CHROMA
Vertical Structures: Pitches and Spectra

User's Guide
and
Musical Applications

Marco Stroppa, Paris
May 1998

INTRODUCTION
This text presents that part of Chroma that is dedicated to chords and spectra and to their implementation as polymorphic CLOS classes. It will follow a step-by-step bottom-up process raising from low-level concepts to my usage in both instrumental music (i.e. more standard CAO) and electronics music (synthesis control).

LOADING THE SYSTEM
This system is a part of Chroma and is usually automatically loaded when calling Chroma. The source files are contained in the directory pointed at by the global variable "LLvps" and is loaded by loading the file "ld-vps.lisp" contained in that directory. However, the file "defgbl.lisp" is also needed, for it contains the global variables.

List and contents of the files
gbl-vps.lisp: contains all the constants of the system (the ones specific to VPS)
symbolic-pitch.list: implements the class "symbolic-pitch"
pitch-conversions.lisp: specifies all the available pitch converters for the class "symbolic-pitch"
utils-vps.lisp: contains some utilities needed for compatibility (coming from sys-vps.ll, the old implementation of the VPS's in LeLisp and from Esquisse - harmonicity and virtual fundamental) and some predicates
vps.lisp: implements the classes VS and VPS
chord.lisp: implements the classes CHORD, SPL, RPL, PPL, CIL and AIL
spectrum.lisp: implements the classes SPECTRUM, FQL, CRL, ARL
user-vps.lisp: macro layer for the above object oriented system
modif-vps.lisp: several functions modifying VS's or some specific representations
match.lisp: matching algorithm applied to VS's

PITCHES AND INTERVALS: REPRESENTATION
Pitches are represented using either the Italian or the American spelling. The two spellings can be mixed within a single chord. It's a little confusing, but there is no problem for the system.

The class "symbolic-pitch" is used as the internal representation and is not to be addressed directly by the user. Normally the pitches are part of a VPS and are rarely employed alone.
EX: (setf note1 'DO)

Pitches
Possible names for pitches without alterations are: DO/UT/C, RE/D, MI/E, FA/F, SOL/G, LA/A, SI/B.

The alterations must be attached directly to the pitch without any space. They can be the following letters: “b, B, f, F” for flat and “d, D, s, S” for sharp.

An octave number can follow the pitch and transform it into an absolute pitch. When no octave is specified, the octave “0” is taken by default.

All the pitches may either be defined as lisp symbols (setf pitch ‘DO) or as lisp strings (setf pitch “DO”). Hence, valid representations are: DOd, “Db2”, MIB4, FS, “Ab2”, etc.

Micro tones
Micro tones are represented by a lisp cons, whose car is a valid pitch symbol and whose cdr is either an integer number or the symbols “q” or “-q”. The integer number indicates a deviation in cents, the symbols represent a quarter-tone higher and lower than the symbolic pitch.

Valid microtonal representations are: (REb . q), (“Af” . -6), (REb2 . 10), (“Gd4” . -q), etc.

Intervals
Intervals can be represented in the following ways:

REDUCED
• one of the these symbols: 1, 2-, 2+, 3-, 3+, 4-, 4, 4+, 5-, 5, 5+, 6-, 6+, 7-, 7+
They mean: unison, minor second, major second, minor third, major third, diminished fourth, ordinatry fourth, augmented fourth, etc.
These intervals are always well-tempered and within one octave.

EXPANDED
• a list containing one of the symbols above, an octave number and an optional frequency deviation in cents.
The octave number gives the distance in octaves. Hence, a minor ninth would be written (2- 1), meaning a minor second one octave apart. In the frequency deviation, the symbols “q” and “-q” are not accepted.

EX of valid intervals: 3+, (3+ 2), (4+ 0 -12), 5-, etc.

PITCH CONVERSIONS
Different conversions are available. They are all in the form [source]**->**[destination] (ex. fq->midi). Unavailable direct conversions should use two direct conversions (ex. to convert from frequency to interval, one should convert from frequency to ratio or pitch and the from ration or pitch to interval).

