

THESE DE DOCTORAT DE L'UNIVERSITE PARIS VI

Spécialité : **Informatique**

Option : **Intelligence Artificielle**

présentée par

Geber Lisboa Ramalho

pour obtenir le grade de

Docteur de l'Université Paris VI

Sujet de la thèse :

CONSTRUCTION D'UN AGENT RATIONNEL JOUANT DU JAZZ

Soutenue le 13 janvier 1997

Devant le jury composé de MM :

Denis BAGGI	<i>rapporteur</i>
Claude CADOZ	<i>examineur</i>
Jean-Gabriel GANASCIA	<i>directeur</i>
Jean-Paul HATON	<i>examineur</i>
François PACHET	<i>examineur</i>
Gerhard WIDMER	<i>rapporteur</i>

A Lisiane et mes enfants
Luís Felipe et Tales

There is a Japanese visual art in which the artist is forced to be spontaneous. He must paint on a thin stretched parchment with a special brush and black water paint in such a way that an unnatural or interrupted stroke will destroy the line or break through the parchment. Erasures or changes are impossible. These artists must practice a particular discipline that of allowing the idea to express itself in communication with their hands in such a direct way that deliberation cannot interfere.

The resulting pictures lack the complex composition and textures of ordinary painting, but it is said that those who see well find something captured that escapes explanation.

This conviction that direct deed is the most meaningful reflection, I believe, has prompted the evolution of the extremely severe and unique disciplines of the jazz or improvising musician.

Bill Evans dans la pochette de "Kind of Blue" (Davis, 1962)

REMERCIEMENTS

Mes remerciements vont d'abord à mon directeur de thèse Jean-Gabriel Ganascia par son ouverture d'esprit, en acceptant guider une thèse sur la musique, lui n'étant pas musicien. Je le remercie aussi pour toutes les suggestions et, en général, pour tout ce qu'il m'a appris sur la science, sur l'IA, sur comment écrire des articles et faire des présentations. Ce sont des enseignements très précieux que je garde pour le reste de ma vie.

J'exprime mon énorme gratitude à François Pachet, excellent musicien, chercheur en IA et grand programmeur Smalltalk, qui a joué un rôle indispensable dans le développement de ma thèse. Nos discussions ont été toujours fructueuses et nos soirées musicales intéressantes.

Je remercie sincèrement Gerhard Widmer et Denis Baggi qui ont fait l'honneur d'être rapporteurs de cette thèse, en faisant plusieurs commentaires utiles, féconds et éclairés sur mon travail. Je remercie aussi Jean-Paul Haton pour ces remarques et questions qui m'ont aidé à mieux comprendre ce que j'avais fait. Je remercie, enfin, Claude Cadoz, le président du jury qui a fait la gentillesse de bien vouloir lire le manuscrit en laissant plein de remarques très pertinentes qui m'ont données des idées pour la suite.

Quelques musiciens, en particulier Eriberto Paredes, Antoine Espagno et Paul Hodgson, m'ont beaucoup aidé à comprendre la nature des connaissances musicales et les processus de création. Je les suis très reconnaissant.

Je remercie tous les thésards et maîtres de conférences du LAFORIA, en particulier les membres de l'équipe ACASA, avec qui j'ai pu échanger des idées, scientifiques et autres pendant plus de quatre ans. Il a été pour moi un grand privilège de pouvoir les côtoyer. Ils m'ont appris beaucoup sur la civilisation et la culture française, ils m'ont fait des suggestions et remarques fondamentales pour mon travail et ils ont même relu des parties du texte de ma thèse. Très particulièrement, je tiens à remercier Vincent Corruble, Jean-Daniel Zucker, Stephan Grolimund et Pierre-Yves Roland. Je pense aussi à Christophe Meyer, Bernard Leroux, Jérôme Thomas, Raphaël Rispoli, Herculano Caetano, Jean-Philippe Jacquet, Zahia Guessoum, Pierre Vinant et Cédric Thiénot.

Un remerciement particulier à Rosalind Greenstein, qui a corrigé et nettement amélioré mon article au AAAI qui a été une percée décisive dans ma thèse, et à et Hélène Giroire, qui a relu la dernière version du manuscrit de thèse en faisant, avec beaucoup de précision, des corrections très opportunes. Mes remerciements aussi à Fabrice Toledano qui a "numérisé" les lignes de base de Ron Carter qui ont servi de base pour mon système.

Il ne faut pas que j'oublie de remercier tous les amis qui pendant mon séjour en France ont donné à moi et ma famille un soutien matériel et spirituel très essentiel.

Ma femme Lisa mérite tous mes remerciements les plus chaleureux, sincères et profonds. Elle a courageusement et généreusement venu avec moi en France en fonction de mes intérêts mais elle n'a jamais me fait ressentir les difficultés de son choix. Tout au contraire, elle m'a toujours soutenu, encouragé et aidé dans la maison, dans le travail et partout ailleurs. Un merci infini !!

J'oublie certainement des noms car, heureusement, nombreuses sont les personnes qui, d'une façon ou d'une autre, m'ont aidé dans le parcours que je viens d'achever. Je les remercie à tous.

Enfin, je remercie le Ministère de l'Éducation du gouvernement brésilien à travers la CAPES pour m'avoir accordé le financement de ce projet de recherche et d'avoir toujours cru, à son aboutissement.

