Chronotopes DD_MM_YYYY-HH:MM:SS

for any-sized ensemble of pitched, equal-tempered instruments
for Philip Pocknee

PHILIP: Are there any dot-to-dot music pieces...
DAVID: ...wait a minute...!

About

This document consists of instructions, computer code and an example score for the works *Chronotopes* and *Chronotopes*^{3}.*

Chronotopes is a work for any ensemble (2 - ∞ players) of pitched, equal tempered instruments.

Chronotopes^{3} is a specific subset of *Chronotopes* for 2 classical guitars and one toy piano that was written for *Inlets Ensemble*.

Both of these pieces are based on the solo piano piece *Chronotope* and use scales and relationships taken from the piano work *Labyrinth II*.

In this work, a series of numbered chords or notes are arranged randomly and out-of-order on sheet(s) of paper – one per performer.

The performers try to find and play each chord or note in order, treating the score like a dot-to-dot puzzle.

All durations are based around the time it takes the performer to find the next chord.

For this reason, the score should not be seen by the performer before the moment of performance, to prevent memorization of the order.

For the same reason, any score can only be used once – each performance should use a new score, specifically made for the occasion.

No score can be used more than once.

The scores are algorithmically generated and take approximately two minutes to produce.

New scores can be generated either by modifying and running the elisp code in this document, or by contacting the composer, using details found at www.davidpocknee.com.

The elisp code generates a series of individual parts, each with the name *Chronotopes* and then the date and time in the following format *DD_MM_YYYY-HH:MM:SS.* This is so that there is no confusion among which parts relate to which version of the piece.

There are a series of parameters in the code that can be specified for each version of the piece – these can be adjusted to taste.

Performance

All performers should start with their part placed on a music stand with the side of the paper containing the notes not visible.

On a cue, given by one of the players, all performers should simultaneously flip over their part so that they can see the notes.

Each performer immediately starts trying to find the note/chord numbered "1".

Upon finding this note/chord they should play it and immediately start searching for the note/chord numbered "2".

---

* I use the *Alien* method of assigning titles. If the series continues I look forward to writing *Chronotopes vs Predator.*

They must hold note/chord “1” until they find note/chord “2”.

Upon finding chord “2” they immediately stop playing note/chord “1” and play note/chord “2”.

They continue in this manner, sustaining the last note/chord they found until they find the chord numbered sequentially higher, until they can no longer find any more chords/notes.

If the performer is not physically capable of sustaining the note/chord for the length needed to find the next one, they should sustain it for as long as is physically possible/their instrument allows e.g. length of a breath/sonic decay.

If a player is performing on an instrument with infinite sustaining capabilities, they should stop playing as soon as they realize there are no more chords to find in the piece.

Dynamics

No dynamics are given in any parts.

There are several options of which dynamics to use in this piece, the piece can be played in several versions:

Wandelweiser Version – all notes/chords should be played as quiet as possible.

First Hague School Version – all notes/chords should be played as loud as possible.

Romantic Version – the dynamics of each note/chord should be played according to the inner-soul of the performer.

Cage Version - the *I Ching* is cast to decide on the dynamics for each note/chord.

Boulez Version – the dynamic series from *Structures* is used.

Mezzo Forte Version – all notes/chords should be played *mf*.

Pixies Version - quietquietquietLOUDLOUDLOUDquietquietquietLOUDLOUDquietquietquiet etc.

Players should either all use the same dynamic option, or each pick their own.

If there are large discrepencies in volume between the dynamic ranges of the instruments, all instruments should match their dynamic range to that of the quietest instrument

e.g. in an ensemble of marimba and guitar, the marimba should scale its loudest note to the guitar's loudest note.

Scores

All versions of *Chronotopes DD_MM_YYYY-HH:MM:SS* exist only as a set of parts, not a full score, for obvious reasons.

Example

At the end of this document is an example work for orchestra and piano.

