POLYMORPHIC VPS CLASS

Marco Stroppa

March 1997

IDEA: create a super-class called VPS (Vertical Pitch Structure) made of 7 classes corresponding to the different representations described below.

All the classes will have specific constructors (user-defined), selectors, modifiers, predicates and info functions.

Some selectors can be processed directly within a given class (i.e. the information is immediately accessible in the internal fields). Ex: (get-pitch-class <list-of-pitches>).

Other selectors (called <u>indirect selectors</u>) may require an on-line conversion from a class to another where the field is directly specified. Theses classes, called <u>virtual definitions</u>, do not change the internal "real" definition. Ex: (get-pitch-class <spectrum>) implies the conversion of the spectrum into a sequence of pitches.

By rule, the only definition that will be hard-encoded is the definition specified by the user. Modifiers that require a change of definition are NOT allowed (the user should first instanciate a new object by specifically converting the old one to a new class).

GLOBAL CONSTRAINTS

All the classes will be made of a sequence of items (frequency, pitch or intervals) in ascending order and not repeated.

All the conversions symbolic pitch -> frequency [Hz] are relative to the global variable (get-gbl 'DIAPASON).

In general, spectra coming from analysis (list of frequencies) and chords will tend to differ with respect to their number of components. It is normal for spectra to have between 10 and 100 components, whereas a chord will have between 3 and 10/20 components maximum. Some processing functions will also not be the same (i.e. "stretch" refers rather to spectra than to chords). In spectra the components will naturally have different amplitudes.

REMARK

Constructor (make ...): create a new object
Selector (get-...): extracts some information from a given object's fields
Modifier (set-...): physically modifies a given object's field
Predicate (is-...): boolean function
Info: mixed information about the object (pretty printing)

REPRESENTATION OF ITEMS

(SYMBOLIC) PITCH

Uses Italian spelling (DO, RE, MI, etc.).

sharp is "**d**", **flat** is "**b**".

octave indication according to **American standards**, middle A is octave "4"

Complete indication: either be a **symbol** ‘DO3, ‘SIb4, etc. or a **list (pitch freq-deviation)**, where "**pitch**" is the symbol above (with octave indication, expressed along the well-tempered scale) and "**freq-deviation**" indicates a deviation from the reference pitch (usually a number [**± cents**]; as a special case, also possible as a symbol "**+q**" or "**-q**" meaning ± one quarter tone).

<u>**Ex**</u>. of possible pitches: DO5, REd3, (SIb4 13), (MI3 -3.3), (LA4 -q), (REb +q)

***μ*-note:** should create a small "dictionary" allowing for alternative notations of each pitch according to Italian, German, American and French spelling. However, careful checking is needed. Ex: (setf DES ‘REb), (setf AIS ‘LAd), (concat DES 4), ...???

SYMBOLIC INTERVAL

Notation of **intervals** is either a **symbol** [±itvl] or a **list**: **(±itvl ±oct [±freq-deviation]).**

Where: "**itvl**" = "**-**" (descending [optional])

1 to 7 = unison to seventh

"**±**" following the interval's size means "**minor/major**" intervals if size is 2,3,6 and 7; "**augmented/diminished**" if 1, 4 and 5.

±freq-deviation = frequency deviation in **cents** or with **±q**.

"**augmented/diminished**" intervals for size 2,3,6 and 7 are not allowed and ought to be enharmonically corrected.

<u>**Ex.**</u> of intervals: 6-, (3+ 2), (3- 0 +13), (4- -2 -7.5), (7+ 2 +q), etc. Same intervals: ascending minor ninth: (2- 1) or (-7+ 2), descending minor second: (-2-) or (7+ -1), etc.

MIDICENTS

Recognized automatically if the value is integer and > 127

NUMERIC INTERVAL (RATIOS)

±(freq[high] / freq[low]) - 1

Ex: ascending perfect fifth = 330/220.= 1.5 - 1 = 0.5
descending perfect fifth = -(330/220).= -(1.5 - 1) = -0.5
ascending octave = 440/220.= 2 - 1 = 1
descending octave = -(440/220).= -(2 - 1) = -1
arbitrary relation = 234/129.= 1.814 - 1 = 0.814

CLASSES

SPL (Symbolic Pitch List): sequence of absolute pitches
RPL (Relative Pitch List): sequence of relative pirches
CIL (Contiguous Interval List): sequence of contiguous intervals
AIL (Anchored Interval List): sequence of intervals from a given ptich/freq
FQL (Frequencies): sequence of frequencies [Hz]
CRL (Contiguous Ratios): sequence of ratios between adjacent frequencies
ARL (Anchored Ratios [Spectrum]): sequence of ratios from a given F0

SPL, RPL, CIL, AIL will rather represent chords; **FQL, CRL, ARL** spectra.

