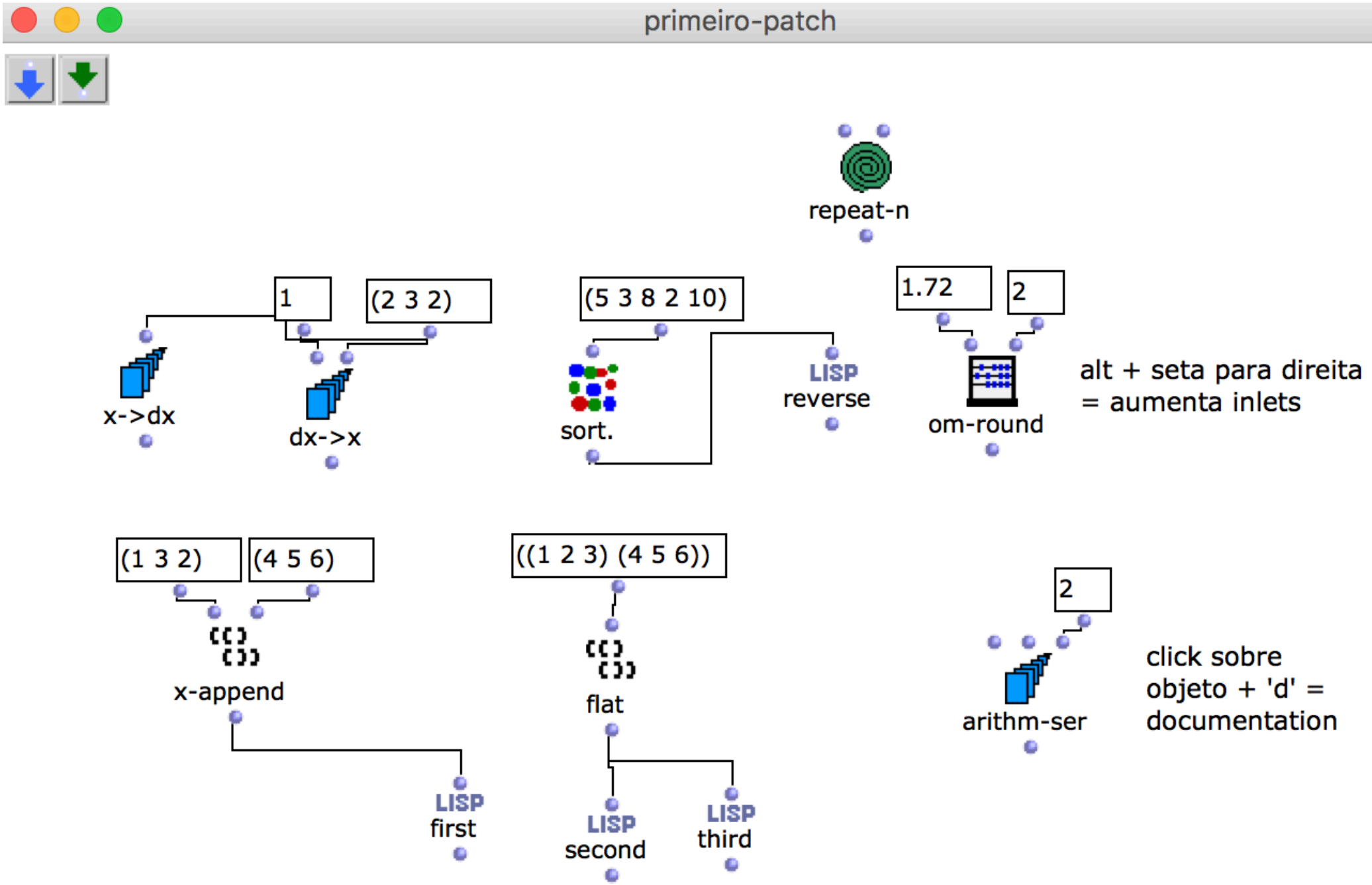
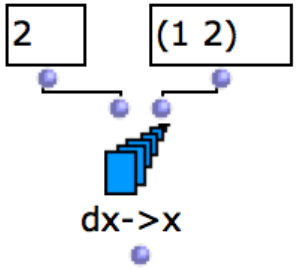
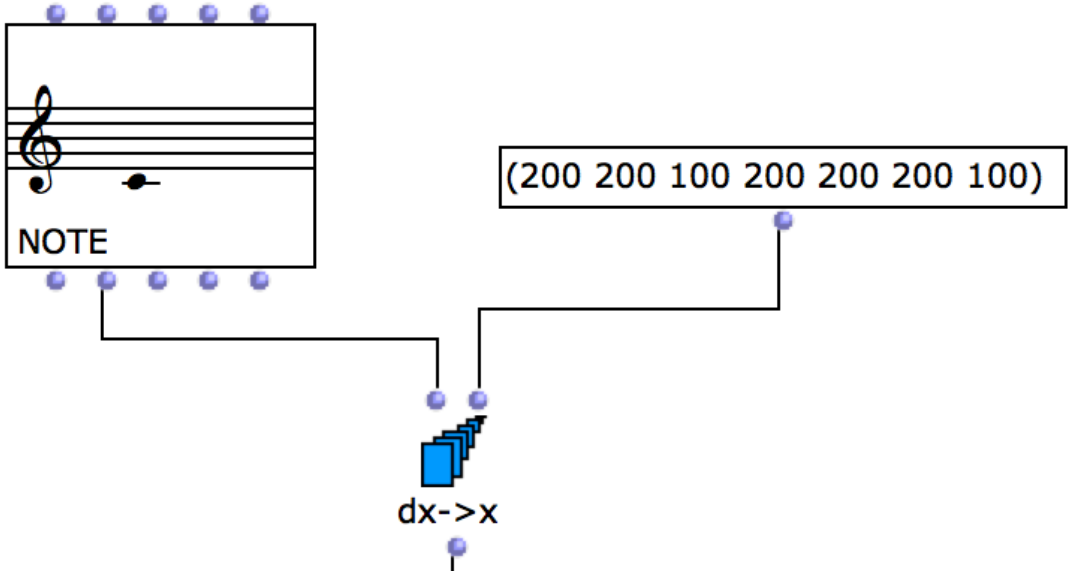

primeiro-patch
repeat-n
1
(2 3 2)
x->dx
dx->x
(5 3 8 2 10)
sort.
LISP
reverse
1.72
2
om-round
alt + seta para direita
= aumenta inlets
(1 3 2)
(4 5 6)
x-append
LISP
first
((1 2 3) (4 5 6))
flat
LISP
second
LISP
third
2
arithm-ser
click sobre
objeto + 'd' =
documentation

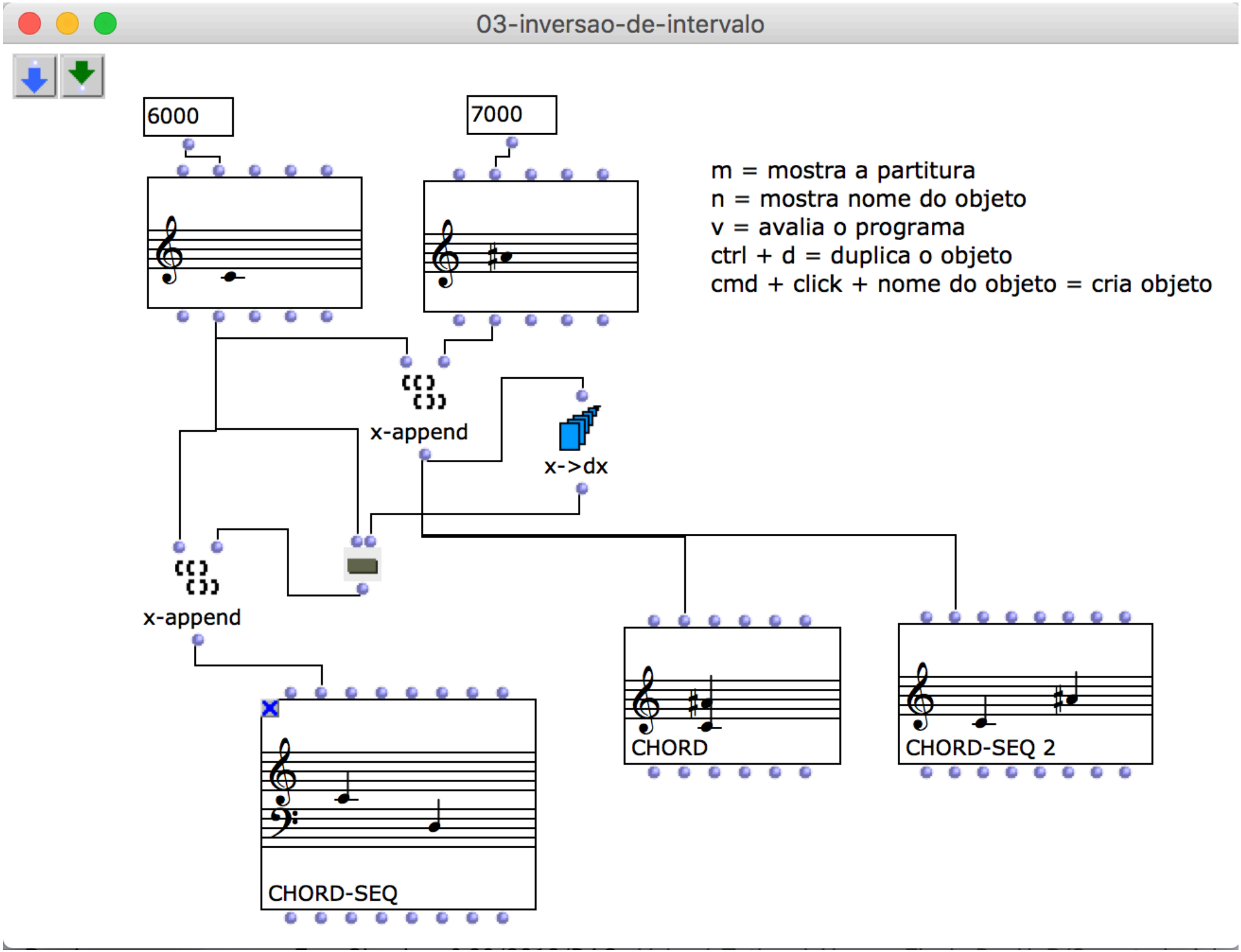
^02-construindo-escalas

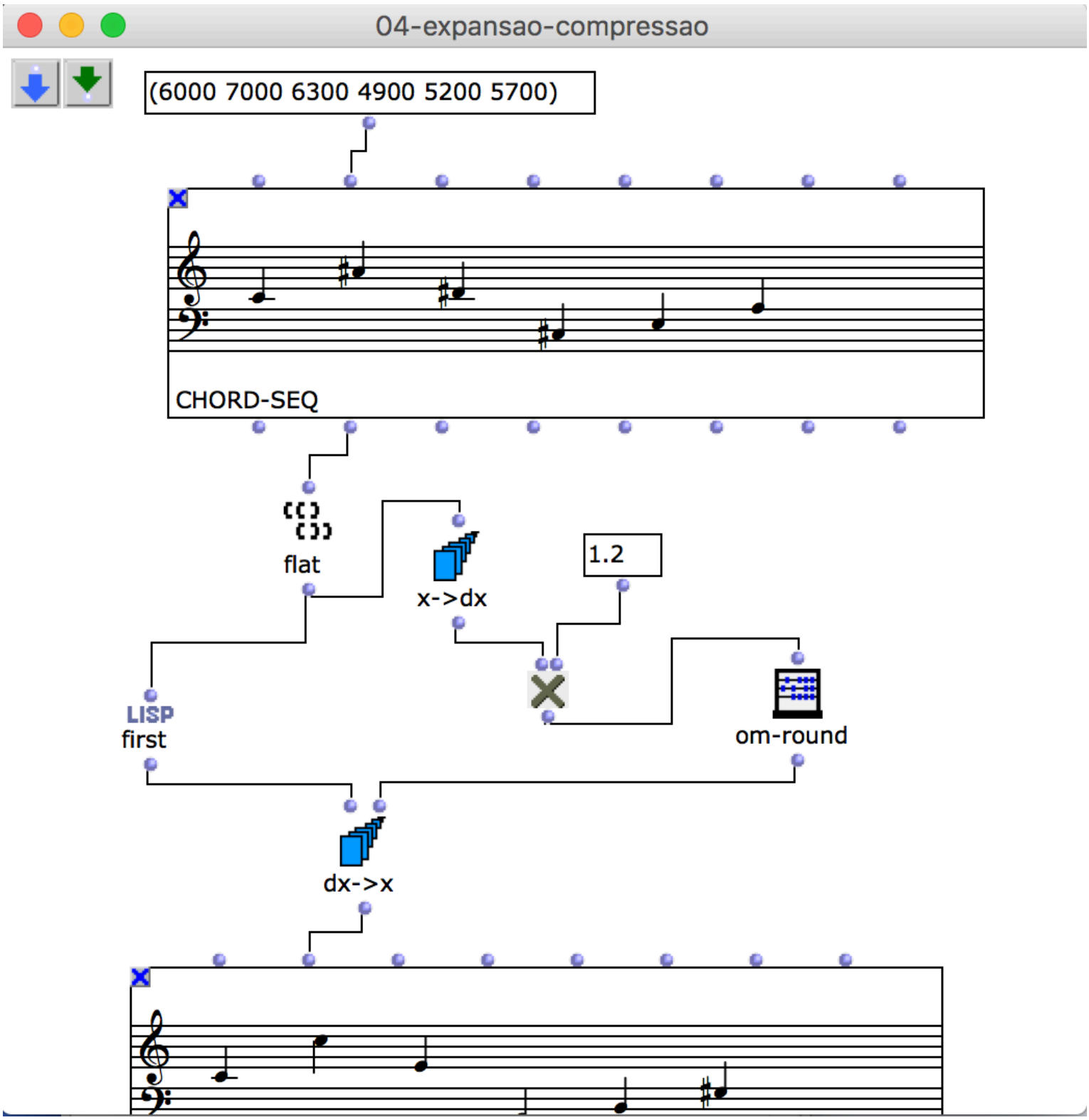
M = mostra partitura
N = mostra nome do objeto

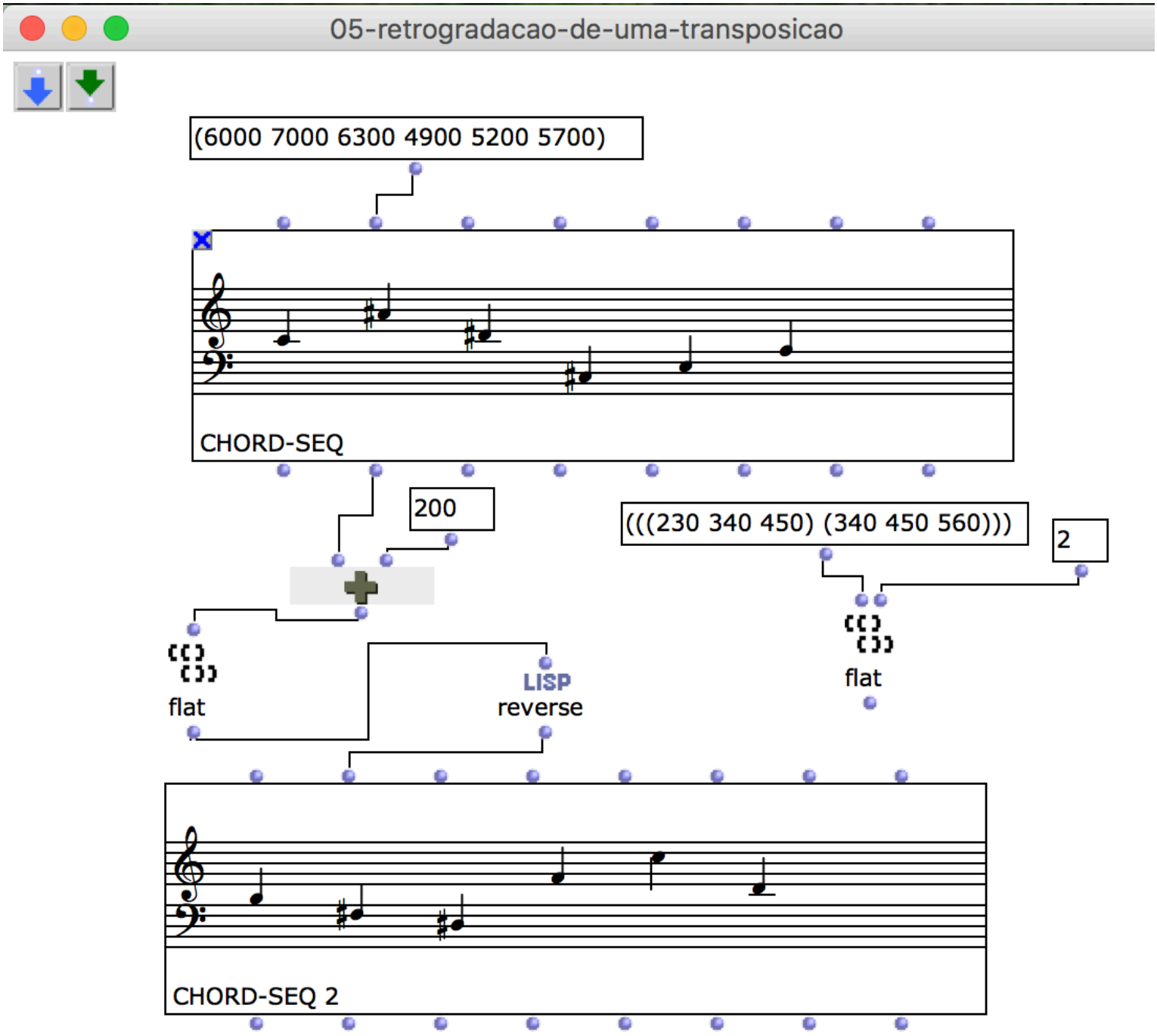
6000+200 = 6200
6200+200 = 6400
6400+100 = 6500

