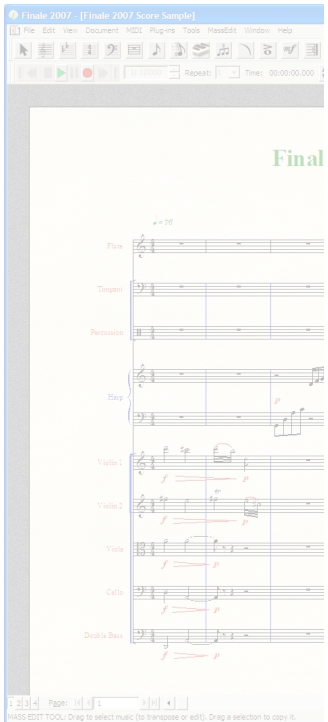
PHELMA - INP Grenoble
Master AST (Art, Science et Technologie)

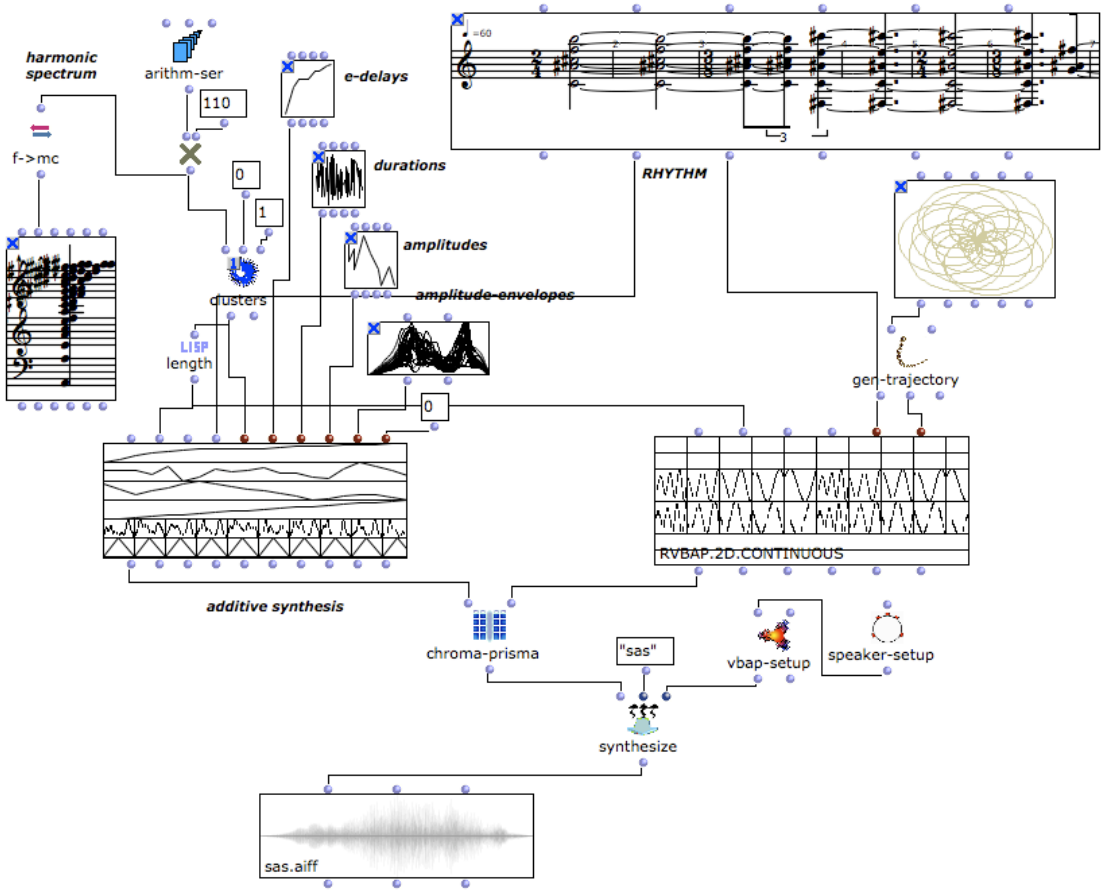
Composition Assistée par Ordinateur
(⇄ Programmation Visuelle)

Carlos Agon - Moreno Andreatta - Jean Bresson
IRCAM - Equipe Représentations Musicales

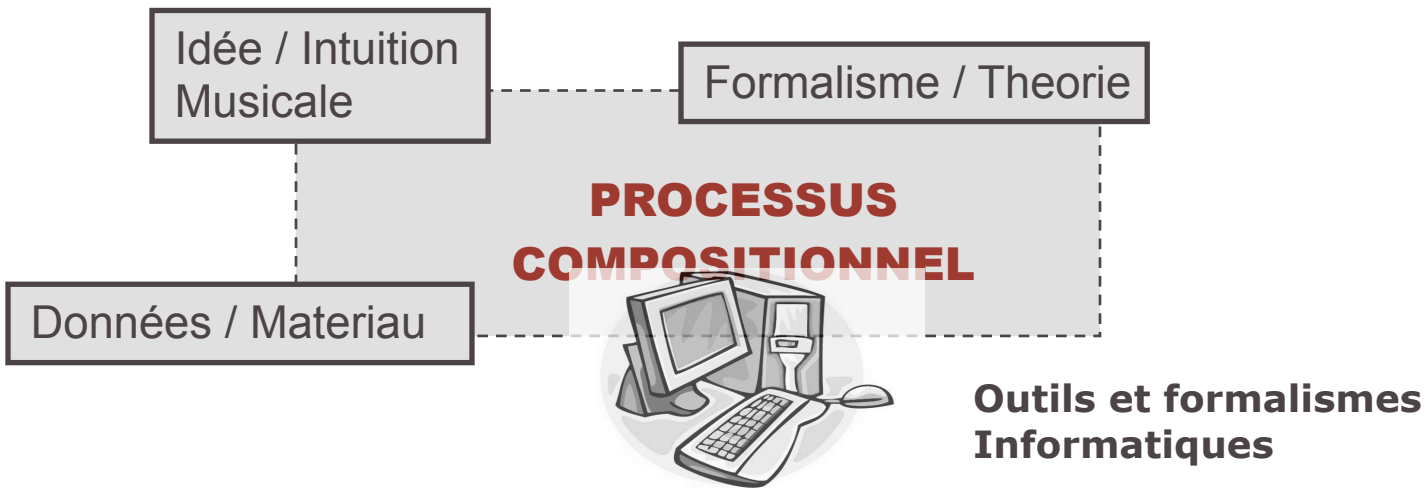
ircam
Centre Pompidou

Environnements de composition assistée par ordinateur...





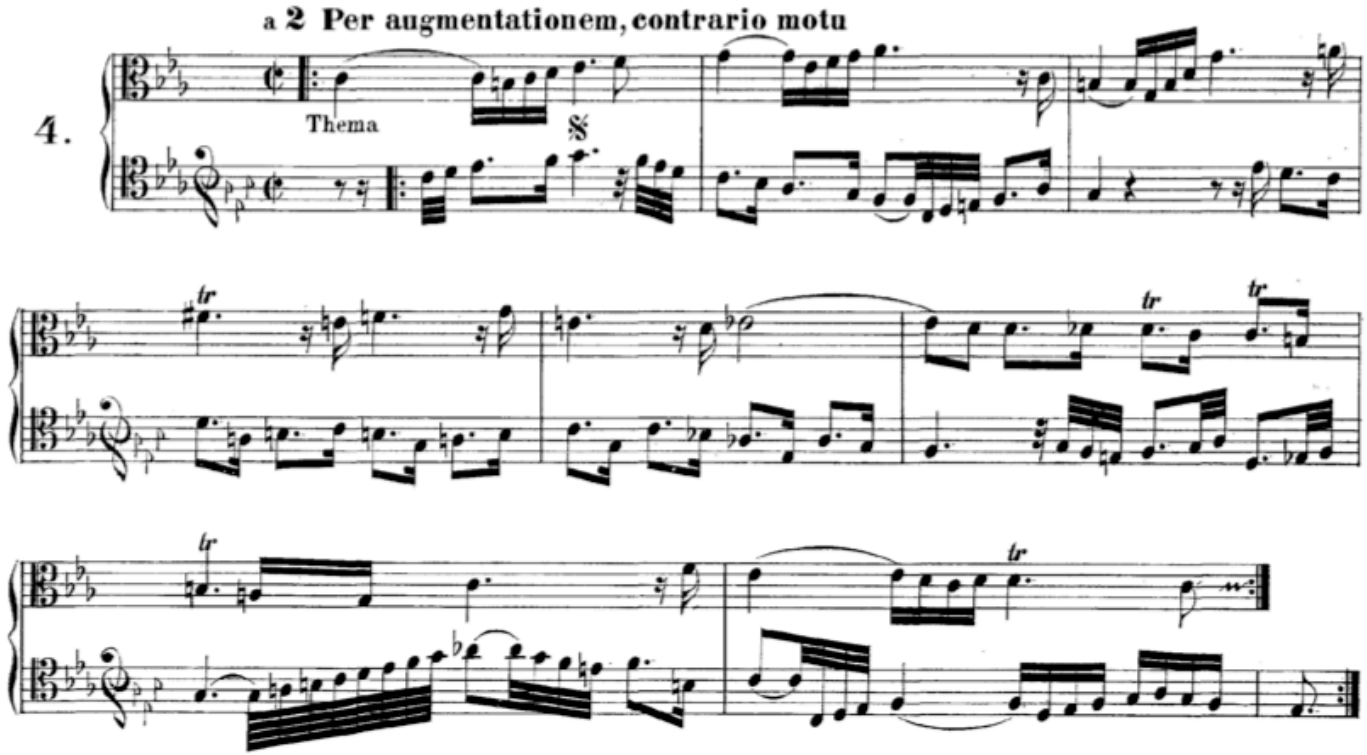
Composition Assistée par Ordinateur (CAO)



G. Assayag & J.P. Cholleton, *L'appareil musical* (1994) : "*L'objectif général est la définition de modèles informatisés utilisables dans des situations où le compositeur désire préparer des matériaux musicaux complexes et structurés, relativement à une certaine formalisation ou un certain ensemble de contraintes qui lui sont propres et qu'il est en mesure d'exprimer de façon cohérente.*"

J.-C. Risset, *Musique, un calcul secret ?* (1977) : "*Dialogue efficace avec la machine afin de contrôler ou agencer détails ou grandes lignes [...] et permettre au musicien de se construire son propre monde.*"

J-S Bach



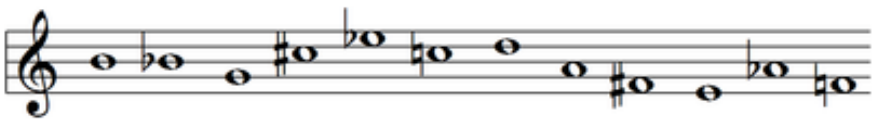
Travail formel

Propriétés combinatoires

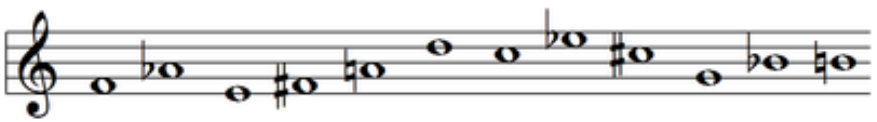
(2e école de Vienne, début du XXe siècle)

Example

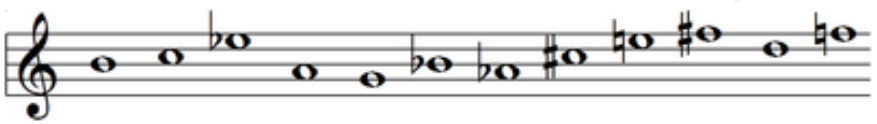
Suppose the prime series is as follows:



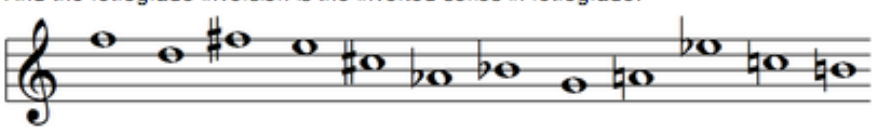
Then the retrograde is the prime series in reverse order:

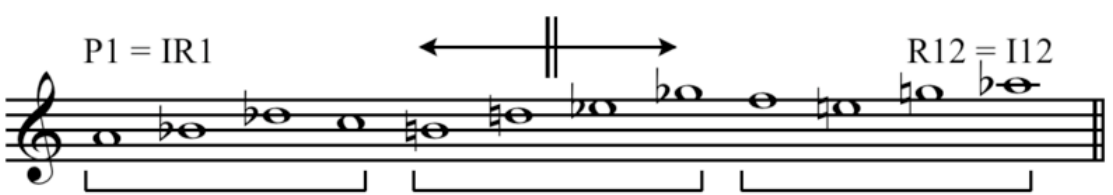


The inversion is the prime series with the intervals inverted (so that a rising minor third becomes a falling minor third):



And the retrograde inversion is the inverted series in retrograde:





Histoire de la CAO: De la musique algorithmique à la musique spectrale...

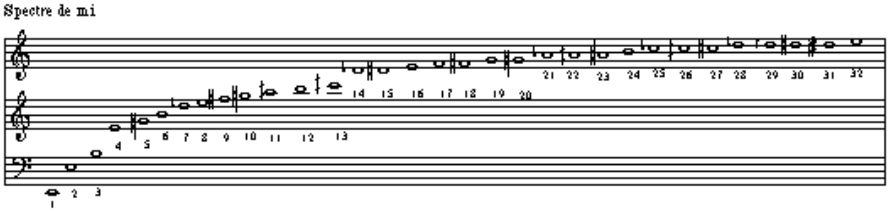
Composition Algorithmique (1958 - ...)

Pierre Barbaud, Iannis Xenakis, André Riotte & Marcel Mesnage, David Cope, ...
~> Programmer l'ordinateur pour qu'il crée de la musique

Musique spectrale et prémices de la CAO contemporaine..

1975: Partiels (Gerard Grisey)

-Matériau de départ: analyse (FFT) d'une note (Mi grave) jouée au trombone.
(fréquences en Hz, durées en sec)



⇒Exploration du contenu spectral par l'orchestre

⇒Transformations du matériau inspirées par des processus electro-acoustiques
(distorsion, modulation en anneau, filtrage...)

