MATHÉMATIQUES ET SCIENCES HUMAINES

MARC CHEMILLIER

Solfège, commutation partielle et automates de contrepoint

Mathématiques et sciences humaines, tome 110 (1990), p. 5-25

<http://www.numdam.org/item?id=MSH_1990__110__5_0>

Math. Inf. Sci. hum., (28e année, n°110, *1990*, p.5-25)

SOLFEGE, COMMUTATION PARTIELLE ET AUTOMATES DE CONTREPOINT

Marc CHEMILLIER[1]

RÉSUMÉ - *Cet article s'inscrit dans un travail d'étude mathématique de la combinatoire musicale. Dans les deux premières parties, on propose un modèle formel de la synchronisation musicale, dont on donne quelques propriétés abstraites en comparant ce modèle à d'autres modèles de synchronisation utilisés pour le parallélisme en informatique théorique. Dans la troisième partie, on décrit un algorithme de production automatique de contrepoint par automates finis.*

SUMMARY - Solfege, partial commutation, and counterpoint automata.
This article is part of a research on a mathematical structure for musical combinatorics. In the first part, we propose an algebraic structure adapted to the representation of the musical synchronisation. We then compare this structure with formal tools which are studied in the field of concurrent programming. In the last part, we describe an algorithm to produce counterpoint by means of finite automata.

INTRODUCTION.

Ce travail s'inscrit dans une recherche sur un modèle théorique général permettant de représenter les relations de base qui existent entre les différents éléments d'un texte musical: relation de *succession* d'une part, mais aussi, pour la musique polyphonique, relation de *simultanéité*. La relation de succession est traditionnellement représentée, dans la théorie des langages formels, en considérant des suites de symboles que l'on met bout à bout au moyen d'une opération de *concaténation*, et conduit à étudier la structure de monoïde libre sur un alphabet. Mais dans le cas d'un texte musical, il est nécessaire d'enrichir la structure de monoïde en définissant une nouvelle opération correspondant à la *superposition* de séquences musicales. Une telle extension est proposée dans [8], et conduit à étudier la structure algébrique de *solfège*.

Après avoir rappelé dans la première partie, les principales définitions du modèle proposé, on étudiera dans la deuxième partie les analogies que ce modèle présente avec deux autres modèles formels utilisés pour l'étude du parallélisme en informatique. En effet, la notion de *polyphonie musicale* qui est au centre du présent travail n'est pas sans ressemblance avec celle de *synchronisation de processus parallèles*. On donnera également quelques résultats théoriques de reconnaissabilité par automates finis concernant le modèle présenté. Dans la troisième partie de cet article, on appliquera les principaux résultats énoncés précédemment à la description d'un algorithme de production automatique de contrepoint.

---

[1] LITP, Université de Paris 7, 4, place Jussieu, 75005 ; (actuellement: LaBRI, Université de Bordeaux I, 351, cours de la Libération, 33405 TALENCE).

Les définitions et notations utilisées dans cet article sont celles de la théorie élémentaire des automates et des langages formels (cf. [18]).

1. SUCCESSION ET SIMULTANÉITÉ.

a. Notion de polyphonie musicale.

Pour introduire le point de vue adopté dans ce travail, revenons à un texte écrit par le compositeur Iannis Xenakis au début des années soixante, intitulé ***Musiques formelles*** (cf. [30]). Ce texte qui exposait les principaux procédés de composition de Xenakis, proposait également une manière originale de considérer les phénomènes musicaux, que l'on trouvera résumée dans les deux citations suivantes : "*Postulat.* - Nous nous refuserons systématiquement un jugement de qualité sur tout événement sonore; ce qui comptera, seront les relations abstraites à l'intérieur de l'événement ou entre plusieurs événements et les opérations logiques que l'on pourra leur infliger. A ce titre, son émission est une sorte d'énoncé, d'écriture, une sorte de symbole sonore que l'on peut, à son tour, noter graphiquement par une lettre, a" (p. 186), et plus loin: "RÉSUMÉ. Soit trois événements a, b, c, émis successivement ; Première étape: on distingue trois événements et c'est tout. Deuxième étape: on distingue une «succession temporelle» c'est-à-dire une correspondance entre événements et instants. Il en résulte : a avant b $\neq$ b avant a (non commutativité)" (p. 190). Les spécialistes en informatique théorique auront reconnu dans l'équation ci-dessus une opération familière appelée *concaténation*. C'est bien à cette opération que l'on fera appel ci-après pour représenter l'idée de *succession* d'événements musicaux, dans le cadre d'un monoïde libre sur un alphabet.

Cependant, l'un des aspects essentiels de la polyphonie musicale est de mettre en jeu, non pas une séquence d'événements sonores, mais plusieurs séquences, synchronisées entre elles. Pour compléter notre modèle, il est donc nécessaire de réfléchir sur la nature du ou des mécanismes de synchronisation qui interviennent dans la musique polyphonique.

Patrick Greussay avait déjà examiné, en 1973, quelques aspects de la synchronisation musicale, en relation avec certaines techniques utilisées pour le parallélisme informatique. Dans sa thèse intitulée *Modèles de description symbolique en informatique musicale* (cf. [19] p.111), il proposait entre autres exemples, l'extrait musical suivant :

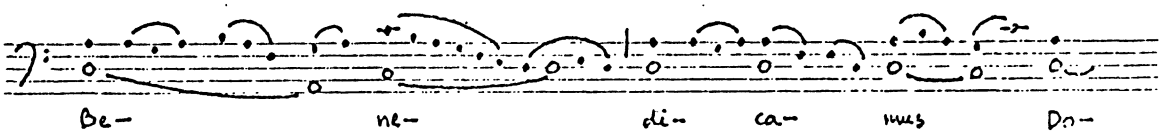
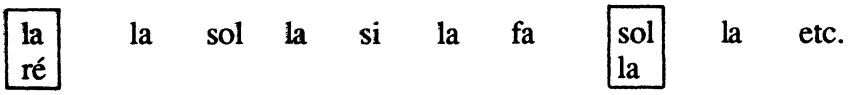


Figure 1. Organum à deux voix de l'Ecole de Compostelle
(environ 1125)

Cet exemple fait apparaître deux processus qui se déroulent en parallèle avec une relative indépendance. La synchronisation des deux voix n'est réalisée qu'en certains points, la voix grave attendant pour passer à la note suivante la fin du mélisme de la voix aiguë. Selon l'expression de Greussay, la partie supérieure "fait avancer" le ténor.

Le point de vue de Greussay consistait à montrer comment on pouvait réaliser en machine une synchronisation de ce type. Dans notre perspective de formalisation de la combinatoire musicale, nous supposerons résolu ce problème de la synchronisation effective, en machine, de

séquences musicales (à l'aide de séquenceurs MIDI par exemple), pour ne retenir de l'exemple ci-dessus que l'observation suivante : les notes de la voix inférieure doivent être émises *en même temps* que certaines notes de la voix supérieure. En plus de la relation de succession, il faut établir une relation de *simultanéité* entre certains événements figurant dans la séquence. L'idée la plus naturelle pour représenter le fragment musical ci-dessus en tenant compte de cette deuxième relation, consiste à admettre que dans la succession temporelle ne figurent pas que des événements isolés, mais aussi des *blocs d'événements simultanés*. Ainsi, la séquence deviendra :



De cette idée de succession de blocs d'événements simultanés, il est possible de dégager un modèle simple de synchronisation, que l'on va exposer dans la section suivante. La deuxième partie de l'article consistera à comparer le modèle ainsi construit avec deux autres modèles de synchronisation bien connus en informatique théorique.

b. Structure algébrique de solfège.

On considère un ensemble E d'*événements*, P = **P**(E) l'ensemble de ses parties sera considéré comme un *alphabet*, et on appelle *blocs* les éléments de l'alphabet. On forme le monoïde libre P* sur P qui est l'ensemble des *séquences musicales*, muni de la traditionnelle opération de concaténation, notée par un point. On note 1 la séquence vide.

Dans P*, on définit une loi de composition, notée ||, correspondant à la ***superposition*** de deux séquences musicales :

1) pour tout u de P*, $u \| 1 = 1 \| u = u$.

2) pour u et v non vides, soient u = au', v = bv', avec a et b dans P. On pose :
$u \| v = (a \cup b)(u' \| v')$.

De la même façon, on définit dans P* l'*interposition* de deux séquences musicales, notée $\perp$:

1) pour tout u de P*, $u \perp 1 = 1 \perp u = 1$.

