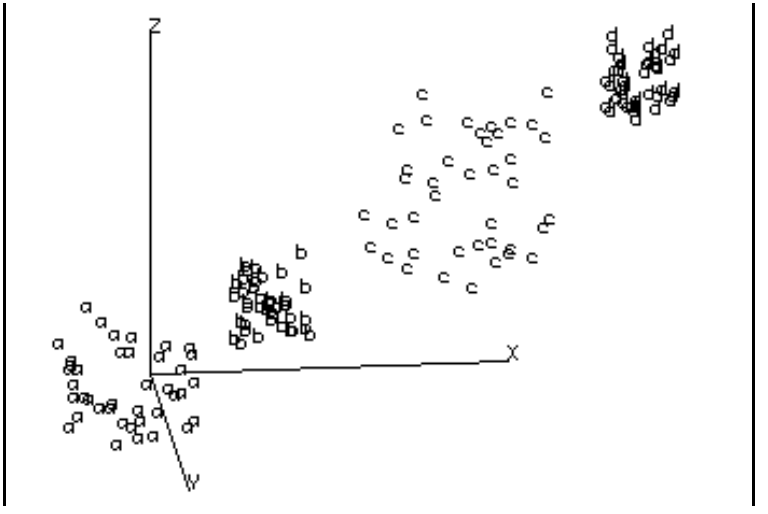
Ircam
documentation

- *Research reports*
- *Musical works*
- *Software*

OpenMusic

Morphologie

Fonctions d'analyse, de reconnaissance, de classification et de reconstitution de séquences symboliques et numériques



Troisième édition, novembre 1999

ircam
Centre
Pompidou

Pour voir la table des matières, cliquez sur le bouton qui se trouve dans le bandeau de la fenêtre Acrobat.
Pour se déplacer d'une page à l'autre, utilisez les boutons de navigation d'Acrobat ou les touches de flèches -> et <- de votre clavier.

Ce manuel a été écrit par François Sarhan, Frédéric Voisin et Jacopo Baboni Schilingi et produit sous la responsabilité éditoriale de Marc Battier - Département de la Valorisation, Ircam.

La librairie Morphologie a été conçue et programmée par Jacopo Baboni Schilingi et Frédéric Voisin.
OpenMusic a été conçu et programmé par Gérard Assayag et Carlos Agon.
Cette documentation correspond à la version 2.1 de la librairie Morphologie, et à la version 2.1 ou supérieure de OpenMusic.

Troisième édition, novembre 1999.

Ircam
1 place Igor-Stravinsky
F-75004 Paris
Tel. 01 44 78 12 33
Fax 01 44 78 15 40
E-mail : ircam-doc@ircam.fr

Groupe d'utilisateurs Ircam

L'utilisation de ce programme et de sa documentation est strictement réservée aux membres des groupes d'utilisateurs de logiciels Ircam. Pour tout renseignement supplémentaire, contactez :

Département de la Valorisation
Ircam
1, Place Stravinsky
F-75004 Paris
France

Courrier électronique: bousac@ircam.fr

Veuillez faire parvenir tout commentaire ou suggestion à :

M. Battier
Département de la Valorisation
Ircam
1, Place Stravinsky
F-75004 Paris
France
Courrier électronique:bam@ircam.fr

Contenu

Présentation

La bibliothèque *Morphologie2* est un ensemble de fonctions et d'algorithmes destinés à l'analyse, la classification, la reconnaissance de formes et à la reconstitution de profils et de séquences tant numériques que symboliques. Des séquences mélodiques, harmoniques ou rythmiques peuvent donc être étudiées selon soit leur dimensions numériques, lorsque cela est possible (hauteur, durée, intensité, etc.), soit selon leur seule structure inhérente lorsque ces séquences possèdent une dimension symbolique. A travers cette bibliothèque, nous tentons aussi un passage du monde numérique— physique — vers celui symbolique —_cognitif_— et inversement.

Si le travail de mise en évidence de classes, catégories, « formes », patterns et structures dans le matériel étudié semble être d'ordre purement musicologique, c'est bien la même attitude qui permet au compositeur de mettre en forme un nombre important de données « brutes » d'origines souvent variées telles que : analyse de spectres (f0, partiels, voisement, harmonicité, etc.), transcriptions, génération de données musicales en CAO, composition.

Ainsi la catégorisation d'une détection de fondamentale, par exemple, permet la quantification, la transcription puis l'analyse d'un chant d'un point de vue tant mélodique que rythmique. Mais les mêmes algorithmes proposés ici permettent aussi la classification d'un corpus musical entier, tel que celui d'une aire géo-culturelle donnée, ou de celui du matériau musical propre qu'un compositeur cherche à ordonner. Tout dépend ici du « codage » utilisé pour représenter les données : ainsi un accord peut être représenté soit du point de vue de ces fréquences fondamentales, soit par le nom des notes qui le constituent, soit encore par un nom arbitraire qui le désigne. *Morphologie* permet d'aborder ces trois types de codage selon le niveau et le point de vue d'où l'on se place. Ainsi la présentation de Jacopo Baboni Schilingi explore *Morphologie* d'un point de vue compositionnel et celui de Frédéric Voisin d'un point de vue musicologique.

Remerciements

Nous tenons à remercier tout particulièrement Gérard Assayag et Carlos Agon Amado de l'équipe Représentations musicales de l'Ircam pour leur aide, leur disponibilité et leurs précieux conseils.

Note sur les versions PatchWork et Open Music de Morphologie

Les utilisateurs de la version Morphologie de PatchWork remarquerons quelques différences dans cette version Open Music.

Certaines fonctions ont en effet été supprimées car l'utilisateur peut les retrouver sous une autre forme dans les autres bibliothèques disponibles dans OpenMusic. Il s'agit des fonctions du menu *Import*, que l'on retrouve dans la bibliothèque *OM-AS*, développée par Hans Tutschku.

La fonction *rbynearest*, que l'on trouvait dans le menu distance dans PatchWork, a été renommée s-class (pour symbole-classification), et déplacée dans le menu classification puisqu'elle met en œuvre un principe d'équivalence fondée sur un seuil de distance et permet donc une classification de chaînes symboliques.

Morphologie et structures musicales

Jacopo Baboni Schilingi

Introduction

Morphologie représente aujourd'hui un travail conjoint entre un compositeur et un ethnomusicologue. Cette intervention essaye d'exposer les raisons qui m'ont poussé à travailler sur le concept des structures appliquées au système génératif en musique, dans le cadre du spectacle « Trois mythologies et un poète aveugle » créé à l'Ircam le 14 novembre 1997.

Contexte

Morphologie a été conçu pour permettre la conversion des données provenant d'un générateur de texte vers un générateur de musique. Un spectacle dans lequel coexistent une génération de texte en temps réel, selon les règles de la grammaire générative et une génération de musique en temps réel, nécessite un contrôle sur les structures pertinentes et sur l'échange des données parmi les deux systèmes.

Morphologie et figure musicale

La figure musicale est porteuse d'informations propres, et définit ce que j'appellerais une sémantique musicale. Ce que je peux affirmer aujourd'hui c'est que la morphologie musicale est à l'origine de ce qui est généralement appelé sémantique musicale.

Je peux distinguer deux attitudes différentes à propos de la morphologie en musique. La première considère la nature d'une entité musicale et la seconde sa matière.

Par le terme nature j'indique la forme globale qu'une entité musicale possède, soit graphiquement, soit sous forme de représentation mentale ; par matière, j'entends la structure plus ou moins abstraite qui constitue une entité musicale.

Dans ce sens, la forme d'une entité est la façon avec laquelle elle se manifeste. La matière est la pure possibilité d'existence de la forme même, et cette dernière est la figure musicale. La forme d'une entité existe seulement quand elle est décomposable et subdivisible en une représentation mentale géométrique. Je dirais même que, souvent, une représentation géométrique est capable de décrire les fonctions structurelles d'une phrase musicale entière. La figure peut exister seulement si elle est segmentable et définissable selon le principe de détermination par lequel une chose se détermine elle

même et a contrario tout ce qui ne lui appartient pas. Ainsi la figure musicale appartient à une dimension discontinue où, à chaque point de discontinuité, réside potentiellement un point de segmentation.

J'entends donc par nature d'une entité, la forme globale avec laquelle elle se manifeste et sa représentation mentale. La **morphologie** musicale est l'étude des configurations et des structures de la nature d'une entité musicale.

Figure musicale

La figure musicale est au sens strict la structure spatiale d'une entité : une gamme descendante a la forme d'une ligne qui descend. Sa nature, bien entendu, est un ensemble de notes qui descendent. Par contre, sa structure est à la fois l'ensemble des notes qui constituent la gamme, l'ensemble des intervalles entre les notes, l'ensemble des directions des intervalles, l'appartenance à des octaves différentes, et l'ensemble des formes primitives de la courbe qui la constitue.

La structure est donc extrêmement articulée, et peut être décrite sur des échelles différentes. La nature d'une gamme musicale qui descend est le fait de descendre. Je peux donc affirmer que la nature d'une entité est son propre devenir au moment précis où elle devient. L'entité existe seulement dans le temps par rapport au temps et à l'espace.

La nature d'une entité est l'expression de sa forme dans le temps. Pour simplifier, il me semble préférable de dire dès maintenant que la nature d'une entité est la figure musicale de l'entité même. L'étude de la morphologie musicale à travers la bibliothèque *Morphologie* est donnée par l'analyse des différents paramètres d'une figure musicale exprimés sous forme de suite symbolique.

Morphologie propose le type d'analyse suivante :

- étude quantitative des patterns
- étude qualitative des patterns
- étude des redondances,
- étude des identités,
- étude des dérivées des paramètres
- étude des comportements primitifs,
- étude des reconstitutions éventuelles,
- étude des distances entre les paramètres,
- étude des classifications automatiques des paramètres.

Je ne prétends pas définir une figure de façon exhaustive, mais plutôt comme un ensemble de champs de possibilités d'analyses. Des analyses qui permettent de mieux nous aider à déterminer la nature d'une identité musicale, de mieux résoudre la problématique liée au principe d'intégration et de dérivation d'une figure musicale.

Je considère la forme et le contenu comme appartenant à la même nature ; de plus, forme ou contenu peuvent être justifiés par la même analyse. En musique, la forme serait le contenu même de la chose exprimée.

Quand la structure devient apparente, pour un temps supérieur à une segmentation immédiate, nous n'avons plus une figure musicale, mais une texture. J'envisage la texture comme quelque chose de nécessairement continu et qui peut présenter des points de segmentation ayant la même pertinence.

Ainsi l'étude de la texture peut être très similaire à l'étude de la figure, tout en sachant que cette dernière est “relativement rapide” et la texture “relativement lente”. En d'autres termes pour que la figure soit, il faut qu'elle puisse être dissociée du contexte.

Si la figure n'est pas dissociable, cela signifie qu'il n'existe que le contexte qui s'apparente alors à la texture. Je crois de plus, que les mêmes différences entre figures simples et figures complexes peuvent être reproduites pour l'étude des textures simples et complexes.

Morphologie essaye de gérer les différents niveaux de segmentation d'une figure ou d'une texture donc de la sémantique musicale. Il existe pourtant une ambiguïté entre forme et structure. La façon la plus honnête de définir la forme est de la considérer comme la limite vers laquelle une structure tend à travers toutes ses analyses possibles.

Si on accepte que la forme soit l'ensemble de toutes les analyses possibles d'une entité musicale, on peut dire que cette forme n'est jamais totalement définie. A l'inverse, une structure peut être décrite de manière exhaustive au sein du système auquel elle se réfère.

Structure

Une structure peut être définie comme un ordre appliqué à un certain nombre d'éléments. Mais cette notion de structure reste sans signification si l'on n'intègre pas le concept de système auquel elle se rapporte. Toute structure acquiert une signification si elle est en relation avec le contexte théorique par lequel le système choisi devient intelligible.

La structure est aussi l'ensemble des phénomènes contingents, c'est à dire que chaque élément dépend des autres et ne peux être sinon « dans » et « par » la relation qu'il a avec les autres éléments dans un système donné. Émerge ainsi le concept de relation entre les éléments d'une structure, en opposition au concept de fonction entre ces mêmes éléments.

En général une fonction est le comportement d'un élément à l'intérieur d'un système donné. Puisque les fonctions d'un élément donné peuvent être exprimées par les caractéristiques du dit élément, cela signifie que la structure est à la base de la description de l'élément. Donc fonction et structure ne sont pas opposées mais complémentaires. Enfin, on peut dire

que la structure est un ensemble d'éléments interdépendants, c'est à dire qu'un changement appliqué à un élément implique une modification de tous les autres éléments de la structure.

Système

En essayant de définir la figure et la structure musicale, j'ai souvent introduit le concept de système qui devient ici la raison même et la possibilité de pouvoir parler d'une entité et de sa propre structure. Parmi les définitions possibles du concept de système, voici celle qui trouve une application directe dans la bibliothèque *Morphologie*.

- Un système possède un certain nombre de classes de repères grâce auquel on peut parler du système en question. Les classes de repères sont les ensembles de définitions intrinsèques et extrinsèques de chaque élément appartenant au système.
- Un système n'existe que s'il est constitué d'éléments.
- Un système est géré par les relations et les interactions des classes de repère des éléments.
- Un système doit être toujours décomposable en classes de repères et exprimable à travers sa structure.

On peut donc toujours mesurer les dimensions typologiques d'un système.

- Les éléments d'un système peuvent être définis d'une façon fonctionnelle, relationnelle voire exhaustive. Quand on définit un élément de manière fonctionnelle, on connaît toutes ses caractéristiques qualitatives et quantitatives. Si l'on change les caractéristiques quantitatives, cela signifie que l'élément est modifiable. A l'inverse, si on modifie le caractère qualitatif cela signifie que l'élément est transformable.

Structure des systèmes

En général un système est un ensemble ordonné apte à produire un résultat. Un système est le produit des interactions des entités qui y résident inscrites dans les limites de l'espace et du temps. Un système gère, sous forme de règles et de lois, les comportements des entités musicales. Les ensembles de ces règles et de ces lois peuvent être infinis. Dans la gestion des paramètres inhérents aux structures musicales, nous sommes obligés de nous confronter à des catégories pertinentes et non-pertinentes, qui agissent directement sur les entités musicales. Une catégorie musicale d'une entité est dite « pertinente » si en modifiant ses paramètres on change la morphologie de l'entité.

Par opposition toutes les catégories qui ne répondent pas à ce principe sont dites « non pertinentes ».

Cette distinction est à la base d'une dichotomie entre ce que l'on conçoit et ce que l'on perçoit. Pour moi, les catégories pertinentes sont les catégories liées à ce que l'on perçoit et les catégories non pertinentes à ce que l'on conçoit. La technique compositionnelle est instaurée dés lors que l'on définie un champs d'existence d'un monde conceptuel et qu'on repro-

duit ce dernier dans espace perceptuel. Dissociées de leur contexte perceptuel, toutes les variations apportées aux catégories pertinentes, c'est-à-dire celles du monde que l'on perçoit, ont, pour moi, le même droit d'existence.

Ainsi, *Morphologie* tente de trouver un équilibre dans les structures des systèmes en opérant sur les catégories pertinentes afin de garantir au compositeur un contrôle artistique assuré par les catégories non pertinentes définies sous forme de structures.

Morphologie

Morphologie représente une application informatique de ce que j'ai exposé précédemment. La première partie de cette bibliothèque étudie différentes façons de définir les structures morphologiques d'une entité musicale. La deuxième partie définit les structures d'une entité selon les principes d'identité et de segmentation. Le chapitre qui traite des reconstitutions permet de générer des applications différentes à partir d'une même analyse structurelle. Le chapitre concernant le concept de distance permet d'exprimer, selon des règles personnelles, la distances qui sépare les structures d'une entité musicale et ses paramètres. Enfin, le chapitre concernant les classifications automatiques tente de définir des catégories variables pour chaque liste d'éléments.

Conclusion

Dans le contexte de la soirée " Trois mythologies et un poète aveugle ", deux générateurs étaient en communication constante pour échanger des données. Un générateur de texte et un générateur de musique étaient prêts à accueillir comme variables des données provenant d'autres systèmes. *Morphologie* est une sorte de traducteur spécifique qui permet de conserver les catégories pertinentes d'un système et de les traduire dans un deuxième système. *Morphologie* permet de définir les invariances d'un système génératif et les variations intrinsèques au système même.

Pour conclure, une simple constatation de caractère esthétique : un système qui gère des catégories pertinentes et non pertinentes assure un contrôle artistique sur n'importe quelle déformation, variation, ou transformation d'une entité musicale. Dans un système compositionnel paramétrique, et qui considère la structure de la composition comme la raison de la composition même, *Morphologie*, représente pour moi un instrument apte à contrôler les structures complexes dans le temps sans jamais porter atteinte à la liberté de choix du compositeur.

Morphologie et l'analyse musicale

Frédéric Voisin

L'idée des fonctions d'analyse de la bibliothèque *Morphologie* est née lors de l'analyse d'un corpus d'environ 300 chants inuit de la côte Est du Groenland, recueilli en 1935-1936 par Paul-Emile .Victor, dans la région d'Ammassalik. Ce travail d'ethnomusicologie s'effectuait dans le cadre du Laboratoire des Langues et Civilisations à Tradition Orale (LACITO) du CNRS.

Je rencontrais, lors de mes analyses, les mêmes problèmes que mon collègue Hervé Rivière qui, lui, analysait des chants des indiens Wayana de Guyane, à savoir l'impossibilité de trouver des structures quelconques par le biais de l'analyse paradigmatique traditionnelle (Ruwet 1972, Nattiez 1975, Arom 1969, 1985). La mise en série de l'ensemble du corpus dans l'analyse, telle que la préconise Nattiez (Nattiez 1995), n'apportait pas plus d'information : pour une personne étrangère à la culture, il était impossible de savoir si tel segment mélodique dégagé par la mise en paradigme était le « prototype » d'une nouvelle classe de segments, ou seulement une variante au sein d'une seule et même classe. De même, il devenait impossible, par ce type d'analyse, de distinguer le matériau mélodique d'un chant de celui des autres chants (mise en série).

Dès lors se posait la problématique des critères de segmentation d'une pièce de musique, l'analyse paradigmatique jouant de façon parfois peu explicite sur plusieurs critères à la fois. « L'air de famille » — l'appartenance à une classe donnée — d'un segment mélodique peut changer selon le critère envisagé. Selon quel critère, ou à partir de quelle « distance » un segment mélodique n'appartient pas à une classe donnée de segments ? Comment évaluer cette distance de façon plus rigoureuse que l'appréhension d'un « air de famille » ? Comment envisager une analyse rigoureuse et exhaustive sur le plan des critères de segmentation ?

L'analyse contrastive et paradigmatique

L'insuccès de l'analyse paradigmatique provenait ici surtout du fait — du moins nous en avions l'intuition, justifiée ou non — qu'elle a une tendance à définir de trop long segments, alors que ces musiques, tant inuit que wayana semblent résulter de la concaténation de micro-segments de trois, quatre ou cinq notes, rarement plus. Certaines observations des wayana eux-mêmes pouvaient être interprétées dans ce sens. De même, les analyses effectuées par Nattiez ou par Pelinski (Pelinski 1985) sur les inuits du Nouveau-Québec et de la Baie d'Hudson montraient l'existence de petits segments auto-

nomes, mais cependant mieux marqués que ceux des chants d'Ammassalik, et insérés — tels des infixes — parmi des segments plus longs.

En élargissant la définition du rythme d'Hervé Rivière (Rivière 1995) — fondée sur les notions de marque, de contraste, ou discontinuités (Thom 1983) — aux critères mélodiques, il est nous est apparu une méthode simple et systématique de segmentation qui se révélait efficace pour l'analyse des mélodies inuit. La procédure est la suivante :

1) relever toutes les marques (critères) possibles dans une dimension donnée (hauteur, rythme, ...);

1b) éventuellement, supprimer les éléments concommitants identiques (du fait que deux éléments identiques ne créent pas de discontinuité) ;

2) segmenter systématiquement la séquence dans chaque dimension et pour chaque critère;

3) faire l'inventaire, critère par critère, des segments différents obtenus;

4) remplacer chaque segment par son « étiquette » arbitraire, afin d'obtenir la structure de la séquence selon chaque critère.

Par la suite, il s'agît de confronter les différentes structures obtenues selon chaque critère et chaque dimension, et d'effectuer la mise en série des analyses des différents chants du corpus. On obtient pour une même séquence un grand nombre de structures possibles dont certaines ne sont pas pertinentes (pas même au niveau « neutre » de la tripartition sémiologique), et ce sur chaque dimension (hauteurs, durées...). Ces structures peuvent parfois concorder, à certains points de segmentation, lesquels définissent alors des points forts de segmentation possible à un niveau supérieur de segmentation multi-critères. Mais il peut arriver qu'il existe un tuilage systématique, ou un recouvrement des segments de chaque dimension, ne permettant pas de définir un point précis d'analyse, mais une zone floue de segmentation correspondant mieux, à mon avis, à la réalité musicale, en tout cas du point de vue de la perception, lorsque celle-ci est globale. A l'inverse, la concommitance de structures superposées peut avoir un certain pouvoir explicatif des procédés compositionnels. L'ambiguïté binaire/ternaire du fameux passage de *Wozzeck* en triolets sur deux hauteurs est parfaitement mise en évidence par ce type d'analyse (si l'on confronte les structures des dimensions hauteurs, durées et intensité).

Du coté d'Ammassalik, une telle analyse permettait non seulement de trouver un « semblant » de structure possible, mais enfin ! de distinguer les matériaux mélodiques propres à chaque chant et leurs structures propres (au moins au niveau « neutre"). Ces chants partagent en effet un grand nombre de micro-segments mélodiques à ambitus restreint, et l'identité de chaque chant semble reposer sur la concaténation de ces micro-segments (combinatoire). Ainsi, les micro-segments :

A : do-sol(b), B : do-ré(mib)-do, C : sol(b)-la(b)-sol(b), D : sol-mi(b)-do [1]

constituent le matériau mélodique de base des trois cent chants recueillis en 1935 dans la région d'Ammassalik (une dizaine d'informateurs de villages différents sont à l'origine de ces chants). D'autre segments, moins récurrents, semblent

1. La tierce do-mib est le plus souvent « neutre » (350 cents) et semble t-il opposée à la tierce majeure (400 c.), voire à la tierce mineure (300 c.). De même, la quinte do-sol est souvent petite, proche de la quarte augmentée (ca. 650 c.) et probablement distinguée de la quinte juste (700 c.).

constituer autant de « signatures » propres à chaque chant et/ou compositeur. Typiquement, les chants sont le plus souvent de la forme : A C D B répété n fois, ou encore, par exemple : B A D B C D B répété (avec variantes) n fois.

Un tel algorithme ayant l'avantage d'être facilement programmable, je me proposais de l'implémenter dans Patchwork afin d'automatiser à la fois le travail d'expérimentation de la méthode et l'analyse des chants d'Ammassalik. Il est vite apparu que cet algorithme et les fonctions qui allaient l'entourer gagnaient à être programmées directement en Lisp (que j'allais donc apprendre par la même occasion) (Cf. les fonctions *structure-1* et *rma-1*).

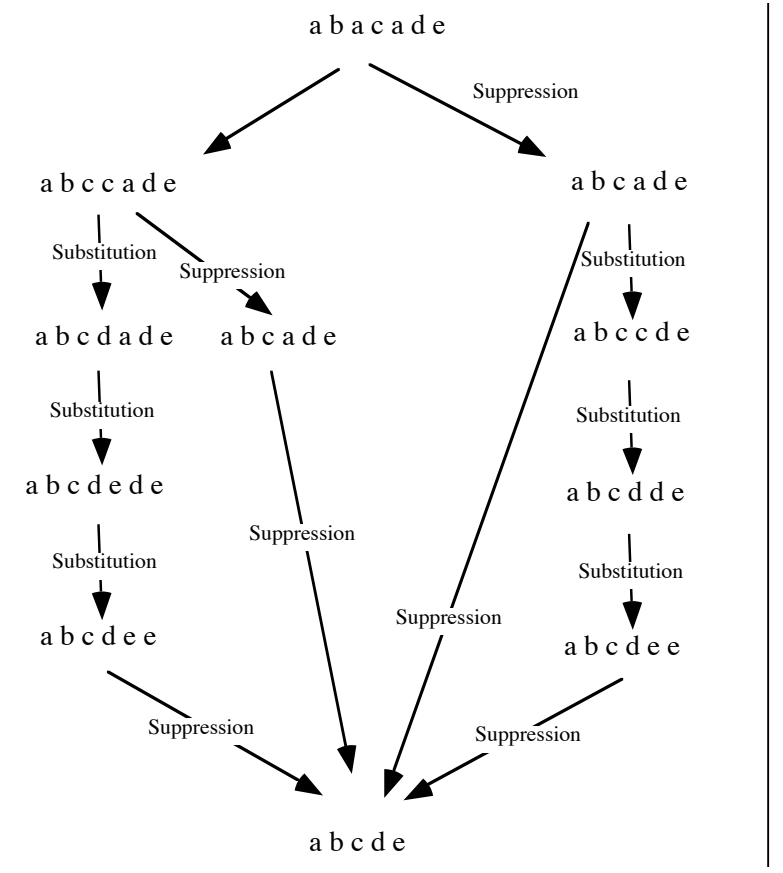
Par opposition, l'analyse paradigmatique semblait maintenant pécher par le manque d'explicitation dans le maniement des critères de segmentation. Le critère principal de segmentation, sinon le seul, dans l'analyse paradigmatique reste la répétition — à l'identique ou avec variation (Ruwet 1972) — qui peut aussi, a priori devenir un algorithme (les procédures que décrit Ruwet n'en sont pas loin). Une analyse de type paradigmatique stricte, pour commencer, c'est à dire un calcul de toutes les combinatoires possibles de tous les segments répétés au moins une fois (à l'identique) dans la séquence (celle-ci « codant » une dimension et une seule), à travers la fonction *structure-2* est une tentative dans ce sens. On peut y observer un grand nombre de structures possibles dès lors qu'on analyse un nombre modéré de segments de longueurs raisonnables. Cette fonction est plus lourde, car l'ensemble des combinaisons possibles est une recherche dans un espace proche du produit cartésien (de dimension exponentielle) des segments trouvés. Cette fonction, si elle reste encore à optimiser, montre quelqu'espoir d'automatiser une analyse paradigmatique, ce d'autant plus si on y intègre une estimation de ressemblance des segments entre eux.

Distances et airs de famille

L'analyse contrastive opérant sur des séquences de symboles, il convient de rester à ce niveau symbolique lors de l'évaluation des distances entre séquences. Un algorithme permettant par exemple de repérer des séquences géniques permet de rester à ce niveau (Milgram 1993), et consiste à rechercher le nombre minimum d'opérations effectuées pour passer d'une chaîne de symboles à une autre. Les trois opérations fondamentales sont :

- *la suppression* d'un élément;
- *l'ajout* d'un élément,
- *la substitution* d'un élément d'une chaîne par un autre.

Ainsi pour passer de la séquence : "a b a c a d e" à celle : "a b c d e", différents « chemins » sont possibles :



La distance retenue est le chemin le plus court, ou plus exactement le chemin de « coût » minimum si l'on donne à chaque opération un coût différent. Ce coût, s'il vaut un pour chaque opération, peut être rapporté à la longueur de la plus petite chaîne, et donc être exprimé en pourcentage de la longueur de la chaîne. Cette distance conserve toutes les propriétés d'une distance réelle, dont en particulier la symétrie et l'inégalité triangulaire.

J'ai ajouté dans les fonctions *distance*, *ldl-distance* un surcoût qui s'exprime lors de l'ajout dans une chaîne d'un élément inexistant dans l'autre chaîne, ce qui peut avoir parfois une pertinence musicale. Dans ce cas, il n'est pas garanti que la distance respecte toujours l'inégalité triangulaire. Il s'agira alors d'une mesure de ressemblance.

Un élément d'une séquence ne doit pas nécessairement être constitué d'une seule lettre. Un accord peut être codé — représenté — par un mot de plusieurs lettres, permettant ainsi d'analyser des séquences où alternent notes isolées et accords. On trouvera dans les utilitaires de la bibliothèque *Morphologie* des outils permettant différentes formes de codage des données musicales.

Appliquée à un corpus de séquences, formules mélodiques, hauteurs, la fonction de distance donne tous les éléments pour une classification par proximité des objets du corpus. On peut par exemple calculer l'arbre de longueur minimale réunissant, selon la théorie des graphes, l'ensemble des points (séquences, formules, chaînes) en fonction de leur distances respectives (Cf. la fonction de la bibliothèque *prim-tree*).Par exemple, les huit séquences suivantes :

```
"a b a c a d e" "a b c d e" "a b a b c" "a b c e" "d e d e" "a g a g f g"
    "a b d a c d e" "a f a f a c f e"
```

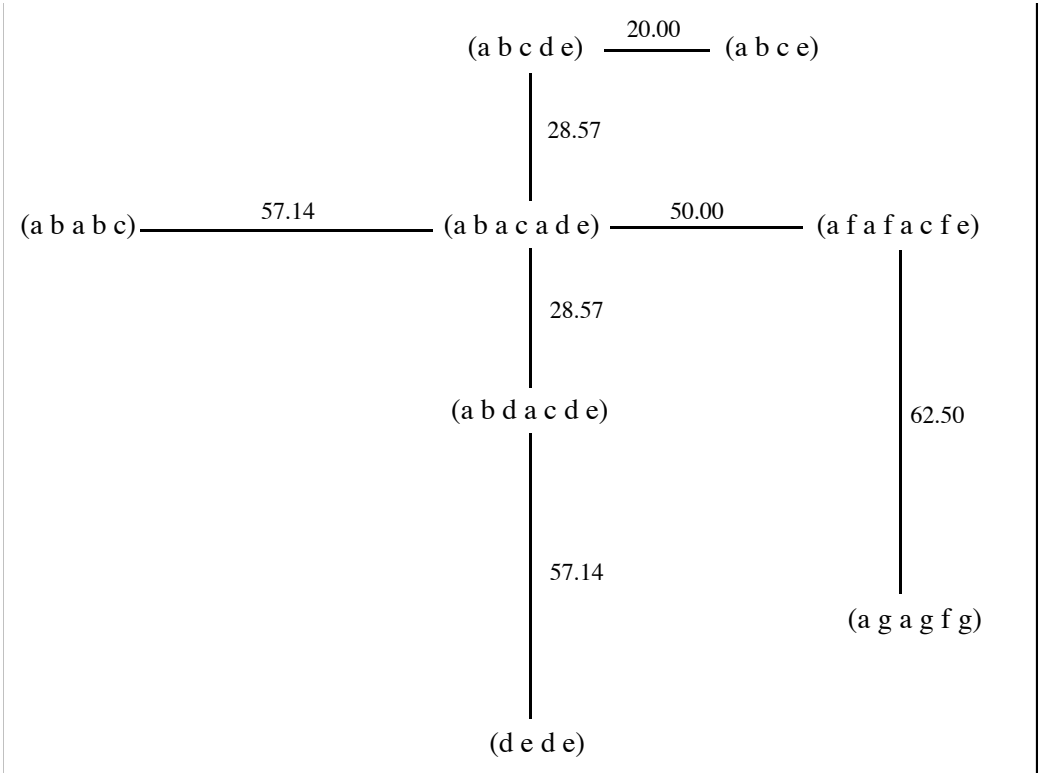
donnent les distances suivantes (exprimées en %) :

```
(a b a c a d e) / (a b c d e) --> 28.571
(a b a c a d e) / (a b a b c) --> 57.143
(a b a c a d e) / (a b c e) --> 42.857
(a b a c a d e) / (d e d e) --> 71.429
(a b a c a d e) / (a g a g f g) --> 71.429
(a b a c a d e) / (a b d a c d e) --> 28.571
(a b a c a d e) / (a f a f a c f e) --> 50.000
(a b c d e) / (a b a b c) --> 60.000
(a b c d e) / (a b c e) --> 20.000
(a b c d e) / (d e d e) --> 60.000
(a b c d e) / (a g a g f g) --> 83.333
(a b c d e) / (a b d a c d e) --> 28.571
(a b c d e) / (a f a f a c f e) --> 62.500
(a b a b c) / (a b c e) --> 60.000
(a b a b c) / (d e d e) --> 100.000
(a b a b c) / (a g a g f g) --> 66.667
(a b a b c) / (a b d a c d e) --> 57.143
(a b a b c) / (a f a f a c f e) --> 62.500
(a b c e) / (d e d e) --> 75.000
(a b c e) / (a g a g f g) --> 83.333
(a b c e) / (a b d a c d e) --> 42.857
(a b c e) / (a f a f a c f e) --> 62.500
(d e d e) / (a g a g f g) --> 100.000
(d e d e) / (a b d a c d e) --> 57.143
(d e d e) / (a f a f a c f e) --> 87.500
(a g a g f g) / (a b d a c d e) --> 71.429
(a g a g f g) / (a f a f a c f e) --> 62.500
(a b d a c d e) / (a f a f a c f e) --> 50.000
```

dont on déduit l'arbre de longueur minimale :

```
(a b a c a d e) (a b c d e) 28.57
(a b c d e) (a b c e) 20.0
(a b a c a d e) (a b d a c d e) 28.57
(a b a c a d e) (a f a f a c f e) 50.0
(a b a c a d e) (a b a b c) 57.14
(d e d e) (a b d a c d e) 57.14
(a g a g f g) (a f a f a c f e) 62.5
```

lequel se construit graphiquement :



On peut très bien ainsi remplacer dans une structure donnée un élément (segment) par celui le plus proche dans l'arbre de longueur minimum, simplifiant ainsi la structure finale. Ce autant pour les fonctions d'analyse contrastive (Cf. fonction *s-class*) que pour celle d'analyse paradigmatique où la distance permet de contrôler et quantifier les paradigmes de segments proches.

Classification et quantification

Lorsque les coordonnées d'un ensemble d'objets sont connues (par exemple sur une dimension, celle des durées, ou dans le plan hauteurs-durées ou encore dans un espace hauteur-temps-intensité), on peut calculer les distances euclidiennes séparant ces objets et ensuite les classer selon différents algorithmes de coalescence. De tels algorithmes, tel que celui des « nuées dynamiques » (Duday et al. 1985) utilisé dans class-1, permettent de traiter dans un temps raisonnable un grand nombre de points. Il peut être autant utilisé pour discrétiser un suivi de F0 en classes de hauteurs, que pour quantifier des durées, ou bien, en combinant plusieurs paramètres tels que durées, hauteurs, intensité et timbre, des classes rythmiques. Nous avons pu remarquer que les classes rythmiques ainsi définies pour les parties de tambour des chants d'Ammassalik, dont le rythme est le plus souvent non mesuré, ou au tempo très variable, sont d'autant mieux définies que l'on traite plusieurs dimensions — voire même des corrélations (en prenant par exemple comme dimension la durée de la note qui précède, et/ou qui suit) lorsque le nombre de dimensions physiques distinctes est petit (analyse par exemple des seuls marqueurs rythmiques issus .d'AudioSculpt).

La fonction class-1 détermine des classes en recherchant des minimums locaux de distances réciproques. L'initialisation de l'algorithme est en général aléatoire (mais on peut définir soi-même, de façon très approximative, le centre de chaque classe), et dans ce cas, il peut arriver que deux classifications successives diffèrent. Cela arrive d'autant plus que les classes sont physiquement peu séparées. On gagnera dans ce cas à faire plusieurs classification, et observer les points qui ne changent jamais de classes, ceux-ci définissant alors des formes « fortes » et stables.

En guise de conclusion

Une bonne part de la partie analytique de la bibliothèque étant issue de recherches ethnomusicologiques en cours, je suis heureux de voir que le lien avec l'approche compositionnelle, telle que celle de Jacopo Baboni Schilingi, est suffisamment fort pour que la réunion de nos codes respectifs (développés séparément) puisse constituer si rapidement une bibliothèque commune. Pour ma part, s'il reste encore à développer et optimiser encore certaines fonctions d'analyse et de représentation graphique, j'avoue que la partie reconstitution semble déjà très séduisante pour l'ethnomusicologie.

Structure de la bibliothèque Morphologie

Remarque

Les différents types de données des objets de *Morphologie* sont :

- des listes (list);
- des entiers (integer);
- des entiers ou flottants (number);
- des symboles — nombres ou caractères — (atom);
- des “strings“ — suite de caractères entre “ “ — (string);
- des matrices en format Common-Lisp (matrix);
- des menus d’options (cf. documentation OpenMusic).

D’autre part, les objets possédant des entrées optionnelles (cf. documentation de OpenMusic), sont présentés avec deux icônes; la première représente l’objet avec ses entrées par défaut, la seconde le représente avec toutes les entrées optionnelles.

ptrn-find

Syntax

(morph2 : :ptrn-find *list*)

input

list liste de nombres ou de symboles

Output

list

Description

Donne tous les patterns de longueur <n> présents dans <list> avec leur nombre d'occurences.

Est considéré comme pattern tout segment de <list> répété au moins une fois.

Si <n> est nul (nil), donne tous les segments répétés quelles que soient leurs longueurs ;

<n> peut être une liste de longueurs souhaitées.

ptrn-smooth

Syntax

(morph2 : :ptrn-smooth *list*)

Input

list list, liste de nombres ou de symboles

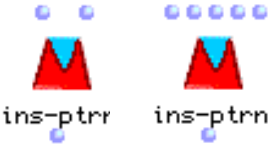
Output

list

Description

Supprime les répétitions locales (éléments contigus identiques) dans une liste de nombres et/ou symboles.

ins-ptrn



Syntax

(morph2 : :ins-ptrn *seq ptrn prof set marg*)

Inputs

seq	list, une liste de nombres ou de symboles
ptrn	list, pattern (segment) dont les positions sont recherchées
prof	integer, nombre d'éléments maximum insérés dans le segment
set	list (optionnel), ensemble des éléments permis commeinsertion dans le segment
marg	menu options (optionnel) (+ - +-) Si l'entrée seq est une liste de valeurs numériques, l'entrée marg défini une marge autour des valeurs données dans set : plus ou moins les valeurs set (+-); seulement plus les valeurs de set (+); seulement moins les valeurs de set (-).

Output

list

Description

Donne les positions, dans la liste d'entrée seq, d'un segment ptrn dans lequel sont inséré des éléments.

min-flex-max

Syntax

(morph2 : :min-flex-max *list which prof chois d-flx*)

Inputs

list	list, une liste de valeurs numériques
which	menu options, (prim, prof, vals, every) Mode de transcription :
prim	donne seulement la succession des primitives;
prof	succession des primitives et la « profondeur » en nombre d'élément de la liste d'entrée, de chaque primitive;
vals	idem. prof et la valeur correspondant à chaque primitive.
every	idem. vals mais ajoute la position, dans la liste d'entrée, de chaque primitive.
prof	integer (optionnel), valeur définissant la profondeur minimale des primitives recherchées
chois	list (optionnel), valeur ou liste de valeurs déterminant la profondeur des primitives recherchées

Output

list

Description

Donne une transcription (analyse) d'une séquence numérique en termes de succession de primitives. On définit trois primitives fondamentales :

minima pour une fonction décroissante-croisssante (col);

maxima pour une fonction croissante-décroissante (sommet);

flex pour une fonction stable (maxima ou minima local).

direct-analysis

Syntax

(morph2 : :direct-analysis *list*)

Inputs

list list, une liste de valeurs numériques

Output

list

Description

Donne le signe de la dérivée locale (-1, +1, 0) pour chaque point de la liste des valeurs données en entrée.

find-permut

Syntax

(morph2 : :find-permut *seq output &optional length ptrn*)

Inputs

list	séquence d'éléments quelconques.
menu	
permutation	renvoie les segments d'éléments permutés
position	renvoie les segments d'éléments et leurs positions dans la séquence.
optionnels	
length	longueur des segments. 2 par defaut.
ptrn	segment dont les permutations sont recherchés. tous par défaut. desactive <length>.

Output

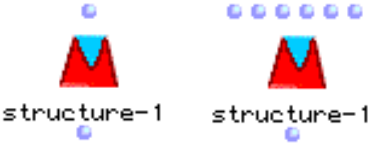
list

Description

Renvoie les permutations de deux éléments de la séquence <seq>. On peut choisir : les segments seulement (mode <permutation>), les segments avec leurs positions dans le séquence (mode <position>), la longueur des segments (optionnel), le segment dont on cherche les permutations. (optionnel 2).

2 Structure

structure-1



Syntax

`(morph2 : :structure-1 seq &optional alpha? smooth result levels smoo2)`

Inputs

- seq — list, une liste de nombres ou de symboles séparés par un espace.

optionnels :

- alpha? — menu options (alpha / num) sortie alpha-numérique ou numérique seulement (numérique :1, alpha : a, avec 1 -> a, 2-> b).
- smooth — menu options (yes / no), supprime de la liste seq les éléments contigus identiques (répétitions immédiates).
  (Dans la mesure où l'analyse consiste à repérer des contrastes et que deux éléments contigus identiques ne font pas de contraste, il est pertinent de supprimer toutes répétitions immédiates).
- result — menu déroulant, 4 options (struct / short / extended / save);
  mode struct : donne toutes les structures obtenues à chaque niveau d'analyse sous forme de liste de liste.
  mode short : donne une liste avec la structure suivante :
  ((critères_de_segmentation)
  ((segments_selon_critères)...)
  ((formes_résultantes)))
  mode extend : donne une description détaillée et explicite de l'analyse contrastive de la séquence : critère, forme résultante et segments repérés
  mode save : écrit dans un fichier texte le résultat de l'analyse détaillée.
- levels — integer, nombre de niveaux d'analyse désirés.
- smoo2 — menu options (yes no), supprime dans les niveaux supérieurs d'analyse les éléments contigus identiques (répétitions immédiates). Il est souvent moins pertinent de supprimer les répétitions au niveau des structures obtenues.

Output

list

Description

Effectue une analyse contrastive d'une séquence quelconque.Le principe de l'analyse contrastive (développée par Hervé Rivière et Frédéric Voisin au LACITO-CNRS pour l'analyse des chants inuit et wayana) consiste à segmenter une séquence donnée à chaque occurrence de chacun des éléments propres de la séquence et à identifier et nommer les segments ainsi définis. Le critère de segmentation étant l'élément lui-même, on obtient autant de segmentations que d'éléments différents présents dans la séquence. La forme résultante est donc une concaténation de segments commençant par le même élément.

Exemple de l'analyse de la séquence (a b a c a b c b c d a b) avec le résultat en mode short :

```
((critères_de_segmentation)
((segments_selon_critères)...)
((formes_résultantes)))
((c d a b) (((1 (a b a)) (2 (c a b)) (3 (c b)) (4 (c d a b))) ((1 (a b a c a b
    c b c)) (2 (d a b))) ((1 (a c)) (2 (a b c b c d)) (3 (a b))) ((1 (a)) (2 (b
    a c a)) (3 (b c)) (4 (b c d a)) (5 (b)))) ((1 2 3 4) (1 2) (3 1 2 3) (1 2 3
    4 5)))
```

qu'il faut lire :

- critères de segmentation (liste dont la longueur est égale aux nombre total d'éléments différents dans la séquence) :

```
 (c d a b)
```

- segments dégagés selon ces critères : (le chiffre qui précède les segments est une numérotation de ces segments)

```
(1 (a b a)) (2 (c a b)) (3 (c b)) (4 (c d a b)))
((1 (a b a c a b c b c)) (2 (d a b)))
((1 (a c)) (2 (a b c b c d)) (3 (a b)))
((1 (a)) (2 (b a c a)) (3 (b c)) (4 (b c d a)) (5 (b))))
```

- formes résultantes :

```
((1 2 3 4) (1 2) (3 1 2 3) (1 2 3 4 5)))
```

Exemple de l'analyse de la séquence (a b c d a b) avec le résultat en mode extend :

donne tous les patterns commençant par un élément différent dans la séquence , cet élément étant appelé mark :

```
mark /c/      STRUCTURE 1.1  : a b
2 patterns  :
  a - (a b)
  b - (c d a b)


 mark /d/      STRUCTURE 1.2  : a b
2 patterns  :
  a - (a b c)
  b - (d a b)
```

```
 mark /a/      STRUCTURE 1.3  : a b
2 patterns  :
  a - (a b c d)
  b - (a b)

 mark /b/      STRUCTURE 1.4  : a b c
3 patterns  :
  a - (a)
  b - (b c d a)
  c - (b)
```

L'entrée optionnelle <levels> permet de choisir combien de fois cette analyse sera effectuée : 1 elle s'appliquera à la séquence de départ,

2 elle s'appliquera à la séquence de départ, et successivement à toutes les structures dégagées lors de la première analyse,

3, à la séquence de départ, aux structures dégagées lors de la première analyse, et aux structures dégagées par la deuxième.

Attention : le nombre de structures obtenues à chaque niveau d'analyse est donc une fonction exponentielle du nombre de segments différents repérés dans la séquence.

ex : (même séquence de départ que précédemment)

```
********** ANALYSIS LEVEL #2 **********

2 structures found on structure 1.1 :


 mark /A/      STRUCTURE 2.1  : A
1 pattern  :
  A - a b

 mark /B/      STRUCTURE 2.2  : A B
2 patterns  :
  A - a
  B - b

2 structures found on structure 1.2  :


 mark /A/      STRUCTURE 2.3  : A
1 pattern  :
  A - a b


 mark /B/      STRUCTURE 2.4  : A B
2 patterns  :
  A - a
  B - b

2 structures found on structure 1.3  :
```

```
 mark /A/      STRUCTURE 2.5  : A
1 pattern  :
  A - a b

 mark /B/      STRUCTURE 2.6  : A B
2 patterns  :
  A - a
  B - b

3 structures found on structure 1.4  :


 mark /A/      STRUCTURE 2.7  : A
1 pattern  :
  A - a b c

 mark /B/      STRUCTURE 2.8  : A B
2 patterns  :
  A - a
  B - b c

 mark /C/      STRUCTURE 2.9  : A B
2 patterns  :
  A - a b
  B - c
```

En fin d'analyse, donne le score (nombre de représentation) de chaque structure au différents niveaux :

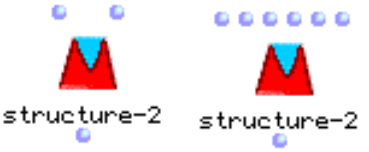
RMA signifie : Recursive Mark Analysis

```
********** SCORES **********

Scores at RMA level 1  :
for structure 1.1 :
a = 1
b = 1
for structure 1.2 :
a = 1
b = 1
for structure 1.3 :
a = 1
b = 1
for structure 1.4 :
a = 1
b = 1
c = 1

Scores at RMA level 2  :
```

```
A B = 5
A = 4
```

Remarque : la structure la plus pertinente n'est pas nécessairement celle obtenant le meilleur score. Le plus souvent, ce sont les structures possédant le moins d'éléments mais le plus de répétitions (AA, AABBAABB) qui semblent les plus pertinentes. Ainsi les structures de type "a b c", "A B C D ..." sont souvent les moins intéressantes, parce qu'elles ne possèdent pas de propriétés (symétries, répétitions, périodicités, etc.).

structure-2



Syntax

`(morph2 : :structure-2 seq n-max alpha? result length  seuil)`

Inputs

seq	list, une liste de nombres ou de symboles séparés par un espace.
n-max	integer, nombre maximum de patterns formant les structures recherchées (permet de limiter le nombre de combinaisons possibles).
alpha?	menu options (alpha / num), donne les formes sous forme alpha-numérique ou sous forme numérique seulement (1 -> a, 2-> b,..., 0 -> °).
result	menu options (extended / pos / mat / p-score / save)
mode extended	donne l'analyse détaillée, patterns trouvés, formes résultantes, taux de complétion.
mode save	écrit dans un fichier texte l'analyse détaillée.
mode pos	donne, pour chaque pattern trouvé, ses positions (depuis 0) dans la séquence.
mode p-score	donne le taux de complétion de chaque structure trouvée.
mode mat	dans une liste de listes, donne la liste des patterns trouvés, et la matrice binaire de complétion des patterns dans la séquence.
optionnels	
length	list (optionnel), liste du nombre minimum et maximum d'éléments pour les patterns recherchés (permet de limiter le nombre de combinaisons possibles).
seuil	integer (optionnel), seuil de complétion, de 0 à 100%, minimal. Seules les structures qui remplissent de plus de ce seuil la séquence originale sont calculées.

Output

list

Description

Effectue l'analyse d'une séquence quelconque selon le critère de la répétition. Il s'agît ici d'une première tentative d'implémentation d'analyse paradigmatique (cf. Ruwet 1972, Nattiez 1975, 1995, Arom 1969, 1985) automatisée.

Sera considéré comme « pattern » toute partie de la séquence répétée au moins une fois. La fonction recherche ensuite toutes les structures possibles résultant de la combinatoire des différents « patterns » trouvés dans la séquence.

Du fait du nombre parfois très élevé de structures possibles à explorer, la fonction peut aboutir à une erreur « Out of memory".

3 Reconstitute

1-0-1-reconst

Syntax

(morph2 : :1-0-1-reconst *list*)

Inputs

list list, une liste signe de dérivées (-1, +1, 0).

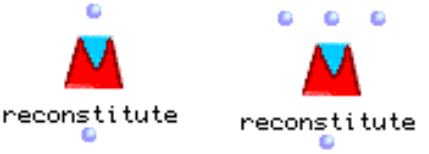
Output

list

Description

Reconstitue une suite de valeurs depuis l'analyse obtenue par l'objet direct-analysis, ou du signe des dérivées locales.

reconstitute



Syntax

```
(morph2 : :reconstitute list which start )
```

Inputs

list	list, une liste de primitives, ou (primitive profondeur), ou (primitive profondeur valeur_du_centre_de_la_prim), ou (primitive profondeur valeur_du_centre_de_la_prim position).
which	menu options (prim / prof / vals / every), Détermine les paramètres pris en compte pour la reconstitution de la série numérique.
start	integer, valeur de départ de la suite à reconstituer.

Output

list

Description

Reconstitue une suite de valeurs depuis des analyses obtenues par l'objet min-flex-max, ou une liste de primitives (min flex max).

4 Distance

euclidian-d

Syntax

(morph2 : :euclidian-d *matrix*)

Inputs

matrix matrix, matrice des coordonnées des points dans un espace.

On peut utiliser l'objet l-matrix dans le menu Classification pour convertir une liste de listes sous forme matricielle. La structure de la liste doit être : ((xi, yi, zi, ...)(xi+1, yi+1, zi+1,...)...) ou bien sous forme matricielle en Lisp : #2A((xi, yi, zi,...) (xi+1, yi+1, zi+1,...)...).

Output

list

Description

Donne la demi-matrice supérieure des distances euclidiennes entre des points dans un espace orthonormé à n-dimensions.

distance

Syntax

(morph2 : :distance *seq1 seq2 change ins/sup scale inex*)

Inputs

seq1	list, une liste de symboles séparés par un espace.
seq2	list, idem.
change	number, coût de substitution d'un élément (en général entre 0 et 1).
ins/sup	number, coût d'insertion ou suppression d'un élément (en général entre 0 et 1).
scale	menu options (relat / absol), échelle du résultat, soit le coût rapporté, en pourcentage, à la longueur de la séquence la plus petite. Soit la distance minimum absolue (coût minimum d'édition).
inex	number, surcoût de substitution ou d'insertion d'un élément étranger à l'autre séquence, ajouté au coût ins/sup.

Output

list

Description

Calcule la « distance d'édition » entre deux listes de symboles (algorithme utilisé parfois pour le séquencement des gènes). Il s'agit de la distance minimale entre deux séquences en termes de coûts : coût de substitution d'un élément d'une séquence par l'élément correspondant de l'autre séquence, et coût de suppression ou d'ajout d'un élément dans l'une ou l'autre séquence. On cherche donc le nombre minimum d'opérations de substitution, d'ajout ou de suppression à effectuer pour « passer » d'une séquence vers l'autre.

ldl-distance

Syntax

`(morph2 : :ldl-distance` *`l-seq change ins/sup scale inex result`*`)`

Inputs

l-seq	list, une liste de liste de symboles.
change	number, coût de substitution d'un élément.
ins/sup	number, coût d'insertion ou suppression d'un élément.
scale	menu options (relat / absol), cf. objet distance.
inex	number, cf. objet distance.
result	menu options (short / extended / save)
mode short	donne une liste de liste correspondant à la demi-matrice supérieure des distances, sous la forme suivante : ((seq$_i$ seq$_{i+1}$ distance$_{i<->i+1}$) ...)
mode extended	description détaillée à l'écran des séquences et de leurs distances réciproques.
mode save	écrit un fichier texte du mode extended.

Output

list, print out, text file.

Description

Applique la distance d'édition à une liste de séquences (cf. distance).

multi-distance

Syntax

(morph2 : :multi-distance *seq1 seq2 change ins/sup wgth inex*)

Inputs

seq1	list, une liste de liste de nombres ou symboles.
seq2	list, une liste de liste de nombres ou symboles.
change	number, cf. objet distance.
ins/sup	number, cf. objet distance.
wgth	list, liste des poids pour chaque élément des listes. Le poids est nul par défaut si le nombre de poids ne correspond pas au nombre d'éléments dans les listes. Inversement, les listes ne doivent pas avoir nécessairement la même longueur; dans ce cas seuls les N-premiers-éléments communs à toutes les listes seront pondérés, les autres ignorés (poids nul).
inex	number, cf. objet distance.

Output

list

Description

Applique la distance d'édition à une liste de séquences avec une pondération (entre 0 et 1) pour chaque élément (ou position) des listes. Ainsi, avec un poids de 0 sur le énième élément des séquences, cet élément sera ignoré dans le calcul des distances. Ainsi, chaque séquence (liste) peut, par exemple, représenter la description d'un objet quelconque en n dimensions. La pondération permet alors de pondérer chaque dimension dans le calcul des distances entre les objets ainsi représentés.

Par exemple, les séquences ((1 l 5 4) (a 1 m 4)), et ((1 2 5 4) (a 2 m 4)) auront une distance de 50% avec un poids de 1 pour chaque élément (1 1 1 1), mais une distance de 0% (nulle) avec un poids nul sur le deuxième élément de chaque liste (1 0 1 1).

Typiquement on calculera les distances entre différentes analyses issues de l'objet min-flex-max en pondérant les différents critères d'analyse.

resemblance

Syntax

`(morph2 : :resemblance` *`a b wocc wref &optional diff`*`)`

Inputs

a	list, une liste ou une liste de listes de symboles.
b	list,une liste de symboles.
Wocc :	flottant, poids de la structure d'occurence (Weight of occurence)
Wref :	flottant, poids de la structure de répétition (Weight of repetition)
wgth	list, liste des poids pour chaque élément des listes.
optionnel	
diss / res	menu déroulant : donne la dissemblance ou la ressemblance

Output

flottant de 0 à 100.

Description

Calcule une valeur de ressemblance entre 0 et 100 (pourcentage) entre deux séquences de symboles selon le critère de leur structure interne, plus précisément selon la distribution des redondances / occurences dans chacune des séquences.

Si une seule liste de liste est en entrée a, la valeur de ressemblance sera calculée entre les listes de cette liste de liste. Le format de sortie sera : ((sous-liste1)(sous-liste2) distance)(sous-liste3)(sous-liste4) distance)...), c'est-à-dire typiquement l'entrée de prim-tree.

L'entrée optionnelle permet de calculer soit le taux de ressemblance ou au contraire le taux de dissemblance (soit une sorte de distance).

prim-tree

Syntax

(morph2 : :prim-tree *distance*)

Inputs

distance — list, une liste de liste exhaustive des distances réciproques entres les éléments dont on recherche l'arbre minimum, selon la structure suivante : ((ei ei+1 dist (ei ei+1) ...).

Output

list

Description

Donne une liste permettant de construire l'arbre de longueur minimal (algorithme de Prim) entre des éléments en fonction de leurs distances respectives.

Exemple :

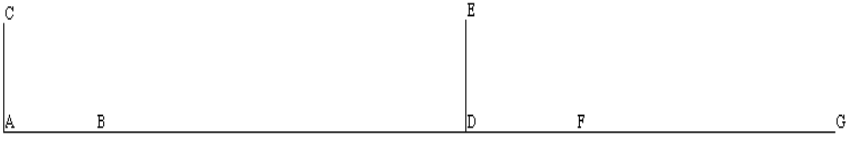
Soit les distances suivantes entre les points a b c d e f g :

```
((a b 0.5) (a c 0.4) (a d 2.51) (a e 3) (a f 3.3) (a g 4.4) (b c 0.55) (b d 2)
    (b e 2.5) (b f 2.81) (b g 4) (c d 2.3) (c e 2.8) (c f 3.2) (c g 4.2) (d e
    0.41) (d f 0.6) (d g 4.5) (e f 0.7) (e g 1.5) (f g 1.4))
```

prim-tree donne :

```
((a c 0.4) (a b 0.5) (b d 2) (d e 0.41) (d f 0.6) (f g 1.4))
```

permettant de construire l'arbre suivant (en repérant les éléments communs à plusieurs listes qui représentent les nœuds) :



tree-path

Syntax

(morph2 : :tree-path *tree start end*)

Inputs

tree	arbre (liste de listes).
start	atome ou liste d'atome
end	atome ou liste d'atome

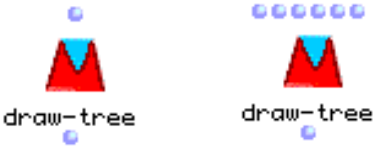
Output

liste

Description

Trouve tous les chemins dans prim-tree de <tree> depuis <start> à <end>.
<start> et <end> peuvent être des atomes (nombre, symbole ou strings selon l'arbre)
ou une liste d'atomes.
Si aucun <start> et/ou <fin> n'esty specifié, tree-path retourne toutes les solutions possibles (paths).

draw-tree



Syntax

(morph2 : :draw-tree *tree*)

Inputs

tree list, une liste correspondant au parcours d'un arbre minimum tel que le crée la fonction Primtree.

Output

graphic window.

Description

Ouvre une nouvelle fenêtre avec une représentation graphique de l'arbre minimum construit par la fonction Prim-tree. Se connecte donc seulement à la sortie de l'objet Prim-tree.

5 Classification

s-class

Syntax

(morph2 : :s-class *seq dist thresh*)

Inputs

seq	list, une liste de nombres ou de symboles.
dist	list, une liste de liste des distances correspondant à la demi-matrice des distances réciproques, sous la forme ((ei ei+1 dist(ei ei+1) ...). Les distances ne doivent pas nécessairement être présentées dans l'ordre de la matrice, mais toutes les distances doivent être représentées.
thresh	number, seuil de distance en dessous duquel les éléments seront substitué par l'élément du sommet le plus proche dans l'arbre minimum.

Output

list

Description

Remplace les éléments d'une liste par les éléments les plus proches en fonction de leur proximité dans l'arbre de longueur minimum (ce dernier étant obtenu depuis la demie matrice des distances entre ces éléments) et un seuil de distance. Cette fonction, sur la base de l'analyse effectuée par l'algorithme Prim-tree, permet une quantification de séquences symboliques par proximité : deux ou plusieurs séquences distantes de moins du seuil donné seront considérées équivalentes; elles seront donc représentées par seulement l'une d'elles au sein de la séquence seq.

La « quantification » des séquences de symboles est donc sensible aux arguments donnés à l'algorithme calculant les distances entre les séquences (*distance ou ldl-distance)*. Ainsi, par exemple, les séquences (d e f g) et (e d f g) d'une part et les séquences (d e f g) et (a e f g) d'autre part présenteront une même distance si un coût nul donné à l'insertion d'un symbole nouveau (cf. objet distance), alors qu'un coût non nul pour l'insertion d'un nouveau symbole mettra en évidence que les deux séquences (d e f g) et (e d f g) sont plus proches que (d e f g) et (a e f g). Dans ce dernier cas, selon le seuil donné à rbynearest, la séquence (a e f g) ne sera jamais confondue avec l'une des autres séquences (d e f g) ou (e d f g).

class-1

Syntax

`(morph2 : :class-1 matrix n alpha? centers verbose)`

Inputs

matrix	matrix, matrice des coordonnées des points dans un n-espace. On peut utiliser l'objet l-matrix pour convertir une liste de listes sous forme matricielle. La structure de la liste doit être : ((xi, yi, zi, ...)(xi+1, yi+1, zi+1,...)...) ou bien sous forme matricielle en Lisp : #2A((xi, yi, zi,...)(xi+1, yi+1, zi+1,...) ...).
n	integer, nombre de classes recherchées.
alpha?	menu options (optionnel), représentation alphanumérique (string) ou numérique des classes.
centers	list (optionnel), liste des coordonnées — même très approximative — des centres de gravité de chaque classe. Dans le cas de classes ambiguës, aide l'algorithme à trouver la bonne classification.
verbose	(optionnel). (yes / no), donne l'état des itérations de l'algorithme pendant le calcul. Un grand nombre d'itérations est le signe d'une stabilisation difficile de l'algorithme, indiquant une classification difficile (ambiguïté des points entre différentes classes)

Output

list

Description

Classification automatique par coalescence, selon l'algorithme dits des « nuées dynamiques » d'un nuage de points dans un espace à n-dimensions (éventuellement une seule) en un nombre de classes défini par l'utilisateur. Peut servir à quantifier des listes de fréquences (F0 par exemple) en un nombre fini de hauteurs, ou des valeurs de durées en classes de durées, mais encore un ensemble de structures ou de formes dont les coordonnées seront déduites par exemple de la fonction prim-tree (en attendant pour la version ultérieure une analyse en composantes principales). Donne dans l'ordre la classe d'appartenance de chaque point dont les coordonnées ont été données.

Exemple

Soit les huit points suivants dans un espace à quatre dimensions :

```
((1 2 3 2) (1 5 3 3.2) (8 1 2 0) (8 5 3 0) (1 1 1 -2.5) (1 -3 1 -2.5) (-2 1 2
    0) (-3 1 2 0)).
```

La classification en deux classes donne : "A A B B B B B B";

en trois classes : "A A B B A A C C";

en quatre classes donne : "B B D D C C A A";

et en cinq classes : "D D A A C E B B".

Remarque : l'initialisation de l'algorithme s'effectue par une répartition aléatoire des c classes sur les n points. Il s'ensuit qu'une même classification peut prendre des formes différentes, dépendantes de l'initialisation aléatoire. Ainsi la classification suivante :"A A B B A A C C" est strictement identique à : "A A D D C C B B". Aussi, lorsqu'il existe une ambiguïté entre les classes (certains points ambigus peuvent être classés tantôt dans une classe, tantôt dans une autre), la classification obtenue est dépendante de l'initialisation aléatoire. On peut pallier cette instabilité en donnant approximativement les coordonnées supposées du centre de chaque classe, ce qui permet d'enlever la sensibilité de la classification à l'initialisation aléatoire (De même lorsque le nombre de classe est trop important par rapport au nombre de points; dans ce cas on peut aboutir à un message d'erreur : réévaluer la fonction jusqu'à obtenir un résultat).

quantize-1

Syntax

(morph2 : :quantize-1*seq class*)

Inputs

seq	list, liste de valeurs de points définis sur une dimension.
class	list, liste des classes de chaque points.

Output

list

Description

Remplace les valeurs des points définis sur une dimension par celle du centre de chaque classe (sera implémenté en n-dimensions dans la version ultérieure).

matrix-center

Syntax

```
(morph2 : :matrix-center matrix )
```

Inputs

matrix	matrix, matrice des coordonnées des points en n-dimensions (utiliser l'objet l-matrix pour convertir une liste de coordonnées (liste) sous forme matricielle Lisp).

Output

list

Description

Donne les coordonnées du centre de gravité d'une matrice des coordonnées de points en n-dimensions (utiliser l-matrix pour convertir une liste de coordonnées de points en matrice).

meta-class1

Syntax

(morph2 : :meta-class1 *matrix*)

Inputs

matrix	matrix, coordonnées des points dans un espace à n dimensions
n	integer, nombre de classes recherchées
iter	integer, nombre d'itérations de l'algorithme

Output

1 = classes fortes;

2 = centres des classes (idem class-1);

3 = probabilité de chaque points d'appartenance a chaque classe.

list

Description

Donne une classification des points de matrix selon le principe des formes fortes : différentes itérations de l'algorithme class-1 sont effectuées et chaque point est attribué à sa classe d'appartenance la plus probable. La probabilité d'appartenance est d'autant plus fiable que le nombre d'itérations est grand.

entropy

Syntax

(morph2 : :Entropy *class res*)

Inputs

class	list, liste d'une distribution de classes d'appartenance de points (telle que le résultat de class-1.
res	menu options (abs / rel), les valeurs sont comprises entre 0 et 2 * log n (entropie maximal d'un ensemble de n points) pour abs (absolu), et entre 0 et 1.0 pour rel (relatif).

Output

list

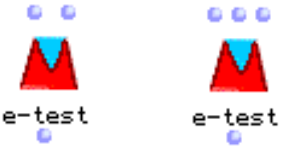
Description

Calcule l'entropie d'une séquence ou d'un ensemble d'éléments selon l'algorithme de Shannon, au sens de la théorie de l'information. Elle est basée sur le nombre de classes d'éléments (une classe étant l'ensemble de tous les éléments identiques) et le nombre d'éléments pour chacune des classes. L'entropie minimale (0.0) est atteinte lorsque tous les éléments sont identiques (nombre de classes = 1). Elle est maximale lorsque chaque classe contient le même nombre d'éléments. Si on considère une séquence aléatoire comme une suite de tirages successifs avec une probabilité égale pour chaque élément,alors l'entropie peut être un indice du degré d'aléatoire de la séquence. Mais, paradoxalement, si on considère une « bonne » forme comme une répartition homogène ("équilibrée") de ces éléments, alors l'entropie ne permettra pas de distinguer une telle forme d'une distribution strictement aléatoire. Cependant, selon la théorie de l'information, une telle structure n'apporte aucune information et, comme pour une séquence aléatoire, l'entropie sera maximale.

Deux modes :

- absolu : entropie absolue, c a dire sa valeur maximale = log2 nbre-de-classe
- relatif -> normalisé entre 0 et 1.0 .

e-test



Syntax

```
(morph2 : :e-test clusters test &optional out )
```

Inputs

clusters	list, liste de différentes classifications d'un ensemble de points.
test	menu option (min / max), défini la recherche de la classification possédant l'entropie minimale ou maximale.
out	menu option (cluster / nth) optionnel, en mode cluster,donne la classification répondant au test d'entropie, en mode nth, donne la position (0 = premier élément) dans la liste de listes d'entrée (clusters) de la classification répondant au test.

Output

list

Description

Typiquement, se connecte à la sortie de l'object meta-class1, donne la classification possédant l'entropie maximale ou bien celle minimale. Peut également se connecter à différents objets class-1 aux arguments différents, tout particulièrement au nombre de classes recherchées différent, rassemblés dans une liste.

delta

Syntax

(morph2 : :delta *list round*)

Inputs

list	list, liste de valeurs numériques.
round	integer, arrondi à 1/x (1000 -> millième, 100 -> centième...).

Output

list

Description

Calcule les différences entre les valeurs consécutives avec un arrondi.

smooth

Syntax

`(morph2 : :smooth` *`list window mode start end`*`)`

Inputs

list	list, liste de valeurs numériques.
window	integer, taille de la fenêtre de lissage.
mode	menu options (nth / aver / med0 / med1/ med2). Mode de lissage.
nth :	ne prend qu'une valeur toute les fenêtres;
aver :	remplace toutes les valeurs dans la fenêtre par leur valeur moyenne;
med0 :	filtrage médian (exclusion des valeurs aberrantes);
med1 :	filtrage médian puis « moyennage » au sein de la fenêtre;
med2 :	filtrage médian, « moyennage » et interpolation linéaire;
start	integer (optionnel), ignore les n premières valeurs.
end	integer (optionnel), ignore les n dernières valeurs.

Output

list

Description

Fonction de lissage de courbes telles que détetection de fondamentales, etc.

midiseq->alpha

Syntax

(morph2 : :midiseq->alpha *chordseq approx*)

Inputs

chordseq	integer ou list, valeur ou liste de valeurs en midicents.
approx	integer (optionnel), approximation : 0 -> 1/2 ton, 1-> 1/4 ton ...

Output

list

Description

Convertit une liste de valeurs (midicents) en symboles alphanumériques correspondant au nom des notes.

mc->alpha

Syntax

(morph2 : :mc->alpha *midicents approx*)

Inputs

midicents	integer, valeur ou liste de valeurs en midicents.
approx	integer, (optionnel) approximation : 0 -> 1/2 ton, 1-> 1/4 ton.

Output

list

Description

Convertit une liste de valeurs (midicents) en chaînes de caractères ("strings") correspondant au nom des notes.

num->alpha

Syntax

```
(morph2 : :num->alpha list)
```

Inputs

list	list, liste ou liste de liste de valeurs entre 0 et 26.

Output

list

Description

Convertit une liste de valeurs ou une liste de listes de valeurs (midicents) en symboles. 0 -> °, 1 -> a, 2 -> b... (les valeurs ne doivent pas être supérieures à 26). Les éléments de chaque sous liste sont concaténés en un seul « mot".

Exemple (1 2 (1 3 4) 1 5) -> (a b a c d a e).

str->symb

Syntax

(morph2 : :str->symb *strings*)

Inputs

strings list, string, liste ou liste de liste de « strings".

Output

list

Description

Convertit une « string » (chaîne de caractères) ou une liste de « strings » ou une liste de liste de « strings » en liste ou liste de liste de symboles.

concatstrings

Syntax

(morph2 : :concatstrings *lofstrings*)

Inputs

lofstrings list, string ou liste ou liste de liste de « strings".

Output

list

Description

Concatène en une seule « string » (chaîne de caractères) une liste de « strings".

l-matrix

Syntax

(morph2 : :l-matrix *list*)

Inputs

list	list, liste de coordonnées en n-dimensions sous la forme : ((xi, yi, zi,...)(xi+1, yi+1, zi+1,...)...).

Output

list

Description

Convertit une liste de coordonnées de points définis dans un espace à n-dimensions sous forme matricielle Lisp.

Glossaire

Après chaque définition est mentionnée la (ou les) fonction(s) à laquelle elle se rapporte.

Altitude

Dans l'analyse d'une séquence en succession de primitives, le nombre d'éléments est une information perdue.

Pour min et max, la profondeur de ces primitives est le nombre d'éléments qui les précèdent et suivent, par couple. Pour flex, la profondeur est définie par le nombre d'élément qui constituent la primitive.

(min flex max)

Analyse contrastive

Segmentation d'une séquence donnée à chaque occurrence de chacun des éléments propres de la séquence et à identifier et nommer les segments ainsi définis. Le critère de segmentation étant l'élément lui-même, on obtient autant de segmentations que d'éléments différents présents dans la séquence. La forme résultante est donc une concaténation de segments commençant par le même élément.

Analyse paradigmatique

Calcul de toutes les combinatoires possibles de tous les segments répétés au moins une fois (à l'identique) dans une séquence (celle-ci « codant » une dimension et une seule).

(*structure-2*).

Critère

Cf. Mark . Ici, le critère est la marque et inversement, ce qui a l'avantage de ne pas faire appel a des données (ou connaissances) externes a la séquence analysée.

Mark

élément d'une séquence dont chaque occurence est un critère de segmentation (ou d'un niveau de segmentation).

Pattern

Au sens utilisé le plus souvent dans morphologie, sous-séquence d'une séquence répétée au moins une fois (de facon strictement identique ou non). En ce sens, le pattern est une sous-séquence caractéristique (une forme) dont la propriété est d'être répétée n fois dans une séquence. Une telle definition induit implicitement et facilement un traitement séquentiel des données, et donc non spatial sauf si le séquentiel est representé par le temps et le spatial (accord, polyphonie...) sous forme de « mots » (concaténation de plusieurs éléments appartenants à des séquences simultanées ou isochrones).

Primitive

Profil de la courbe dans l'environnement immédiat de chaque point ou la dérivée est nulle.

(min-flex-max)

Score

Nombre d'occurences de chaque structure ("AB"...) pour l'ensemble des segmentations effectuées.

(structure-1, structure-2)

Bibliographie

Adorno, T.W. "QUASI UNA FANTASIA", Editions Gallimard, Paris, 1982.

Arom, S. "ESSAI D'UNE NOTATION DES MONODIES À DES FINS D'ANALYSE", *Revue de Musicologie*, LV/2, 172-216, 1969.

Balpe, J.-P. "HYPERDOCUMENTS, HYPERTEXTES, HYPERMEDIAS", Eyrolles, Paris, 1990.

Baboni Schilingi, J., COMPOSITION PAR MODÈLES INTERACTIFS† : SYSTÈMES GÉNÉRATIFS ET APPLICATIONS MUSICALES, DEA de Musique et musicologie du XXe siècle, sous la direction de H. Dufourt et M. Battier, EHESS, septembre 1998.

Balpe, J.-P. "HYPERTEXT ET HYPERMEDIAS Vol. 1 - N° 1/1997, pages 11 à 22, 1997.

Bastide, R. "SENS ET USAGE DU TERME STRUCTURE DANS LES SCIENCES HUMAINES ET SOCIALES", ed. Mouton, Paris, 1972.

Budd, M. “MUSIC AND EMOTION”, International library of Philosophy, 1992.

Davies, S. "MUSICAL MEANING AND EXPRESSION", Cornell University Press. London, 1991.

Delattre, P. "SYSTEMES DE TRANSFORMATION", in Encyclopaedia Universalis, France S.A., 1985.

Delattre, P. "LANGAGE INTERDISCIPLINAIRE ET THÉORIE DES SYSTEMES", in Structure et Dynamique des Systèmes, Séminaires interdisciplinaires du collège de France, Doin ed., Paris, 1976.

Delattre, P. "SYSTEME , STRUCTURE, FONCTION, EVOLUTION", essay d'analyse épistémologique. 2éme édition, Ed. Maoline, 1971.

Duday & al. "ANALYSE DES DONNEES NUMERIQUES", Dunod, Paris, 1985.

Einstein , A. "LA THEORIE DE LA RELATIVITE RESTREINTE ET GENERALE", Dunod, Paris, 1990.

Escal, F. "LE COMPOSITEUR ET SES MODELES", PUF, Paris, 1984.

Farouki, N. "LA RELATIVITE", Dominos Flammarion, 1993.

Ferneyhough, B. "COLLECTED WRITINGS", James Boros and Richard Toop. San Diego, 1995.

Giorello, G. "*SISTEMI*", ENCICLOPEDIA EINAUDI "SOCIETA' ET TECNICA" vol. 15.

Griffiths, P. "MODERN MUSIC AND AFTER", Oxford University Press., 1995.

Giré, A. "MODELES MATHÉMATIQUES DE SYSTEMES ÉVOLUTIFS HÉRÉDITAIRES", Presses Universitaires de Lyon, Lyon, 1987.

Genette, G. "PALIMPSESTES", Edition du Seuil. Paris, 1982.

Keeney, B.P. “L’ESTETICA DEL CAMBIAMENTO”, Casa Editrice Astrolabio, 1987.

Klein, E. "LA PHYSIQUE QUANTIQUE", Dominos Flammarion, 1996.

Imberty, M. “SUONI EMOZIONI SIGNIFICATI”, Editrice CLUEB, 1986.

I.R.C.A.M. “LA COMPOSITION ASSISTEE PAR ORDINATEUR”, Centre Georges-Pompidou, Paris, 1993.

Lorenzetti/Antonelli “PROCESSI COGNITIVI IN MUSICA », Collana diretta da M.Cesa-Bianchi Franco, Angeli, 1986.

Lostia, M. “MUSICA E PSICOLOGIA”, Collana diretta da M.Cesa-Bianchi Franco, Angeli, 1989.

Laurson, M. "PATCHWORK A VISUAL PROGRAMMING LANGUAGE AND SOME MUSICAL APPLICATIONS", Sibelius Academy, 1996.

MacCallum/Riess “AN APPLICATION OF PRINCIPAL DIRECTION SCALING TO AUDITORY PATTERN PERCEPTION”, Ohio State University, 1993.

Nattiez, J-.J. "FONDEMENTS D'UNE SEMIOLOGIE DE LA MUSIQUE", Union Générale d'Edition, Paris, 1975.

Nattiez, J.-J. "Analyse d'"un" chant inuit", NDROJE BALENDRO. MUSIQUE, TERRAINS ET DISCIPLINES, (Dehoux, Fürniss, Olivier, Rivière, Voisin éd.), SELAF, Paris, 1995.

Pelinsky, R. "LES INUIT DU CARIBOU", Thèse de doctorat, Université de Montréal, Montréal (tapuscrit), 1985.

Orcalli, O. “FENOMENOLOGIA DELLA MUSICA SPERIMENTALE”, Sonus Edizioni Musicali. Potenza, 1993.

Piana et al. : “LA PERCEZIONE MUSICALE”, Guerini Studio, 1993.

Raffman, D. “LANGUAGE, MUSICA AND MIND”, Bradford Book. London, 1993.

Rivière, H. "D'un point de vue rythmique...", NDROJE BALENDRO. MUSIQUE, TERRAINS ET DISCIPLINES, (Dehoux, Fürniss, Olivier, Rivière, Voisin éd.), SELAF, Paris, 1995.

Ruwet, N. "MUSIQUE, LANGAGE ET POESIE", Le Seuil, Paris, 1972.

Segal, H. “SOGNO, FANTASIA E ARTE”, Raffaello Cortina Editore, 1991.

Sloboda, J.A. “THE MUSICAL MIND”, Oxford psychology series N.5., 1985.

Schalkoff, R. "PATTERN RECOGNITION, STATISTICAL STRUCTURAL AND NEURAL APPROACHES", John Wiley & Sons, 1992.

Schwanauer/Levitt "MACHINE MODELS OF MUSIC", MIT Press. London, 1993.

Steele JR. G.L. "COMMON LISP, THE LANGUAGE SECOND EDITION", Digital Press, 1990.

Thom, R. "PARABOLES ET CATASTROPHES", Flammarion (Champs), Paris, 1980.

Voisin, F. & Cloarec-Heiss, F. "Echelles musicales et données linguistiques : vers une histoire des sociétés oubanguiennes", NDROJE BALENDRO. MUSIQUE, TERRAINS ET DISCIPLINES, (Dehoux, Fürniss, Olivier, Rivière, Voisin éd.), SELAF, Paris, 1995.

Walliser, B. "SYSTEMES ET MODELES", Editions du Seuil, Paris, 1977.

Wasemberg, B. "L’âme de la méduse. Idées sur la complexité du monde", Editions du Seuil, Paris, 1997.

Xenakis, I. FORMALIZED MUSIC. Indiana University Press. 1971.

Index