[1'16"]

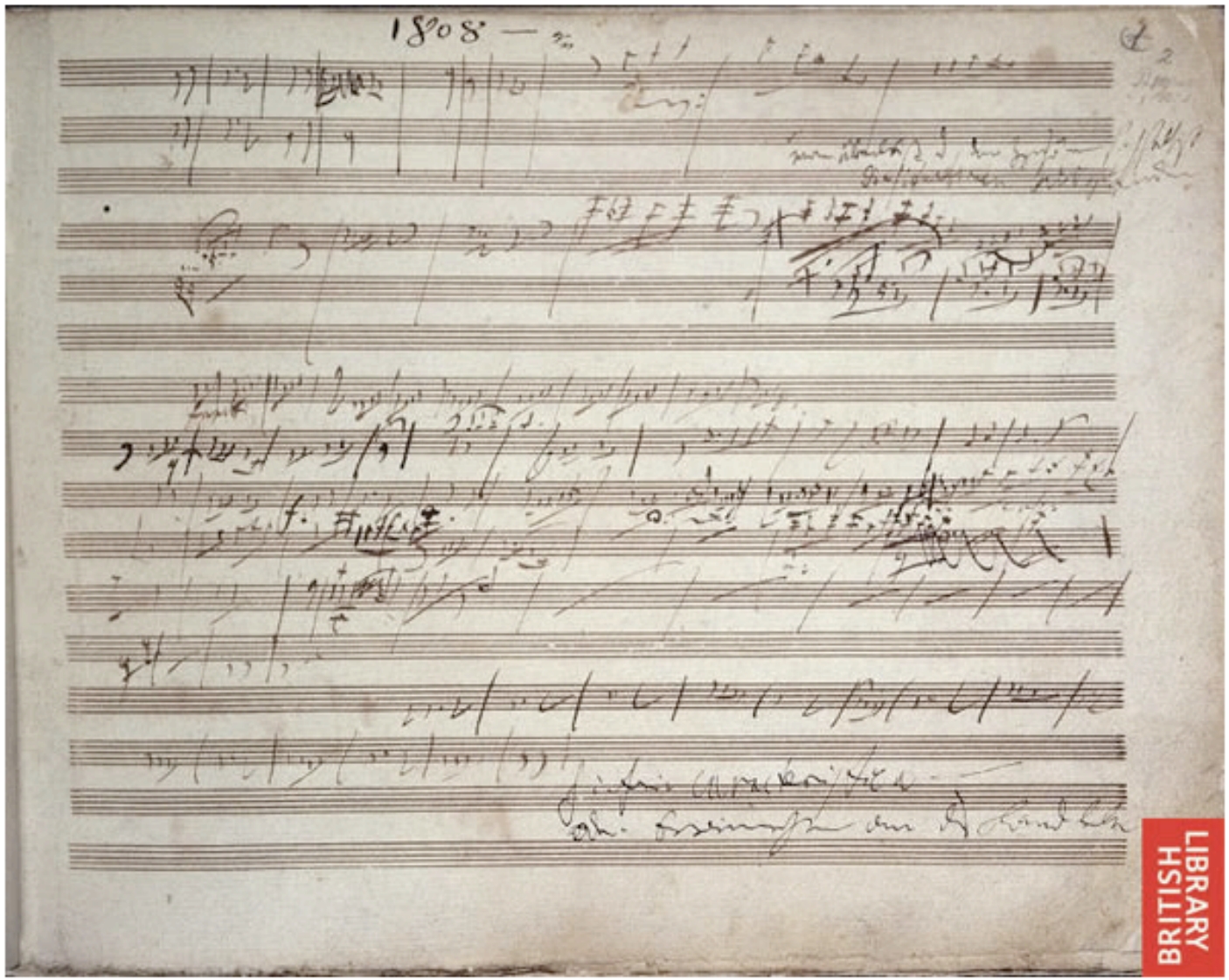
Développement de la musique "spectrale":

⇒ intérêt d'une assistance informatique pour l'expérimentation, le traitement et le rendu de matériaux et données musicales complexes.

T. Murail, C. Malherbe, etc.

Otto Laske, *Composition Theory in Koenig's Project One and Project Two*. Computer Music Journal (1981):
"We may view composer-program interaction along a trajectory leading from purely manual control to control exercised by some compositional algorithm (composing machine). The zone of greatest interest for composition theory is the midd le zone of the trajectory, since it al low a great flexibility of approach. The powers of intuition and machine computation may be combined."

Travail sur les esquisses



Le compositeur / programmeur

⇒ CAO : permettre au compositeur d'exprimer, de représenter et d'exécuter les processus de modélisation / création de structures musicales

Programme = Représentation symbolique
d'un objet / processus musical
d'un modèle compositionnel
Décrit des *intentions* musicales à travers un **langage**

Environments de CAO <==> **Langages (de programmation)**

G. Assayag, *Computer Assisted Composition Today* (1998) : "*Nous concevons un tel environnement [de CAO] comme un langage de programmation spécialisé que les compositeurs utiliseront pour construire leur propre univers musical. [...] Ceci nous amène à réfléchir aux différents modèles de programmation existants, aux représentations, interne et externe, des structures musicales qui seront construites et transformées par cette programmation.*"

Langages de programmation pour la musique

Son / Audio

Music N (M. Matthews, Bell Labs, 1960s)
Csound (B.Vercoe, MIT, 1980s)
...

Faust (Orlarey, Grame, 2002)

Max/MSP / PureData
(M. Puckette, IRCAM 1991, UCSD, 1998)

Fonctionels

Lisp

Artic, Nyquist
(R. B. Dannenberg, CMU, 1989, 1997)

SuperCollider (McCartney, 1998)

Formes (Rodet *et al., IRCAM, 1982-85)*

Common Music (H.Taube, CCRMA, 1991)

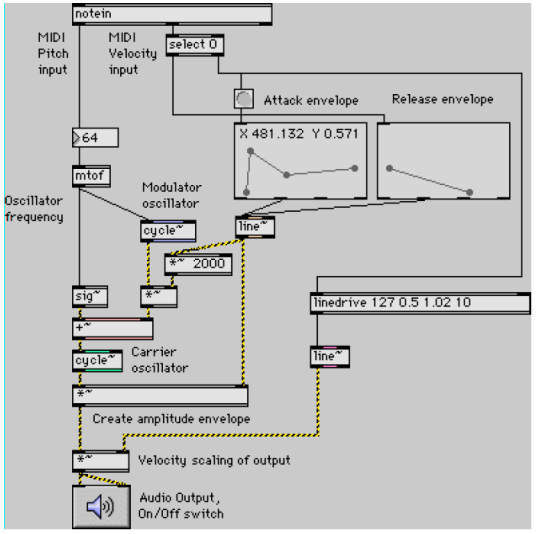
Symbolique

PatchWork (M. Laurson *et al.*, IRCAM, 1989)
OpenMusic (C.Agon *et al.*, IRCAM, 1998)
PWGL (M. Laurson *et al.*, Sybelius Ac., 2002)

Elody (Y. Orlarey *et al.*, Grame, 1997)

```
(de rest? (in-n-notes)
  (when (eq master (fself name))
    (let ((notes (fself?'active-sons)) (t-sum 0))
       (let ((a-note (nthcdr in-n-notes notes)))
          (when (memq a-note'(short-rest'long-rest))
            (repeat in-n-notes (+=t-sum ((nextl notes)?'duration)))
            (ajuste (the other master) in-n-notes (plus time t-sum))
            (when (gt (random 06) 3)
            (setq master (alter master ))))))))
(de the-other (voice)
  (if (eq voice 'VOICE1) 'VOICE2 'VOICE1 ))
(de ajuste (voice in-n-notes the-time)
  (let ((reste ((car (voice?'active-sons))?etime))
        (tsum () ) (list-t () ) (list-o () )
          (r-s (cdr (voice?'active-sons) )))
     (setq tsum reste)
     (let ( (r-s r-s))
        (when (and (lt tsum the-time) r-s)
          (setq tsum (plus tsum (newl list-t
            (apply (get (newl list-o (nextl r-s)) 'duration:) ) ))))
          (self r-s)))
     (if (setq aux (sub tsum reste) )
     (setq fact (div (sub the-time reste) (-tsum reste) )))
     (mapc list-o (lambda (o) (put o'duration:[lambda()
                (mul fact (nextl list- t))]))
)))
```

Langages Visuels



Langages de Programmation Visuels (VPLs)

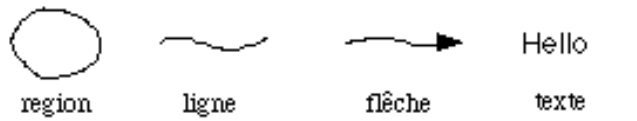
Sémantique à **+ de 1 dimension**

⇒ Eléments visuels supplémentaires (couleurs, symboles, formes, images + relations spatiales)

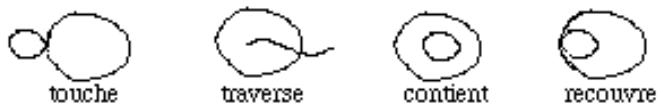
Icônes

Exemple:

Primitives



Relations

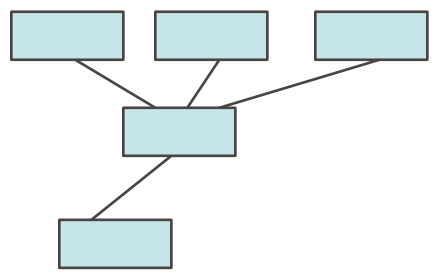


Formulaires

ex. *MS Excel*

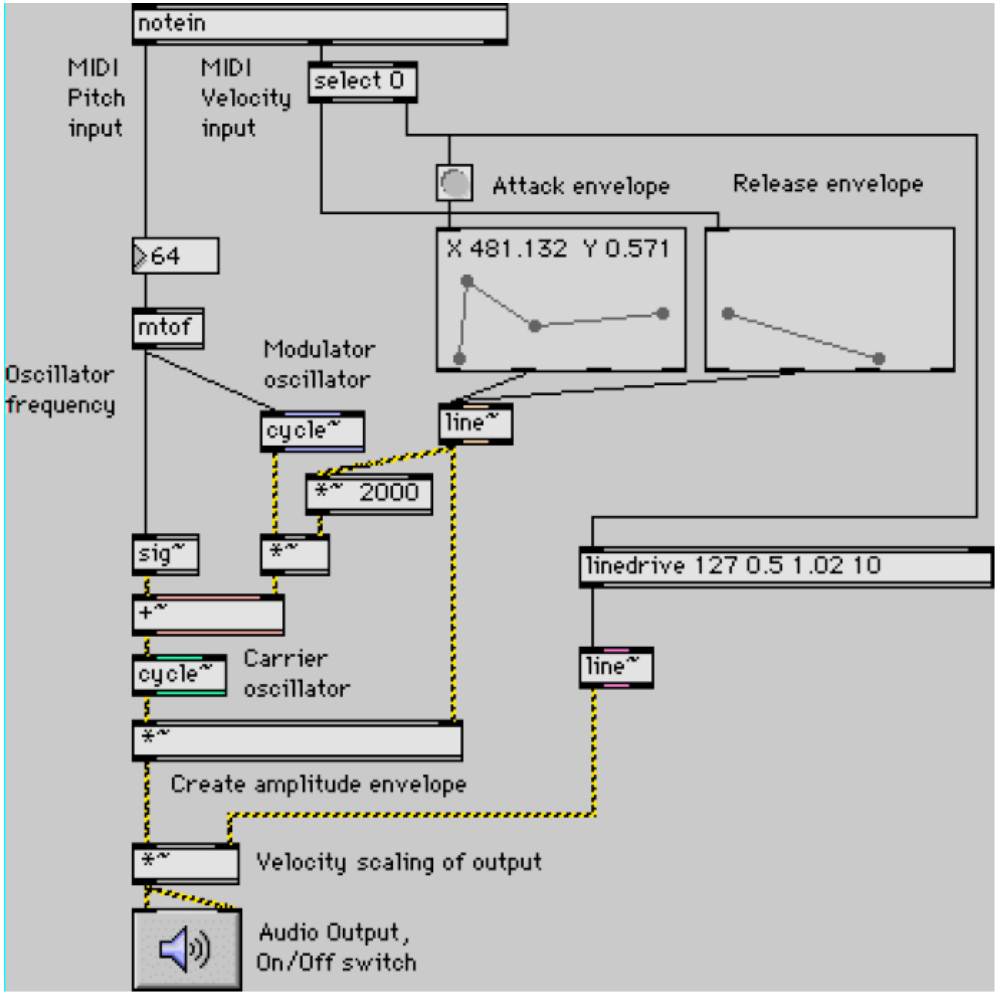
Diagrames

Modules de traitement (*boites*) liés par des *connections*
=> Flux de données entre les boîtes



Max

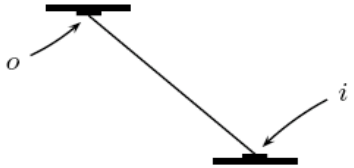
M. Puckette, 1991



PATCH
représentation d'un programme
(data flow graph)

BOITE

CONNECTION

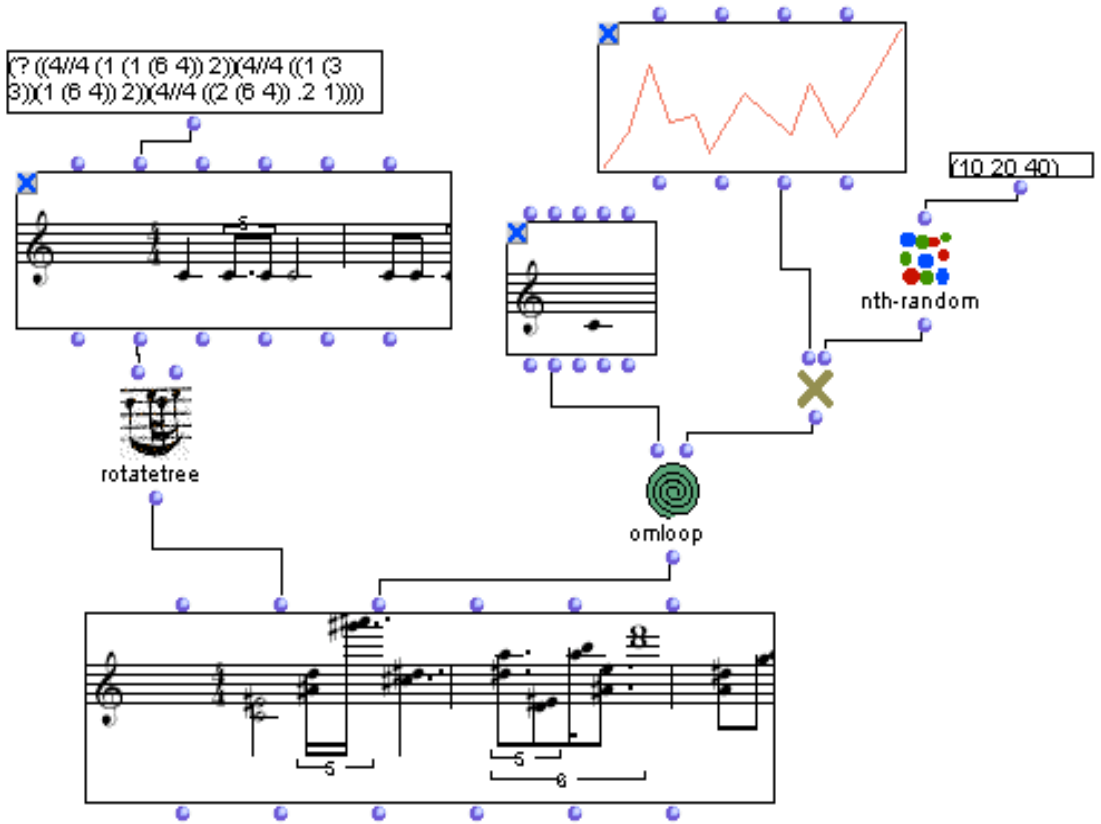


Système réactif

Flot continu de données en entrée / sortie

Système "temps-réel"

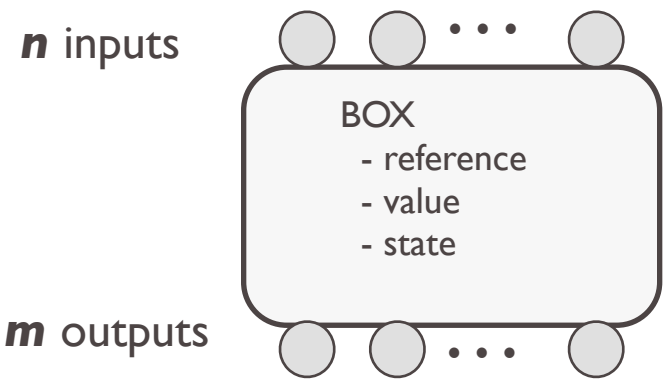
OpenMusic (OM): un environnement de CAO / langage de programmation visuel



C. Agon, G. Assayag, J. Bresson
IRCAM
1997 - 2010

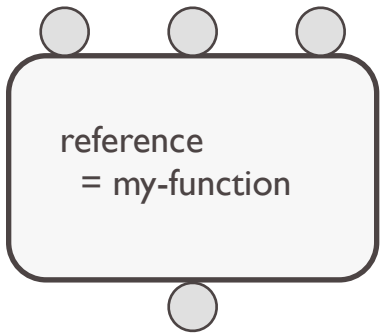
http://repmus.ircam.fr/openmusic/

Boites OM :
éléments de syntaxe et sémantique visuelles



Cas général
REFERENCE = UNE FONCTION

```
(defun my-function (arg1 arg2 arg3)
     (let rep
               [...]
          rep))
```



3 arguments

1 "return value"