This score is ONLY AN EXAMPLE and should NOT be used for performance.

```
       ; -*-Lisp-*-
; Chronotopes/Chronotope3
; for any ensemble (2+ instruments)(including orchestra)
; by David Pocknee
;  The New Fordist Organization 2014 (www.acesinstitute.eu)
;
; for Philip Pocknee
; PHILIP: Hey, are there any dot-to-dot musical pieces?
; DAVID: ...wait a moment...
;
; This code is designed to be run in elisp
; in windows, on a computer with fomus and lilypond installed and accessible via the command line
; This code does three things in succession:
; 1. Generates fomus files for each instrument
; 2. The fomus files are rendered into lilypond files
; 3. These lilypond files are then edited to remove barlines etc.
; 4. The edited lilypond files are run to create pdfs for each instrument.
;
;
; 3 options for scaledirection:
; 1. Pre-selected scale movement that all instruments copy
; 2. start and end point selected, all instruments random walk there
; 3. Pre-selected scale movement, all instruments random walk towards each new scale.


(defun chronotope (totalspaces totalnotes folder multiscale start-scale end-scale)
(setq allinstruments '(
; format:
; (instrument[fomus recognized]
; instrument[as named in score]
; soundingbottomnote(in MIDI)
; soundingtopnote(in MIDI) transposition (not used)
; polyphonic/monophonic)

;---INSERT INSTRUMENTS HERE-----

;Example of Chronotopes instrumentation (used for example score at end of document)
(flute "flute" 60 96 0 "mono")
(oboe "oboe" 58 94 0 "mono")
(bflat-clarinet "clarinet" 51 87 0 "mono")
(bassoon "bassoon" 34 64 0 "mono")
(horn "french_horn" 41 69 0 "mono")
(bflat-trumpet "trumpet-in-Bb" 55 85 0 "mono")
(tenor-trombone "tenor-trombone" 40 70 0 "mono")
(tuba "tuba" 29 65 0 "mono")
(piano "piano" 21 108 0 "poly")
(violin "violin_I" 55 91 0 "mono")
(violin "violin_II" 55 91 0 "mono")
(viola "viola" 48 84 0 "mono")
(cello "violoncello" 36 64 0 "mono")
(contrabass "contrabass" 28 64 0 "mono")

; Chronotopes3 instrumentation:
;(violin "toy_piano" 53 89 0 "mono")
;(guitar "guitar_1" 40 76 0 "mono")
;(guitar "guitar_2" 40 76 0 "mono")

;------------
))
(setq reel-in-the-time (current-time))
(setq date (format-time-string "%d_%m_%Y-%H_%M_%S" reel-in-the-time nil))
(setq datetitle (format-time-string "%d_%m_%Y-%T" reel-in-the-time nil))

; The lists below are the melodic modes used in the piece:
; They are arranged into a graph system identical to the piece “Labyrinth 2”
; first 7 numbers = scale itself
; next 8 numbers = connecting scales
; next two numbers = co-ordinates
; last number = scale number
(setq scales '(
( 0 2 4 5 7 9 11 1 2 3 4 1 2 3 4 5 7 0 )
( 0 2 3 5 7 9 11 0 0 0 0 6 6 6 6 2 6 1 )
( 0 2 4 5 7 9 10 0 0 0 0 5 6 7 7 4 6 2 )
( 0 2 4 6 7 9 11 0 0 0 0 7 7 8 9 6 6 3 )
( 1 2 4 5 7 9 11 0 0 0 0 8 8 8 8 8 6 4 )
( 0 2 4 5 7 8 10 2 2 2 2 10 10 10 10 1 5 5 )
( 0 2 3 5 7 9 10 1 1 2 2 10 10 11 11 3 5 6 )
( 0 2 4 6 7 9 10 2 2 3 3 11 11 12 12 5 5 7 )
( 1 2 4 6 7 9 11 3 3 4 4 12 12 13 13 7 5 8 )
( 0 2 4 6 8 9 11 3 3 3 3 13 13 13 13 9 5 9 )
( 0 2 3 5 7 8 10 5 5 6 6 14 14 15 15 2 4 10 )
( 0 1 3 5 7 9 10 6 6 7 7 15 15 15 15 4 4 11 )
( 1 3 4 6 7 9 11 7 7 8 8 16 16 16 16 6 4 12 )
```

```
( 1 2 4 6 8 9 11 8 8 9 9 16 16 17 17 8 4 13 )
( 0 2 3 5 6 8 10 10 10 10 10 18 18 18 18 1 3 14 )
( 0 1 3 5 7 8 10 10 10 11 11 18 18 19 19 3 3 15 )
( 1 3 4 6 8 9 10 12 12 13 13 20 20 21 21 7 3 16 )
( 1 2 4 6 8 10 11 13 13 13 13 21 21 21 21 9 3 17 )
( 0 1 3 5 6 8 10 14 14 15 15 23 23 22 22 2 2 18 )
( 1 3 5 7 8 10 11 15 15 15 15 15 15 15 15 4 2 19 )
( 1 3 5 6 8 9 11 16 16 16 16 23 23 23 23 6 2 20 )
( 1 3 4 6 8 10 11 16 16 17 17 23 23 22 22 8 2 21 )
( 0 1 3 4 6 8 10 18 18 18 18 21 21 21 21 3 1 22 )
( 1 3 5 6 8 10 11 20 20 21 21 18 18 18 18 7 1 23 )

)
)

(setq instrumentnumber 0)

(defun scale-route ()
(setq this-scale start-scale)
(setq scale-labyrinth nil)
(setq scale-labyrinth (cons start-scale scale-labyrinth))
(setq xdest (nth 15 (nth end-scale scales)))
(setq ydest (nth 16 (nth end-scale scales)))
(setq xcur (nth 15 (nth this-scale scales)))
(setq ycur (nth 16 (nth this-scale scales)))
(setq this-scale-list (nth this-scale scales))
(setq no-of-scales-used 1)

; Selecting a path through the scales
(while (<= no-of-scales-used totalnotes)
(setq this-scale-list (nth this-scale scales))

(if (equal multiscale "random")
    (progn
      (setq xdest (+ 1 (random 9)))
      (setq ydest (+ 1 (random 7)))
      )
  )

; (print this-scale)
  (if (and (> xdest xcur)(> ydest ycur))
      (progn
      (setq this-scale (nth (random 2) (list (nth 9 this-scale-list)(nth 10 this-scale-list)))))
  (if (and (> xdest xcur)(< ydest ycur))
      (progn
      (setq this-scale (nth (random 2) (list (nth 11 this-scale-list)(nth 12 this-scale-list)))))
  (if (and (< xdest xcur)(> ydest ycur))
      (progn
      (setq this-scale (nth (random 2) (list (nth 7 this-scale-list)(nth 8 this-scale-list)))))
  (if (and (> xdest xcur)(< ydest ycur))
      (progn
      (setq this-scale (nth (random 2) (list (nth 13 this-scale-list)(nth 14 this-scale-list)))))
  (if (and (> xdest xcur)(= ydest ycur))
      (progn
      (setq this-scale (nth (random 4) (list (nth 9 this-scale-list)
               (nth 10 this-scale-list)(nth 13 this-scale-list)(nth 14 this-scale-list)))))
  (if (and (= xdest xcur)(> ydest ycur))
      (progn
      (setq this-scale (nth (random 4) (list (nth 7 this-scale-list)
               (nth 8 this-scale-list)(nth 9 this-scale-list)(nth 10 this-scale-list)))))
  (if (and (= xdest xcur)(< ydest ycur))
      (progn
      (setq this-scale (nth (random 4) (list (nth 11 this-scale-list)
               (nth 12 this-scale-list)(nth 13 this-scale-list)(nth 14 this-scale-list)))))
  (if (and (< xdest xcur)(= ydest ycur))
      (progn
      (setq this-scale (nth (random 4) (list (nth 7 this-scale-list)
               (nth 8 this-scale-list)(nth 11 this-scale-list)(nth 12 this-scale-list)))))
)))))))))

(setq no-of-scales-used (+ 1 no-of-scales-used))
(setq scale-labyrinth (cons this-scale scale-labyrinth))
(setq xcur (nth 15 (nth this-scale scales)))
(setq ycur (nth 16 (nth this-scale scales)))
)

; Organizes when the scale changes
(setq scale-labyrinth (reverse scale-labyrinth))
(print scale-labyrinth)
(if (not (equal multiscale "random"))
    (progn
      (delete-dups scale-labyrinth)
      (setq iterations
```

```
            (/ totalnotes (length scale-labyrinth))
            )
      )
  )
)

(if (or (equal multiscale "group")(equal multiscale "random"))
    (progn
      (scale-route)
      )
  )

; From here to the bottom will repeat until all instruments in list 'allinstruments' are used,
;writing the file to wherever the argument in the main function specifies
(while (<= (+ 1 instrumentnumber) (length allinstruments))
(if (equal multiscale "individual")
    (progn
      (scale-route)
      )
  )


(setq time 0)
(setq voicestaff 1)
(setq countednumber 1)
(setq currentinstr (nth instrumentnumber allinstruments))
(setq usedspaces nil)
(setq scalenotes nil)
(setq scale-destroyer nil)

; all the stuff for the top header of the FOMUS file:
(save-current-buffer
  (setq newfile (format "%s-chronotopes_%s" (nth 1 currentinstr) date))
  (set-buffer (get-buffer-create (format "%s" newfile)))

(insert (format "title \"Chronotopes %s (%s part)\"" datetitle (nth 1 currentinstr)))
(newline)
(insert "author \"David Pocknee \"")
(newline)
(insert "staves-clefchange-score 1")
(newline)
(insert "transpose-keysigs no")
(newline)
(insert (format "part <id \"%s\" inst %s>" (nth 1 currentinstr) (nth 0 currentinstr)))
(newline)
(newline)
(insert (format "part \"%s\"" (nth 1 currentinstr)))
(newline)
)

; The gubbins for the actual notes:

(while (<= countednumber totalnotes)

; totalspaces = total free bars on whole page
; totalnotes = total number of bars to fill on page

; this bit selects a random bar and works out if it is empty:
(if (equal (nth 5 currentinstr) "mono")
    (setq monopolyspaces (* 3 totalspaces))
  (setq monopolyspaces totalspaces)
)

(setq notepos "full")
(while (and (equal notepos "full")(< (length usedspaces) totalnotes))
  (setq notepos (random monopolyspaces))
  (if (equal (memq notepos usedspaces) nil)
      ; if the bar is empty:
    (progn
      (setq usedspaces (cons notepos usedspaces))
      ; Where TIME (i.e. beat position) is set:
      (setq time (* 4 notepos))
     )
; if the bar is full:
    (setq notepos "full")
    )
)

; controls when a new scale is chosen i.e after how many rich scales repetitions
(setq scalenumber (round (* (/
                      (float (- (length scale-labyrinth) 1))
                       (float (- totalnotes 1)))
                     (- countednumber 1)
```

```
                    )))
(setq scalic (nth scalenumber scale-labyrinth))
(setq currentscale (nth scalic scales))
(setq each-scale-note (nth (random 7) currentscale))

(if (equal (length scale-destroyer) 0)
    (progn
      (setq scale-destroyer (list 0 1 2 3 4 5 6))
      (setq destroyedlist (nth (random (length scale-destroyer)) scale-destroyer))
      (setq chosenpitch (nth destroyedlist currentscale))
      (setq scale-destroyer (delq destroyedlist scale-destroyer))
      (setq each-scale-note chosenpitch)
      )
  (progn
      (setq destroyedlist (nth (random (length scale-destroyer)) scale-destroyer))
      (setq chosenpitch (nth destroyedlist currentscale))
      (setq scale-destroyer (delq destroyedlist scale-destroyer))
      (setq each-scale-note chosenpitch)
      )
)

(if (equal (nth 5 currentinstr) "mono")
(progn
(setq pitch 200)
(setq noteinrange 0)
(while (equal noteinrange 0)
  (if (> pitch (nth 3 currentinstr))
      (progn (setq noteinrange 0)
             (setq pitch (+ each-scale-note
                            (+
                             (* 12 (truncate (/ (nth 2 currentinstr) 12.0)))
                             (* 12
                                (random
                                 (-
                                  (ceiling (/ (nth 3 currentinstr) 12.0))
                                  (truncate (/ (nth 2 currentinstr) 12.0))
                                  )))))))
    (if (< pitch (nth 2 currentinstr))
        (progn (setq noteinrange 0)
               (setq pitch (+ each-scale-note
                              (+
                               (* 12 (truncate (/ (nth 2 currentinstr) 12.0)))
                               (* 12
                                  (random
                                   (-
                                    (ceiling (/ (nth 3 currentinstr) 12.0))
                                    (truncate (/ (nth 2 currentinstr) 12.0))
                                    )))))))
      (setq noteinrange 1)
      )
    )
  )
; (print pitch)
(save-current-buffer
  (setq newfile (format "%s-chronotopes_%s" (nth 1 currentinstr) date))
  (set-buffer (get-buffer-create (format "%s" newfile)))
(newline)
(insert (format "time %s dur 4 dyn 50 pitch %s [x %s] ;" time pitch countednumber))
(setq countednumber (+ 1 countednumber))
)
)

; POLYPHONIC -----------------

(progn
  (setq scale1 (butlast currentscale 11))
  (setq shuffled nil)
  (setq shuffler nil)
  (setq lowinstr (nth 2 currentinstr))
  (setq highinstr (nth 3 currentinstr))
 (while (> (length scale1) 0)
   (setq shuffled
         (nth (random (length scale1)) scale1))
   (setq shuffler (cons shuffled shuffler))
   (setq scale1 (delq shuffled scale1))
   )
 (setq all-chord-notes (random (length shuffler)))
 (setq left-chord-notes (random all-chord-notes))
 (setq right-chord-notes (- all-chord-notes left-chord-notes))

 (if (> left-chord-notes 5)
     (setq left-chord-notes 5)
   )
```

```
 (if (> right-chord-notes 5)
     (setq right-chord-notes 5)
   )
 (if (> right-chord-notes 0)
     (setq right-chosen (last shuffler  right-chord-notes))
   (setq right-chosen "ignore")
   )
 (if (> left-chord-notes 0)
     (setq left-chosen
           (butlast shuffler (- (length shuffler) left-chord-notes)))
   (setq left-chosen "ignore")
   )

; LEFT HAND ----------------------------------------

(if
    (not
     (equal left-chosen "ignore"))
    (progn
      (setq left-chosen (sort left-chosen '<))
      (setq left-min (nth 0 left-chosen))
      (setq left-max (nth (- (length left-chosen) 1) left-chosen))

      (if (equal right-chosen "ignore")
          (setq if-right 0)
       (setq if-right 12)
       )

      (setq left-random-octave
            (+
             (* 12 (truncate (/ lowinstr 12.0)))
             (* 12
                (random
                 (-
                  (ceiling (/ (- highinstr if-right) 12.0))
                  (truncate (/ lowinstr 12.0))
                  )))))
      (while
          (not
           (and
            (> (+ left-random-octave left-min) lowinstr)
            (< (+ left-random-octave left-max) highinstr)
            )
           )

       (setq left-random-octave
           (+
            (* 12 (truncate (/ lowinstr 12.0)))
            (* 12
               (random
                (-
                 (ceiling (/ (- highinstr if-right) 12.0))
                 (truncate (/ lowinstr 12.0))
                 )
                )
               )
            )
           )
       )

;      (print (format "left-random %s" left-random-octave))
      (setq unpackcounter 1)
      (while
          (<= unpackcounter (length left-chosen))
       (setq pitch (nth (- unpackcounter 1) left-chosen))
       (setq unpackcounter (+ unpackcounter 1))
;      (print (format "pitch-left %s" (+ pitch left-random-octave)))
       (setq pitch-left (+ pitch left-random-octave))
; NEW
       (setq left-voice 2)
           (if (equal right-chosen "ignore")
               (progn
                 (if (>= left-random-octave 60)
                     (setq left-voice 1)
                   )
                 (setq left-number (format "[x %s]" countednumber))
                 )
             (setq left-number " ")
             )

;endofnew
       (save-current-buffer
         (setq newfile (format "%s-chronotopes_%s" (nth 1 currentinstr) date))
```