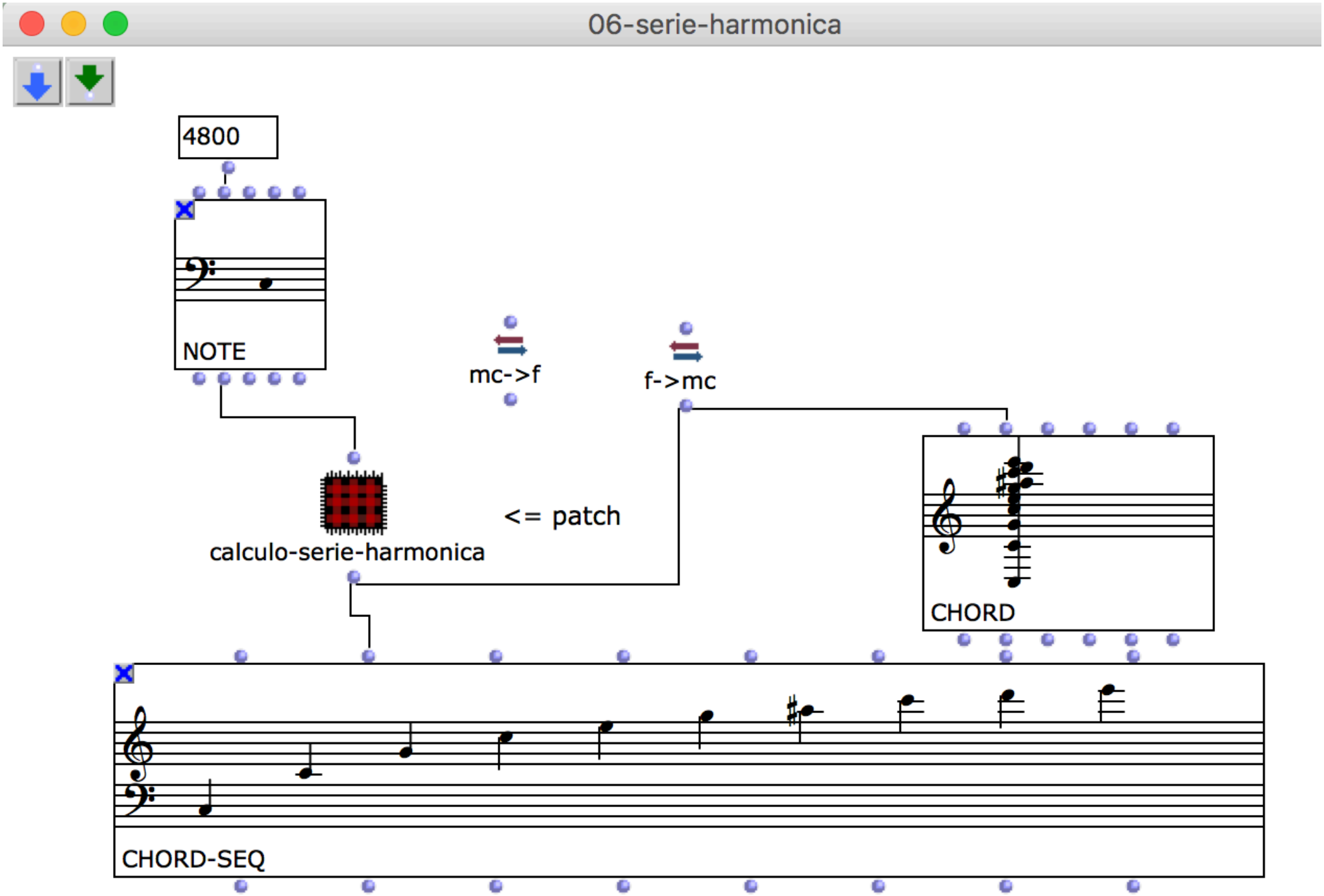


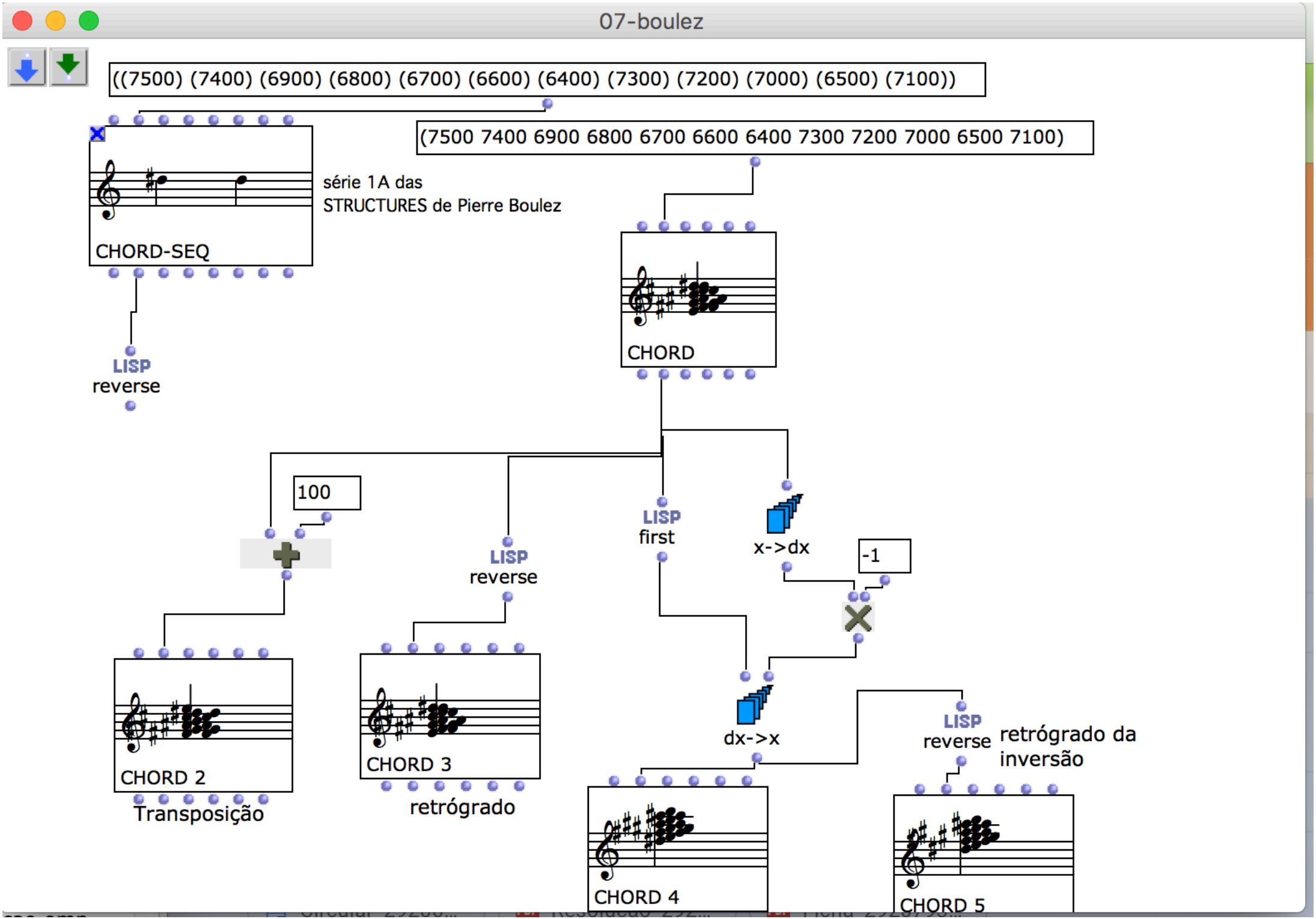



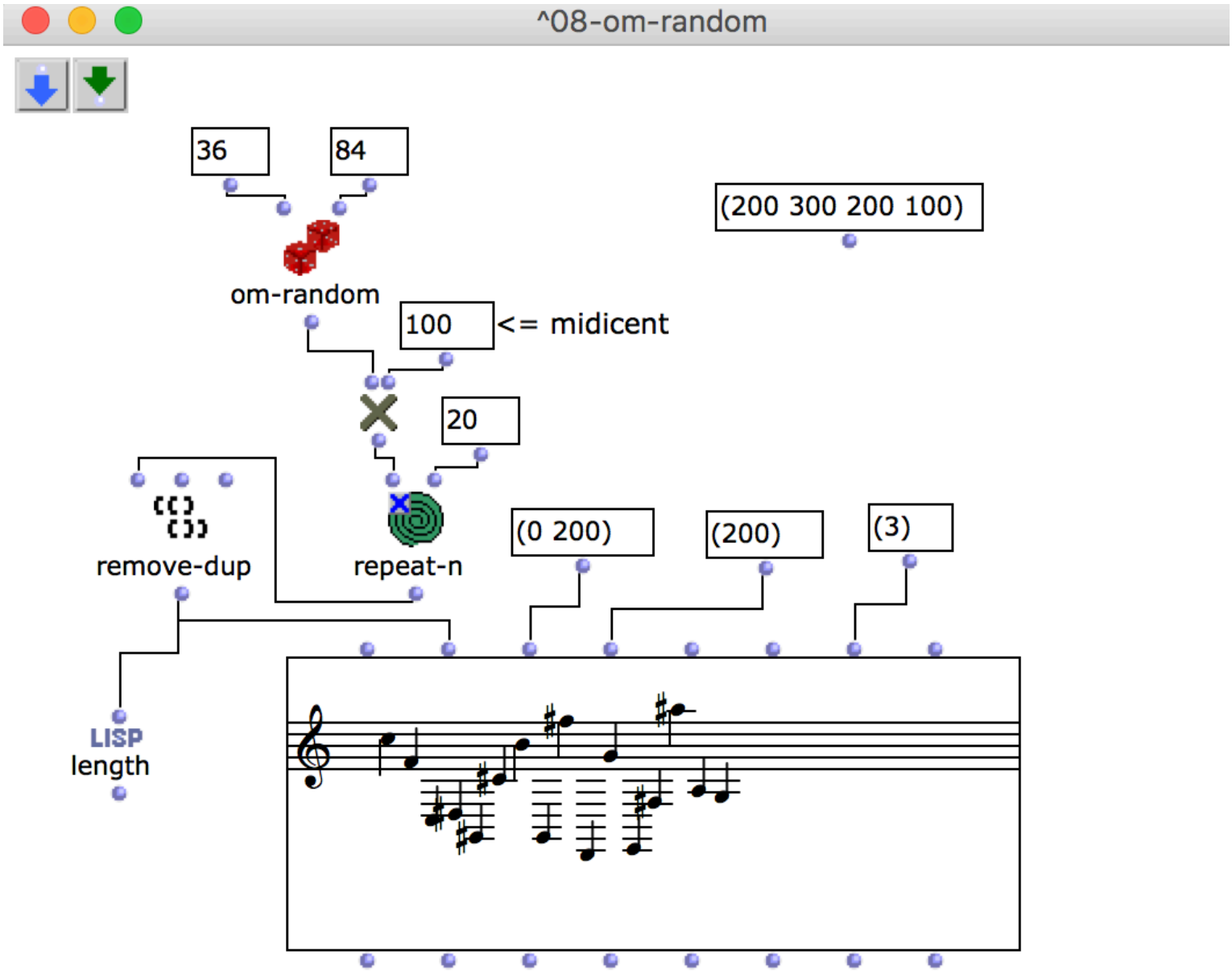

03-inversao-de-intervalo
6000
7000
m = mostra a partitura
n = mostra nome do objeto
v = avalia o programa
ctrl + d = duplica o objeto
cmd + click + nome do objeto = cria objeto
x-append
x->dx
x-append
CHORD-SEQ
CHORD
CHORD-SEQ 2

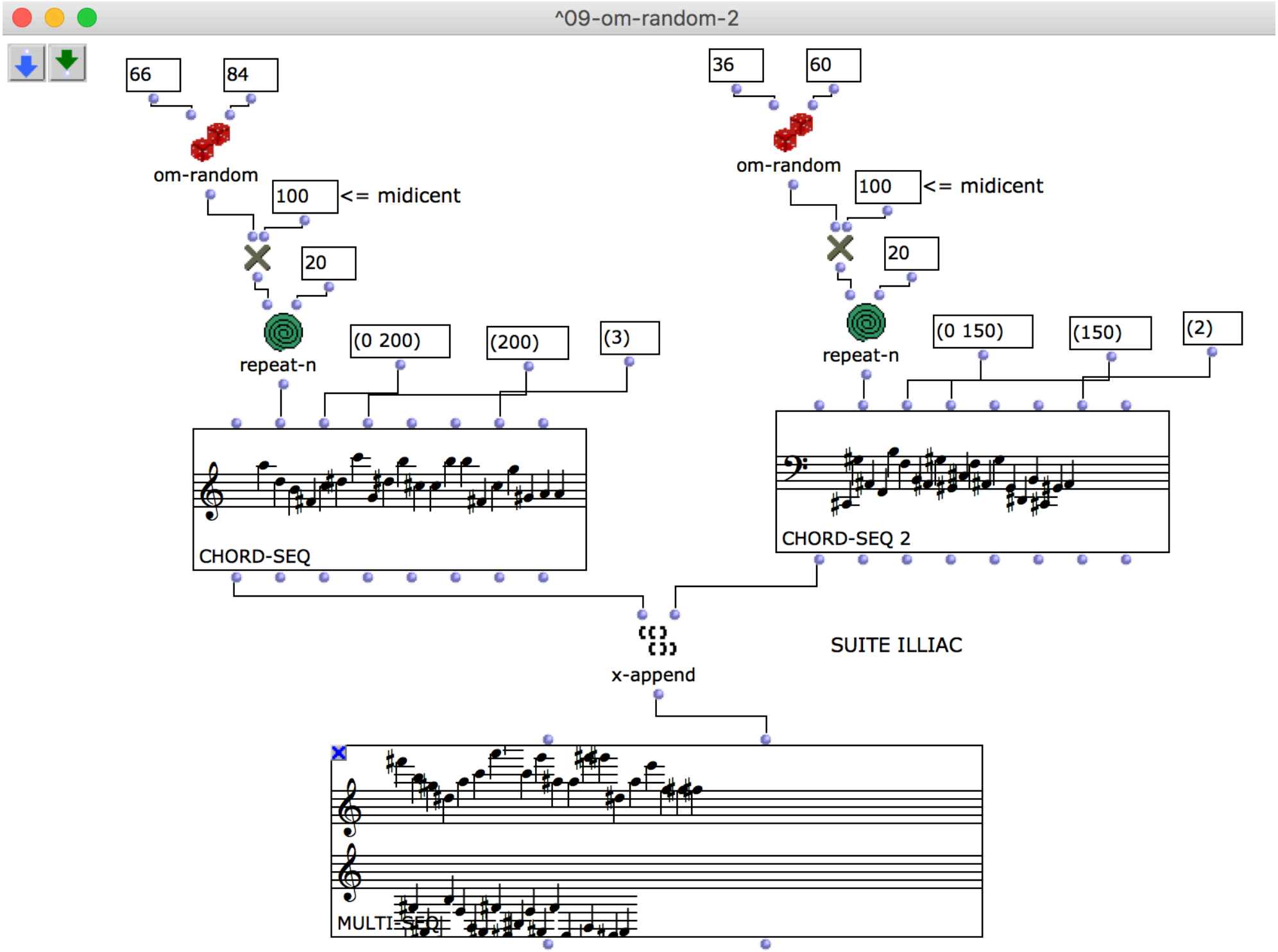

04-expansao-compressao
(6000 7000 6300 4900 5200 5700)
CHORD-SEQ
flat
x->dx
1.2
LISP
first
om-round
dx->x

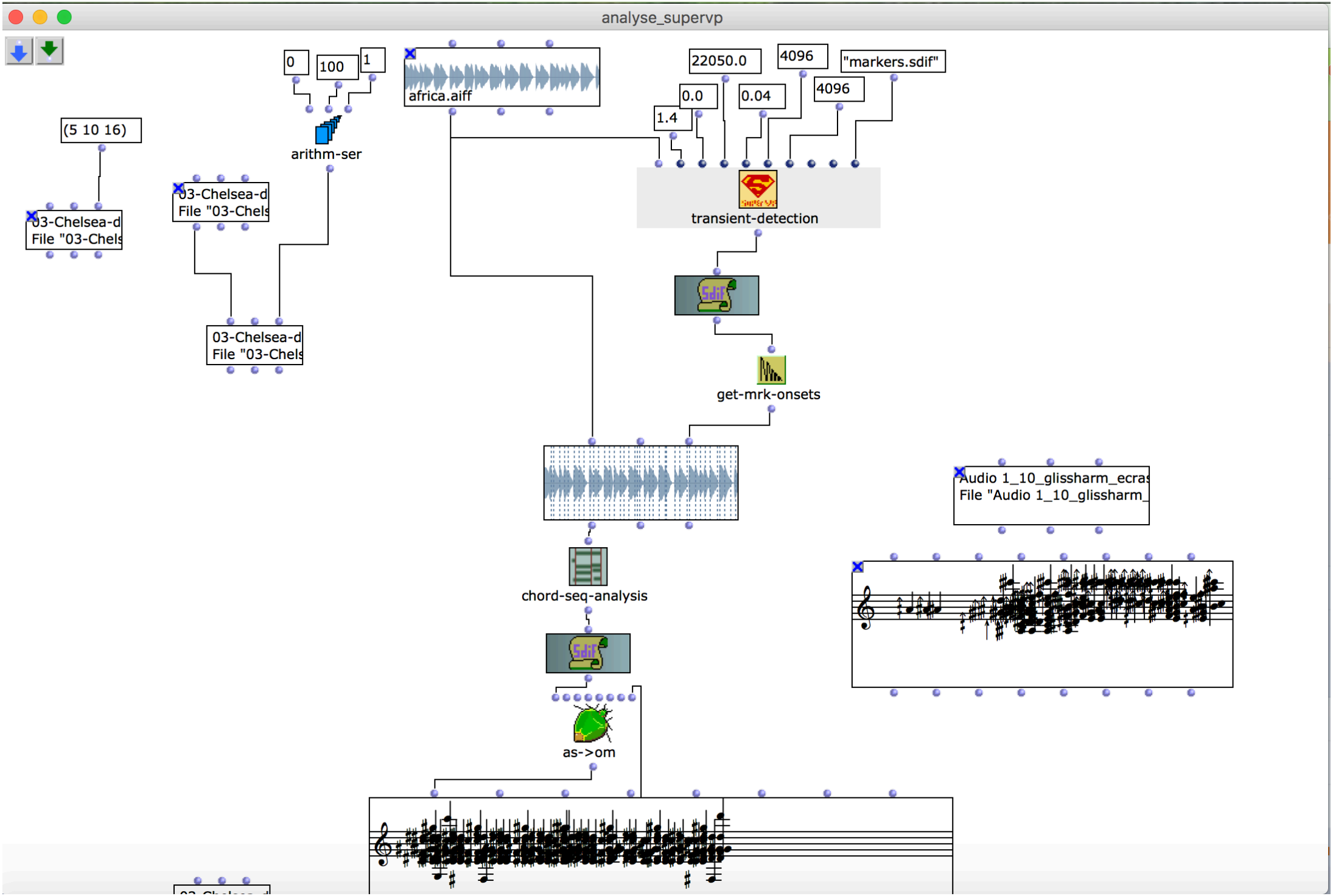

05-retrogradacao-de-uma-transposicao
(6000 7000 6300 4900 5200 5700)
CHORD-SEQ
200
(((230 340 450) (340 450 560)))
2
flat
LISP
reverse
flat
CHORD-SEQ 2