Evaluation

evaluation d'une boîte =

Détermination des valeurs des *inputs*

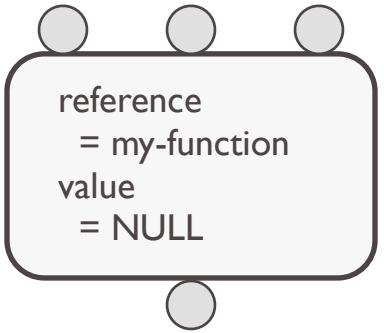
Appel (*application*) de la *reference*

```
; Eval the output indexed by 'numout' for the box 'self'
(defmethod omng-box-value ((self OMBoxCall) &optional (numout 0))
  (let ((args (mapcar #'omng-box-value (inputs self))))
    (nth numout (multiple-value-list (apply (reference self) args)))
  ))
```

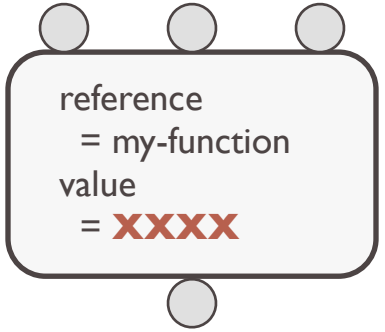
VALUE = (apply 'myfunction '(a b c))

INPUT = [value, connection]

[a, NULL] [b, NULL] [c, NULL]

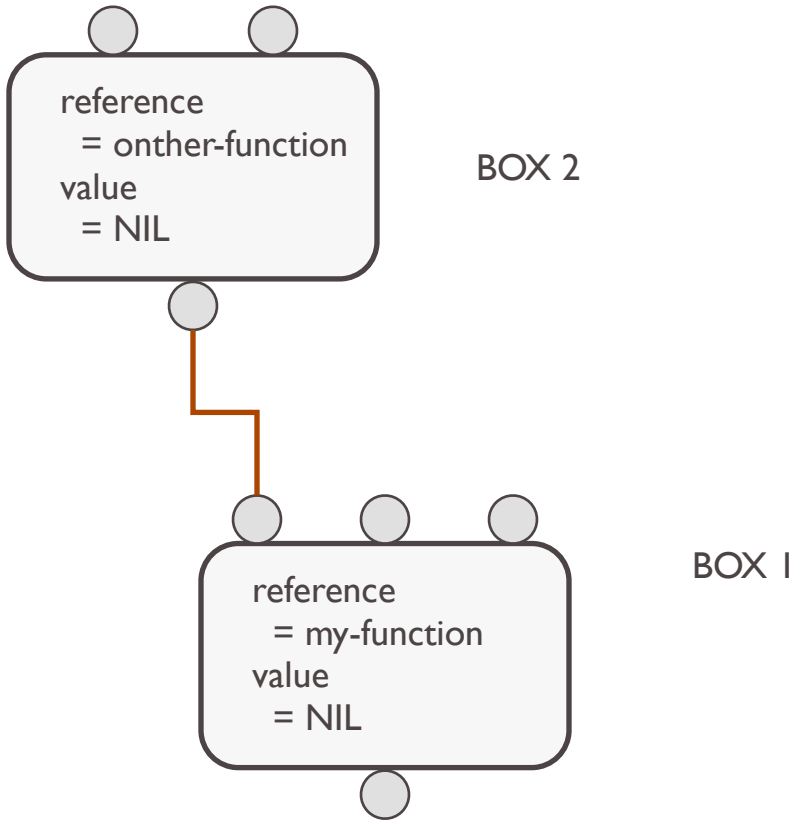


EVAL



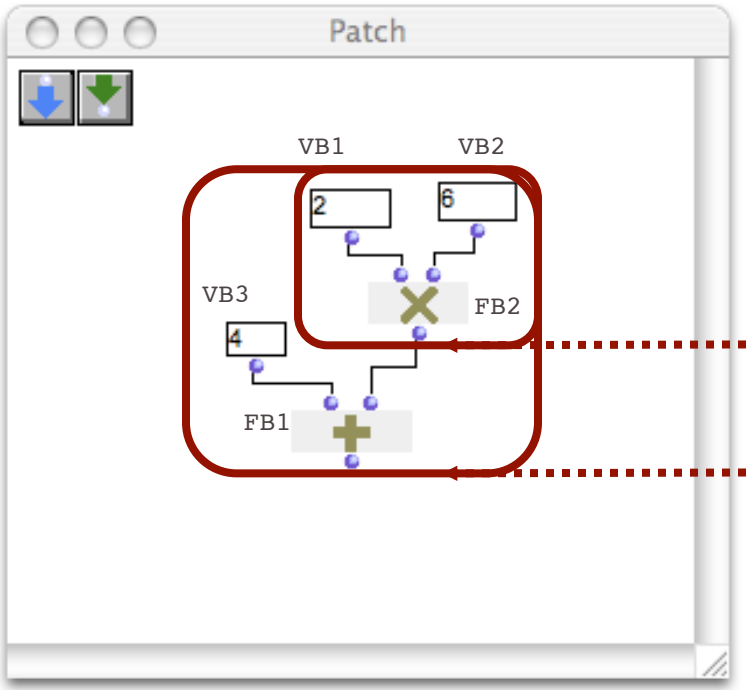
Inputs - Connections

INPUT = '(value connection)
CONNECTION = NIL | '(Box numOut)




```
; Returns the value corresponding to a box input.
(defmethod omng-box-value ((self Input) &optional (numout 0))
  (declare (ignore numout))
  (if (connection self)
     (omng-box-value (car (connection self)) (cadr (connection self)))
     (value self)))
```

Modèle d'execution "*demand-driven*"



VB = "value" box
FB = "function" box

"value" boxes :

```
(omng-box-value VB1)
⇒ 2

(omng-box-value VB2)
⇒ 6
```

```
(omng-box-value FB2)
⇒ (apply 'x (list (omng-box-value VB1) (omg-box-value VB2))
⇒ (apply 'x 2 6)

(omng-box-value FB1)
⇒ (apply '+ (list 5 (omng-box-value B2)))
⇒ (apply '+ (list 5 (apply 'x 2 6)))
```

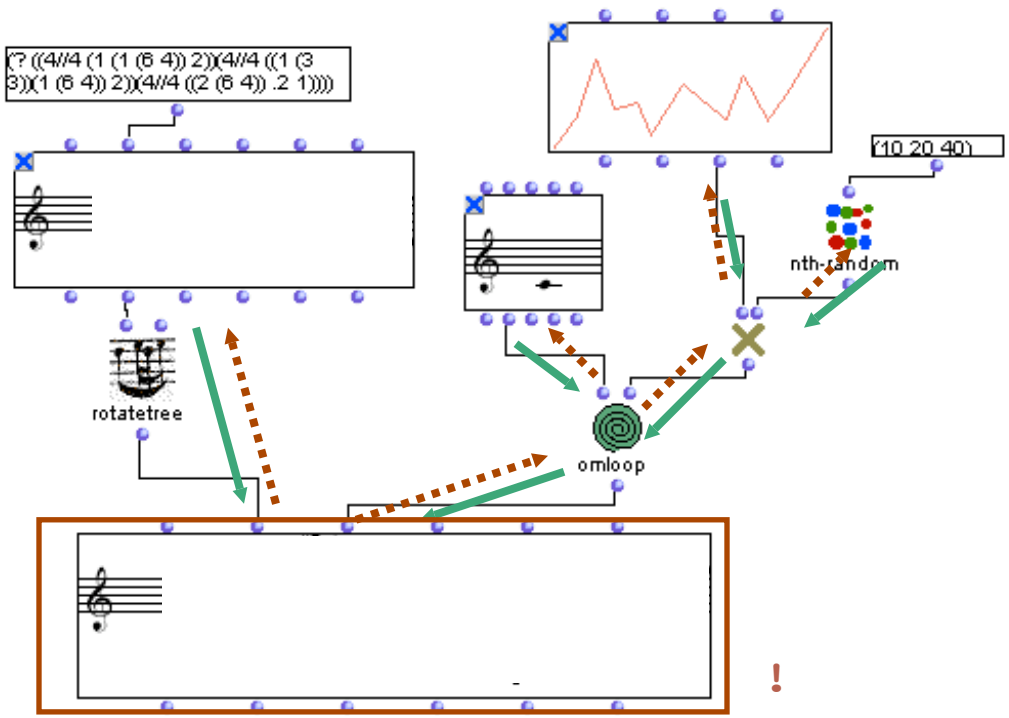
(+ 4 (x 2 6))

=> 48

data flow / control flow

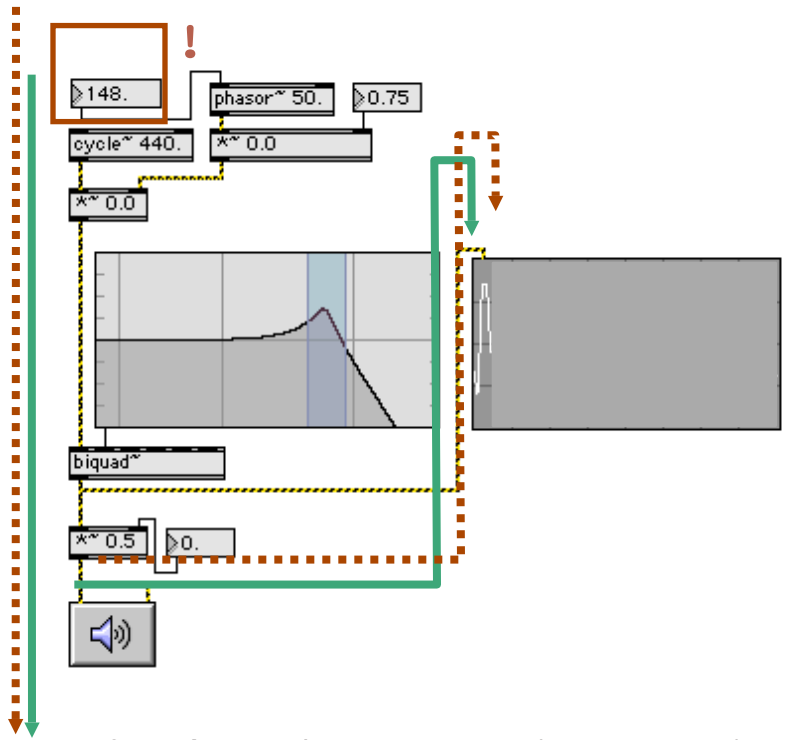
control
data flow

demand-driven
ex. OM, VPL [Lau-Kee *et al.*, 1991]



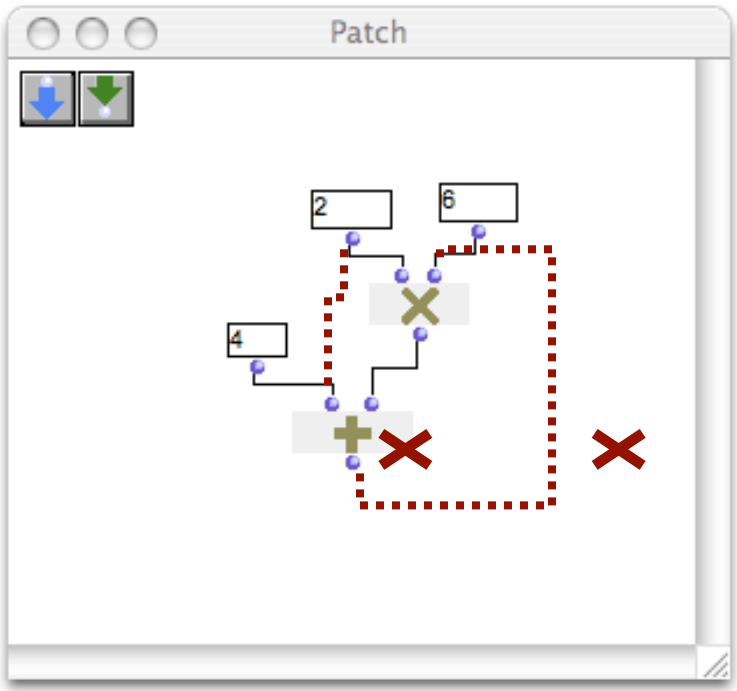
- modèle fonctionnel
- calcul "hors-temps" sur des entrées / sorties statiques

data-driven
ex. Prograph, LabView, Khoros, Max/MSP, Pd, etc.
etc.



- flux d'entrée/sortie continu (signal, events)
- système réactif

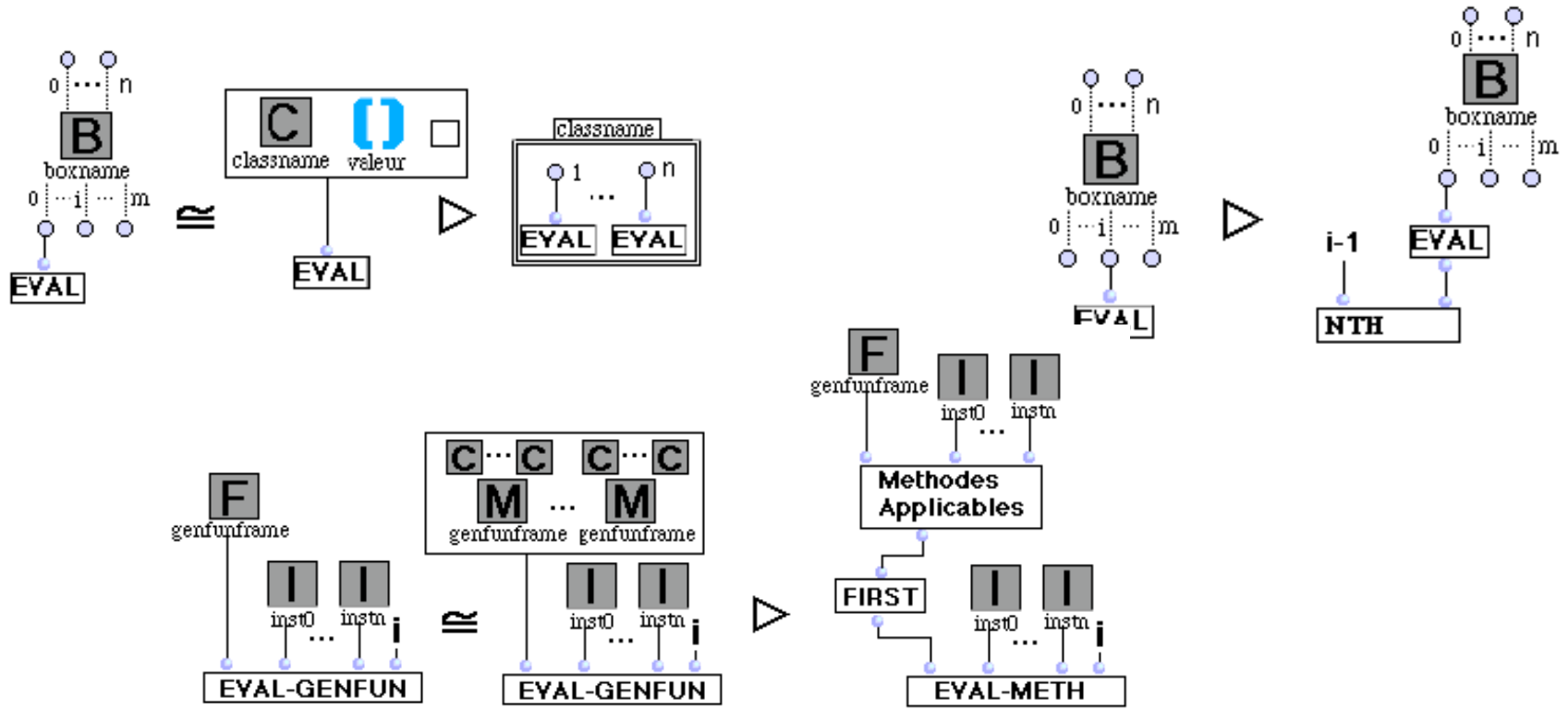
Correction syntaxique




```
(+ 4 (x 2 6))
```