2) pour u et v non vides, soient u = au', v = bv', avec a et b dans P. On pose :
$u \perp v = (a \cap b)(u' \perp v')$.

Si |u| désigne la longueur du mot u , on a :

$|uv| = |u| + |v|$,

$|u \| v| = \max(|u|, |v|)$,

$|u \perp v| = \min(|u|, |v|)$.

L'ensemble P* se trouve ainsi muni de trois opérations ., ||, $\perp$, vérifiant les propriétés suivantes :

1) (P*, .) est un monoïde d'élément neutre 1.

2) (P*, ||, $\perp$) est un treillis distributif.

3) . est distributive **à gauche** sur || et $\perp$.

4) 1 est neutre pour ||.

On convient d'appeler *solfège* un ensemble S muni de trois lois ., ||, $\perp$ vérifiant les quatre propriétés ci-dessus. Un tel ensemble est donc ordonné par sa structure de treillis, et dispose d'un plus petit élément, qui est aussi neutre pour une opération compatible à gauche avec la relation d'ordre. L'ensemble **N** des entiers naturels, muni des opérations +, max, min est un exemple d'une telle structure.

Si X et Y sont des parties de P*, on définit la *superposition* de X et Y:

$X \parallel Y = \{x \parallel y, x \in X, y \in Y\}$,

ainsi que la *superposition à longueurs égales* de X et Y :

$X \pi Y = \{x \parallel y, x \in X, y \in Y, |x| = |y|\}$

EXEMPLE. Soient u et v les deux séquences suivantes :

u = la ø si ø do ø,

v = la ø sol fa mi ré mi ø.

La concaténation, la superposition, et l'interposition de u et v seront respectivement :

uv = la ø si ø do ø la ø sol fa mi ré mi ø,

u || v = la ø si do
sol fa mi ré mi ø,

u $\perp$ v = la ø ø ø ø ø.

Ces trois séquences musicales sont représentées figure 2. Dans la convention adoptée ici, les événements correspondent uniquement à l'*attaque* des notes. Chaque bloc a pour durée une croche, et le bloc vide ø indique que la séquence se prolonge sans qu'il y ait une nouvelle attaque de note.

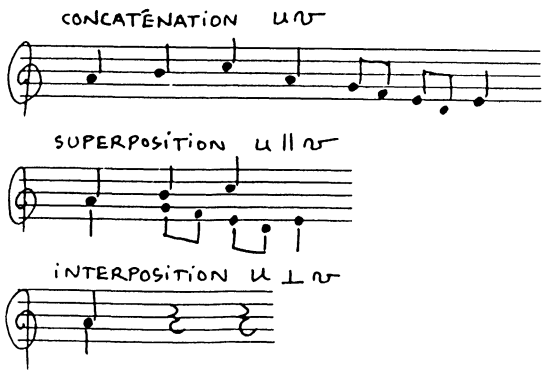


Figure 2.

Dans la suite de cet article, on examinera successivement deux autres types de synchronisation, utilisés pour le parallélisme en informatique et dont les propriétés formelles sont bien connues :

1) le produit de synchronisation (dans un monoïde partiellement commutatif libre), et

2) le shuffle littéral.

On comparera chacun de ces deux modèles avec celui de la superposition qui a été présenté ci-dessus. La comparaison entre le shuffle littéral et la superposition sera également l'occasion d'énoncer quelques résultats théoriques de reconnaissabilité par automates finis concernant la superposition.

2. MODÈLES DE PARALLÉLISME.

c. Monoïdes partiellement commutatifs libres.

Considérons deux processus, tous deux constitués d'une succession d'actions qui peuvent être respectivement: a ou b pour le premier processus, et a ou c pour le second. Par exemple :

p_1 = b b a b, et

p_2 = c a c.

La règle de synchronisation est la suivante : les actions communes aux deux processus doivent être exécutées en commun par l'un et par l'autre.

Le résultat de cette synchronisation peut être schématisé comme suit :

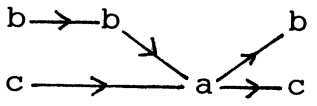


Figure 3.

La relation de succession entre les occurrences des différentes lettres apparaît ici comme une relation d'ordre partiel. Les objets de ce type ont été définis formellement par Viennot sous le nom d'*empilements de pièces*(cf. [28] p.324]).

Si l'on peut intervertir l'ordre dans lequel sont effectuées les actions b et c, en revanche il n'est pas possible de permuter a et b, ni a et c. Ainsi, si l'on représente séquentiellement les différentes actions à exécuter pour la synchronisation de p_1 et p_2, alors c b b a b c, et b c b a b c conviennent, mais b c a b b c ne convient pas. Dans le cas général, on dira que l'ensemble des lettres est muni d'une relation de *commutation partielle.*

Si A est l'alphabet, et θ cette relation (supposée symétrique et antiréflexive), on forme sur le monoïde A*, la congruence engendrée par les relations du type ab $\approx$ ba, pour (a,b) appartenant à θ. Le quotient de A* par cette congruence est noté M(A,θ). C'est le *monoïde*

partiellement commutatif libre sur A associé à θ, et ses éléments sont des *traces*. Pour u dans A*, [u] désigne sa classe dans M(A,θ), et Φ est la surjection canonique de A dans M(A,θ), qui à u associe $\Phi(u) = [u]$. Dans l'exemple ci-dessus, le résultat de la synchronisation de p_1 et p_2 est une classe de M(A,θ), dont un représentant est c b b a b c.

On peut imaginer d'associer à une trace différentes représentations sous forme de successions de blocs d'actions simultanées (ou plutôt d'actions "pouvant être exécutées simultanément"). Par exemple, la trace de la figure 3 pourrait être représentée par :

b b a b.
c c

Dans cette perspective, qu'en est-il de la superposition : $p_1 \| p_2$? On constate qu'elle ne convient pas pour représenter la synchronisation des deux processus p_1 et p_2 :

$p_1 \| p_2$ = b b a b.
c a c

Il est facile de voir qu'une condition nécessaire et suffisante pour que les deux modes de synchronisation coïncident, est que les actions communes à deux processus apparaissent dans les mêmes positions pour chacun de ces deux processus. Ce n'est pas le cas ici, puisque a apparaît en troisième position dans p_1, et en deuxième position dans p_2.

Il est possible de donner une formulation précise de cette propriété. C'est ce que nous allons faire ci-après, mais il faut auparavant rappeler quelques résultats élémentaires concernant les monoïdes partiellement commutatifs libres.

d. Produit de synchronisation et superposition.

Une trace de M(A,θ), c'est-à-dire une classe d'équivalence, s'exprime au moyen du *produit de mixage*. Celui-ci a été introduit par De Simone [13], de la manière suivante.

Soient A et B deux alphabets, on considère le monoïde libre (A $\cup$ B)*. Pour toute partie C de A $\cup$ B, on définit la projection sur C, notée π_C, comme étant le morphisme de (A $\cup$ B)* dans C* tel que $\pi_C(x) = x$ pour x appartenant à C, et $\pi_C(x) = 1$ pour x appartenant à (A $\cup$ B)-C. Pour A et B (inclus dans A $\cup$ B), on définit les deux projections π_A et π_B. Soient X et Y deux parties respectivement de A* et B*. Le *produit de mixage* de X et Y, noté X $\amalg$ Y est une partie de (A $\cup$ B)* définie par :

$$X \amalg Y = \pi_A^{-1}(X) \cap \pi_B^{-1}(Y).$$

On note Δ la diagonale de A $\times$ A. Une partie A' de A est une *clique* pour la relation de commutation θ (resp. de non-commutation $\bar{\theta}$), si la restriction de $\theta \cup \Delta$ (resp. $\bar{\theta}$) à A' est A' $\times$ A'. Un *recouvrement* par cliques de θ (resp. $\bar{\theta}$) est une famille de parties $A_1, A_2, \ldots A_p$ de A telles que $\theta \cup \Delta$ (resp. $\bar{\theta}$) soit égal à l'union des $A_i \times A_i$, pour $i \leq p$. Le résultat suivant donne l'expression de la classe d'un mot u de A* au moyen du produit de mixage [14] :

PROPOSITION 1. *Si* $A_1, \ldots A_p$ *est un recouvrement par cliques de* $\bar{\theta}$, *alors pour* u *et* v *dans* A*, *on a :*

1) $u \approx v$ *si et seulement si* $\pi_i(u) = \pi_i(v)$ *pour tout* $i \leq p$, π_i *désignant la projection* π_{A_i} *de* A* *dans* A_i*.
2) $[u] = \pi_1(u) \text{ m } \pi_2(u) \ldots \text{ m } \pi_p(u)$.

Si B est une partie de A, on généralise pour $M(A,\theta)$ la notion de *projection*, en définissant π_B de $M(A,\theta)$ dans $M(B,\theta|_B)$ par $\pi_B([u]) = [\pi_B(u)]$. Cette définition a bien un sens, car si $u \approx v$, alors pour tout $i \leq p$, $\pi_i(\pi_B(u)) = \pi_{Ai\cap B}(u) = \pi_B(\pi_i(u)) = \pi_B(\pi_i(v)) = \pi_i(\pi_B(v))$, donc d'après le théorème ci-dessus, $\pi_B(u)$ est équivalent à $\pi_B(v)$. A l'aide de ces projections, on peut énoncer un résultat analogue au précédent :

PROPOSITION 2. *Si* t *et* t' *sont deux éléments de* $M(A,\theta)$, *alors* $t = t'$ *si et seulement si* $\pi_i(t) = \pi_i(t')$ *pour tout* $i \leq p$.

Le produit de mixage permet de donner une définition précise du modèle de synchronisation présenté dans la section précédente, sous la forme d'une opération sur les traces appelée *produit de synchronisation*, notée $\boxtimes$ (cf. [15] p. 6). Sous sa forme générale, il s'agit d'une opération ensembliste dans $M(A,\theta)$. Soit $A_1, A_2, \ldots A_p$ un recouvrement par cliques de $\bar{\theta}$. Remarquons d'abord que tous les mots de A_i* sont *rigides* (i. e. seuls dans leur classe). Ceci vient de ce que $\theta|_{A_i}$ est vide. On peut donc les identifier à leur trace, et identifier A_i* à $M(A_i,\theta|_{A_i})$.
Si $X_1, \ldots X_p$ sont des parties de A_1*, ... A_p*, on pose :

$X_1 \boxtimes \ldots X_p = \{t_1 \text{ m } \ldots \text{ m } t_p, t_i \in X_i \text{ pour } i \leq p\}$.

En considérant le cas où tous les X_i sont réduits à des singletons t_i, on peut définir pour tout p-uplet $t_1, \ldots t_p$ de A_1*, ... A_p* tel que $t_1 \text{ m } \ldots \text{ m } t_p$ soit une classe non vide de $M(A,\theta)$ notée t, le produit de synchronisation des t_i :
$t_1 \boxtimes \ldots \boxtimes t_p = t$.

EXEMPLE. La relation de non-commutation $\bar{\theta}$ correspondant à l'exemple de la figure 3 est représentée figure 4. On a représenté également un recouvrement par cliques de $\bar{\theta}$: $A_1 = \{a,b\}$, et $A_2 = \{a,c\}$.

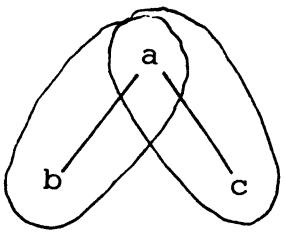


Figure 4.

Les mots : u = c b b a b c, et v = b c b a b c, ont des projections identiques respectivement sur A_1* et sur A_2* : b b a b, et c a c, c'est-à-dire p_1 et p_2. Ils représentent donc la même trace, et cette trace est le produit de synchronisation des deux processus :

$[u] = [v] = p_1 \boxtimes p_2$.

Par contre, le mot b c a b b c se projette sur A_1^* en b a b b, donc il donne une trace différente.

Pour énoncer la propriété étudiée mettant en relation la superposition et le produit de synchronisation, il reste à établir un lien entre les séquences de blocs d'événements simultanés (ou "pouvant être exécutés simultanément"), et les traces de $M(A,\theta)$. On utilise pour cela la surjection canonique Φ de A^* dans $M(A,\theta)$, en la *prolongeant* de manière adéquate.

On forme $P = \mathbf{P}(A)$, et P^* le solfège associé. On considère l'ensemble C, inclus dans P, des cliques de la relation de commutation θ, et on forme le sous-monoïde C^* de P^* (cf. [14] p. 55). Par définition des cliques de θ, C contient les singletons de A, donc A^* peut être identifié à un sous-monoïde de C^*. Les cliques de la relation de commutation regroupent les actions qui peuvent être exécutées *simultanément.*

On prolonge alors Φ en un morphisme de C^* dans $M(A,\theta)$. Pour simplifier les notations, on notera avec le même symbole Φ et son prolongement à C^*. Il suffit de définir Φ pour tous les éléments de C, de la manière suivante :

1) pour a dans A, $\Phi(a) = [a] = a$.

2) pour c dans C, avec card $c \geq 2$, $\Phi(c) = \Phi(c\text{-}a).a$.

Cette définition a bien un sens, car les éléments de c commutent tous entre eux.

EXEMPLE. Soit t la trace étudiée dans l'exemple de la section précédente (figure 3). L'élément de C^* ci-dessous appartient à $\Phi^{-1}(t)$:

b b a b.
c c

Si $A_1, A_2, \ldots A_p$ désigne un recouvrement par cliques de la relation de non-commutation $\bar{\theta}$, et t une trace, on note $|t|_i$ le nombre de lettres de A_i apparaissant dans t, et $|t|_{ij}$ le nombre de lettres de $A_i \cap A_j$ apparaissant dans t. Dire que les éléments de $A_i \cap A_j$ qui apparaissent dans un processus t de A_i^* apparaissent *dans la même position* dans t' de A_j^*, peut s'exprimer de manière équivalente par la condition :

$|t \perp t'|_{ij} = |t|_j = |t'|_i$.

On est alors en mesure d'énoncer la proposition suivante :

PROPOSITION. *Soient* $t_1, \ldots t_p$ *appartenant à* $A_1^*, \ldots A_p^*$, *tels que leur produit de synchronisation soit défini. Alors :*

$t_1 \boxtimes \ldots \boxtimes t_p = \Phi(t_1 \| \ldots \| t_p)$

si et seulement si $|t_i \perp t_j|_{ij} = |t_i|_j = |t_j|_i$, *pour tous* i, j.

DÉMONSTRATION. On note t le produit de synchronisation des t_i, et z leur superposition. On note respectivement π_i, π_a les projections associées à A_i et {a}.

CONDITION NÉCESSAIRE. Supposons qu'il existe t_i et t_j, et une lettre a de $A_i \cap A_j$ qui n'apparaît pas dans la même position dans t_i et dans t_j. En posant $k = |t_i|_a$ on a : $|t_i \perp t_j|_a < k$. On a $\pi_a(t_j) = \pi_a(\pi_j(t)) = \pi_a(t) = \pi_a(\pi_i(t)) = \pi_a(t_i)$, car a appartient à A_i et A_j, donc les nombres d'occurrences de a dans t_j et t sont égaux à k. Soit n le nombre d'occurrences de a dans $t_i \parallel t_j$. On a : $n = k+k-|t_i \perp t_j|_a > k$. D'où $\Phi(z) \neq t$.

CONDITION SUFFISANTE. Supposons $|t_i \perp t_j|_{ij} = |t_i|_j = |t_j|_i$, pour tous i, j. Vérifions que z appartient à C*. S'il existe un bloc de z qui n'est pas une clique pour la relation de commutation, ce bloc contient deux lettres a et b distinctes ne commutant pas. Elles appartiennent donc à un A_k. Si c est la lettre de t_k qui occupe la même position, alors c diffère de l'une des deux lettres, par exemple de a. Dès lors, a est une lettre de $A_i \cap A_k$ apparaissant dans t_i et qui n'apparaît pas dans t_k à la même position, ce qui est contraire à l'hypothèse.

Pour montrer que $\Phi(z) = t$, vérifions $\pi_i(\Phi(z)) = \pi_i(t)$ quelque soit $i \leq p$ (cf. Proposition 2). Pour i donné, on construit u tel que $z = t_i \parallel u$, en posant : u de longueur $|z|$, et chaque bloc u[k] de u vaut : $z[k] - t_i[k]$ si $k \leq |t_i|$, et z[k] sinon. D'après la définition de Φ, on a alors : $\Phi(z[k]) = \Phi(u[k])\ \Phi(t_i[k])$ si $k \leq |t_i|$, et $\Phi(z[k]) = \Phi(u[k])$ sinon. Par ailleurs, u[k] ne contient aucun élément de A_i, car sinon, pour un certain $j \neq i$, cet élément serait dans $A_i \cap A_j$, en position k dans t_j, mais pas en position k dans t_i, ce qui contredit l'hypothèse. D'où : $\pi_i(\Phi(u[k])) = 1$ pour tout k, ce qui donne $\pi_i(\Phi(z)) = \pi_i(\Phi(t_i)) = t_i$, donc $\Phi(z) = t$. □

e. Shuffle littéral.

Nous allons à présent examiner le deuxième type de synchronisation annoncé à la fin de la première partie : le shuffle littéral. La règle de synchronisation des deux processus en jeu est la suivante : chacun des processus exécute une action à tour de rôle. Si par exemple on considère les deux processus ci-après :

$p_1 = a_1\ a_2$, et

$p_2 = b_1\ b_2\ b_3$,

alors le résultat de la synchronisation sera la séquence suivante :

$(a_1\ b_1)\ (a_2\ b_2)\ b_3$,

ce qui conduit à la définition d'une nouvelle opération entre séquences, appelée *shuffle littéral.*

Une définition formelle est donnée dans [3], p. 0-15. Pour $u = u_1...u_n$ et $v = v_1...v_n$ de même longueur, soit $I(u,v) = u_1v_1...u_nv_n$ le mot obtenu en alternant une lettre de u et une lettre de v, de 1 à n. Ensuite, pour u et v de longueurs différentes, soit par exemple v = v'w, où |v'| = |u|. On pose alors: $u \sqcup_1 v = I(u,v')w$ et $v \sqcup_1 u = I(v',u)w$.

La relation entre ce mode de synchronisation et celui représenté par la superposition apparaîtra clairement grâce à l'exemple donné figures 5 et 6, où les deux types de synchronisation sont appliqués aux processus $p_1 = a_1\ a_2\ a_3\ a_4$ et $p_2 = b_1\ b_2\ b_3$:

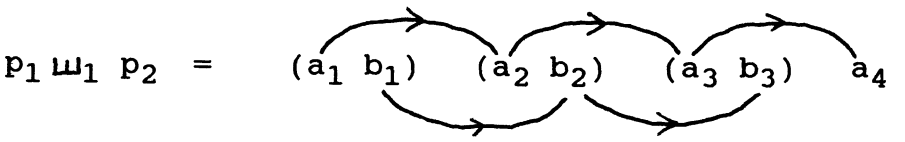


Figure 5.

On passe du shuffle littéral à la superposition en regroupant dans un même bloc d'événements simultanés, les lettres apparaissant à l'intérieur d'un même facteur dans la factorisation ci-dessus.

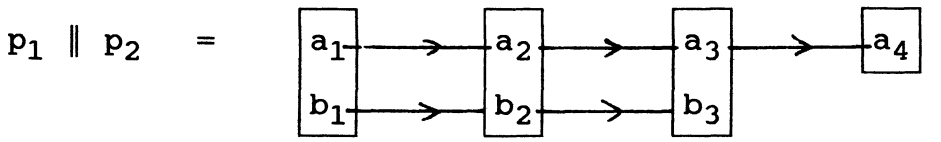


Figure 6.

Il en résulte une analogie très forte entre ces deux modes de synchronisation, ce qui peut se traduire formellement en donnant au shuffle littéral une nouvelle définition, calquée sur celle de la superposition :

1) pour tout u de A*, $u \sqcup_1 1 = 1 \sqcup_1 u = u$,

2) pour u et v non vides, soient u = au', v = bv', avec a et b dans A. On pose : $u \sqcup_1 v = (ab)(u' \sqcup_1 v')$.

Notons que la condition 1 exprime que le mot vide est neutre pour le shuffle littéral.

Malgré cette analogie, la réduction opérée en regroupant dans un même bloc les lettres d'un même facteur fait de la superposition une opération plus simple que le shuffle littéral, et les propriétés formelles dont jouit la première sont plus nombreuses et plus systématiques que celles qu'on peut vérifier concernant la seconde.

Par exemple, on a vu que la superposition est commutative et associative. Il est facile de voir que le shuffle littéral n'est pas commutatif, et l'exemple ci-après montre qu'il n'est pas non plus associatif :

$(a \sqcup_1 b) \sqcup_1 c = ab \sqcup_1 c = (ac)b$,

$a \sqcup_1 (b \sqcup_1 c) = a \sqcup_1 bc = (ab)c$.

Si l'on se place à présent du point de vue de la reconnaissabilité par automates finis, on démontre que les deux opérations se comportent de la même façon lorsqu'elles combinent entre eux deux langages reconnaissables. On a en effet le résultat suivant (les principaux résultats énoncés ci-après sont détaillés dans [10]) :

THÉORÈME 1. *Si* X *et* Y *sont des langages reconnaissables, alors* $X \parallel Y$, $X \perp Y$ *et* $X \sqcup_1 Y$ *sont aussi reconnaissables.*

Les situations divergent lorsqu'on s'intéresse aux opérations *itérées*, construites de la manière suivante : soit X un langage, on pose :

$X^{(0)} = \{1\}$,
$X^{(1)} = X$,
$X^{(n)} = X^{(n-1)} \parallel X$ pour tout n entier,

puis on définit $X^{o} = \bigcup_{n \in N} X^{(n)}$,

et de même à partir du shuffle littéral, on définit $X^{\sqcup_1 *}$.

Dans le cas d'un *langage* X *fini*, on voit de manière évidente que le langage X^{o} est lui aussi fini. Par contre, pour le shuffle littéral itéré, cette remarque est fausse. Il faut la remplacer par le résultat suivant (cf. [3]) :

PROPOSITION: *Si* X *est fini, alors* $X^{\sqcup_1 *}$ *est reconnaissable.*

Si l'on considère à présent un *langage* X *reconnaissable*, on observe à nouveau des comportements différents. Pour la superposition itérée, on a le théorème suivant (cf. [10]) :

THÉORÈME 2 (Latteux). *Si* X *est reconnaissable, alors* X^{o} *est aussi reconnaissable.*

Ce résultat est d'autant plus remarquable que pour le shuffle littéral itéré, on a un résultat négatif : le langage $(a^2(ab)^*b^2)^{\sqcup_1 *}$ n'est pas reconnaissable.

Dans la suite de cet article, on montre comment le résultat énoncé ci-dessus concernant la reconnaissabilité par automate fini de la superposition de deux langages reconnaissables (théorème 1) peut être utilisé pour concevoir un algorithme de *production automatique de contrepoint.*

3. AUTOMATES FINIS ET CONTREPOINT.

Les traités d'harmonie et de contrepoint traditionnels établissent à des fins pédagogiques, des "règles" déterminant l'ensemble des séquences musicales jugées conformes à un style donné. Le plus souvent, ces règles prescrivent, pour une situation donnée, l'éventail des transitions acceptables, et l'exercice de l'élève consiste à produire une séquence musicale en "appliquant les règles".

En fait, cette manière de concevoir la production de musique par application de règles ne se limite pas au strict contexte pédagogique: en dehors de l'improvisation complètement libre, toute forme de production musicale la manifeste, à des degrés divers. L'application de règles peut être parfois complètement mécanisée comme dans le cas des "automates musicaux": la boîte à musique d'un nourisson en est un exemple familier dans lequel l'automate se réduit à une boucle qui tourne sur elle-même. Mais cet exemple un peu particulier n'est pas aussi insignifiant qu'il n'y paraît. A la fin du XVIIIème siècle, de nombreux fascicules ont été publiés qui proposaient

au lecteur le moyen de composer des pièces de musique automatiquement, "sans même savoir la musique". L'un d'eux, paru sous le titre *Ludus melothedicus* (cote BN: Vm 81137), est reproduit en figure 7.


LU'DUS MELOTHEDICUS
OU
LE JEU DE DEZ
HARMONIQUE
Contenant plusieurs Calculs
DEDIÉ
A Monsieur le Comte
DE LAMBERG
Chambellan de leur Majesté Impériale
A PARIS
A LYON

Figure 7.

Dans ces méthodes de composition automatique, l'algorithme est le suivant: on tire au hasard une première mesure dans une liste de mesures possibles, puis une deuxième mesure dans une autre liste, etc. jusqu'au tirage de la dernière mesure dans une dernière liste. Les listes sont bien sûr conçues pour que les enchaînements des différentes mesures tirées se fassent naturellement, quelque soient les résultats des tirages. La méthode a eu suffisamment de succès à l'époque pour qu'on l'applique à la conception de petites machines à composer appelées "componium" (figure 8): les listes de mesures sont matérialisées par des cylindres, enfilés côte-à-côte sur un même axe, les tirages étant réalisés en faisant tourner ces différents cylindres les uns après les autres. On est en droit de considérer ces machines comme les premiers prototypes réalisés en informatique musicale.

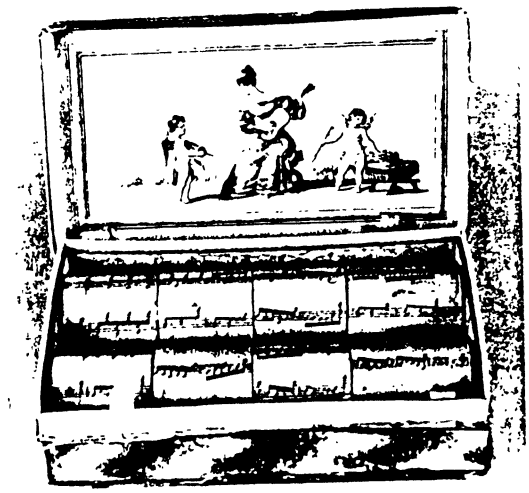


Figure 8. Componium portatif, vers 1830.
Paris. Musée Instrumental du CNSM (E. 1908)

Du point de vue des langages formels, on se trouve en présence de langages obtenus par *factorisation* :

$$L = X_1 \; X_2 \; \ldots \; X_n.$$

Dans l'un de ces jeux musicaux, attribué à Mozart (cote BN : Vm^7 6912, cf aussi [24]), l'auteur suggère également d'utiliser l'*étoile* du langage ci-dessus : "Veut-on avoir un Walzer plus long on recommence de la même manière, et ainsi cela va à l'infini", d'où le langage :

$$L' = (X_1 \; X_2 \; \ldots \; X_n)^*.$$

On obtient des *expressions rationnelles* élémentaires. Les automates finis correspondant à ces langages reconnaissables sont représentés figure 9, ainsi que quelques exemples de premières et de deuxièmes mesures tirées du *Ludus melothedicus*.

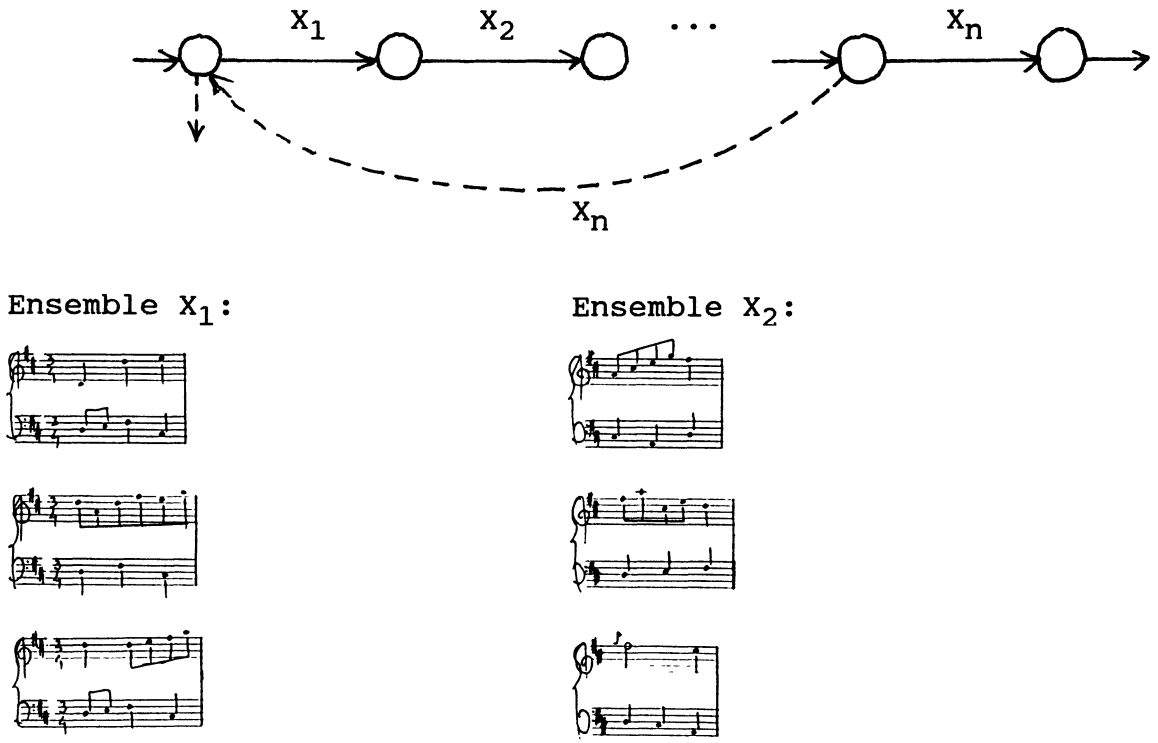


Figure 9.

A dire vrai, l'idée de définir des règles de transitions musicales à l'aide de graphes ou d'automates finis ne s'est pas cantonnée à ces jeux musicaux. On la retrouve à l'oeuvre au XXème siècle dans des pièces de musique savante, comme en témoigne l'extrait de la *3ème Sonate pour piano* de Pierre Boulez (1957) reproduit figure 10.

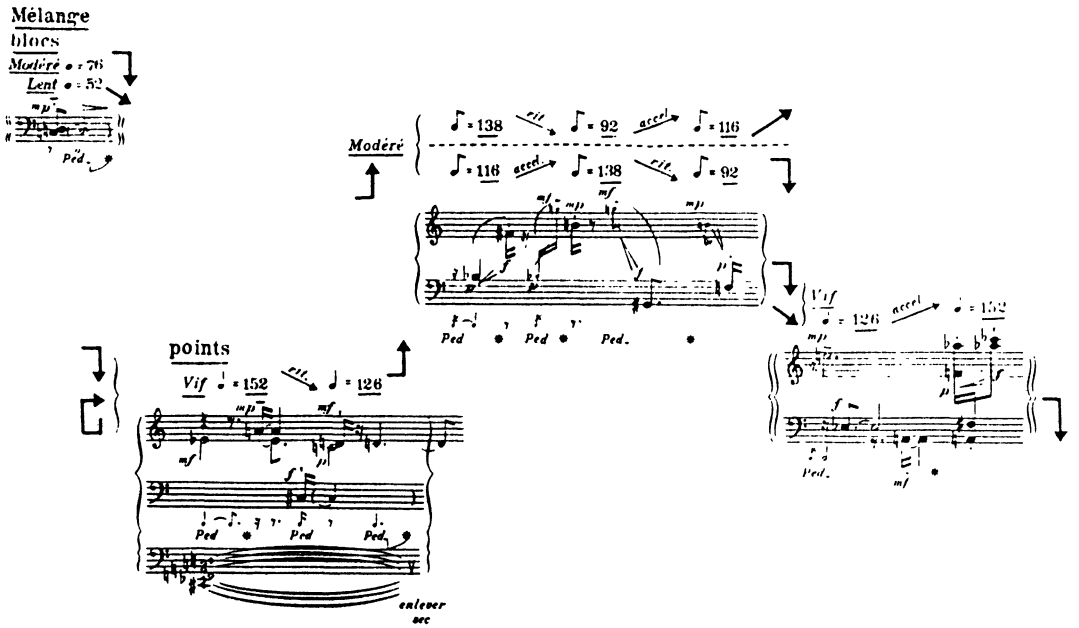


Figure 10. Pierre Boulez, *3ème Sonate* (ed. Universal).

Dans le cas général de la "composition musicale", la part d'automatisme est plus ou moins importante dans l'application de règles pour contrôler les transitions du discours musical. Mais les exemples décrits ci-dessus laissent entrevoir l'intérêt que peut présenter le modèle des *automates finis* pour l'application de l'informatique à la génération musicale.

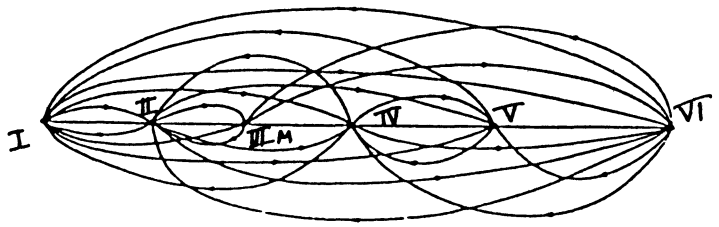
REMARQUE. La *Suite Illiac* de Hiller et Issacson, généralement considérée comme l'acte de naissance de l'informatique musicale (1959, cf. [20]), proposait déjà un traitement automatique des règles du contrepoint traditionnel. L'informatique musicale a depuis suscité de nombreuses recherches sur la possibilité de traiter par ordinateur des ensembles de règles musicales, à partir de modèles formels divers. Mentionnons quelques travaux récents dans ce domaine: Jan Vandenheede étudie les possibilités du langage PROLOG pour "spécifier et gérer des règles musicales de façon confortable" (cf. [26]) ; Kemal Ebcioglu propose un système expert pour harmoniser des chorals dans le style de Bach au moyen de "deux cent règles environ, réparties en plusieurs groupes, dont chacun observe le choral d'un point de vue différent, comme le squelette des accords, les lignes mélodiques des parties individuelles, et la structure hiérarchisée du chant et de la basse" (cf. [16]) ; Christoph Lischka s'intéresse au même problème, d'un point de vue connexioniste (cf. [21]) ; enfin, Bernd Streitberg introduit la notion de "température" d'une séquence de contrepoint comme étant le nombre de violations des règles contrapunctiques dans la séquence (cf. [25]). Pour le problème plus spécifique de l'écriture canonique, Giancarlo Bizzi a proposé un important travail de formalisation fondé sur les "carrés magiques"(cf. [4]).

f. La théorie de Pierre Barbaud.

L'étude systématique des règles élémentaires de la musique tonale au moyen d'un formalisme rigoureux faisant référence aux automates finis, a été introduite au milieu des années soixante dans les deux ouvrages historiques de Pierre Barbaud *Introduction à la composition musicale automatique* (1965, [1]) et *La musique discipline scientifique* (1968, [2]). Pour illustrer l'approche de Pierre Barbaud, plaçons-nous dans la tonalité de DO majeur. On considère dans cette tonalité l'ensemble des accords parfaits que l'on peut construire sur chaque degré de la gamme, et on note par des chiffres romains ces "degrés harmoniques". On évitera cependant

l'accord de quinte diminuée construit sur la sensible et on remplacera l'accord mineur du troisième degré par un accord parfait majeur (noté IIIM) empruntant au ton relatif LA mineur. On obtient la liste : I, II, IIIM, IV, V, VI, avec I = {do, mi, sol}, etc. On s'intéresse alors aux séquences harmoniques formées par des accords de cette liste, supposés tous à l'état fondamental. Les transitions sont données par un automate fini comme celui de la figure 11 (cf. [2] p. 56). La matrice de transition indique pour chaque accord l'ensemble des accords qui peuvent lui succéder, au moyen de 1 placés dans la ligne correspondante. Voici un exemple de séquence harmonique obtenue par l'automate ci-dessous :

I IV VI IIIM VI V I IV V I.



	I	II	III M	IV	V	VI
I	0	1	1	1	1	1
II	1	0	1	1	1	1
III M	0	1	0	0	0	1
IV	1	1	0	0	1	1
V	1	0	0	1	0	0
VI	1	1	1	1	1	0

Figure 11.

Pour obtenir les lignes mélodiques de la polyphonie, on part d'une position arbitraire pour le premier accord de la séquence, et on construit les positions des accords suivants au moyen de la "règle des indices" (cf. [2] p. 51). La polyphonie est supposée à quatre voix, et la fondamentale de chaque accord est doublée.

1) Si la basse monte d'une seconde, d'une tierce, ou d'une quarte, alors la fondamentale, la tierce, et la quinte du premier accord sont suivies respectivement de la quinte, la fondamentale, et la tierce de l'accord suivant.

2) Si la basse descend d'une seconde, d'une tierce, ou d'une quarte, alors la fondamentale, la tierce, et la quinte du premier accord sont suivies respectivement de la tierce, la quinte, et la fondamentale de l'accord suivant.

L'exemple de la figure 12 donne tous les enchaînements possibles à partir du II, pour une position initiale comprenant, du grave à l'aigu: fondamentale, quinte, fondamentale, tierce.

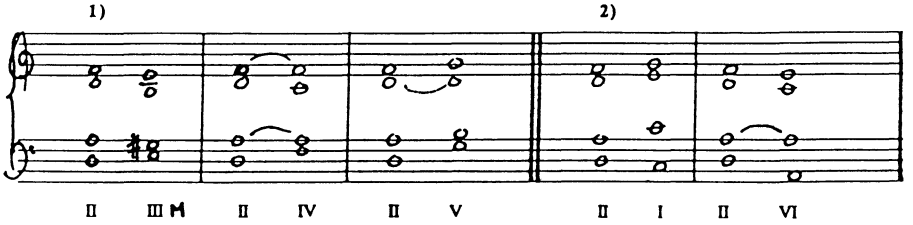


Figure 12.

L'automate de la figure 11 n'est pas déterministe, c'est-à-dire qu'à un même accord peuvent succéder plusieurs accords distincts. Pour introduire un critère de choix entre ces différentes possibilités, Pierre Barbaud remplace la matrice de transition par une matrice stochastique donnant des probabilités d'apparition pour les successeurs d'un accord donné. "Nous pouvons ainsi espérer, en modifiant la matrice stochastique correspondant aux probabilités d'apparition des éléments de notre univers harmonique, canaliser le hasard vers des «habitudes», des «tics», peut-être vers un certain style harmonique, en un mot simuler mieux la réalité vivante qu'en s'en tenant, pour chaque mot du discours, à une équiprobabilité d'apparition" ([1] p. 9). La figure 13 donne une matrice stochastique pour le même automate (cf. [2] p. 57).

	I	II	III M	IV	V	VI
I	0	0.20	0.02	0.40	0.25	0.13
II	0.14	0	0.50	0.01	0.30	0.05
III m	0	0.27	0	0	0	0.73
IV	0.24	0.02	0	0	0.72	0.02
V	0.77	0.02	0.01	0.10	0	0.10
VI	0.01	0.90	0.04	0.03	0.02	0

Figure 13.

Dans l'exemple décrit précédemment, on observe que les transitions mélodiques sont déterminées par la "règle des indices", *une fois établie* la succession harmonique de la séquence. Ainsi, la difficulté bien connue des constructions polyphoniques tonales, qui est de concilier les impératifs verticaux (choix des accords et de leurs enchaînements) et les impératifs horizontaux (choix des enchaînements mélodiques), a été surmontée en traitant *d'abord* les premiers, et *ensuite* les seconds. Le point de vue de Pierre Barbaud est en quelque sorte une transposition algorithmique du principe exposé par Rameau : "Si l'on voit d'abord dans la résonance des corps sonores d'où nous vient le sentiment naturel de l'harmonie, on verra de même par les sons communs à l'harmonie des différents sons fondamentaux qui se succèdent immédiatement d'où nous vient celui de sa succession, appelée mélodie dans chaque partie en particulier" (Jean-Philippe Rameau, *Observations sur notre instinct pour la musique, et sur son principe*, 1754, [22] p. 165).

L'algorithme de contrepoint automatique qui est présenté dans la section suivante a la particularité de traiter de manière strictement équivalente les règles harmoniques et les règles mélodiques. C'est en effet le théorème énoncé ci-dessus, concernant la reconnaissabilité de la superposition de parties reconnaissables, qui permet de combiner des automates reconnaissant différentes mélodies, en un automate reconnaissant la polyphonie constituée par ces mélodies, et qui rend ainsi équivalentes les deux dimensions du problème de construction musicale traité. L'algorithme est de plus entièrement déterministe, c'est-à-dire non probabiliste, et calcule *toutes les séquences* qui satisfont aux règles introduites (à condition que ces règles puissent être schématisées par des automates finis). Dans la terminologie des systèmes experts, l'algorithme se présente donc comme un *moteur d'inférence* qui agit sur une *base de données* (par exemple les tessitures des instruments ou des voix employés, les motifs choisis, etc.), et sur une *base de règles*, que l'on peut modifier à loisir.

g. Contrepoint automatique.

L'algorithme proposé repose sur deux résultats théoriques de reconnaissabilité par automate fini. Le premier est le résultat de la section e, concernant la *superposition à longueurs égales* de parties reconnaissables (théorème 1). Le second, qui est un résultat élémentaire de la théorie générale des automates, concerne l'*intersection* de parties reconnaissables (cf. [18] p. 17) :

THÉORÈME 3. *Si* X *et* Y *sont des parties reconnaissables par automate fini, alors l'intersection* $X \cap Y$ *est aussi reconnaissable par automate fini.*

DÉMONSTRATION. Soient $\mathbf{A} = (Q, i, T)$ et $\mathbf{A'} = (Q', i', T')$ deux automates reconnaissant respectivement X et Y. L'automate $\mathbf{A_1} = (Q \times Q', (i, i'), T \times T')$ reconnaît $X \cap Y$, avec les flèches :

$a : (p, p') \rightarrow (q, q')$

si l'on a les flèches $a : p \rightarrow q$ dans **A**, et $a : p' \rightarrow q'$ dans **A'**. □

La construction d'un automate $\mathbf{A_2}$ reconnaissant $X \pi Y$ est à peu de choses près la même (algorithme de Dan Timis). On formera les flèches :

$a \cup b : (p, p') \rightarrow (q, q')$

si l'on a les flèches $a : p \rightarrow q$ dans **A**, et $b : p' \rightarrow q'$ dans **A'**.

Pour simplifier l'algorithme de contrepoint automatique, on se placera dans le cas particulier associé aux définitions suivantes. On dira que w est une *mélodie* sur E si w appartient à $(E+\emptyset)^*$. On dira que w est une *polyphonie stricte à n voix* sur E s'il existe une partition $E_1, E_2, \ldots E_n$ de E et n mélodies $w_1, w_2, \ldots w_n$ sur les E_i telles que $w = w_1 \| w_2 \ldots \| w_n$. Si $E \cap F = \emptyset$, et si w est une polyphonie stricte à n voix sur E et w' une polyphonie stricte à m voix sur F, alors $w \| w'$ est une polyphonie stricte à n+m voix sur $E \cup F$. On supposera désormais que les ensembles X et Y intervenant dans les résultats ci-dessus sont des polyphonies strictes sur des ensembles disjoints. Dans ce cas, si les automates **A** et **A'** sont déterministes, alors l'automate $\mathbf{A_2}$ reconnaissant $X \pi Y$ est aussi déterministe. Remarquons que l'automate $\mathbf{A_1}$ est lui aussi déterministe, dès que **A** et **A'** le sont.

Pour illustrer le principe de l'algorithme, étudions un exemple de contrepoint à deux voix soumis à deux règles, l'une mélodique, l'autre harmonique. Les deux ensembles d'événements sont E = {fa, sol, la} pour la voix A, et F = {do, ré, mi} pour la voix B. Chacune des voix A et B est soumise à une règle mélodique, dont on a donné une représentation ci-dessous sous forme d'automate. Pour tracer l'automate de la figure 14 (voix B), il est plus commode de se placer dans un sous-monoïde *libre* du monoïde de départ $(F+\emptyset)^*$: on considère le code $K = (F+\emptyset)(F+\emptyset)$, et on forme le sous-monoïde K^*.

RÈGLE 1 (mélodique). Chacune des voix ne peut admettre que deux rythmes (la blanche et la noire). Le mouvement des blanches est libre, mais les noires sont soumises au mouvement conjoint, par rapport à la note précédente et par rapport à la note suivante.

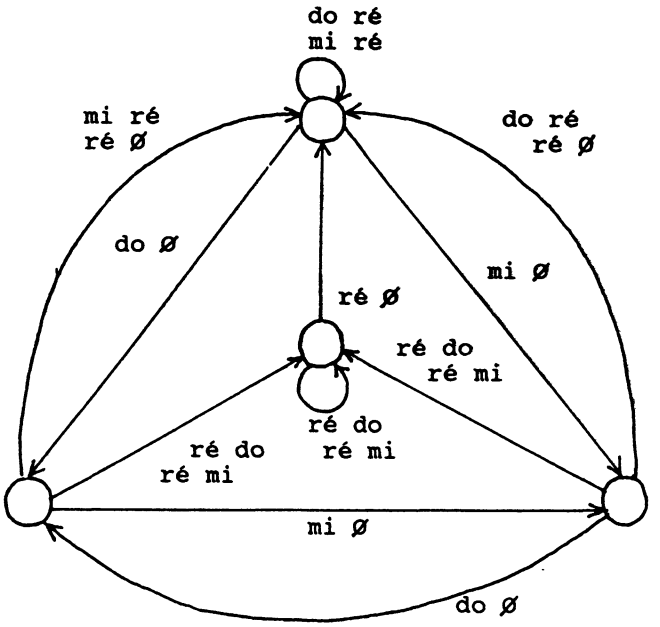


Figure 14. Règle 1, mélodique (voix B).

Pour le contrepoint à deux voix (A, B), on impose une règle harmonique dont l'automate est donné figure 15. Si P est l'ensemble des blocs qui peuvent apparaître dans la polyphonie, on définit un sous-ensemble C d'éléments de P qui sont appelés *consonances*. La règle ci-dessous définit le langage (CP)*.

RÈGLE 2 (harmonique). On doit trouver des consonances sur les temps forts, les dissonances étant autorisées sur les temps faibles.

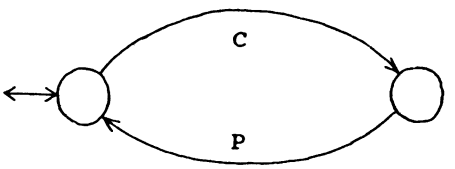


Figure 15. Règle 2, harmonique (polyphonie (A, B)).

Si M_A et M_B sont les ensembles de mélodies sur E et F satisfaisant à la règle mélodique 1, et si H est l'ensemble des polyphonies à deux voix satisfaisant à la règle harmonique 2, alors l'ensemble des séquences musicales obtenues en appliquant ces deux règles peut s'écrire $(M_A \pi M_B) \cap H$, et les théorèmes 1 et 3 permettent de construire l'automate reconnaissant cet ensemble. Les étapes de l'algorithme sont représentées dans l'organigramme de la figure 16.

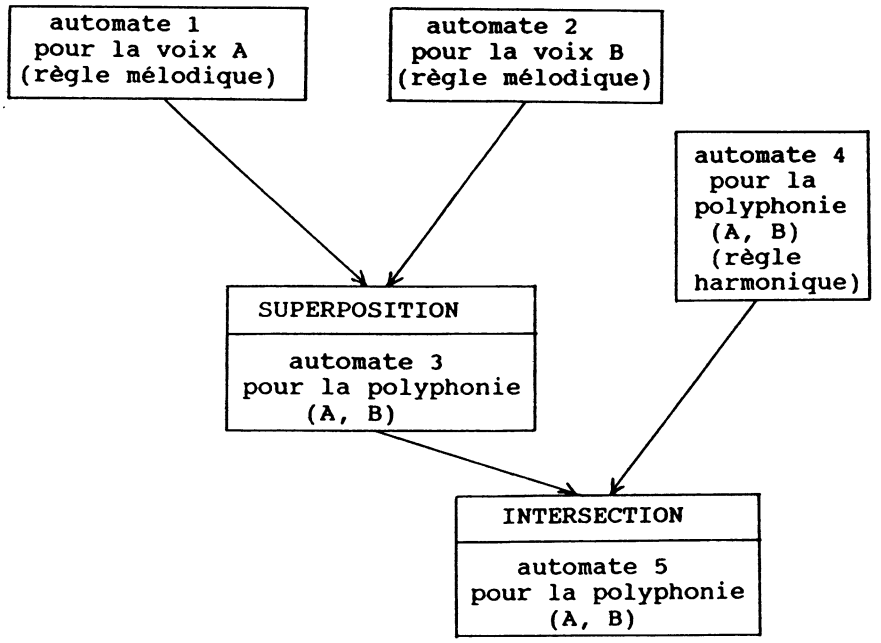


Figure 16.

ALGORITHME. Dans le cas général, l'algorithme se décompose en deux types de calculs. On suppose que la polyphonie étudiée est stricte.

1) Si l'on a plusieurs règles concernant des ensembles disjoints de voix de la polyphonie, on construit au moyen de la *superposition* un automate concernant la polyphonie toute entière.

2) Si l'on a plusieurs règles concernant la polyphonie toute entière, on les combine au moyen de l'*intersection.*

Notons que l'on peut également introduire des règles qui ne concernent ni la polyphonie toute entière, ni un sous-ensemble de voix, mais qui concernent la succession harmonique. Il suffit pour cela de définir un morphisme de "chiffrage harmonique", de P* ensemble des polyphonies étudiées, dans D* où D est l'ensemble des "degrés harmoniques". L'image réciproque d'une partie reconnaissable de D* sera une partie reconnaissable de P*.

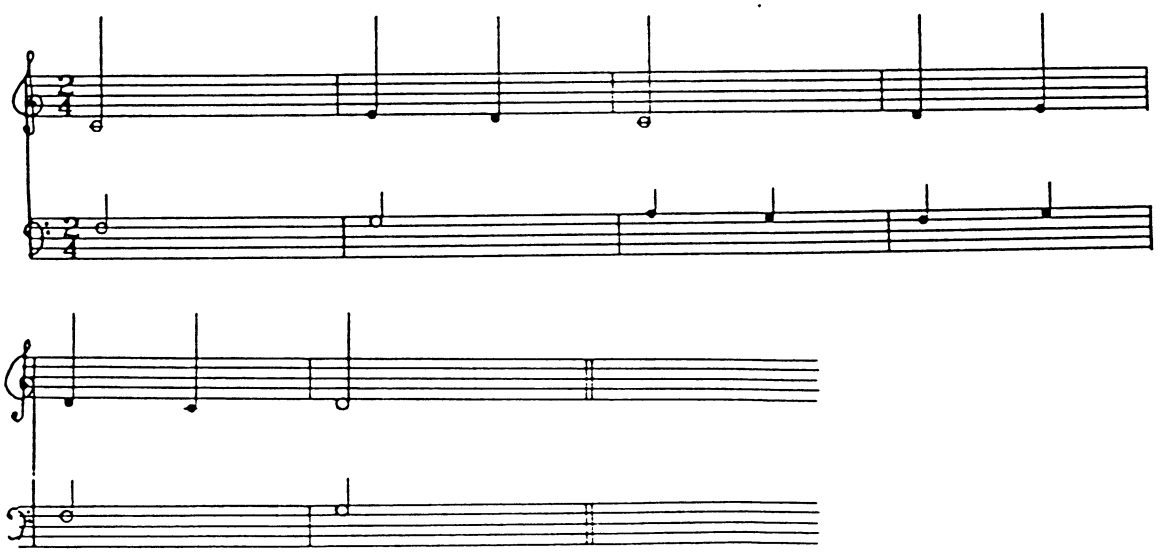


Figure 17.

La figure 17 montre une séquence reconnue par l'automate construit au cours de l'exemple exposé dans cette section[2]. Cet exemple bien que rudimentaire, montre comment l'algorithme permet de combiner des ensembles arbitraires de règles musicales. Plus précisément, il transforme une définition "équationnelle" (sous forme d'une liste de contraintes) de l'ensemble des séquences musicales cherchées, en une définition "paramétrique" équivalente (le paramètre étant le chemin dans l'automate obtenu, qui définit une séquence correcte, et qui peut être choisi arbitrairement). Il y a donc bien *résolution*, au sens que l'on donne généralement à ce mot en mathématiques.

S'il n'y a pas de limites théoriques au nombre d'application des deux opérations de base de l'algorithme, on se trouve cependant confronté à des difficultés de capacités de machine dues au problème suivant: l'automate construit pour la superposition ou l'intersection de parties reconnaissables *n'est en général pas minimal*, même si les deux automates de départ le sont. Il est donc nécessaire de recourir à une procédure de minimisation d'automate (cf. [5]) chaque fois qu'une opération de base est effectuée.

BIBLIOGRAPHIE

[1] BARBAUD P., *Introduction à la composition musicale automatique*, Paris, Dunod, 1965.

[2] BARBAUD P., *La musique discipline scientifique*, Paris, Dunod, 1968.

[3] BERARD B., *Shuffle littéral, étude formelle et applications*, thèse, Univ. Paris 7, LITP 85-30, 1985.

[4] BIZZI G., *Miroirs invisibles des sons. La construction des canons: réponse à une énigme*, trad. A.-L. DEBELLEMANIERE, Annales littéraires de l'Université de Besançon, vol. 342, Les Belles Lettres, 1986.

[5] CARDON A., M. CROCHEMORE, "Partitioning a graph in o(|A| $\log_2$|V|)", *Theo. Comp. Sci.*, vol. 19 (1982), 82-98.

[6] CARTIER P., D. FOATA, *Problèmes combinatoires de commutation et de réarrangements*, Lecture Notes in Math. n° 85, Springer Verlag, 1969, 8-17.

[7] CHEMILLIER M., *Contrepoint et informatique*, rapport de stage à l'IRCAM, Nov.-Déc. 1986.

[8] CHEMILLIER M., "Monoïde libre et musique", *RAIRO Inf. Theo.*, vol. 21, n° 3 et 4 (1987), 341-371 et 379-417.

[9] CHEMILLIER M., D. TIMIS, "Toward a theory of formal musical languages", *Proc. of the ICMC 88*, Cologne, 1988, 175-183.

[10] CHEMILLIER M., "Langages musicaux et automates : la rationalité du langage sériel", *Actes du Colloque "Structures musicales et assistance informatique"*, MIM, 36 bd Pardigon 13004 Marseille, octobre 1990, (à paraître).

[11] CHEMILLIER M., *Structure et méthode algébriques en informatique musicale*, thèse, Université Paris 7, LITP, 90-4, 1990.

[2] Cet exemple a été tracé par un logiciel expérimental implémenté en LISP à partir de l'algorithme, au cours d'un travail à l'IRCAM en Novembre-Décembre 1986, avec la collaboration de Dan Timis et Gérard Assayag (cf. [7]). La notation musicale est obtenue par une fonction DRAWMEL mise au point par Gérard Assayag (informaticien membre fondateur du CRIME, Collectif de Recherche en Informatique Musicale Etc.).

[12] CORI R., D. PERRIN, "Automates et commutation partielle", *RAIRO Inf. Theo.*, vol. 19, n° 1 (1985), 21-32.

[13] DE SIMONE R., "Langages infinitaires et produit de mixage", *Theo. Comp. Sci.*, vol. 31 (1984), 83-100.

[14] DUBOC C., *Commutation dans les monoïdes libres*, thèse, LITP 86-25, 1986.

[15] DUBOC C., *Mixed product and the asynchronous automata*, LITP 86-60, 1986.

[16] EBCIOGLU K., "An expert system for schenkerian synthesis of chorales in the style of J.S. Bach", *Proc. of the ICMC 84*, Paris IRCAM, 1984, 135-142.

[17] EBCIOGLU K., "An expert system for harmonizing four-part chorales", *Comp. Mus. J.*, vol. 12, n° 3 (1988), 43-51.

[18] EILENBERG S., *Automata, languages and machines*, vol. A, Academic Press, 1974.

[19] GREUSSAY P., *Modèles de descriptions symboliques en analyse musicale*, thèse, Univ. Paris 8, 1973.

[20] HILLER L., *Expérimental music composition with an electronic computer*, New York, McGraw Hill, 1959.

[21] LISCHKA C., "Connectionist models of musical thinking", *Proc. of the ICMC 87*, Urbana-Champaign, 1987, 190-196.

[22] PERRIN D., "Words over a partially commutative alphabet", *Combinatorial algorithms on words*, ed. Apostolico et Galil, NATO-ASI series, 329-340, Springer Verlag, 1984.

[23] RAMEAU J.-P., "Observations sur notre instinct pour la musique, et sur son principe", 1754, *Musique raisonnée*, Paris, Stock, 1980, 145-200.

[24] RIOTTE A., "Une jonction nouvelle entre art et science: l'informatique musicale", *Euro-spectra*, 1974, *Musique et ordinateur*, Ed. Centre Exper. Spectacle, 1983, 106-119.

[25] STREITBERG B., K. BALZER, "The sound of mathematics", *Proc. of the ICMC 88*, Cologne, 1988, 158-165.

[26] VANDENHEEDE J., "Expériences musicales avec PROLOG II", *IRCAM Actualité de la Rech.*, Mai 1986.

[27] VANDENHEEDE J., "Musical experiments with PROLOG II", *Proc. of the ICMC 86*, La Haye, 1986, 5-10.

[28] VIENNOT G., "Problèmes combinatoires posés par la physique statistique", *Astérisque*, n° 121-122 (1985), 225-246.

[29] VIENNOT G., "Heaps of pieces, I: basic definitions and combinatorial lemmas", *Proc. "Combinatoire énumérative"*, Montréal, 1985, Lecture Notes in Mathematics n° 1234, 321-350, Springer Verlag, 1985.

[30] XENAKIS I., *Musiques formelles*, Richard-Masse, 1963.