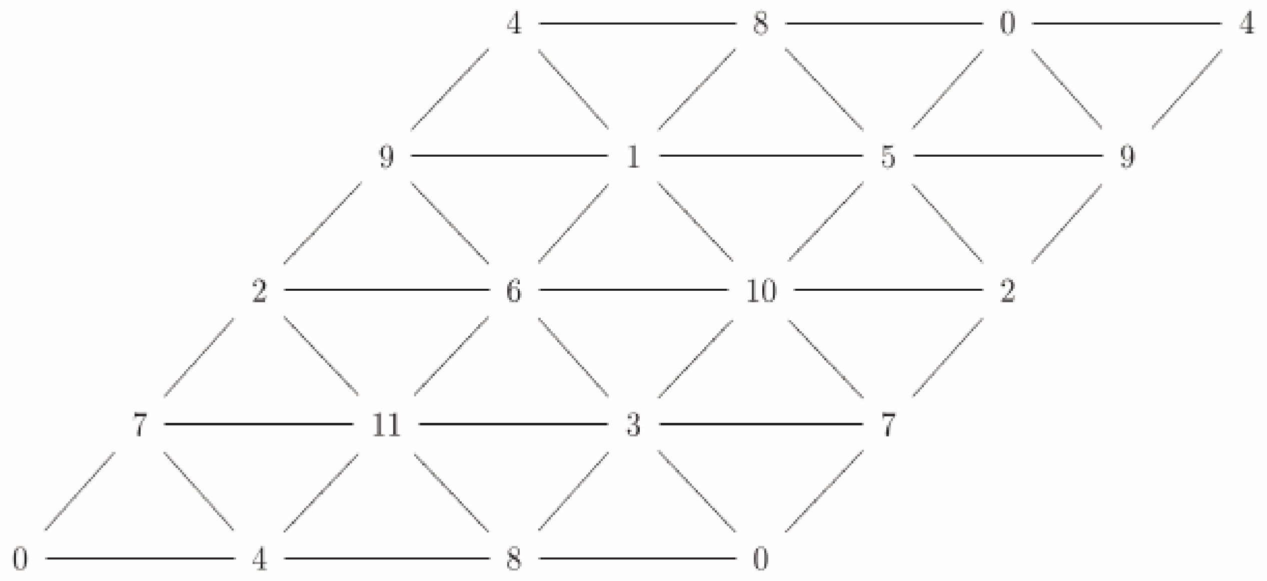


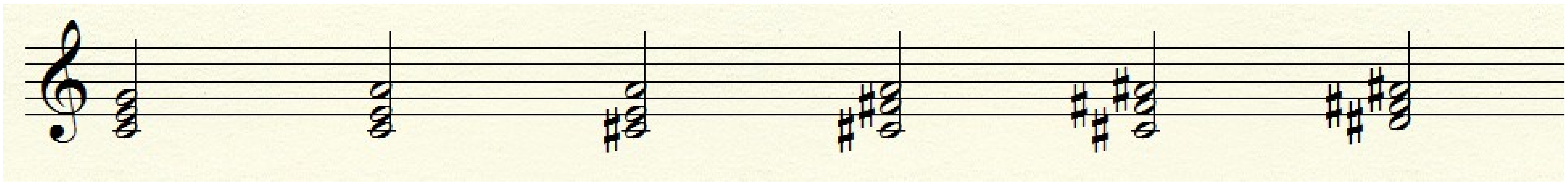
Figure 6. A fundamental region for $C(3,4,5)$.