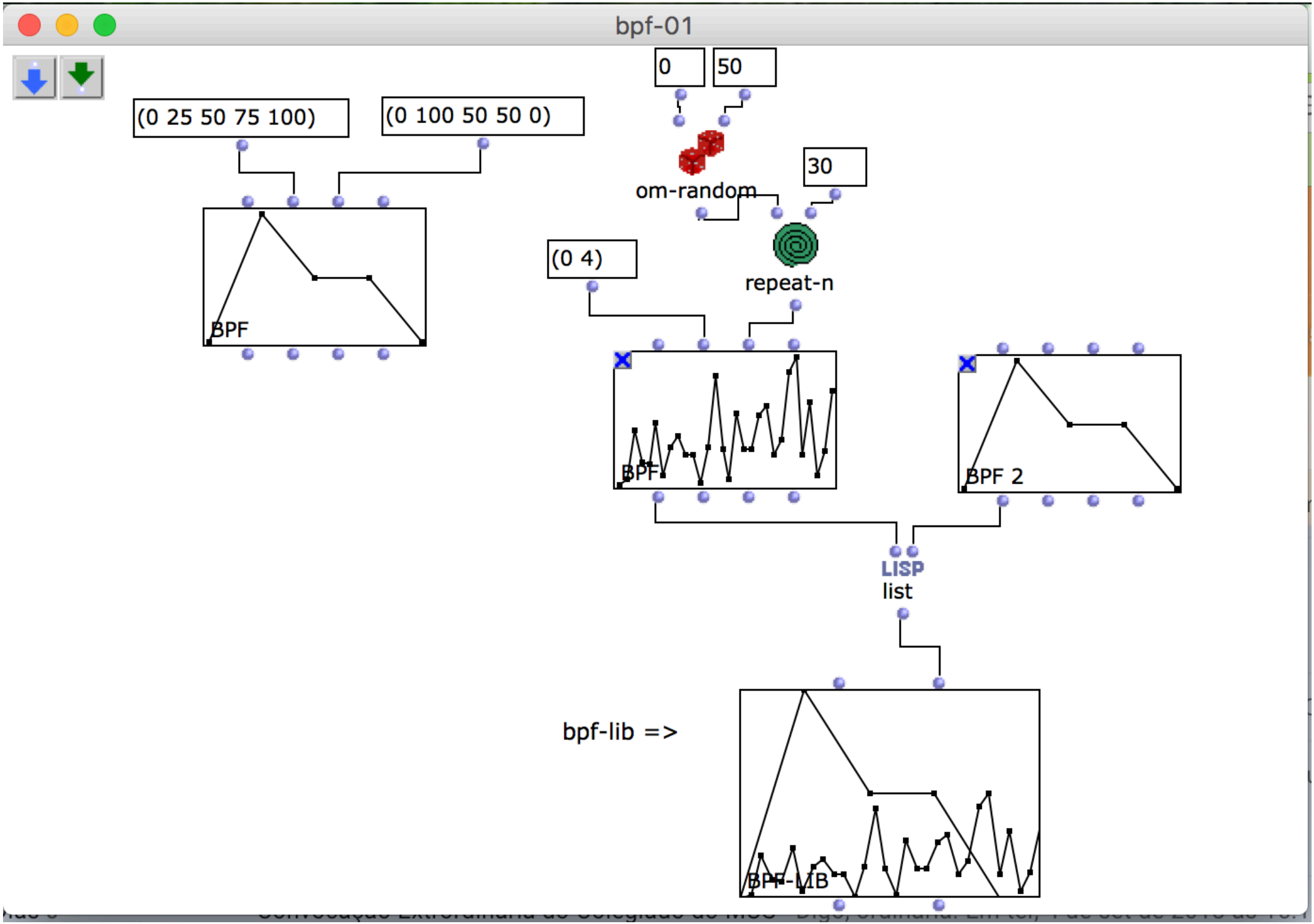

06-serie-harmonica
4800
NOTE
mc->f
f->mc
<= patch
calculo-serie-harmonica
CHORD
CHORD-SEQ


07-boulez
((7500) (7400) (6900) (6800) (6700) (6600) (6400) (7300) (7200) (7000) (6500) (7100))
(7500 7400 6900 6800 6700 6600 6400 7300 7200 7000 6500 7100)
CHORD-SEQ
série 1A das
STRUCTURES de Pierre Boulez
LISP
reverse
CHORD
100
LISP
reverse
LISP
first
x->dx
-1
CHORD 2
Transposição
CHORD 3
retrógrado
dx->x
LISP
reverse
retrógrado da
inversão
CHORD 4
CHORD 5


^08-om-random
36
84
om-random
(200 300 200 100)
100
<= midicent
20
remove-dup
repeat-n
(0 200)
(200)
(3)
LISP
length

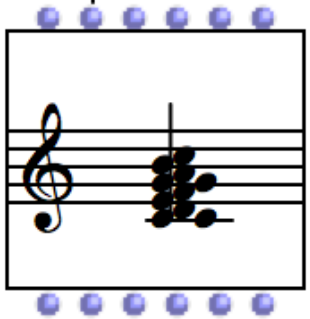

^09-om-random-2
66
84
om-random
100
<= midicent
20
repeat-n
(0 200)
(200)
(3)
CHORD-SEQ
36
60
om-random
100
<= midicent
20
repeat-n
(0 150)
(150)
(2)
CHORD-SEQ 2
x-append
SUITE ILLIAC
MULTI-SEQ

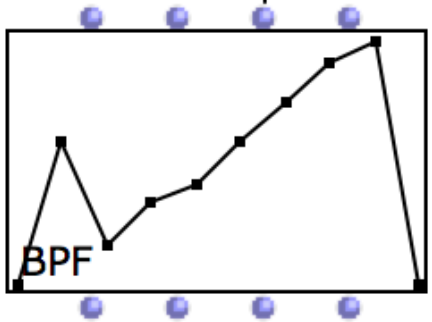

analyse_supervp
0
100
1
africa.aiff
22050.0
4096
"markers.sdif"
0.0
0.04
4096
1.4
(5 10 16)
arithm-ser
03-Chelsea-d
File "03-Chels
transient-detection
get-mrk-onsets
chord-seq-analysis
Audio 1_10_glissharm_ecras
File "Audio 1_10_glissharm_
as->om

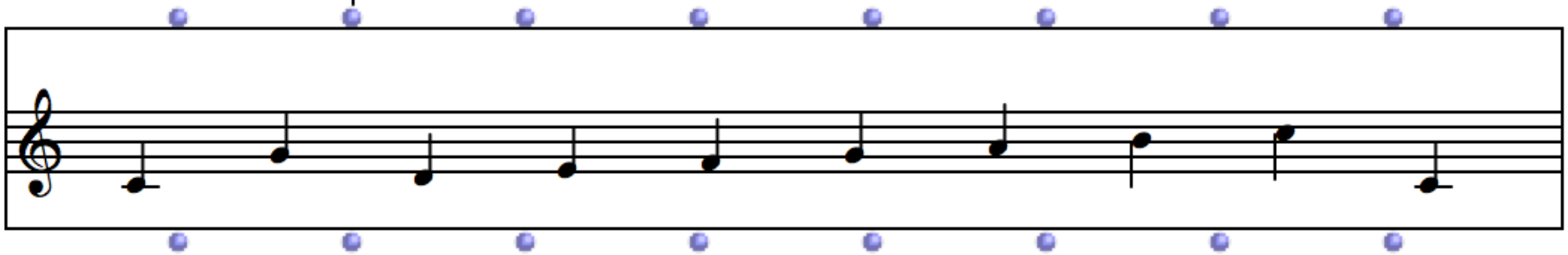

bpf-01
(0 25 50 75 100)
(0 100 50 50 0)
BPF
0
50
om-random
30
repeat-n
(0 4)
BPF
BPF 2
LISP
list
bpf-lib =>
BPF-LIB

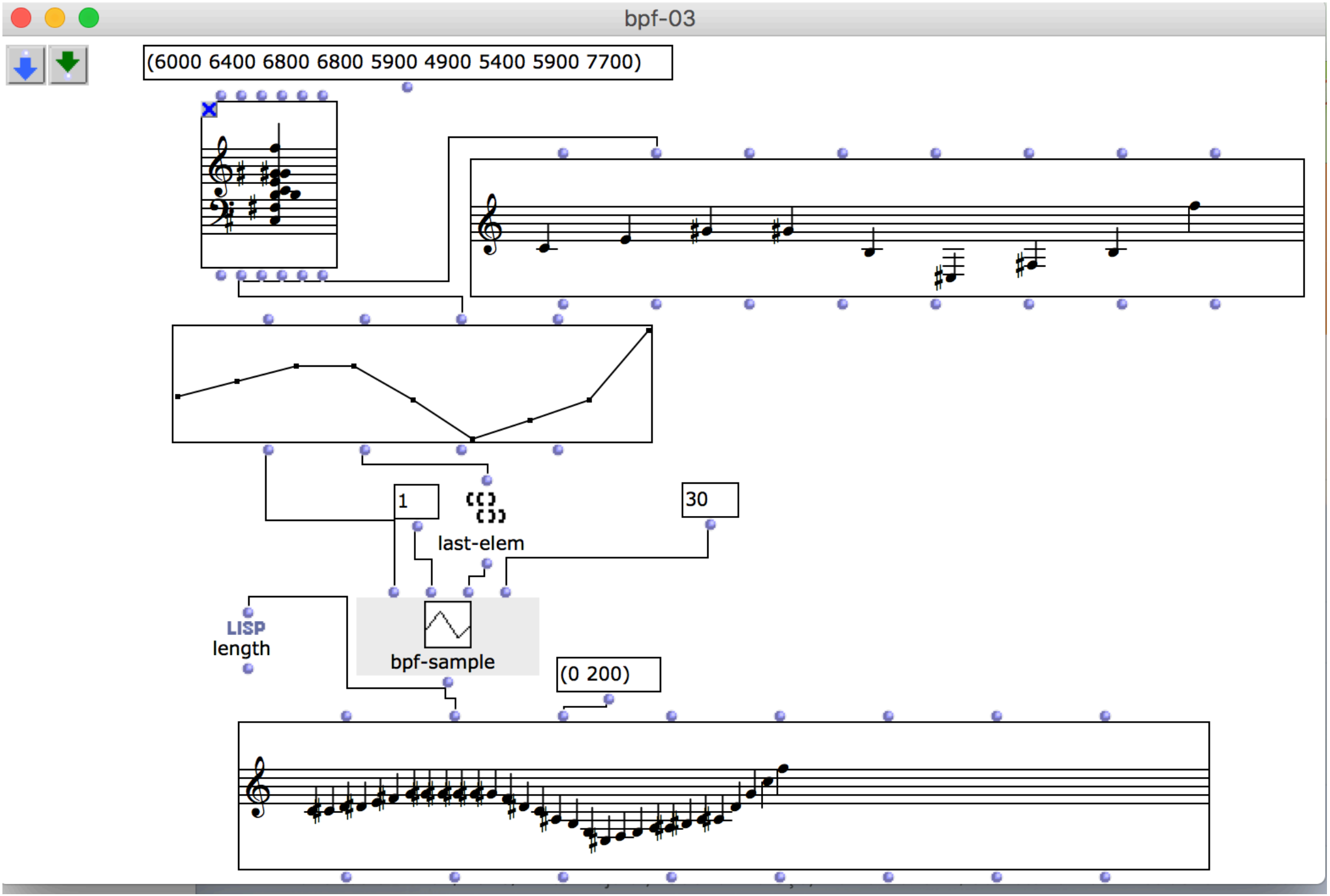
^bpf-02

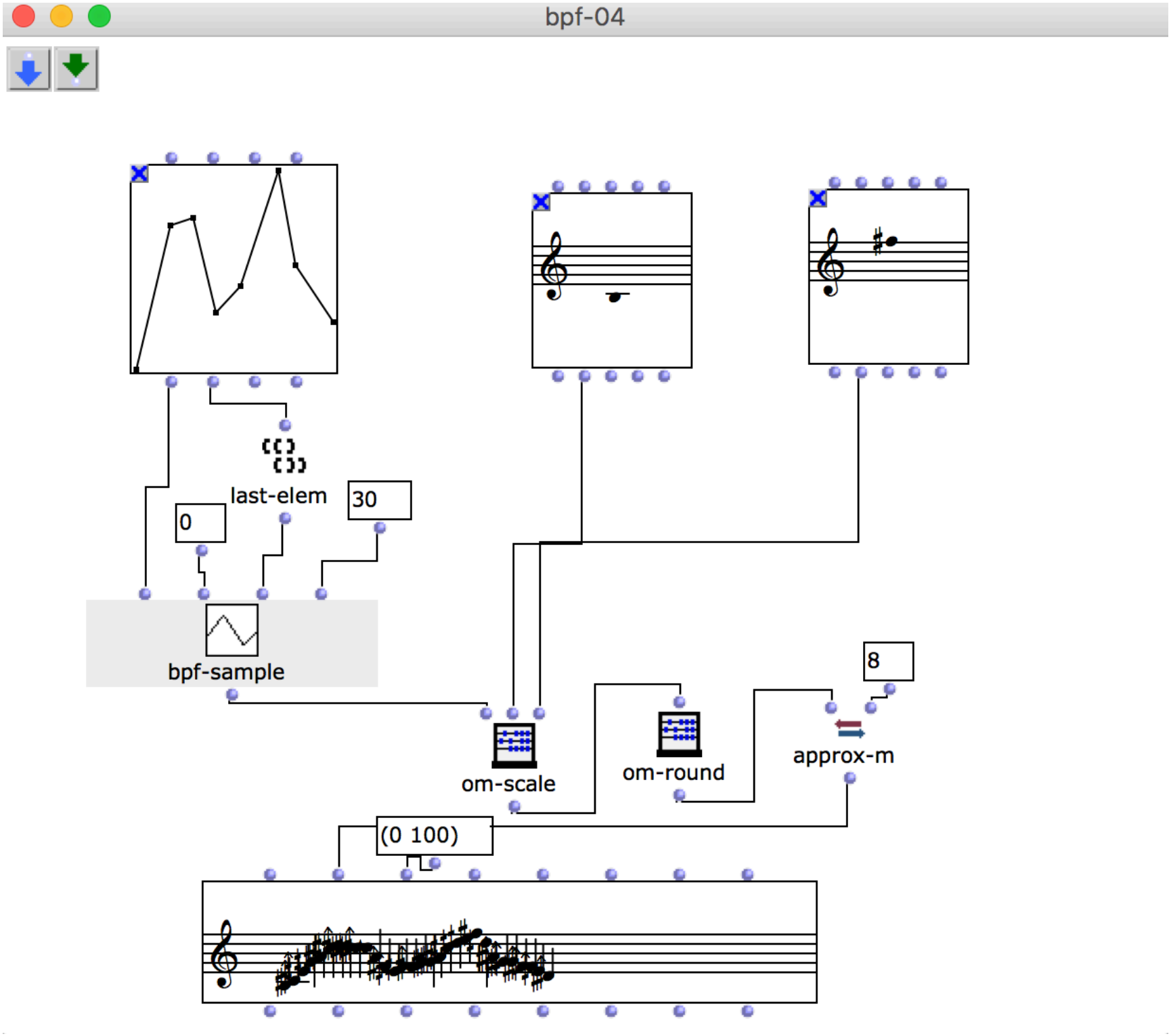
(6000 6700 6200 6400 6500 6700 6900 7100 7200 6000)