RESUME

La recherche en intelligence artificielle a déjà produit des programmes ayant une bonne expertise pour des tâches comme la classification et la prédiction. Les tâches de conception et de création, dont les artistiques, semblent poser plus de problèmes. C'est dans ce contexte, que nous avons entrepris un travail visant étudier la possibilité des techniques actuelles de l'Intelligence Artificielle de modéliser la connaissance nécessaire pour construire une machine capable de créer et jouer, en temps réel, des lignes de basse de jazz. En faisant appel à la notion d'agent rationnel, nous avons conçu un modèle de résolution de problèmes original qui intègre divers types de connaissances et, en particulier, des raisonnements à partir de cas et de règles. Selon ce modèle, la tâche d'un agent jazzman se déroule en trois étapes qui se succèdent continuellement. Premièrement, à travers une analyse harmonique particulière, l'agent établit le "segment" de la grille pour lequel il va créer une phrase musicale. Deuxièmement, en fonction de ses données perceptives (la grille d'accords, et ce que lui et les autres musiciens viennent de jouer), l'agent active un ensemble d'actions potentielles (PACTs), comme "jouer peu dissonant" et "jouer beaucoup de notes". Ces PACTs peuvent être vues comme des "consignes musicales" portant sur plusieurs "traits musicaux" (par exemple, dissonance et densité) de la phrase mélodique devant être jouée. Troisièmement, l'agent récupère de sa "Mémoire Musicale" (une base de cas contenant des fragments de lignes de basse jouées par un bassiste humain) le fragment qui respecte le mieux les PACTs activées. Nous avons implanté le système ImPact, contenant 267 cas et plus de 80 règles de production, et les résultats musicaux obtenus sont aussi bons que ceux des deux "meilleurs" programmes actuels pouvant générer des lignes de basse (Band-in-box et NeurSwing). En outre, notre modèle a pour principal avantage de rendre facile l'ajout de nouvelles connaissances.

ABSTRACT

Artificial intelligence programs have reached a certain success in performing some classes of tasks such as classification and prediction. However, the design and creation tasks, particularly the artistic ones, seem to be more problematic. In this context, our work has been devoted to the study of the strengths and limitations of current AI techniques in modelling the knowledge necessary to build a machine capable of creating and playing jazz bass lines in real time. Based on the notion of rational agent, we have proposed a original problem solving model that integrates different kinds of knowledge and reasoning, in particular case and rule based ones. According to our model, the task of a jazz player agent can be divided into three main steps, which occur continuously. First, by means of a particular harmonic analysis, the agent establishes the "temporal segment" within which a melodic phrase must be created and played. Second, the agent activates an ensemble of "Potential ACTIONs" (PACTs), such as "play with high dissonance" and "play a lot of notes". These PACTs can be seen as musical instructions concerning different musical facets (e.g. dissonance and density) of the melodic phrase to be created and played. Third, the agent uses its "Musical Memory" (a case base containing melodic previously played by human musicians), to retrieve the fragment that fits the best the activated PACTs. To validate our model, we have implemented the system ImPact, containing 267 cases and more than 80 production rules. The musical results have been as good as those of the two "best" current programs which can generate bass lines (Band-in-a-box and NeurSwing). The main advantage of our model is that additional knowledge can be easily incorporated to improve the system's musical performance.

TABLES DES MATIERES

AVANT-PROPOS: UNE VERSION DU TEST DE TURING POUR LE JAZZ?

INTRODUCTION	1
A - Recherches dans les domaines artistiques : les enjeux pour l'IA.....	1
a) La démarche d'IA en tant que science expérimentale.....	1
b) Motivations pour l'IA de la modélisation des activités artistiques	4
c) Quelques difficultés posées par les activités artistiques.....	6
B - Cadre de la thèse.....	9
a) De l'art au bassiste de jazz	9
b) Approche, hypothèses de travail et objectifs.....	12
C - Notre contribution.....	16
a) Brève description de la tâche du bassiste de jazz	16
b) Description de notre modèle	17
1. CARACTERISATION DE LA TACHE D'UN AGENT JAZZMAN	33
1.1. La notion d'agent rationnel	34
1.1.1. Comportement et structure.....	34
1.1.2. Environnement	35
1.2. L'agent jazzman	37
1.2.1. Données perceptives	37
1.2.2. Environnement.....	39
1.2.3. Le fonctionnement global	40
1.2.4. Une formalisation de la tâche	42
1.2.5. Quelques problèmes posés.....	44
2. CONNAISSANCES ET RAISONNEMENT CHEZ LES JAZZMEN.....	47
2.1. Habileté instrumentale (skill)	48
2.2. Écoute	49
2.2.1. Dans l'apprentissage	49
2.2.2. Structures et multiples traits.....	49
2.2.3. Anticipation et évaluation dynamique	51
2.3. Fondements en théorie musicale	52

2.3.1.	La grille d'accords	53
2.3.2.	La construction même de l'improvisation	55
2.4.	Au-delà du "note par note" et du "la note à cause de la note"	57
2.4.1.	Patterns	57
2.4.2.	Traits musicaux	59
2.5.	Pratique et apprentissage.....	61
2.6.	Conclusions	63
3.	CONNAISSANCES ET RAISONNEMENT DANS LES PROGRAMMES JOUANT DU	
	JAZZ	67
3.1.	Panorama des programmes	68
3.1.1.	Tâches	68
3.1.2.	Approches	69
3.2.	Données perceptives et environnement	71
3.2.1.	Problèmes de l'écoute	71
3.2.2.	Données auditives effectivement incorporées	72
3.2.3.	Prise en compte globale de l'environnement	73
3.3.	Action	74
3.4.	Raisonnement.....	75
3.4.1.	Fondements en théorie musicale	75
3.4.2.	Granularité du raisonnement	76
3.4.3.	Structures	79
3.4.4.	Traits musicaux	80
3.4.5.	Réutilisation de patterns	84
3.5.	Résultats	86
3.6.	Conclusions	87
4.	VUE D'ENSEMBLE DE NOTRE MODELE.....	89
4.1.	Perception.....	89
4.1.1.	La grille	90
4.1.2.	La notion de Scénario	90
4.1.3.	Le module de perception.....	93
4.2.	Exécution	93