Available representations:

• **fq:** frequency [Hz].
• **pch**: pitch (see above).
• **itvl**: interval (see above).
• **midi:** midi (with decimals if non tempered values are needed).
• **ratio**: the numerical ratio between the second and the first argument.
• **semitones**: the number of semitones (positive if ascending, negative if descending) between the first and the second argument.

Summary of the available conversions

• = yes, - = not available

------->	fq	pch	midi	ratio	itvl	semitones
fq	\	*	*	*	-	-
pch	*	\	*	-	*	*
midi	-	*	\	-	-	*
ratio	*	-	-	\	*	*
itvl	*	-	*	*	\	*
semitones	-	-	-	*	*	\

FREQUENCY
(fq->pch freq [approx])
freq: single frequency or list of frequencies
approx: approximation when converting [cents]. An approximation of 50 will round the result up to a quarter tone. Default: 0 (=> the highest precision).
Remark: one can also include symbolic
EX: (fq->pch 443), (fq->pch '(441 442 443) 25), (fq->pch '(441 DO2 (RE3 . 23) 460) 25) [also possible, pitches will not be changed!], etc.

(fq->midi freq)
freq: single frequency or list of frequencies
EX: (fq->midi 443), (fq->midi '(441 442 443)), etc.

(fq->ratio freq)
freq: list of at least 2 frequencies
Result**:** always a list of N-1 elements
NB: values are always positive unless one of the frequencies are negative. Values between 0 and 1 indicate descending ratios, 1 being the unison.
EX: (fq->ratio '(443 552 234)), etc.

PITCH
NB: The argument of the "pitch" conversion can always be either a pitch or a number, usually meaning a frequency [Hz]. In this case, however, the value will be treated differently depending on the conversion.

(pch->fq pitch [diapason])
pitch: single pitch or list of pitches or frequencies. Frequencies will return the same unconverted value.

diapason: reference frequency for “LA4” [Hz]. Default: 440.
EX: (pch->fq “DO2”), (pch->fq ‘(DO3 . -23)), (pch->fq ‘(RE2 440 (MIb3 . 23) 550 SI2) 442), etc.

(pch->midi pitch [diapason])
pitch: single pitch or list of pitches or frequencies. Frequencies will return the same unconverted value. However, in this case the list will contain a mixture of midi values (ex. 69, 72, etc.) and frequencies in Hz that is not very useful.
diapason: reference frequency for “LA4” [Hz]. Default: 440.
EX: (pch->midi ‘DO2), (pch->midi ‘(DO2 (RE3 . 8)) 442), etc.

(pch->itvl pitch)
pitch: list of at least 2 pitches or frequencies.
Bug: due to the internal implementation, here the frequencies will be considered MIDI values and not proper frequencies.
Result: always a list of N-1 elements. Interval is always in its expanded form (ex. (2 0) for a minor second).
EX: (pch->itvl ‘(DO2 RE2)), (pch->itvl ‘(LA4 (MIb3 . -12) SI2)), (pch->itvl ‘(“DO3” (“MIb2” . -q) 69 SI2)), etc.

(pch->semitones pitch)
pitch: list of at least 2 pitches or frequencies.
Bug: due to the internal implementation, here the frequencies will be considered MIDI values and not proper frequencies.
Result: always a list of N-1 elements.
EX: (pch->semitones ‘(DO2 RE2)), (pch->semitones ‘(LA4 (MIb3 . -12) SI2)), (pch->semitones ‘(“DO3” (“MIb2” . -q) 69 SI2)), etc.

MIDI
(midi->pch midi [approx])
midi: single midi value or a list of values.
EX: (midi->pch 69), (midi->pch 69.234), (midi->pch ‘(69 71.2 44)), (midi->pch 69.123 10), (midi->pch ‘(69 71.2 44) 50), etc.

(midi->semitones midi-list)
midi-list: list of at least 2 elements
Result**:** always a list of N-1 elements
NB: positive values indicate an ascending interval, negative values a descending one. Unison is 0.
EX: (midi->semitones ‘(69 70 73 45)), etc.