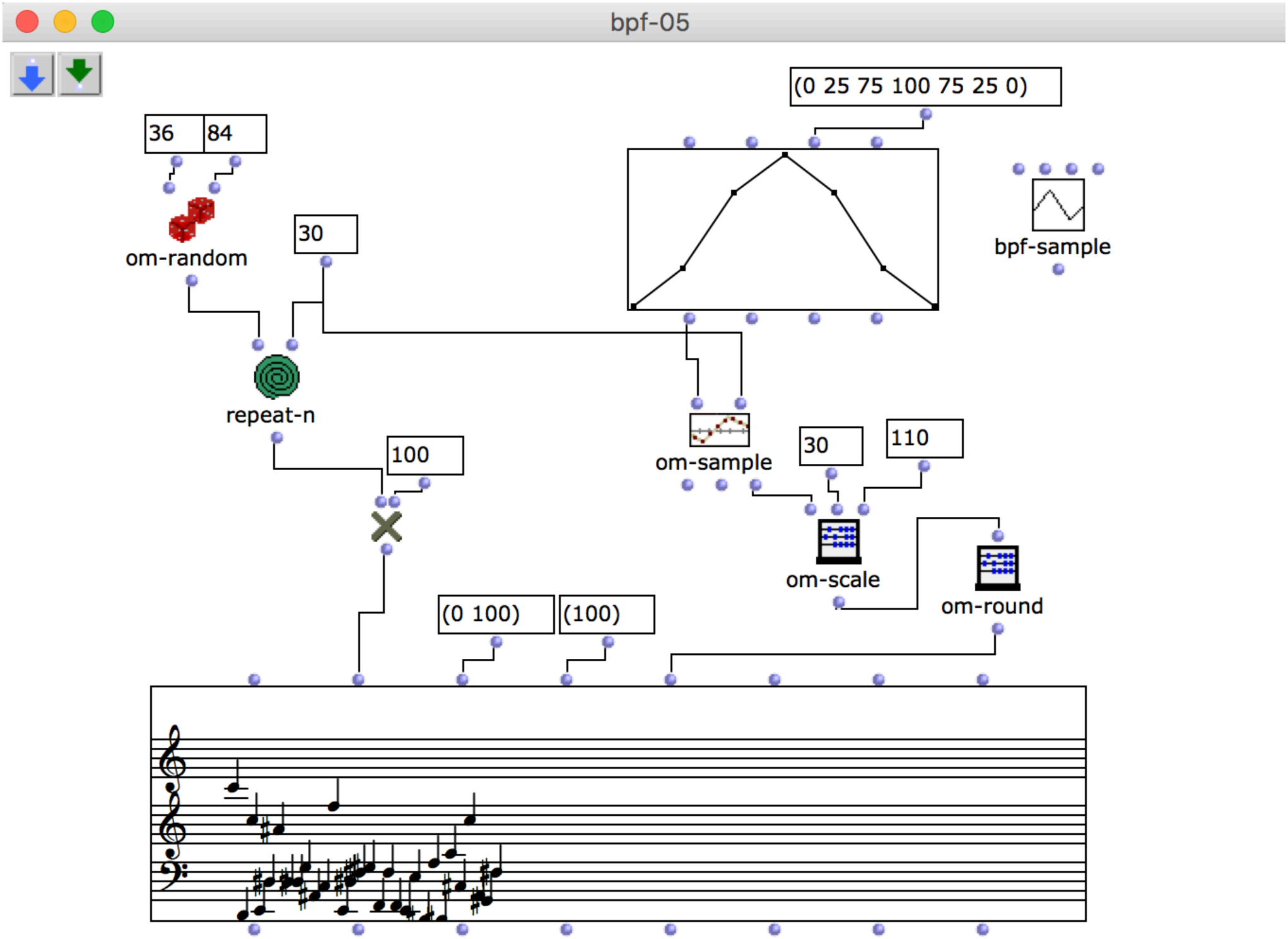


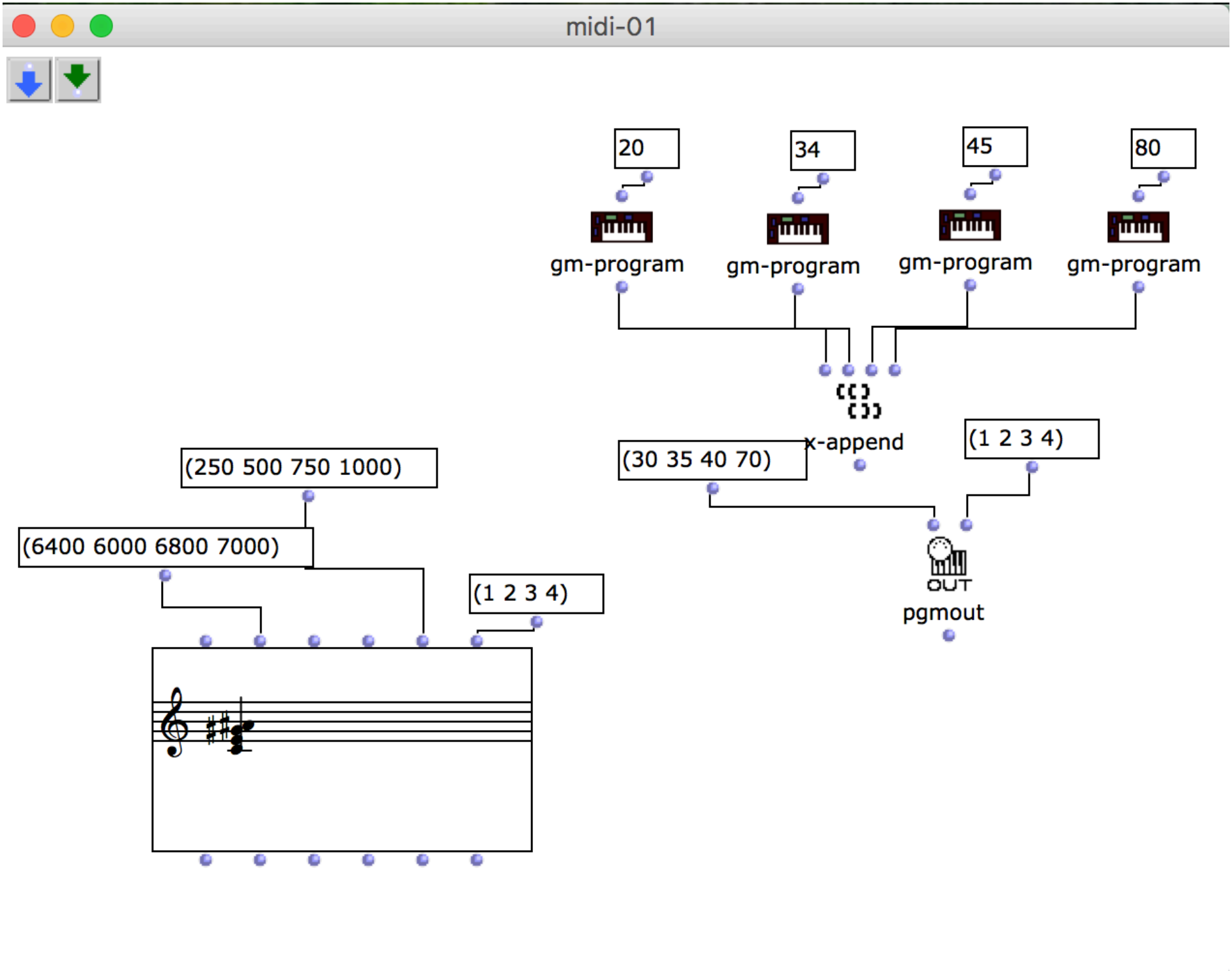


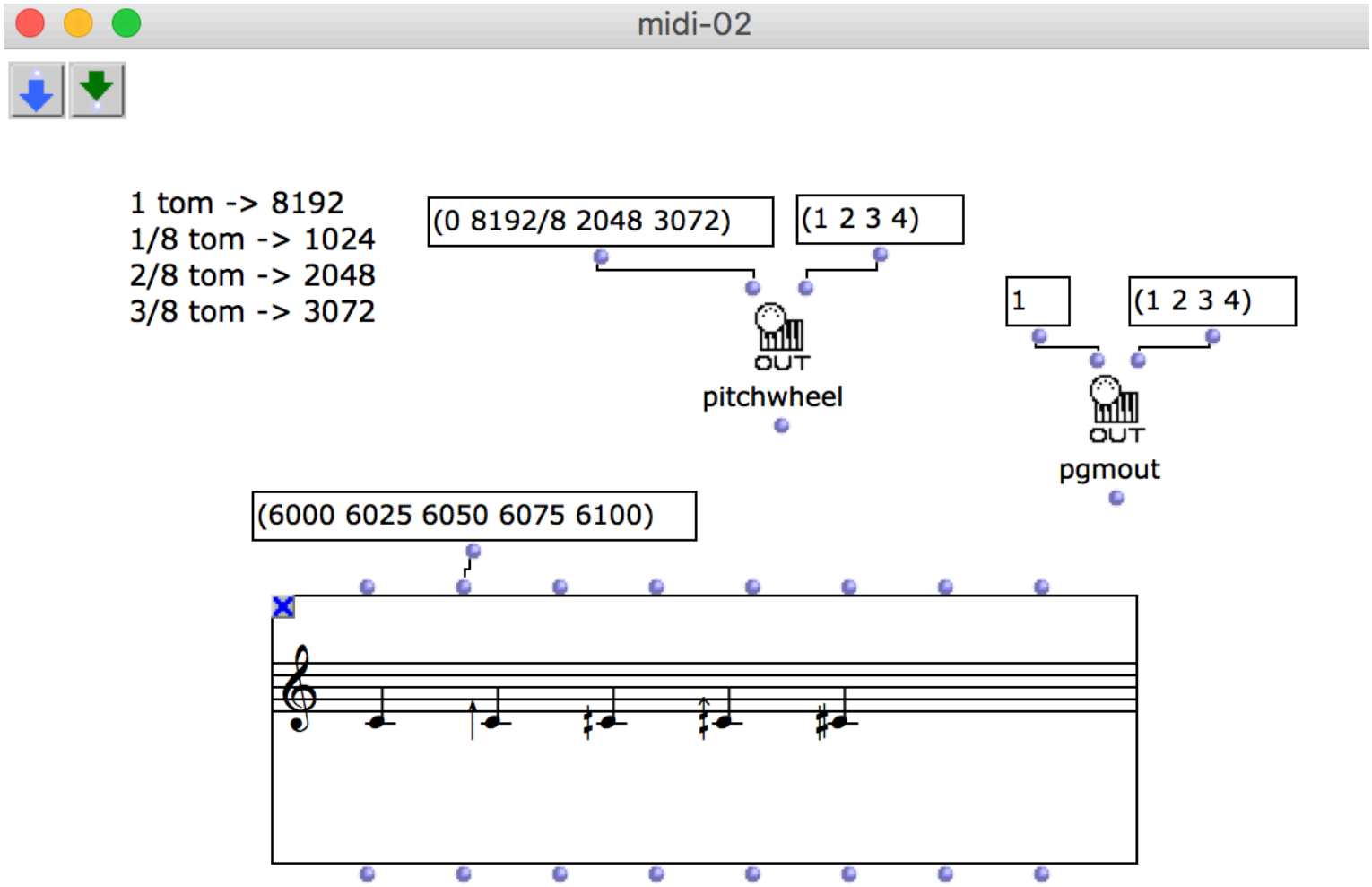



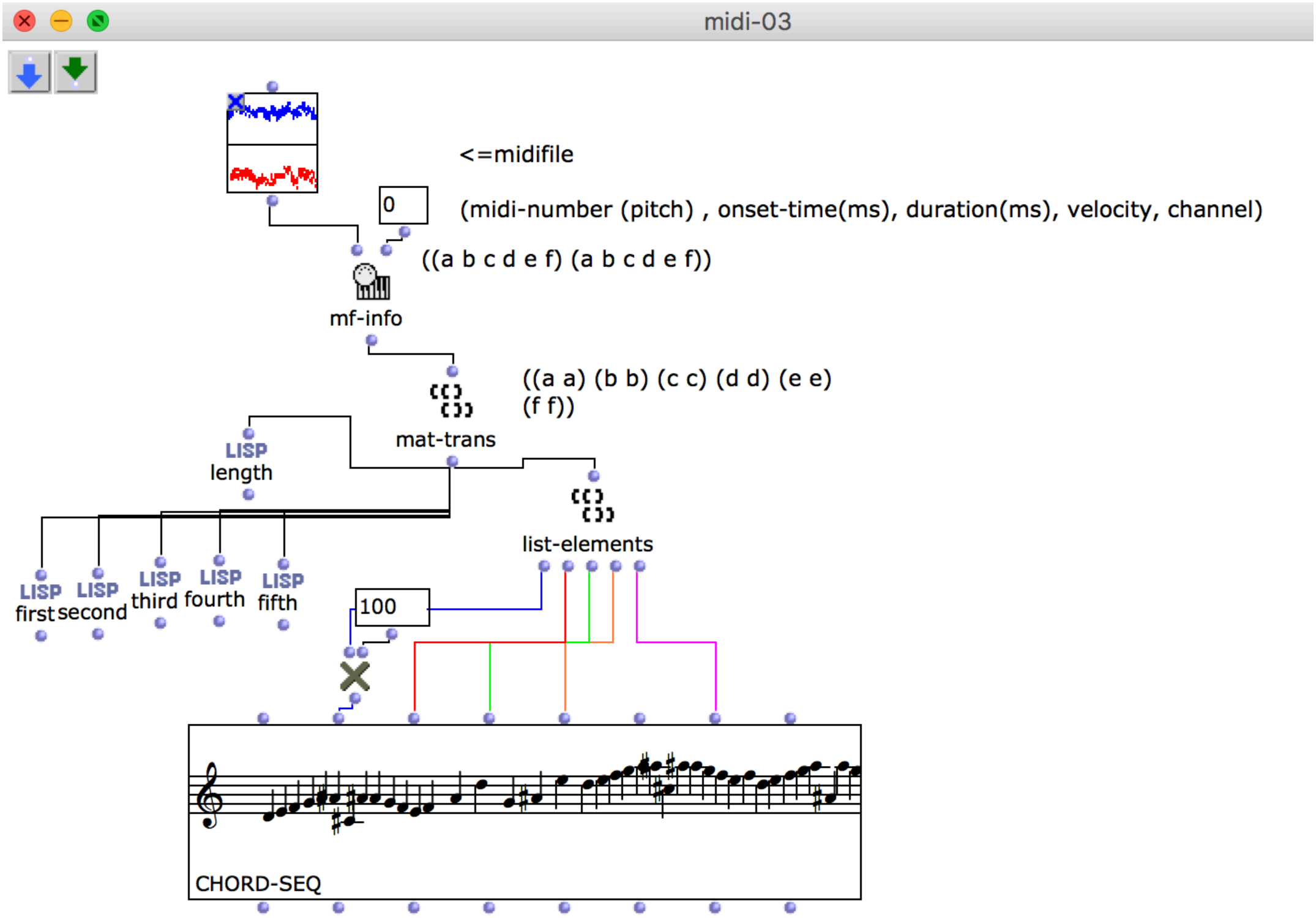

bpf-03
(6000 6400 6800 6800 5900 4900 5400 5900 7700)
1
30
last-elem
LISP
length
bpf-sample
(0 200)

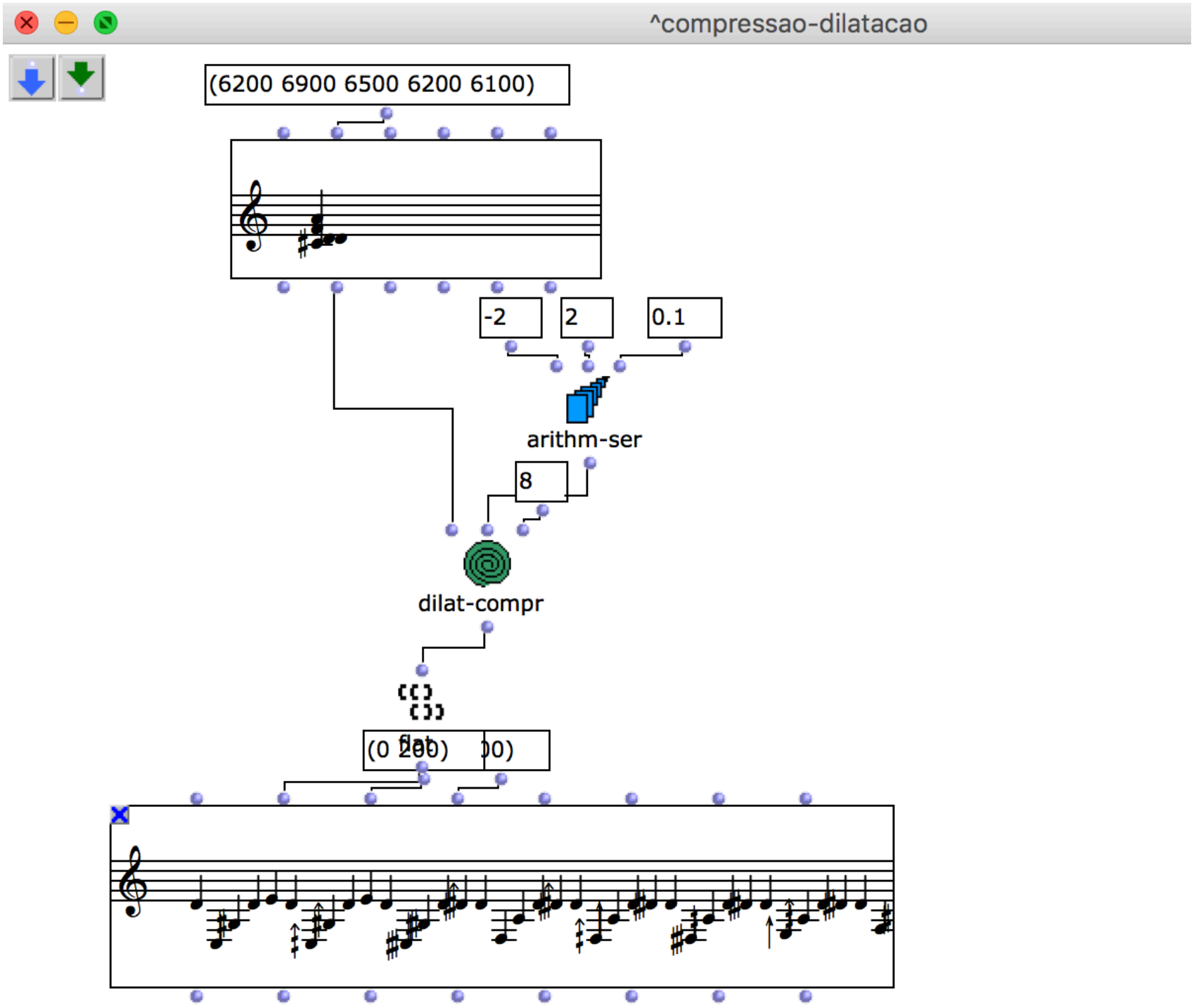

bpf-04
last-elem
0
30
bpf-sample
om-scale
om-round
8
approx-m
(0 100)

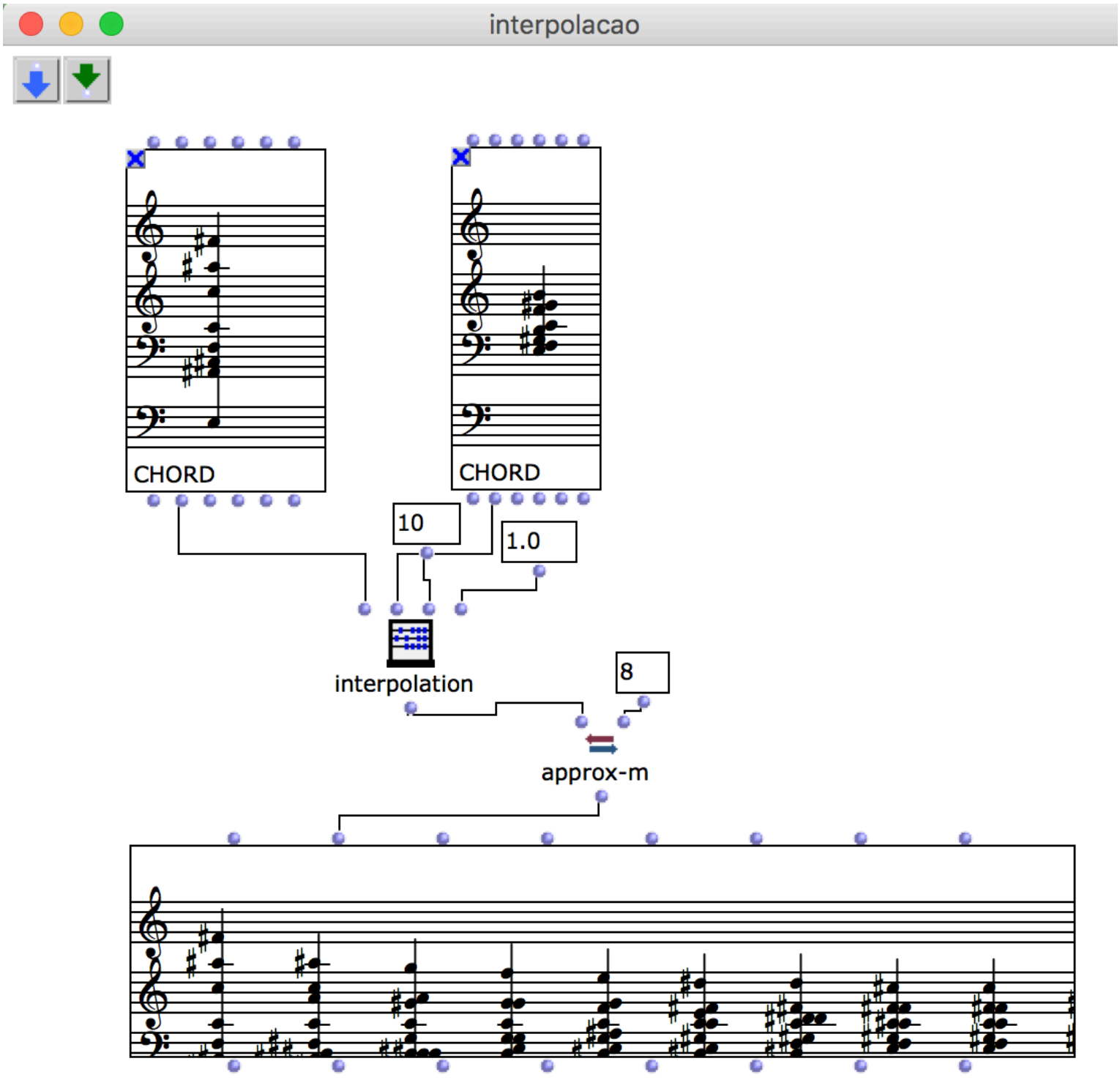

bpf-05
(0 25 75 100 75 25 0)
36
84
om-random
30
bpf-sample
repeat-n
om-sample
100
30
110
om-scale
om-round
(0 100)
(100)