4.3. Raisonnement.....	96
4.3.1. Improvisation vue comme une résolution de problèmes	96
4.3.2. Segmentation de la grille et granularité du raisonnement.....	98
4.3.3. Fixation des critères concernant les notes à jouer.....	100
4.3.3.1. Activation et sélection des PACTs	101
4.3.3.2. Assemblage des PACTs	104
4.3.4. La notion de Mémoire Musicale : une base de cas pour des fragments mélodiques de ligne de basse	109
4.3.4.1. L'évolution du rôle de la Mémoire Musicale	110
4.3.4.2. Indexation des cas.....	117
4.3.5. L'humeur de l'agent.....	122
5. VUE DETAILLEE DES PACTs	125
5.1. Considérations préliminaires sur la représentation des PACTs.....	125
5.1.1. Le répertoire PACTs de basse.....	125
5.1.2. Quel formalisme ?.....	127
5.2. Les PACTs standards.....	132
5.2.1. Représentation.....	132
5.2.2. Notation et autres notions de base	135
5.2.3. Activation et segmentation	137
5.2.4. Jouabilité	138
5.2.5. Préférence	139
5.2.6. Combinabilité.....	140
5.2.6.1. Les trois cas "canoniques"	141
5.2.6.2. Les autres cas	144
5.2.7. Combinaison	146
5.2.8. Propagation	148
5.3. Les PACTs transformationnelles	150
5.4. Les PACTs "à découper"	150
5.5. Les PACTs de la Mémoire Musicale.....	152
5.5.1. Représentation.....	154
5.5.2. Acquisition de cas et anti-propagation.....	156
5.5.3. Similarité.....	160
6. VUE DETAILLEE DU PROCESSUS D'IMPROVISATION	165

6.1. MusES : une plate-forme de représentation de connaissances en musique tonale.....	165
6.2. Architecture et boucles principales.....	166
6.3. Segmentation de la grille.....	170
6.3.1. Une analyse harmonique particulière.....	171
6.3.2. Détection des "chord chunks".....	172
6.3.3. Résolution des conflits.....	174
6.3.4. Remplissage des "trous".....	176
6.3.5. Influence et taille minimale de la Mémoire Musicale.....	177
6.4 La détermination des PACTs à être assemblées.....	179
6.4.1. Mise à jour de l'Humeur.....	179
6.4.2. Activation des nouvelles PACTs.....	180
6.4.3. Sélection des PACTs à être assemblées.....	183
6.5. Assemblage des PACTs.....	184
6.6. Utilisation de la Mémoire Musicale.....	187
6.6.1. Organisation de la Mémoire Musicale.....	187
6.6.2. Récupération et réutilisation des cas.....	190
7. IMPLEMENTATION ET EXPERIMENTATIONS.....	199
7.1. Le système ImPact.....	199
7.1.1. Données générales.....	200
7.1.2. Bases de règles et de cas.....	201
7.1.3. Les interfaces.....	201
7.2. Expérimentations.....	204
7.2.1. Évaluation empirique des différents aspects du modèle.....	205
7.2.1.1. Ajout de cas.....	205
7.2.1.2. Granularité.....	207
7.2.1.3. Adaptabilité.....	209
7.2.1.4. Activation et assemblage des PACTs.....	211
7.2.1.5. Scénario.....	216
7.2.1.6. Humeur.....	219
7.2.1.7. Le manque de temps.....	221
7.2.1.8. Bilan.....	224
7.2.2. Comparaison avec Band-in-a-box et NeurSwing.....	225
7.2.2.1. Développement.....	226

7.2.2.2. Diversité	227
7.2.2.3. Fluidité.....	229
7.2.2.4. Richesse mélodique	230
7.2.2.5. Richesse rythmique	231
7.2.2.6. Rendu auditif	231
7.2.2.7. Sens harmonique	232
7.2.2.8. Swing.....	233
7.2.2.9. Bilan	234
7.2.3. Comparaison avec des musiciens humains	235
CONCLUSIONS ET PERSPECTIVES.....	239
ANNEXE A - LIGNES DE BASSE JOUEES PAR RON CARTER (AABERSOLD, 1979).....	250
ANNEXE B - GLOSSAIRE MUSICAL	263
REFERENCES BIBLIOGRAPHIQUES ET DISCOGRAPHIQUES.....	267

LISTE DES FIGURES

Figure B.1 - Tâche du bassiste de jazz	17
Figure B.2 - Passage généré sur Feuilles Mortes	26
Figure B.3 - Passage généré sur FEUILLES MORTES (complété de B.2 - version 1)	28
Figure B.4 - Passage généré sur FEUILLES MORTES (complété de B.2 - version 2)	30
Figure 1.1.1 - Modèle général d'un agent rationnel	35
Figure 1.2.1 - Grille de STELLA BY STARLIGHT	38
Figure 1.2.2 - Agent jazzman artificiel et son environnement	41
Figure 1.2.3 - Position des modules sur la grille	41
Figure 1.2.4 - Déroulement du morceau (les chorus)	43
Figure 2.2.1 - Fragment de ligne de basse de Ron Carter au début de STELLA BY STARLIGHT	52
Figure 2.3.1 - Grille de STELLA BY STARLIGHT dont la description a été enrichie par la mise en relief des sections et des schémas d'accords	55
Figure 2.4.1 - Cinq patterns mélodiques de Miles Davis sur Dmin7 G7 	57
Figure 2.6.1 - Tâche du bassiste de jazz revisitée : descriptions plus riches	64
Figure 3.1.1 - Exemple de grammaire définissant le rythme d'une mesure de ligne de basse	69
Figure 3.4.1 - Quelques exemples de segments d'improvisation	78
Figure 3.4.2 - Concrétisation des traits en termes de notes	81
Figure 4.1.1 - Disponibilité des données du scénario	92
Figure 4.2.1 - Exemples de changement binaire-ternaire	94
Figure 4.2.2 - Passage de ligne de basse où le module d'exécution ajuste la dernière note des segments	96
Figure 4.2.3 - Passage de ligne de basse sans ajustement de la dernière note des segments	96
Figure 4.3.1 - Exemple d'une segmentation possible de STELLA BY STARLIGHT	100
Figure 4.3.2 - Sélection des PACTs activées	104
Figure 4.3.3 - Illustration graphique de l'assemblage de PACT	107
Figure 4.3.4 - Étapes du raisonnement de notre agent	109
Figure 4.3.5 - Fragment de ligne de basse joués par Ron Carter sur un II-V mineur aux mesures 29-30 de STELLA BY STARLIGHT	112
Figure 4.3.6 - STELLA BY STARLIGHT : Utilisant uniquement des règles de production pour calculer les notes	114

Figure 4.3.7 - Fragment de ligne de basse joués par Ron Carter sur un II-V mineur aux mesures 25-26 de STELLA BY STARLIGHT	115
Figure 4.3.8 - Fragment de ligne de basse joué par Ron Carter sur STELLA BY STARLIGHT aux mesures 67-68	117
Figure 4.3.9 - Le cycle du raisonnement à partir de cas.....	118
Figure 4.3.10 - Activation et assemblage des PACTs en tant que raffinement de la requête de la base de cas	121
Figure 5.1.1 - Descriptions standards représentées par un graphe conceptuel	129
Figure 5.2.1 - Classes de PACTs	133
Figure 5.2.2 - Réification des attributs	134
Figure 5.2.3 - Hiérarchie de descriptions standards (HDS) des PACTs standards de basse.....	135
Figure 5.2.4 - Attributs et aspects de la BassStandardPact.....	136
Figure 5.4.1 - Attributs et aspects de la BassTBDPact.....	152
Figure 5.5.1 - Attributs et aspects de la BassAccomplishedPact	154
Figure 5.5.2 - Attributs et aspects de LocalContext	155
Figure 5.5.3 - Attributs et aspects de ChordChunk.....	155
Figure 5.5.4 - Fragments possibles de ligne de basse à être acquis.....	157
Figure 5.5.5 - Attributs et aspects de Case	158
Figure 5.5.6 - Cas obtenu à partir d'une ligne de basse de Ron Carter sur STELLA BY STARLIGHT	160
Figure 5.5.7 - Passage de la PACT standard vers le cas cible.....	161
Figure 6.2.1 - Notre modèle d'agent jazzman	166
Figure 6.3.1 - STELLA BY STARLIGHT : identification des tous les schémas d'un lexique donné	173
Figure 6.3.2 - STELLA BY STARLIGHT : les schémas après résolution des conflits.....	175
Figure 6.3.3 - Segmentation de FEUILLES MORTES : Mémoire Musicale plus complète	178
Figure 6.3.4 - Segmentation de FEUILLES MORTES basée sur les schémas de STELLA BY STARLIGHT	178
Figure 6.4.1 - PACTs ne pouvant être sélectionnées.....	184
Figure 6.6.1 - Exemple d'indexation de cinq cas de disputes	188
Figure 6.6.2 - Organisation de la Mémoire Musicale.....	190
Figure 6.6.3 - Classe Transposition.....	195

Figure 6.6.4 - Premier chorus de FEUILLES MORTES généré à partir de fragments de la ligne de basse jouée par Ron Carter sur STELLA BY STARLIGHT	197
Figure 7.1.1 - Interface de la Mémoire Musicale et d'édition des descripteurs d'un fragment mélodique et de son contexte	202
Figure 7.1.2 - Interface gérant la trace des cas utilisés.....	203
Figure 7.1.3 - Interface principale d'ImPact	204
Figure 7.2.1 - Deux versions de ligne de basse sur le premier chorus de ALL OF ME.....	206
Figure 7.2.2 - Deux versions de ligne de basse sur deux chorus de BLUESETTE.....	208
Figure 7.2.3 - Deux versions de ligne de basse sur deux chorus de GIANT STEPS	210
Figure 7.2.4 - Ligne de basse sur deux chorus de FEUILLES MORTES sans activation et assemblage de PACTs	213
Figure 7.2.5 - Ligne de basse sur deux chorus de FEUILLES MORTES avec activation et assemblage de PACTs	214
Figure 7.2.6 - Deux lignes de basse jouées sur le premier chorus de ALL OF ME en réaction à des événements H "on joue <i>hot</i> " et D "le pianiste joue très consonant"	218
Figure 7.2.7 - Ligne de basse jouée sur le premier chorus d'ALL OF ME en réaction à des événements H "on joue <i>hot</i> ", P "le public siffle" et M "un musicien dit: "n'aie pas peur""	220
Figure 7.2.8 - Nouvelle version de ligne de basse jouée sur FEUILLES MORTES (raccourcissement du raisonnement).....	223
Figure 7.2.9 - Ligne de basse jouée par Band-in-a-box sur les deux premiers chorus de FEUILLES MORTES	227
Figure 7.2.10 - Ligne de basse jouée par NeurSwing sur le premier chorus et partie du deuxième chorus de FEUILLES MORTES	228
Figure 7.2.11 - Ligne de basse jouée sur les deux premiers chorus de FEUILLES MORTES par Tyrone Wheeler	236

LISTE DES TABLEAUX

Tableau B.1 - Boucle principale de notre programme.....	27
Tableau 1.1.1 - Exemples de types d'agents et de leurs caractérisations	36
Tableau 1.2.1 - Caractérisation d'un agent jazz et d'un agent compositeur.....	37
Tableau 2.4.1 - Description d'un solo de Miles Davis par David Baker.....	60
Tableau 3.1.1 - Présentations des programmes	68
Tableau 3.4.1 - Masques de réutilisation des patterns de <code>Band-in-a-box</code>	86
Tableau 4.3.1 - Exemples de PACTs.....	101
Tableau 4.3.2 - Rapports entre l'Humeur et quelques dilemmes du jazz.....	123
Tableau 5.2.1 - Fonction <code>segmenter</code>	138
Tableau 5.2.2 - Fonction <code>pactPréféréeEntre</code>	140
Tableau 5.2.3 - Fonction <code>combinabilité</code>	146
Tableau 5.2.4 - Fonction <code>combiner</code>	146
Tableau 5.2.5 - Fonction <code>combinaisonTotaleEntre</code>	146
Tableau 5.2.6 - Fonction <code>combinaisonPartielleEntre</code>	147
Tableau 5.2.7 - Fonction <code>remplaceSousArbre</code>	148
Tableau 5.2.8 - Fonction <code>ajusterParamètresDeTemps</code>	148
Tableau 5.2.9 - Règle <code>repeatedNotes</code>	149
Tableau 5.2.10 - Règle <code>melodicFragmentHigh</code>	150
Tableau 5.4.1 - Fonction <code>discrétiser</code>	152
Tableau 5.5.1 - Quelques schémas d'accords de STELLA BY STARLIGHT.	156
Tableau 5.5.2 - Fonction <code>createCase</code>	158
Tableau 5.5.3 - Fonction <code>anti-propager</code>	159
Tableau 5.5.4 - Fonction <code>commeUnePactAccomplie</code>	161
Tableau 6.2.1 - Fonction <code>jouer</code> (boucle principale).....	167
Tableau 6.2.2 - Fonction <code>raisonnementEstTrèsEnAvance</code>	168
Tableau 6.2.3 - Fonction <code>derniersÉvénements</code>	168
Tableau 6.2.4 - Fonction <code>calculerBoutDImprovisationPréssé</code>	169
Tableau 6.2.5 - Fonction <code>calculerBoutDImprovisation</code>	170
Tableau 6.3.1 - Fonction <code>segmenterGrilleEnSchémasDAccords</code>	170
Tableau 6.3.2 - Règle <code>deuxCinqMajeur</code>	172
Tableau 6.3.3 - Règle <code>cinqUnMineurContreSixDeuxCinqUn</code>	174
Tableau 6.3.4 - Règle <code>subsume</code>	175

Tableau 6.3.5 - Règle schémaValide	176
Tableau 6.3.6 - règle schémaNonValide	177
Tableau 6.4.1 - Règle policeArrive	179
Tableau 6.4.2 - Règle publicApplaudit.....	180
Tableau 6.4.3 - Fonction activerDesNouvellesPacts	180
Tableau 6.4.4 - Règle rythmePlusDenseAuDebutDImprovisation	181
Tableau 6.4.5 - Règle presqueRépétitionSelonLaRépétitionDuSchéma	181
Tableau 6.4.6 - Règle intensitéBatteur	182
Tableau 6.4.7 - Règle ambianceChaude	182
Tableau 6.4.8 - Fonction sélectionnerPacts	184
Tableau 6.5.1 - Fonction assemblerPacts	185
Tableau 6.5.2 - Règle pactJouable.....	185
Tableau 6.5.3 - Règle courvertureDirecteTotaleMutuelle.....	186
Tableau 6.5.4 - Règle combinabilitéPartielle	186
Tableau 6.5.5 - Règle dernièrePact.....	187
Tableau 6.6.1 - L'algorithme général de la récupération et réutilisation d'un cas	187
Tableau 6.6.2 - Fonction calculerLesNotes	191
Tableau 6.6.3 - Fonction casCibleAPartirDe.....	191
Tableau 6.6.4 - Fonction adaptabilité	194
Tableau 6.6.5 - Fonction transformationsDAdaptation	194
Tableau 6.6.6 - Fonction adapter	195
Tableau 6.6.7 - Fonction transférerInformations.....	196
Tableau 6.6.8 - Adaptabilité et similarité des cas 10 et 51 vis-à-vis du schéma d'accord (Mineur8) des mesures 15 et 16 de FEUILLES MORTES	197
Tableau 7.2.1 - Tableau comparatif des lignes de basse de Band-in-a-box, ImPact et NeurSwing	226

LISTE DES FORMULES

Formule 1.2.1 - ERE.....	42
Formule 1.2.2 - ERE*	42
Formule 1.2.3 - Paramètres d'une note	43
Formule 1.2.4 - Segment d'Improvisation	43
Formule 1.2.5 - Improvisation.....	43
Formule 1.2.6 - Fonction ProchainSegment.....	44
Formule 4.1.1 - EvenBase	91
Formule 4.1.2 - EvenMotif.....	91
Formule 4.1.3 - EvenGlobal	91
Formule 4.1.4 - EvenPublic.....	92
Formule 4.3.1 - Humeur	122
Formule 5.2.1 - spécialisé(p,a)	137
Formule 5.2.2 - A*	137
Formule 5.2.3 - S*	137
Formule 5.2.4 - activée(p)	137
Formule 5.2.5 - jouable(p)	138
Formule 5.2.6 - strictementPlusJouable(p ₁ ,p ₂)	139
Formule 5.2.7 - approximativementPlusJouable(p ₁ ,p ₂)	139
Formule 5.2.8 - plusJouable(p ₁ ,p ₂)	139
Formule 5.2.9 - préféréeParRapportAuTemps (p ₁ ,p ₂).....	140
Formule 5.2.10 - conflitOuRedondanceDirecte(p ₁ ,p ₂ ,a)	141
Formule 5.2.11 - conflitOuRedondanceHiérarchique(p ₁ ,p ₂ ,a)	141
Formule 5.2.12 - couvertureDirecteTotale(p ₁ ,p ₂)	141
Formule 5.2.13 - couvertureHiérarchiqueTotale (p ₁ ,p ₂)	142
Formule 5.2.14 - couvertureDirecteNulle(p ₁ ,p ₂).....	142
Formule 5.2.15 - couvertureDirecteNullePlus(p ₁ ,p ₂)	143
Formule 5.2.16 - pasDeConflictOrRedondanceHiérarchique(p ₁ ,p ₂).....	145
Formule 5.5.1 - simil(C,S).....	162
Formule 5.5.2 - similarité(c ₁ ,c ₂).....	162
Formule 6.6.1 - réutilisabilité(cc,cs)	193

LISTE DES MORCEAUX CITES

All of Me	- Seymour Simons & Gerald Marks
Bluesete	- Toots Thielemans
All the Things You Are	- Jerome Kern & Oscar Hammerstein II
Body and Soul	- E. Heyman, R. Sour, F. Eyton & J. Green
Feuilles Mortes (<i>Autumn Leaves</i>)	- Joseph Kosma
Giant Steps	- John Coltrane
It's You or no One	- Cahn & Styne
Stella By Starlight	- N. Washington & V. Young
The Song is You	- Jerome Kern & Oscar Hammerstein II
What is this Thing Called Love	- Cole Porter



AVANT-PROPOS: UNE VERSION DU TEST DE TURING POUR LE JAZZ?

Me voilà un vendredi soir au "Petit Journal Montparnasse" au 13, rue du Commandant Mouchotte, cet endroit connu par les musiciens et amateurs de jazz à Paris. Il y avait, comme c'est souvent le cas, de la fumée et une très bonne ambiance : des belles filles bavardant gaiement, des jeunes découvrant le jazz, des vieux semblant avoir connu Sidney Bechet ou Django Reinhardt. Alors, les musiciens arrivent et montent sur scène. Certains visiblement fatigués, d'autres avec un sourire timide, comme cachant des choses qui ne sauraient être révélés qu'à travers leurs instruments. Ils se sont mis à jouer "Now's the Time" de Charlie Parker et puis une autre que je ne connaissais pas. Le plaisir fou qu'ils éprouvaient en jouant était flagrant. C'était aussi beau à regarder qu'à écouter. Il me paraissait à ce moment-là que la lourdeur des nombreuses heures de travail auxquelles chacun doit se soumettre n'existait plus. Petit à petit le public se mêlait à cette ambiance. C'était vraiment aussi beau à regarder qu'à écouter. Pendant la pause, en tant que chercheur en intelligence artificielle (IA), je n'ai pas pu m'empêcher de me poser un certain nombre de questions. Simuler l'activité d'un de ces musiciens par une machine est-il pensable ? Comment l'interaction entre les musiciens (et entre les musiciens et le public) s'établit-elle ? Qu'est-ce qui fait qu'un bon musicien ne joue jamais de la même façon le même morceau et qu'à chaque fois ce soit bien fait ? Quelles connaissances donner aux machines ? Est-ce que tout se ramène à avoir des connaissances, de l'expertise ?

Alors, j'ai pensé à Turing, qui pour éviter, en quelque sorte, de donner une définition de l'intelligence, a proposé son test limite (Turing, 1950). Pour ne pas définir non plus ce qu'est la création musicale, on pourrait imaginer une version jazz du test de Turing qui consisterait à isoler un des musiciens derrière un paravent et à le remplacer par une machine d'un morceau à l'autre. Si les autres musiciens ne pouvaient pas faire la distinction, alors la machine aurait réussi le test. Évidemment, cette version jazz du test de Turing est problématique. D'une part la communication visuelle est coupée (tout comme dans le test de Turing originel, d'ailleurs). D'autre part, le timbre d'un synthétiseur serait peut-être immédiatement reconnu, malgré les avancés des techniques de "modélisation physique" des instruments (Florens & Cadoz, 1991). Mais, actuellement on est encore loin de pouvoir produire en temps réel des timbres identiques à certains instruments traditionnels, surtout en ce qui concerne les différents modes de jeux. De toute façon, ce test de Turing en lui-même n'est pas important. En revanche, savoir comment il faut programmer une machine pour qu'elle improvise de façon satisfaisante, et même savoir ce qui est satisfaisant, soulève des problématiques importantes pour la recherche en IA.

INTRODUCTION

A - RECHERCHES DANS LES DOMAINES ARTISTIQUES : LES ENJEUX POUR L'IA

L'IA est-elle une science ? Si c'est le cas, quelle est sa méthodologie ? Peut-on parler d'expérimentation en IA comme en science ? Sinon quelle est son statut ? Comment évaluer les programmes d'IA ? Les programmes sont-ils des théories ? En quoi les programmes sont-ils différents de ceux des autres branches de l'informatique ?

a) La démarche d'IA en tant que science expérimentale

Certainement poussées par une maturité acquise dans ces 40 dernières années de recherche, mais aussi par le besoin de répondre à un certain malaise méthodologique (Bundy, 1990), ces anciennes questions ont récemment pris un nouvel élan aux États-Unis. La réflexion sur les fondements et la méthodologie de l'IA est au cœur de débats dans les conférences et ateliers de ces quatre dernières années¹. Comme on peut s'y attendre, rien n'est pour l'instant tranché, cependant de bonnes pistes ont été esquissées, ou plutôt, de vieilles pistes ont été renforcées. C'est le cas notamment de l'idée que l'IA est une science expérimentale.

Parmi les chercheurs qui soutiennent cet argument, Simon et Newell sont ceux qui incarnent cette idée le plus parfaitement. En 1976, à l'occasion de l'*ACM Turing Award Lecture*, ces chercheurs ont fait une présentation intitulée *Computer Science as Empirical Inquiry: Symbols and Search* (Newell & Simon, 1976), où ils défendent l'idée que l'informatique, et en particulier l'IA, sont des disciplines fondamentalement empiriques. Comme exemple, ils discutent la démarche empirique entreprise pour l'étude de deux hypothèses centrales en IA : l'hypothèse des systèmes symboliques physiques et l'hypothèse de la recherche heuristique. Selon eux, si l'IA et l'informatique en général n'ont pas été appelées sciences expérimentales dès leur naissance, c'est parce que, tout comme l'astronomie, l'économie et la géologie, leurs

¹ En témoignent les présentations de Pat Hayes, John McCarthy, Aaron Sloman et Marvin Minsky dans le dernier IJCAI à Montréal. Autre exemple : les actes de l'atelier spécialisé sur le sujet qui a réuni entre autres Schank, Dennet, Winograd, Dreyfus, Chandrasekaran, etc. (Partridge & Wilks, 1990).

expérimentations sortent du cadre stéréotypé des méthodes expérimentales. En IA, "chaque programme informatique construit est une expérimentation"... "on construit des programmes comme une manière de découvrir des nouveaux phénomènes ou d'analyser ceux que l'on connaît déjà" [p. 114]. Ainsi, la démarche de l'IA serait assimilée à un cycle de *génération et test* à long terme où l'on essaie de généraliser des théories à partir de programmes construits pour résoudre des problèmes concrets dans différents domaines, le but de la recherche en IA étant la compréhension des conditions nécessaires pour que les machines exhibent un comportement intelligent.

Cette démarche empirique de compréhension par construction et observation est d'ailleurs typique de ce que Simon (Simon, 1981) appelle les *sciences de l'artificiel*. Les sciences naturelles sont caractérisées par la description et l'analyse, pour montrer que la complexité de la nature n'est qu'un masque de la simplicité. Les sciences de l'artificiel, à leur tour, sont caractérisées par la prescription et la synthèse, pour comprendre comment conformer les objets (naturels comme pour la médecine ou artificiels comme pour l'économie) aux intentions et objectifs de l'homme (Simon, 1981) [pp. 3-8].

La démarche expérimentale de l'IA est possible car les programmes, n'étant pas des boîtes noires, peuvent nous renseigner sur le rapport entre leur structure et leur comportement. C'est dans cette perspective que l'on pourrait dire que l'IA est une science comportementale : une science qui étudie le comportement des programmes censés accomplir intelligemment des tâches données. Au fur et à mesure que de nouvelles tâches sont abordées, quelques principes généraux sont dégagés pour ensuite être réutilisés dans le développement d'autres systèmes ou dans l'amélioration des systèmes existants. L'histoire de l'IA témoigne de l'influence que des réalisations particulières ont eue dans la définition des théories ou méthodes plus générales. Le Logic Theorist peut être vu comme la racine du GPS; MYCIN et DENDRAL ont inspiré les systèmes experts; les méthodes de diagnostic et de sélection de catalogue ont été généralisées par la classification heuristique; AM et BACON sont des références clés de la découverte scientifique (*machine discovery*); etc.

Dans ce même sens, Partridge et Wilks (Partridge, 1990) prétendent que la différence méthodologique principale entre l'IA et le génie logiciel est que la première est de type RUDE (*Run, Understand, Debug and Edit*) tandis que la deuxième est plutôt SPIV (*Specify, Prove, Implement and Verify*), voire une version plus simplifiée, le SAT

(*Specification And Testability of program behavior*). La supposition de base de la méthodologie SPIV est que l'on peut, ou que l'on devrait pouvoir, spécifier les applications à l'avance. Or, on sait que l'explicitation et la représentation de connaissances d'un expert humain ne peuvent se faire que par des allers et retours successifs à travers lesquels le modèle informatique se raffine et évolue². Ceci est dû à la fois à la nature heuristique de l'expertise (car elle est conséquence d'une *rationalité limitée*), aux problèmes d'accès aux connaissances par les cognititiens et enfin aux limitations des langages informatiques de représentation de connaissances. A cet égard, les efforts menés par des méthodologies d'acquisition de connaissances comme KADS sont très révélateurs. D'une part ils montrent qu'il est possible de trouver des abstractions des systèmes à base de connaissances en forme de bibliothèques de méthodes de résolution de problèmes ou d'ontologies du domaine (Wielinga, Schreiber & Breuker, 1992). D'autre part, l'expérience acquise dans l'utilisation de ces méthodologies montre l'insuffisance de ces généralisations pour une tâche particulière (Breuker, 1993).

La difficulté de définir de façon consensuelle les buts et la méthodologie de l'IA demeure, sans doute à cause de la cohabitation de différents projets en IA. Il y a l'IA dite appliquée ou *IA technologique*, où les techniques courantes d'IA sont utilisées pour produire des logiciels pour l'industrie. Il y a ensuite le projet appelé *sciences cognitives*, dont le but est de comprendre l'intelligence humaine à l'aide des outils d'IA. Il y a enfin l'IA appelée *IA fondamentale*, qui est celle qui étudie "le comportement des programmes" telle que préconisée par Newell et Simon dans l'article cité plus haut. Ce n'est pas notre intention de discuter ici ni les origines de ces différents projets ni leurs frontières précises³, d'autant plus que, fréquemment, les chercheurs travaillent à la fois dans deux projets différents. Ce que nous avons voulu souligner dans cette section, c'est la vocation empirique de l'IA, vocation qui donne au programme un rôle capital. D'où l'importance de poursuivre, en parallèle au travail d'abstraction des principes généraux sous-jacents aux programmes, le développement de systèmes qui s'attaquent à des

²Ce point de vue est partagé par Le Roux dont la thèse centrale est que le processus d'acquisition se fait de façon constructive, comme dans une spirale (Le Roux, 1994)

³Voir à ce propos les ouvrages généraux comme (Laurière, 1987), (Ganascia, 1990b), (Rich & Knight, 1983) et (Russel & Norvig, 1995).

tâches de plus en plus diverses et proches de la complexité du monde réel. Chaque domaine d'application fonctionne comme une source de problèmes qui permettent la consolidation, la modification ou encore la remise en cause des techniques d'IA courantes. C'est dans ce cadre que nous essayerons de montrer dans la prochaine section l'intérêt de notre recherche sur la création musicale, un domaine très riche qui pose des problèmes difficiles et qui n'a pas été suffisamment étudié en IA jusqu'à présent.