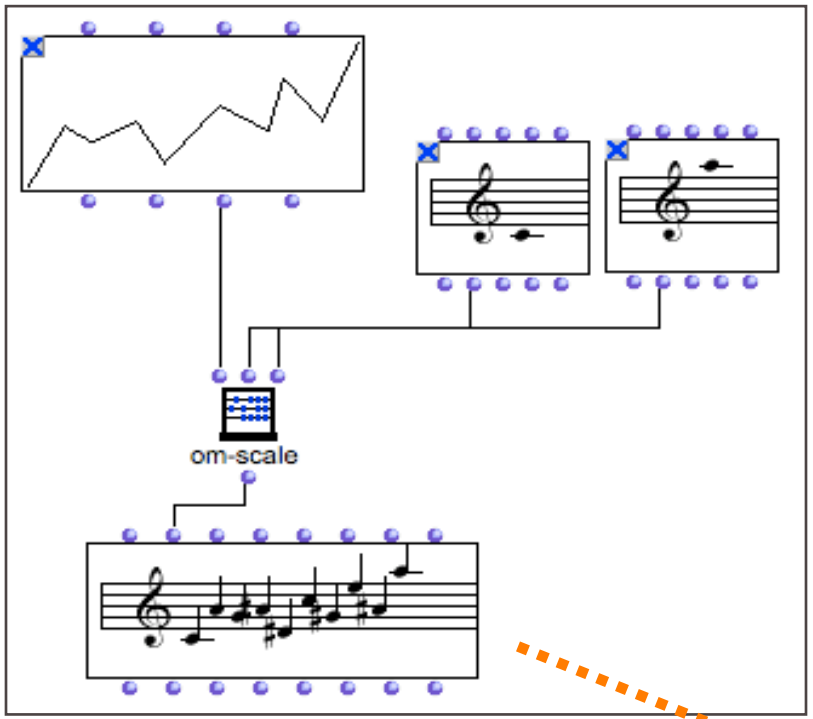
OM: Sémantique visuelle du langage

C. Agon: *OpenMusic, Un Langage de Programmation Visuelle pour la Composition Musicale*, Thèse Paris 6, 1998.



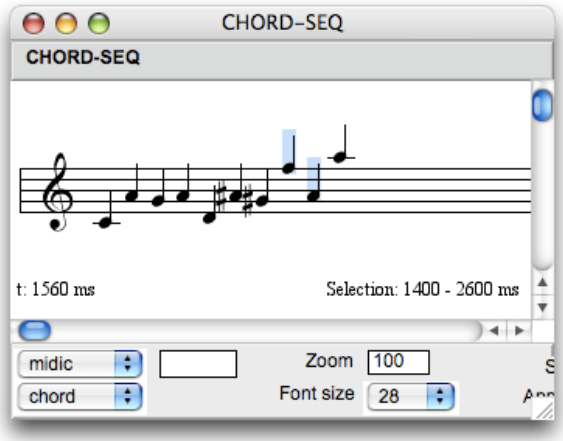
Langage réflexif bati sur la meta-programmation
(i.e. capable se représenter lui-même)

Factories / Editeurs



Edition de la *valeur* :
contrôle "manuel" et intervention
dans le calcul

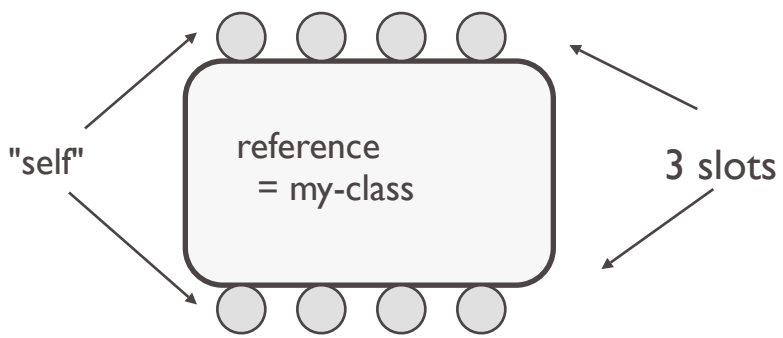
[autre aspect important du point de vue de la CAO : dualité objet / processus, etc.]

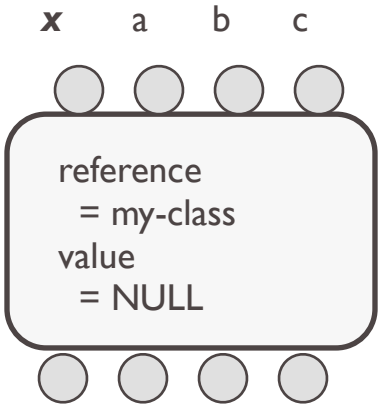


Boîtes "factory"

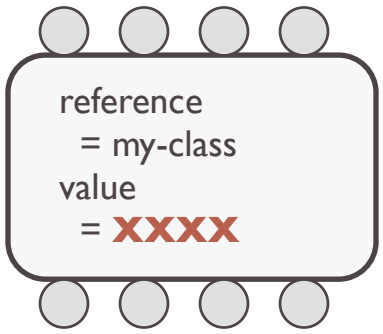
REFERENCE = UNE CLASSE

```
(defclass my-class ()
     ((slot 1 ...)
      (slot 2 ...)
      (slot 3 ...)))
```



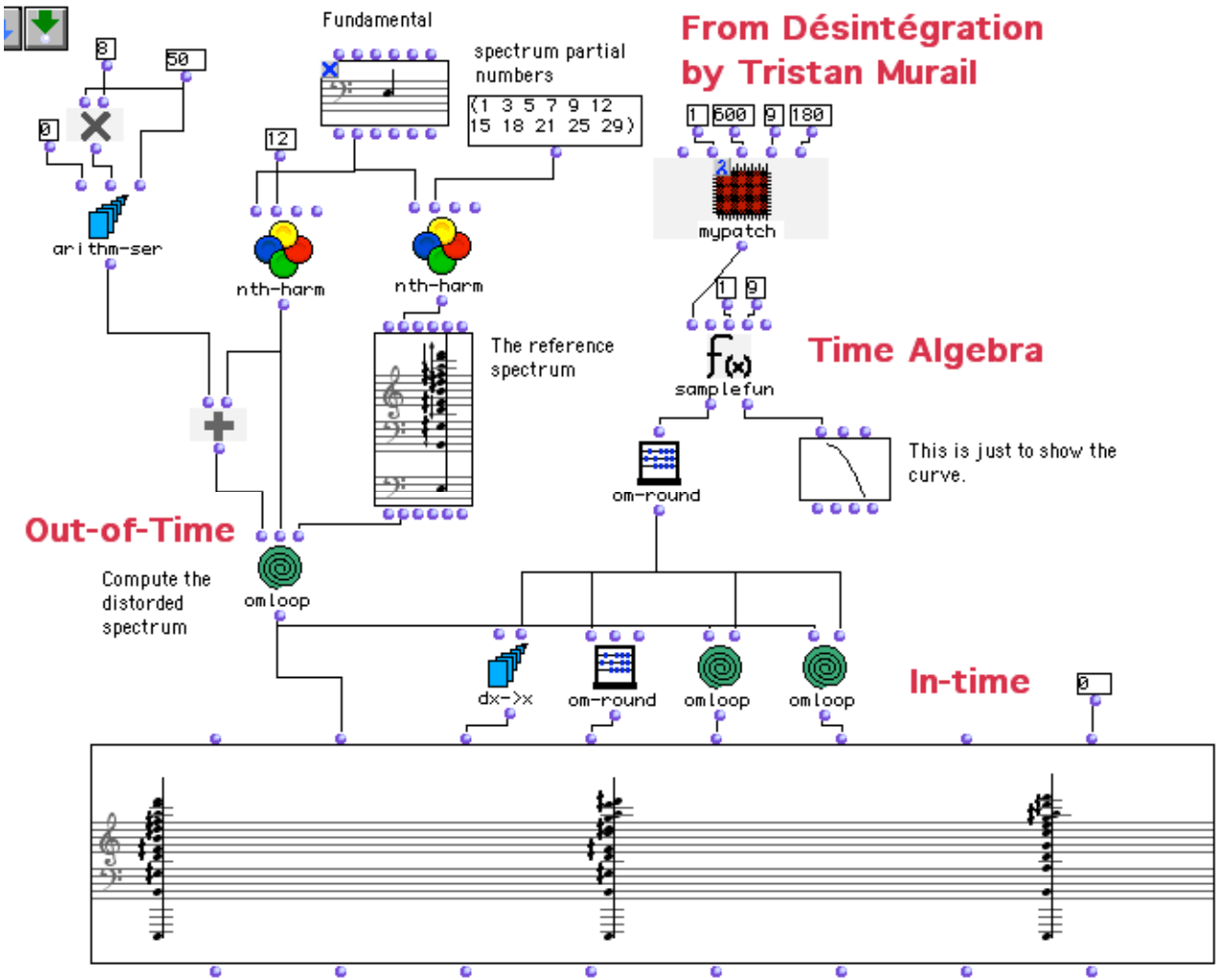


EVAL




```
VALUE = (make-instance 'my-class
                       :slot1 a :slot2 b ...)
```

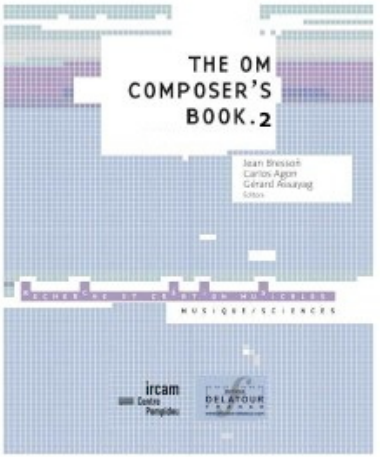
OpenMusic : Spectral music revisited



Desintegration (T. Murail)

[0'18"]

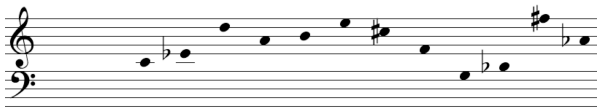
Retour sur quelques notions et problèmes informatiques avec OpenMusic

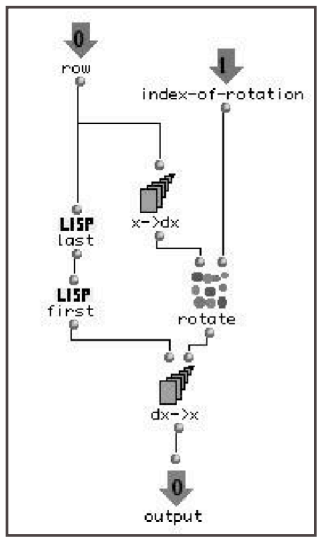


The OM Composer's Book
J. Bresson, C. Agon, G. Assayag (Eds.)

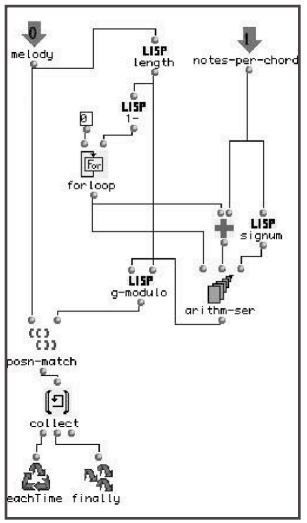
traitement de hauteurs sous forme de listes

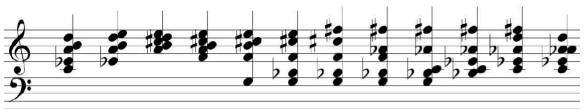
J. Kretz
Second Horizon
(2002)