*these transformations comprise the product of two z12 cyclic groups, with a toriodal structure*

# A* 'best-first' search

Given two chords, find a 'tonnetz' path from one to the other

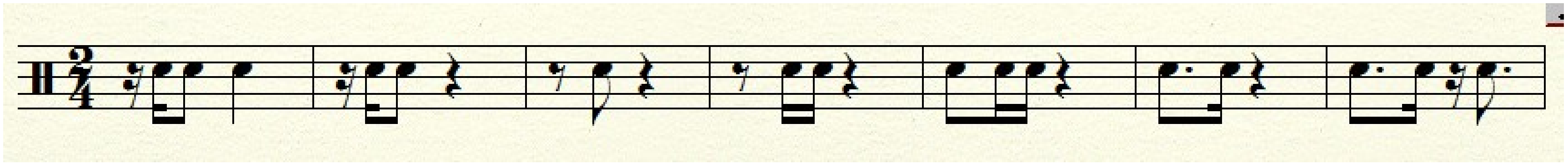
```
(generic-path #'tonnetz-func '(0 4 7) '(3 6 10))
```



# “fromto-stepper”

Treating attack-points as codewords, move stepwise from one rhythm to another

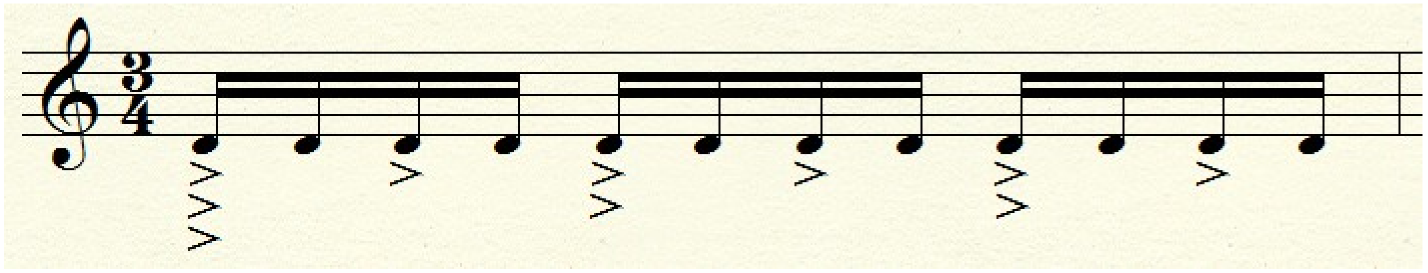
```
CM> cw1
(0 1 1 0 1 0 0 0)
CM> cw2
(1 0 0 1 0 1 0 0)
CM> (fromto-stepper cw1 cw2)
((0 1 1 0 1 0 0 0) (0 1 1 0 0 0 0 0) (0 0 1 0 0 0 0 0)
(0 0 1 1 0 0 0 0)(1 0 1 1 0 0 0 0)
(1 0 0 1 0 0 0 0) (1 0 0 1 0 1 0 0))
```