CONSTRUCTORS AND EXAMPLES OF REPRESENTATION

SPL

List of pitches according to format above, including octave number and microtonal deviations.

```
;       NAME:                   CONSTRUCTOR
;       TYPE:           Expr with 2 arguments
;       CALL:           (make ‘SPL apl)
;       FUNCTION:               create an object of class APL and super-class VPS
;       VALUE:                  the object
;       SOURCE:                 $LLvps/vps.ll

; EX1: (setf my-spl (make ‘SPL ‘(DO4 LAb4 RE5 SOL5 REb6)))
; EX2: (setf my-spl (make ‘SPL ‘ ((DO4 +12) (LAb4 -5) RE5 SOL5 (REb6 -q))))

; TESTS         Format of pitches (corrrect spelling + presence of octave)
;               Ascending order
```

RPL

The same as above, but octave number is eliminated or, if it appears it refers to the distance from the beginning of the VPS.

```
;       NAME:                   CONSTRUCTOR
;       TYPE:           Expr with 2 arguments
;       CALL:           (make ‘RPL rpl)
;       FUNCTION:               create an object of class RPL and super-class VPS
;       VALUE:                  the object
;       SOURCE:                 $LLvps/vps.ll

; EX1: (setf my-rpl (make ‘RPL ‘(DO LAb RE1 SOL1 REb2)))
; EX2: (setf my-rpl (make ‘RPL ‘ ((DO +12) (LAb -5) RE1 SOL1 (REb2 -q))))

; TESTS         Format of pitches (corrrect spelling only)
;               Ascending order
```

CIL

List of contiguous intervals, that is intervals between adjacent pitches.

```
;      NAME:              CONSTRUCTOR
;      TYPE:        Expr with 2 arguments
;      CALL:        (make ‘CIL cil)
;      FUNCTION:          create an object of class CIL and super-class VPS
;      VALUE:             the object
;      SOURCE:            $LLvps/vps.ll

; EX1: (setf my-cil (make ‘CIL ‘(6- 4+ 4 4+))
; EX2: (setf my-cil (make ‘CIL ‘ ((6- +7) (4+ 5) 4 (4+ -q)))

; TESTS      Format of intervals (corrrect spelling)
; REMARKS    By definition all the intervals are supposed to be ascending
;            The number of items of a CIL = (number of items of an APL) - 1
```

AIL

List of intervals with respect to an arbitrary reference (pitch or frequency).

```
;      NAME:              CONSTRUCTOR
;      TYPE:        Expr with 3 arguments
;      CALL:        (make ‘AIL ref ail)
;      FUNCTION:          create an object of class AIL and super-class VPS
;      VALUE:             the object
;      SOURCE:            $LLvps/vps.ll

; EX1a: (setf my-ail (make ‘AIL ‘LA4 ‘(-6+ -2 4 7- (3+ 1)))
; EX1b: (setf my-ail (make ‘AIL 200.0 ‘(-6+ -2 4 7- (3+ 1))))
; EX2: [do5 = 523.25]
;      (setf my-cil (make ‘AIL 523.25.‘ ((0 -1 12) (3+ 5) 2 4 (2- 1 -q)))

; TESTS      Format of intervals (corrrect spelling + ascending order)
;            Correct spelling of reference (pitch with or without octave or frequency)
; REMARKS    If the reference is a pitch without octave, the intervals are computed
;            relative to that pitch and expressed symbolically.
;            If the reference is a frequency, the intervals are always expressed
;            symbolically.
;            The number of items of an AIL = (number of items of a CIL) + 1
```

FQL

List of frequencies [Hz], of corresponding amplitudes [db] [optional, default = 0.0db = max] of corresponding bandwidths [Hz] [optional, default = no bandwidth]. This representation is better suited to express spectra issued from analysis data.

```
;      NAME:              CONSTRUCTOR
```

; TYPE: Expr with 2/4 arguments
; CALL: (make ‘FQL fql [amp bw])
; FUNCTION: create an object of class FQL and super-class VPS
; VALUE: the object
; SOURCE: $LLvps/vps.ll

; EX1: do4 = 261.63, lab4 = 415.3, re5 = 587.33, sol5 = 784, dod6 = 1108.7
; (setf my-fql (make ‘FQL ‘(261.63 415.3 587.33 784 1108.7)))

; TESTS Format of frequencies (number)
; Ascending order
; The corresponding amplitudes and bandwidths must have the same number of items, otherwise an error is triggered.