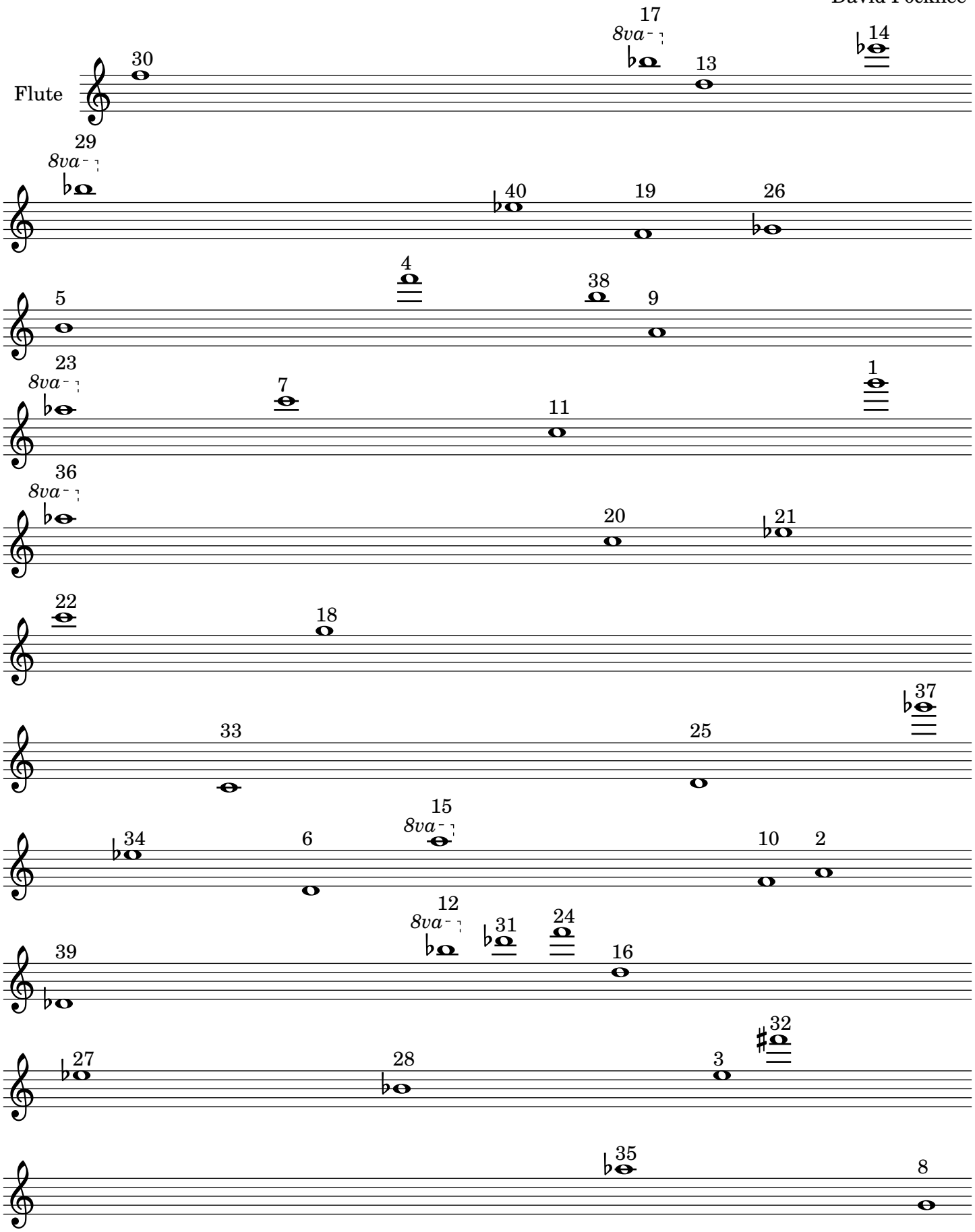
```
          (set-buffer (get-buffer-create (format "%s" newfile)))
          (newline)
          (insert (format "time %s dur 4 dyn 50 pitch %s voice %s staff %s %s ;"
                time pitch-left left-voice left-voice left-number)))
        )
       )
  )
; RIGHT HAND --------------------------------
(if
    (not
     (equal right-chosen "ignore"))
    (progn
      (setq right-chosen (sort right-chosen '<))
      (setq right-min (nth 0 right-chosen))
      (setq right-max (nth (- (length right-chosen) 1) right-chosen))
 ;     (print (format "right-min %s" right-min))
 ;     (print (format "right-max %s" right-max))

      (if (equal left-chosen "ignore")
         (setq if-left lowinstr)
       (setq if-left left-random-octave)
       )

      (setq right-random-octave
           (+
            (* 12 (truncate (/ if-left 12.0)))
            (* 12
               (random
                (-
                 (ceiling (/  highinstr 12.0))
                 (truncate (/ if-left 12.0))
                 )))))
      (while
         (not
          (and
           (> (+ right-random-octave right-min) if-left)
           (< (+ right-random-octave right-max) highinstr)
           )
          )

       (setq right-random-octave
           (+
            (* 12 (truncate (/ if-left 12.0)))
            (* 12
               (random
                (-
                 (ceiling (/ highinstr 12.0))
                 (truncate (/ if-left 12.0))
                 )))))
       )
;      (print (format "right-random %s" right-random-octave))
      (setq r-unpackcounter 1)
      (while
         (<= r-unpackcounter (length right-chosen))
       (setq pitch (nth (- r-unpackcounter 1) right-chosen))
       (setq r-unpackcounter (+ r-unpackcounter 1))
       (setq pitch-right (+ pitch right-random-octave))
;       (print (format "pitch-right %s" (+ pitch right-random-octave)))
       (setq right-voice 1)
       (if (equal left-chosen "ignore")
           (progn
             (if (< right-random-octave 60)
                  (setq right-voice 2)
              )
            )
         )

       (save-current-buffer
         (setq newfile (format "%s-chronotopes_%s" (nth 1 currentinstr) date))
         (set-buffer (get-buffer-create (format "%s" newfile)))
         (newline)
         (insert (format "time %s dur 4 dyn 50 pitch %s voice %s staff %s [x %s] ;"
               time pitch-right right-voice right-voice countednumber)))
       )
      )
  )
       (setq countednumber (+ 1 countednumber))
)
)
)

(save-current-buffer
  (setq newfile (format "%s-chronotopes_%s" (nth 1 currentinstr) date))
```

```
  (set-buffer (get-buffer-create (format "%s" newfile)))
  (write-file (format "%s%s.fms" folder newfile))
  (kill-buffer (format "%s.fms" newfile))
   )
(setq instrumentnumber (+ 1 instrumentnumber))
)
; (setq dirtest "c:/users/david/documents/chronotope2/batchorchestratest")
(shell)
(shell-process-cd folder)
;(shell-process-cd "c:\\users\\david\\documents\\chronotope2\\testfiles2")
(shell-command "for %F in (*.fms) do (fomus -i %F -o %~nF.ly)")
(progn
  (setq directory (directory-files folder))
;  (nth 2 directory)
  (mapcar
   (function
    (lambda (x)
      (if
          (and
           (> (length x) 3)
           (equal (substring x -3 nil) ".ly")
           )
          (progn
            (find-file (format "%s/%s" folder x))
            (save-current-buffer
              (set-buffer (get-buffer-create x))
              (goto-char (point-max))
              (insert "\\layout {")(newline)
              (insert "\\context {")(newline)
              (insert "\\Staff")(newline)
              (insert "\\override MultiMeasureRest #'transparent = ##t")(newline)
              (insert "\\override MultiMeasureRestNumber #'transparent = ##t")(newline)
              (insert "\\override MultiMeasureRestText #'transparent = ##t")(newline)
              (insert "\\override BarLine #'break-visibility = #'#(#f #f #f)")(newline)
              (insert "\\remove \"Time_signature_engraver\"")(newline)
              (insert "}")(newline)
              (insert "\\context {")(newline)
              (insert "\\Score")(newline)
              (insert "\\remove \"Bar_number_engraver\"")(newline)
              (insert "}")(newline)
              (insert "}")
              (goto-char (point-max))

              (search-backward "composer")
              (goto-char (line-end-position))
              (newline)
;             (insert "subtitle = \"for any size ensemble of equal-tempered instruments\"")(newline)
       ;       (insert "subtitle = \"for Philip Pocknee\"")(newline)
              (insert (format "tagline = \"The New Fordist Organization %s (www.acesinstitute.eu)\""
                       (format-time-string "%Y" reel-in-the-time nil)))(newline)
              (write-file (format "%s/%s" folder x))
              (kill-buffer (format "%s" x))
             )
           )
        )
       )
     )
   directory
   )
  )
(shell-command "for %F in (*.ly) do (lilypond %F)")
)

;INSTRUCTIONS:
; 1. Execute (C-x-e) the code above.
; 2. Execute the command at the bottom of the page, adjusting the variables as follows:
; Variables:
; totalspaces = total number of bars on the page
; totalnotes = total number of numbered notes/chords created
; folder = folder to output .fms files to
; multiscale = "group" - every part has the same path through the scale labyrinth
             ; "individual" - each part takes its own path through the scale labyrinth
              ; "random" all parts follow the same random path through the labyrinth
; start-scale = the scale all instruments start with
; end-scale = the scale all instruments end on
; (chronotope totalspaces totalnotes folder multiscale start-scale end-scale)

(chronotope 70 40 "c:/users/david/documents/chronotope2/robertjorge/scores/" "group" 0 23)
```