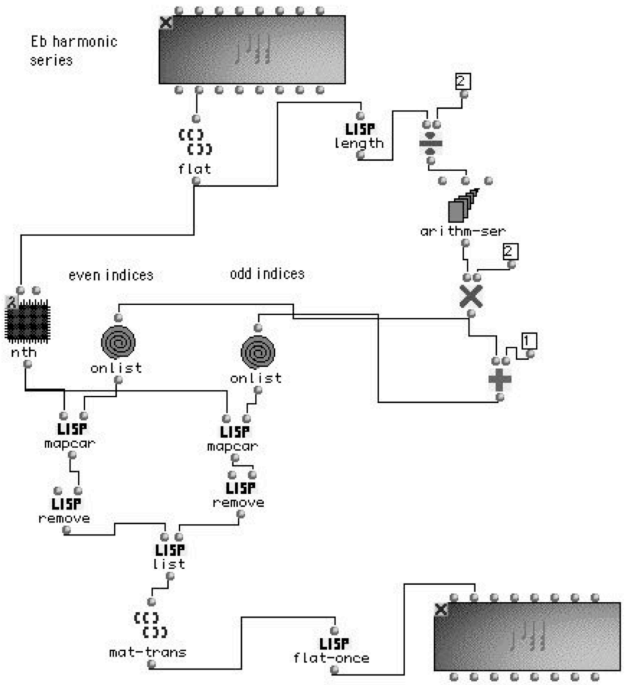




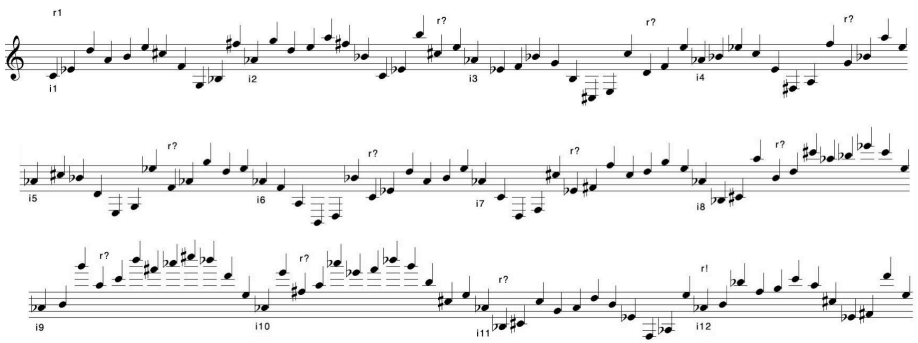
Chords generation

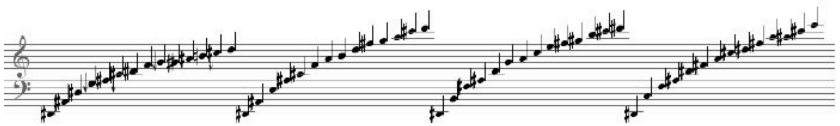






Permutations of the series interval

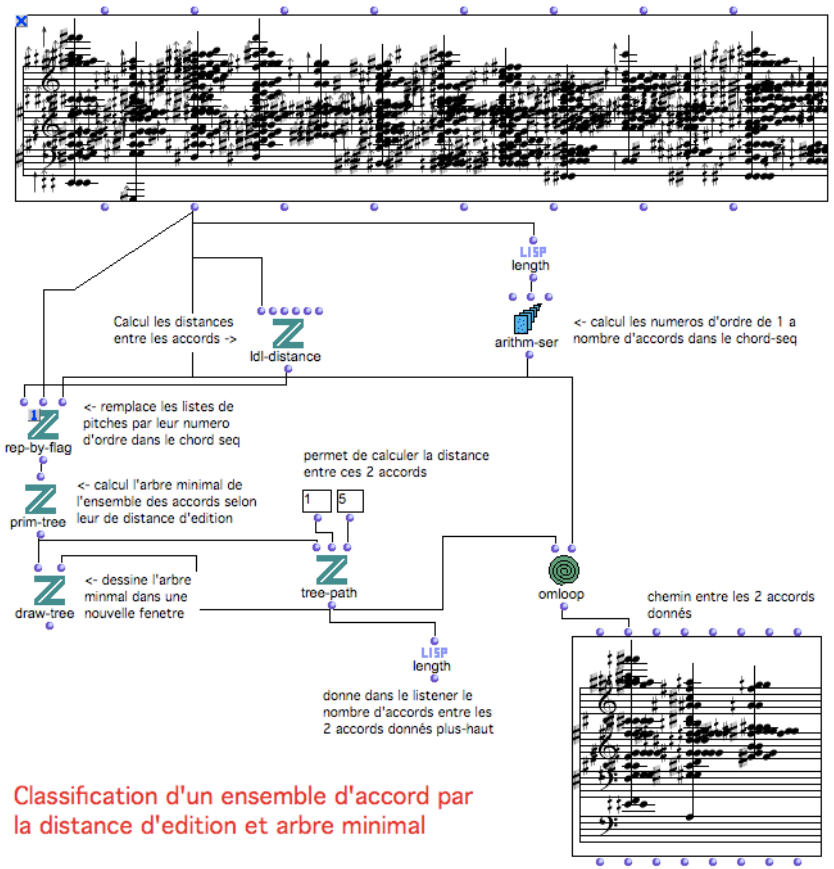




K. Nez
Sculpted Implosipons
(1999)

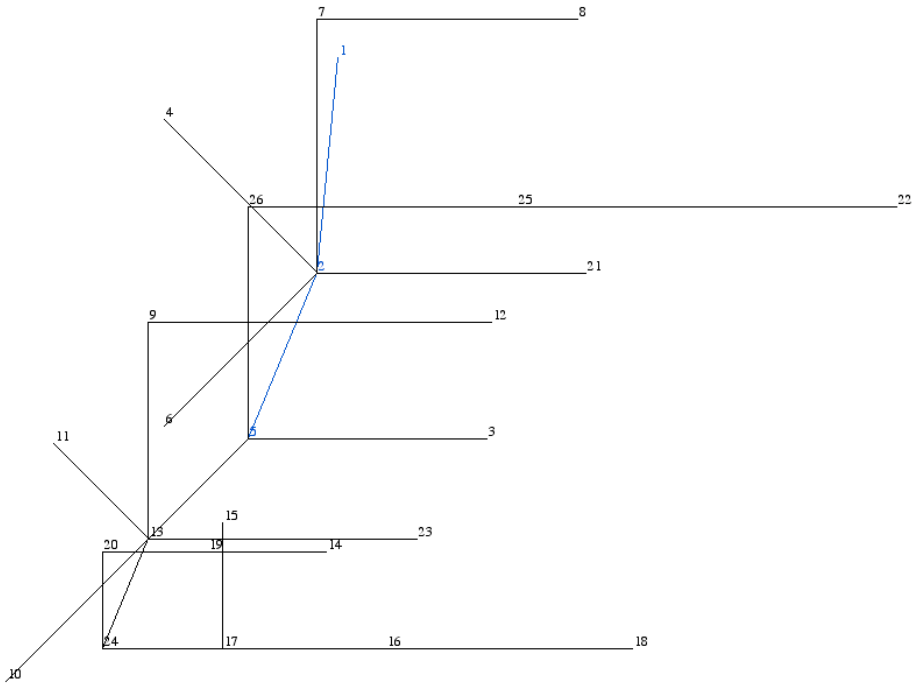
Problèmes de classification
Recherche combinatoire

Ph. Leroux
VOI(REX)
(2002)



Classification d'un ensemble d'accord par
la distance d'edition et arbre minimal

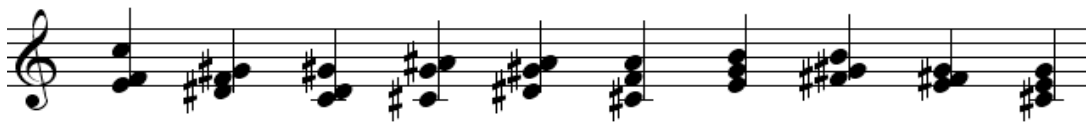
Minimum Weight Spanning Tree
--> optimal sequence of chords following a given classification criteria



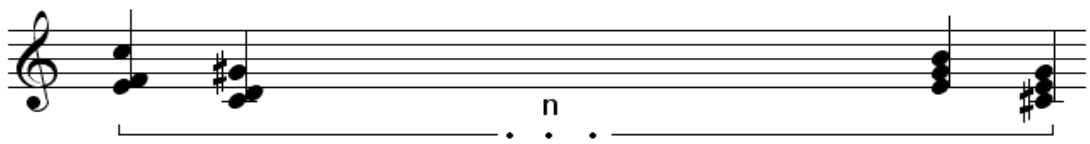
Abstraction

Créer un programme capable de générer un nombre *N* d'accords de *M* notes entre *A* et *B*

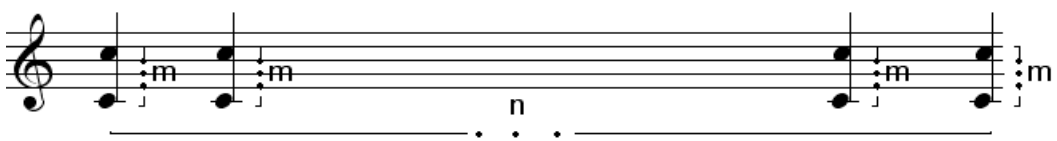
```
(repeat 10 (repeat 3 (random DO3 DO4)))
```



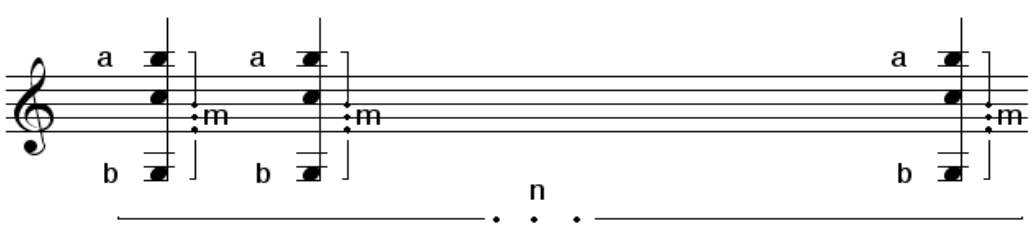
F_1(n)=(repeat n (repeat 3 (random DO3 DO4)))



F_2(n m)=(repeat n (repeat m (random DO3 DO4)))



F_3(n m a b) =(repeat n (repeat m (random a b)))

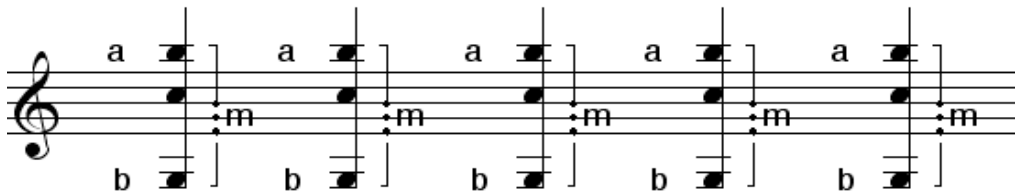


Application

Créer un programme capable de générer un nombre *N* d'accords de *M* notes entre *A* et *B*

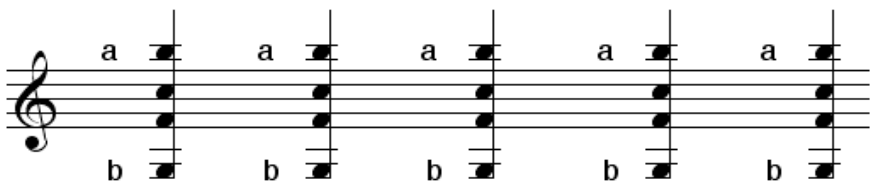
F_3(n m a b) (5)

F_4(m a b) =(repeat 5 (repeat m (random a b)))



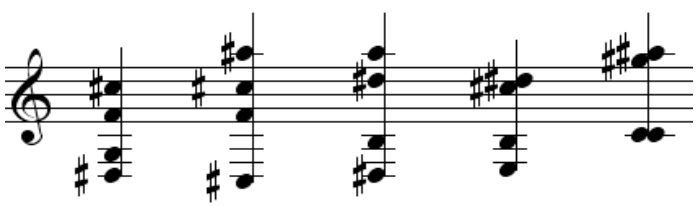
F_4(m a b) (4)

F_5(a b)=(repeat 5 (repeat 4 (random a b)))



F_5(a b)(C2)(C5)

(repeat 5 (repeat 4 (random C2 C5)))



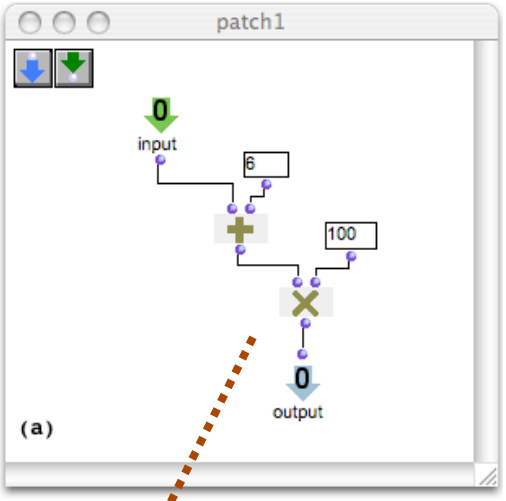
Abstraction

```
(defmethod omng-box-value ((self OMBoxPatch))
  (let ((thepatch (reference self))
        (args (mapcar #'omng-box-value (inputs self))))
    (unless (compiled? thepatch)
      (compile-patch thepatch))
    (apply (patch-function thepatch) args)
  ))
```

```
(lambda (x)
  (* (+ x 6) 100))
```

ou

```
(defun my-function (x)
  (* (+ x 6) 100))
```

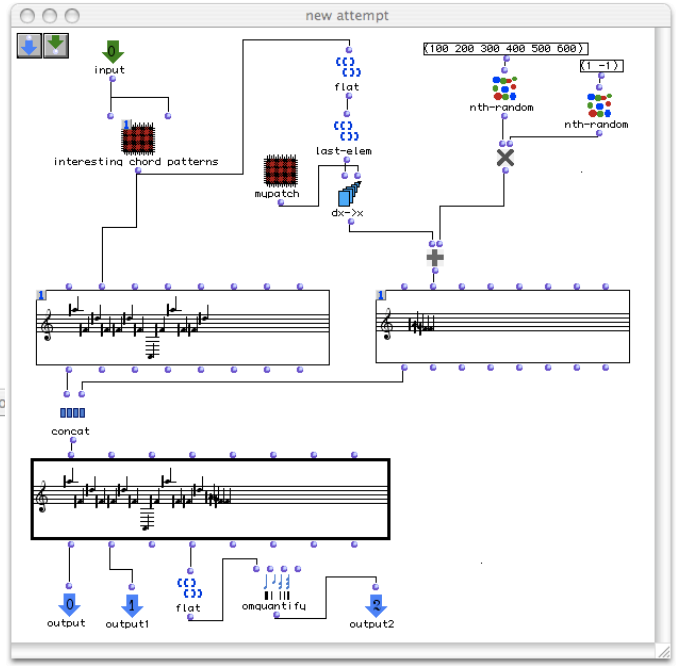


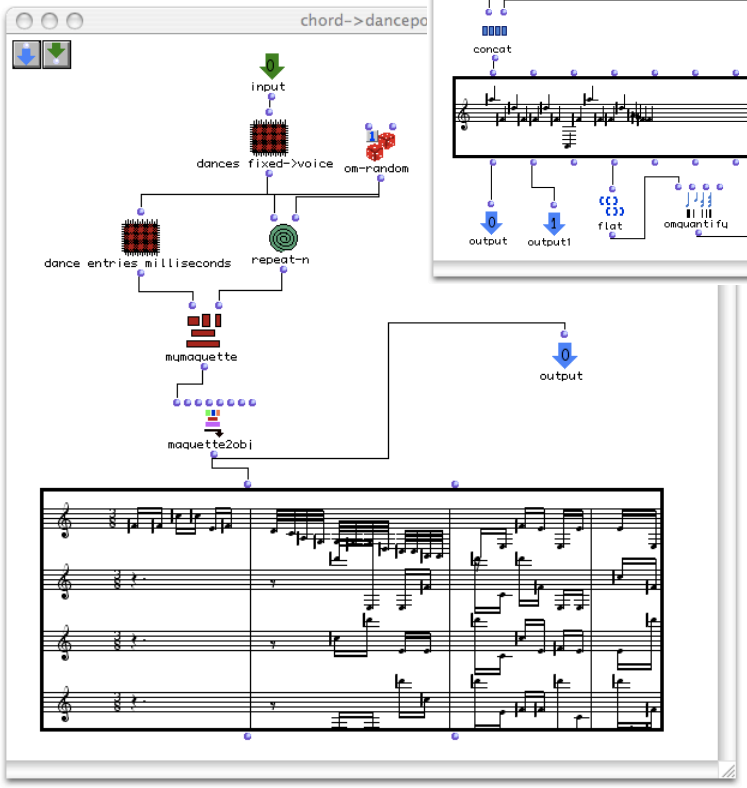

```
(/ (my-function 5) 10)
```

```
; Generates the Lisp code for a function box
(defmethod gen-code ((self OMBoxCall))
  `(,(reference self) ,@(mapcar #'gen-code (inputs self))))
```

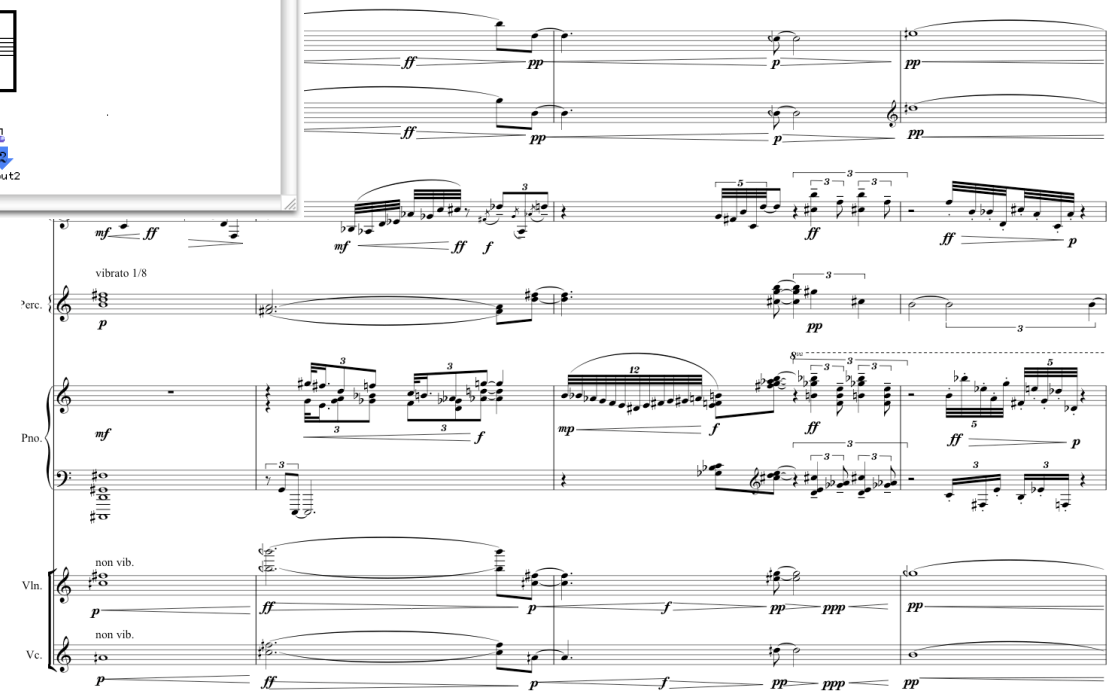
```
; Generates the Lisp function for patch 'self'
(defmethod compile-patch ((self OMPatch))
  (let* ((out-boxes (output-boxes self))
         (in-boxes (input-boxes self)))
    (eval `(defun ,(patch-function self) (,@(mapcar #'name in-boxes))
             (values ,@(mapcar #'gen-code out-boxes)))
    (setf (compiled? self) t)
  ))
```