CRL

List of ratios of frequencies between adjacent intervals. <=0 (descending, impossible), (0 = unison, 1 = octave), of corresponding amplitudes [db] [optional, default = 0.0db = max] and of corresponding bandwidths [Hz] [optional, default = no bandwidth].

Freq(x+1) = freq(x) + freq(x) * ratio(x)

; NAME: CONSTRUCTOR
; TYPE: Expr with 2 arguments
; CALL: (make ‘CRL crl)
; FUNCTION: create an object of class CRL and super-class VPS
; VALUE: the object
; SOURCE: $LLvps/vps.ll

; EX1: (setf my-crl (make ‘CRL ‘(0.5873 0.4142 0.3345 0.4145)
; (do4 = 261.63, lab4 = 415.3, re5 = 587.33, sol5 = 784, dod6 = 1108.7)

; TESTS Format of intervals (corrrect spelling + ascending order)
; Correct spelling of reference (pitch with or without octave or frequency)

;REMARKS The number of items of a CRL = (number of items of a FQL) - 1

ARL

List of ratios with respect to a F0. Freq(x) = F0 * ratio(x) of corresponding amplitudes [db] [optional, default = 0.0db = max] and of corresponding bandwidths [Hz] [optional, default = no bandwidth].

; NAME: CONSTRUCTOR
; TYPE: Expr with 3 arguments
; CALL: (make ‘ARL F0 arl)
; FUNCTION: create an object of class ARL and super-class VPS
; VALUE: the object

```
;        SOURCE:              $LLvps/vps.ll

; EX1a: (setf my-arl (make ‘ARL 100.0 ‘(2.6163 4.153 5.8733 7.84 11.087)))
; (do4 = 261.63, lab4 = 415.3, re5 = 587.33, sol5 = 784, dod6 = 1108.7)

; EX1b: (setf my-arl (make ‘ARL ‘LA5 ‘(0.2973 0.4719 0.6674 0.8909 1.2599)))

; TESTS       Format of ratios (number + ascending order).
;             Correct spelling of reference (pitch with or without octave or frequency).

; REMARKS     In an AIL intervals are expressed symbolically, in a CRL the are ratios.
;             In both cases the F0 can be either numeric or symbolic.
;             The number of items of a ARL = (number of items of a CRL) + 1
```

THESHOLDS

Chords tend to have fewer items than spectra. The global variable *MAX-NN* gives the estimated average maximum number of notes a chord is made of. Above this value, it is more likely a spectrum. This is used, for example, when converting spectra to chords: by default the converted chord will have *MAX-NN* notes.

To differenciate between a FQL and a CRL (they both have a list of floats) one will use the global variable MINFQ. If (< MINFQ (max <list of floats>)) it is a list of frequencies (FQL), otherwise it is a list of contiguous ratios (CRL).

Could add a method to rescale a spectrum with Fletcher&Munson curves.

FIELDS (attributes)****

name / Common Lisp form / explanation

contents: (DO4 LAb4 RE5 SOL5 DOd6); list of pitches with octave value according to format above.

note-list: (DO LAb RE SOL DOd); list of the pitches without octave number.

CIL (Contiguous Interval List): (
AIL (Anchored Interval List):
GIL (Global Interval List):

NN (Number of Notes):
NCIL / NAIL / NGIL (Number of intervals):

surface
density

hom (Coefficient of Homogeneity):
st-dev (standard deviation):
CS (Coefficient of Stability):_

first/ / last /nth item

MAIN METHODS****

*****

POLYMORPHIC TRANSFORMATIONS****

What to do to:

1 VPS
2 VPS
etc.

****

Modification of the "direct" fields (those contained in the "real" definition) is operated directly. For indirect fields chose one of three possible alternatives:

1. impossible -> refusal + warning
2. possible, but modification is stored as such, without changing the internal representation, so that a "history" of the sequence of modifications is accessible
3. modification is done and its influence is converted back into the direct representation, which may imply some irretrievable loss of data