Chronotopes 10_02_2014-21:56:25 (flute part)

for Philip Pocknee

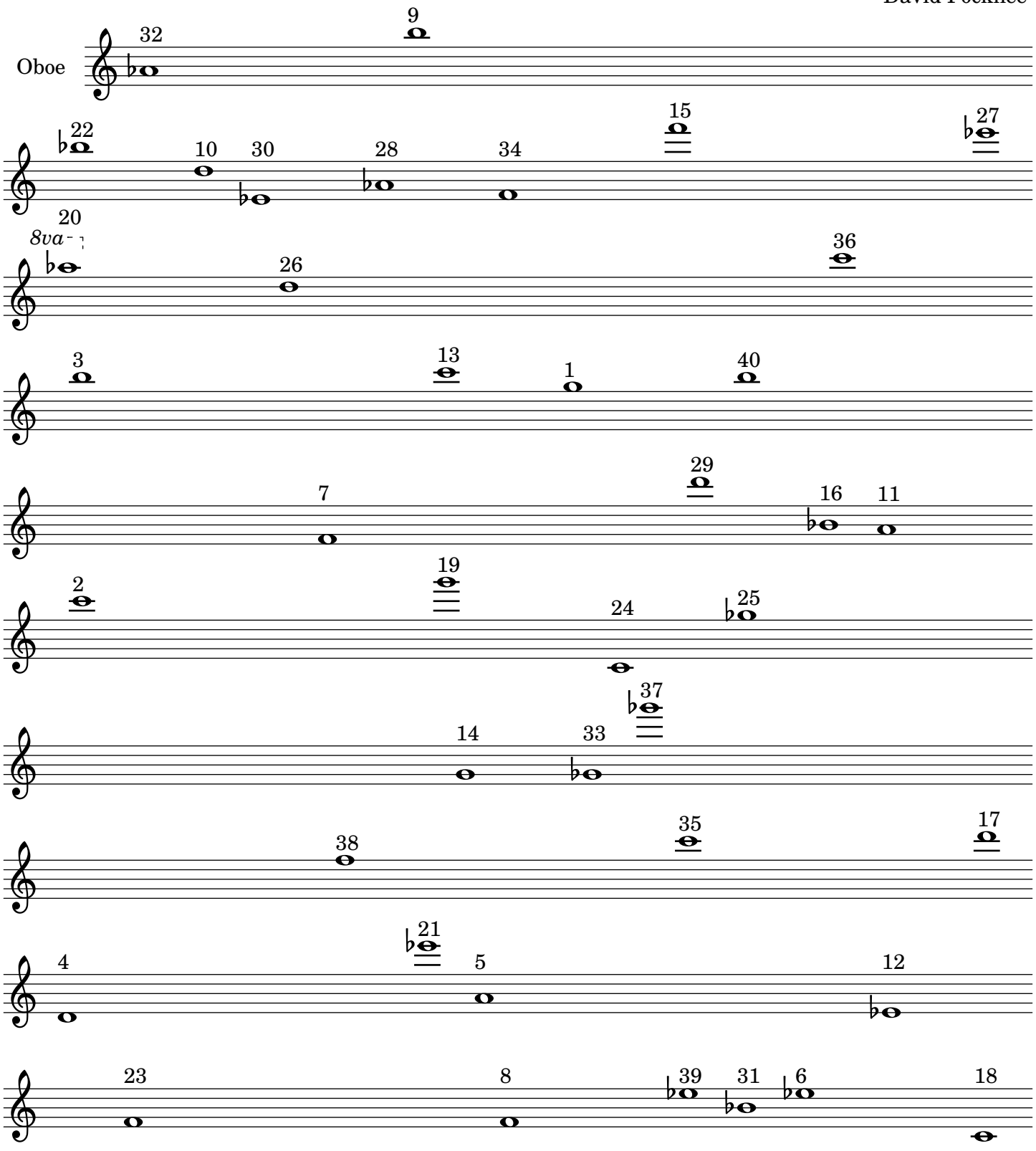
David Pocknee



Chronotopes 10_02_2014-21:56:25 (oboe part)

for Philip Pocknee

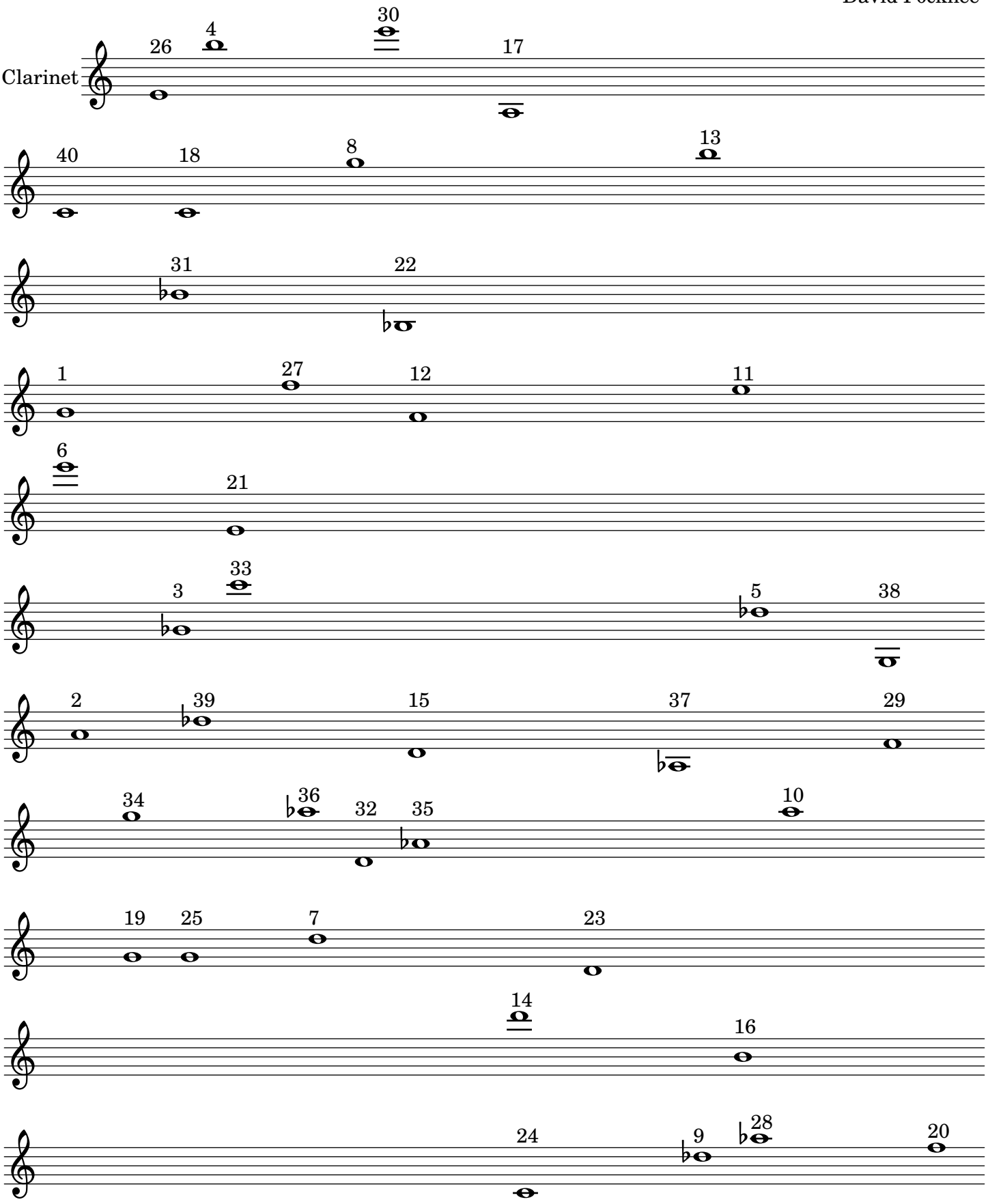
David Pocknee



Chronotopes 10_02_2014-21:56:25 (clarinet part)