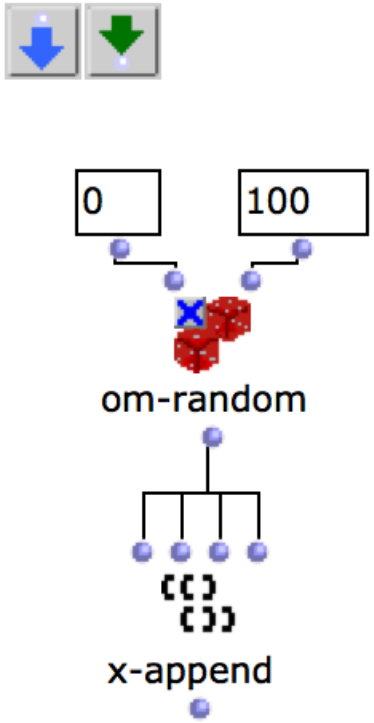

midi-01
20
34
45
80
gm-program
gm-program
gm-program
gm-program
x-append
(30 35 40 70)
(1 2 3 4)
OUT
pgmout
(250 500 750 1000)
(6400 6000 6800 7000)
(1 2 3 4)


midi-02
1 tom -> 8192
1/8 tom -> 1024
2/8 tom -> 2048
3/8 tom -> 3072
(0 8192/8 2048 3072)
(1 2 3 4)
OUT
pitchwheel
1
(1 2 3 4)
OUT
pgmout
(6000 6025 6050 6075 6100)

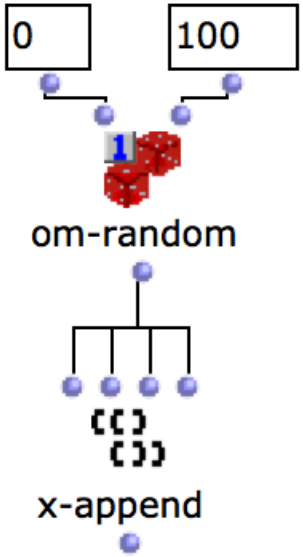

midi-03
<=midifile
0
(midi-number (pitch) , onset-time(ms), duration(ms), velocity, channel)
((a b c d e f) (a b c d e f))
mf-info
((a a) (b b) (c c) (d d) (e e)
(f f))
mat-trans
LISP
length
LISP
first
LISP
second
LISP
third
LISP
fourth
LISP
fifth
list-elements
100
CHORD-SEQ


^compressao-dilatacao
(6200 6900 6500 6200 6100)
-2
2
0.1
arithm-ser
8
dilat-compr
(0 200)

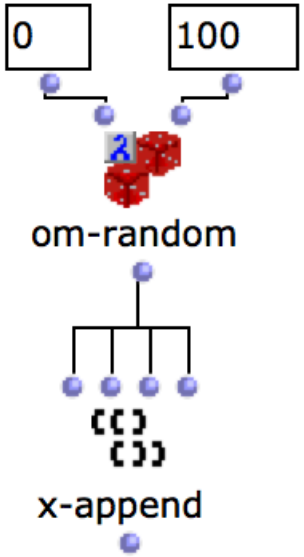

interpolacao
CHORD
CHORD
10
1.0
interpolation
8
approx-m




OM => (31 31 31 31)

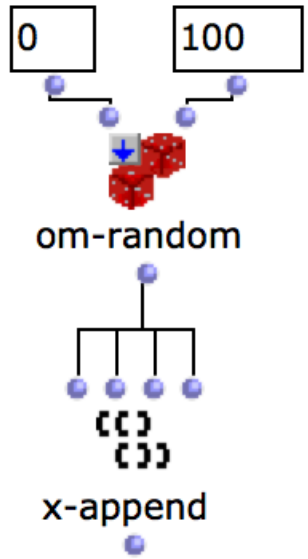



OM => (14 14 14 14)
OM => (54 54 54 54)
OM => (78 78 78 78)



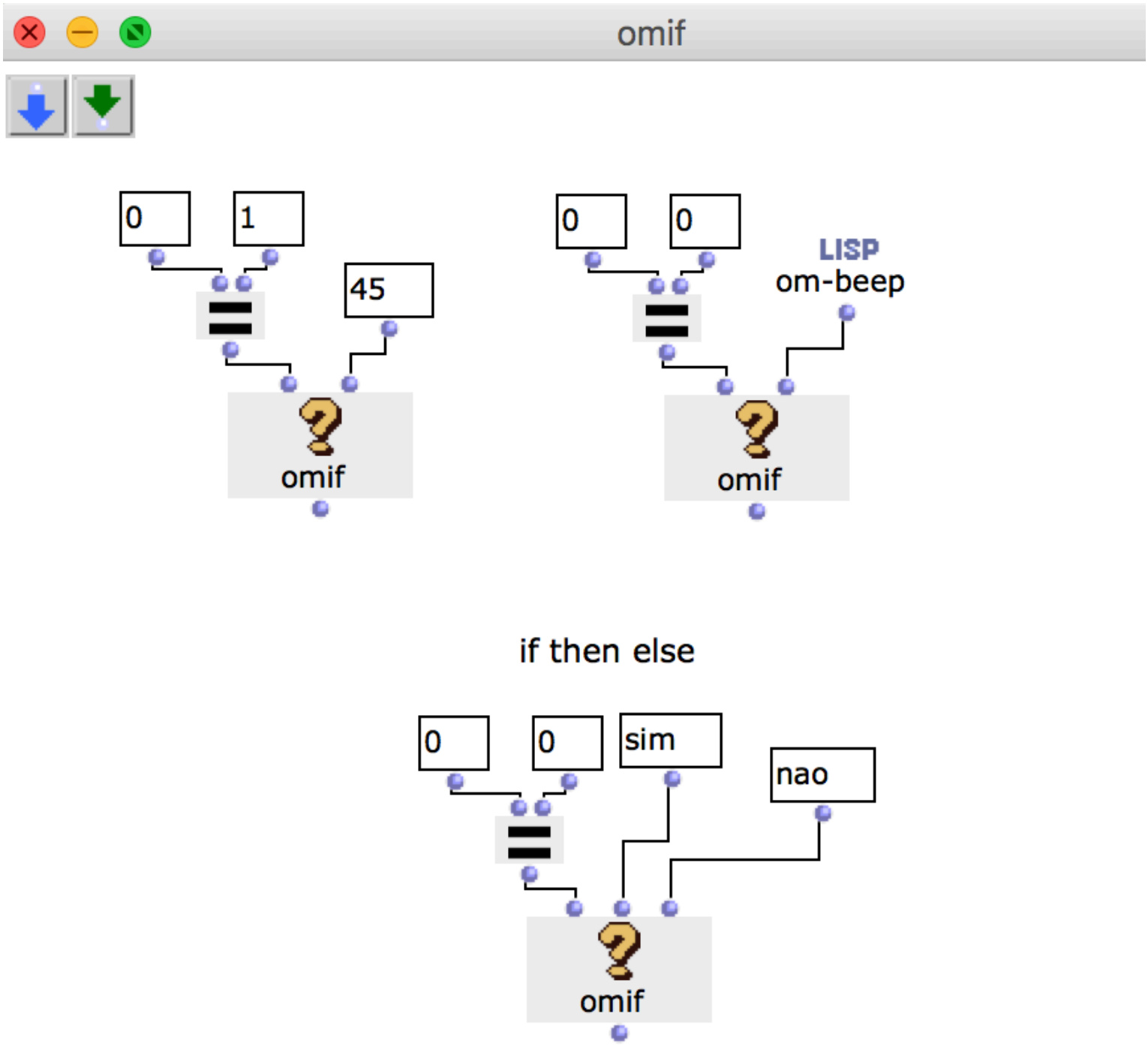

OM => (#<anonymous interpreted function 406000089C> #<anonymous interpreted function 40600008CC> #<anonymous interpreted function 40600008FC> #<anonymous interpreted function 406000092C>)

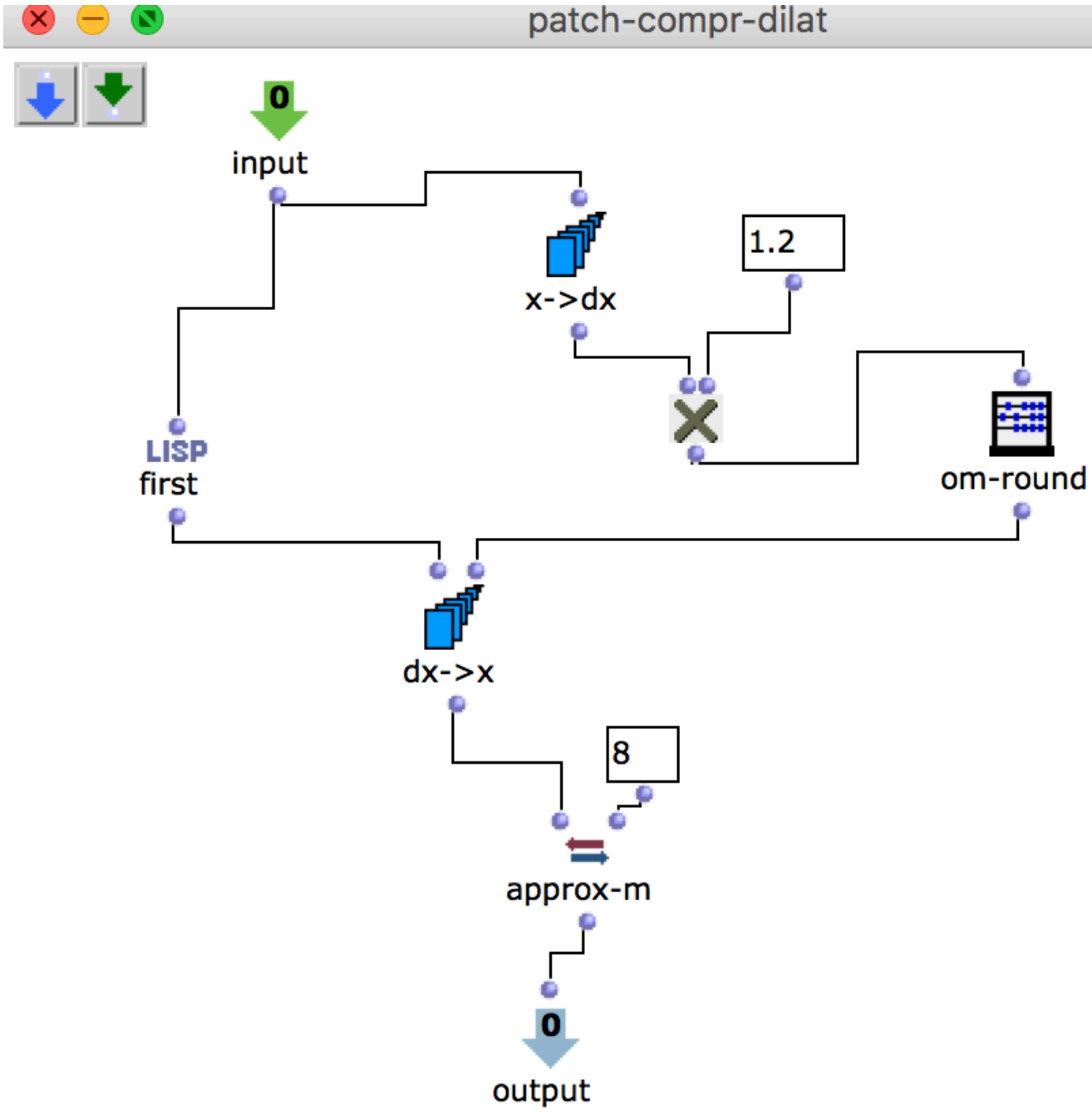
Retorna a funcao (o algoritmo utilizado para operar a informacao)

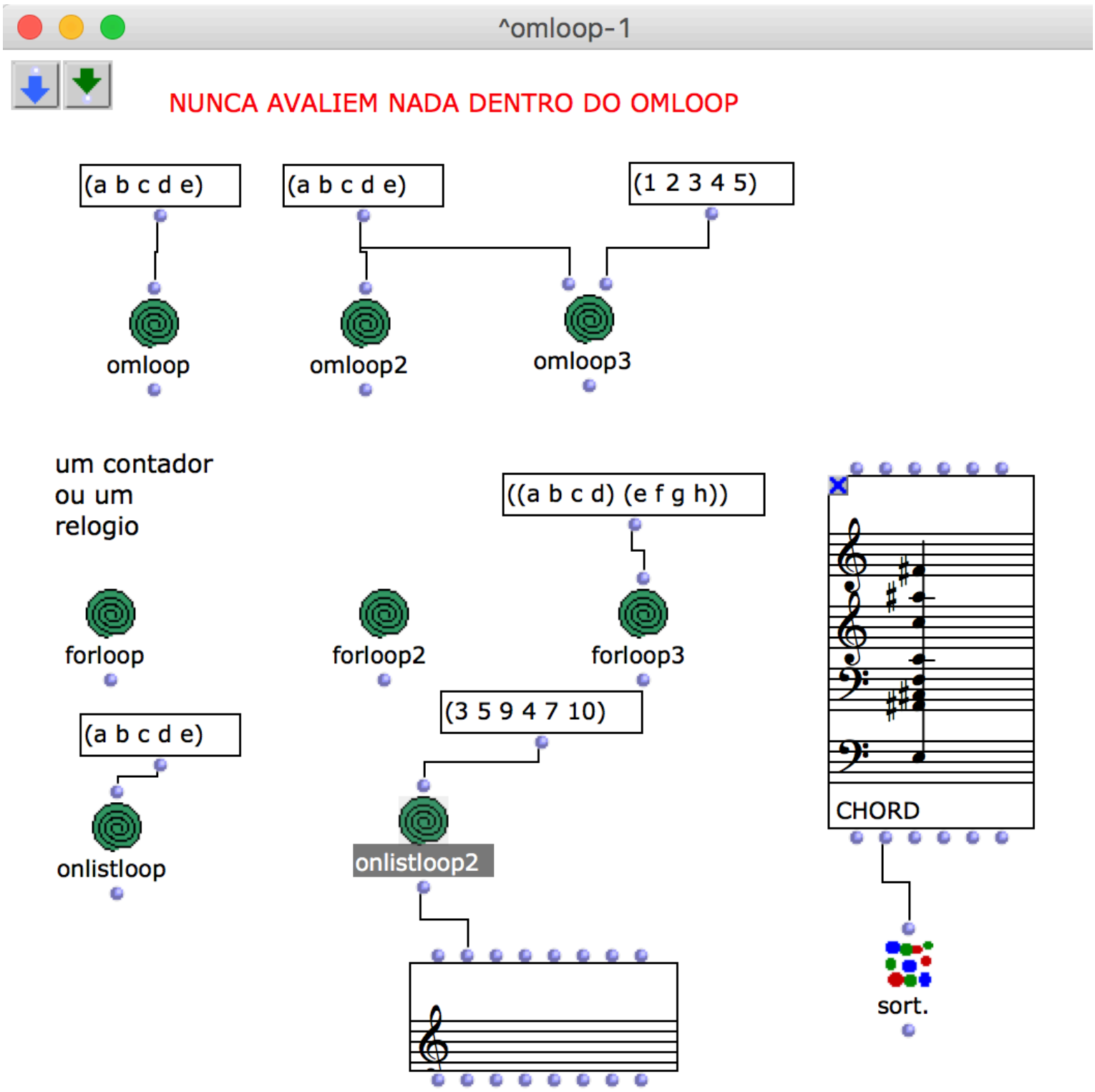


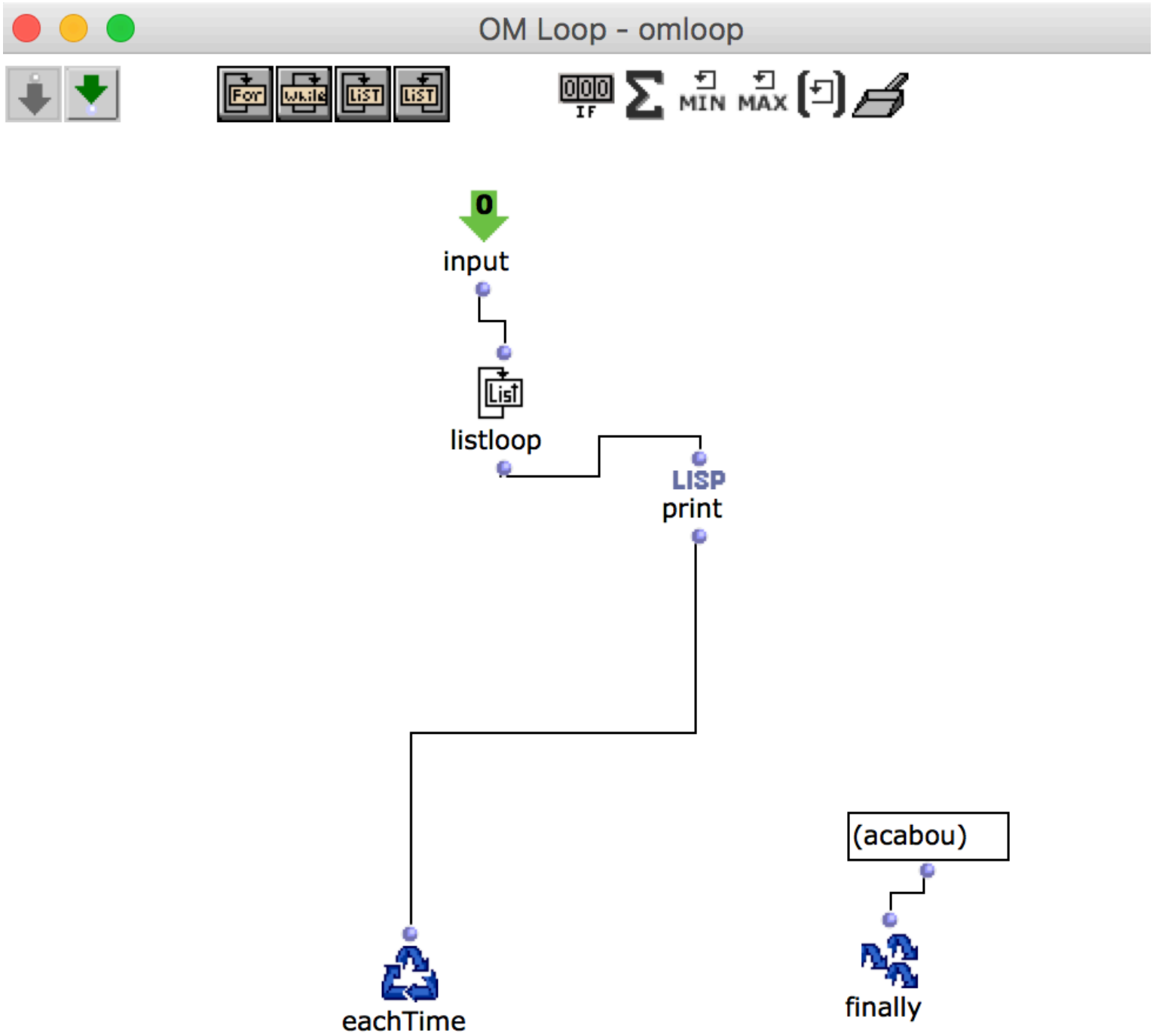

OM => (#<omgenericfunction om-random 40D0C27D9C> #<omgenericfunction om-random 40D0C27D9C> #<omgenericfunction om-random 40D0C27D9C> #<omgenericfunction om-random 40D0C27D9C>)

Retorna o objeto em si (self)

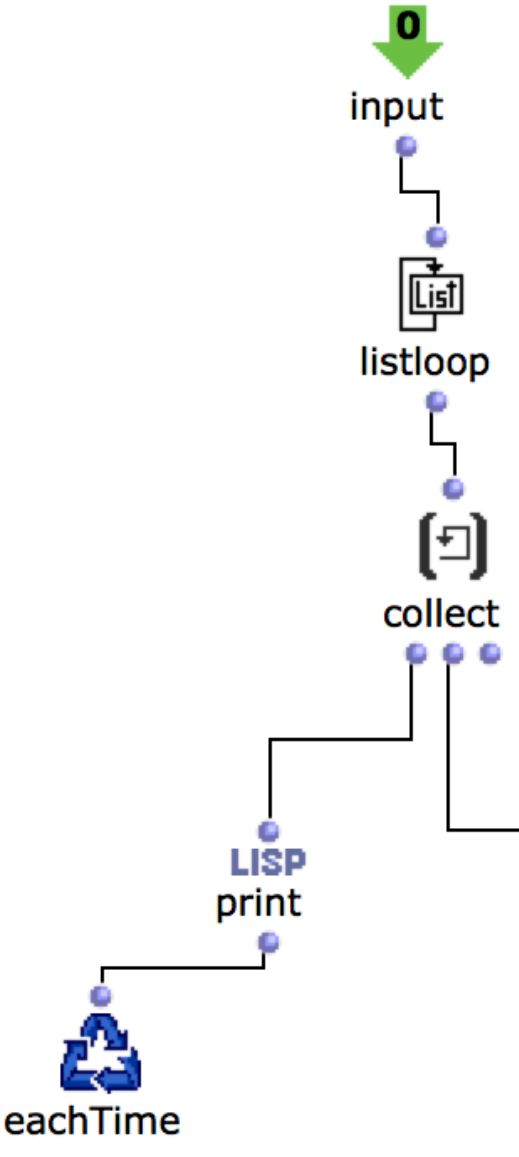

omif
0
1
45
omif
0
0
LISP
om-beep
omif
if then else
0
0
sim
nao
omif


patch-compr-dilat
0
input
x->dx
1.2
LISP
first
om-round
dx->x
8
approx-m
0
output


^omloop-1
NUNCA AVALIEM NADA DENTRO DO OMLOOP
(a b c d e)
(a b c d e)
(1 2 3 4 5)
omloop
omloop2
omloop3
um contador
ou um
relogio
((a b c d) (e f g h))
forloop
forloop2
forloop3
(a b c d e)
(3 5 9 4 7 10)
onlistloop
onlistloop2
CHORD
sort.