S. Mawhinney
***Starbog* (2006)**

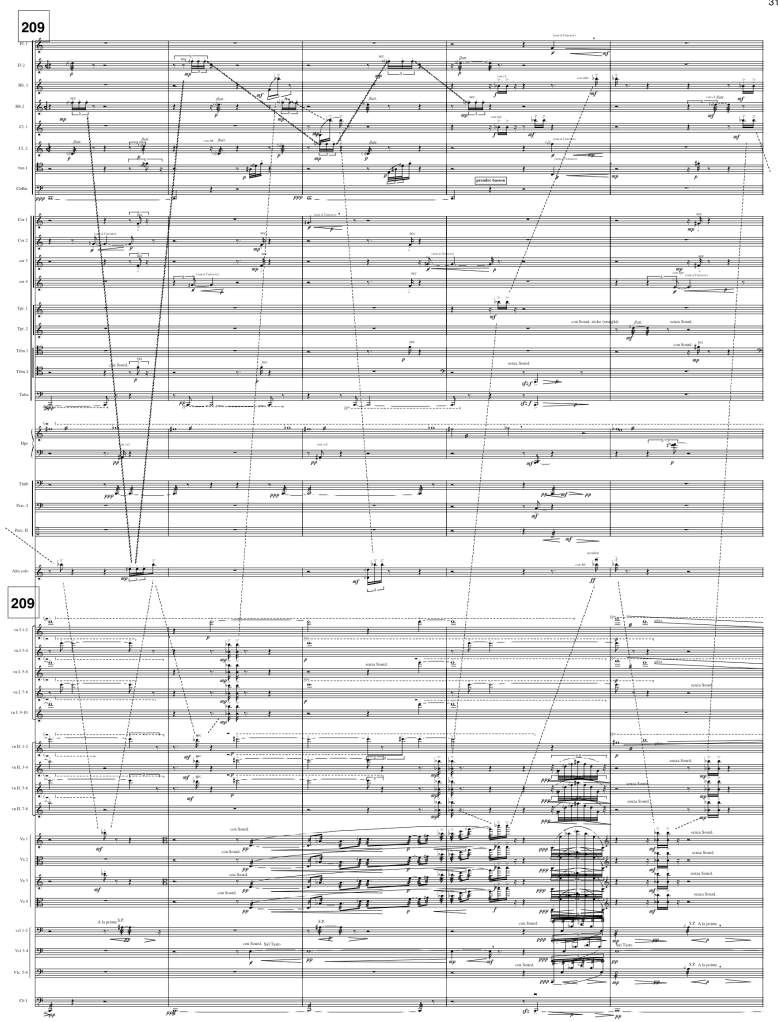


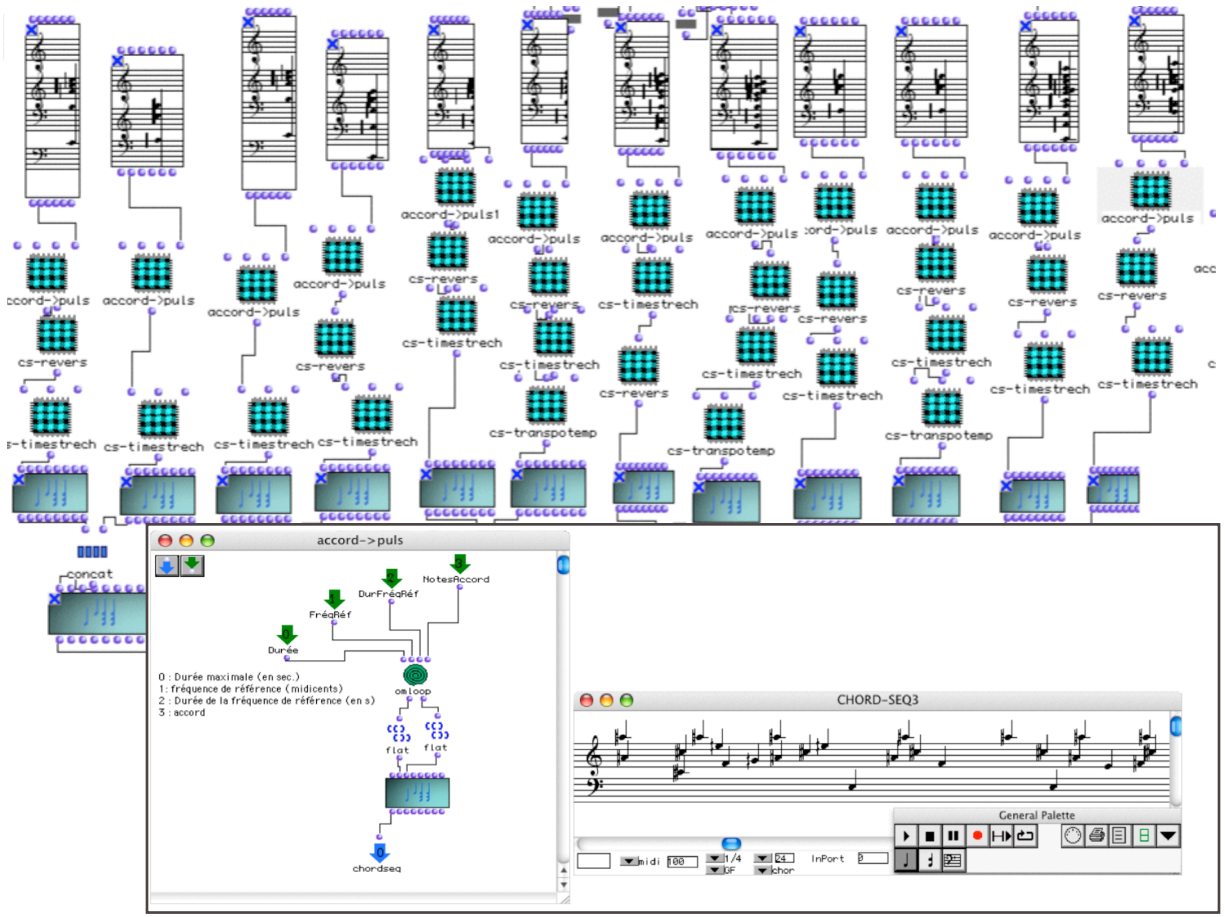


Abstractions procédurales pour composition algorithmique



F. Lévy
Hérébo-Ribotes (2002)

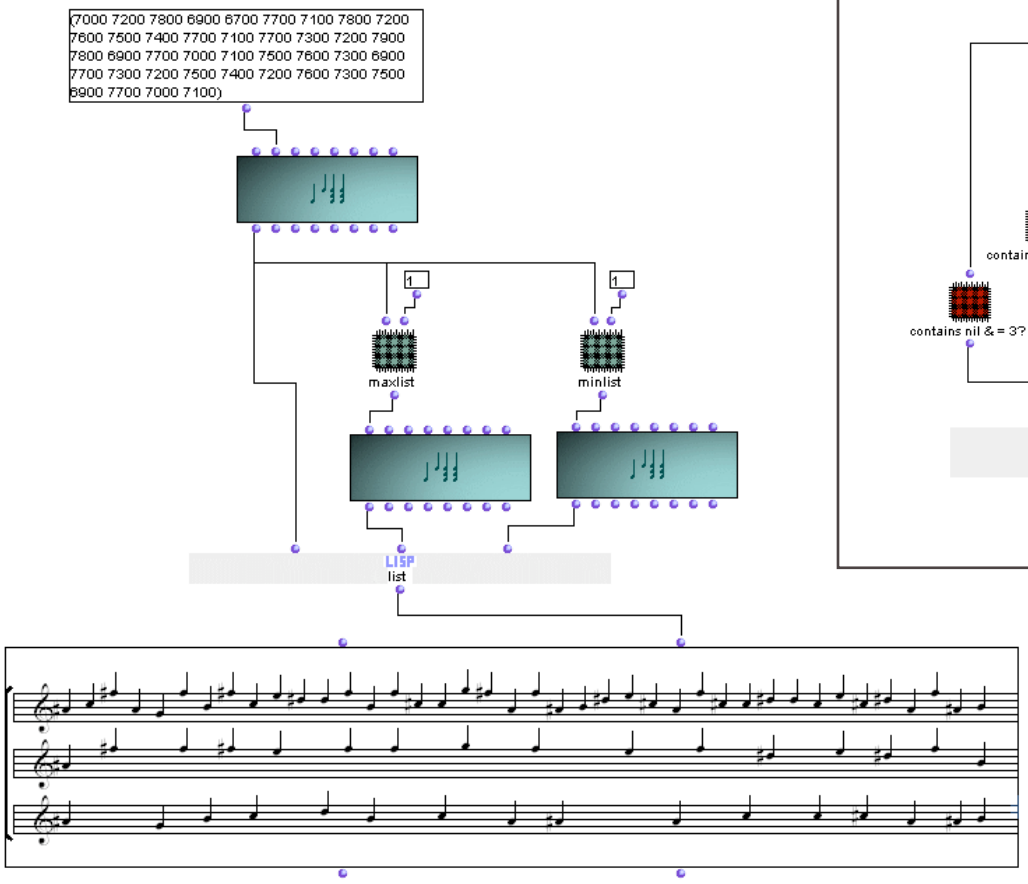


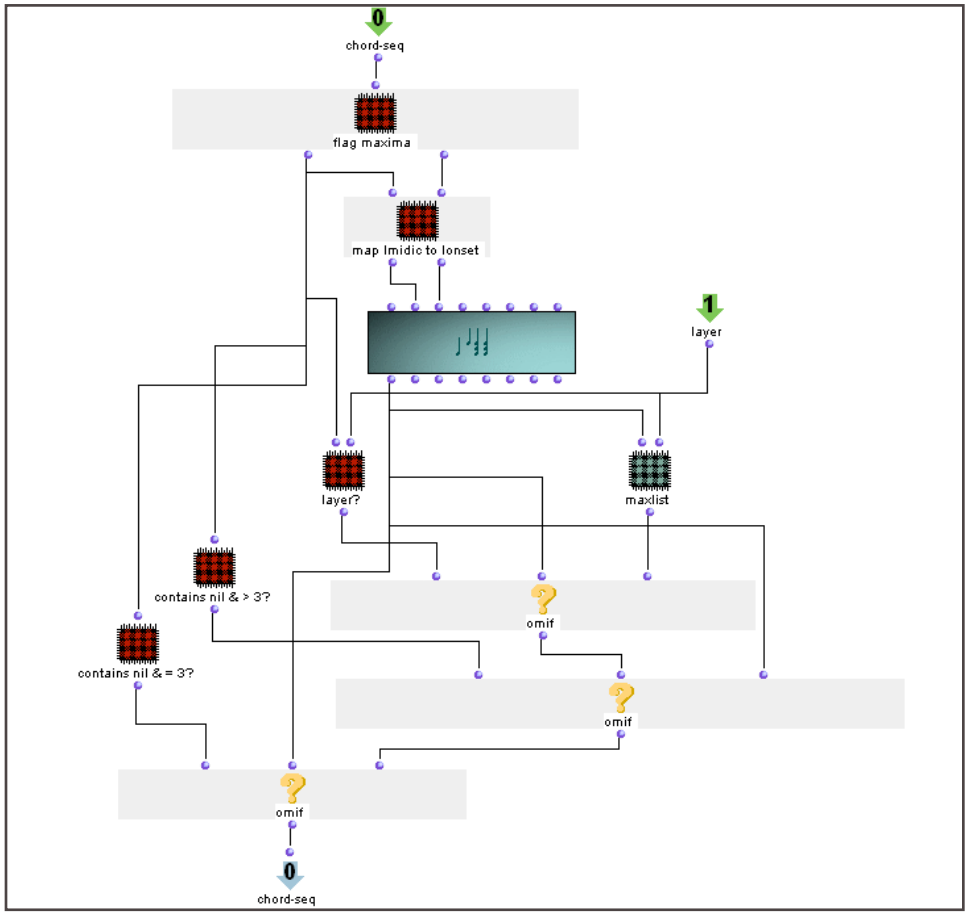


Recursivité

"fonctions faisant appel à elles-mêmes"

C. Melen
Six Metal Fugue
(2006)

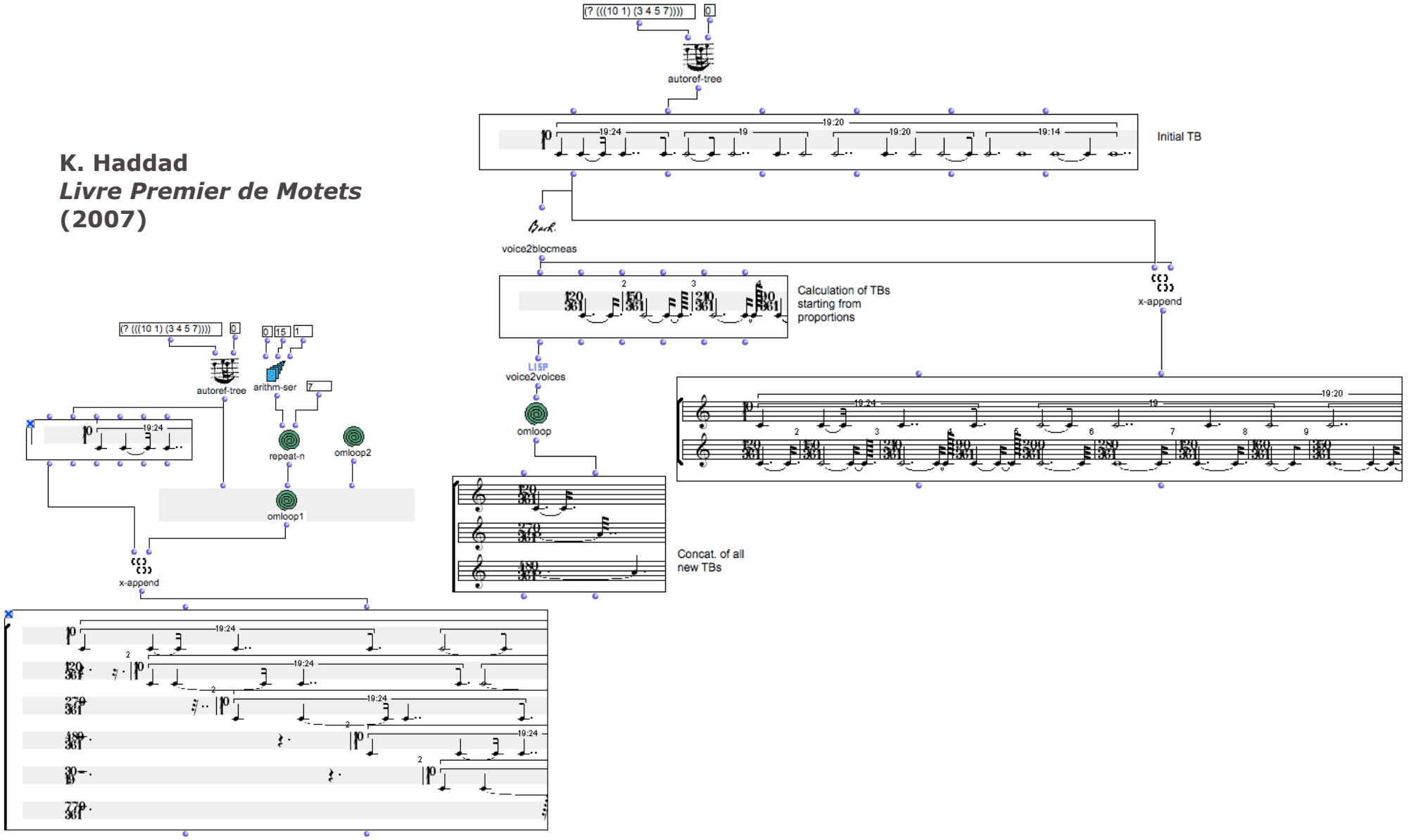




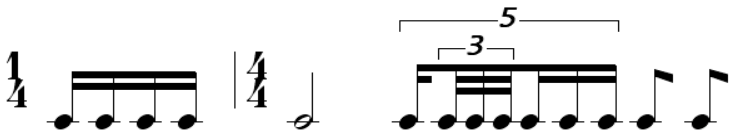
contour reduction algorithm (R. Morris)

traitement des rythmes sous forme d'arbres

K. Haddad
Livre Premier de Motets
(2007)



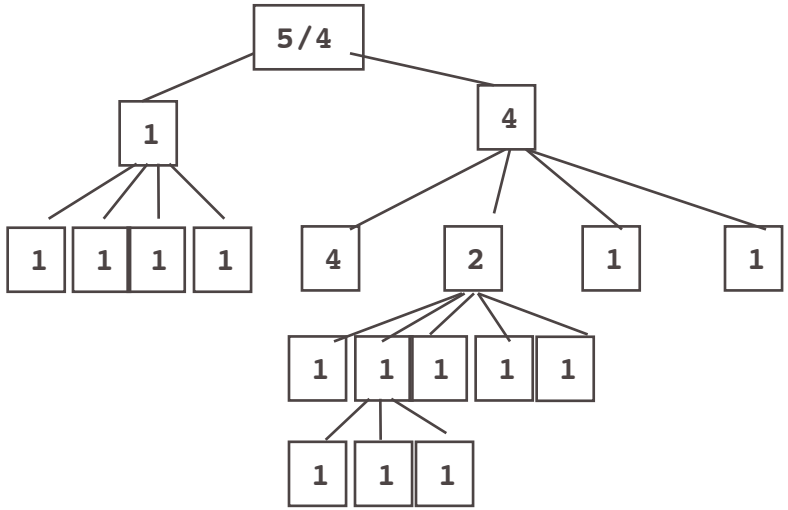
Exemple (récursivité)