for Philip Pocknee

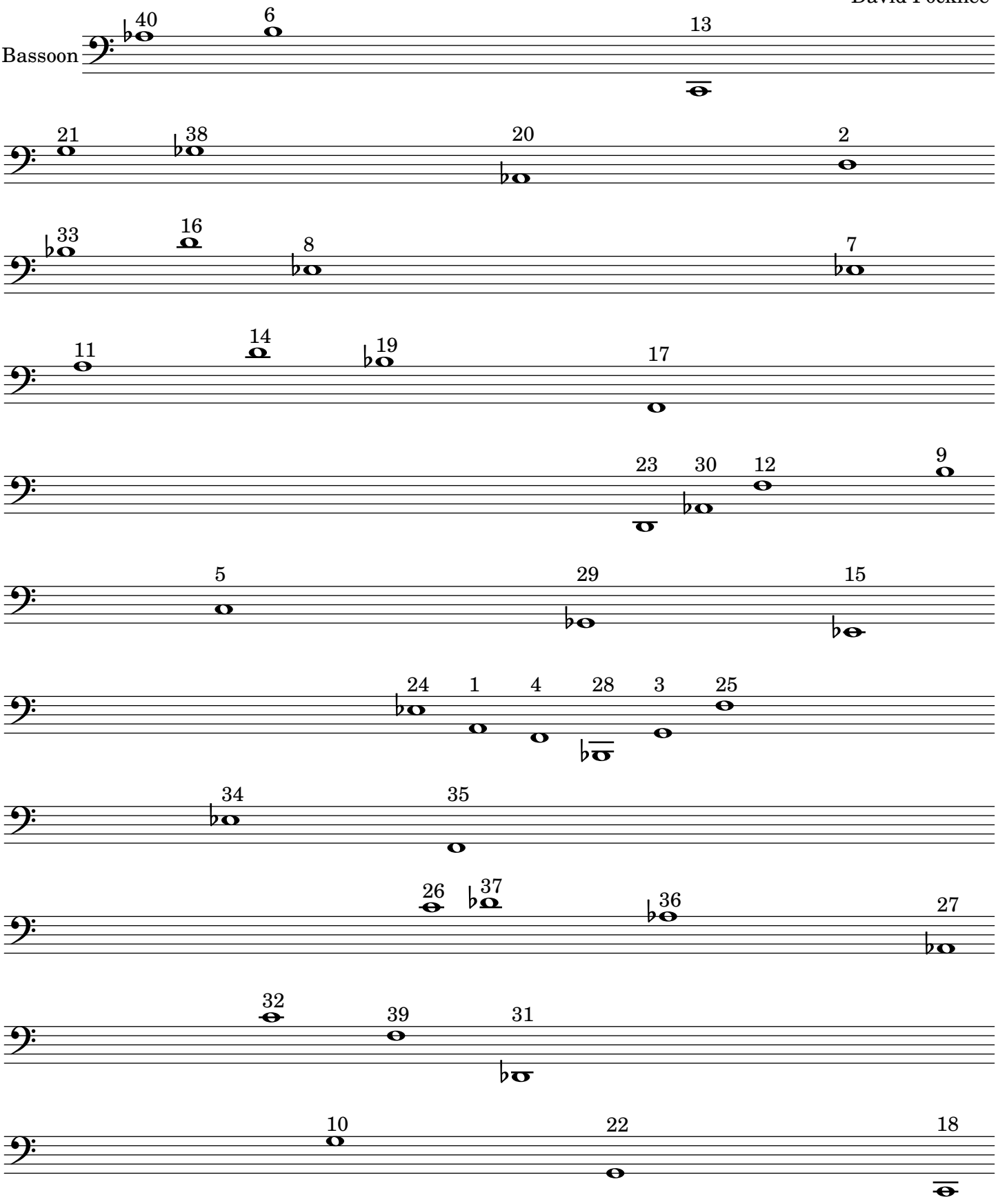
David Pocknee



Chronotopes 10_02_2014-21:56:25 (bassoon part)

for Philip Pocknee

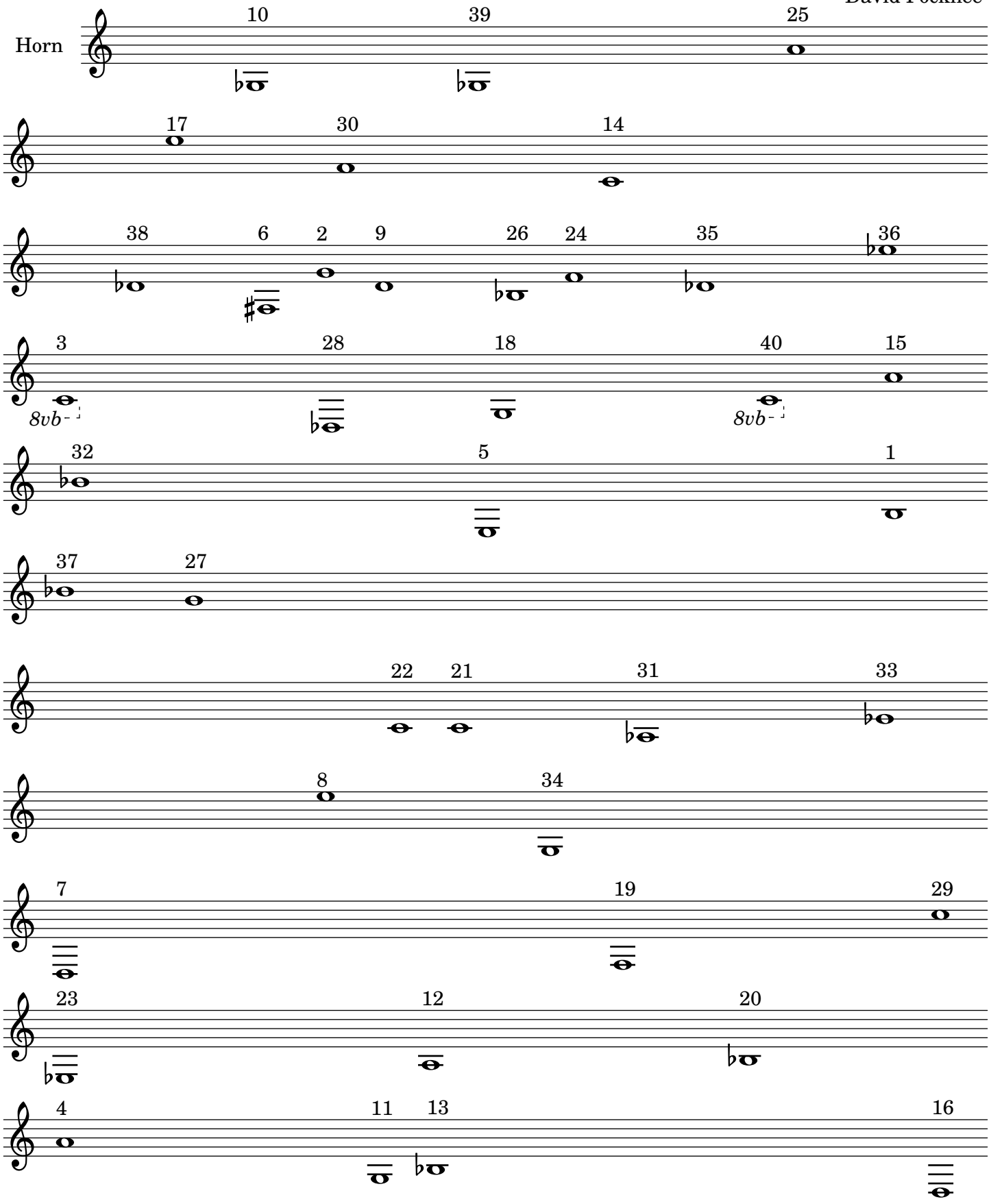
David Pocknee



Chronotopes 10_02_2014-21:56:25 (french_horn part)

for Philip Pocknee

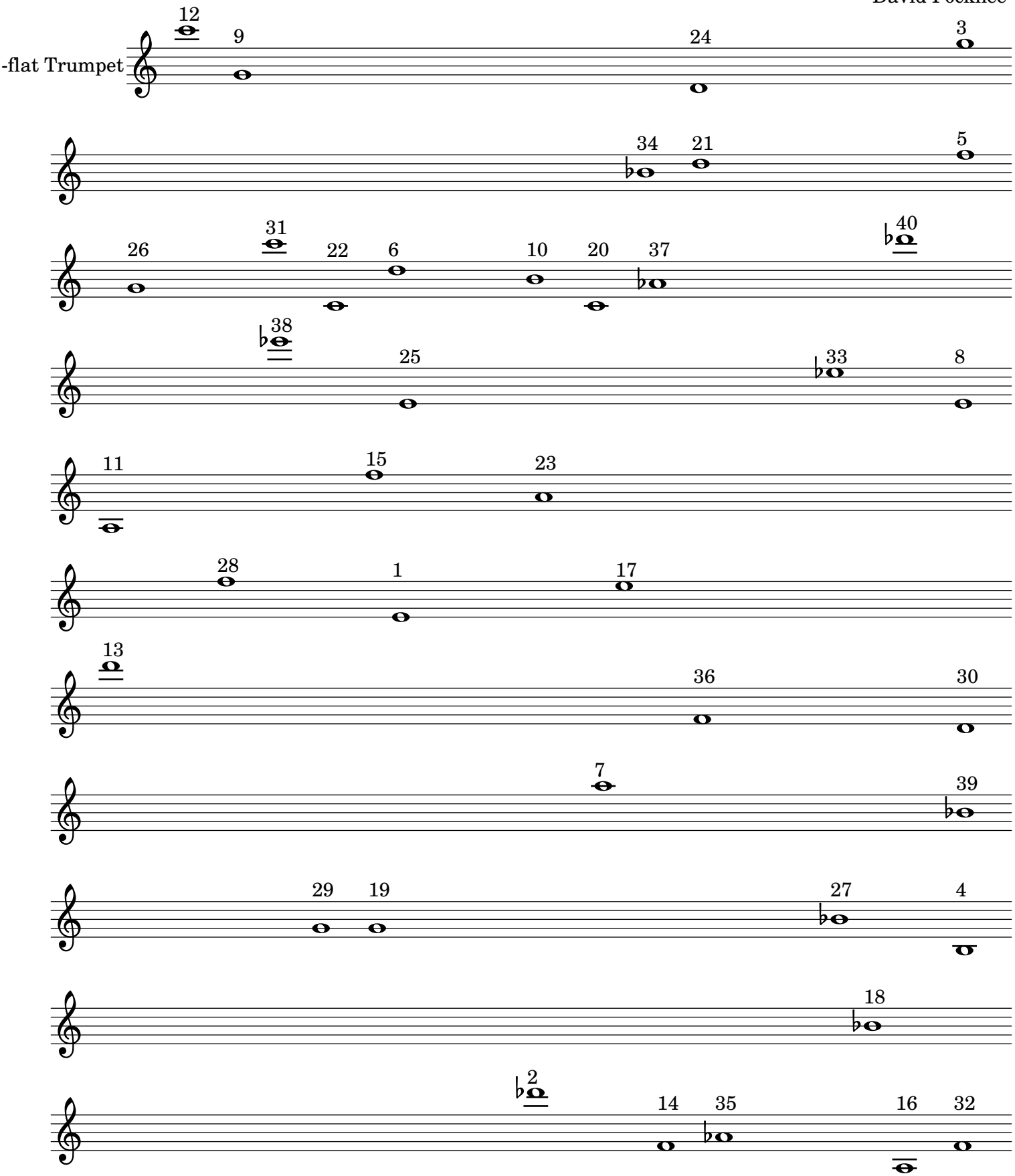
David Pocknee



Chronotopes 10_02_2014-21:56:25 (trumpet-in-Bb part)