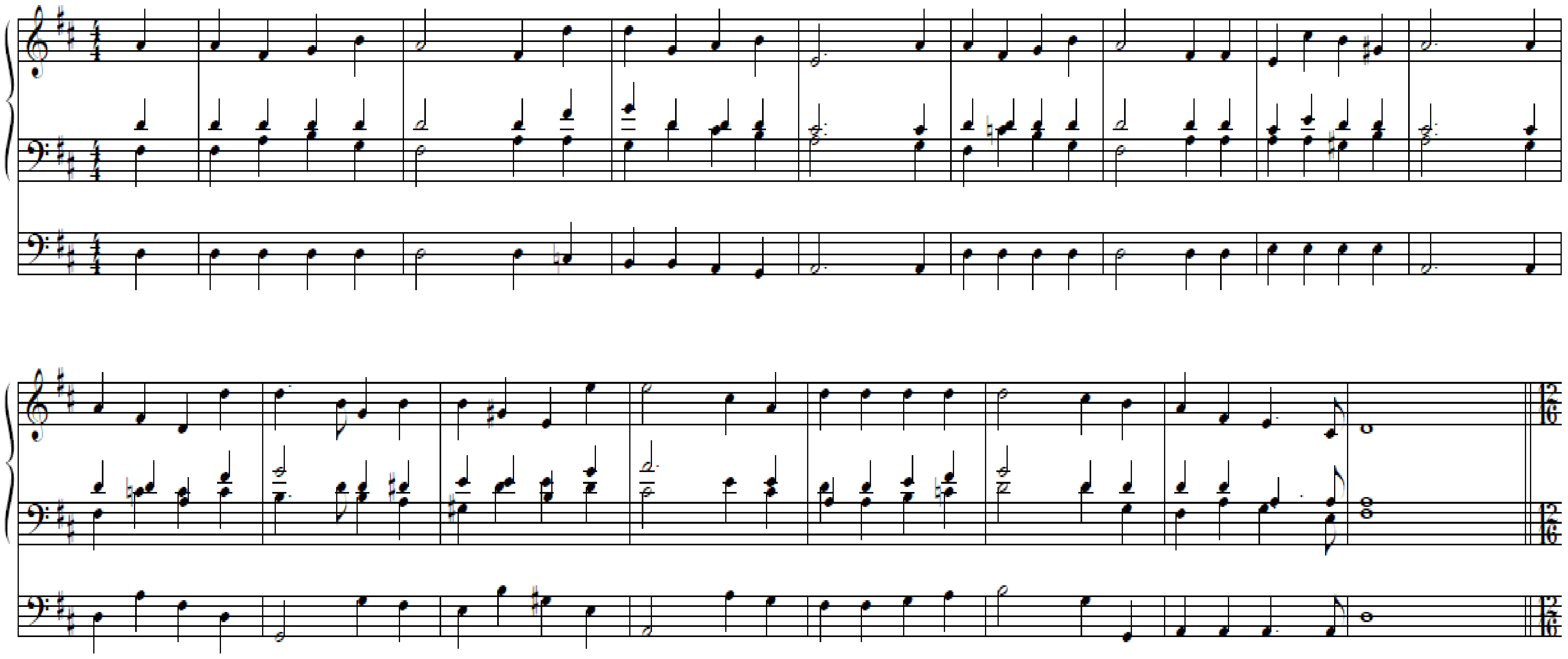
## “Mel-stress”

Metrically distribute the likelihood of a pitch onset

```
CM> stressvector
(4 1 2 1 3 1 2 1 3 1 2 1)
```



# Case Study – Lancashire Variations


(Henry Thomas Smart, 1836)

# Variation A

- paths found between chords via Reger transformations
- result is 'smoothed' (repeated pitches removed from chords)
- chords are arpeggiated & repeats are removed

```
(events
 (splay
  (norpt
   (flatten
    (mapcar #'safesort
        (smoothlist
         (flatter lanca-rgrbranch)))))
  (ferncyc '(1) '(6)))
 "rgrarp.mid" :play 'nil)
```

# Variation B

**Pitch**: soprano melody in 3-voice 'self-stretto' canon
at P5 down, 3-note delay

**Rhythm**: durations = size of chord * 16th

```
(events
 (let ((pits
   (not-flat
    (self-stretto sopr 3 -7 3))))
   (splay pits (durweight pits .25)))
 "stretto1.mid" :play 'nil)
```

# Variation C

**Top Line**: 2nd-order Markov chain of soprano D major scale degrees; repeated notes are tied

**Bottom Line**: every 5, then 4, pitches doubled down P5

```
(events
 (let* ((ipits (play-mode
                (melint->line 39
                (next (markov-analyze
                   (melint (modenums sopr dmajor))
                   :order 2)
                  120))
           dmajor))
        (tpits (make-ties ipits)))
   (list
    (splay (first tpits) (transp (second tpits) .25 #'*))
    (splay (transp (slowline ipits '(5 4)) -7) .25)))
 "jumper.mid" :play 'nil)
```

# Variation D

**Pitch**: 'Slonim' harmonization of soprano w/E5,B4 made into three lines

**Rhythm**: Each line takes its own randomized hymn rhythm ('theselens') at 3 x 16th note

```
(events
 (loop for lin in
       (mapcar #'make-ties
          (chds->lines
       (slonim '(64 59) sopr)))
       collect
       (splay (first lin)
         (sum-across
          (transp (theselens 3) .25 #'*)
          (second lin))))
 "wow.mid" :play 'nil)
```

