

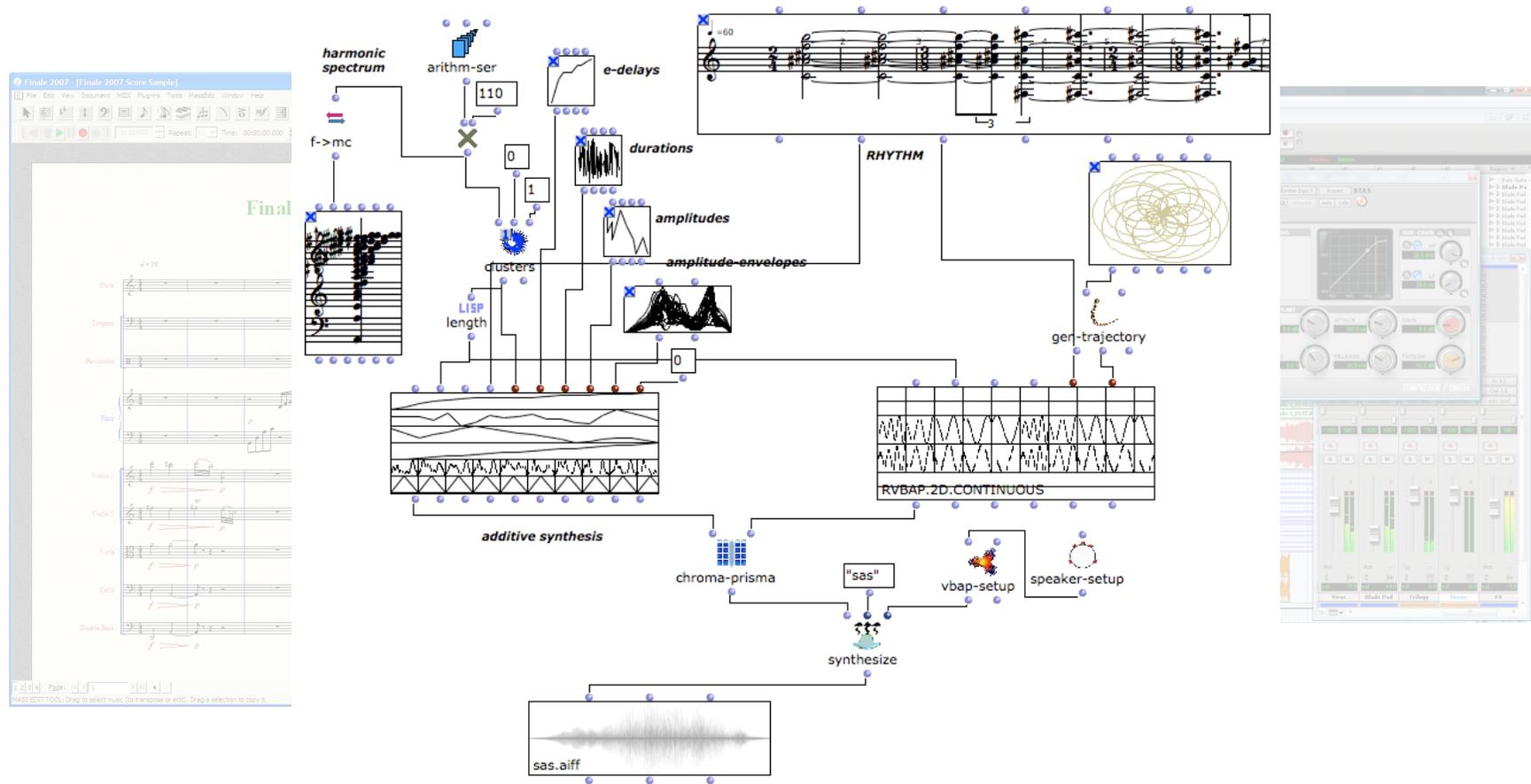
PHELMA - INP Grenoble  
Master AST (Art, Science et Technologie)

# Composition Assistée par Ordinateur ( $\Leftrightarrow$ Programmation Visuelle )

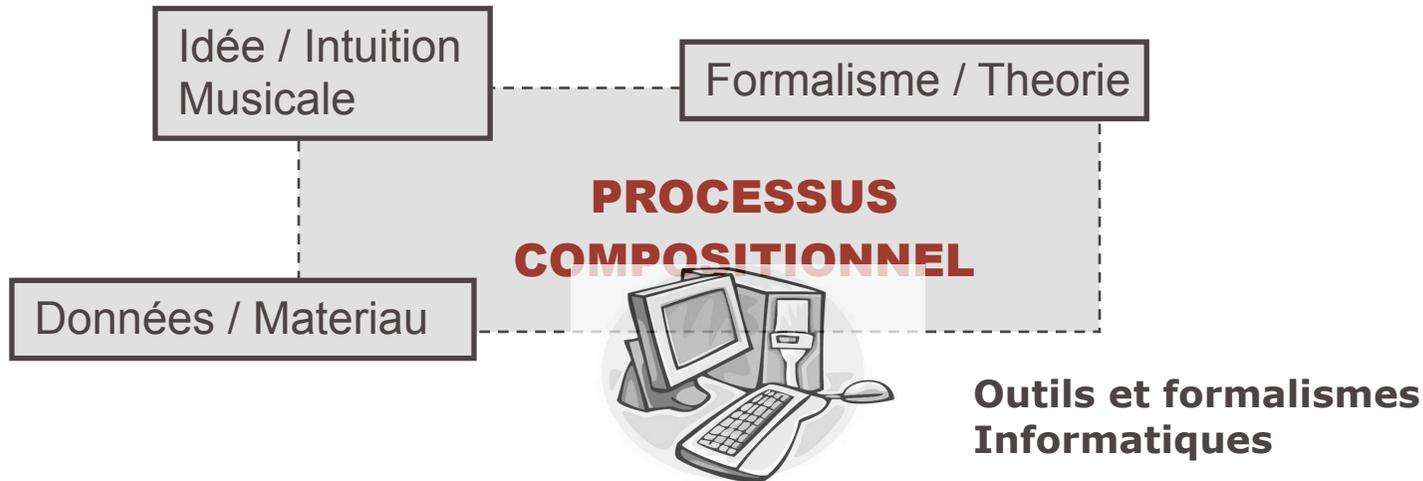
Carlos Agon - Moreno Andreatta - Jean Bresson  
IRCAM - Equipe Représentations Musicales



# 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."

4. **a 2 Per augmentationem, contrario motu**

Thema

The image shows a musical score for a piece by J-S Bach, numbered 4. The title is 'a 2 Per augmentationem, contrario motu'. The score is in G major (one sharp) and 3/4 time. It consists of three systems of two staves each. The first system is labeled 'Thema'. The notation includes various rhythmic values, including dotted notes and sixteenth notes, and features several trills marked 'tr'. The piece concludes with a double bar line and repeat dots.

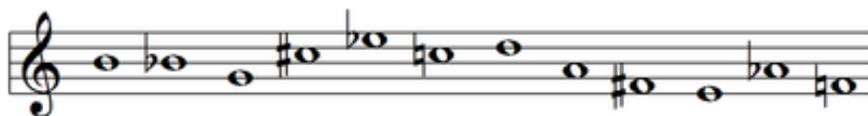
# Travail formel

## Propriétés combinatoires

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

### Example

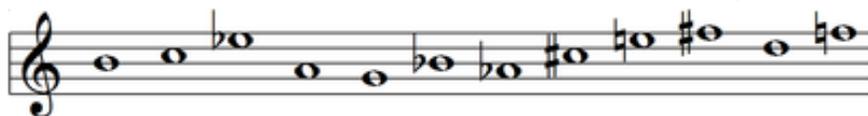
Suppose the prime series is as follows:



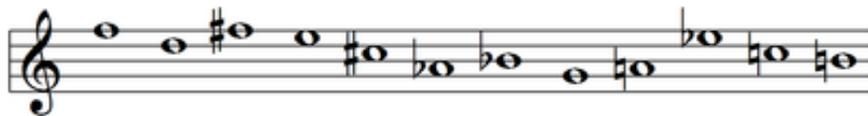
Then the retrograde is the prime series in reverse order:



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..

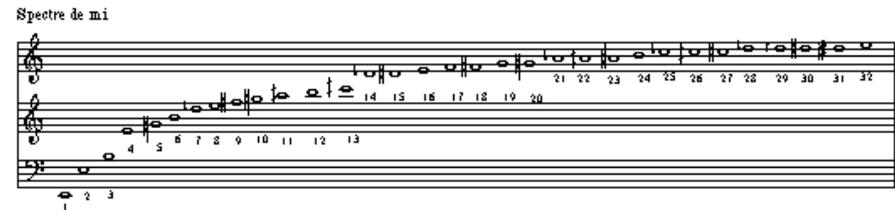
### 1975: *Partiels* (Gerard Grisey)

-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 électro-acoustiques (distorsion, modulation en anneau, filtrage...)



[1'16"]



## Développement de la musique "spectrale":

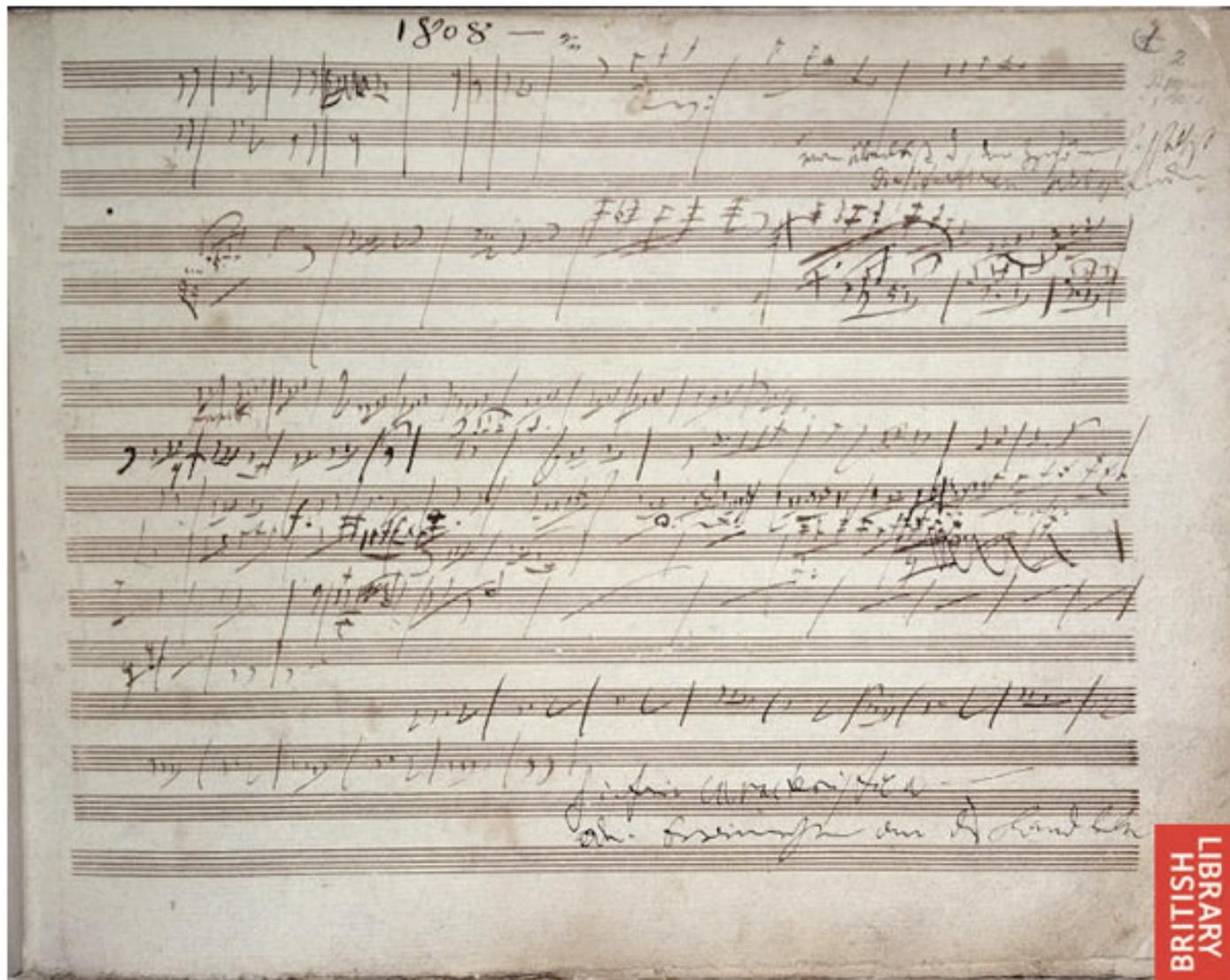
⇒ intérêt d'une assistance informatique pour l'expérimentation, le traitement et le rendu de matériaux et données musicales complexes.