for Philip Pocknee

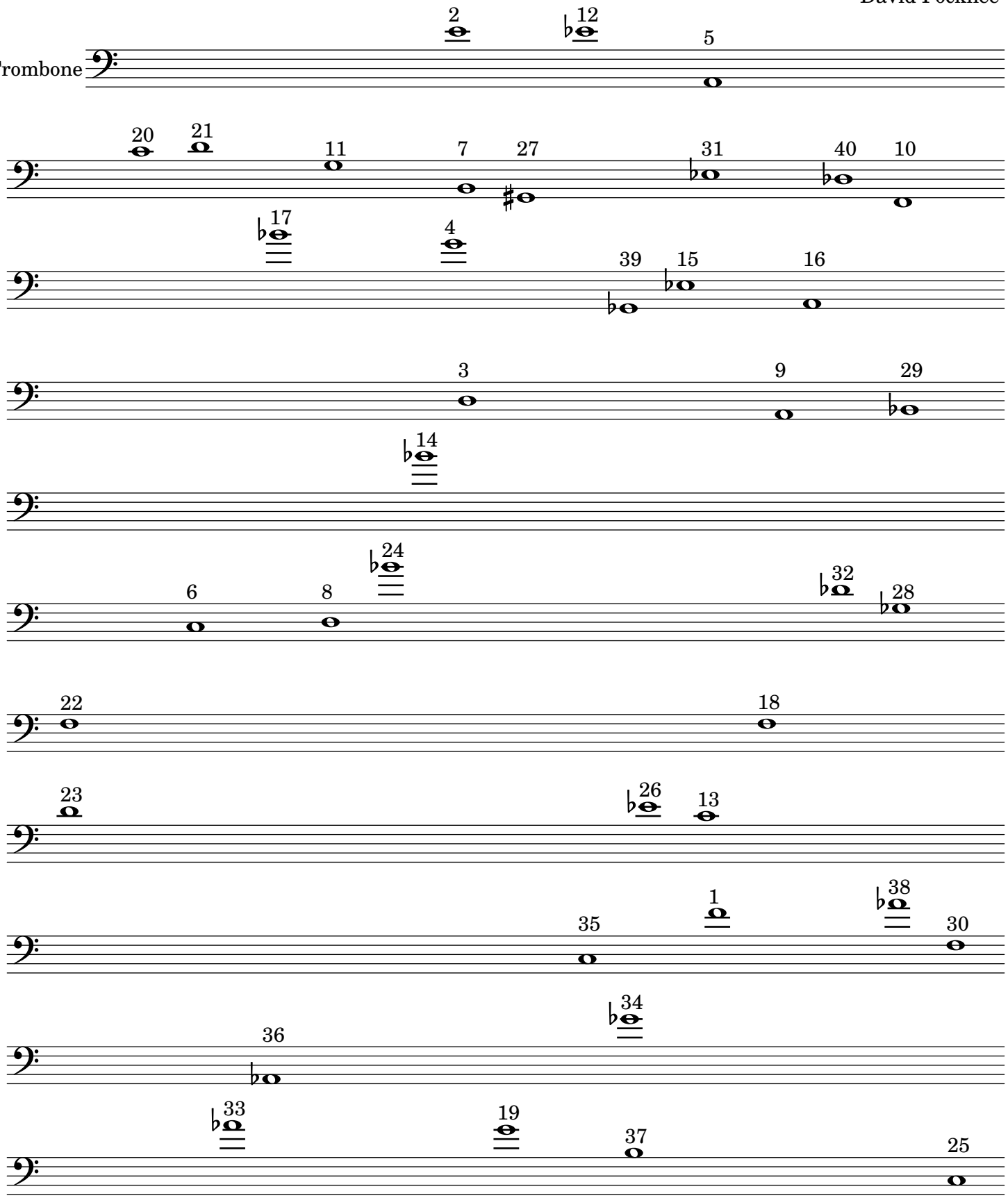
David Pocknee



Chronotopes 10_02_2014-21:56:25 (tenor-trombone part)

for Philip Pocknee

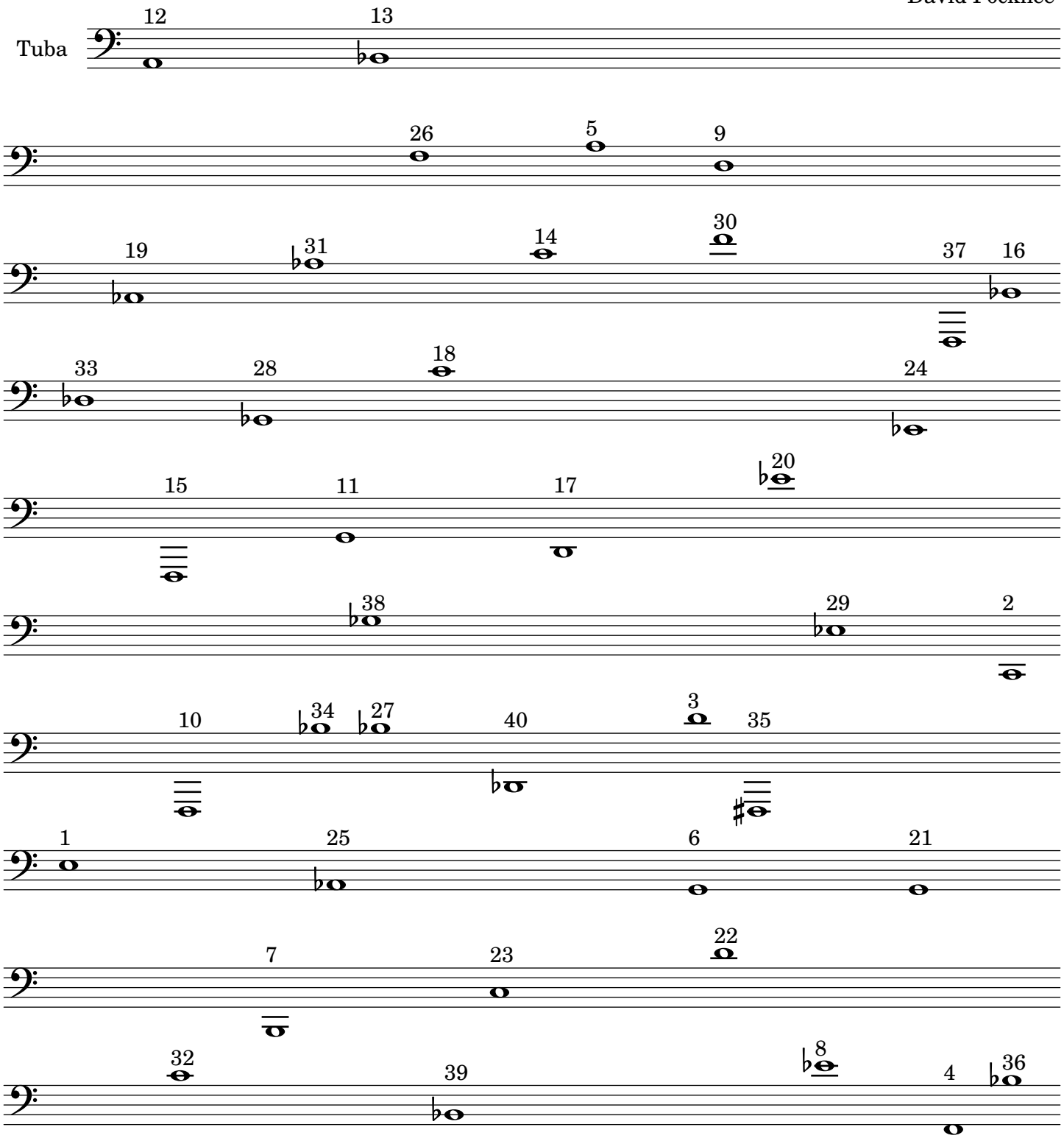
David Pocknee



Chronotopes 10_02_2014-21:56:25 (tuba part)

for Philip Pocknee

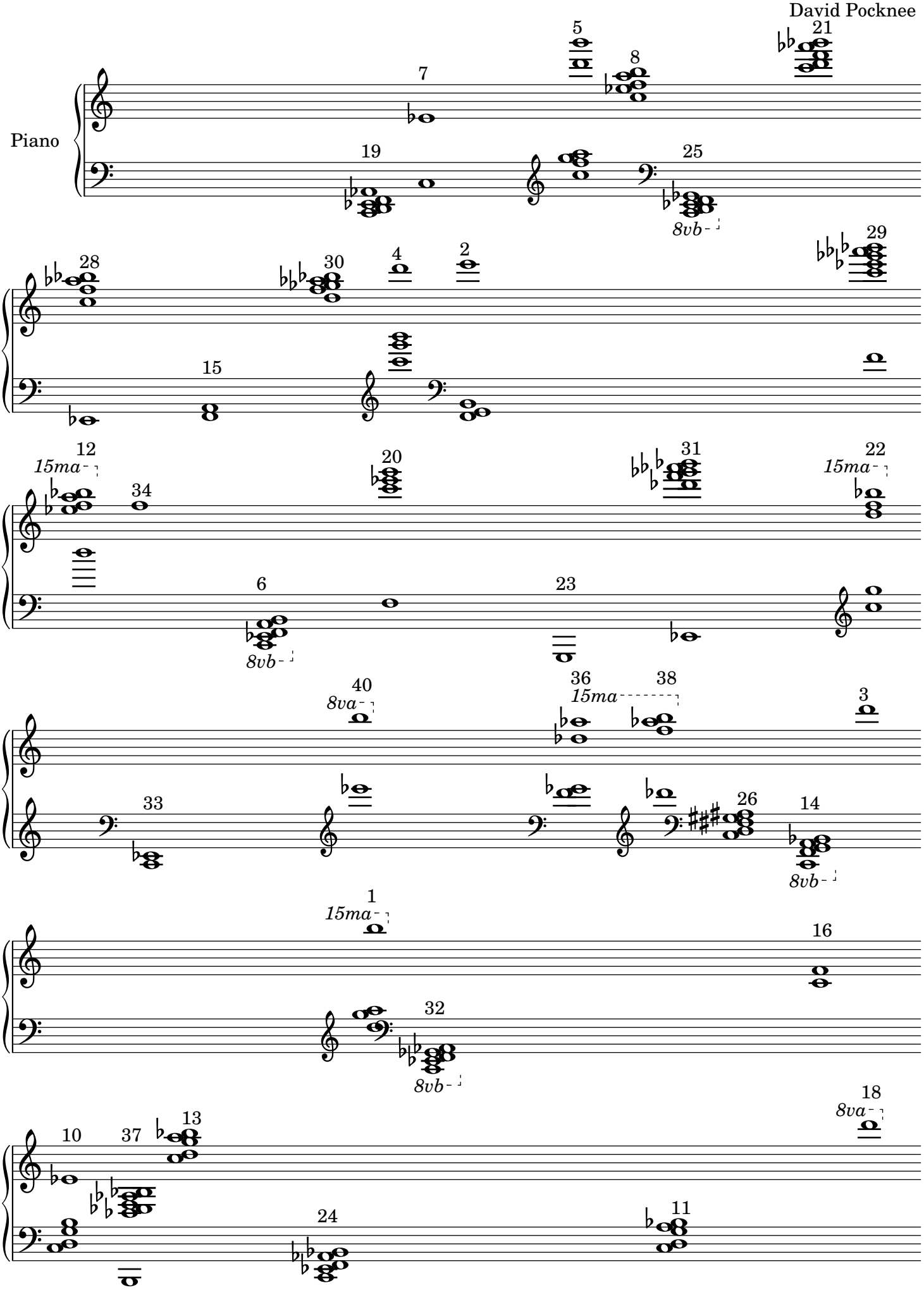
David Pocknee

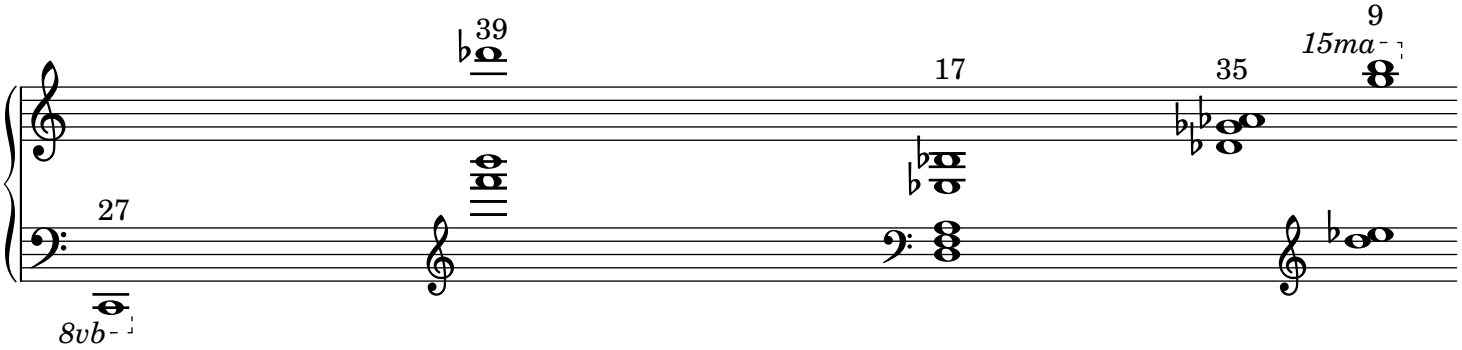


Chronotopes 10_02_2014-21:56:25 (piano part)

for Philip Pocknee

David Pocknee

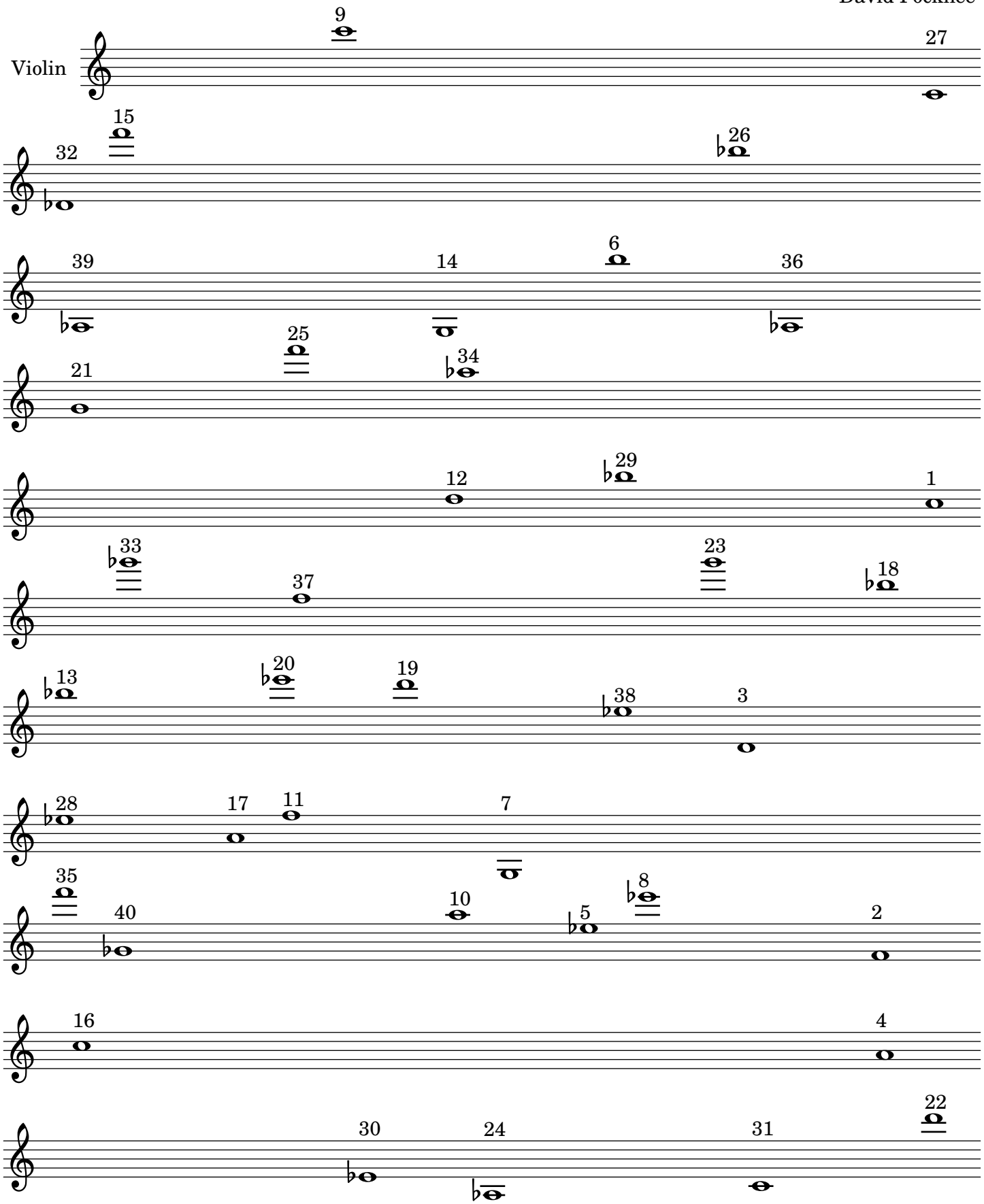




Chronotopes 10_02_2014-21:56:25 (violin_I part)

for Philip Pocknee

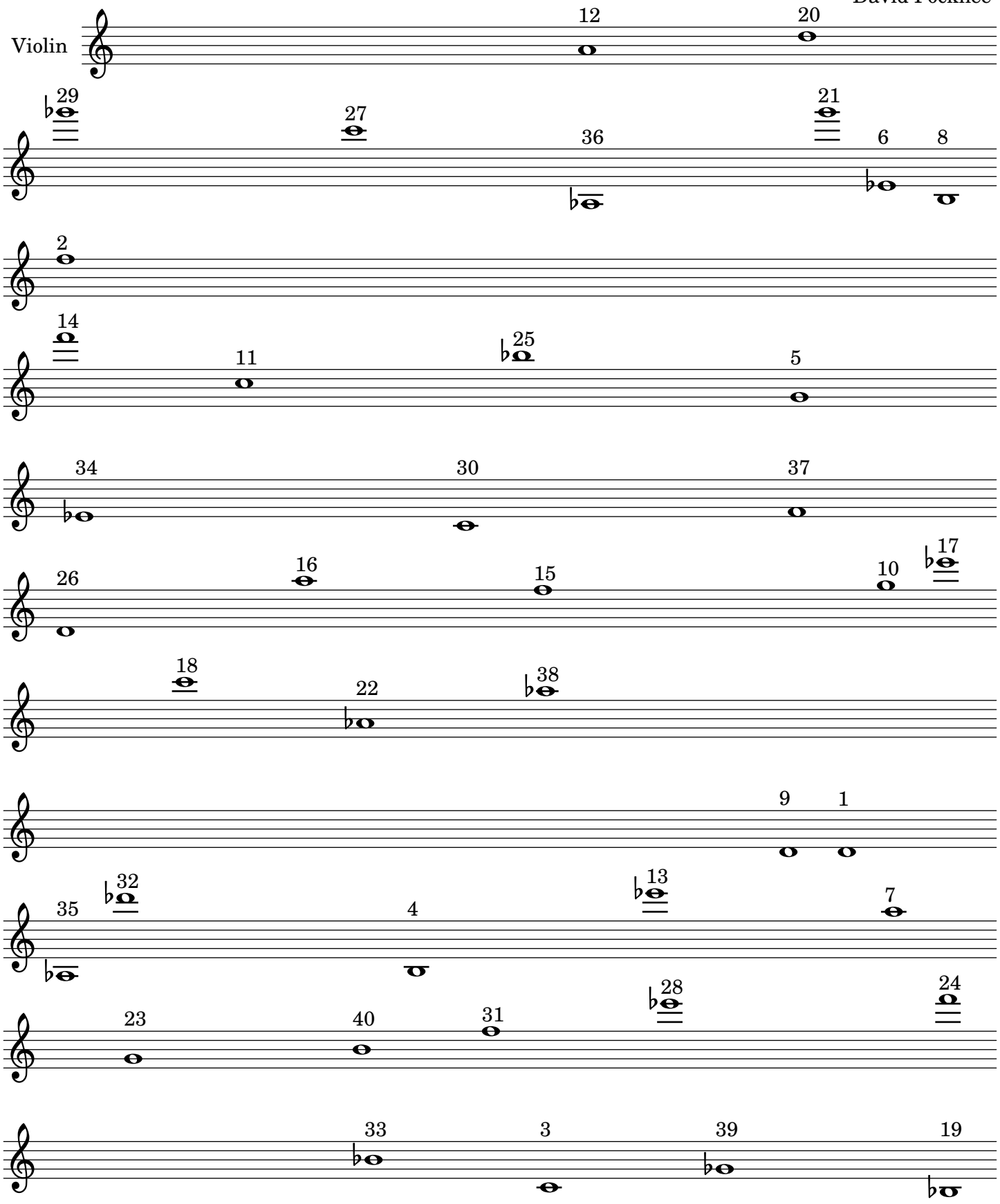
David Pocknee



Chronotopes 10_02_2014-21:56:25 (violin_II part)

for Philip Pocknee

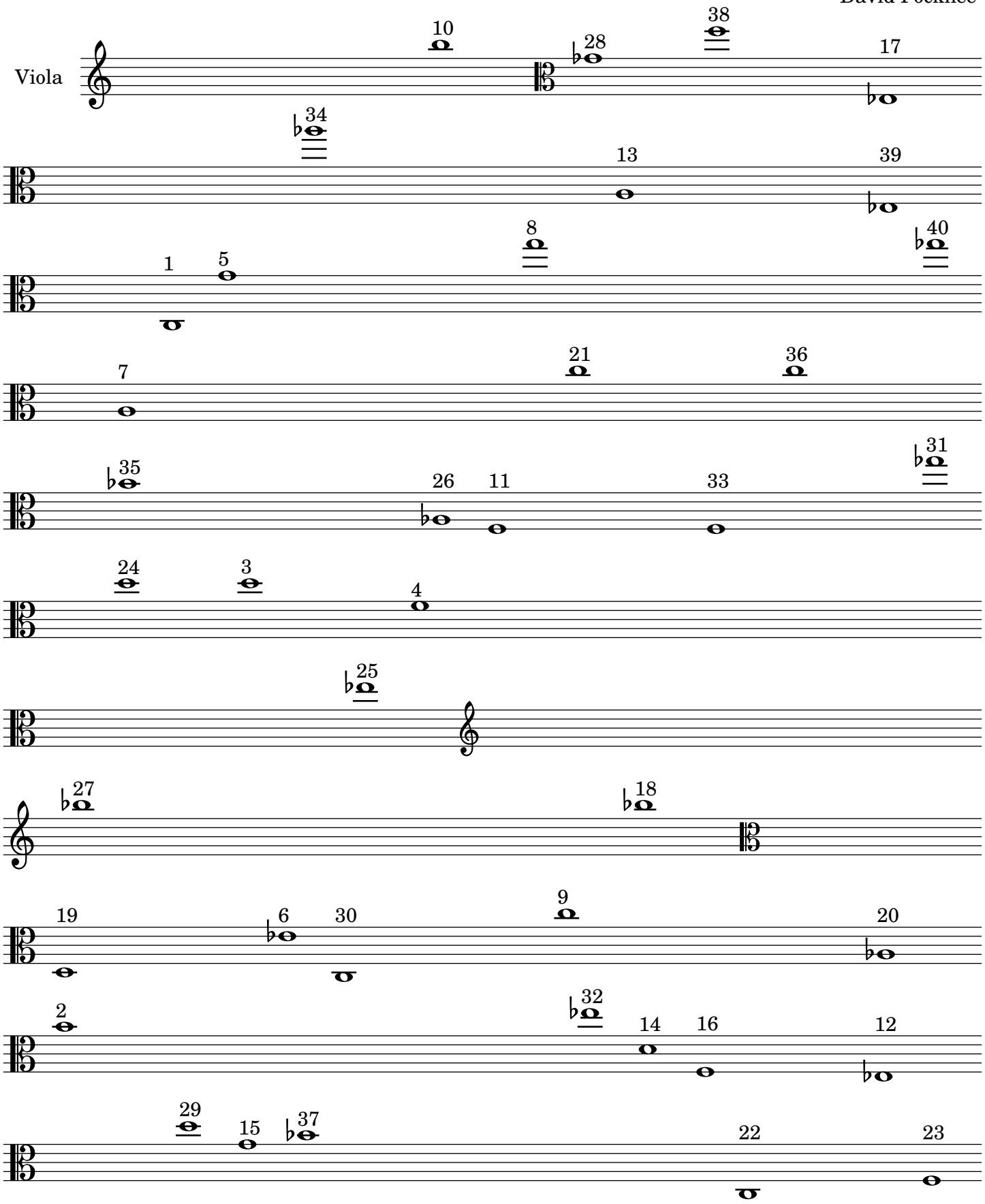
David Pocknee



Chronotopes 10_02_2014-21:56:25 (viola part)

for Philip Pocknee

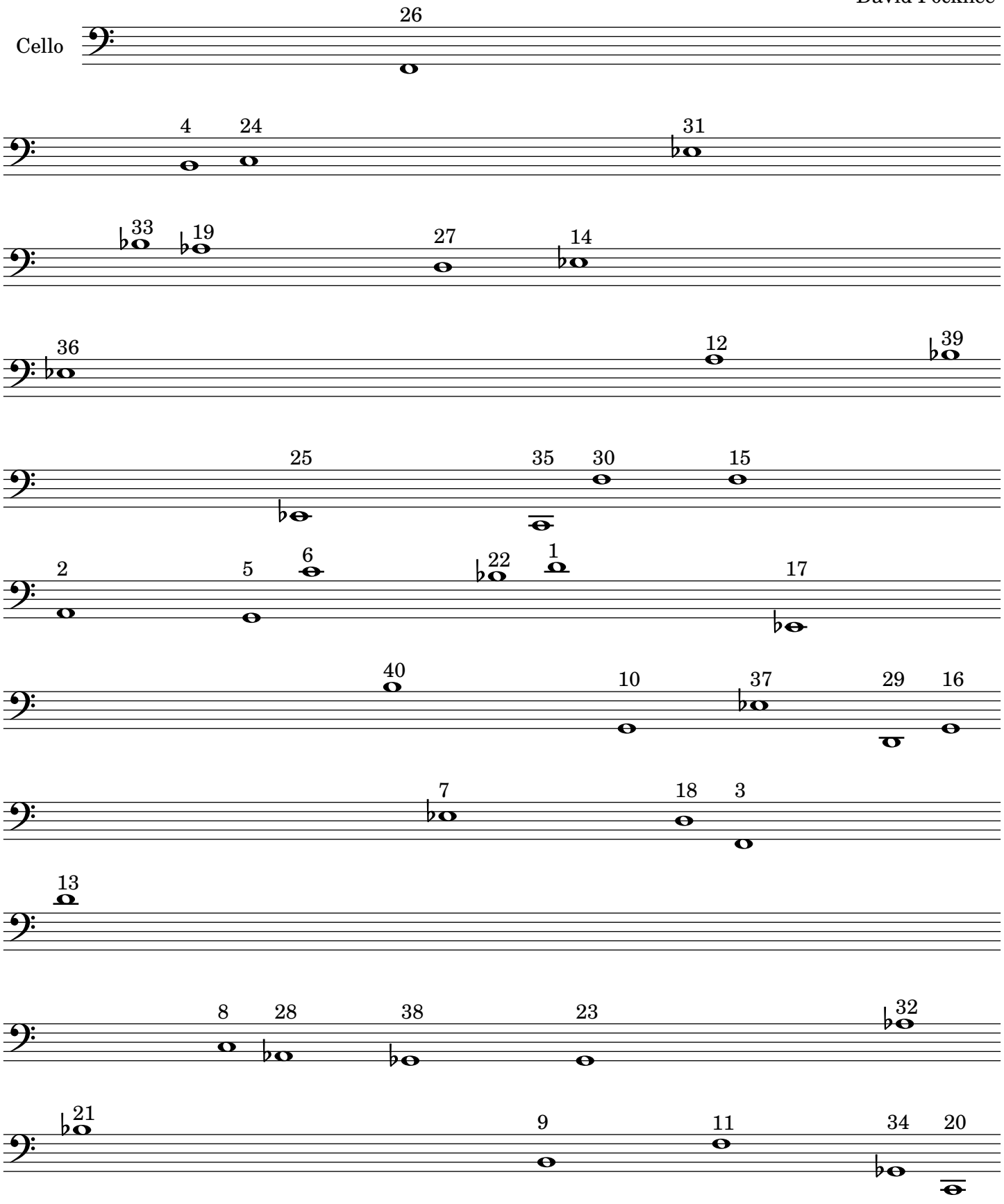
David Pocknee



Chronotopes 10_02_2014-21:56:25 (violoncello part)

for Philip Pocknee

David Pocknee



Chronotopes 10_02_2014-21:56:25 (contrabass part)

for Philip Pocknee

David Pocknee