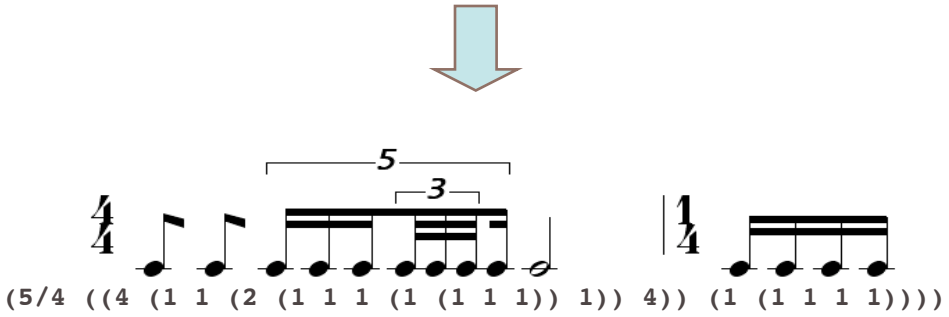

```
(5/4 ((1 (1 1 1 1)) (4 (4 (2 (1 (1 (1 1 1)) 1 1 1)) 1 1))))
```

Inversion rythmique
=> traitement récursif de l'arbre

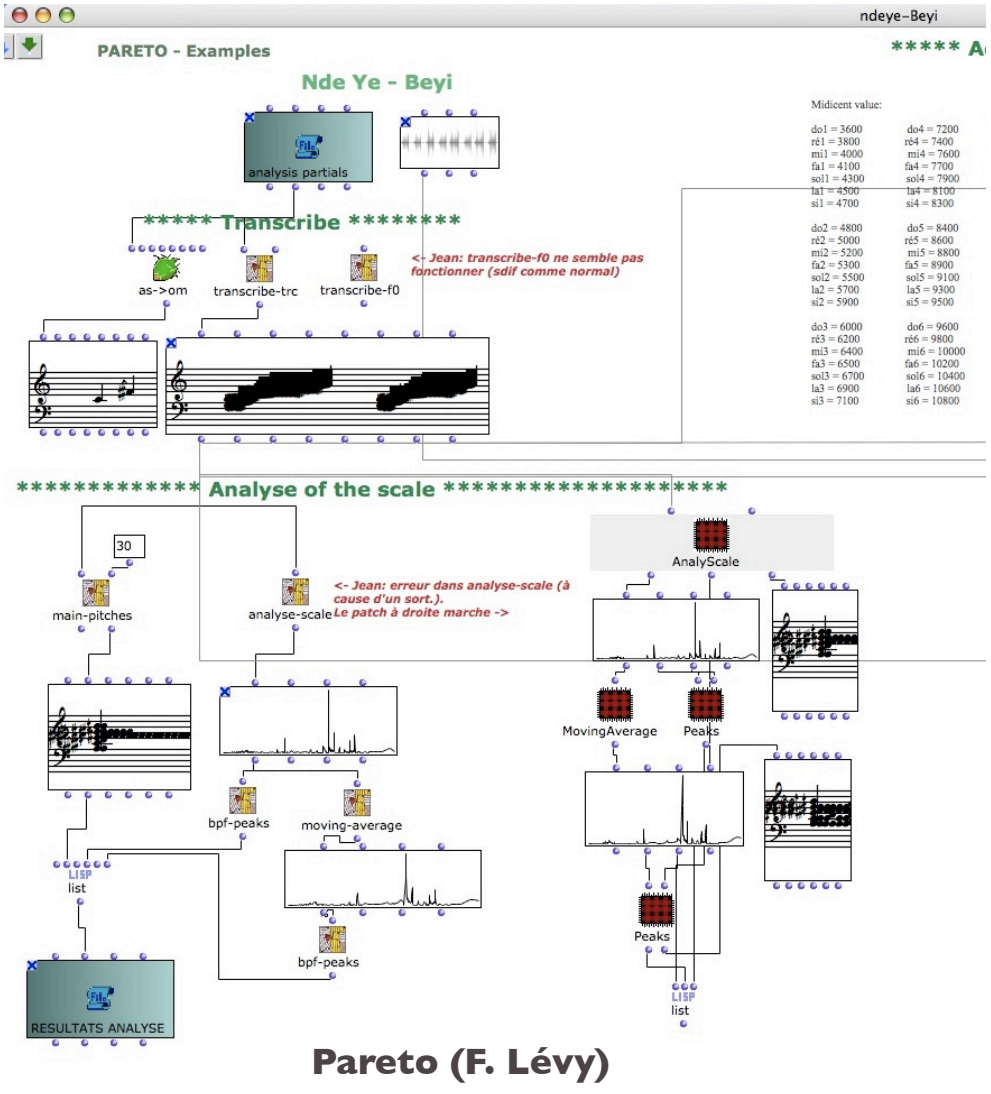
$$R(a) = (root\ , (R(a_n),\ldots,R(a_1)))$$



$$a = (root\ , (a_1 \ldots\ a_n))$$

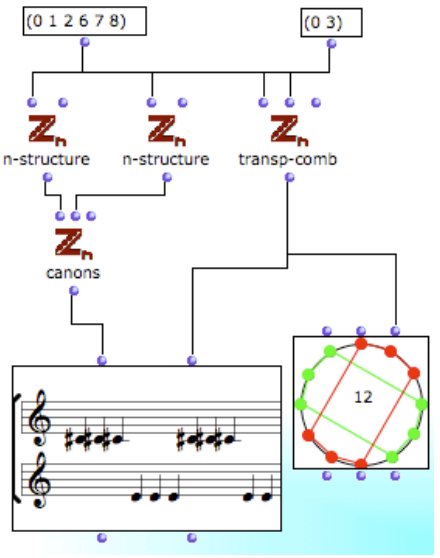


Analyse “computationelle” des structures musicales



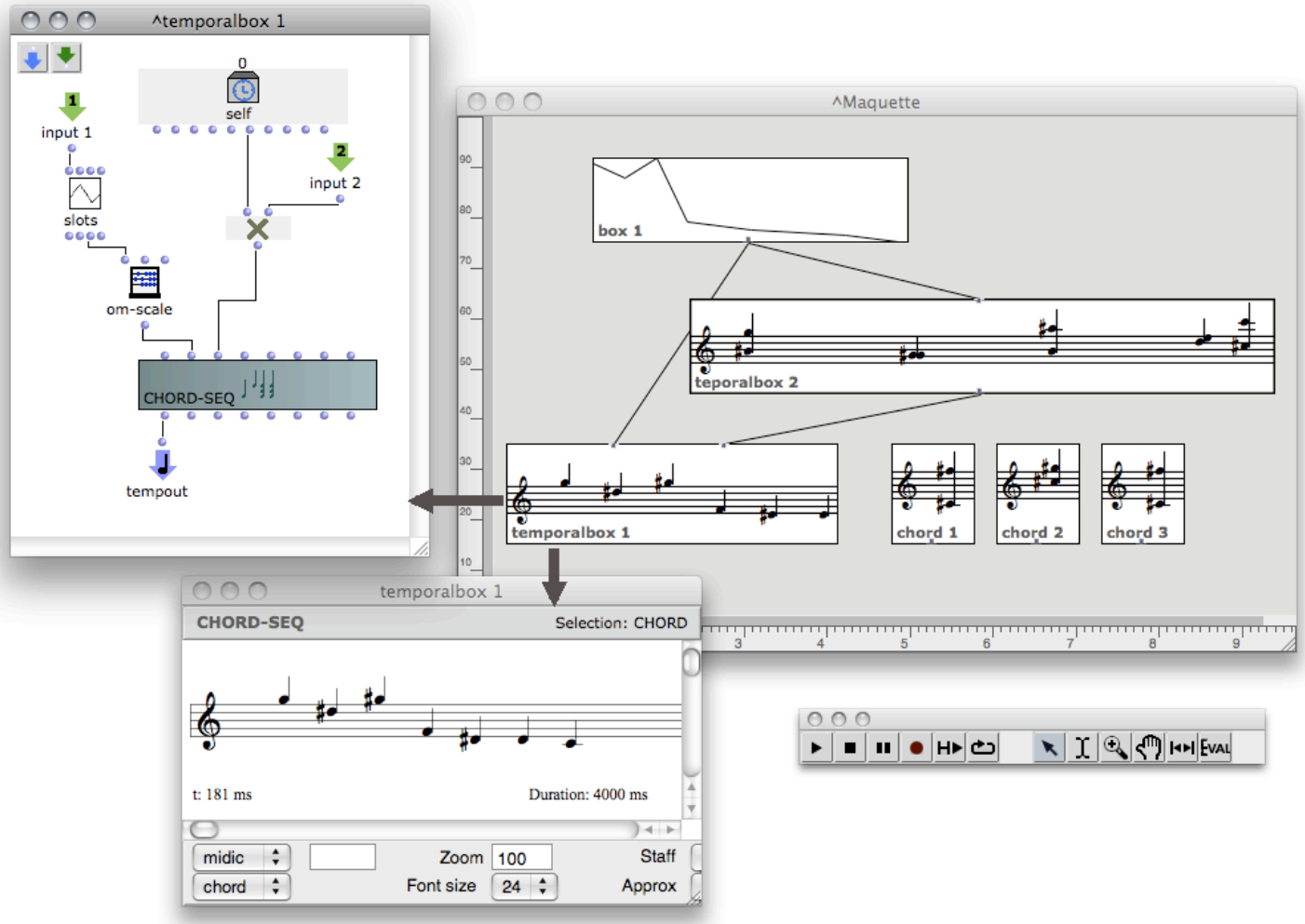
Pareto (F. Lévy)

MathTools
(M. Andreatta, C. Agon)



Représentations temporelles pour la composition
(*maquette*)

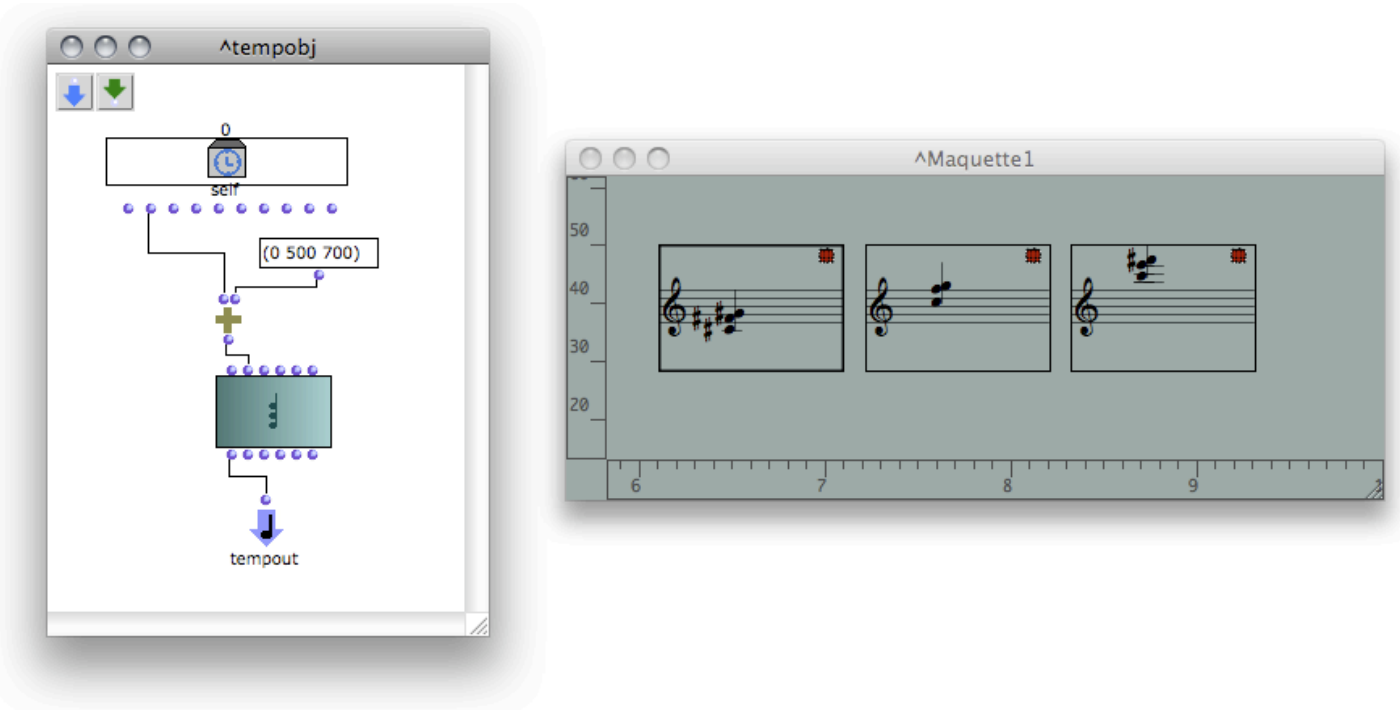
MAQUETTE: time structure/visual program



MAQUETTE

Visual Meta Object Programming
(Agon, 2003)

When the time layout drives calculus...