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 middle zone of the trajectory, since it allows 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**

Environnements de CAO  $\Leftrightarrow$  **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)

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

**SuperCollider** (McCartney, 1998)

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

**Common Music** (H. Taube, CCRMA, 1991)

**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)

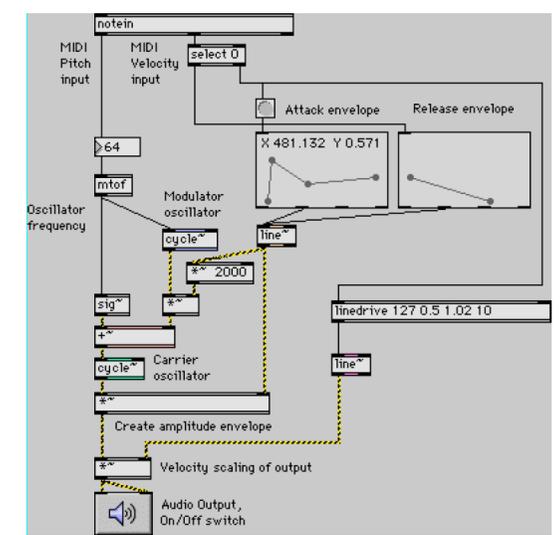
## Fonctionnels

## Lisp

## Symbolique

```
(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 (next1 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))?'time))
          (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 (new1 list-t
            (apply (get (new1 list-o (next1 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 (next1 list- t))]))))
          )))
  )))
```

## Langages Visuels



# Langages de Programmation Visuels (VPLs)

Sémantique à **+ de 1 dimension**

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

## Icônes

Exemple:

Primitives



region



ligne

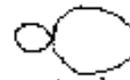


flèche

Hello

texte

Relations



touche



traverse



contient



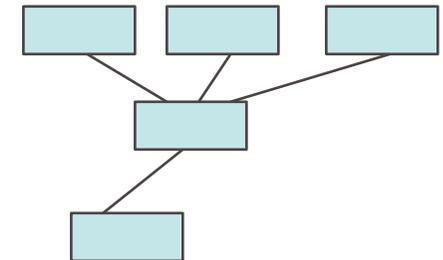
recouvre

## Formulaires

ex. MS Excel

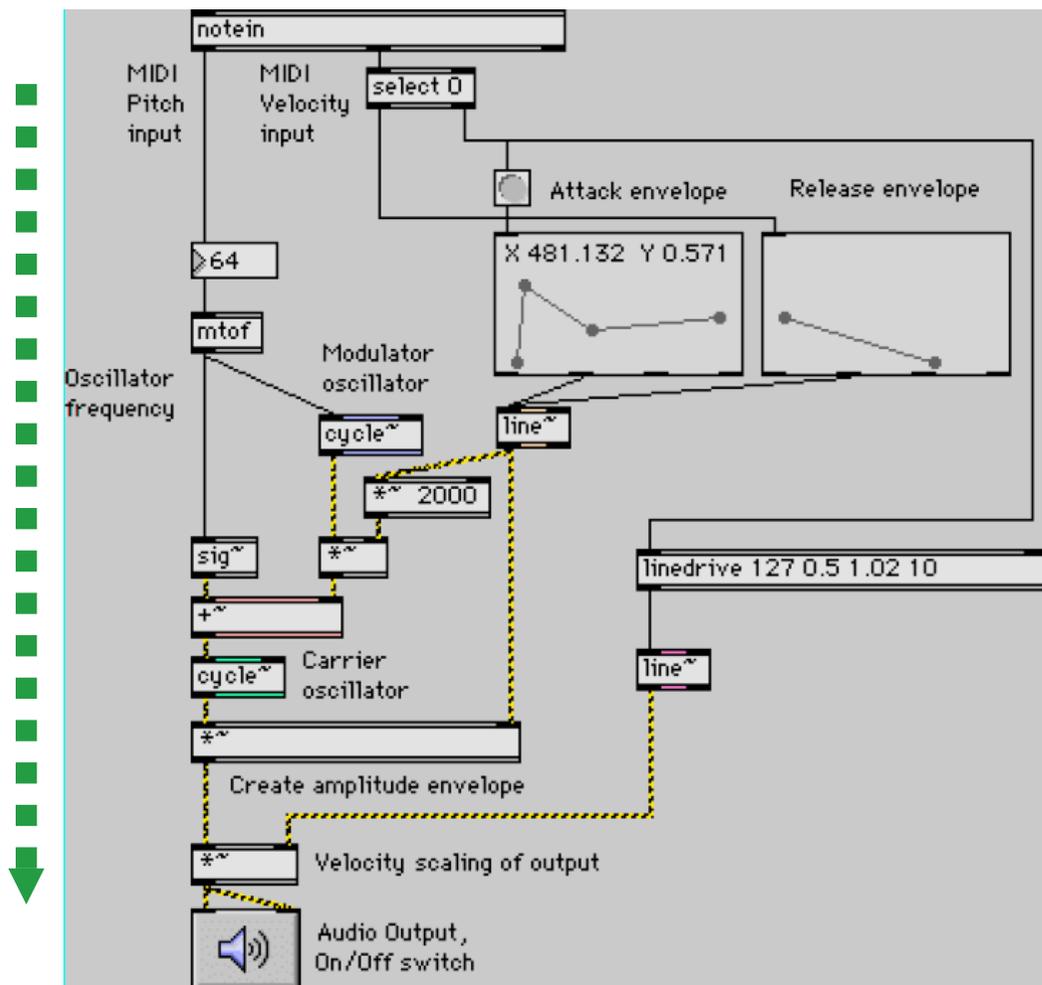
## Diagrammes

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



# Max

M. Puckette, 1991



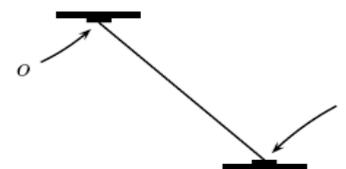
## PATCH

représentation d'un programme  
(data flow graph)

BOITE



CONNECTION

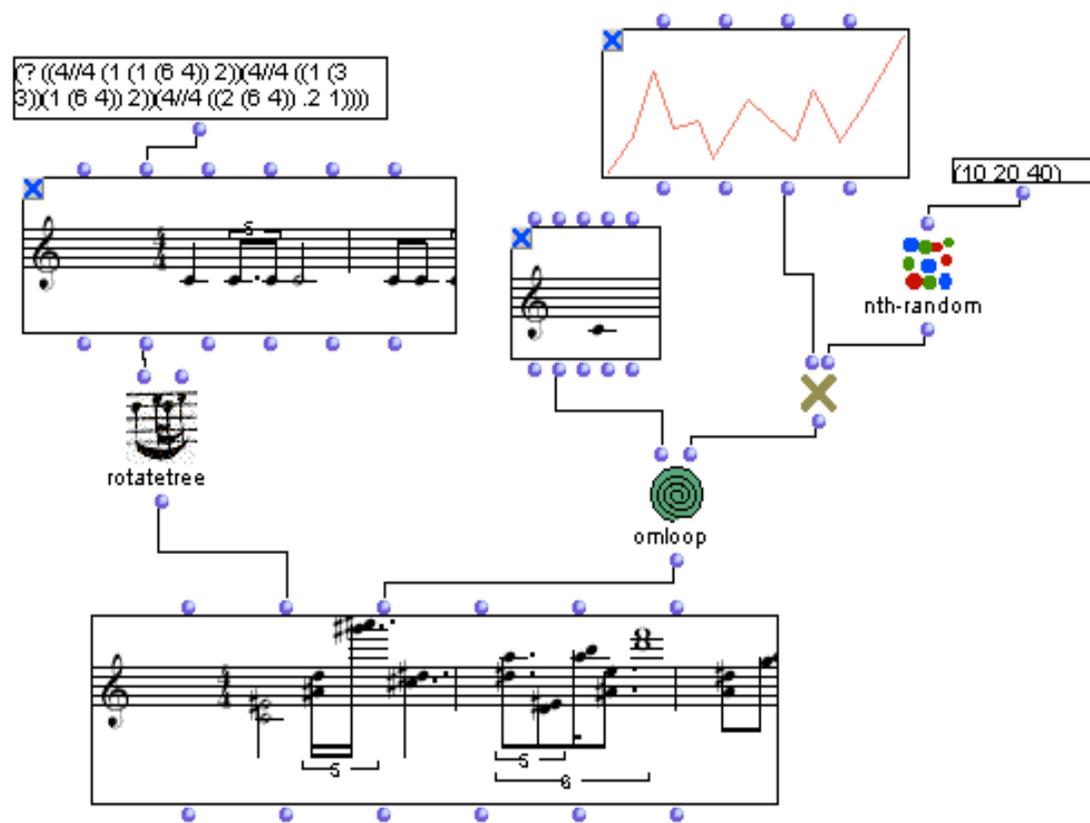


**Système réactif**

**Flot continu de données en entrée / sortie**

**Système "temps-réel"**

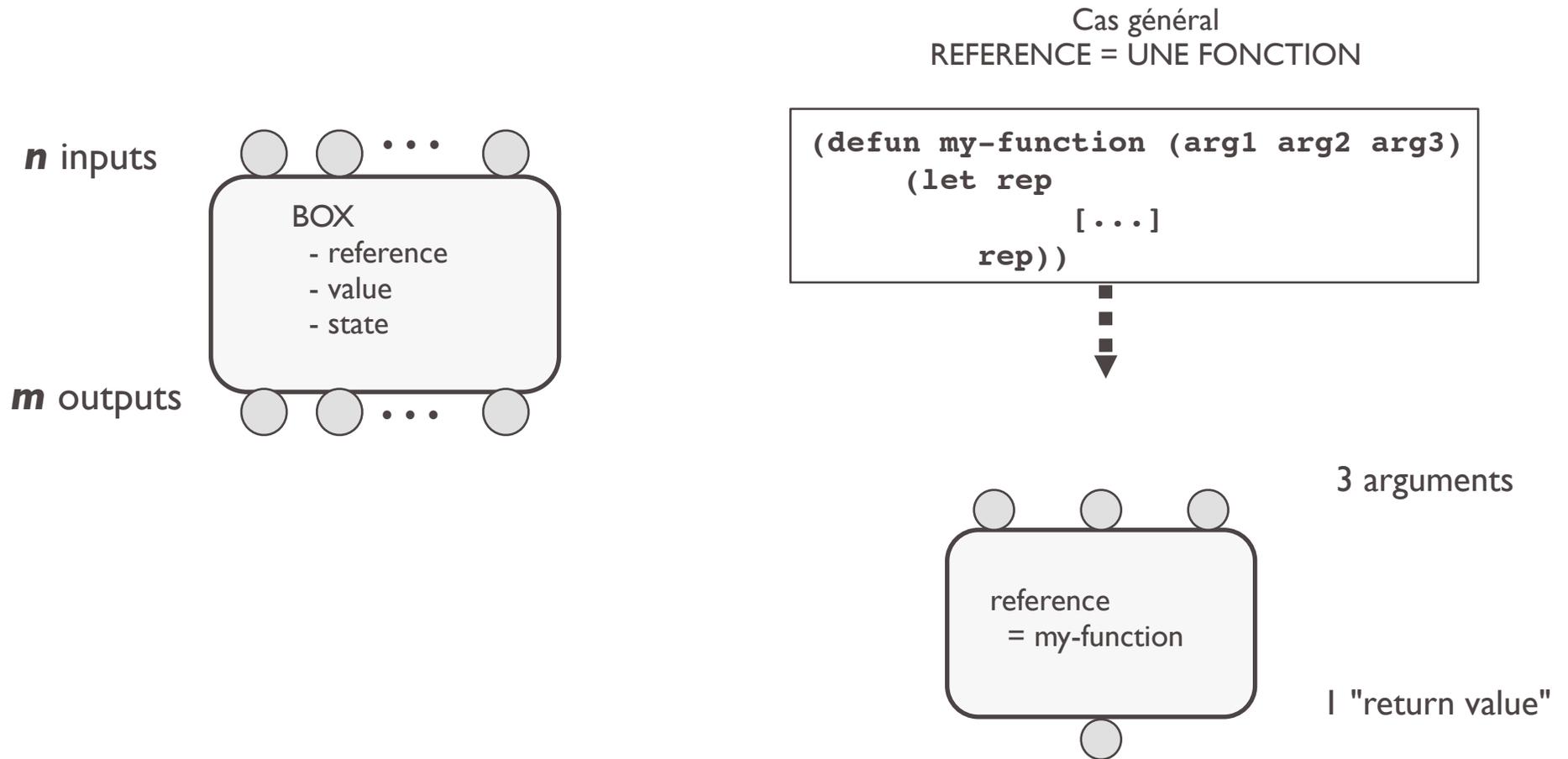
# OpenMusic (OM): un environnement de CAO / langage de programmation visuel



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

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

# Boîtes OM : éléments de syntaxe et sémantique visuelles



# Evaluation

**evaluation** d'une boîte =

Détermination des valeurs des *inputs*

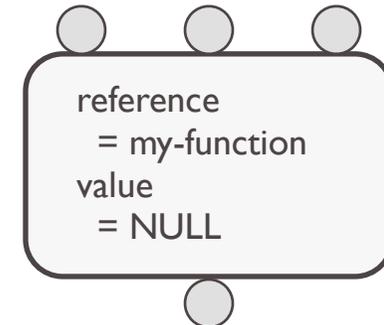
Appel (*application*) de la *reference*

```
; 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))))  
  ))
```

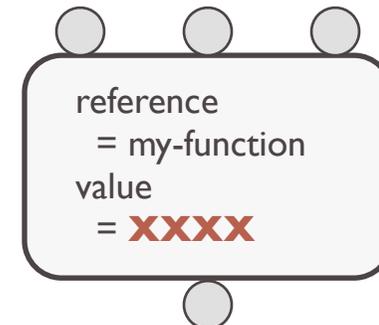
**VALUE = (apply 'myfunction '(a b c))**

INPUT = [ value, connection ]

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

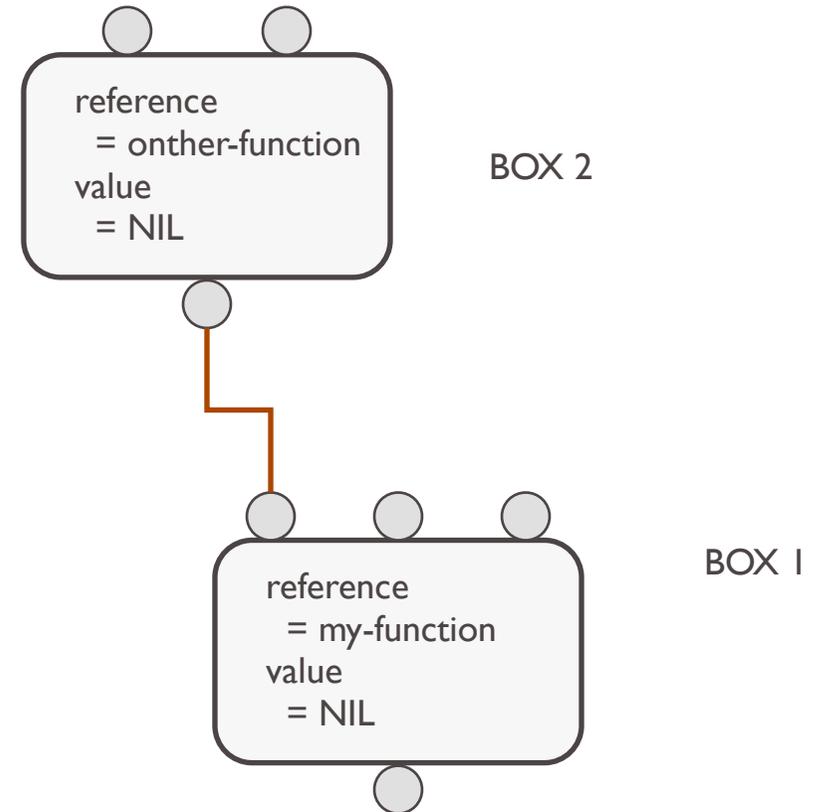


**EVAL**



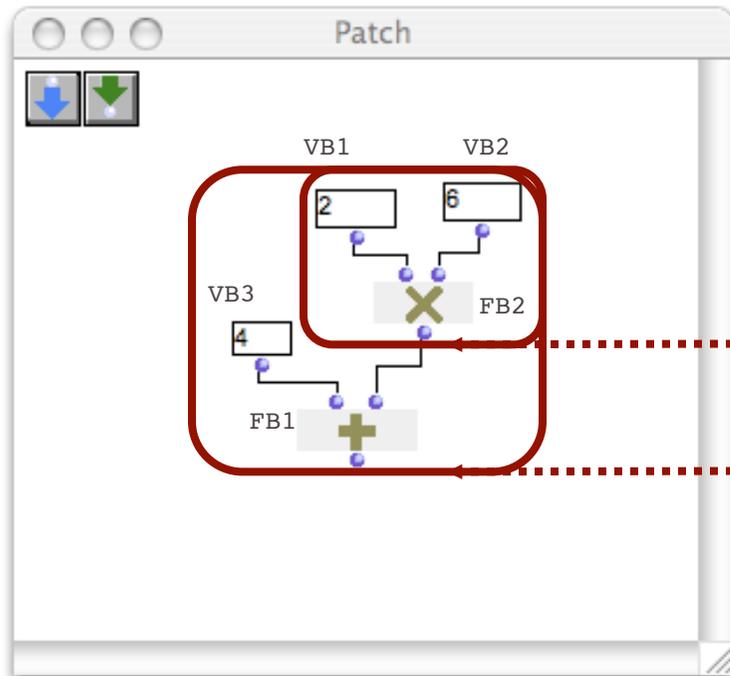
# Inputs - Connections

**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'exécution "demand-driven"



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

"value" boxes :

```
(omng-box-value VB1)  
=> 2
```

```
(omng-box-value VB2)  
=> 6
```

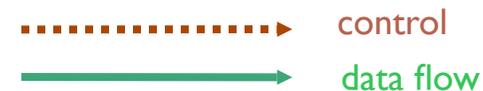
```
(omng-box-value FB2)  
=> (apply 'x (list (omng-box-value VB1) (omng-box-value VB2)))  
=> (apply 'x 2 6)
```

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

(+ 4 (x 2 6))

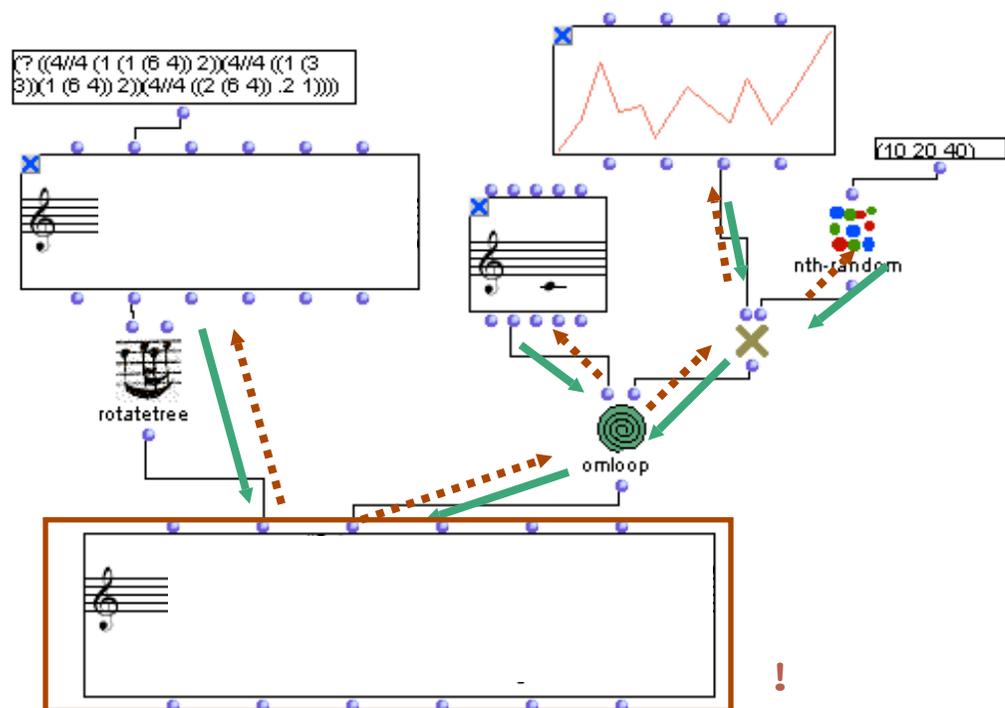
=> 48

# data flow / control flow



## demand-driven

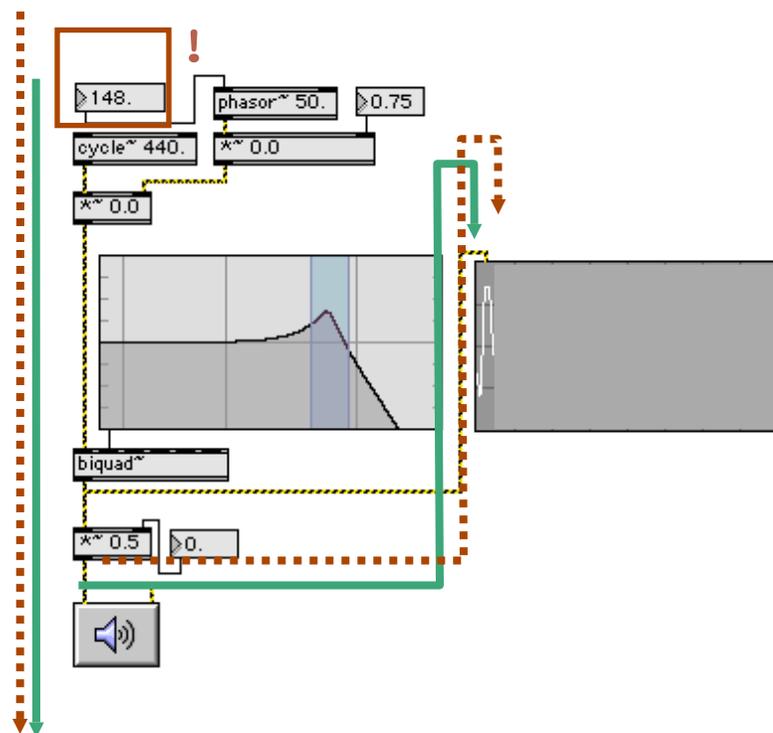
ex. OM, VPL [Lau-Kee et al., 1991]



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

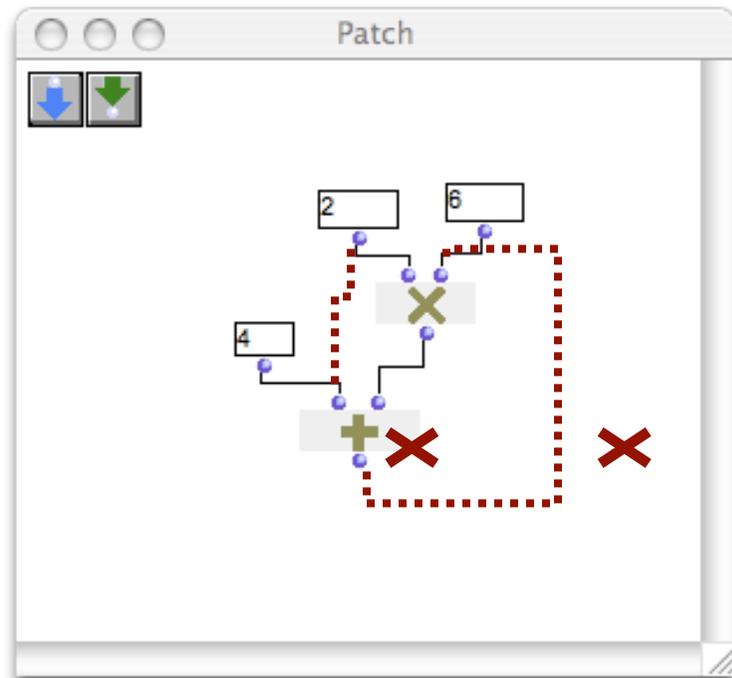
## data-driven

ex. Prograph, LabView, Khoros, Max/MSP, Pd, etc. etc.



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

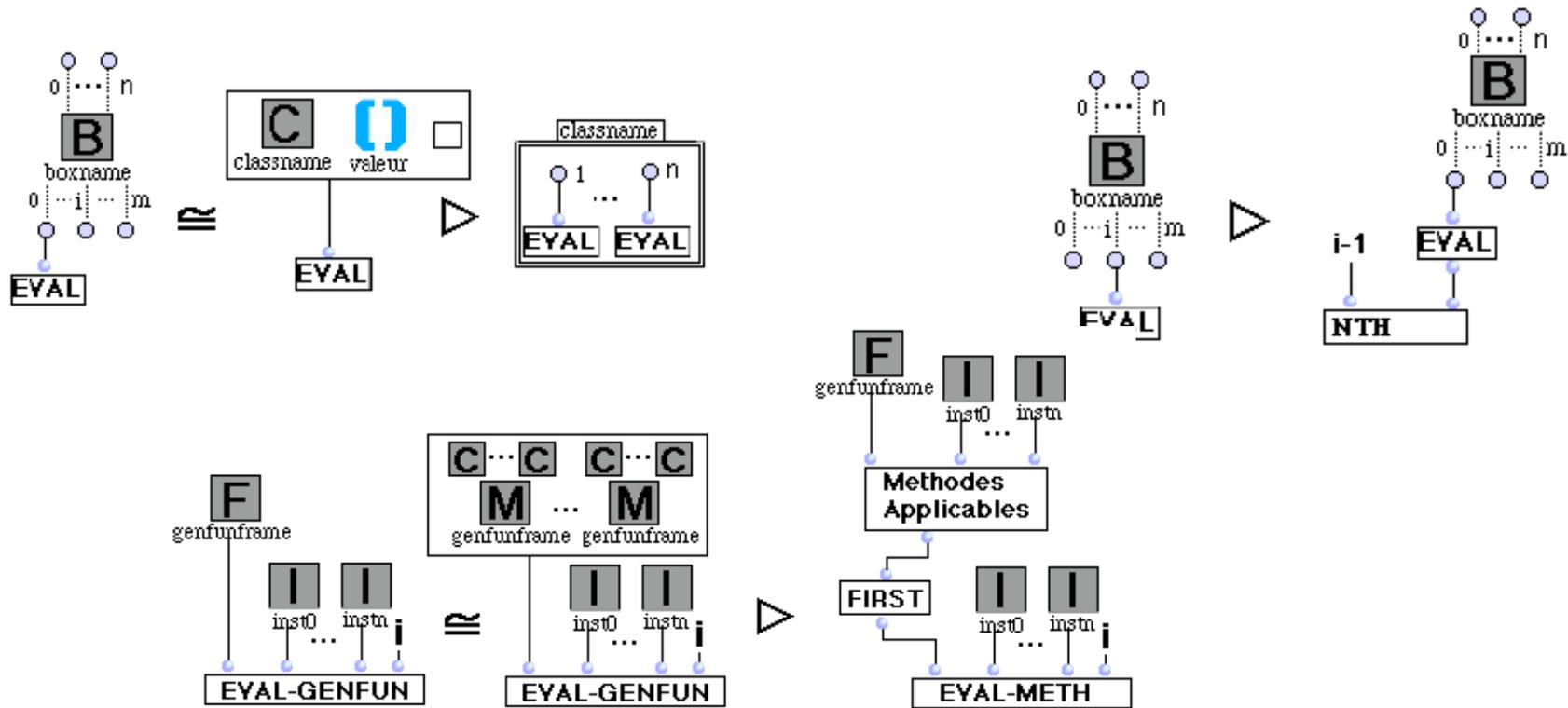
# Correction syntaxique



(+ 4 (x 2 6))

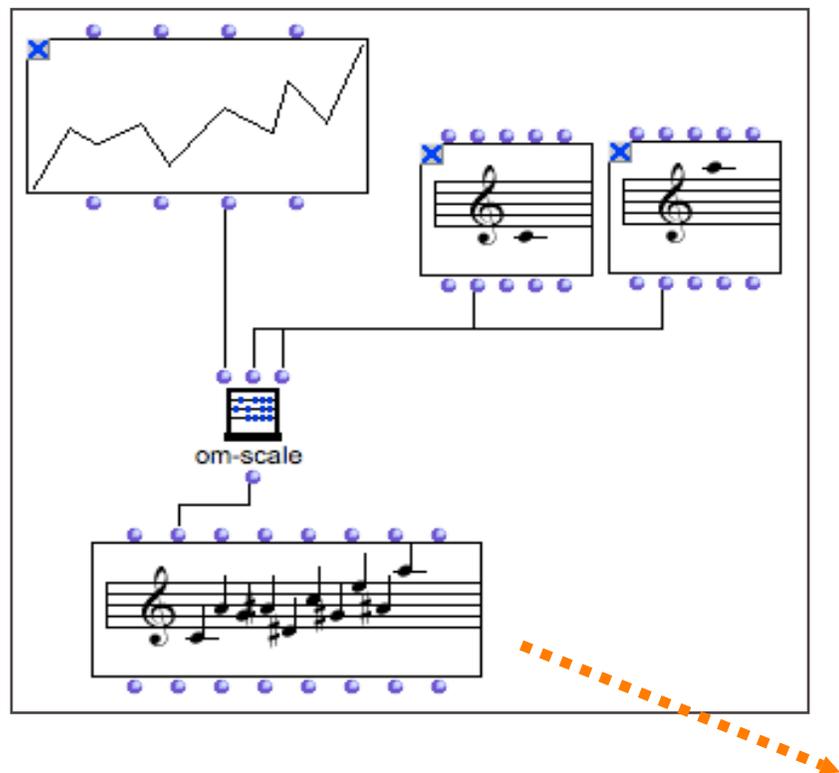
# OM: Sémantique visuelle du langage

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



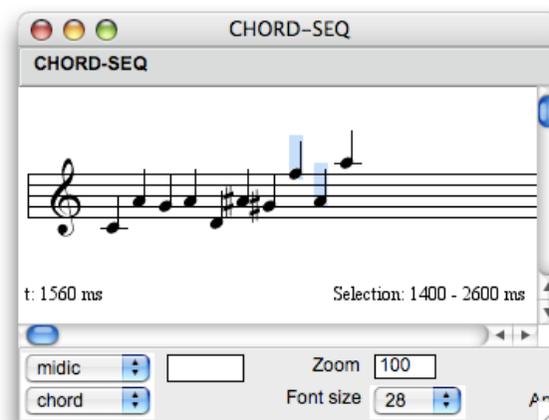
Langage réflexif bâti sur la meta-programmation  
(i.e. capable se représenter lui-même)

## Factories / Editeurs



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

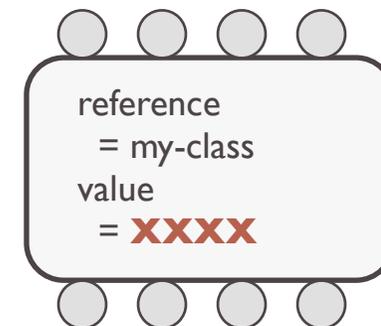
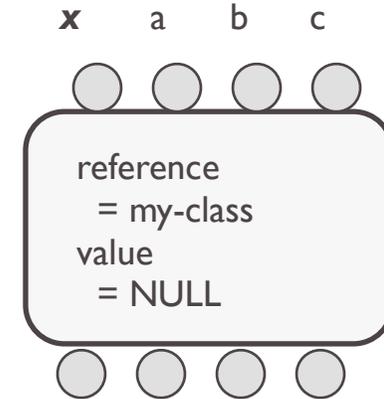
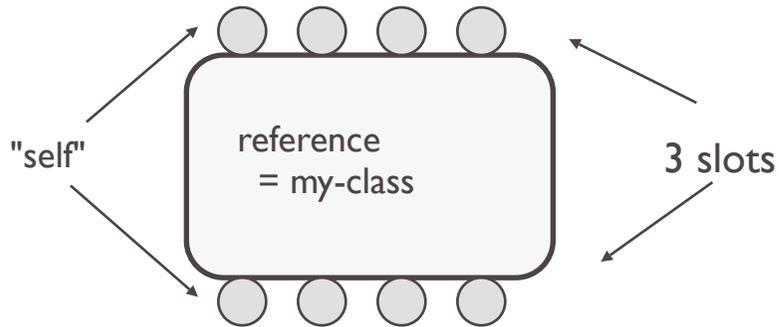
[ autre aspect important du point de vue de la CAO :  
dualité objet / processus, etc. ]



# Boîtes "factory"

REFERENCE = UNE CLASSE

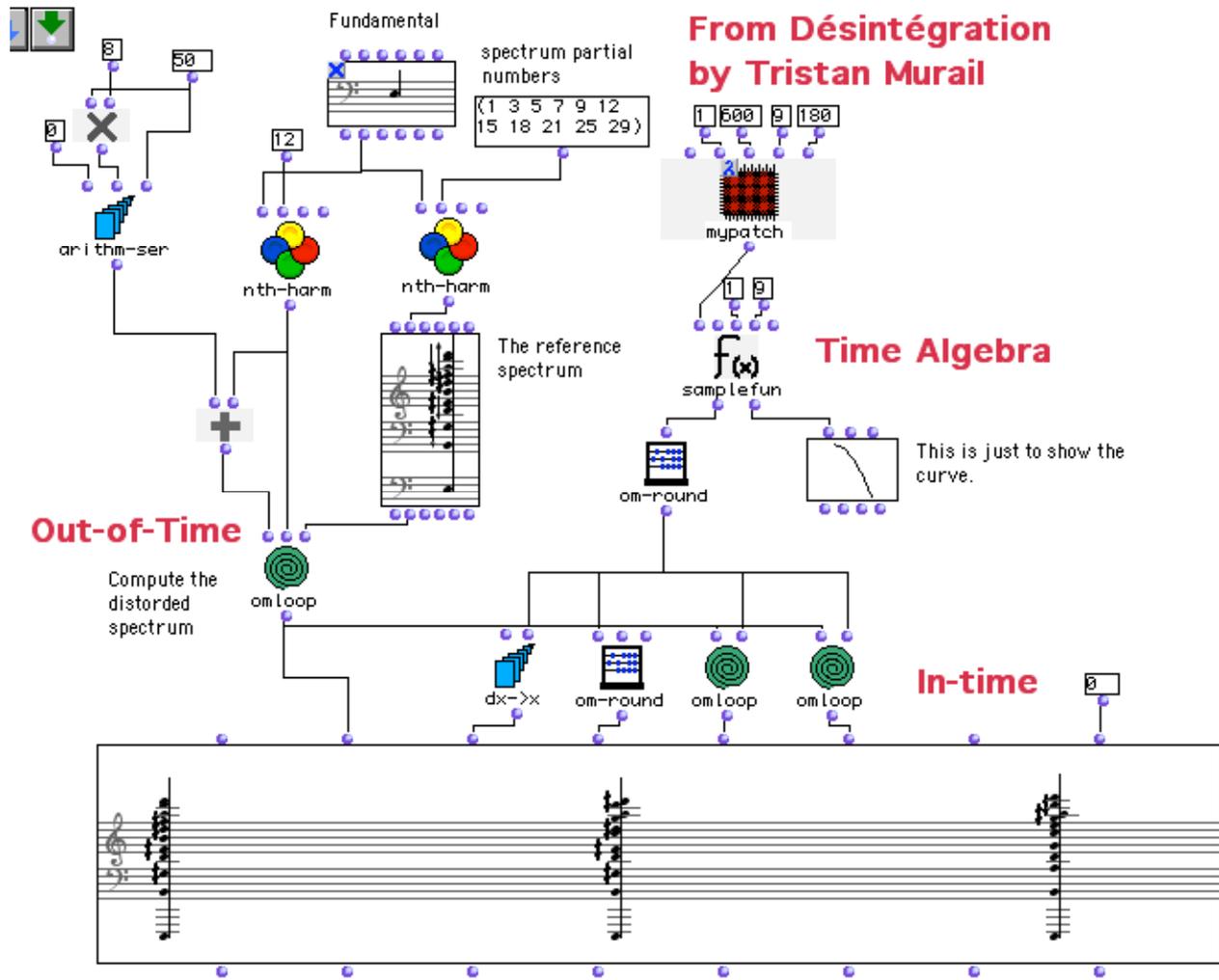
```
(defclass my-class ()  
  ((slot 1 ...)  
   (slot 2 ...)  
   (slot 3 ...)))
```



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

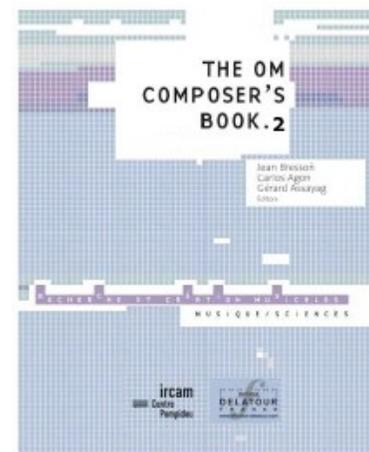
# OpenMusic : Spectral music revisited

Desintegration (T. Murail)



[0'18"]

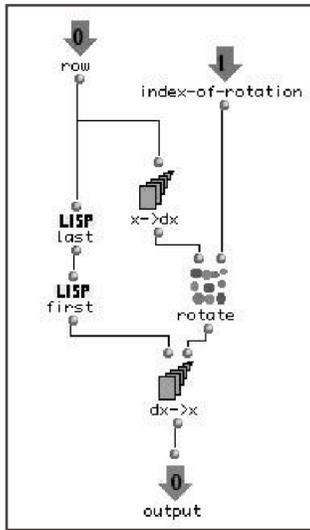
# Retour sur quelques notions et problèmes informatiques avec OpenMusic



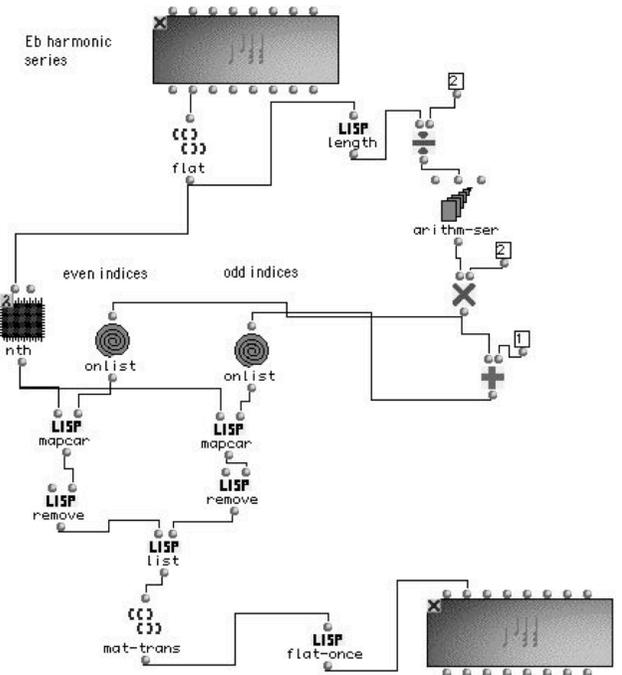
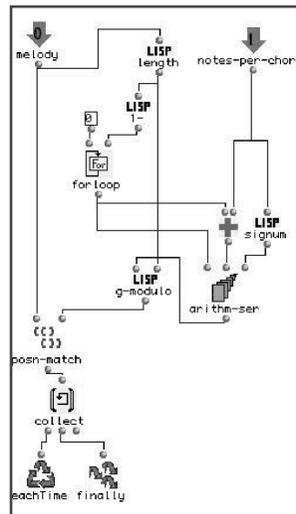
*The OM Composer's Book*  
J. Bresson, C. Agon, G. Assayag (Eds.)

J. Kretz  
*Second Horizon*  
 (2002)

traitement de hauteurs sous forme de listes



Chords generation



Permutations of the series interval



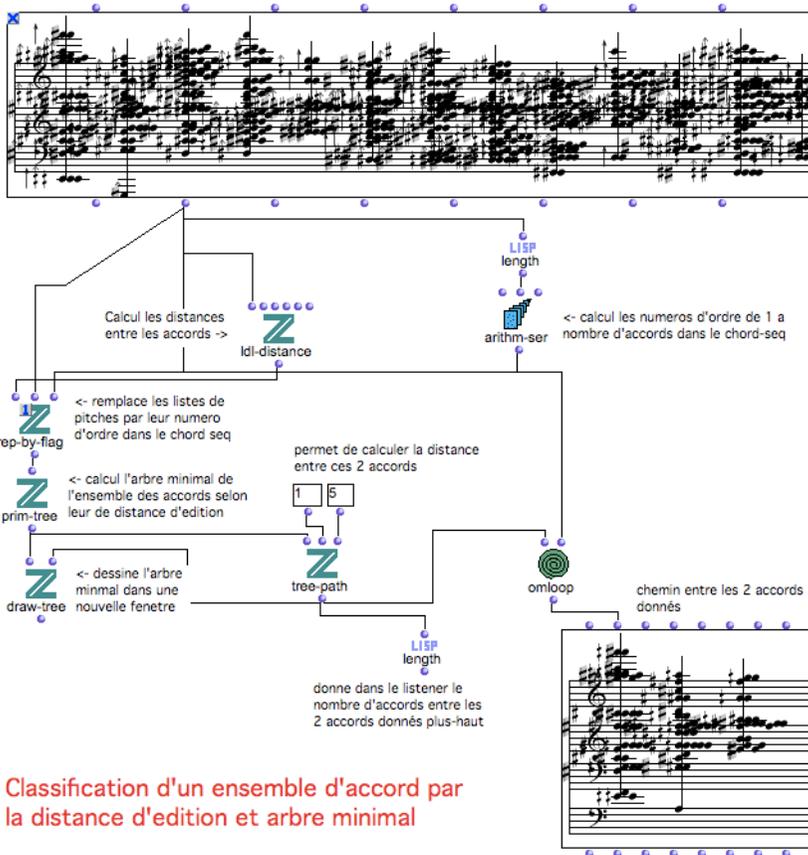
K. Nez  
*Sculpted Implosions*  
 (1999)



# Problèmes de classification

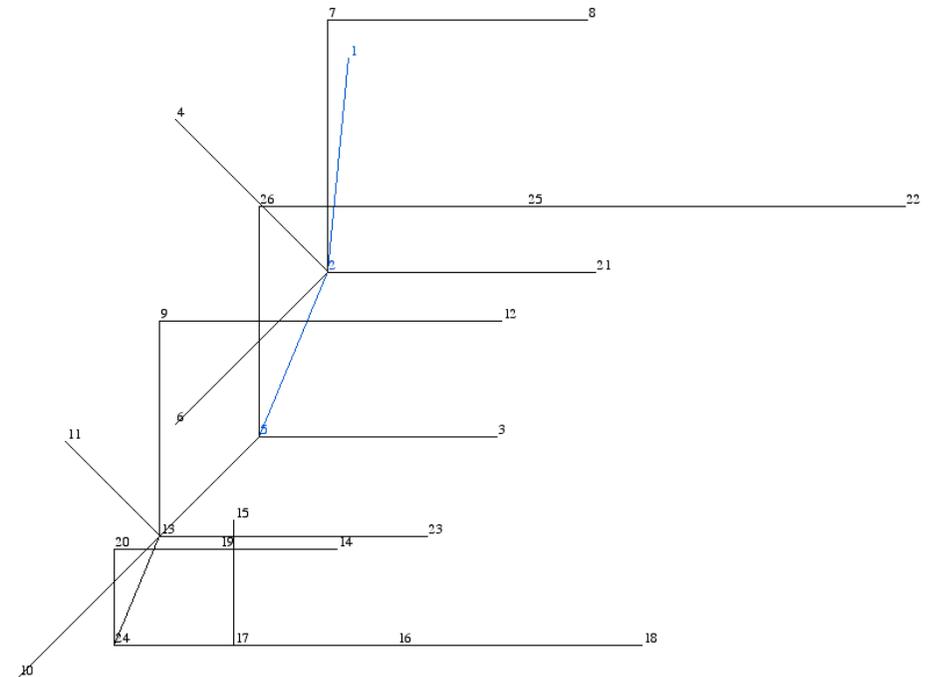
## Recherche combinatoire

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



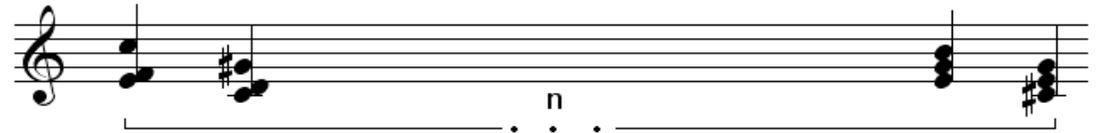
# Abstraction

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

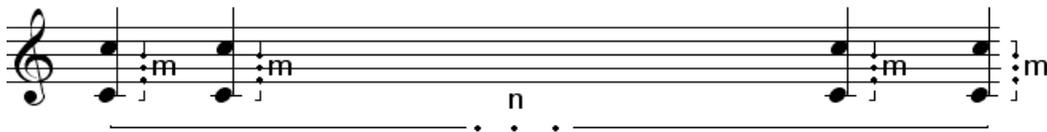
```
(repeat 10 (repeat 3 (random D03 D04)))
```



```
F1(n)=(repeat n (repeat 3 (random D03 D04)))
```



```
F2(n m)=(repeat n (repeat m (random D03 D04)))
```



```
F3(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  $B$

$F_3(n m a b) (5)$

$F_4(m a b) = (\text{repeat } 5 (\text{repeat } m (\text{random } a b)))$

Musical notation for  $F_3(n m a b) (5)$ . It shows five measures on a treble clef staff. Each measure contains two notes: 'a' on the second line and 'b' on the first space. The notes are connected by a vertical line with a horizontal bar at the top. A brace on the right side of each measure indicates that the notes are repeated  $m$  times. The label 'm' is placed below the brace. The notes are separated by a small interval, and the measures are separated by a larger interval.

$F_4(m a b) (4)$

$F_5(a b) = (\text{repeat } 5 (\text{repeat } 4 (\text{random } a b)))$

Musical notation for  $F_4(m a b) (4)$ . It shows five measures on a treble clef staff. Each measure contains two notes: 'a' on the second line and 'b' on the first space. The notes are connected by a vertical line with a horizontal bar at the top. A brace on the right side of each measure indicates that the notes are repeated  $m$  times. The label 'm' is placed below the brace. The notes are separated by a small interval, and the measures are separated by a larger interval.

$F_5(a b) (C2) (C5)$

$(\text{repeat } 5 (\text{repeat } 4 (\text{random } C2 C5)))$

Musical notation for  $F_5(a b) (C2) (C5)$ . It shows five measures on a treble clef staff. Each measure contains two notes: 'a' on the second line and 'b' on the first space. The notes are connected by a vertical line with a horizontal bar at the top. A brace on the right side of each measure indicates that the notes are repeated  $m$  times. The label 'm' is placed below the brace. The notes are separated by a small interval, and the measures are separated by a larger interval.

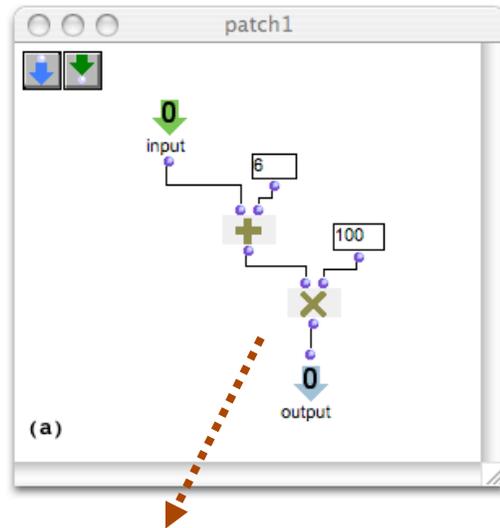
# Abstraction

```
(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)
```

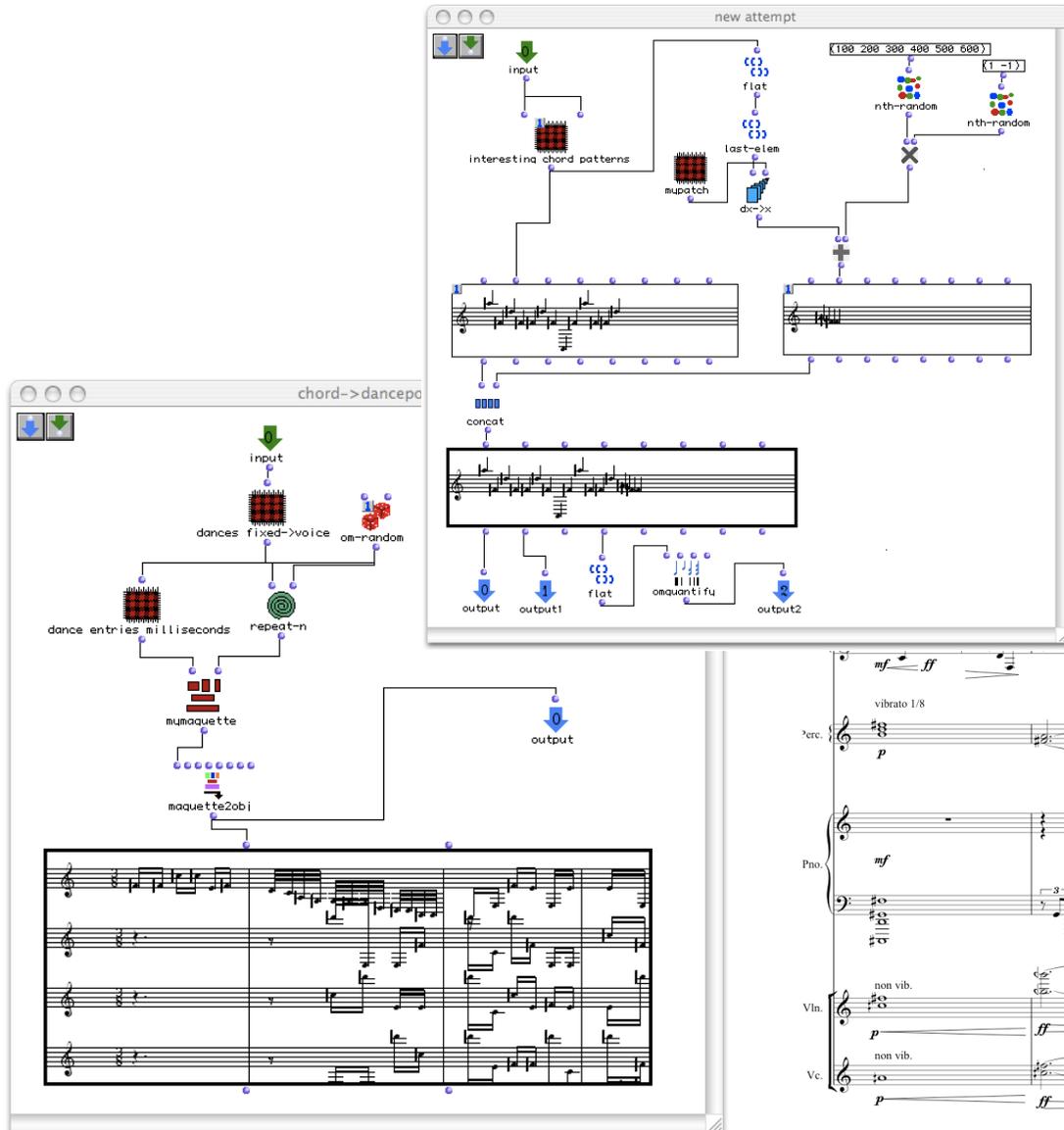
```
; Generates the Lisp code for a function box
(defmethod gen-code ((self OMBoxCall))
  '(,(reference self) ,@(mapcar #'gen-code (inputs self))))
```

```
; 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



The musical score for "Starbog (2006)" is presented in a multi-staff format. It includes parts for Piano (Pno.), Violin (Vln.), and Voice (Vc.). The score is characterized by dynamic markings such as *mf*, *ff*, *pp*, *f*, *mp*, *fff*, and *ppp*. Performance instructions include "vibrato 1/8" for the voice part and "non vib." for the violin and voice parts. The score features complex rhythmic patterns, including triplets and sixteenth-note runs, and uses various articulation marks like slurs and accents.

# F. Lévy

## Hérébo-Ribotes (2002)

209

209

31

accord->puls

cs-revers

cs-timestrech

cs-transptemp

concat

chordseq

Notesaccord

FréqRéf

DurFréqRéf

dur

freq

acc

cs

0 : Durée maximale (en sec.)  
 1 : fréquence de référence (midicents)  
 2 : Durée de la fréquence de référence (en s)  
 3 : accord

CHORD-SEQ3

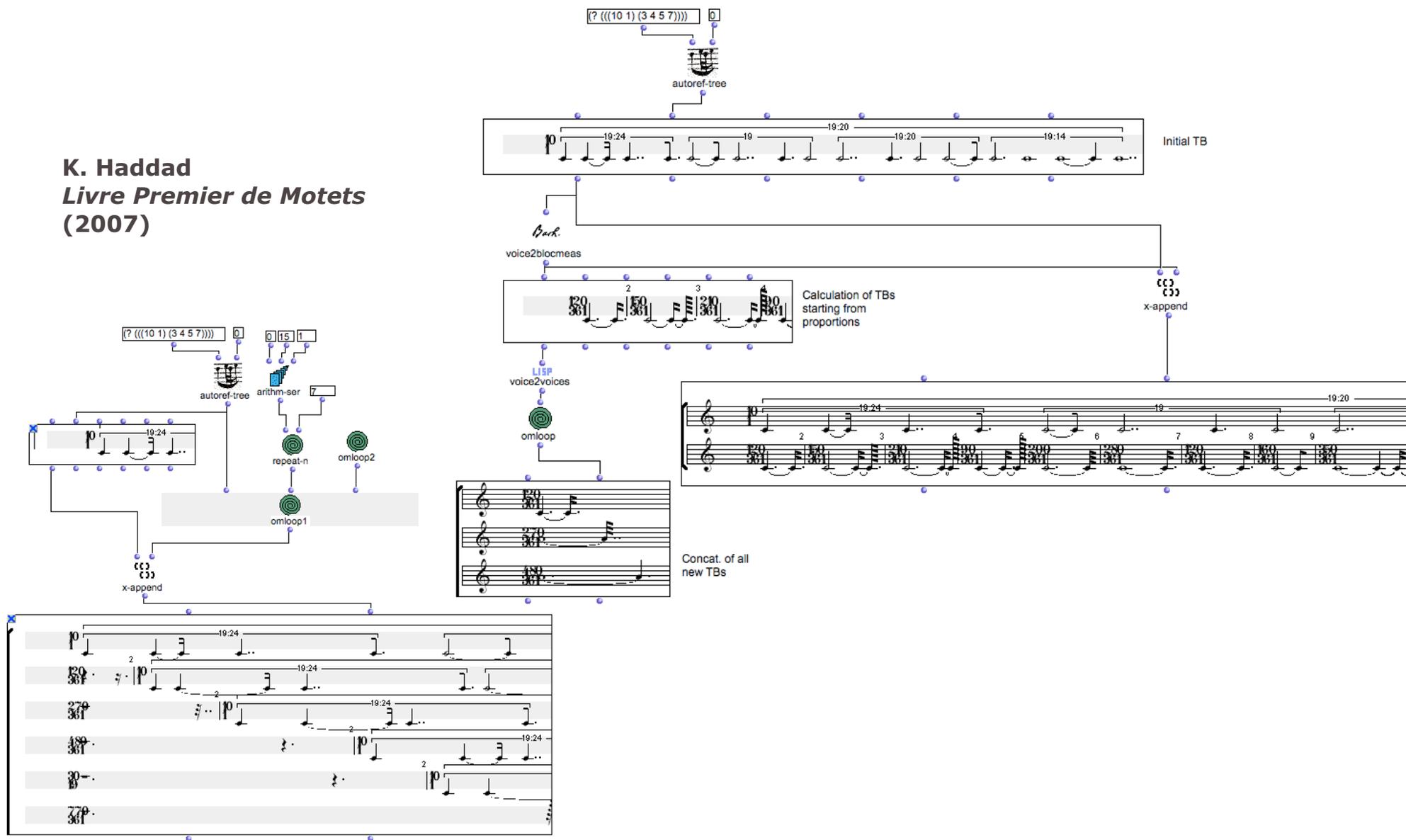
General Palette

hide 100 1/4 16 InPort 0

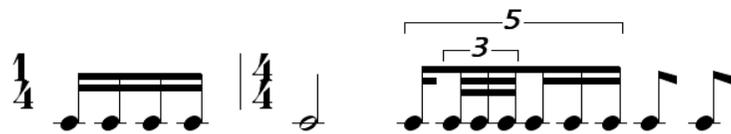


# traitement des rythmes sous forme d'arbres

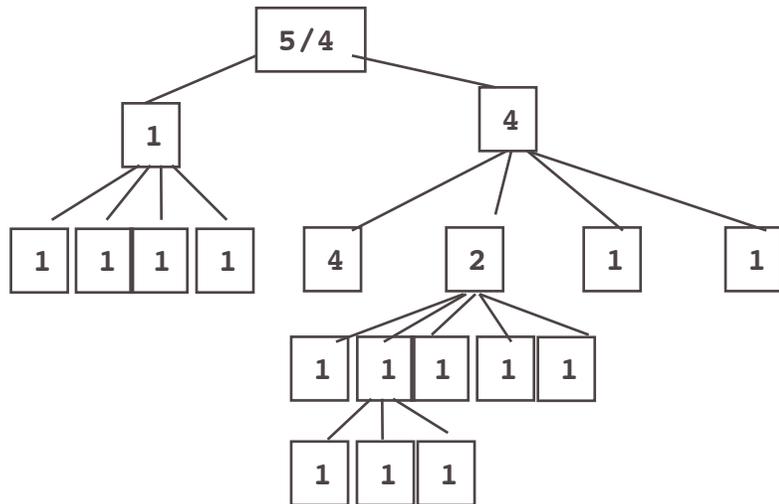
**K. Haddad**  
**Livre Premier de Motets**  
**(2007)**



# Exemple (récursivité)



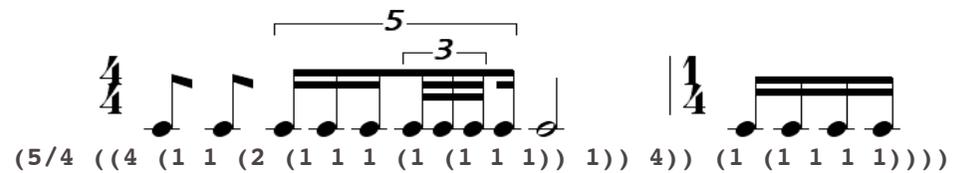
(5/4 ((1 (1 1 1 1)) (4 (4 (2 (1 (1 (1 1 1)) 1 1 1)) 1 1))))



$a = (\text{root}, (a_1 \dots a_n))$

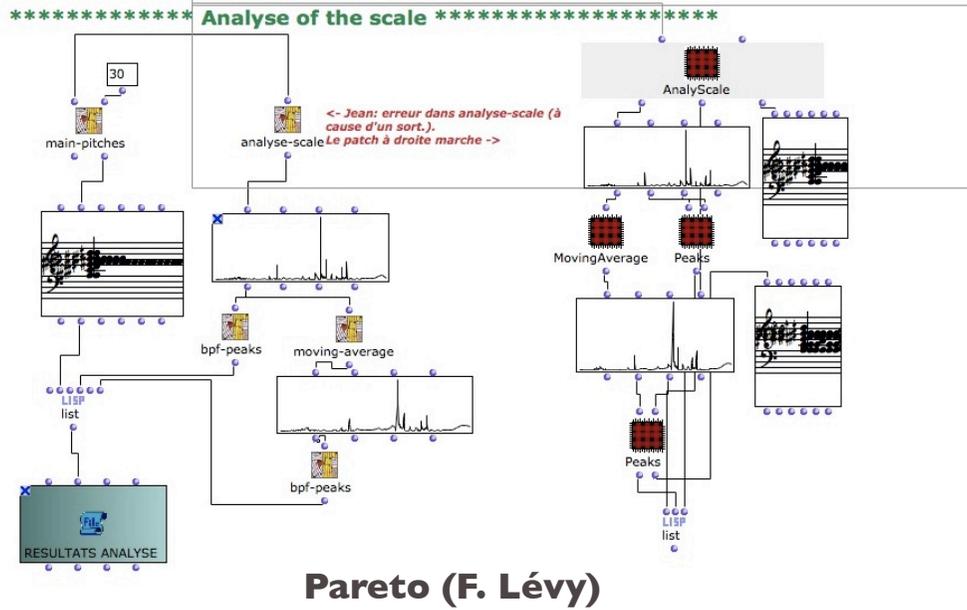
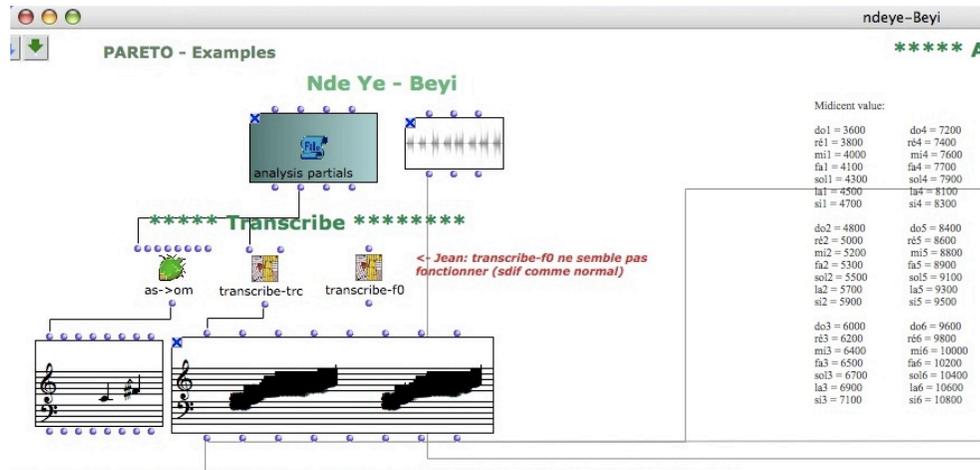
## Inversion rythmique => traitement récursif de l'arbre

$R(a) = (\text{root}, (R(a_n), \dots, R(a_1)))$

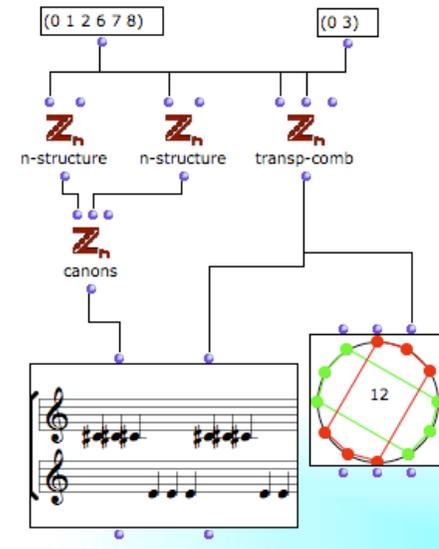


(5/4 ((4 (1 1 (2 (1 1 1 (1 (1 1 1)) 1)) 1)) 4)) (1 (1 1 1 1))))

# Analyse "computationnelle" des structures musicales

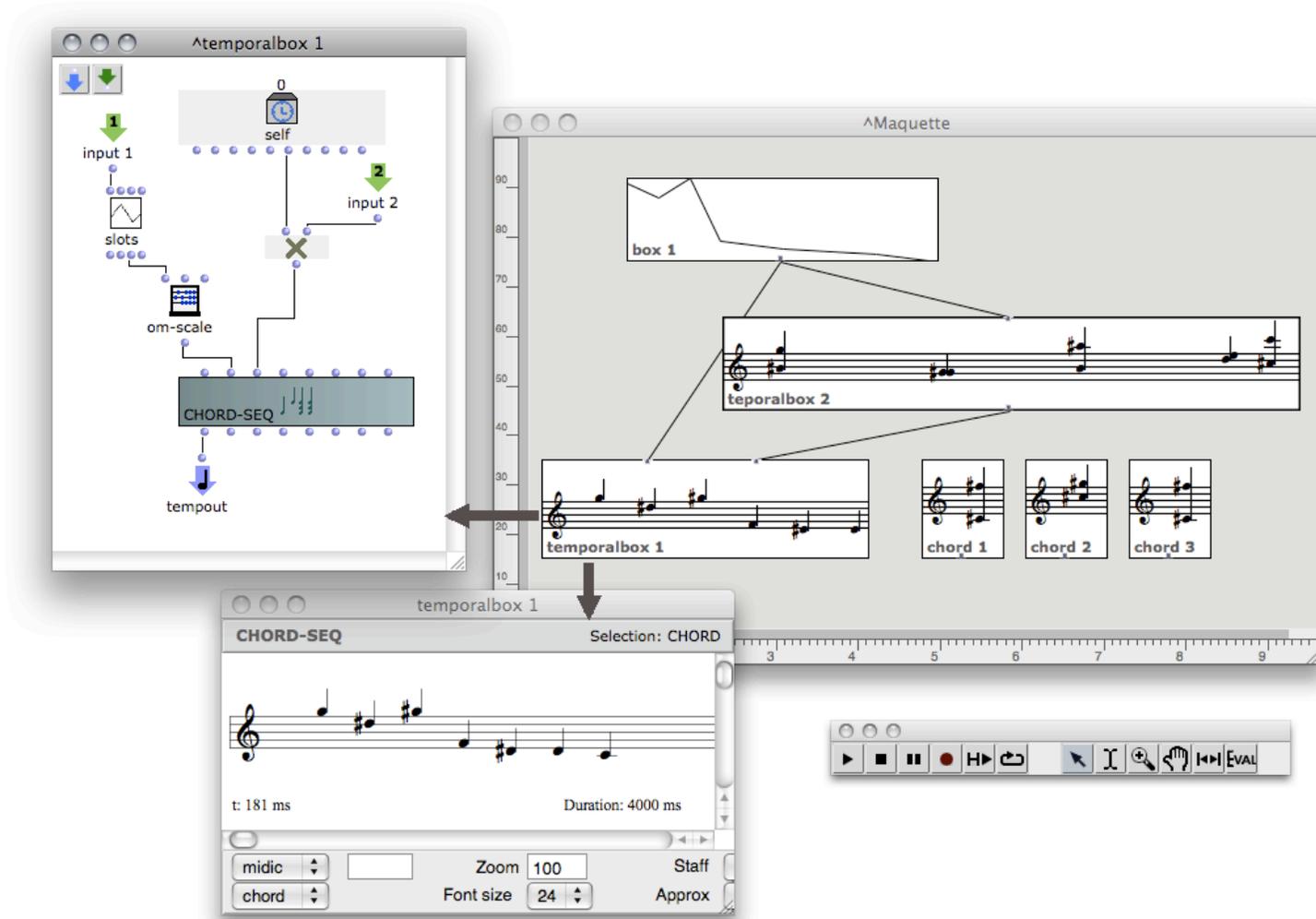


## MathTools (M. Andreatta, C. Agon)



Représentations temporelles pour la composition  
(*maquette*)

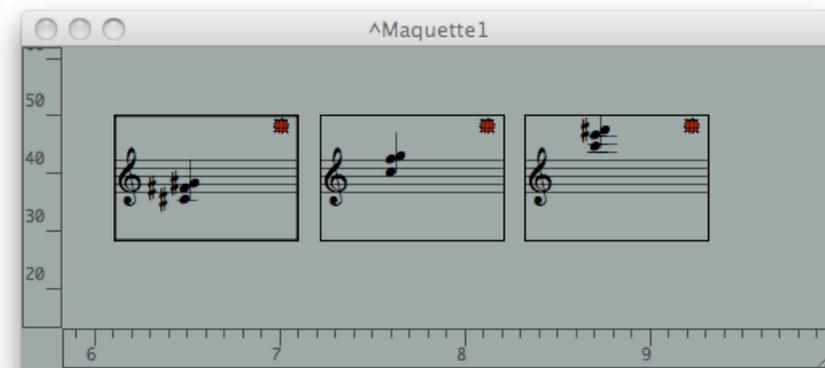
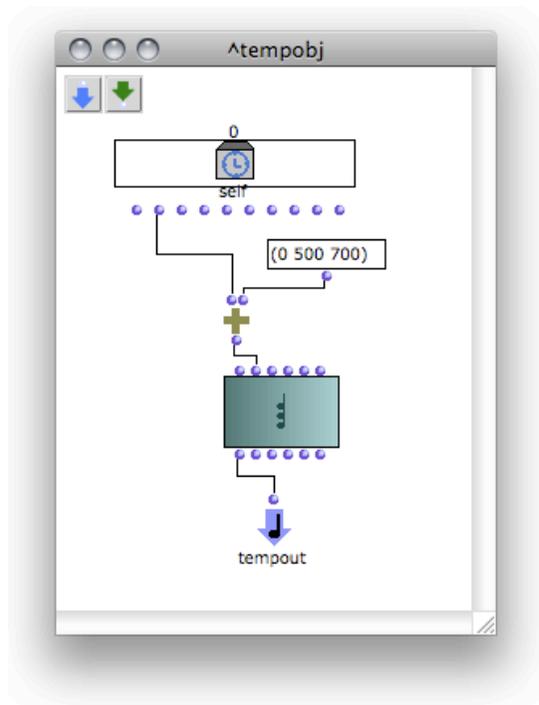
# MAQUETTE: time structure/visual program



# MAQUETTE

Visual Meta Object Programming  
(Agon, 2003)

*When the time layout drives calculus...*

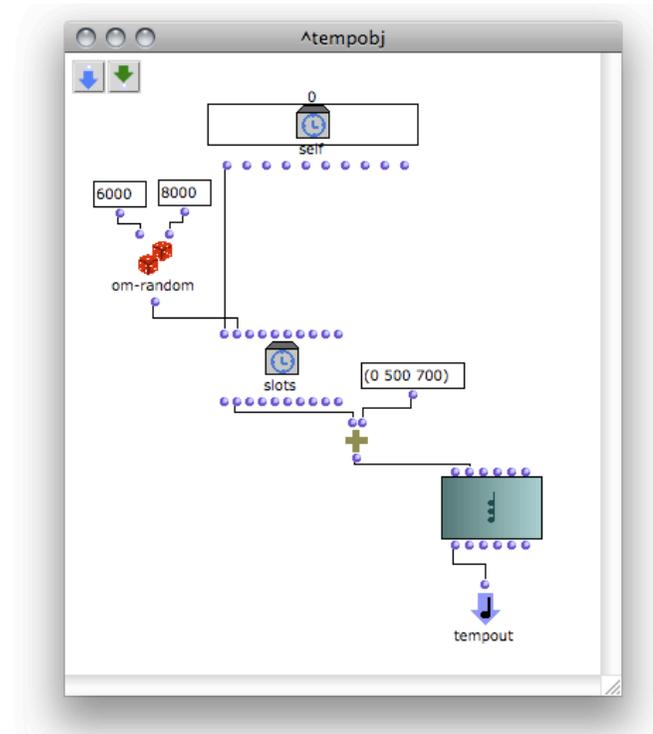
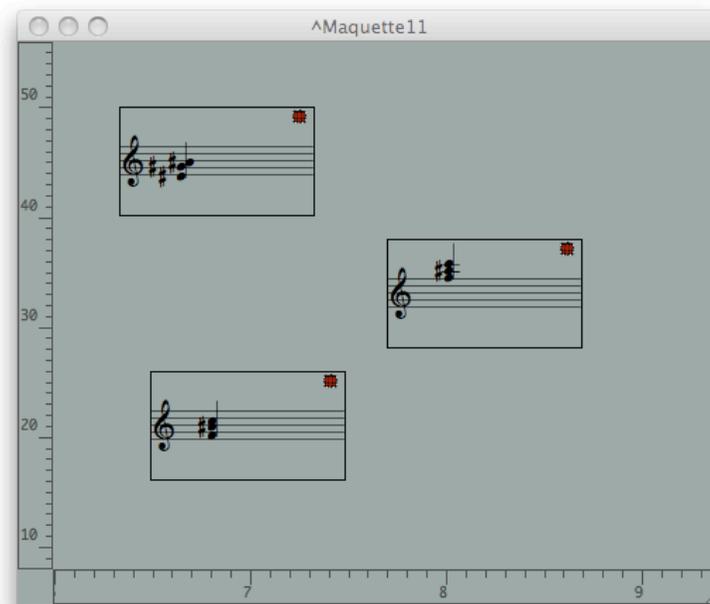


*Ex. La hauteur (transposition) dépend de la position (dans le temps)*

# 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

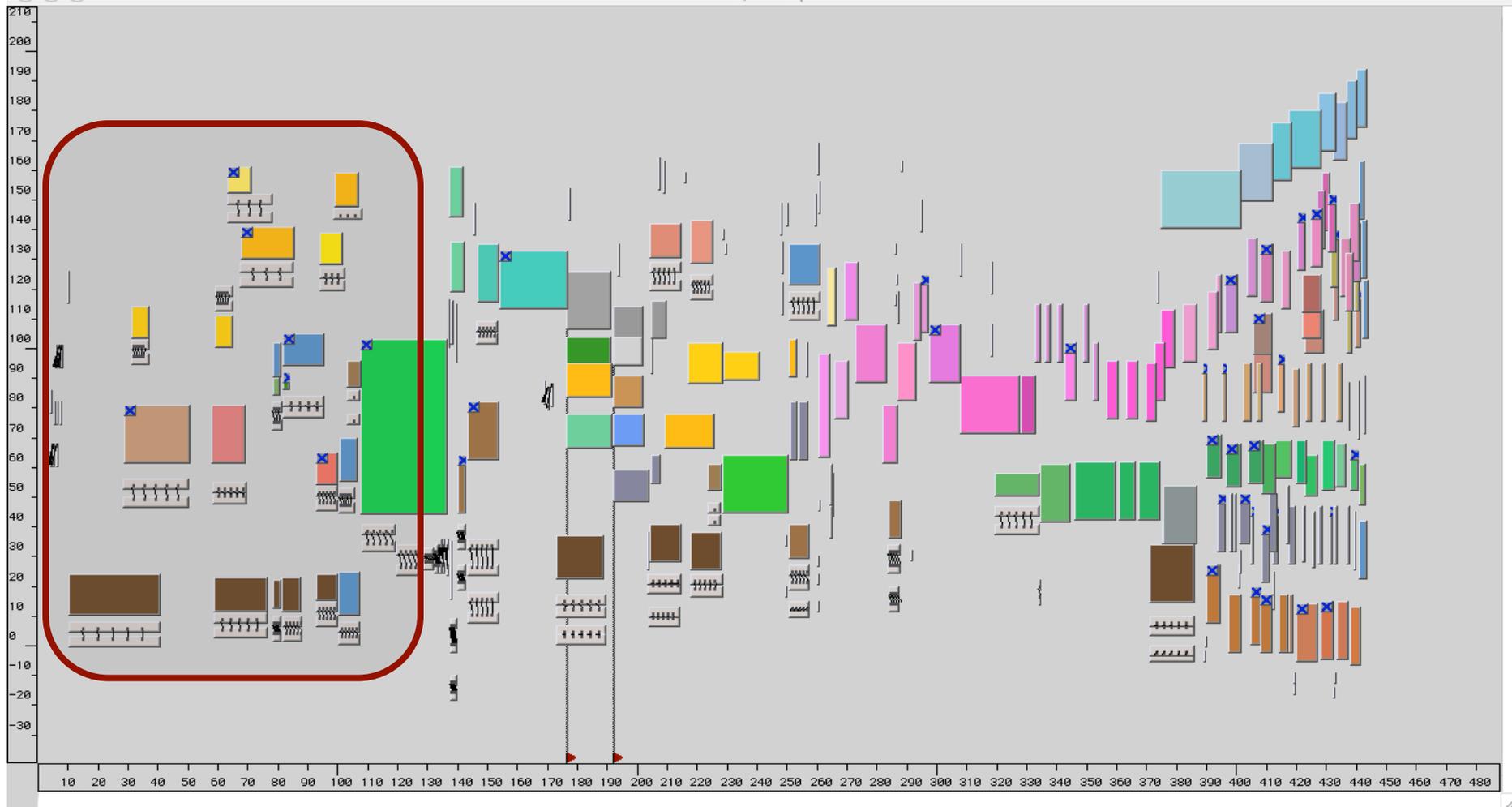
*Canon 4 a 2, per Augmentationem contrario Motu  
Notulis crescentibus crescat Fortuna Regis*

**a 2 Per augmentationem, contrario motu**

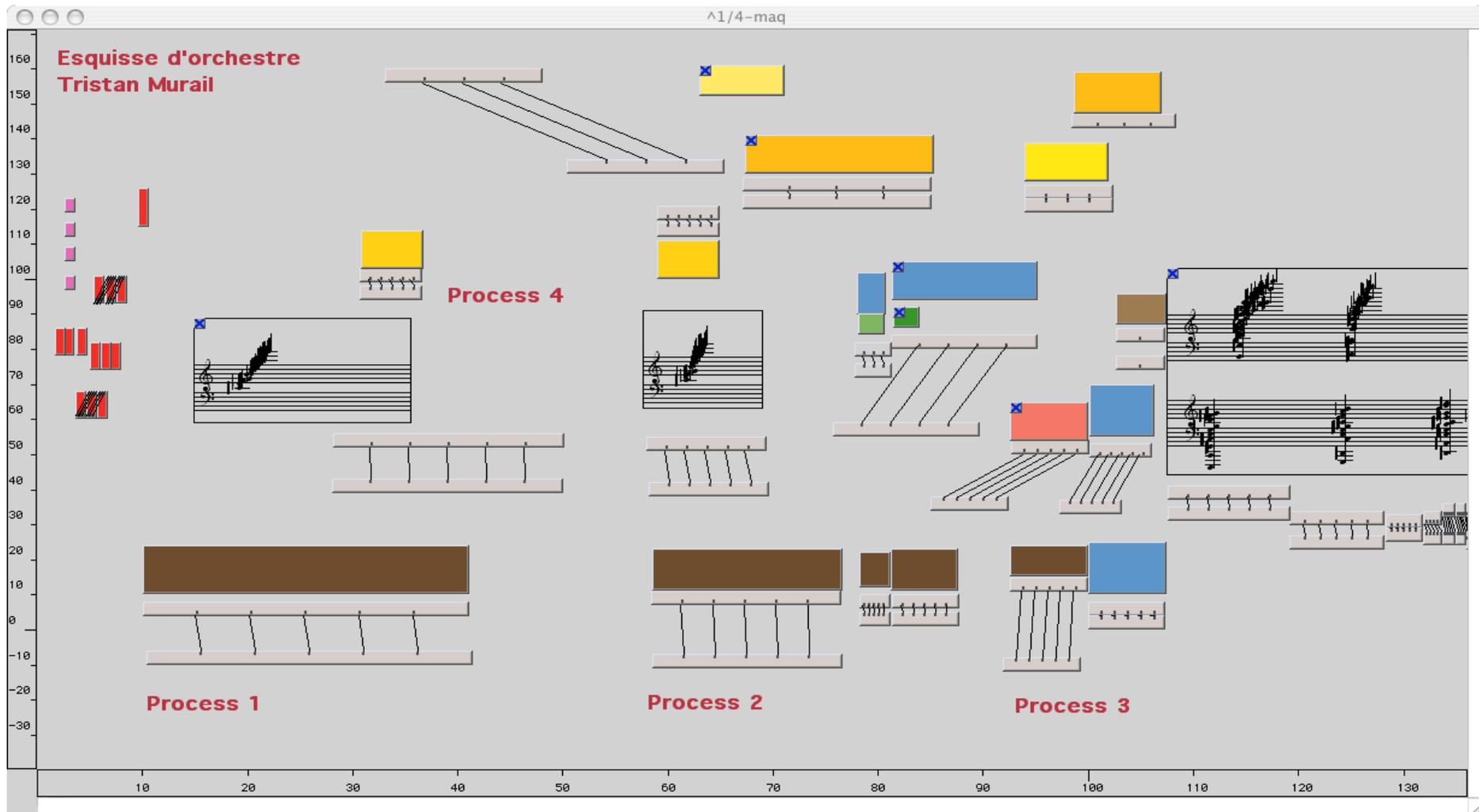
4. Thema

The image displays a musical score for a canon by J.S. Bach. It is titled 'Canon 4 a 2, per Augmentationem, contrario Motu' and includes the Latin subtitle 'Notulis crescentibus crescat Fortuna Regis'. The score is in G minor (two flats) and 3/4 time. It consists of three systems of two staves each. The first system is labeled '4. Thema' and shows the beginning of the canon. The second and third systems show the continuation of the canon with trills (tr) and other musical ornaments. The notation includes various rhythmic values, accidentals, and dynamic markings.





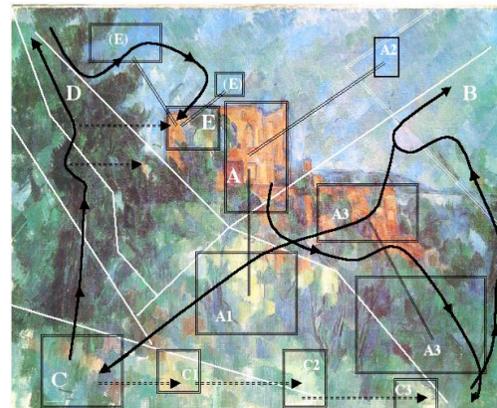
[maquette by T. Murail]



[maquette by T. Murail - zoom]



# H. Parra Strette (2003)



color-to-rhythm

File

LISP car

LISP mapcar

LISP mapcar

LISP mapcar

LISP sixth

LISP seventh

LISP eighth

LISP ninth

LISP tenth

LISP nth

LISP nth

output5 output6 output7 output8 output9 output10 output11

```
((24 88 96)
(215 92 92)
(5 91 74)
(115 45 52)
(19 88 76)
(201 69 77)
(216 92 92)
(21 83 83)
(14 93 80)
(86 50 87)
(82 46 77)
(17 98 82)
(139 65 62)
(205 57 67)
(9 79 66))
```



# J.-L. Hervé Encore (2000)

Extract [1'15'']



The image displays a complex music software interface with several overlapping windows:

- Acadence entière:** A large window showing a piano roll with notes and chords across a vertical axis (0-70) and a horizontal axis (1-3).
- Aphrase 1:** A window showing a piano roll with notes and chords across a vertical axis (0-100) and a horizontal axis (1-7).
- Atrait multi ascyn d32:** A window showing a piano roll with notes and chords across a vertical axis (0-100) and a horizontal axis (0.1-0.3).
- Atempobj:** A window showing a hierarchical tree structure with nodes like 'self', 'ni', 'LISP first', 'LISP second', 'LISP nth', 'list', and 'harmonic-field'. It includes a 'self' icon and a 'put sequence' output.
- tempobj MULTI-SEQ:** A window showing musical notation on a staff with notes and rests. Below the staff, it displays 't: 458 ms' and 'duration: 488 ms'. It also has controls for 'midic' and 'chord'.

Orange arrows indicate data flow: one arrow points from 'Acadence entière' to 'Aphrase 1', and another points from 'Aphrase 1' to 'tempobj MULTI-SEQ'. A red arrow points from 'Atrait multi ascyn d32' to 'Atempobj'.