RATIO
(ratio->fq ratio ref-fq)
ratio: a single ratio or a list of contiguous ratios
ref-fq: starting frequency. The new frequencies will be computed by multiplying the current frequency with the current ratio.
Result**:** always a list of N+1 elements
EX: (ratio->fq ‘(0.5 1.5 2.0) 440) -> (440 220 330 660)
(ratio->fq () 440) -> (440.0)
(ratio->fq 0.5 440) -> (440.0 220.0), etc.

(ratio->itvl ratio)
ratio: a single ratio or a list of ratios
EX: (ratio->itvl '0.5) -> (1 -1)
(ratio->itvl '(0.5)) -> ((1 -1))
(ratio->itvl '(3.4 4.5 0.99)) -> ((6+ 1 19) (2+ 2 4) (1 0 -17)), etc.

(ratio->semitones ratio)
ratio: a single ratio or a list of ratios
EX: (ratio->semitones 0.5) -> -12
(ratio->semitones '(0.5 1.6 2.0) -> (-12.0 8.136862 12.0), etc.

INTERVALS
(itvl->fq itvl ref)
itvl: single interval symbolic interval or list of contiguous intervals.
ref: starting frequency.
Result**:** always a list of N+1 elements
Bug: a single interval must be a unique symbol (ex. '6+); if it is microtonal (ex. (6+ 0 -12)) it has to be used in the form in a list, otherwise the system will treat the argument as a list of symbolic intervals.
EX: (itvl->fq '6+ 440) -> (440.0 739.988)
(itvl->fq '(6+ 3- (3- 0 -12)) -> (440.0 739.988 880.0 1039.2735), etc.

(itvl->midi itvl ref)
itvl: single interval symbolic interval or list of contiguous intervals.
ref: starting midi note
Result**:** always a list of N+1 elements
Bug: a single interval must be a unique symbol (ex. '6+); if it is microtonal (ex. (6+ 0 -12)) it has to be used in the form in a list, otherwise the system will treat the argument as a list of symbolic intervals.
EX: (itvl->midi '6+ 60) -> (60 69)
(itvl->midi '(6+) 60) -> (60 69)
(itvl->midi '((6+ 0 12)) 60) -> (60 1728/25)
(itvl->midi '(6+ (3- 0 -50) (1 0 50)) 60) -> (60 69 143/2 72), etc.

(itvl->ratio itvl)
itvl: single interval symbolic interval or list of contiguous intervals.
Result**:** always a list of N elements
Bug: a single interval must be a unique symbol (ex. '6+); if it is microtonal (ex. (6+ 0 -12)) one has to use the form in a list, otherwise the system will treat the argument as a list of symbolic intervals.
EX: (itvl->ratio '6+) -> (1.68179)
(itvl->ratio '(2- (3- 1) (2- 0 -50))) -> (1.059 2.378 1.029), etc.

(itvl->semitones itvl)
itvl: single interval symbolic interval or list of contiguous intervals.
Result**:** always a list of N elements
Bug: a single interval must be a unique symbol (ex. '6+); if it is microtonal (ex. (6+ 0 -12)) one has to use the form in a list, otherwise the system will treat the argument as a list of symbolic intervals.

EX: (itvl->semitones '6+) -> (9)
(itvl->semitones '(2- (3- 1) (2- 0 -50))) -> (9 15 1/2), etc.

SEMITONES
(semitones->ratio val)
val: single number of semitones or list of semitones.
EX: (semitones->ratio 6) -> 1.414
(semitones->ratio '(6)) -> (1.414)
(semitones->ratio '(6 1 0 12) -> (1.414 1.059 1 2), etc.
(semitones->ratio '(6.1 1.5 0.5 -12) -> (1.422 1.090 1.029 1/2), etc.
etc.

(semitones->itvl val)
val: single midi value or a list of values.
EX: (semitones->itvl 6) -> (4+ 0)
(semitones->itvl '(6)) -> ((4+ 0))
(semitones->itvl '(6 1 0 12) -> ((4+ 0) (2- 0) (1 0) (1 1)), etc.
(semitones->itvl '(6.1 1.5 0.5 -12) -> ((4+ 0 10) (2- 0 -50) (1 0 50) (1 1)),

POLYMORPHISM
All the system is conceived polymorphically, so that several types of data can be processed by the same method. The result will be of the same type as the input data. For example, if the function "transpose" is applied to a single pitch, the result will be another single pitch, whereas if the same function is applied to a chord, the result will be another chord, with all the pitches correctly transposed.