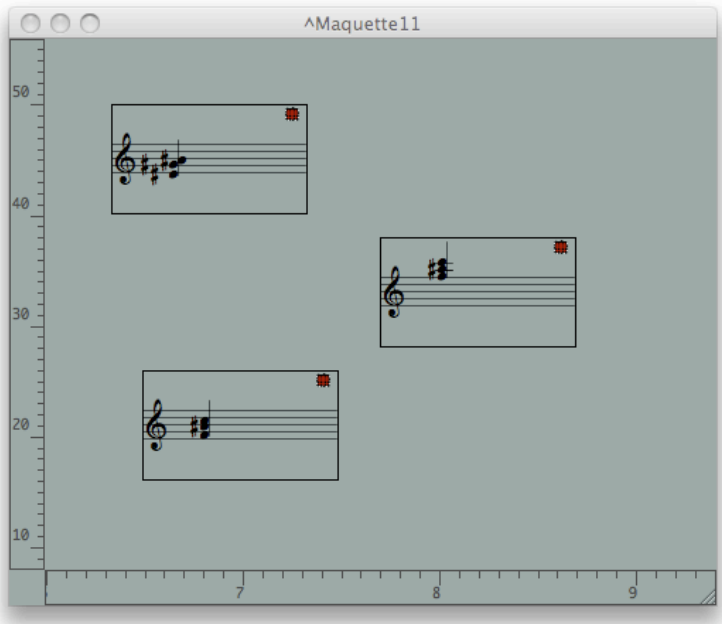


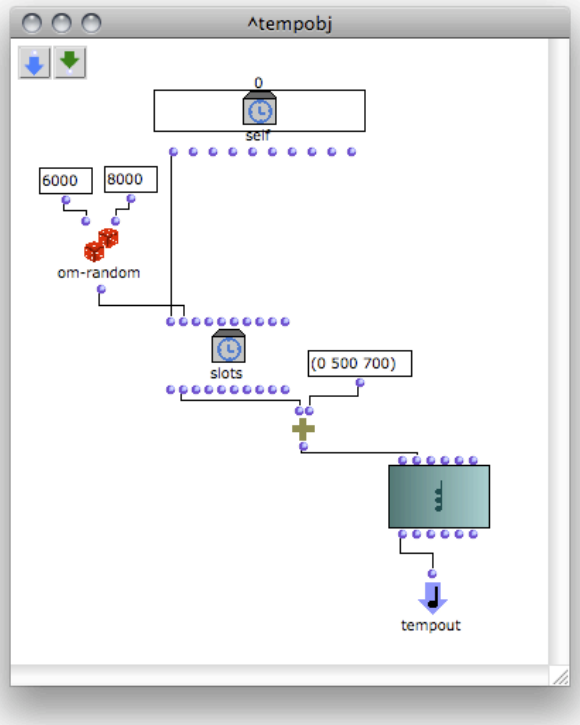
Ex. La hauteur (transposition) dépend de la position (dans le temps)

MAQUETTE

Visual Meta Object Programming
(Agon, 2003)

... and calculus determines the time layout

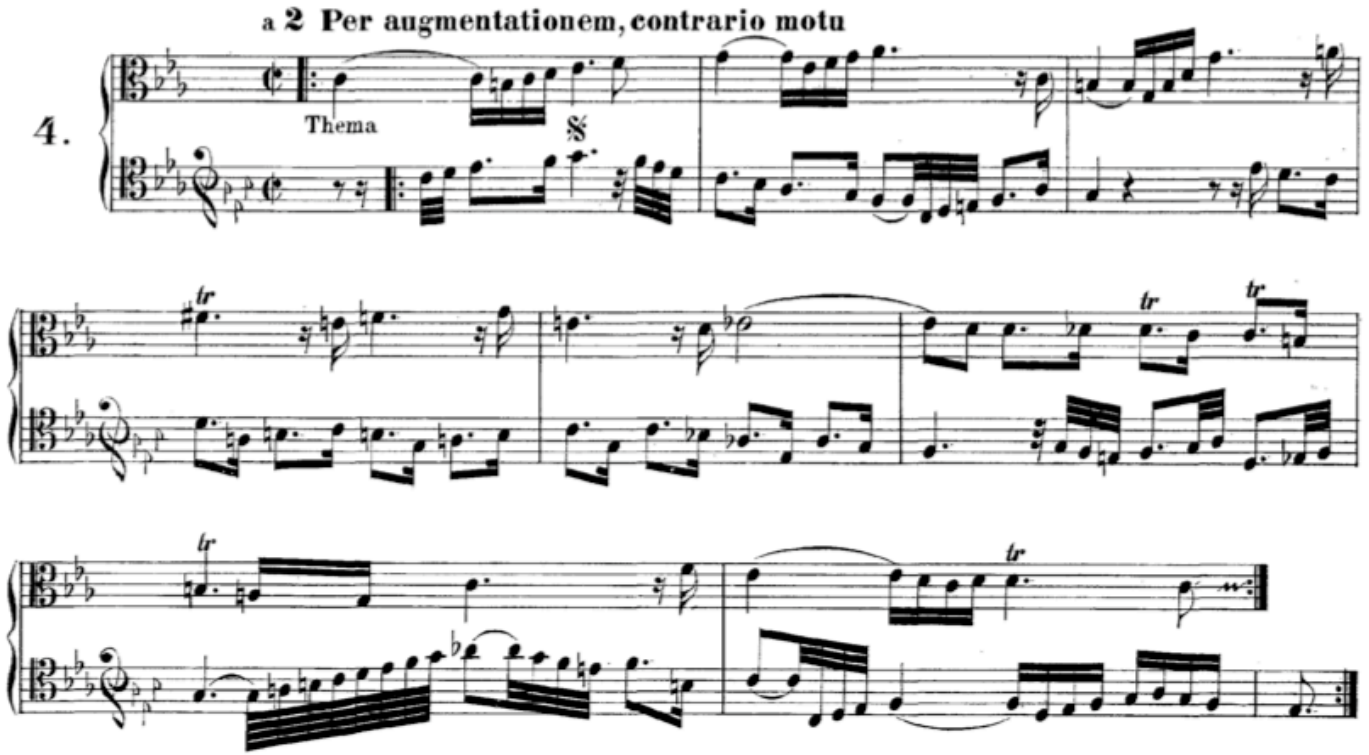


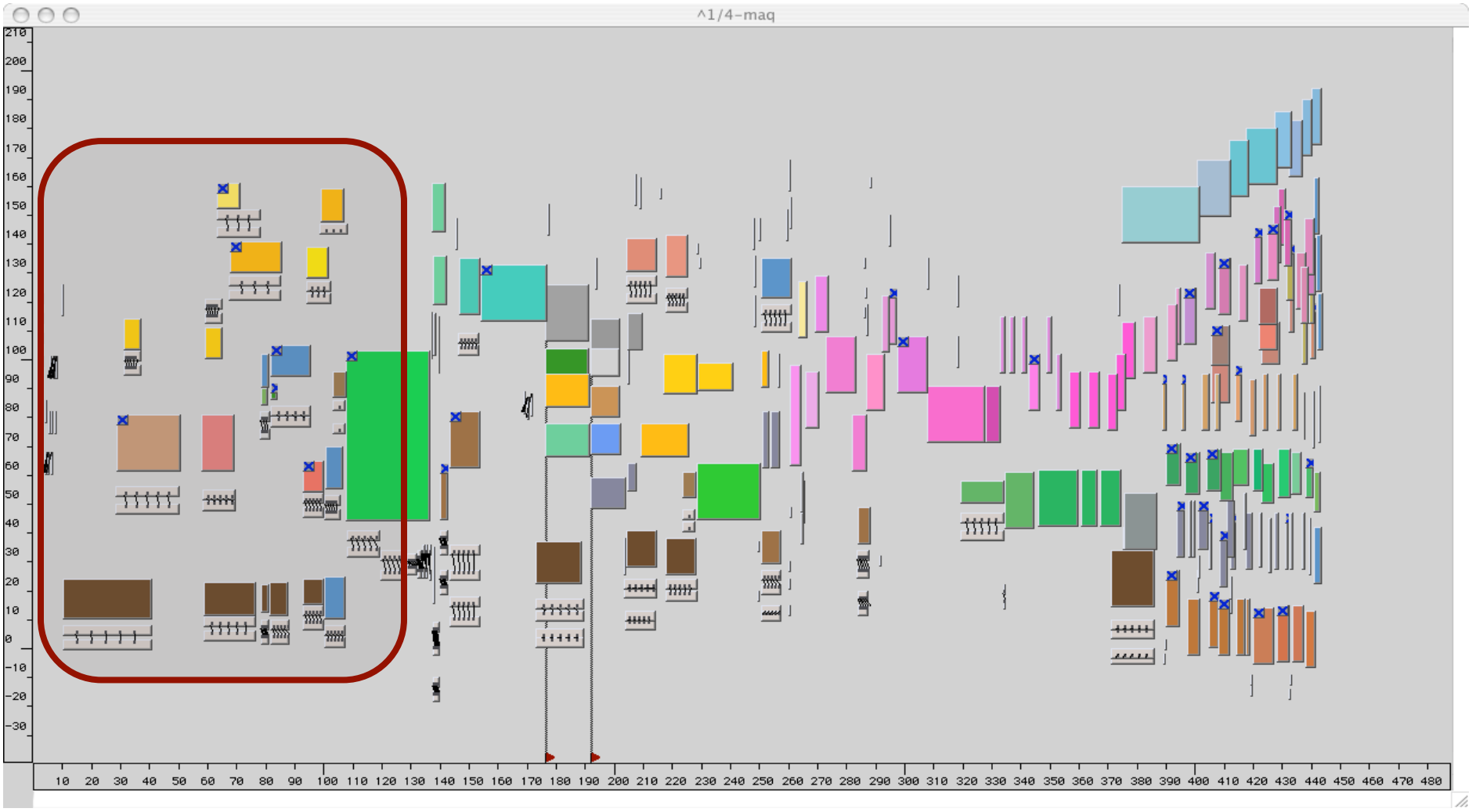


Ex. La position (dans le temps) dépend du calcul des objets

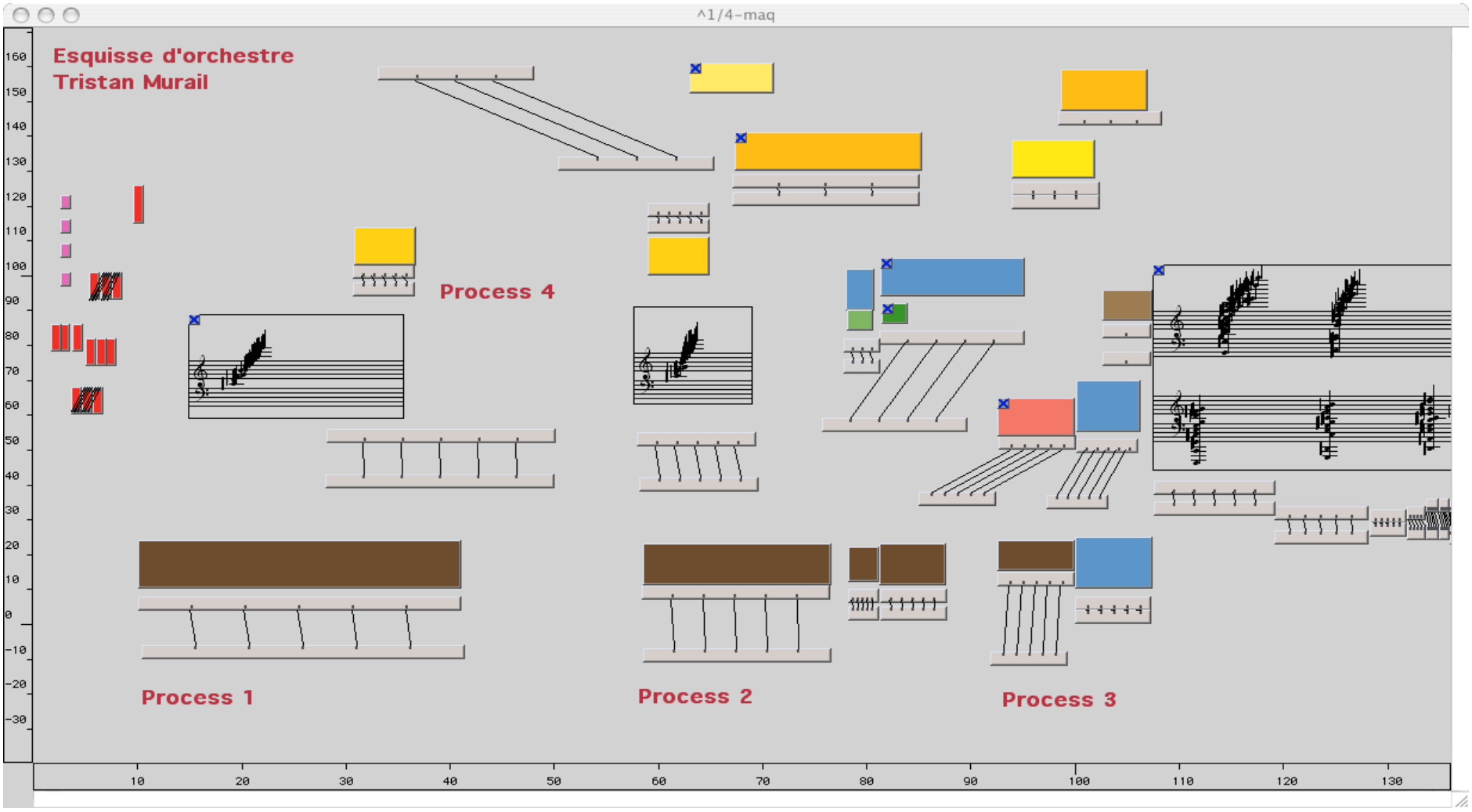
J-S Bach: Offrande Musicale

Canon 4 a 2, per Augmentationen contratio Motu
Notulis crescentibus crescat Fortuna Regis



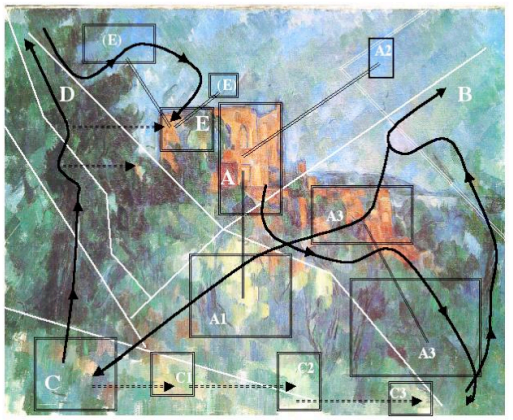


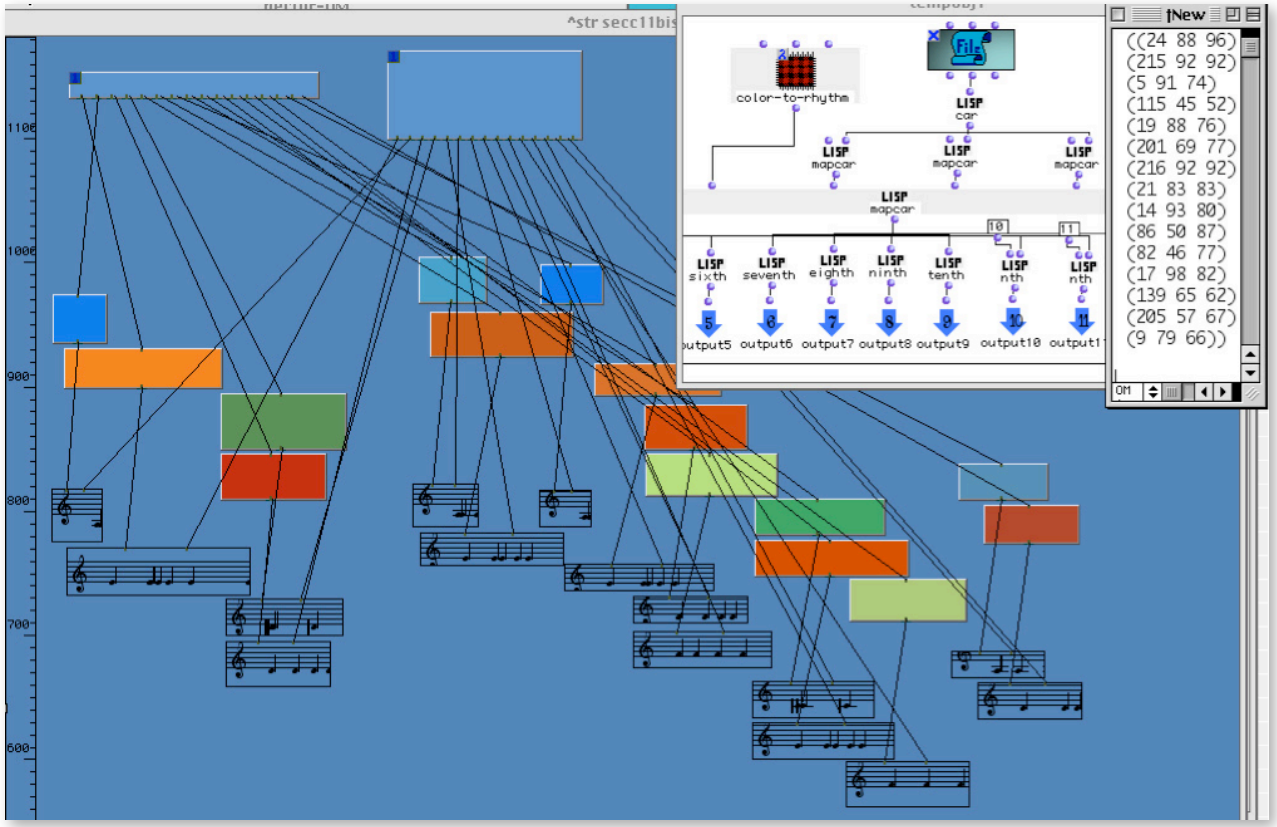
[maquette by T. Murail]

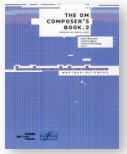


[maquette by T. Murail - zoom]

H. Parra
***Strette* (2003)**







J.-L. Hervé
***Encore* (2000)**

Extract [1'15"]