Informatics for Musicology (IPM) 2024/25

Jupyter Notebooks

Teacher: JN Oliveira

Department of IT at U. Minho, in collaboration with ENSICO

26-Nov class : TP class on IPM libraries in Haskell

> **Summary** Continued exploration of IPM (Haskell) libraries for 'Computer Aided Musicology'. Continuation of the study of verticalization operations. Synthesis and manipulation of chords and arpeggios. Examples and case study: *Alegretto* from Beethoven's 7th symphony (1770-1827).

⚠️ **Important** : run without moving the next cells.

In [ ]:
```
: opt  no - lint
: m  Data . Char
: m  Date . List
: m  Date . List . Split
: m  Data . Ratio
```

Modules developed for the discipline:

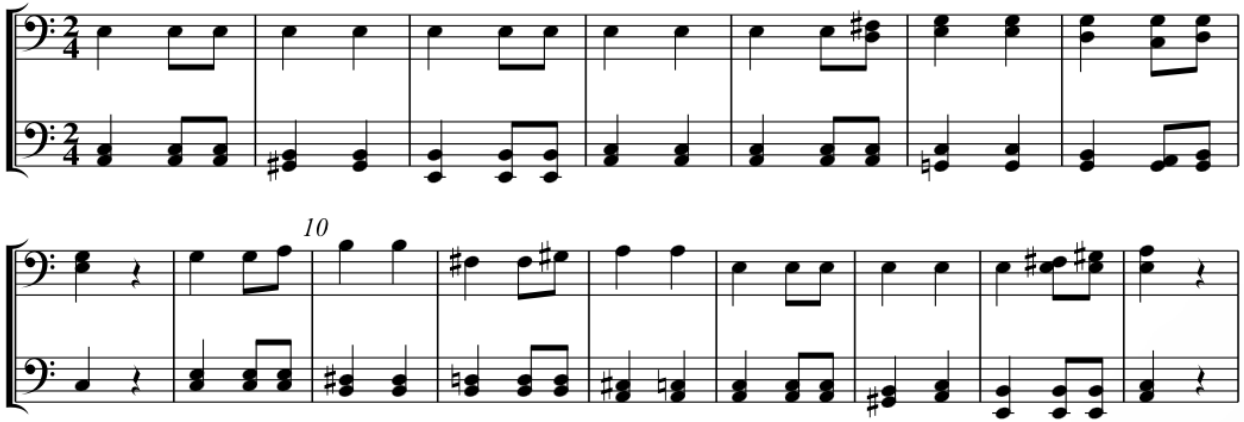
In [ ]:
```
: l  ../ src / Cp . hs
: l  ../ src / Reducer . hs
: l  ../ src / Ipm . hs
: l  ../ src / Abc . hs
```

Data ("case studies"):

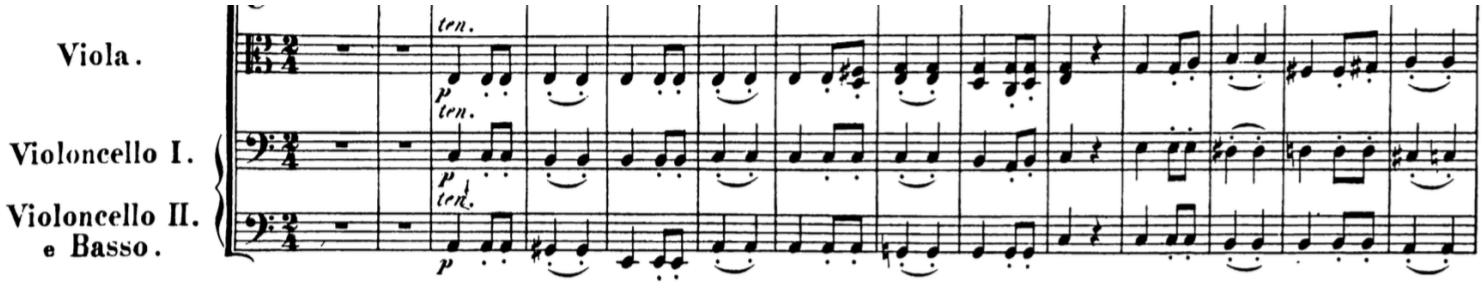
In [ ]:
```
: l  ../ src / CS . hs
```

🔴Verticalization

The following reduction of the beginning of the second movement (*Alegretto*) of Beethoven's 7th symphony



was obtained from the original score - viola, cello and bass parts -



through functions whose study will be the main objective of this *notebook* .

11.1 The musical fragment in question is available in the form of a list `op93ii` of four parts, namely:

In [ ]:
```
( abcPlayM  "C"  "2/4"  .  P )  op93ii
```

Before running the following cells, indicate the meaning of what you hope to obtain. What information can these cells give you about `op93ii` ?