Some methods are defined only for a given representation (ex. get-cs works only with spl's, see below). When passing a different representation

VPS: MULTIPLE REPRESENTATIONS
A VPS is a Vertical Pitch Structure (see below) and is implenented as a set of CLOS classes. Class structure:

VS	CHORD	SPL
		RPL
		CIL
		AIL
		PPL
	SPECTRUM	FQL
		CRL
		ARL

Schematic Representation
We will first describe the procedures to instanciate and access the objects using CLOS. Further down we will present a "user-friendlier" interface providing some macros that mask the CLOS syntax and do not require keywords.

VS: Vertical Structure, the root of everything. The only test performed at this stage is on the existance of the input arguments + some very generic functions such as print.

VPS: Vertical Pitch Structure, the son of the root. For "vertical" we mean that it consists of a list of items that do not repeat themselves (no unisons, for example) and are in an ascending order, either explicit or implicit. Other information, such as the current diapason is also initialized here.

CHORD: A list of Pitches of symbolic Intervals (formerly called a VPS).

- **SPL:** Symbolic Pitch List, a list of symbolic pitches. Ex. '(RE3 LA3 FAd4)
- **RPL:** Relative Pitch List, a list of pitches whose octave number is relative to the first pitch in the list. Ex. '(RE LA FAd1). Here the F sharp is 1 octave above the low D.
- **CIL:** Contiguous Interval List, a list of contiguous intervals. Ex. '(5 6+). This object contains N-1 elements.
- **AIL:** Anchored Interval List, a list of intervals with respect to a reference pitch (anchor). Ex. '(-4+ 2- 7-) with anchor = SOLd3.
- **PPL:** Pure Pitch List, internal represention not to be used.

SPECTRUM: A list of Frequencies of Ratios.

- **FQL:** Frequency List, a list of frequencies [Hz]. When this class is used to represent analysis data (from AudioSculpt, for example) two parallel lists of Amplitudes and Band Widths are associated to the list of frequencies. They must have the same number of elements. Ex. '(100 150 300 450).
- **CRL:** Contiguous Ratios List, a list of contiguous ratios. Ex. '(1.5 2 1.6). This object contains N-1 elements. Since the elements must be in ascending order, the ratios are always > 1.0
- **ARL:** Anchored Ratios List, a list of ratios with respect to a reference frequency (anchor). When the anchor is low, this corresponds to a spectrum (anchor = f0). Ex. '(0.5 0.75 1.5 2.22) with an anchor of 200 or '(10 15 30 45) with an anchor (f0) of 10.

Schematic examples

```
;SPL (Symbolic Pitch List)
;    '(DO4 LAb4 RE5 SOL5 REb6)
;    '(DO4 (LAb4 23) RE5 (SOL5 -12) REb6)

;RPL (Relative Pitch List)
;    '(DO LAb RE1 SOL1 REb2)
;    '((DO . 23) (LAb . -34) RE1 SOL1 (REb2 . 77))

;PPL (Pure Pitch List)
;    '(DO LAb RE SOL REb)
;    '((DO . 23) (LAb . -34) RE SOL (REb . 77))

;CIL (Contiguous Interval List)
;    '(6- 4+ 4 4+)
;    '((6- 0 12) (4+ 1 -21) (4 0 7) (4+ 1 -2))

;AIL (Anchored Interval List)
```

; '(-6+ -2- 4 7- (3+ 1)) 'LA4
; '((-6+ 0 12) -2- (4 0 -22) 7- (3+ 1 17)) + reference: 'LA4

;**FQL** (Frequency List)
; '(261 415 587 784 1108)

;**CRL** (Contiguous Ratios)
; '(1.5873 1.4142 1.3345 1.4145)

;**ARL** (Spectrum / Anchored Ratios)
; '(2.61 4.153 5.873 7.84 11.07) + reference: 100.0

For all the possible methods and user-friendlier routines, see the user's manual of the VPS's in MCL.