# Variation E

**Pitch**: four-part chords directly from the hymn, in order

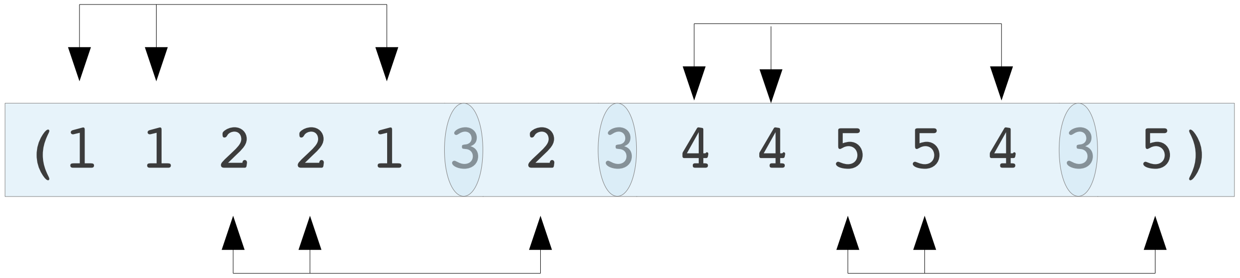
**Rhythm**: attack points from resclassvec 5,9 (duration resultant)

```
CM> (resclassvec 5 9)
(2 0 0 0 0 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 1
 0 0 0 0 1 0 1 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0)
```

```
(events
 (splay lanca-pits
   (makecyc
    (transp
     (code->durs
      (resclassvec 5 9)) .25 #'*)))
 "syncope.mid" :play 'nil)
```

# Variation F

Pitch: "tilevec15" applied to:1 = bass, 2 = tenor, 3 = rest,
4 = alto, 5 = soprano



```
(events
 (play-ties
  (list
   (make-ties
    (place-tiles
     (list
      (makecyc bass)
      (makecyc tenor)
      'r
      (makecyc alto)
      (makecyc sopr))
     (copylist tilevec15 18))))
  .25)
 "tile.mid" :play 'nil)
```

# Variation H

**Pitch**:
"lanca-stravbranch" = sopr w/ slonim C#5,E5,F#5 branched
via 'stravrot-func'
"pits" = 'lanca-stravbranch' smoothed & shuffled,
matched by consonance with soprano line in bass (augmented 5x)
**Rhythm**: each chord in 'pits' is evenly spaced within an 8th note

```
(events
 (let ((pits
    (shuffle-all
     (smoothlist (flatter lanca-stravbranch)))))
   (splay
    (consmatch
     (menses (transp sopr -24) 5)
     (flatten pits))
    (ornadurs pits .5)))
 "oyeah.mid" :play 'nil)
```

# Variation J

**Pitch**: Soprano line doubled at -P5 and -M9, then branched via 'stravrot-func' ("sbranch2"). Each chord sorted w/'closest-mod-list' to make conjunct, then split into lines.

**Rhythm**: attack points @ 8th from all multiples of 3,4,7

```
(events
 (playchds->lines
  (closest-mod-list
   (flatter sbranch2))
  (makecyc
   (transp (code->durs (resclassvec 3 4 7)) .5 #'*)))
 "sbranch2.mid" :play 'nil)
```

## Variation K

**Pitch**: Chorale pitches moved to nearest pitch in Ab major

**Rhythm**: Identical to hymn

```
(events
 (playchds->lines
  (mapcar (lambda (x) (tintab x (transp-mode ionian 8)))
      lanca-pits)
  lanca-durs)
 "chorale-ab.mid" :play 'nil)
```

# Music Composition with LISP

Drew Krause
LispNYC
November 13, 2012

www.wordecho.org