In [ ]:
```
     op93ii
length op93ii map  length
```

In [ ]:
```
rhythm  =  map  snd
---
( length  .  nub  .  map  rhythm )  op93ii
```

11.2 It is now intended that each part does `op93ii` not repeat successive equal notes. Adapt

```
(abcPlayM "C" "2/4" . P) op93ii
```

in order for this to happen, and listen to the result.

In [ ]:
```
( abcPlayM  "C"  "2/4"  ......  )  op93ii
-- abcShow
```

11.3 - We can decompose the list `op93ii` into its four parts. Complete the following cell to obtain the missing ones:

In [ ]:
```
v1  =  head  op93ii
v2  =  .....
vc  =  .....
ba  =  last  op93ii
```

11.4 - Starting by defining

In [ ]:
```
beet  =  abcPlayM  "C"  "2/4"
r  =  rhythm  ba
```

run the following cells and observe the results. At the end, say in your own words what the function does `dchords` .

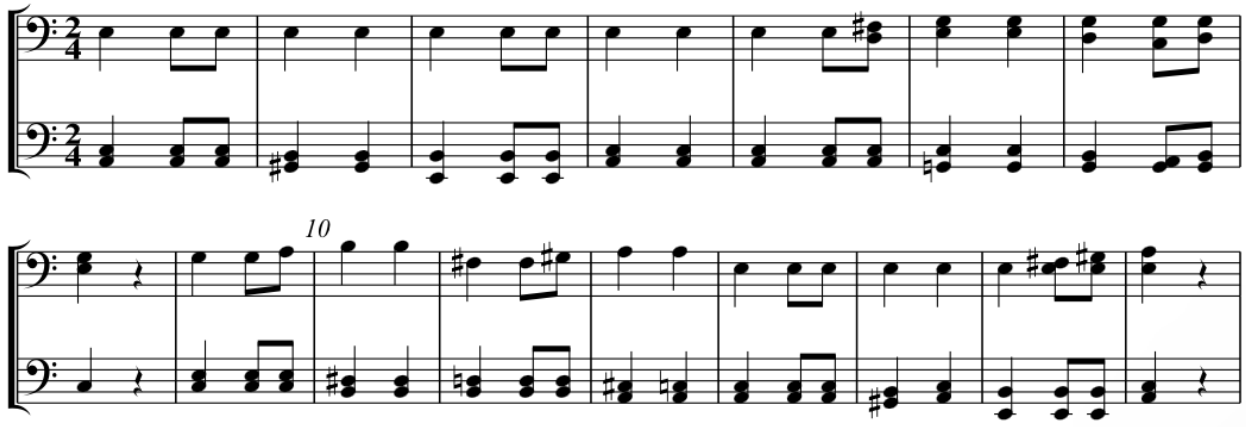
In [ ]:
```
( beet  .  dchords  r )  op93ii
```

In [ ]:
```
( beet  .  dchords  bin )  op93ii
```

In [ ]:
```
s  =  3 % 8  :  1 % 8  :  s
---
( beet  .  dchords  s )  op93ii
```

So what is the difference between `dchords` and `chordify` ? Answer in the next cell.

11.5 - Now program the next cell to obtain the reduction `op93ii` shown at the beginning, that is:



In [ ]:
```
....
---
( beet  .  P )  ....
```

11.6 - Finally, compare what was obtained above in

> `beet (dchords s op93ii)`

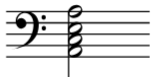
with the result in the next cell - what is the relationship between `dvert` and `dchords` ?

In [ ]:
```
dvert  bin  op93ii
```

11.7 - Above, `dvert` produced sequences of several notes for the same duration, e.g.

> `(["A,","E,","C,","A,,"],1 % 2)`

which we can consider a good representation of the notion of **chord** . In fact,



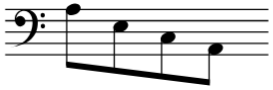
this is what you get by running the next cell, converting the chord to Abc notation.

In [ ]:
```
abch  ([ "A," , "E," , "C," , "A,," ], 1  %  2 )
---
abcplease  it
```

11.8 - In the opposite direction, we can now think about arpeggiating a given chord, for example, converting the chord

> `(["A,","E,","C,","A,,"],1 % 2)`

in



This is what you'll get by running the next cell:

In [ ]:
```
harp  ([ "A," , "E," , "C," , "A,," ], 1  %  2 )
```

11.9 - Analyzing the details of the function definition `harp`

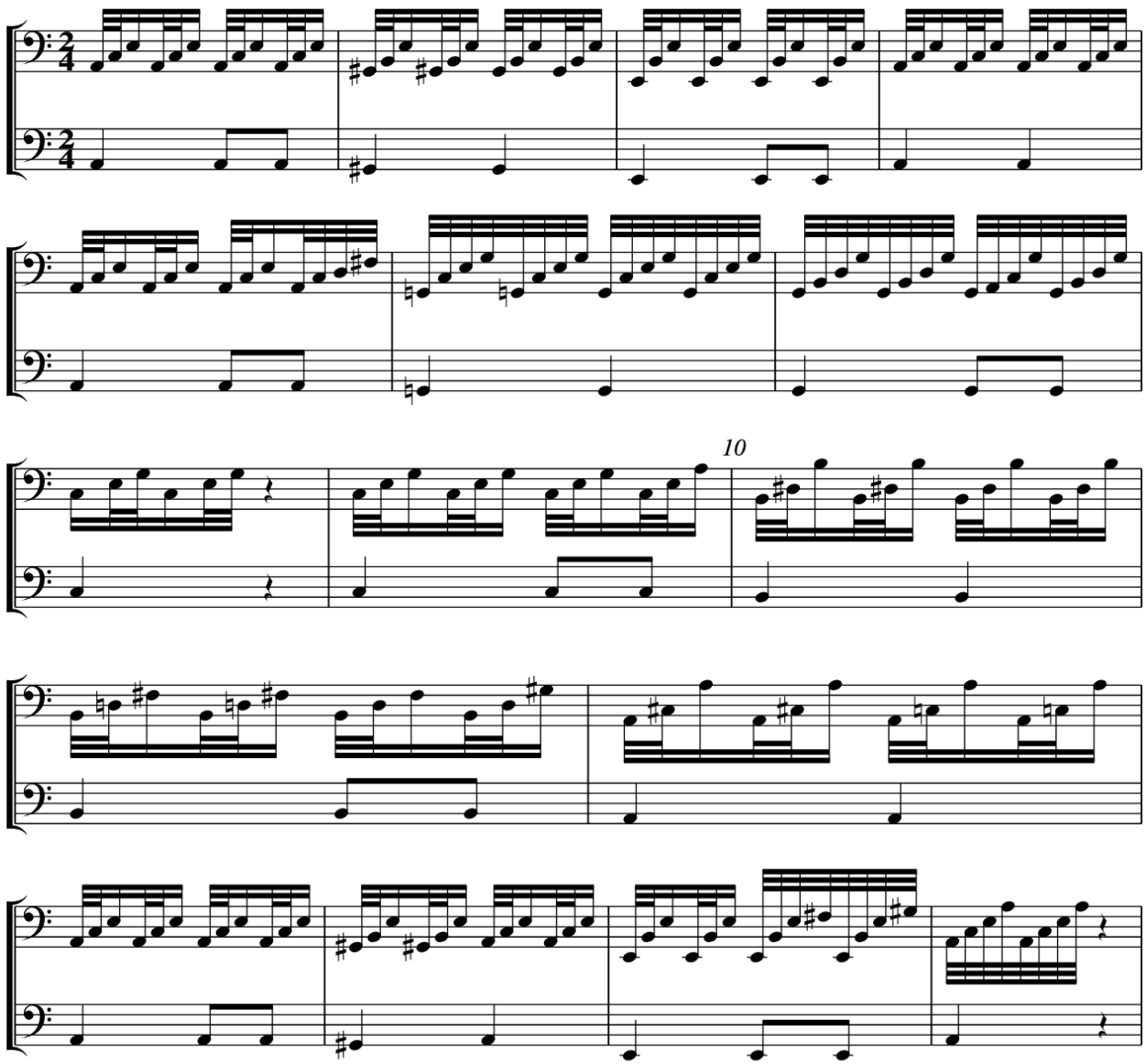
```
harp(ns,d) = zip ns s where s = (d*(1%length ns)) : s
```

used in the previous cell, explain how it works in your own words. What is the role of `s` in the definition?

In short:

Designation	Meaning	Detailed description
`chordify`	chords (horizontal axis)	`chordify s m` groups the notes `m` into chords according to prescribed durations in `s`
`dvert`	verticalization	`dvert d [a,b,..]` vertically groups the notes of several melodic lines `[a,b,..]` seamlessly sampled by `d`
`dchords`	chords (vertical axis)	`dchords d [a,b,..]` performs a verticalization and converts it into chords according to Abc notation
`dchunksOf`	divide into blocks	`dchunksOf d m` divides melody `m` into a sequence of melodies according to durations specified in `d`
`harp`	arpeggiate	`harp (n,d)` arpeggiates the chord `(n,d)` by dividing the duration `d` by the number of notes `n`

11.10 - (**Synthesis** - to be carried out after class) Project in the cell below the necessary steps to, using all the operators studied so far, generate from `op93ii` the "variation" that is shown in the upper pentagram of:



In [ ]: