Grammaires de substitution harmonique dans un improvisateur automatique

Mémoire de stage du D.E.A. A.T.I.A.M.

Equipe Représentations Musicales IRCAM CNRS UMR 9912

Responsables du stage : Marc Chemillier et Emmanuel Saint-James

30 juin 2004

Jean-Brice GODET

Table des matières

Remerciements

Je remercie d'abord et évidemment les membres de l'équipe de Représentations Musicales pour leur accueil et leur bonne humeur : Gérad Assayag pour son hospitalité, Carlos Agon pour son HDR, Karim Haddad pour son humour anglais.

Je remercie particulièrement Marc Chemillier et Emmanuel Saint-James pour leurs conseils, leur soutien et la confiance qu'ils m'ont accordés dans les directions à prendre pour ce travail.

Je remercie Benoît Meudic qui m'a ouvert les portes de son bureau et m'a aidé à me familiariser avec certaines subtilités informatiques et Olivier Lartillot, prédécesseur prestigieux, qui m'a laissé son ordinateur et donc son contenu.

Je remercie l'équipe enseignante du D.E.A., et Cyrille Defaye pour leur patience quotidienne pendant cette année.

Merci aux étudiants évidemment :
Carl pour ses cours de Latex et sa capacité à répondre à la même question idiote dix fois d'affilées, Cyril pour sa présence et sa disponibilité, Arshia pour mes poumons, Matthias pour ses jeux de mots, et puis tous les autre pour pleins de raisons qui tiennent du partage et de la générosité.

Merci aussi et surtout à Juliette, Daniel et Faustine pour leur soutien et pour le reste...

Sans oublier Léo, Marion, Edwige, Paul, Alexandre, Sébastien, Eric, Ariana, Artur, Marc-Antoine, Charlotte, Olivier, Fred, Louis, Nicolas et tous ceux que j'ai croisés cette année.

Chapitre 1

Introduction

"L'improvisation est une forme de clandestinité, de résistance" — B.Lubat
(in *L'improvisation musicale* p. 267)

1.1 L'improvisation : problématique contemporaine ?

Improvisation,
le mot est lâché. Mais qu'est-ce que l' improvisation ?

Il n'y a pas de réponse unique et la bonne question serait plutôt : Qu'est ce qu' une improvisation ?

Une composition en temps-réel... Un geste artistique spontané... Un moyen pour le musicien de s'émanciper du diktat de l'écrit ou de contraintes sociales(cf.[1])... Un partage lorsqu'elle est collective... Une voie musicale alternative ?

Aucune de ces réponses ne semble satisfaisante en tant que telle. Elles ne sont que des pistes de réflexions qui peuvent nous éclairer pour mieux cerner cette pratique artistique.

L'improvisation est avant tout dépendante d'un contexte culturel.

Comme le montre Derek Bayley dans [2], l'improvisation est une pratique musicale qui s'inscrit dans un contexte socioculturel. Il expose dans son livre des entrevues des organistes improvisateur, des musiciens de jazz, de flamenco, de culture indienne classique, de rock progressif... et montre comment chacun a une vision radicalement différente de son art et de sa propre pratique.

Le musicien indien s'inscrit par exemple dans un tradition séculaire fondée sur la transmission orale du savoir. Ses improvisations consistent en d'infimes variation de mélodies ou de rythmes transmis par des maîtres. Il témoigne cependant que son art n'est pas figé : chaque maître ou musicien ne transmet pas que ce qu'il a appris mais aussi une part de sa propre expérience.

Le musicien de jazz envisage l'improvisation comme un choix judicieux de notes en fonction d'un contexte harmonico-mélodico-rythmique et de la "direction" qu'il souhaite donner à son improvisation.

Le musicien de flamenco insiste quant à lui sur l'importance du développement de sa propre personnalité musicale en respectant une tradition bien précise.

Devant la diversité qu'offre la pratique de l'improvisation, il est bien difficile de cerner une définition unique de l'improvisation.

De la contemporanéité de l'improvisation.

L'une des faiblesses de l'improvisation est aussi sa caractéristique principale, elle est éphémère et vit dans l'instant. C'est sans doute pourquoi d'aucuns la considèrent comme un art mineur. Dans la culture occidentale classique, l'improvisation est réservée aux organistes dans un mimétisme au mythe fondateur, Bach.

On oublie trop souvent que l'improvisation est une pratique aussi ancienne que la musique et qu'elle n'est réservée ni à une classe d'instrument, ni à une classe sociale. L'improvisation n'est pas une problématique contemporaine au sens strict du terme, elle est une pratique participante de l'histoire de la musique.

Le premier homme a avoir joué un rythme ne l'a sans doute pas lu sur les parois d'une grotte. Le chant grégorien était pour l'essentiel improvisé sur un canevas. Plus près de nous Beethoven, Liszt et Chopin étaient outre des compositeurs, des improvisateurs remarquables. L'avènement du jazz, qui en a fait l'une de ses principales caractéristiques, ainsi que l'apparition des supports enregistrés permettent à l'improvisation de prendre une part croissante dans le paysage musical de la fin du XX^{me} siècle. Il existe aujourd'hui un courant très important de "musiques improvisées" qui regroupent des virtuoses et des compositeurs de talent.

Si elle n'est pas une problématique contemporaine au sens strict, l'improvisation fait aujourd'hui partie intégrante de la création offrant des productions qui n'ont pas à rougir de celles de la musique écrite.

1.2 Intégration de l'informatique dans l'improvisation : le projet O'Max

"En hommage au Macintosh Plus [...] et plus généralement encore à l'informatique musicale, passagère clandestine mais opérante [...] dans cette traversée capricieuse" — J.L.Chautemps
(Notes de pochettes du disque *# 06*, 1988, Universal Music)

Depuis quelques années, on assiste à une utilisation de plus en plus massive de l'informatique musicale dans des contextes d'improvisation. Cela est du en partie à l'explosion de l'utilisation des ordinateurs portables et aux performances accrues des logiciels temps-réel. La possibilité de déplacer son matériel a

transformé la machine en instrument à part entière, augmentant la convivialité et les contextes possibles de son utilisation.

Mené au sein de l'équipe Représentations Musicales de l'I.R.C.A.M., le projet O'Max est porté par la collaboration de deux chercheurs : Gérard Assayag et Marc Chemillier. Il consiste en la construction d'un improvisateur automatique utilisant deux paradigmes de l'informatique musicale : celui de Max (cf. "Systèmes réactifs" Carl seleborg 2004) et celui d'Open Music (cf. [4]). La partie "Temps-Réel" étant dédiée à Max alors que la partie concernant le traitement symbolique est laissée à Open Music.

Dans le fonctionnement d'O'Max, deux processus sont en jeux, le premier est un algorithme d'apprentissage, l'oracle des facteurs, le second est un processus de substitution d'accords, basé sur les grammaires formelles. Le sujet de mon stage porte sur le deuxième aspect.

Lors de l'utilisation d'O'Max, il est possible de modifier l'accompagnement en temps-réel en substituant des accords dans la grille harmonique qui fait partie de l'accompagnement. Le problème principal est la taille de la boucle d'interaction : il faut attendre la réalisation d'une grille complète avant que les modifications soient effectives. Notre travail a donc consisté à mieux comprendre les structures des modifications pour pouvoir imaginer un mécanisme plus interactif.

1.3 Contenu du mémoire

Nous commençons par rappeler le formalisme de la théorie des langages et des automates, base indispensable pour la clarté de l'exposé. Puis nous exposons la grammaire de steedman, qui est une grammaire dépendante du contexte, et son formalisme. Nous montrons les outils que nous avons du inventer pour explorer le langage généré par cette grammaire.

Nous proposons ensuite un formalisme qui nous est apparu comme plus rigoureux, même s'il est moins intuitif, qui permet de décrire les séquences harmoniques dérivées par Steedman. A l'aide de ce formalisme et grâce à l'étude précédente, nous avons établi deux résultats importants :

- La "décontextualisation" de la grammaire de Steedman.
- Deux propriétés formelles fondamentales sur le langage généré.

Le second résultat devrait trouver une application directe dans le projet O'Max.

Chapitre 2

Théorie des langages et des automates

"Mathematics is the language of nature. Everything around us can be represented and understood through numbers. If you graph the numbers of any system, patterns emerge. Therefore, there are patterns everywhere in nature."
(Max Cohen in *Pi*,1997, written and directed by Darren Aronofsky)

Dans cette partie nous nous attachons à présenter de façon claire et concise les principaux concepts de la théorie des grammaires formelles.

Nous rappelons tout d'abord en introduction le contexte linguistique des débuts de cette théorie, puis nous présentons une série de définitions sur les grammaires formelles et les catégories de Chomsky, enfin nous proposons une sorte de résumé de l'article de Mark J. Steedman "A Générative Grammar for Jazz Sequence" point de départ de mon travail pour ce stage.

2.1 Introduction en forme de rappel historique

La théorie des langages et des automates est apparue au milieu des années 1950 sous l'impulsion de Noam Chomsky. Elle s'est développée en liaison avec les théories de l'information de Shannon et de la théorie des machines abstraites de Turing.

Elle a tout d'abord été développée dans le cadre d'une tentative de modélisation des langues naturelles et plus particulièrement en vue d'une formalisation des analyses syntaxiques de phrases. Par exemple, pour une phrase globalement notée S, pour Sentence, on décompose la structure en un groupe nominal et un groupe verbal, et on écrit :

$$S \rightarrow (GN)(GV)$$

Le groupe verbal peut-être à son tour décomposé en groupes différents :

verbe transitif + groupe nominal objet + autres groupes nominaux compléments, et on écrit :

$$(GV) \rightarrow (VT)(GN)$$

Le groupe nominal peut lui aussi être décomposé en :

$$(GN) \rightarrow (Art)(Nom)(Dt)(Dt)$$

où (Art) est un article, (Dt) est un déterminatif. Un déterminatif peut être par exemple un adjectif ou une proposition relative.

Nous voyons ainsi apparaître des règles dites de *grammaires formelles*, dont les *signes non-terminaux* sont $S, (GN), (GV), (VT) \ldots$ et dont les *signes terminaux* sont les mots du lexique.

Cette manière de décomposer la façon dont est construite une phrase est pertinente dans de nombreux cas simples, voire simplistes. Mais cela n'est pas tenable en général. Dans la langue française, par exemple, il y a des accords. Lors de la génération de la phrase, il faudra accorder le verbe selon la structure du groupe nominal sujet. Cela induit par conséquence des dépendances du contexte et une complexification des règles souvent difficile à formaliser.

Cette approche est maintenant essentiellement abandonnée, sauf pour analyser les phrases les plus simples, parce qu'elle ne rend pas bien compte des règles lexicales.

Malgré les insuffisances de cette théorie pour l'étude des langues naturelles, elle connaît une expansion considérable depuis que l'on s'est aperçu de son adéquation à la description des langages de programmation : les concepts, issus de la théorie des langages, et tout particulièrement des langages algébriques, de grammaire, de dérivation, d'automate à pile, sont à la base de tous les algorithmes d'analyse syntaxique, et partant des compilateurs. La théorie des langages constitue donc l'un des fondements de la science informatique.

2.2 Grammaires formelles

2.2.1 Quelques définitions

Les définitions suivantes sont adaptées de [3].

Definition 1 (Alphabet):
Un alphabet est un ensemble fini V dont les éléments sont appelés lettres ou signes.

Definition 2 (Mot):
Un mot sur V est une suite finie d'éléments de V

Exemple 1:
Soit $V = \{a, b, c\}$, alors $a, aa, abcba$ sont des mots sur V.

Remarques :

1) Etant donné deux mots, on peut les juxtaposer par une opération de concaténation. La notation a^n désigne le mot $a...a$ qui contient n symboles a.

2) Par convention, on définit le mot de longueur nulle appelé le mot vide et noté ε, qui est l'élément neutre pour l'opération de concaténation. Ainsi $a\varepsilon bc\varepsilon ab = abcab$.

3) Nous noterons V^*, l'ensemble de tous les mots sur V, y compris le mot vide noté ε.

4) Un mot u est noté comme la succession de ses lettres : $u = u_1...u_n$

Definition 3 (Grammaire):
Une grammaire formelle *est la donnée d'un quadruplet* $G = (s, N, T, R)$ *où :*
- s *est l'axiome.*
- T *est l'alphabet terminal.*
- N *est l'alphabet non-terminal.*
- R *est un sous-ensemble fini de* $(N \cup T)^* \times (N \cup T)^*$ *dont les éléments sont appelés les règles de la grammaire (ou système de réécriture).*

On suppose de plus que T *et* N *sont disjoints.*

Remarques :
1) $s \in N$. L'axiome est un élément de l'alphabet non-terminal.

2) Les règles de la grammaire sont des couples que nous noterons par convention : $X \to Y$ où $X \in (N \cup T)^*$ et $Y \in (N \cup T)^*$.

2.2.2 Notions de dérivation (de réécriture ou de déduction)

Definition 4 (Dérivation ou réécriture ou déduction):
Soit G *une grammaire ;* X *et* Y *des mots sur* $(N \cup T)^*$.

Nous dirons que Y *peut-être déduit ou dérivé de* X *par la grammaire* G *en une étape si on peut écrire* $X = X_1PX_2$, $Y = X_1QX_2$ *et* $P \to Q$ *est une règle de* G.

Ceci signifie que X contient un sous mot P et que Y est obtenu à partir de X en remplaçant P par Q, en utilisant la règle $P \to Q$ de la grammaire G, sans toucher aux autres parties de X.

Dans ce cas nous écrirons :

$$X \xrightarrow[R]{1} Y$$

Le 1 signale que la dérivation est en une étape c'est à dire qu'une règle de la grammaire a été appliquée une fois sur le mot X pour obtenir Y.

Cela veut dire que nous avons :

$$X_1PX_2 \xrightarrow[R]{1} X_1QX_2 \text{ avec } P \to Q \text{ règle de } G.$$

Exemple 2:
$T = \{a, b\},\ N = \{s, A, B\},$ et $R_1 = \{s \to asb, s \to ab\}$

Alors on a :

$$asb \xrightarrow[R_1]{1} aasbb$$

Exemple 3:
$T = \{a, b\},\ N = \{s, A, B\},$ et $R_2 = \{s \to ABs, s \to BAs, A \to a, B \to b, s \to \varepsilon\}$

Remarquons qu'alors on a :

$$BAs \xrightarrow[R_2]{1} BAABs$$

Definition 5 (Dérivation en n étapes):
Soit G une grammaire ; X et Y des mots sur $(N \cup T)^*$.

Nous dirons que le mot Y sur $(N \cup T)^*$ est déduit ou dérivé du mot X sur $(N \cup T)^*$ en n étapes par R, si on a une suite de mots $X_2, \ldots, X_{n-1}$ sur $(N \cup T)^*$ avec une suite de dérivations en 1 étape joignant X_{i-1} à X_i et X_{n-1} à Y :

$$X \xrightarrow[R]{1} X_1 \xrightarrow[R]{1} X_2 \to \cdots \to X_k \xrightarrow[R]{1} X_{k+1} \to \cdots \to X_{n-1} \xrightarrow[R]{1} Y$$

Nous passons ainsi de X à X_1 par application d'une règle de R, ..., de X_k à X_{k+1} par application d'une règle de R , ... , et de X_{n-1} à Y par application d'une règle de G.

Nous convenons alors de dire qu'il existe une dérivation ou une déduction par la grammaire R de Y à partir de X.

Soit en notation abrégée :

$$X \xrightarrow[R]{n} Y$$

Remarque :
Une dérivation est dite directe si n=1.

2.2.3 Langage Engendré par une Grammaire

Definition 6 (Langage engendré):
Le langage engendré par la grammaire G est l' ensemble $L(G)$ des mots M de T^*, donc des mots en lettres terminales, tels qu'il existe une dérivation permettant d'obtenir M à partir de s par application des règles de réécriture R de G. Ou encore avec des notations standards $L(G)$ est l'ensemble des mots M tels que $\exists\ n \geq 1, s \xrightarrow[R]{n} M$.

Exemple 4:
Soit $G_1 = (s, N, T, R)$ définit comme précédemment :
$T = \{a, b\},\ N = \{s, A, B\},$ et $R_1 = \{s \to asb, s \to ab\}$
Alors le langage engendré par G_1, $L(G_1)$ est :

$$L_1 = \{a^n b^n, n \geq 1\}$$

Démonstration :
Montrons-le par double inclusion :

$L_1 \subset L(G_1)$ $ab \in L(G_1)$ par la règle $s \to ab$, et $a^n b^n$ pour $n \geq 1$ est obtenu par la dérivation :

$$s \xrightarrow[G_1]{1} asb \xrightarrow[G_1]{1} a^2 sb^2 \to \cdots \to a^{n-1} sb^{n-1} \xrightarrow[G_1]{1} a^n b^n$$

en appliquant la règle $s \to asb$, $n-1$ fois et la règle $s \to ab$ à la fin.

$L_1 \supset L(G_1)$ Soit M un mot de $L(G_1)$. Par définition, on a une dérivation :

$$s \xrightarrow[G_1]{1} M_1 \xrightarrow[G_1]{1} M_2 \to \cdots \to M_{n-1} \xrightarrow[G_1]{1} M$$

Ici M est un mot sur T, donc il ne contient pas s.

Si la dérivation est de longueur 1, c'est $s \xrightarrow[G_1]{1} M$ et la seule possibilité puisque M ne contient pas s, est d'avoir appliqué la règle $s \to ab$, donc $M = ab$.

Si la dérivation est de longueur $n > 1$, la seule possibilité est que l'on applique $s \to ab$ à la dernière étape : $Mn - 1 \to M$. Tous les mots intermédiaires $M, \ldots, M_{n-1}$ contiennent s et donc toutes les dérivations $M_k \to M_{k+1}$ sont obtenues par application de $s \to asb$. Il est alors facile de voir par récurrence sur k que $M_k = a^k sb^k$ et donc $M = a^n b^n$.

Remarques :
1) Le langage engendré par la grammaire $L(G)$ est toujours un sous-ensemble de T^*. Les mots de ce langage sont toujours sur l'alphabet terminal.
2) La dérivation utilisera des lettres non-terminales.
3) Un mot du langage $L(G)$ engendré par G est toujours dérivé à partir de l'axiome ou signe source s.
4) Le langage $L(G)$ peut-être vide (par exemple si la grammaire G ne contient aucune règle dont le premier membre est s)
5) Le langage $L(G)$ peut contenir le mot vide que l'on notera ε.

Definition 7 (Ambiguïté d'une grammaire):
On dit d'une grammaire qu'elle est ambiguë, lorsqu'un mot du langage engendré par cette grammaire peut-être obtenu par au moins deux dérivations différentes.

Exemple 5:
Soit $G = (s, N, R, T)$ avec $N = \{s, \varepsilon\}, T = \{a, b\}$ et $R = \{s \to sasbs, s \to \varepsilon\}$

On obtient le mot $abab$ avec les deux dérivations suivantes :

1) $s \to sasbs \to asbs \to abs \to absasbs \xrightarrow{3} abab$

2) $s \to sasbs \to sasb \to sab \to sasbsab \xrightarrow{3} abab$

2.3 Hiérarchie ou Catégories de Chomsky

Nous allons maintenant considérer trois classes de langages engendrées par des grammaires formelles satisfaisant des conditions plus particulières. Ces classes seront notées L_0, L_1, L_2 et L_3 suivant qu'elles sont engendrées par des grammaires de type 0, 1, 2 et 3. Elles constituent la *hiérarchie de Chomsky.*

De manière symétrique aux grammaires qui génèrent des mots appartenant à un langage, il existe des machines qui permettent de dire si un mot appartient à un langage. Chaque langage peut-être reconnu par une machine propre à la catégorie à laquelle il appartient. Sans rentrer dans des détails qui ne font pas parties de notre sujet, nous mentionnerons ces machines à titre indicatif. Il est en effet intéressant de noter que plus un langage est de type élevé, plus il est aisé de le reconnaître.

2.3.1 Langage ou grammaire de type 0 : L_0

L_0 correspond aux langages engendrés par une grammaire quelconque, ce sont les *"langages récursivement énumérables"*. Aucune restriction n'est imposée à une grammaire de ce type.

Les machines de Turing permettent de reconnaître ce type de langage.

2.3.2 Langage ou grammaire de type 1 : L_1

Les grammaires de type 1 sont dites *contextuelles* ou *dépendantes du contexte* (en anglais *"context-sensitive"*).

Toute règle de la grammaire est de la forme :

$$u_1 P u_2 \to u_1 Q u_2 \text{ avec } u_1,\ u_2 \in (V \cup T)^*, P \in T \text{ et } Q \in (V \cup T)^* \backslash \{\varepsilon\}$$

Attention, le fait que Q ne puisse pas être le mot vide est une restriction essentielle ici.

Une grammaire dépendante du contexte est donc une une grammaire où l'on ne change qu'un seul signe non-terminal à la fois, en le remplaçant par un mot non vide de $(T \cup N)^*$. La règle dépendant du contexte ne peut se faire que si A est encadré par les suites u_1, u_2 données. Les suites u_1 et u_2 forment le contexte de P pour pouvoir appliquer la règle.

Les automates linéairement bornés permettent de reconnaître ce type de langage.

Exemple Classique :

Le langage $\{a^n b^n c^n, n \geq 1\}$ est engendré par une grammaire dépendante du contexte.

En effet une solution pour le générer est la grammaire suivante :

$T = \{a, b, c\}$,

$V = \{s, T, U, W, X, Y, Z, B\}$,

$R = \{s \rightarrow abc, s \rightarrow aTBc, TB \rightarrow UB, UB \rightarrow UW, UW \rightarrow BW, BW \rightarrow BT, Tc \rightarrow XBcc, BX \rightarrow BY, ZY \rightarrow ZB, ZB \rightarrow XB, aX \rightarrow aaT, aX \rightarrow aa, B \rightarrow b, BY \rightarrow ZY\}$

2.3.3 Langage ou grammaire de type 2 : L_2

Les grammaires de type 2 sont dites *algébriques* (en anglais : *"context-free"*).

Toute règle de la grammaire est de la forme :

$$A \rightarrow B \text{ avec } A \in N \text{ et } B \in (V \cup T)^*$$

Les automates à mémoire en pile permettent de reconnaître ce type de langage.

Exemple :

Le langage $\{a^n b^n, n \geq 1\}$ est engendré par une grammaire de type 2.

En effet :

$$T = \{a, b\},\ N = \{s, A, B\},\ \text{et } R = \{s \rightarrow asb, s \rightarrow ab\}$$

est une grammaire qui permet de générer $\{a^n b^n, n \geq 1\}$.

2.3.4 Langage ou grammaire de type 3 : L_3

Les grammaires de type 3 sont dites *régulières*.

Toute règle est de la forme :

$$A \rightarrow PB, A \rightarrow P \text{ avec } A, B \in NetP \in T^*$$

Les automates finis permettent de reconnaître de type de langage.

Exemple :

Le langage $\{\ a^n, n \geq 1\ \}$ est engendré par une grammaire de type 3.

En effet :

$$T = \{a\}, N = \{s\}, R = \{s \rightarrow a, s \rightarrow as\}$$

Cette grammaire est dite *régulière à gauche*.

2.3.5 Propriété

Clairement, plus les restrictions imposées aux grammaires sont importantes, moins il y a de langages qui peuvent être engendrés par de telles grammaires. Enonçons la propriété fondamentale suivante :

Propriete 1 (Propriété):

$$L_3 \subset L_2 \subset L_1 \subset L_0$$

et les inclusions sont strictes.

On peut démontrer cette propriété en montrant que : $\{a^n b^n c^n, n \geq 1\} \in L_1$, mais $\notin L_2$, puis que $\{a^n b^n, n \geq 1\} \in L_2$, mais $\notin L_3$.

On caractérise ainsi un langage par sa catégorie soit en montrant qu'il est généré par une grammaire de type donné, soit en construisant une machine qui le reconnaît.

Chapitre 3

La grammaire de Steedman

"Take care of the sevenths and the sound will take care of themselves" — Traditionnal

3.1 Notion de "grille" harmonique

Nous nous attachons ici à définir le plus clairement possible la notion de grille harmonique.

Jacques Siron dans *La Partition Intérieure* (cf. [21] p.407) définie la grille comme suit :

> "Dans le jargon du jazz, la grille d'accords (chord chart) est la représentation graphique habituellement utilisée pour symboliser la trame harmo-rythmique".

Nous dirons de plus qu'une grille harmonique est la répartition caractéristique d'une suite d'accords sur une nombre fini de mesure.

3.1.1 Notation traditionnelle des accords

Un accord est représenté par une fondamentale et un chiffrage.

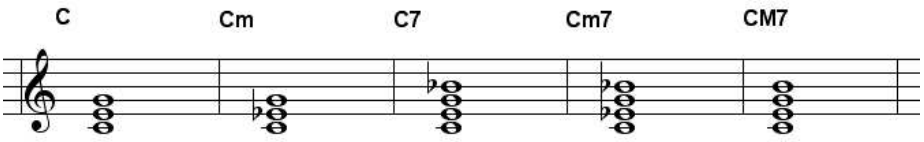
La fondamentale est notée en notation anglo-saxonne (A = La, B = Si), elle indique la note sur laquelle va être construit l'accord. La fondamentale peut prendre l'ensemble des valeurs de la gamme chromatique.

Le chiffrage indique la composition de l'accord, c'est à dire le choix et le nombre de notes constituant l'accord. Les chiffrages les plus simples sont :

- pas de chiffrage : l'accord est une triade majeur construite sur la fondamentale.
- m : l'accord est une triade mineure construite sur la fondamentale.
- 7 : l'accord est une triade majeure à laquelle on ajoute une septième mineure par rapport à la fondamentale.

- $m7$: l'accord est une triade majeure à laquelle on ajoute une septième mineure par rapport à la fondamentale.
- $M7$: l'accord est une triade majeure à laquelle on ajoute une septième majeure par rapport à la fondamentale.

Exemple : Accords sur Do



Il existe d'autres chiffrages d'accords passablement plus complexes et d'autres notations équivalentes aux précédentes. Pour une étude plus exhaustive, se reporter à l'excellent [21].

3.1.2 Notation traditionnelle des grilles

Une grille peut avoir une durée de deux mesures à trente deux (voire soixante-quatre pour certaines).

Exemple 6 (Blues en Do):

C	C	C	$C7$
F	$F7$	C	$C7$
G	$G7$	C	$C7$

La manière dont sont réalisés effectivement les accords est appelée *voicing* et est laissée au choix du musicien. Ce choix est une des caractéristique du style du musicien. La grille est, en général, accompagnée d'une ligne mélodique caractéristique du morceaux appelée thème. Le principe d'exécution d'une grille est celui d'un cycle harmonique répété ad libitum et sur lequel se succèdent une ou des improvisations mélodiques appelées chorus. La structure la plus commune est "exposition du thème, chorus, réexposition du thème". On trouve toutefois de nombreuses variantes à ce schéma un peu simpliste, surtout depuis l'explosion créative du Free Jazz et des musiques improvisées européenne.

Dans notre étude, nous considérons les grilles transposées comme équivalentes. Nous avons donc adopté à la manière de Steedman la notation fonctionnelle. Les accords sont symbolisés par un degré, et non une note, et un chiffrage.

Exemple 7:
Voici une grille de quatre mesures :

C	C	C	$C7$

Nous la représenterons comme suit :

I	I	I	$I7$

Comme le note Jacques Siron dans [21] p.408 :

> "Lorsqu'il y a plusieurs accords par mesure, il existe plusieurs système de symbolisations. La barre oblique est souvent utilisée comme signe de demi-mesure."

Exemple 8:

I	I	$IIm7$ / $V7$	$IM7$

3.2 Les règles de Steedman

La grammaire étudiée ici est due à Steedman, qui a proposé dans son article[19] en 1984 une liste de six règles engendrant des variantes harmoniques de la grille standard du blues de douze mesures :

I	I	I	$I7$
IV	IV	I	$I7$
V	$V7$	I	$I7$

Chaque règle est en fait la représentation symbolique d'une famille de règles. Les chiffres romains constituent l'alphabet non-terminal, tandis que la notation traditionnelles sont forme d'accord dont la fondamentale est noté en notation anglo-saxonne constitue l'alphabet terminal. L'alphabet terminal est très peu utilisé dans l'article, et les règles permettant de passer des symboles terminaux aux symboles non-terminaux ne sont pas explicitées. Chaque règle est transmutée sur chacun des degrés notés en chiffre romain.

Exemple 9:
La règle 1 par exemple correspond en fait à 36 règles, dont chacune porte sur un degré (soit 3 règles par degré).

Règle 1 : $x(m)(7) \rightarrow x(m)x(m)(7)$

s'écrit : I7 $\rightarrow$ I I7 pour l'accord de septième situé sur le premier degré.

ou IIm7 $\rightarrow$ IIm IIm7 pour l'accord de septième contenant une tierce mineure situé sur le second degré.

Ainsi les notations de Steedman permettent de réduire le nombre de règles à expliciter et lui permettent de gagner en clarté.

Nous présentons ici successivement les 6 règles accompagnées de commentaires concernant chacune d'elles.

Règle 0 : $S12 \rightarrow I(m)\ I7\ IV(m)\ I(m)\ V7\ I(m)$

Cette règle tient lieu d'axiome pour la génération d'un corpus de grilles de blues.

On peut voir le mot obtenu par application de cette règle comme une ossature caractéristique commune à toutes les grilles de blues exposées dans le texte de Coker [**?**] auquel Steedman fait référence. Cette règle est essentielle en ce qu' elle fixe le cadre dans lequel vont être appliquées les autres règles afin de générer le langage souhaité. Elle ne concerne cependant pas notre utilisation future car elle réduit l'application au cas du blues et nous l'avons étendu à toutes les grilles possibles.

Règle 1 : $x(m)(7) \rightarrow x(m)\ x(m)(7)$

Nous supposons ici que les parenthèses doivent être présentes (ou absentes) des deux cotés de la règle simultanément. La règle 1 s'applique à tous les types d'accords qu'ils soient de septième ou non. C'est elle qui permet d'augmenter le nombre d'accord dans la grille en "repoussant" les accords de septième ou en doublant les triades. Nous appellerons cette règle la règle de *dédoublement.*

Si l'on réécrit une séquence à l'aide de cette règle le nombre d'accords de la séquence augmente, mais pas sa durée. Steedman introduit une convention importante pour toutes les règles de sa grammaire, elle consiste en ce que la durée du membre de gauche de la règle soit la même que celle du membre de droite. Ainsi, les deux accords du membre de droite ont chacun une durée égale à la moitié de la durée de l'accord de gauche. Cette division du temps ne peut cependant pas être appliquée indéfiniment et Steedman la limite volontairement au quart de mesure(p.59 in [19]). Cette limitation est discutable sur un plan formel, mais justifiée sur un plan musical par le fait qu'on ne trouve qu'exceptionnellement plus d'un accord par temps dans une grille de jazz.

Remarque (Sur l'utilisation de la règle 1 par Steedman) :

Il existe une ambiguïté au niveau de cette notation pour savoir si les () doivent être présentes simultanément dans les deux membres de la règle ou non, et pour savoir si un accord sans terminaison est nécessairement une triade.

Si les parenthèses ne sont pas obligatoirement présentes simultanément, la famille de règles est en fait de 18 règles pour chaque degrés. En partant de l'idée suggérée par Steedman dans son article, les () doivent l'être.

Cependant si un accord sans terminaison est nécessairement une triade, ce que Steedman semble affirmer et qui correspond à la tradition la plus courante,

alors il ne peut obtenir certaines grilles du corpus de Coker qu'en se contredisant :

Exemple 1 page 60 in [19] : $I/I/I/I7||IV/IV/I/I||V7/V7/I/I/$

Exemple 2 page 60 in [19] : $I/IV/I/I7||IV/IV/I/I||V7/V7/I/I/$

Ces deux grilles sont impossibles à obtenir à partir de l'axiome et en appliquant la règle 1, car les accords de septièmes ne peuvent se dédoubler.

Règle 2 : $x(m)(7) \rightarrow x(m)(7)\ Sd_x$

La notation Sdx renvoie au degré de sous-dominante du degré considéré. Cette règle permet ainsi d'obtenir une séquence de type I IV. Nous n' utiliserons cette règle qu'une fois dans la suite car elle est spécifique au corpus que Steedman a voulu générer et à certaines caractéristiques harmoniques locales.

Règle 3a : $w\ x7 \rightarrow D_x(m)7\ x7$

Règle 3b : $w\ xm7 \rightarrow D_x7\ xm7$

La notation D_x renvoie le degré de la dominante du degré considéré. Cette règle est primordiale, elle permet de générer des cadences de type V I et elle est commune à toutes les grilles de jazz. Le w peut être remplacé par n'importe quel accord qui ne soit pas de septième. Remarquons que seul un accord de septième permet de générer une cadence. L'accord de septième joue ainsi un rôle de "générateur cadentiel" primordial et permet les substitutions tritoniques dues à la règle 4.

Règle 4 : $D_x7\ x(m)(7) \rightarrow Stb_x(m)(7)\ x(m)(7)$

La notation Stb_x renvoie le degré de la sus-tonique "bémolisée" du degré de l'accord considéré. Découlant directement de la précédente, la règle 4 permet la fameuse substitution tritonique. C'est aussi une règle centrale qui comme la règle 3 est applicable à de nombreuses grilles.

Règle 5 : $x\ x\ x\ \rightarrow x\ St_x\ xm\ Mxm$

Règle 6 : x(m) x(m) $\left\{ \begin{array}{c} D_x \\ St_x\ (m)(7) \\ L_x(m)(7) \end{array} \right\} \rightarrow$ x(m) #x°7 $\left\{ \begin{array}{c} D_x \\ St_x(m) \\ M_x(m) \end{array} \right\}$

Les règles 5 et 6 ne seront pas utilisées dans la suite. Elles ne sont utilisées par steedman que pour générer des cas particuliers de grilles de blues et ne sont pas encore prises en compte dans nos applications.

La règle 5 renvoie une cadence qui n'est pas du type dominante/tonique et que l'on ne trouve qu'une fois dans l'ensemble des grilles de Coker (Exemple (d) in Stedman's table page 54). Cette règle nous apparaît comme *ad hoc*.

La règle 6 est dédiée à l'introduction d'accords de passages particuliers : les accords de "septième diminuée"(*diminished seventh chord*).

3.3 Exploration du langage généré par la grammaire de Steedmann

Dans cette section nous décrivons un programme écrit en Common Lisp qui permet de générer à partir d'une grammaire, toutes les grilles harmoniques appartenant au langage généré par cette grammaire.

Ce programme vient compléter celui écrit par Marc Chemillier et décrit dans son article ([5] page 20). Le principe est de reprendre la grammaire de Steedman en modifiant l'axiome pour qu'il donne la grille de base sur laquelle on applique des règles choisies dans le corpus. Nous ne nous intéressons ici qu'aux règles de dédoublement, cadentielles et tritoniques (règle 1, 3, 4), pour les raisons que nous avons déjà exposées ci-dessus. Nous évoquerons plus tard une utilisation possible de la règle 2.

Pour implémenter les règles de la grammaire de Steedman, nous commençons par expliciter toutes les règles "contenues" dans une seule.

La règle 1 :

$$x(m)(7) \rightarrow x(m)\ x(m)(7)$$

donne ainsi "naissance" à

$$x7 \rightarrow x\ x7$$

$$x \rightarrow x\ x$$

et à

$$xm7 \rightarrow xm\ xm7$$

La grammaire a ensuite été modifiée pour contrôler le découpage de la grille en mesures, en introduisant des / dans l'énoncé des règles.

La règle 1 est modifiée de la manière suivante :

$$x \rightarrow x\ x$$

et remplacée par :

$$/x/ \rightarrow /x\ x/$$

La division minimale de la grille est ainsi fixée à la demi-mesure.

Les règles 3 et 4 sont dédoublées. La forme originelle :

$$w\ x7 \rightarrow D_x7\ x7$$

est complétée par :

$$w\ /\ x7 \rightarrow D_x7\ /\ x7$$

La première s'applique à l'intérieur d'une même mesure, alors que la seconde s'applique de part et d'autre d'une barre de mesure.

Les règles de la grammaire de Steedman sont donc les suivantes :

R_{1-1} $/x/ \rightarrow /x\ x/$
R_{1-2} $/x7/ \rightarrow /x\ x7/$
R_{1-3} $/xm7/ \rightarrow /x\ xm7/$
R_{3a-1} $w\ x7 \rightarrow D_x7\ x7$

R_{3a-2} $w\ x7 \rightarrow D_xm7\ x7$
R_{3a-11} $w\ /\ x7 \rightarrow D_x7/x7$
R_{3a-12} $w/x7 \rightarrow D_xm7/x7$
R_{3b} $w\ xm7 \rightarrow D_x7\ xm7$

R_{4-1} $D_x7\ x7 \rightarrow Stb_x7\ x7$

R_{4-2} $D_x7\ x \rightarrow Stb_x\ x$
R_{4-3} $D_x7\ xm7 \rightarrow Stb_xm7\ xm7$
R_{4-11} $D_x7\ /\ x7 \rightarrow Stb_x7\ /\ x7$

R_{4-12} $D_x7\ /\ x \rightarrow Stb_x\ /\ x$
R_{4-13} $D_x7\ /\ xm7 \rightarrow Stb_xm7\ /\ xm7$

Si le principe du programme de Marc Chemillier est de :

> "[...]tirer au hasard une règle, de chercher une position dans la grille pour l'appliquer, puis d'effectuer la substitution. Si la règle tirée n'est pas applicable, on en tire une autre. On recommence ainsi l'opération un nombre de fois fixé à l'avance, ou bien on s'arrête si aucune règle n'est applicable."([5] page 21).

Nous nous sommes attachés à générer toutes les grilles possibles afin de comprendre les qualités communes à toutes les grilles générées.

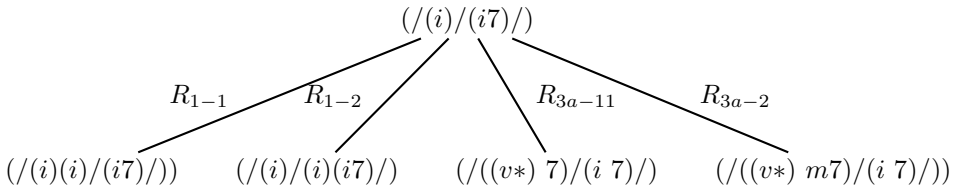
Nous avons choisi de représenter le résultat sous forme d'un arbre. Dans cet arbre, le père est constituée par la grille sur laquelle le programme va substituer les accords selon les règles de Steedman.

L'ensemble des fils est obtenu à partir du père grâce à la fonction "génération-suivante". Elle prend en argument une grille et une liste de numéros de règle et renvoie les grilles résultant de l'application d'une règle à un endroit dans la grille. Chaque grille obtenue est donc une dérivation directe de la grille entrée en argument.

Exemple 10:

```
? (generation-suivante-can '(/ (i) / (i 7) /) '(1-1 1-2 1-3 3a-1
 3a-2 3a-11 3a-12 3b 4-1 4-2 4-3 4-11 4-12 4-13))
((/ (i) (i) / (i 7) /) (/ (i) / (i) (i 7) /)
  (/ ((v *) 7) / (i 7) /) (/ ((v *) m7) / (i 7) /))
```

Soit en fait sous forme d'arbre :



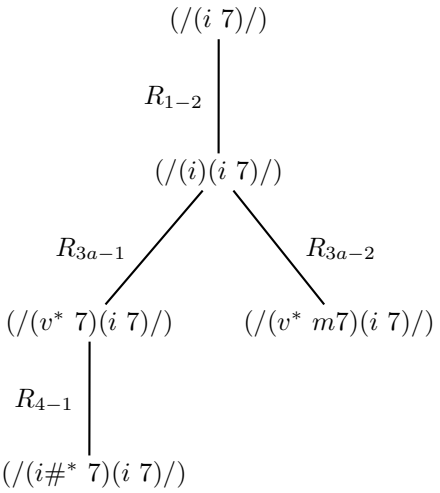
Le marquage par une étoile est expliqué comme suit :

> "Quand le degré d'un accord est modifié, on le marque par une * pour indiquer que l'accord ne doit plus être affecté à la variable w".([5] page 22)

Le marquage permet d'éviter à la fonction de boucler en renvoyant toujours les mêmes positions pour les substitutions.

Nous pouvons ensuite générer l'ensemble de l'arbre par récursion sur chaque élément obtenu. La récursion s'arrête quand aucune substitution n'est plus possible, c'est à dire quand la fonction qui renvoie la liste d'emplacements où les substitutions sont possibles, renvoie nil.

La figure suivante présente l'arbre obtenu pour la grille : $(/(i7)/)$. Chaque branche est commentée par la règle qui a été appliquée pour obtenir le noeud suivant.



Le même arbre obtenu en Common Lisp :

```
? (liste-num-steedman '(/ (i 7) /)
  '(1-1 1-2 1-3 3a-1 3a-2 3a-11 3a-12 3b 4-1 4-2 4-3 4-11 4-12 4-13))
((0 (/ (i 7) /)) (1 (/ (i) (i 7) /)) (2 (/ ((v *) 7) (i 7) /))
(3 (/ ((i# *) 7) (i 7) /)) (2 (/ ((v *) m7) (i 7) /)))
```

Chaque élément de la liste résultante est en fait une liste dont le Car est un chiffre qui correspond au nombre de règles qui ont été appliquées sur la grille de départ pour obtenir la grille qui est contenue dans le Cdr.

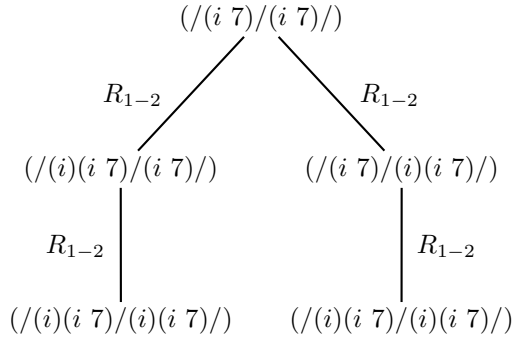
La seconde tâche a été d'imaginer une procédure d'affichage permettant une plus grande lisibilité du résultat. C'est la fonction affiche-liste-num qui remplit cette tâche.

Exemple 11:

```
? (affiche-liste-num (liste-num-steedman '(/ (i 7) /)
  '(1-1 1-2 1-3 3a-1 3a-2 3a-11 3a-12 3b 4-1 4-2 4-3 4-11 4-12 4-13)))
("(0 (/ (i 7) /))
" " (1 (/ (i) (i 7) /))
" "  (2 (/ ((v *) 7) (i 7) /))
" "   (3 (/ ((i# *) 7) (i 7) /))
" "  (2 (/ ((v *) m7) (i 7) /))
")
```

Nous avons dû ensuite faire face à des problèmes de redondance dans la génération de l'arbre. La grammaire étant ambiguë, il est possible de générer deux grilles identiques par deux "chemins" différents (cf. Définition 7).

Exemple 12 (Arbre de dérivation de la séquence $(/(i\ 7)/(i\ 7)/)$):



Pour éviter ces redondances nous avons utilisé une table de hachage. Les grilles générées sont donc stockées dans la table de hachage au fur et à mesure de leur production. Puis à chaque grille générée on teste si elle existe dans la table, si elle existe on arrête la récursion sinon on continue. Dans l'exemple précédent le dernier élément de la deuxième branche ne sera pas généré car déjà généré dans la première branche.

En ajoutant un compteur pour le calcul du nombre de grilles, nous avions à notre disposition les outils suffisants pour étudier le langage généré par la grammaire de Steedman.

Nous nous sommes très vite rendu compte que la taille du langage était directement liée à l'axiome ou grille de départ et à sa composition. Une grille ne contenant aucun accord de septième n'engendre que très peu de grilles car très peu de règles lui sont applicables.

Exemple 13 (Séquence ne contenant aucun accord de septième):

```
? (affiche-liste-num (liste-num-steedman '(/ (i) / (i) / (i) / (i) /)
                     '(1-1 1-2 1-3 3a-1 3a-2 3a-11 3a-12 3b
                       4-1 4-2 4-3 4-11 4-12 4-13)))
("(0 (/ (i) / (i) / (i) / (i) /))
" " (1 (/ (i) (i) / (i) / (i) / (i) /))
" "  (2 (/ (i) (i) / (i) (i) / (i) / (i) /))
" "   (3 (/ (i) (i) / (i) (i) / (i) (i) / (i) /))
" "    (4 (/ (i) (i) / (i) (i) / (i) (i) / (i) (i) /))
" "   (3 (/ (i) (i) / (i) (i) / (i) / (i) (i) /))
" "  (2 (/ (i) (i) / (i) / (i) (i) / (i) /))
" "   (3 (/ (i) (i) / (i) / (i) (i) / (i) (i) /))
" "  (2 (/ (i) (i) / (i) / (i) / (i) (i) /))
" " (1 (/ (i) / (i) (i) / (i) / (i) /))
" "  (2 (/ (i) / (i) (i) / (i) (i) / (i) /))
" "   (3 (/ (i) / (i) (i) / (i) (i) / (i) (i) /))
" "  (2 (/ (i) / (i) (i) / (i) / (i) (i) /))
" " (1 (/ (i) / (i) / (i) (i) / (i) /))
" "  (2 (/ (i) / (i) / (i) (i) / (i) (i) /))
" " (1 (/ (i) / (i) / (i) / (i) (i) /))
")
```

Inversement plus les accords de septième sont nombreux plus la taille du langage associé est importante.

Exemple 14 (Avec un accord de septième situé au milieu de la séquence):

```
? (affiche-liste-num (liste-num-steedman '(/ (i) (i 7) / (i) (i) /)
                     '(1-1 1-2 1-3 3a-1 3a-2 3a-11 3a-12 3b
                       4-1 4-2 4-3 4-11 4-12 4-13)))
("(0 (/ (i) (i 7) / (i) (i) /))
" " (1 (/ ((v *) 7) (i 7) / (i) (i) /))
" "  (2 (/ ((i# *) 7) (i 7) / (i) (i) /))
" " (1 (/ ((v *) m7) (i 7) / (i) (i) /))
```

Exemple 15 (Avec un accord de septième situé à la fin de la séquence):

```
? (affiche-liste-num (liste-num-steedman '(/ (i) (i) / (i) (i 7) /)
                     '(1-1 1-2 1-3 3a-1 3a-2 3a-11 3a-12 3b
                       4-1 4-2 4-3 4-11 4-12 4-13)))
("(0 (/ (i) (i) / (i) (i 7) /))
" " (1 (/ (i) (i) / ((v *) 7) (i 7) /))
" "  (2 (/ (i) ((ii *) 7) / ((v *) 7) (i 7) /))
" "   (3 (/ ((vi *) 7) ((ii *) 7) / ((v *) 7) (i 7) /))
" "    (4 (/ ((ii# *) 7) ((ii *) 7) / ((v *) 7) (i 7) /))
" "     (5 (/ ((ii# *) 7) ((ii *) 7) / ((i# *) 7) (i 7) /))
" "     (5 (/ ((ii# *) 7) ((v# *) 7) / ((v *) 7) (i 7) /))
" "      (6 (/ ((vi *) 7) ((v# *) 7) / ((v *) 7) (i 7) /))
```

```
" "       (7 (/ ((vi *) 7) ((v# *) 7) / ((i# *) 7) (i 7) /))
" "        (8 (/ ((vi *) 7) ((ii *) 7) / ((i# *) 7) (i 7) /))
" "      (6 (/ ((ii# *) 7) ((v# *) 7) / ((i# *) 7) (i 7) /))
" "   (3 (/ ((vi *) m7) ((ii *) 7) / ((v *) 7) (i 7) /))
" "    (4 (/ ((vi *) m7) ((ii *) 7) / ((i# *) 7) (i 7) /))
" "    (4 (/ ((vi *) m7) ((v# *) 7) / ((v *) 7) (i 7) /))
" "     (5 (/ ((vi *) m7) ((v# *) 7) / ((i# *) 7) (i 7) /))
" "   (3 (/ (i) ((ii *) 7) / ((i# *) 7) (i 7) /))
" "   (3 (/ (i) ((v# *) 7) / ((v *) 7) (i 7) /))
" "    (4 (/ ((ii# *) m7) ((v# *) 7) / ((v *) 7) (i 7) /))
" "     (5 (/ ((ii# *) m7) ((v# *) 7) / ((i# *) 7) (i 7) /))
" "      (6 (/ ((ii# *) m7) ((ii *) 7) / ((i# *) 7) (i 7) /))
" "    (4 (/ (i) ((v# *) 7) / ((i# *) 7) (i 7) /))
" "  (2 (/ (i) ((ii *) m7) / ((v *) 7) (i 7) /))
" "   (3 (/ ((vi *) 7) ((ii *) m7) / ((v *) 7) (i 7) /))
" "    (4 (/ ((vi *) 7) ((ii *) m7) / ((i# *) 7) (i 7) /))
" "     (5 (/ ((ii# *) m7) ((ii *) m7) / ((i# *) 7) (i 7) /))
" "    (4 (/ ((ii# *) m7) ((ii *) m7) / ((v *) 7) (i 7) /))
" "   (3 (/ (i) ((ii *) m7) / ((i# *) 7) (i 7) /))
" "  (2 (/ (i) (i) / ((i# *) 7) (i 7) /))
" "   (3 (/ (i) ((v# *) m7) / ((i# *) 7) (i 7) /))
" "    (4 (/ ((ii# *) 7) ((v# *) m7) / ((i# *) 7) (i 7) /))
" "     (5 (/ ((vi *) m7) ((v# *) m7) / ((i# *) 7) (i 7) /))
" " (1 (/ (i) (i) / ((v *) m7) (i 7) /))
")
```

Exemple 16 (Avec deux accords de septième au milieu et à la fin de la séquence):

```
? (affiche-liste-num (liste-num-steedman '(/ (i) (i 7) / (i) (i 7) /)
                   '(1-1 1-2 1-3 3a-1 3a-2 3a-11 3a-12 3b
                     4-1 4-2 4-3 4-11 4-12 4-13)))
("(0 (/ (i) (i 7) / (i) (i 7) /))
" " (1 (/ ((v *) 7) (i 7) / (i) (i 7) /))
" "  (2 (/ ((v *) 7) (i 7) / ((v *) 7) (i 7) /))
" "   (3 (/ ((v *) 7) ((ii *) 7) / ((v *) 7) (i 7) /))
" "    (4 (/ ((v *) 7) ((ii *) 7) / ((i# *) 7) (i 7) /))
" "    (4 (/ ((v *) 7) ((v# *) 7) / ((v *) 7) (i 7) /))
" "     (5 (/ ((v *) 7) ((v# *) 7) / ((i# *) 7) (i 7) /))
" "   (3 (/ ((v *) 7) ((ii *) m7) / ((v *) 7) (i 7) /))
" "    (4 (/ ((v *) 7) ((ii *) m7) / ((i# *) 7) (i 7) /))
" "   (3 (/ ((i# *) 7) (i 7) / ((v *) 7) (i 7) /))
" "    (4 (/ ((i# *) 7) ((ii *) 7) / ((v *) 7) (i 7) /))
" "     (5 (/ ((i# *) 7) ((ii *) 7) / ((i# *) 7) (i 7) /))
" "     (5 (/ ((i# *) 7) ((v# *) 7) / ((v *) 7) (i 7) /))
" "      (6 (/ ((i# *) 7) ((v# *) 7) / ((i# *) 7) (i 7) /))
" "    (4 (/ ((i# *) 7) ((ii *) m7) / ((v *) 7) (i 7) /))
" "     (5 (/ ((i# *) 7) ((ii *) m7) / ((i# *) 7) (i 7) /))
```

```
" "    (4 (/ ((i# *) 7) (i 7) / ((i# *) 7) (i 7) /))
" "     (5 (/ ((i# *) 7) ((v# *) m7) / ((i# *) 7) (i 7) /))
" "   (3 (/ ((v *) 7) (i 7) / ((i# *) 7) (i 7) /))
" "    (4 (/ ((v *) 7) ((v# *) m7) / ((i# *) 7) (i 7) /))
" "  (2 (/ ((v *) 7) (i 7) / ((v *) m7) (i 7) /))
" "   (3 (/ ((i# *) 7) (i 7) / ((v *) m7) (i 7) /))
" "  (2 (/ ((i# *) 7) (i 7) / (i) (i 7) /))
" " (1 (/ (i) (i 7) / ((v *) 7) (i 7) /))
" "  (2 (/ ((v *) m7) (i 7) / ((v *) 7) (i 7) /))
" "   (3 (/ ((v *) m7) ((ii *) 7) / ((v *) 7) (i 7) /))
" "    (4 (/ ((v *) m7) ((ii *) 7) / ((i# *) 7) (i 7) /))
" "    (4 (/ ((v *) m7) ((v# *) 7) / ((v *) 7) (i 7) /))
" "     (5 (/ ((v *) m7) ((v# *) 7) / ((i# *) 7) (i 7) /))
" "   (3 (/ ((v *) m7) ((ii *) m7) / ((v *) 7) (i 7) /))
" "    (4 (/ ((v *) m7) ((ii *) m7) / ((i# *) 7) (i 7) /))
" "   (3 (/ ((v *) m7) (i 7) / ((i# *) 7) (i 7) /))
" "    (4 (/ ((v *) m7) ((v# *) m7) / ((i# *) 7) (i 7) /))
" "  (2 (/ (i) ((ii *) 7) / ((v *) 7) (i 7) /))
" "   (3 (/ ((vi *) 7) ((ii *) 7) / ((v *) 7) (i 7) /))
" "    (4 (/ ((ii# *) 7) ((ii *) 7) / ((v *) 7) (i 7) /))
" "     (5 (/ ((ii# *) 7) ((ii *) 7) / ((i# *) 7) (i 7) /))
" "     (5 (/ ((ii# *) 7) ((v# *) 7) / ((v *) 7) (i 7) /))
" "      (6 (/ ((vi *) 7) ((v# *) 7) / ((v *) 7) (i 7) /))
" "       (7 (/ ((vi *) 7) ((v# *) 7) / ((i# *) 7) (i 7) /))
" "        (8 (/ ((vi *) 7) ((ii *) 7) / ((i# *) 7) (i 7) /))
" "      (6 (/ ((ii# *) 7) ((v# *) 7) / ((i# *) 7) (i 7) /))
" "   (3 (/ ((vi *) m7) ((ii *) 7) / ((v *) 7) (i 7) /))
" "    (4 (/ ((vi *) m7) ((ii *) 7) / ((i# *) 7) (i 7) /))
" "    (4 (/ ((vi *) m7) ((v# *) 7) / ((v *) 7) (i 7) /))
" "     (5 (/ ((vi *) m7) ((v# *) 7) / ((i# *) 7) (i 7) /))
" "   (3 (/ (i) ((ii *) 7) / ((i# *) 7) (i 7) /))
" "   (3 (/ (i) ((v# *) 7) / ((v *) 7) (i 7) /))
" "    (4 (/ ((ii# *) m7) ((v# *) 7) / ((v *) 7) (i 7) /))
" "     (5 (/ ((ii# *) m7) ((v# *) 7) / ((i# *) 7) (i 7) /))
" "      (6 (/ ((ii# *) m7) ((ii *) 7) / ((i# *) 7) (i 7) /))
" "    (4 (/ (i) ((v# *) 7) / ((i# *) 7) (i 7) /))
" "  (2 (/ (i) ((ii *) m7) / ((v *) 7) (i 7) /))
" "   (3 (/ ((vi *) 7) ((ii *) m7) / ((v *) 7) (i 7) /))
" "    (4 (/ ((vi *) 7) ((ii *) m7) / ((i# *) 7) (i 7) /))
" "     (5 (/ ((ii# *) m7) ((ii *) m7) / ((i# *) 7) (i 7) /))
" "    (4 (/ ((ii# *) m7) ((ii *) m7) / ((v *) 7) (i 7) /))
" "   (3 (/ (i) ((ii *) m7) / ((i# *) 7) (i 7) /))
" "  (2 (/ (i) (i 7) / ((i# *) 7) (i 7) /))
" "   (3 (/ (i) ((v# *) m7) / ((i# *) 7) (i 7) /))
" "    (4 (/ ((ii# *) 7) ((v# *) m7) / ((i# *) 7) (i 7) /))
" "     (5 (/ ((vi *) m7) ((v# *) m7) / ((i# *) 7) (i 7) /))
```

```
" " (1 (/ ((v *) m7) (i 7) / (i) (i 7) /))
" "  (2 (/ ((v *) m7) (i 7) / ((v *) m7) (i 7) /))
" " (1 (/ (i) (i 7) / ((v *) m7) (i 7) /))
")
```

Evidemment plus l'axiome est long plus le nombre de grilles générées est important. D'autre part, si la séquence ne contient qu'un accord de septième plus l'accord est situé vers la fin de la grille, plus le nombre de grilles générées est important comme on le voit sur les exemples ci-dessus.

Chapitre 4

Pour un autre cadre formel *ou* "décontextualisation" de la grammaire de Steedman

Lorsque Steedman écrit sa grammaire, il cherche avant tout à montrer qu'il peut produire un corpus de grilles d'un même style à partir d'un unique axiome et d'un ensemble de règles suffisamment explicites. Pour ce faire il développe un formalisme assez intuitif identifiant les règles de substitution couramment admises dans le Jazz, à des règles de grammaire formelles. La grammaire qu'il obtient est de ce fait une grammaire contextuelle relativement difficile à implémenter dans le sens de l'analyse et dont les propriétés sont assez faibles.

Il est cependant remarquable que le langage engendré soit fini quel que soit l'axiome utilisé. Il semblait donc raisonnable de penser qu'il existait une grammaire sinon régulière au moins algébrique qui engendrait un langage sinon équivalent au moins similaire. Sous l'impulsion d'Emmanuel Saint-James, nous nous sommes donc atteler à une tache que je ne croyais pas réalisable : décontextualiser la grammaire de Steedman.

L'intérêt de cette "décontextualisation" est double :

Catégoriser le langage engendré : Si on peut trouver une grammaire de type 2 (ou même de type 3) qui engendre le langage engendré par la grammaire de Steedman, on pourra alors construire un automate à pile (ou même fini) qui reconnaîtra ce langage. Ce problème est un des problèmes ouverts donnés par F. Pachet dans son article [17].

Implémenter O'Max autrement : En écrivant une grammaire de type 2 ou 3, nous pensons pouvoir exhiber des propriétés qui nous permettraient de réduire la boucle d'interaction dans O'Max en générant les séquences de façon itérative.

Le problème n'est pas simple. Il ne suffit pas de remplacer tel accord par tel autre, sinon on aboutit à des combinaisons harmoniques incohérentes. L'information portant sur le contexte doit être conservée dans le nouveau formalisme

mais doit être contenue autre part. Il s'agit aussi de repenser la manière dont la séquence harmonique va être construite car les règles contextuelles de Steedman ne modifient pas la taille de la séquence sur laquelle on applique les règles.

4.1 Définition du Cadre Formel

Pour des questions de rigueur et de compréhension, nous avons du repenser la grammaire de Steedman dans un cadre plus algébrique et modifier quelque peu le formalisme original. Nous définissons ainsi les accords comme des couples (au sens du produit cartésien) et les fonctions D,S et T (équivalentes aux notations D_x, Stb_x et Sd_x) comme des opérateurs sur ces couples.

Definition 8 (Notion d'accord):
Un accord u est un couple (a, b) tel que $u \in A \times B$ où :
$A = \{i, i\#, ii, ii\#, iii, iv, iv\#, v, v\#, vi, vi\#, vii\}$ *et*
$B = \{\varepsilon, 7\}$.

Exemple 17:
L'accord de $7^{\text{ème}}$ situé sur le premier degré est noté $(i, 7)$.

Definition 9 (Les fonctions D, S et T):
Soit $A = \{i, i\#, ii, ii\#, iii, iv, iv\#, v, v\#, vi, vi\#, vii\}$, on identifie A à $\mathbb{Z}/12\mathbb{Z}$, l'opération d'addition est donc naturellement celle induite par $(\mathbb{Z}/12\mathbb{Z}, +)$. Nous définissons successivement les fonctions S, D et S_d comme des fonctions de A dans lui-même qui à tout élément a de A associent respectivement son successeur $(S(a) = a + 1)$, son 7^{me} successeur $(D(a) = a + 7)$ et son 5^{me} successeur $T(a) = a + 5$.

Exemple 18:

$$S(i) = i\#,\ S(vii) = i$$

$$D(i) = v,\ D(v) = ii$$

$$T(i) = iv$$

Propriete 2 (Sur les fonctions D et S):
Quelques propriétés sur les fonctions D et S, que nous ne démontrerons pas :
1)D et S sont commutatives. $D \circ S = S \circ D$
2) D et S sont idempotentes d'ordre 12 : $D^{12} = Id$ et $S^{12} = Id$.
3) Il existe un nombre fini de couple (p, q) pour lesquels $D^p S^q = Id$:

$$(5,1)(10,2)(3,3)(8,4)(1,5)(6,6)(11,7)(4,8)(9,9)(2,10)(7,11)$$

Remarque sur les notations :
Nous conviendrons que pour une séquence d'accord, lorsque Steedman note ses règles 3 et 4 :

$$w\ x7 \rightarrow D_x7\ x7$$

$$D_x7\ x(7) \rightarrow \mathrm{Stb_x}(7)\ x(7)$$

l'écriture sous forme de couple équivalente est :

$$(a, \varepsilon)(x, 7) \rightarrow (D(x), 7)(x, 7)$$

$$(D(x), 7)(x, 7) \rightarrow (S(x), 7)(x, 7)$$

4.2 La grammaire régulière *ou* générations de séquences d'accords

Nous définissons la grammaire algébrique comme suit :

4.2.1 Les Alphabets

Definition 10 (L'alphabet Terminal):

$$T = A \times B$$

$$où\ A = \{I, I\#, II, II\#, III, IV, IV\#, V, V\#, VI, VI\#, VII\}$$

$$et\ B = \{\varepsilon, m, 7, m7\}$$

Un symbole de l'alphabet terminal est donc un accord représenté sous la forme d'un couple et dont le degré est en majuscule.

Definition 11 (L'alphabet Non-Terminal):

$$N \subset (d, s)^* x\ où\ x \in \{i, i\#, ii, ii\#, ...vii, i7, i\#7, ii7, ii\#7, ...vii7\}$$

N est un sous-ensemble fini de $(d, s)^* x$.

De la finitude de N : Pour définir une grammaire formelle l'alphabet non-terminal doit être fini. En fait la longueur des mots de N n'excédera jamais 18 caractères (et $card(N) < 2^{18}$). Nous expliquerons pourquoi en définissant une famille de règles "spéciales" qui pour des raisons de compréhension générale sont exposées après les règles usuelles.

4.2.2 Les règles de la Grammaire

Dans cette section, nous nous restreignons aux accords majeurs et aux accords de septième de dominante et aux règles qui leur sont associées.

$$\begin{array}{ll} R_{1-1} & x \rightarrow x\ x \\ R_{1-2} & x7 \rightarrow x\ x7 \\ R_{3a-1} & w\ x7 \rightarrow D_x7\ x7 \\ R_{4-1} & D_x7\ x7 \rightarrow Stb_x7\ x7 \\ R_{4-2} & D_x7\ x \rightarrow Stb_x\ x \end{array}$$

Nous utiliserons de plus la règle 2. Nous la traiterons cependant à part car nous ne l'utiliserons qu'une fois et uniquement pour montrer un cas particulier.

Les règles 3 et 4

La première règle source ou axiome permet d'engendrer une séquence se terminant par un accord quelconque et dont le contenu dépend naturellement de cet accord. L'axiome est indicé par x qui prend sa valeur dans $A = \{i, i\#, ii, ii\#, iii, iv, iv\#, v, v\#, vi, vi\#, vii\}$.

S'il est indicé par x la séquence harmonique se termine par un accord majeur et on a l'ensemble de règles suivant :

$$s_x \rightarrow x(X, \varepsilon)$$

Nous avons de plus besoin de :

$$s_x \rightarrow \varepsilon$$

La règle de dédoublement s'écrit alors :

$$x \rightarrow x(X, \varepsilon)$$

S'il est indicé par $x7$ alors la chaîne de production se termine par un accords de septième et on a :

$$s_{x7} \rightarrow x7(X, 7)$$

Comme précédemment, il nous faut :

$$s_{x7} \rightarrow \varepsilon$$

Les 5 règles suivantes permettent d'engendrer des séquences de 2 accords.

$$x7 \rightarrow \varepsilon$$

$$x7 \rightarrow dx7(D(X), 7)$$

$$x7 \rightarrow sx7(S(X), 7)$$

$$dx7 \rightarrow \varepsilon$$

$$sx7 \rightarrow \varepsilon$$

Les 7 règles suivantes ajoutées aux 6 premières permettent de générer des séquences d'accords de longueur 3.

$$dx7 \rightarrow ddx7(D^2(X), 7)$$

$$dx7 \rightarrow sdx7(S(D(X)), 7)$$

$$sx7 \rightarrow sdx7(D(S(X)), 7)$$

$$sx7 \rightarrow ssx7(S^2(X)), 7)$$

$$ddx7 \rightarrow \varepsilon$$

$$sdx7 \rightarrow \varepsilon$$

$$ssx7 \rightarrow \varepsilon$$

Toutes ces règles sont des règles correspondant à une grammaire régulière à gauche. Nous ne donnons pas ici l'ensemble des règles de la grammaire, mais à l'instar de Steedman une forme contractée des règles.

Definition 12 (Les Règles D et S):
Soit u un élément de N qui contienne p fois l'élément d et q fois l'élément s, soit u_1 un élément de N qui contienne $p+1$ fois l'élément d et q fois l'élément s, soit u_2 un élément de N qui contienne p fois l'élément d et $q+1$ fois l'élément s, alors on a :

$$u \rightarrow \varepsilon$$

Règles D :

$$u \rightarrow u_1(D^{p+1}p \circ S^q(X), 7)$$

Règles S :

$$u \rightarrow u_2(D^p \circ S^{q+1}q(X), 7)$$

Le principe général de cette grammaire est de "mémoriser" les transformations que l'on a effectuées sur l'accord initial à l'aide du mot de l'alphabet non terminal qui se trouve à droite de la règle.

Exemple 19:
Nous montrons dans cet exemple comment obtenir la séquence :

$$(VI,7)(II,7)(V,7)(I,7)$$

On applique tout d'abord l'axiome :

$$s_{i7} \rightarrow i(I,7)$$

Puis la règle D :

$$i \rightarrow di(D(I),7)$$

Qui permet d'écrire par dérivation directe :

$$i(I,7) \rightarrow di(V,7)(I,7)$$

Ensuite la règle D :

$$di \rightarrow ddi(D^2(I),7)$$

Qui permet d'écrire par dérivation directe :

$$di(V,7)(I,7) \rightarrow ddi(II,7)(V,7)(I,7)$$

Puis la règle D :

$$ddi \rightarrow dddi(D^3(I),7)$$

Qui permet d'écrire par dérivation directe :

$$ddi(II,7)(V,7)(I,7) \rightarrow dddi(VI,7)(II,7)(V,7)(I,7)$$

Enfin :

$$ddi \rightarrow \varepsilon$$

Nous avons donc la chaîne de production suivante :

$$s_{i7} \rightarrow i(I,7) \rightarrow di(V,7)(I,7) \rightarrow ddi(II,7)(V,7)(I,7) \rightarrow dddi(VI,7)(II,7)(V,7)(I,7)$$

$$\rightarrow (VI,7)(II,7)(V,7)(I,7)$$

Ou encore dans une écriture synthétique :

$$s_{i7} \rightarrow (VI,7)(II,7)(V,7)(I,7)$$

La règle 1

Ajoutons maintenant la règle 1 dite de "dédoublement", cette règle correspond dans notre formalisme à un "retard" à l'application des règles de substitutions.

$$\begin{array}{ll} R_{1-1} & /x/ \rightarrow /x\ x/ \\ R_{1-2} & /x7/ \rightarrow /x\ x7/ \\ R_{1-3} & /xm7/ \rightarrow /x\ xm7/ \end{array}$$

Revenons d'abord aux séquences de longueur 2, la règle de "dédoublement" s'écrit en fait :

$$x7 \rightarrow x7(X, \varepsilon)$$

Pour les séquences de longueur deux :

$$dx7 \rightarrow dx7(D(X), \varepsilon)$$

Mais aussi, à considérer le fait que l'on peut obtenir :

$$dx7 \rightarrow dx7(S(X), 7)$$

De même :

$$sx7 \rightarrow sx7(S(X), \varepsilon)$$

$$sx7 \rightarrow sx7(D(X), 7)$$

Comme dans la section précédente, on obtient la définition suivante :

Definition 13 (La règle 1):
Soit u un élément de N qui contienne p fois l'élément d et q fois l'élément s, soit u_1 un élément de N qui contienne $p+1$ fois l'élément d et q fois l'élément s, soit u_2 un élément de N qui contienne p fois l'élément d et $q+1$ fois l'élément s, alors on a :

$$u \rightarrow \varepsilon$$

$$u \rightarrow u(D^p \circ S^q(X), 7)$$

$$u \rightarrow u_1(D^p \circ S^{q+1}(X), 7)$$

$$u \rightarrow u_2(D^{p+1} \circ S^q(X), 7)$$

avec $p+q=n$.

Exemple 20:
Nous montrons dans la suite comment obtenir la séquence :

$$(V\#,7)(I\#,7)(V,7)(I,7)$$

Voici les règles que nous appliquons pour obtenir cette séquence :

$$s_{i7} \rightarrow i7(I,7)$$

$$i7 \rightarrow di7(D(I),7)$$

$$di7 \rightarrow di7(S(I),7)$$

$$di7 \rightarrow ddi7(D \circ S(I),7)$$

$$ddi7 \rightarrow \varepsilon$$

Et la chaîne de production de la séquence :

$$s_{i7} \rightarrow i7(I,7) \rightarrow di7(V,7)(I,7) \rightarrow di7(II\#,7)(V,7)(I,7)$$
$$\rightarrow ddi7(V\#,7)(II\#,7)(V,7)(I,7) \rightarrow (V\#,7)(II\#,7)(V,7)(I,7)$$

La règle 2

La règle 2 est particulière car elle permet de faire suivre un accord de septième par sa sous-dominante. Son utilisation est loin d'être systématique, nous n'en montrerons qu'une application.

Règle 2 : $x(m)(7) \rightarrow x(m)(7)Sd_x$

$$\begin{array}{ll} R_{2-1} & /x/ \rightarrow /x\ S_{dx}/ \\ R_{2-2} & /x7/ \rightarrow /x7\ S_{dx}7/ \\ R_{2-3} & /xm7/ \rightarrow /xm7\ S_{dx}7/ \end{array}$$

Nous utiliserons la définition suivante :

Definition 14:
Soit u un élément de N contenant la lettre x, alors on a :

$$u \rightarrow ud_x$$

et

$$d_x \rightarrow (S_d(a),b)$$

Exemple 21:
Nous montrons dans l'exemple suivant comment obtenir la cadence $(I,7)(IV,7)$.
On applique successivement les règles suivantes :

$$s_i \rightarrow id_i$$
$$i \rightarrow i(I,7)d_i$$
$$d_i \rightarrow i(I,7)(IV,7)$$
$$i \rightarrow \varepsilon$$

Et on obtient le schéma de dérivation suivant :

$$s_i \rightarrow id_i \rightarrow i(I,7)d_i \rightarrow i(I,7)(IV,7) \rightarrow (I,7)(IV,7)$$

Les règles spéciales (ou "de la finitude de N, l'explication") :

Afin d'introduire les règles spéciales, remarquons tout d'abord un cas particulier de génération de séquence.

Soit u une séquence harmonique générée par notre grammaire et tel qu'elle suive le schéma de génération suivant :

$$s_x \rightarrow x(X,7) \rightarrow dx(D(X),7) \cdots \rightarrow d^{10}x(D^{10}(X),7) \rightarrow d^{11}x(D^{11}(X),7)$$

Si l'on décide de générer un accord supplémentaire qui soit en rapport de quinte avec le dernier accord généré, on obtient avec les règles précédentes les symboles suivants :

$$d^{12}x(D^{12}(X),7)$$

Or comme nous l'avons dans la Propriété 1, $D^{12} = Id$. Nous appliquerons donc la règle suivante :

$$d^{11}x(D^{11}(X),7) \rightarrow x(X,7)$$

La séquence ainsi produite parcourt en fait le cycle des quintes en entier.

De l'étude de ce cas particulier, nous tirons un ensemble de règles qui viennent s'ajouter aux règles déjà existantes.

La propriété 1 énoncée précédemment nous affirme qu'il existe un nombre fini de combinaisons des fonctions D et S qui égalent l'*identit*. Par cette propriété, nous définissons pour chaque combinaison une règle spéciale.

Definition 15 (Les règles Spéciales):
Soit u un élément de N tel que :
1) u contienne x avec $x \in \{i, i\#, ii, ii\#, ...vii, i7, i\#7, ii7, ii\#7, ...vii7$
2) u soit tel qu'il contienne $p-1$ fois le symbole s et q fois le symbole d.
3) $(p,q) \in \{(12,0)(5,1)(10,2)(3,3)(8,4)(1,5)(6,6)(11,7)(4,8)(9,9)(2,10)(7,11)(0,12)\}$
Alors on a les règles suivantes :

$u \to x(X, \varepsilon)$ *si* u *ne contient pas de* 7 $u \to x(X, 7)$ *sinon.*

De manière symétrique, soit u *un élément de* N *tel que :*
1) u *contienne* x *avec* $x \in \{i, i\#, ii, ii\#, ...vii, i7, i\#7, ii7, ii\#7, ...vii7$
2) u *soit tel qu'il contienne* p *fois le symbole* s *et* $q-1$ *fois le symbole* d.
3) $(p, q) \in \{(12, 0)(5, 1)(10, 2)(3, 3)(8, 4)(1, 5)(6, 6)(11, 7)(4, 8)(9, 9)(2, 10)(7, 11)(0, 12)$
Alors on a les règles suivantes :

$u \to x(X, \varepsilon)$ *si* u *ne contient pas de* 7,
$u \to x(X, 7)$ *sinon.*

Il est à noter que l'alphabet non-terminal est d'une taille assez inhabituelle, et que le nombre de règles l'est tout autant, c'est en fait ce qui nous permet de prendre en compte toutes les possibilités de grilles que l'on peut obtenir par les règles 1, 2 et 3 de la grammaire de Steedmann.

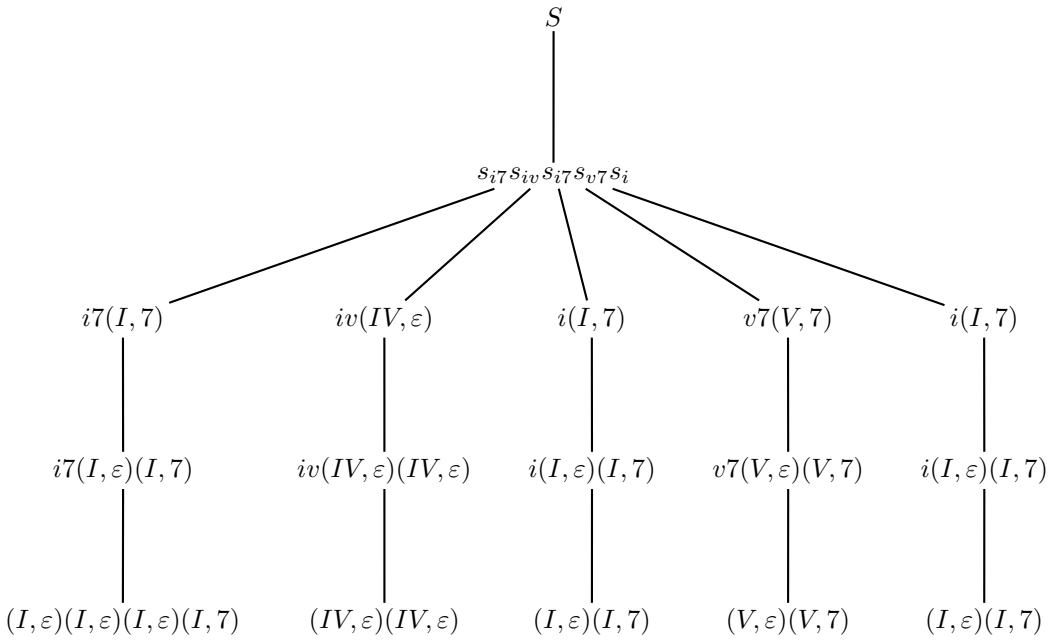
4.3 La grammaire algébrique ou générations de grilles complètes

La méthode que nous proposons pour générer des grilles complètes consiste à insérer des cadences sur une "ossature" de base.

Nous présentons tout d'abord un exemple simple, puis nous montrons comment obtenir le fameux *Blues for Alice* de Charlie Parker.

$$S \to s_{i7} s_{iv} s_{i7} s_{v7} s_i$$

L'arbre de dérivation pour obtenir une grille de blues standard est :



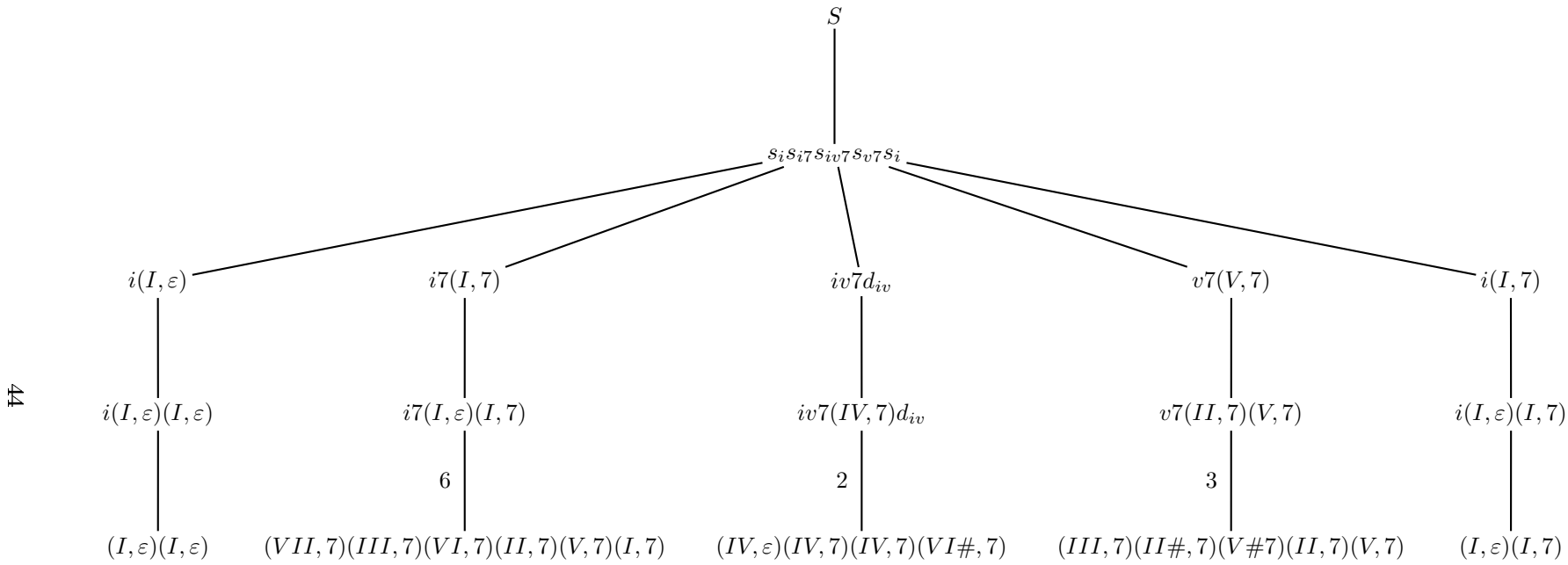
En considérant que chaque accord dure une mesure, on a le résultat escompté :

I	I	I	$I7$
IV	IV	I	$I7$
V	$V7$	I	$I7$

Pour obtenir le *Blues for Alice* il nous faut tenir compte de la règle 2 de Steedman :

Règle 2 : $x(m)(7) \rightarrow x(m)(7)\ Sd_x$

Cette règle, qui selon Steedman est à utiliser avec parcimonie, est ici nécessaire une fois pour générer la sixième mesure de la grille voulue. Montrons tout d'abord l'arbre de dérivation :


S
$s_i s_{i7} s_{iv7} s_{v7} s_i$
$i(I,\varepsilon)$
$i7(I,7)$
$iv7d_{iv}$
$v7(V,7)$
$i(I,7)$
$i(I,\varepsilon)(I,\varepsilon)$
$i7(I,\varepsilon)(I,7)$
$iv7(IV,7)d_{iv}$
$v7(II,7)(V,7)$
$i(I,\varepsilon)(I,7)$
6
2
3
$(I,\varepsilon)(I,\varepsilon)$
$(VII,7)(III,7)(VI,7)(II,7)(V,7)(I,7)$
$(IV,\varepsilon)(IV,7)(IV,7)(VI\#,7)$
$(III,7)(II\#,7)(V\#7)(II,7)(V,7)$
$(I,\varepsilon)(I,7)$

On remplace les symboles non terminaux grâce aux chaînes de dérivation suivantes :

$$i \xrightarrow{1} (I, \varepsilon)$$

$$i7 \xrightarrow{6} (VII, 7)(III, 7)(VI, 7)(II, 7)(V, 7)$$

$$v7 \xrightarrow{5} (III, 7)(II\#, 7)(V\#7)(II, 7)$$

$$iv7 \xrightarrow{2} (IV, 7)(VI\#7)$$

Et avec le bon découpage, on obtient :

I	$VII7$ / $III7$	$VI7$ / $II7$	$V7$ / $I7$
IV	$IV7$ / $VI\#7$	$III7$	$II\#7$ / $V\#7$
$II7$	$V7$	I	$I7$

Ce qui correspond à la grille voulue si l'on ignore les terminaisons mineures des accords de septième.

La prise en compte des accords mineur est une des perspectives de ce travail. Il ne devrait pas être trop difficile d'en tenir compte, la résistance provient, à notre avis, avant tout du choix du formalisme.

Chapitre 5

Propriété sur les séquences harmoniques "steedmaniennes"

Au vue de la forme que prennent les règles de génération d'une cadence que nous appellerons *steedmannienne*, nous avons dégagé une propriété sur l'ensemble des mots du langage généré par une grammaire de Steedman. Nous utilisons de plus la propriété dégagée par Marc Chemillier dans son article [5].

La première proposition ne concerne que les séquences d'accords modifiés par les règles 3 et 4 de Steedman. Les règles 3 et 4 ne modifient pas la longueur des séquences d'accords car elles ont le même nombre de termes à gauche de la règle, qu'à droite. Nous traiterons ensuite le cas de la règle 1 qui elle modifie la longueur de la séquence en ajoutant un élément à chaque fois qu'elle est appliquée.

Dans les propositions suivantes, une suite d'accord est notée $u_n \dots u_1$ et non pas $u_1 \dots u_n$. Nous nous sommes autorisés cet artifice de notation car il ne modifie en rien la démonstration et permet une plus grande clarté du raisonnement.

5.1 Proposition ne prenant en compte que les règles 3 et 4

Proposition 1 (Règles 3 et 4):
Soit une séquence harmonique de longueur n, *notée* $u_n \dots u_1$ *où* $u_k = (a_k, b_k)$, *avec* $b_1 = 7$ *et* $b_k = \varepsilon$ *pour tout* i *tel que* $1 < k \leq n$.

Soit une séquence de longueur n *notée* $U_n \dots U_1$ *où* $U_k = (A_k, B_k)$ *obtenue par application des règles 3 et 4 de la grammaire de Steedman sur la séquence* $u_n \dots u_1$.

Alors pour tout élément U_k , $k \leq n$, *on a :*

$$U_k = \begin{cases} u_k \text{ et c'est vrai pour tout } l > k \\ \text{ou} \\ (D^p \circ S^q(a_1), 7) \quad \text{où } p+q = k-1 \end{cases}$$

Exemple 1 :
Soit la séquence

$$(vi, 7)(ii, 7)(v, 7)(i, 7)$$

obtenue par trois applications de la règle 3 :$w\ x7 \rightarrow D_x7\ x7$ sur n'importe quelle séquence de quatre accords se terminant par $(i, 7)$ et ne contenant pas d'autre accord de septième que le dernier par exemple :

$$(i, \varepsilon)(i, \varepsilon)(i, \varepsilon)(i, 7)$$

La séquence originale et la séquence modifiée sont de même longueur et chaque élément U_i de la séquence s'écrit comme suit :

$$U_k = (D^p(i), 7)$$

où $p = k - 1$. En effet :

$$(vi, 7) = (D^3(i), 7),\ (ii, 7) = (D^2(i), 7),\ (v, 7) = (D(i), 7)$$

Démonstration :
Nous démontrons cette propriété par récurrence sur l'indice de l'accord considéré.

Soit une séquence quelconque de longueur n obtenue par application des règles 3 et 4 de Steedman sur une séquence qui ne contenait pas d'autres accords de septième que le dernier.

Montrons que l'avant-dernier accord, U_2, vérifie la propriété énoncée :

Soit il n'a pas été modifié et alors $U_2 = u_2$

Soit il a été généré par l'application directe de la règle 3 ou de la règle 4 et alors il est de la forme :

$$\begin{cases} (D(a_1), 7) \\ \text{ou} \\ (S(a_1), 7) \end{cases}$$

Supposons maintenant que les k derniers accords, $k \leq n - 1$, vérifient la propriété énoncée précédemment, alors on a la propriété pour U_k

$$U_k = \begin{cases} U_k \\ \text{ou} \\ (D^p \circ S^q(a_1), 7) \quad \text{où } p+q = i-1 \end{cases}$$

Montrons que la propriété est vraie pour le $(k+1)^{me}$ accord.

Si u_k n'est pas un accord de septième, on ne peut appliquer ni la règle 3, ni la règle 4 et u_{k+1} ne peut être modifié, donc $U_{k+1} = u_{k+1}$.

Si u_k est un accord de septième :

Soit on ne modifie pas u_{k+1} auquel cas $U_{k+1} = u_{k+1}$.

Soit, en appliquant les règles 3 ou 4 sur $u_{k+1}u_k$, on obtient respectivement $U_{k+1} = (D(a_k), 7)$ ou $(S(a_k), 7)$.

Or par la propriété de récurrence, U_k est de la forme $(D^p \circ S^q(a_1)), 7)$ où $p+q = k-1$.

Donc

$$U_{k+1} = \begin{cases} u_{k+1} \\ \text{ou} \\ (S \circ D^p \circ S^q(a_1)), 7) \text{ où } p+q = k-1 \\ \text{ou} \\ (D \circ D^p \circ S^q(a_n)), 7) \text{ où } p+q = k-1 \end{cases}$$

Or comme S et D commutent (cf. proposition 1) :

$$U_{k+1} = \begin{cases} u_{k+1} \\ \text{ou} \\ (D^p \circ S^{q+1}(a_1), 7) \text{ où } p+q = k-1 \\ \text{ou} \\ (D^{p+1}(S^q(a_n)), 7) \text{ où } p+q = k-1 \end{cases}$$

Soit après un changement d'indice :

$$U_{k+1} = \begin{cases} u_{k+1} \\ \text{ou} \\ (D^p \circ S^q(a_n)), 7) \quad \text{où } p+q = k \end{cases}$$

5.2 Proposition prenant en compte les règles 1,3 et 4

Nous traitons ensuite le cas des séquences générées avec la règle 1 de dédoublement.

Proposition 2 (Les règles 1, 3 et 4):
Soit une séquence harmonique de longueur m, notée $u_m \dots u_1$ où $u_k = (a_k, b_k)$, avec $b_1 = 7$ et pour tout $k \geq 2, b_k = \varepsilon$.

Soit une séquence de longueur n,$n \geq m$ notée $U_n \dots U_1$ où $U_k = (A_k, B_k)$ obtenue par application des règles 1, 3 et 4 de la grammaire de Steedman sur la séquence $u_n \dots u_1$.

Alors pour tout élément U_k , $i \leq n$, on a :

$$U_k = \begin{cases} u_{k+l} \\ ou \\ (a_{k-1}, \varepsilon) \\ ou \\ (D^p \circ S^q(a_1), 7) \quad où\ p+q = k-1-l \end{cases}$$

où l est le nombre d'applications de la règle 1 sur les accords situés après u_k.

Exemple 2 : Soit la séquence

$$(v\#, 7)(i\#, 7)(v, 7)(i, 7)$$

obtenue par le processus suivant :

$$(i, \varepsilon)(i, 7) \rightarrow (v, 7)(i, 7) \rightarrow (i\#, 7)(i, 7) \rightarrow (i\#, \varepsilon)(i\#, 7)(i, \varepsilon)(i, 7) \rightarrow (v\#, 7)(i\#, 7)(v, 7)(i, 7)$$

On vérifie alors que :

$$(v, 7) = (D(i), 7); (i\#, 7) = (S(i), 7); (v\#, 7) = (D(S(i)), 7)$$

Remarques :

On remarque que $U_1 = u_1$.

En fait $l = m - n$. Seule la règle 1 augmente la taille de la chaîne, ajoutant 1 élément à chaque utilisation.

Démonstration :

Comme pour la proposition précédente nous allons démontrer la proposition par récurrence sur l'indice de l'élément considéré.

Soit une séquence quelconque de longueur n obtenue par application des règles 1, 3 et 4 de Steedman sur une séquence qui ne contenait pas d'autres accords de septième que le dernier.

Montrons que l'avant dernier accord vérifie la propriété énoncée : Soit il n'a pas été modifié par une des règles de Steedman et $U_2 = u_2$.

Soit il a été généré avec la règle 1 auquel cas on a : $U_2 = (a_1, \varepsilon)$.

Soit il a été généré par les règles 3 ou 4 et alors U_2 est bien de la forme voulue :

$$\begin{cases} (D(a_1), 7) \\ \text{ou} \\ (S(a_1), 7) \end{cases}$$

Supposons la propriété vraie pour tous les accords jusqu'au i^{me} avec $i < n-1$ et montrons qu'elle est vraie pour l'accord U_{i+1}.

Si $U_k = u_{k+l}$, alors : Seule la règle 1 peut-être appliquée car la séquence initiale ne contenait pas d'autre accord de septième que le dernier. l est augmenté de 1 et $U_{k+1} = u_{k+l+1}$.

Si $U_k = (a_{k-1}, \varepsilon)$ alors : Seule la règle 1 $(a, \varepsilon) \rightarrow (a, \varepsilon)(a, \varepsilon)$ peut-être appliquée et $U_{k+1} = (a_k, \varepsilon)$.

Si $U_k = (D^p(S^q(a_1)), 7)$ où $p + q = k - 1 - l$

Alors par application des règles 3 ou 4 et par changement d'indice on trouve l'identité voulue comme dans la propriété précédente.

5.3 La notion de vecteur-steedman

A l'aide des propriétés décrites précédemment, j'ai écrit deux programmes en Common Lisp qui prend en argument un accord et renvoie une chaîne respectant les susdites propriétés.

La première fonction *change−accord* que j'ai écrite renvoie un accord modifié n fois. Cette modification s'effectue itérativement en choisissant aléatoirement à chaque itération la fonction D ou la fonction S. Si on note $(a, 7)$, l' accord pris en argument, le résultat est bien de la forme voulue : $(D^p \circ S^q(a), 7)$ avec $p + q = n$.

Exemple 22:

```
? (change-accord '(i 7) 2)
(v 7)
? (change-accord '(i 7) 2)
(i# 7)

? (change-accord '(i 7) 4)
(vi 7)
```

Pour construire la séquence complète de taille n, il suffit alors de construire la séquence élément par élément en décrémentant n à chaque élément. La fonction *cadence* réalise cette tâche. Elle prend en argument un accord et le nombre d'accords que la séquence voulue doit contenir.

Exemple 23:

```
? (cadence '(i 7) 4)
((vi 7) (ii 7) (i# 7) (i 7))
? (cadence '(i 7) 4)
((ii# 7) (ii 7) (i# 7) (i 7))

? (cadence '(i 7) 8)
((i# 7) (iv# 7) (iv 7) (iii 7) (vi 7) (ii 7) (v 7) (i 7))
```

La troisième fonction est directement issue de la seconde. Elle vise à produire des séquences vérifiant la propriété 2, donc générées par les règles 1, 3 et 4 de Steedman. La difficulté pour écrire cette fonction vient de la prise en compte de

la règle 1 de "dédoublement". Comme nous l'avons déjà mentionné, la règle 1 correspond à un retard à l'application des règles 3 et 4. Pour prendre en compte ce retard, nous définissons un *vecteur-steedman*. Chaque élément de ce vecteur correspond à la variable n, qui dans l'application précédente est décrémentée à chaque génération d'un élément. Ainsi pour la fonction *cadence* la valeur de ce *vecteur-steedman* pour une séquence de taille n est : $(n\ n-1 \ldots 2\ 1)$.

Pour tenir compte de la règle 1, nous spécifions différemment le *vecteur-steedman*. Ce vecteur doit être décroissant, chaque élément est soit égal au précédent soit égal à l'élément précédent moins un et le dernier élément doit être égal à un.

Exemple 24:

```
? (vecteur-steedman 4)
(3 3 2 1)
? (vecteur-steedman 4)
(2 2 2 1)

? (vecteur-steedman 6)
(4 4 3 2 1 1)
```

La fonction *cadence-vecteur* prend en argument un accord et un vecteur de type *vecteur-steedman* générant une séquence d'accords respectant la propriété 2.

Exemple 25:

```
 ? (cadence-steedman '(i 7) 4)
((ii 7) (v 7) (v 7) (i 7))
 ? (cadence-steedman '(i 7) 4)
((ii 7) (v 7) (i# 7) (i 7))

 ? (cadence-steedman '(i 7) 6)
((vi# 7) (vi 7) (ii 7) (i# 7) (v 7) (i 7))
```

Ici encore il conviendrait d'affiner cette fonction pour qu'elle puisse générer les accords de type m et $m7$.

Bilan et Perspectives

Le bilan de ce stage est positif à plus d'un titre.

Le nouveau formalisme que nous avons dégagé devrait permettre de prendre en compte, de manière plus rigoureuse, des subtilités harmoniques ignorées dans les modèles précédents.

D'autres propriétés concernant les séquences harmoniques devraient pouvoir être dégagées. La prise en compte des accords m et $m7$ est sans aucun doute la prochaine étape concernant ce sujet. Il semble d'ailleurs que la génération d'accords $m7$ est directement liée à la parité de la variable n qui représente la distance à l'accord générateur. Cette question est à étudier.

La décontextualisation devraient permettre de construire un automate à mémoire en pile reconnaissant les grilles engendrées par la grammaire de Steedman. Le langage étudié étant de type 2 et non de type 1, nous sommes assurés de l'existence de cet automate.

Le projet O'Max devrait s'enrichir de ces travaux. La partie contrôlant les substitutions harmoniques doit être en partie repensée pour inclure une plus grande réactivité. Les fonctions que nous avons montrées à la fin de cet exposé sont prometteuses. Il est toutefois nécessaire de réfléchir sur certains points :

- Doit-on implémenter l'ensemble des grilles possibles en dur ou garder l'idée de générer en "temps-réel" les substitutions ?
- Quel(s) moyen(s) de contrôle utiliser ?
- Jusqu'à quel niveau les substitutions sont-elles pertinentes ?

Enfin, comme je l'ai exposé dans mon introduction, l'improvisation est une pratique extrêmement diversifiée présentant de nombreux aspects. Il serait intéressant de travailler à l'élaboration d'un formalisme permettant de rendre compte de cette diversité et d'en implémenter un modèle. Ce travail pourrait faire l'objet d'une thèse.

Bibliographie

[1] CARLES P., COMOLLI J.-L., Free Jazz/Black Power, Gallimard-Folio 2000

[2] BAYLEY D., L'improvisation sa nature et sa pratique, Outre Mesure 1999

[3] AUTEBERT J.M., Théories des langages et des automates, Masson 1994

[4] ASSAYAG G., AGON C., Logiciel OpenMusic, Cdrom Forum Ircam, 1997.

[5] CHEMILLIER M., Formal structure of jazz chord sequences generated by Steedman's grammar , To appear in Soft Computing, special issue on Formal Systems and Music, G. Assayag, V. Caffagna, M. Chemillier (eds.)

[6] CHEMILLIER M., Monoïde libre et musique, RAIRO Informatique théorique, vol. 21, n° 3 et 4, 1987, 341-371, 379-417.

[7] CHEMILLIER M., TIMIS D., Toward a theory of formal musical languages, Proceedings of the ICMC 88, Cologne, 1988, 175-183.

[8] CHEMILLIER M., Structure et méthode algébriques en informatique musicale, Thèse, Université Paris 7, 1990.

[9] CHEMILLIER M., Langages musicaux et automates : la rationalité du langage sériel, Actes du colloque Musique et assistance informatique, Marseille, 1990, 302-318.

[10] CHEMILLIER M., Aspects mathématiques des applications en informatique musicale des automates finis, Séminaire logique & Algorithmique, vol. IX, université de Caen, 1992, 31-61.

[11] CHEMILLIER M., Automata and music, Proceedings of the ICMC 92, San-Jose, 1992, 370-371.

[12] CHEMILLIER M., PACHET F. (eds.), Recherches et applications en informatique musicale, Hermès, 1998.

[13] CHEMILLIER M., Générateurs musicaux et singularités, JIM 99, 6èmes journées d' informatique musicale, CNET-CEMAMu, Issy-les-Moulineaux, 1999, 167-177.

[14] COKER J., Improvising Jazz, 1964, réed. Fireside, 1987.

[15] PACHET F., Computer Analysis of Jazz Chord Sequences : Is Solar a Blues ?, MIRANDA E.R. (ed.), Readings in Music and AI, Harwood, 1998.

[16] PACHET F., CARRIVE J., Intervalles temporels circulaires et application à l ?analyse harmonique, CHEMILLIER M., PACHET F. (eds.), Recherches et applications en informatique musicale, Hermès, 1998, 17-30.

[17] PACHET F., Sur la structure algébrique des séquences d ?accords de jazz, JIM 98, 5èmes journées d'informatique musicale , LMA Marseille, La Londe-les-Maures, 1998.

[18] PACHET F., Surprising Harmonies, JIM 99, 6èmes journées d'informatique musicale, CNET-CEMAMu, Issy-les-Moulineaux, 1999, 187-206.

[19] STEEDMAN M.J., A Generative Grammar for Jazz Chord Sequences, Music Perception, vol. 2, n°1, 1984, 52-77.

[20] STEEDMAN M.J., The Blues and the Abstract Truth : Music and Mentals Models, GARNHAM A., OAKHILL J. (eds.), Mentals Models in Cognitive Science, Erlbaum, Mahwah, NJ, 1996.

[21] SIRON J., La Partition Intérieure/Jazz et musiques improvisées, Outre-Mesure 1992.