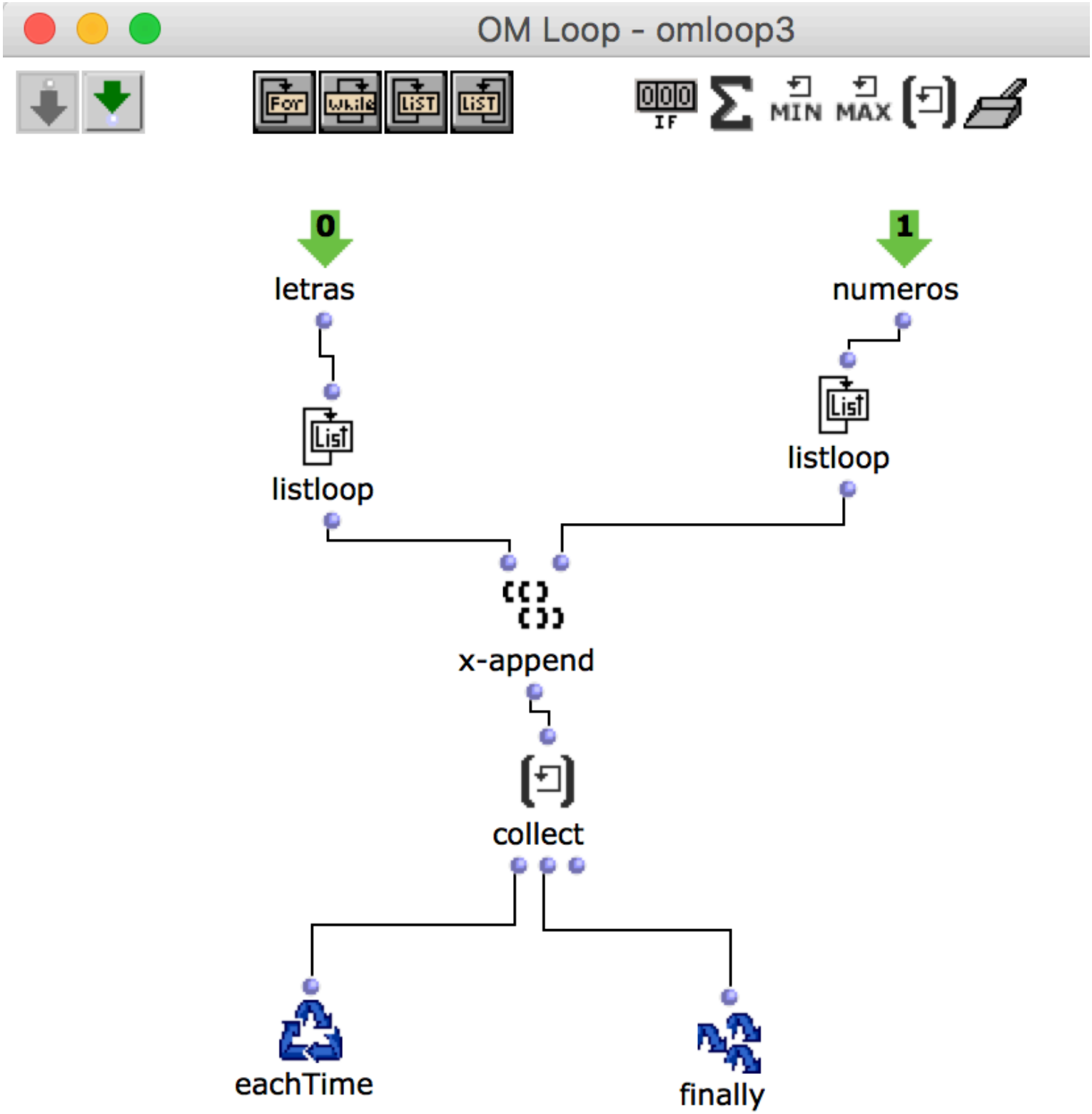

OM Loop - omloop
For
While
List
List
IF
MIN
MAX
0
input
List
listloop
LISP
print
(acabou)
eachTime
finally

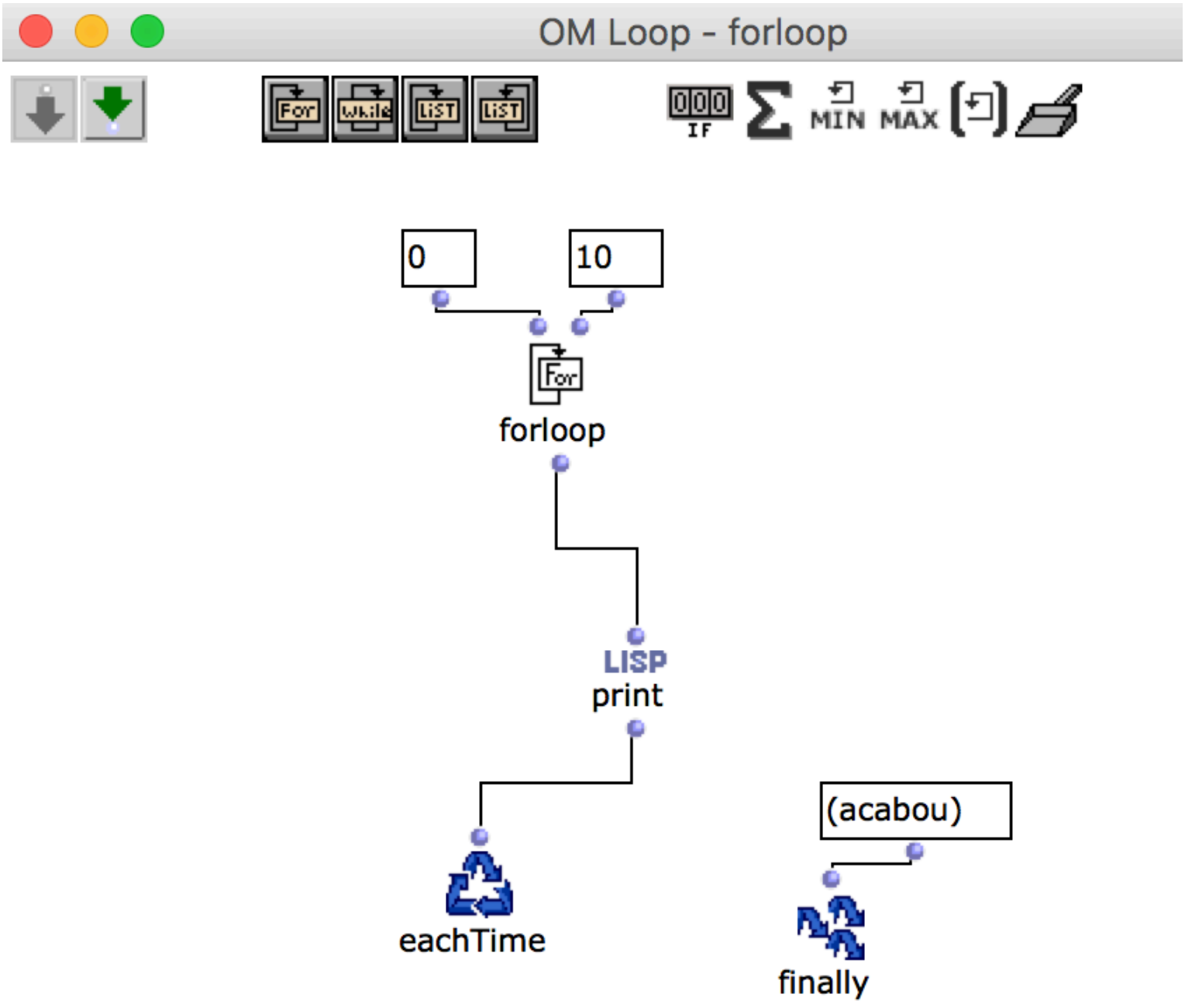
^omloop2

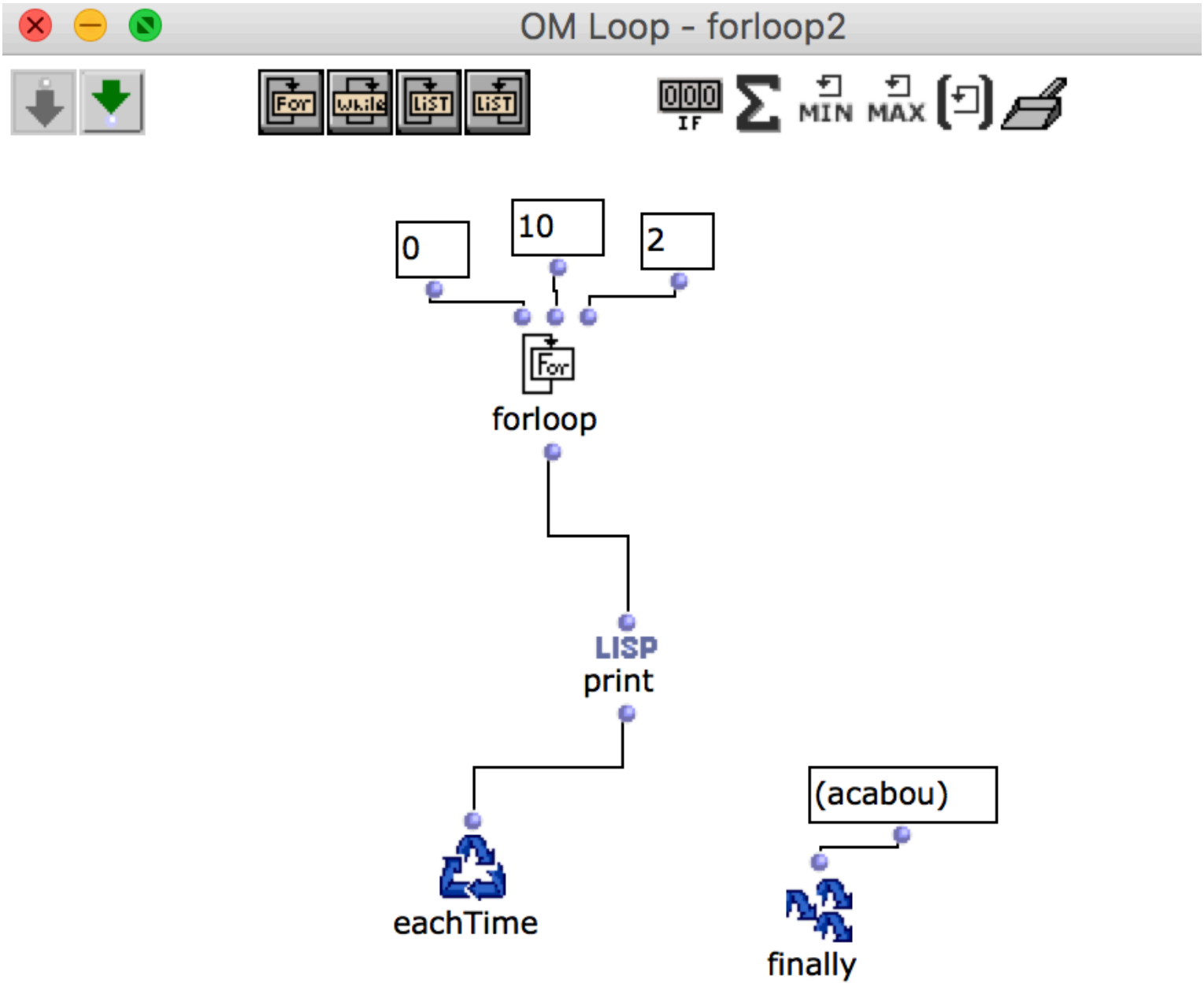



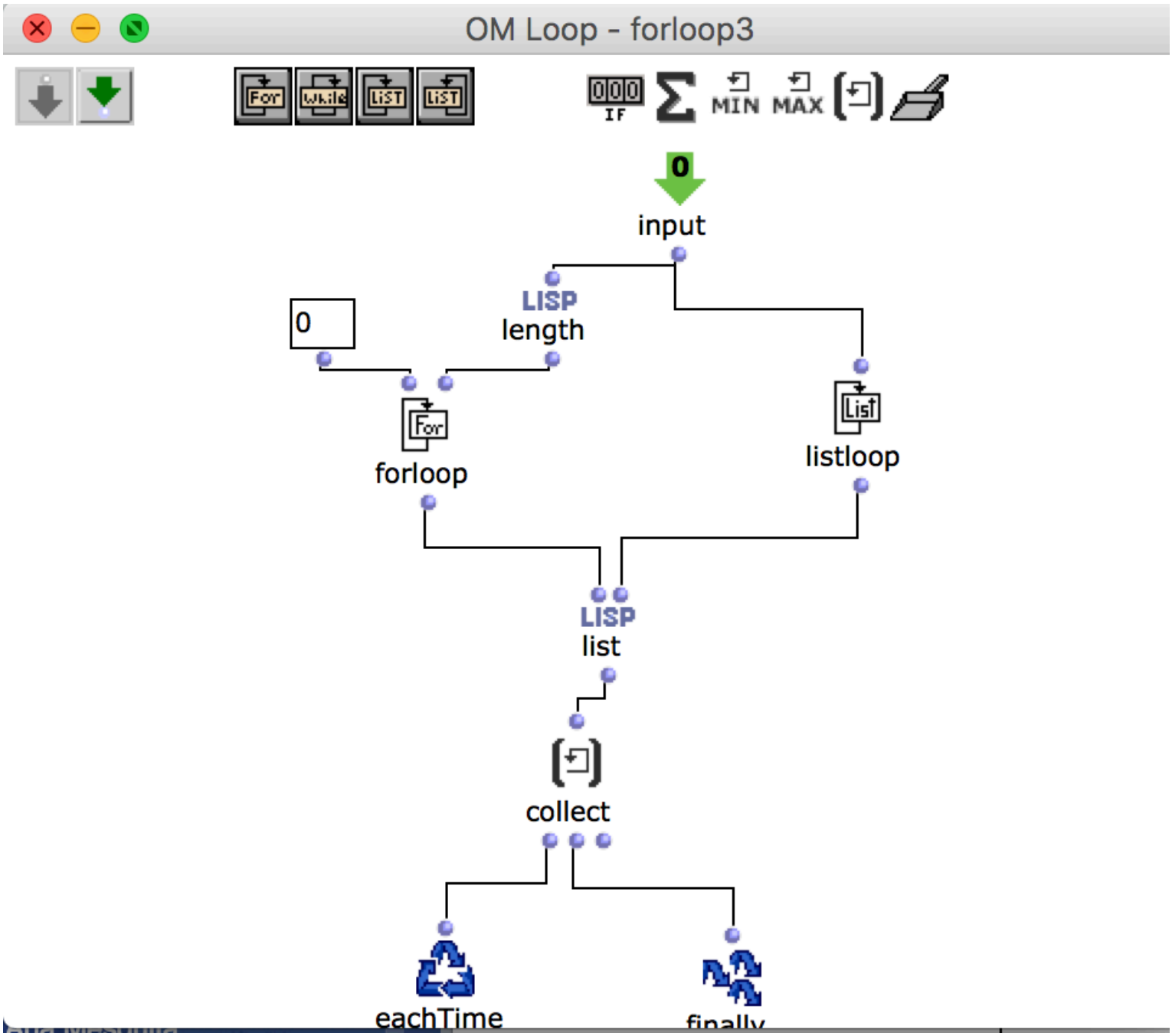

a) enumera uma lista
b) envia uma mensagem no fim da lista
para parar o processo

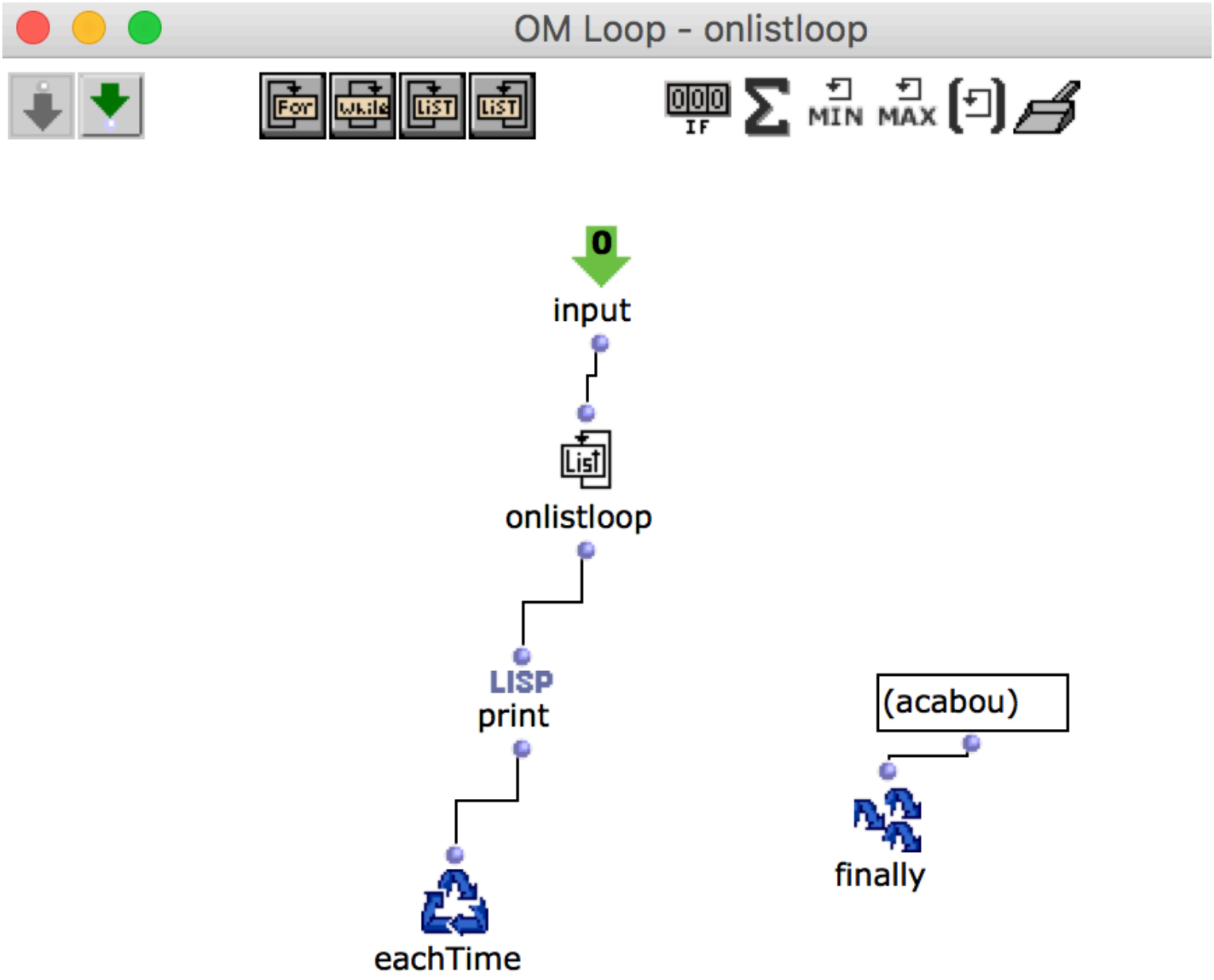
constroi uma lista, que depende da lista que esta sendo avaliada
Saidas:
1 - toma o ultimo valor
2 - faz uma copia do estado interno com todos os valores coletados
3 - reinicializa o collect

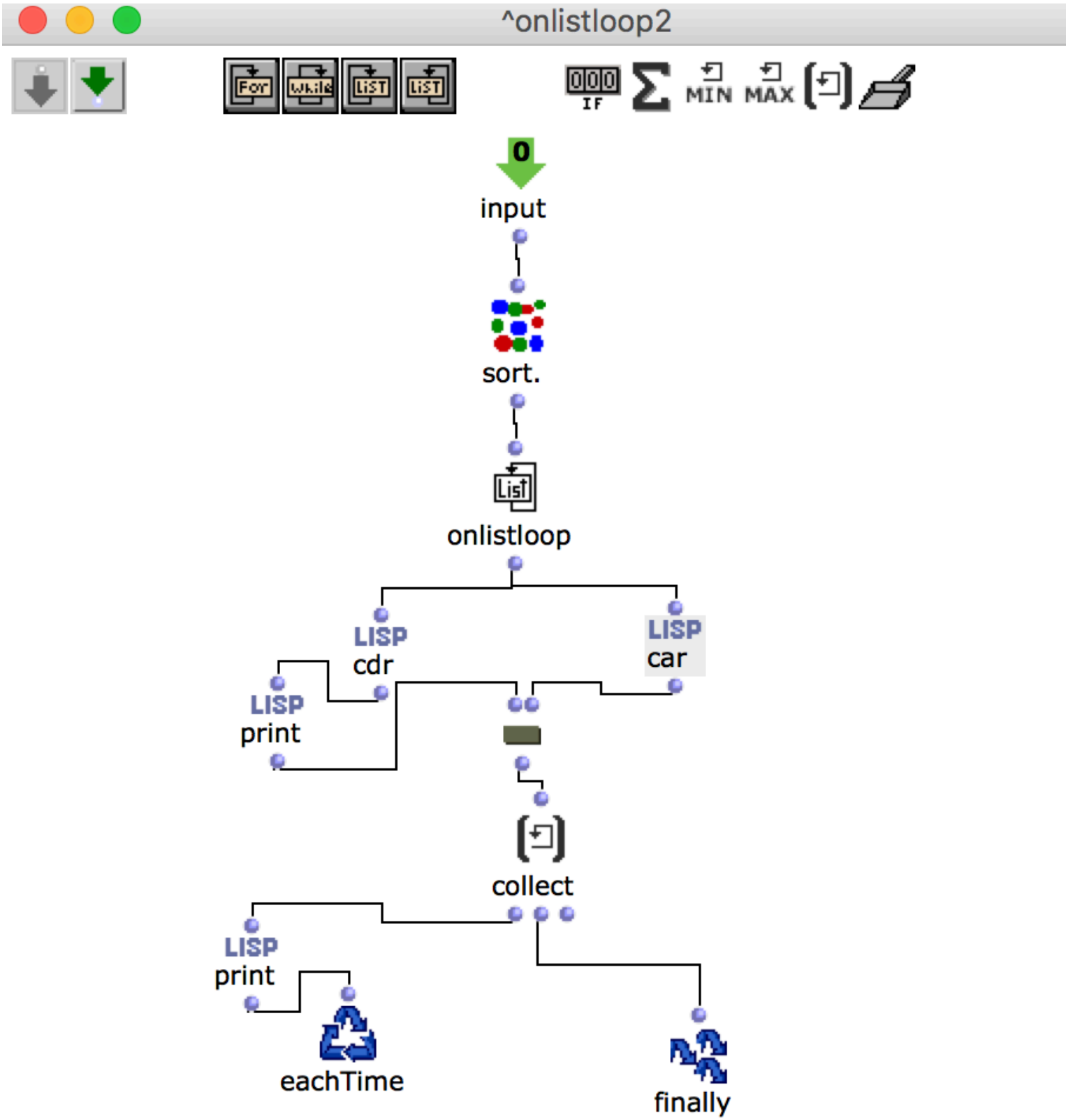

OM Loop - omloop3
For
while
LiST
LiST
IF
MIN
MAX
0
letras
List
listloop
1
numeros
List
listloop
x-append
collect
eachTime
finally

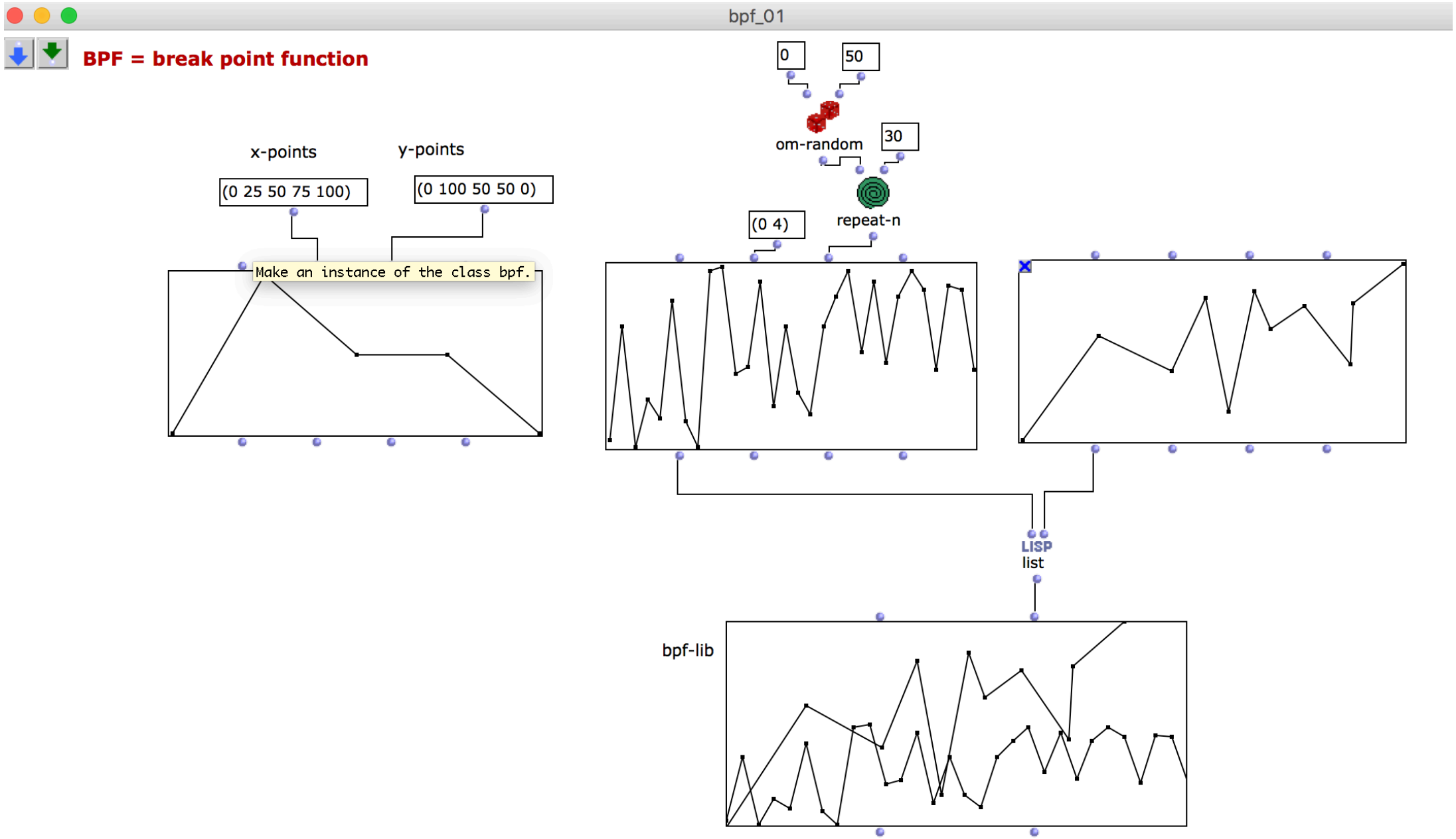

OM Loop - forloop
For
while
LiST
LiST
000
IF
MIN
MAX
0
10
For
forloop
LISP
print
eachTime
(acabou)
finally


OM Loop - forloop2
For
while
LiST
LiST
IF
MIN
MAX
0
10
2
For
forloop
LISP
print
eachTime
(acabou)
finally


OM Loop - forloop3
For
while
LiST
LiST
IF
MIN
MAX
0
input
LISP
length
0
For
forloop
List
listloop
LISP
list
collect
eachTime
finally


OM Loop - onlistloop
For
while
LiST
LiST
IF
MIN
MAX
0
input
List
onlistloop
LISP
print
eachTime
(acabou)
finally


^onlistloop2
For
while
LiST
LiST
000
IF
MIN
MAX
0
input
sort.
List
onlistloop
LISP
cdr
LISP
car
LISP
print
collect
LISP
print
eachTime
finally


bpf_01
BPF = break point function
x-points
y-points
(0 25 50 75 100)
(0 100 50 50 0)
Make an instance of the class bpf.
0
50
om-random
30
repeat-n
(0 4)
LISP
list
bpf-lib

# Série harmonique distordue

Formule générale

Tristan Murail

n

(1 2 3 5 9 13 17)

1

1.11 <= dist.

<= om^

mc->f

f->mc

(0 1024 2048 3072)

(1 2 3 4)

gm-program

OUT

pitchwheel

OUT

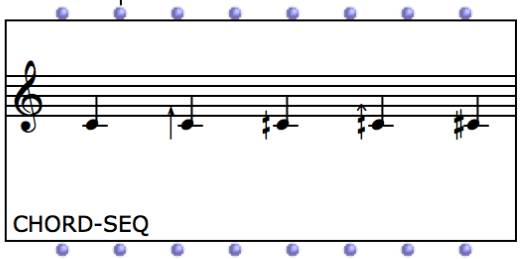
pgmout

sequence

Huitieme_de_ton

0

(6000 6025 6050 6075 6100)




huitième de ton

0

(1 2 3 4)

gm-program

(0 1024 2048 3072)

OUT

pitchwheel

OUT

pgmout

sequence

sixième de ton

(5 6 7)

gm-program

(0 1351 2703)

OUT

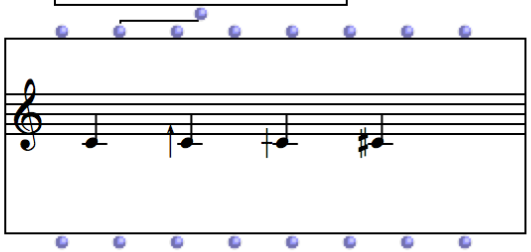
pitchwheel

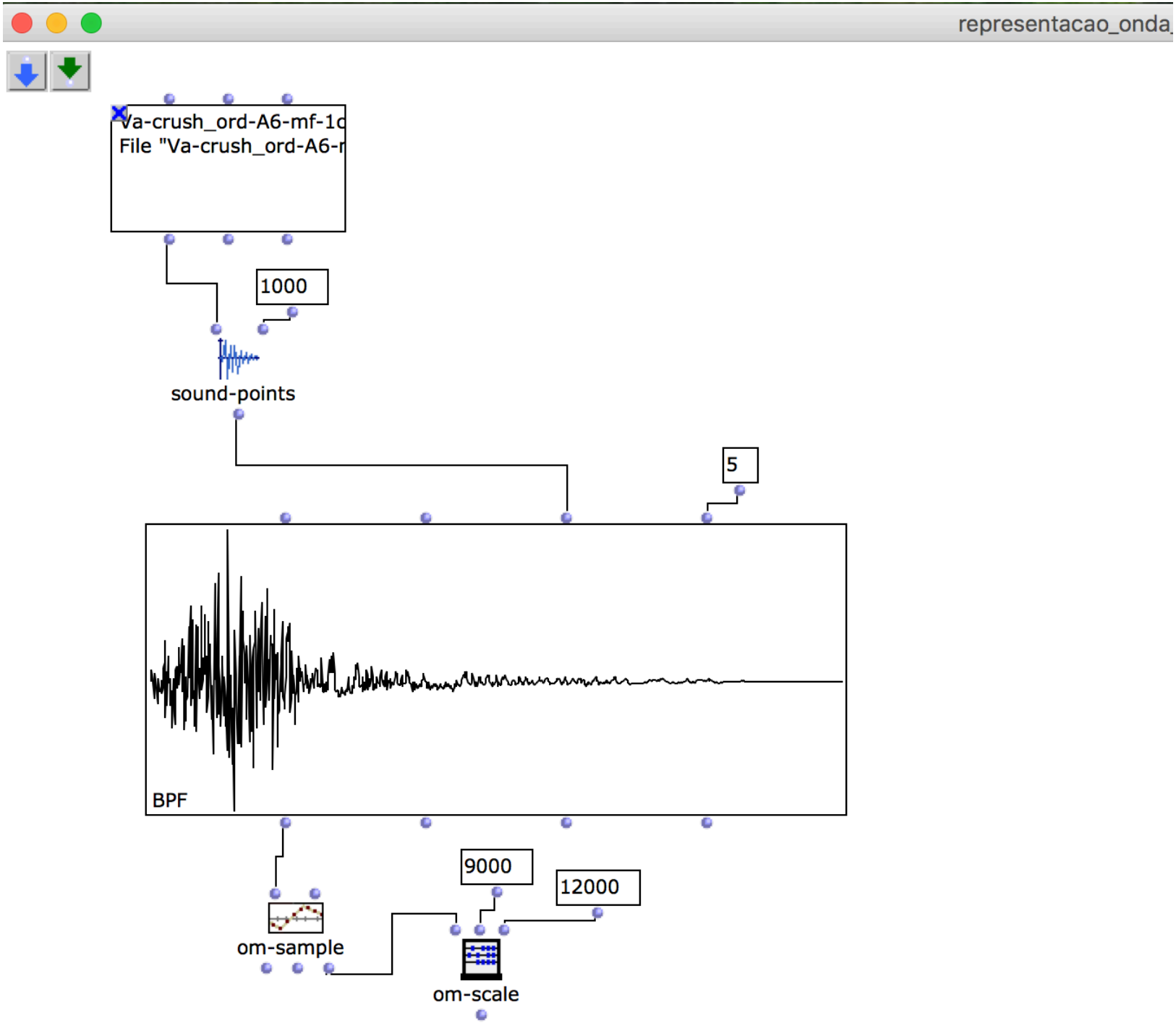
OUT

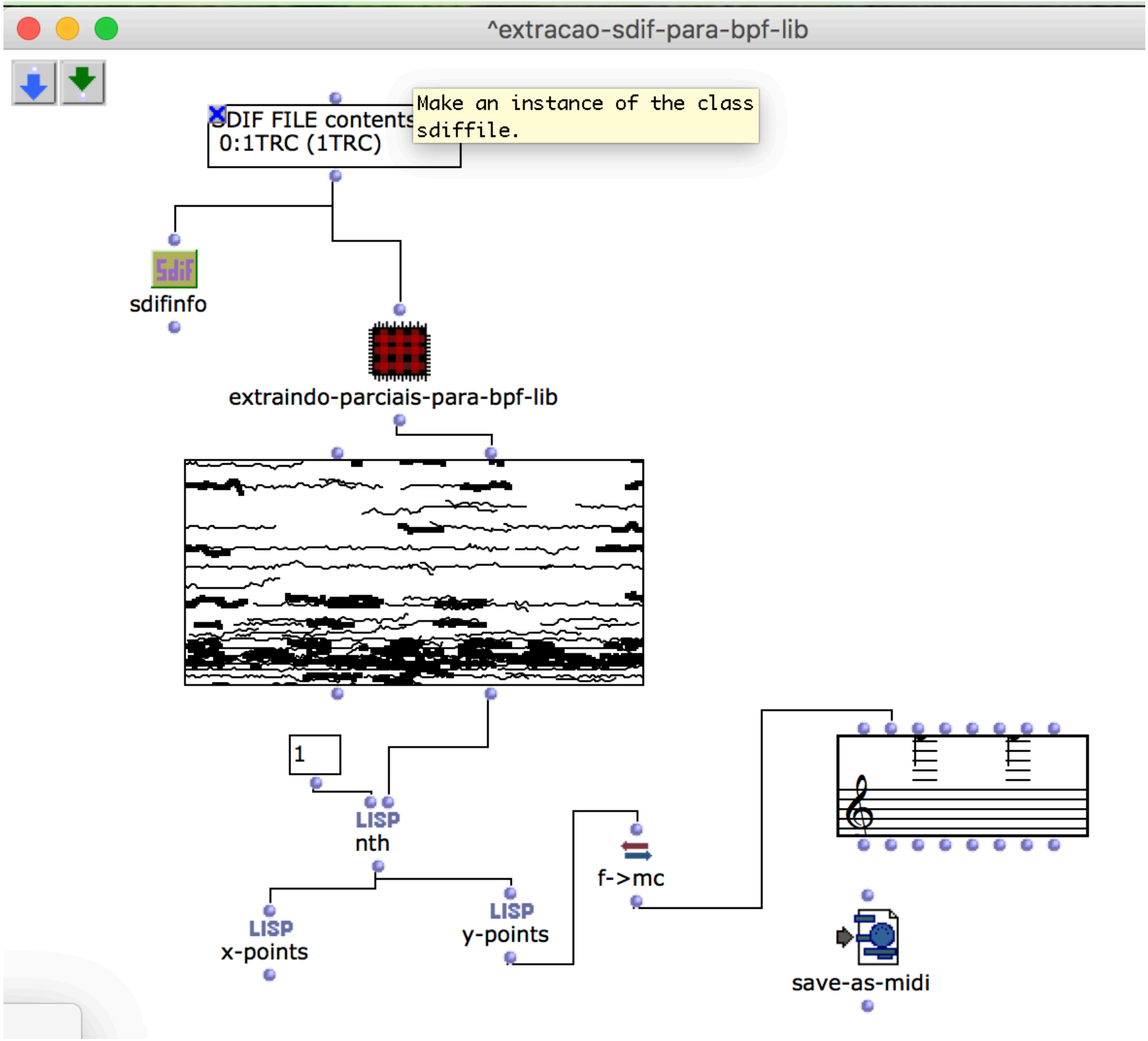
pgmout

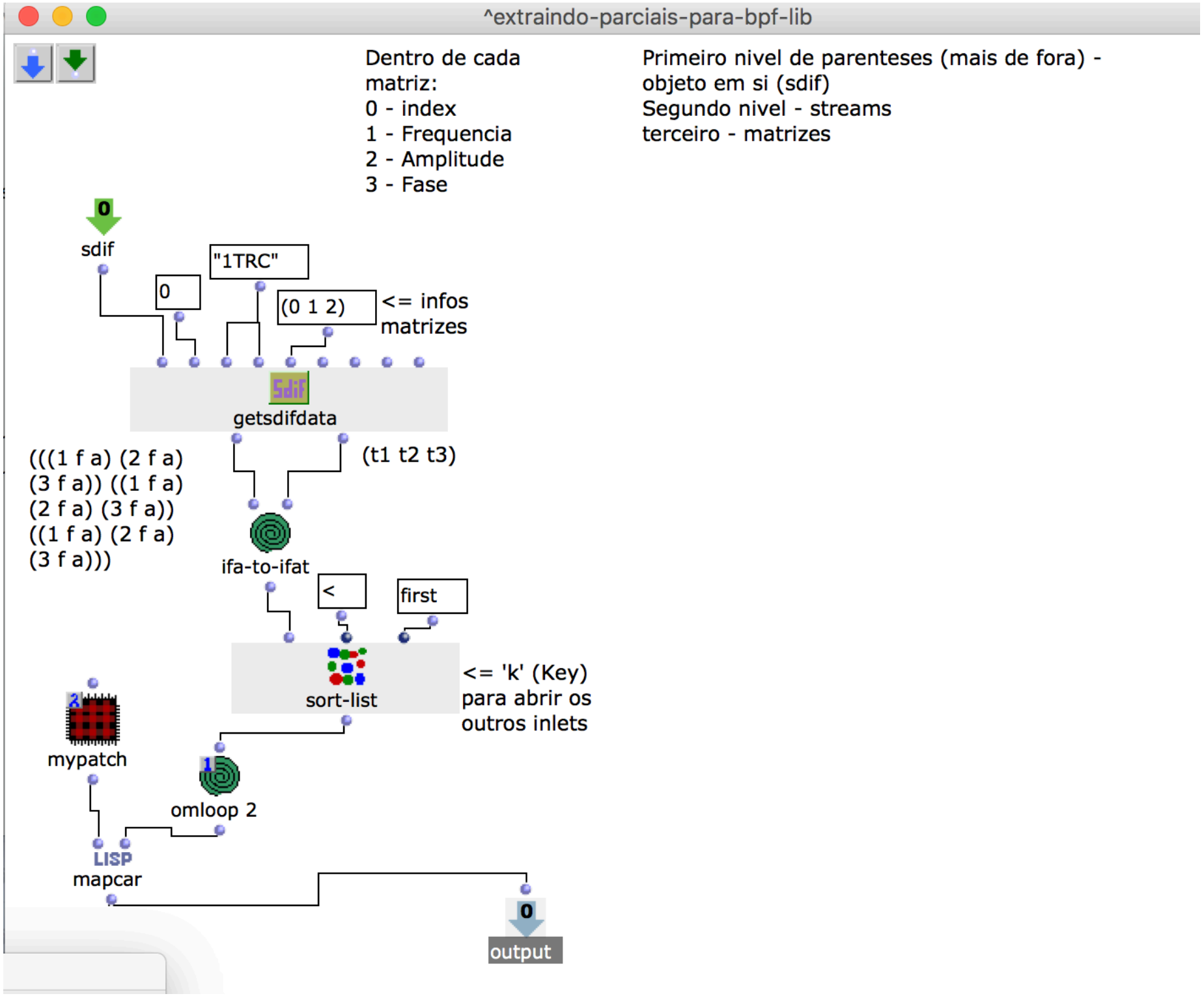
sequence

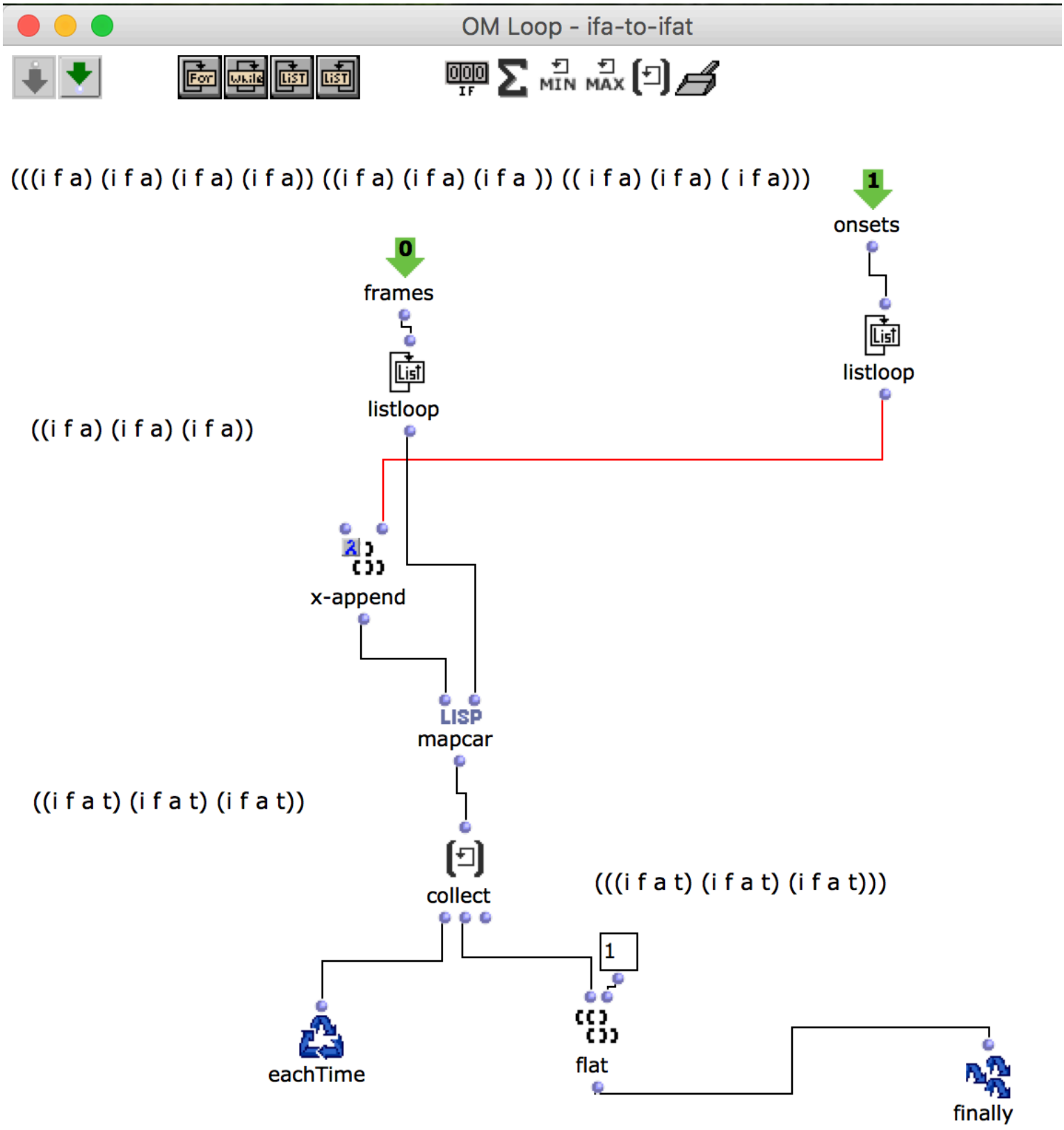
(6000 6033 6066 6100)

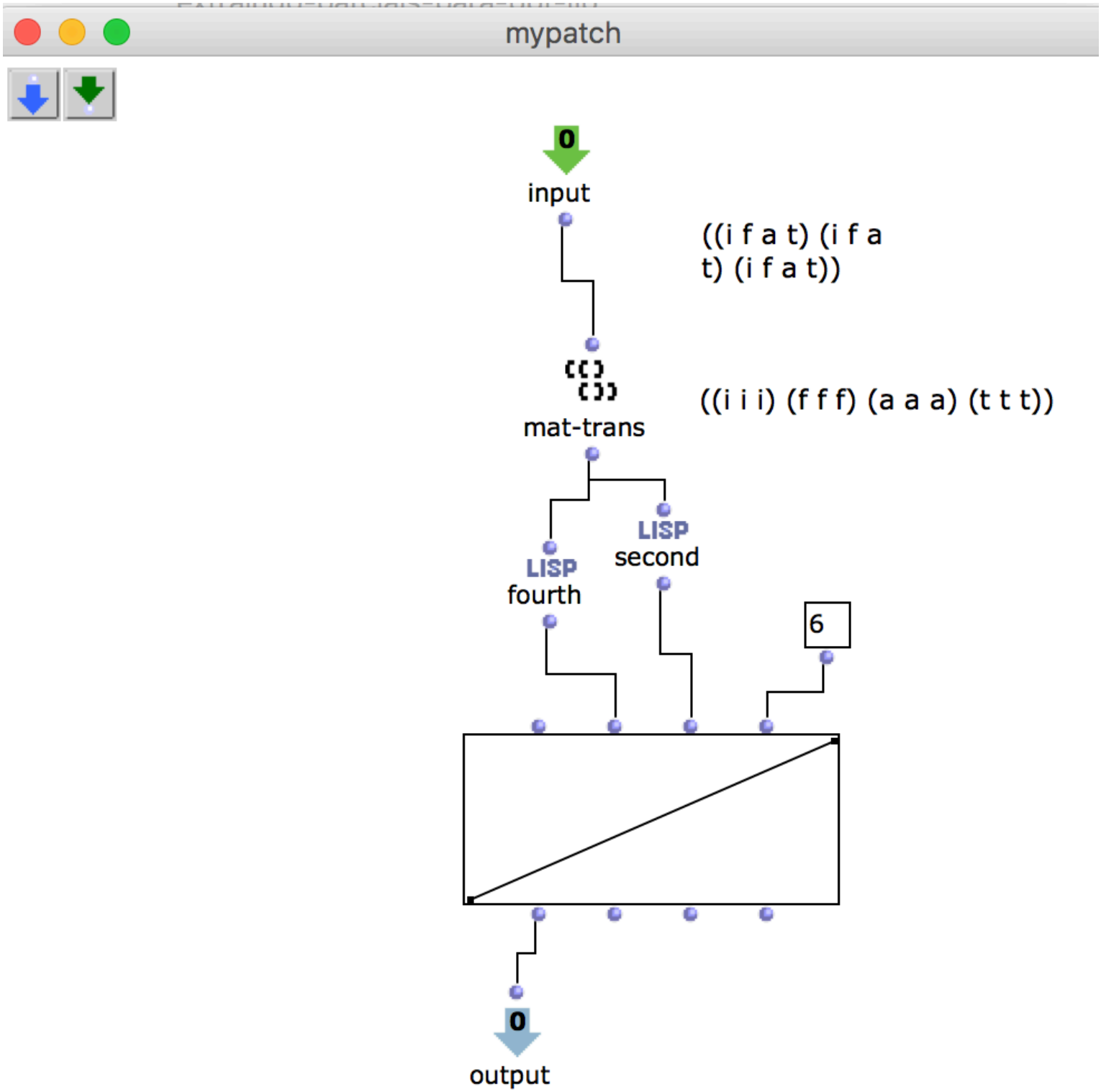


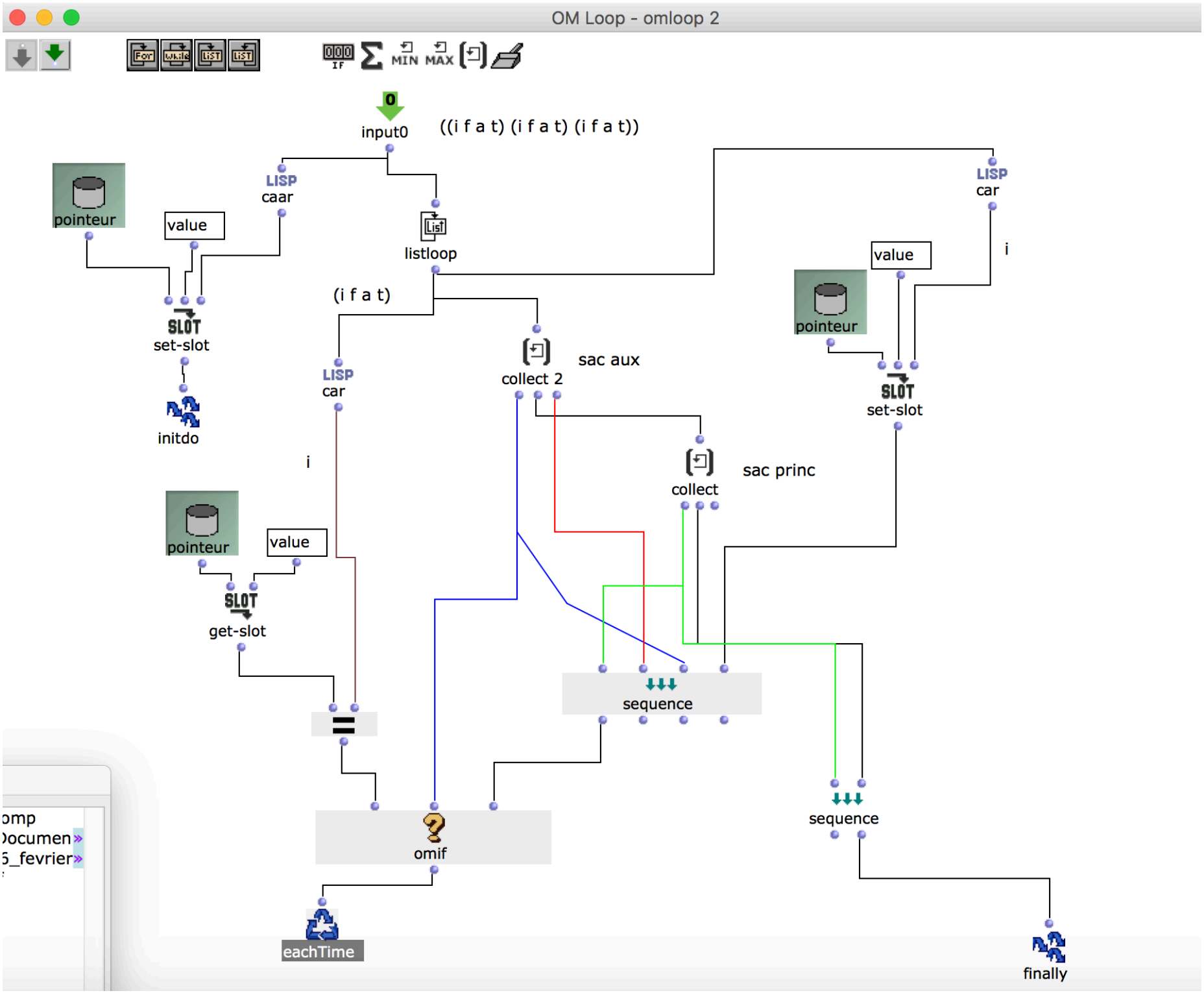

representacao_onda
Va-crush_ord-A6-mf-1c
File "Va-crush_ord-A6-r
1000
sound-points
5
BPF
9000
12000
om-sample
om-scale


^extracao-sdif-para-bpf-lib
SDIF FILE contents
0:1TRC (1TRC)
Make an instance of the class sdiffile.
sdifinfo
extraindo-parciais-para-bpf-lib
1
LISP
nth
LISP
x-points
LISP
y-points
f->mc
save-as-midi


^extraindo-parciais-para-bpf-lib
Dentro de cada matriz:
0 - index
1 - Frequencia
2 - Amplitude
3 - Fase
Primeiro nivel de parenteses (mais de fora) - objeto em si (sdif)
Segundo nivel - streams
terceiro - matrizes
0
sdif
"1TRC"
0
(0 1 2)
<= infos matrizes
SdiF
getsdifdata
(((1 f a) (2 f a) (3 f a)) ((1 f a) (2 f a) (3 f a)) ((1 f a) (2 f a) (3 f a)))
(t1 t2 t3)
ifa-to-ifat
<
first
sort-list
<= 'k' (Key) para abrir os outros inlets
mypatch
1
omloop 2
LISP
mapcar
0
output


OM Loop - ifa-to-ifat
For
While
List
List
IF
MIN
MAX
(((i f a) (i f a) (i f a) (i f a)) ((i f a) (i f a) (i f a )) (( i f a) (i f a) ( i f a)))
1
onsets
0
frames
List
listloop
List
listloop
((i f a) (i f a) (i f a))
x-append
LISP
mapcar
((i f a t) (i f a t) (i f a t))
collect
(((i f a t) (i f a t) (i f a t)))
1
eachTime
flat
finally


mypatch
0
input
((i f a t) (i f a
t) (i f a t))
mat-trans
((i i i) (f f f) (a a a) (t t t))
LISP
fourth
LISP
second
6
0
output


OM Loop - omloop 2
input0
((i f a t) (i f a t) (i f a t))
LISP
caar
LISP
car
pointeur
value
listloop
value
i
pointeur
SLOT
set-slot
(i f a t)
SLOT
set-slot
initdo
LISP
car
collect 2
sac aux
i
collect
sac princ
pointeur
value
SLOT
get-slot
sequence
omif
sequence
eachTime
finally