b) Motivations pour l'IA de la modélisation des activités artistiques

Ce n'est pas un hasard si l'être humain s'appelle lui-même *Homo Sapiens*. Ses capacités intellectuelles sont la clé de l'évolution de l'espèce. L'éventail d'activités quotidiennes faisant appel à ce que l'on appelle intelligence est innombrable, même si l'on se restreint aux tâches pouvant être assimilées à la résolution de problèmes. Cependant, faisant un survol sur la recherche développée en IA dans ces 30 ou 40 dernières années, on voit que cet éventail est déséquilibré, certaines activités ou tâches ayant été nettement moins étudiées que d'autres. Parmi les domaines les plus étudiés, on identifie les tâches formelles comme les mathématiques, les jeux, l'optimisation ou encore les tâches d'analyse comme la classification, l'évaluation, la prédiction ou le *monitoring* (McDermott, 1989; Wielinga, Schreiber & Breuker, 1992). Or, selon la démarche empirique de l'IA, il serait souhaitable qu'un effort soit fait pour élargir les domaines d'application à d'autres activités humaines. Parmi ces activités, celles de conception, et en particulier de conception non routinières, dont les artistiques, mériteraient une étude plus approfondie. En effet, puisque l'art est un domaine difficile et peu exploré, la recherche en IA peut bénéficier d'une série d'interrogations, dont voici quelques-unes :

- Comment construire des programmes capables de créer des œuvres d'art ou d'avoir une appréciation esthétique sur des œuvres d'art données ?
- Quelles sont les faiblesses, les points forts et la généralité des techniques et des paradigmes courants si on les emploie à la construction de ces programmes artistiques ?
- Dans quelle mesure les techniques particulières employées dans ces programmes seraient-elles extensibles à des domaines non artistiques ?
- Qu'est-ce qui fait la particularité de l'art par rapport à la nature des connaissances et quels sont les problèmes posés par leur représentation et manipulation ?

- Que doivent être les principes généraux des systèmes travaillant dans les diverses formes artistiques (la peinture, la littérature, la musique, etc.) ?

On voit que ces questions s'insèrent dans la démarche expérimentale de l'IA et qu'en bonne partie elles pourraient être traduites dans n'importe quel domaine complexe et non suffisamment étudié en IA. Cela dit, il convient de souligner que la recherche dans des domaines artistiques a une motivation principale pour l'IA, et cette motivation tient dans le mot "créativité". Une compréhension plus profonde de la simulation des "processus créatifs" sur une machine serait extrêmement importante pour la construction de systèmes intelligents. Bien entendu, le problème est que l'on connaît très peu de choses sur la créativité et on évite même d'en parler, vu l'atmosphère de mystère et l'aura de surnaturel autour de ce mot. En effet, on ne sait pas si la créativité est un processus cognitif comme les autres, et moins encore quels sont les critères pour dire qu'une idée ou un comportement est créatif.

Malgré ces difficultés, plusieurs efforts ont été entrepris pour essayer de montrer qu'il n'y a pas de raisons pour qu'une machine ne puisse être "créative" et de mieux comprendre comment effectivement construire de telles machines. Quelques systèmes ont été développés dans le domaine artistique pour le dessin abstrait (Cohen, 1981)); pour l'écriture de romans (Rumelhart, 1975), de poèmes (Masterman, 1971) ou d'histoires (Davey, 1978; Mehegan, 1984); et, aussi, pour la composition musicale en plusieurs styles comme la musique classique romantique (Cope, 1991; Ebcioğlu, 1992) et la musique électronique contemporaine (Hiller & Leonard, 1959). En découverte scientifique plusieurs systèmes ont été également conçus (Langley et al., 1987; Lenat & Brown, 1984). D'autres chercheurs ont mené des expérimentations avec différentes approches comme l'analogie (Hofstadter, 1986; Mitchell, 1993), les processus aléatoires (Johnson-Laird, 1992), le raisonnement à partir de cas (Schank, 1982; Schank, 1990) et la résolution de problèmes (Newell, Shaw & Simon, 1962; Simon, 1995). Quelques réflexions d'ordre plus général sur ces différentes approches ont été proposées par Boden (Boden, 1992), avec la "théorie des espaces conceptuels", par Rowe et Partridge avec le "modèle de la mémoire émergente" (Rowe & Partridge, 1993) et dans un atelier spécialisé sur le thème IA et créativité (AAAI, 1993).

La problématique de la créativité est un des thèmes de recherche de l'équipe ACASA du LAFORIA, au sein de laquelle ce travail de thèse a été développé. Dans

l'équipe, plusieurs expérimentations ont été faites sur ce thème, en particulier dans les domaines de la découverte scientifique en médecine (Corruble, 1996; Corruble & Ganascia, 1994a; Corruble & Ganascia, 1994b) et de la découverte en base de données (Zucker et al., 1994). Des réflexions approfondies ont été aussi menées sur le modèle de créativité proposé par Schank (Ganascia, 1990a). Selon Schank, la créativité est liée à la capacité de proposer des explications (ou solutions) à des situations (ou problèmes) auxquelles une personne ou une machine n'a jamais été confrontée auparavant. Schank considère que ces nouvelles explications (ou solutions) peuvent être générées à partir de la réutilisation, avec des adaptations, des expériences vécues dans le passé.

Bien que la créativité demeure un sujet très controversé, nous pensons qu'elle est le principal atout de la recherche sur l'art par rapport aux autres domaines peu explorés en IA jusqu'à l'heure actuelle.

c) Quelques difficultés posées par les activités artistiques

Avant d'énumérer quelques difficultés posées par la construction de programmes générateurs d'œuvres d'art, il importe d'examiner deux idées de caractère non scientifique qui sont couramment acceptées. En premier lieu, l'idée que tout ce qui touche à la créativité échappe par définition à une formalisation. Cette objection n'a pas lieu d'être tant que l'on n'essaye pas de la formaliser. En deuxième lieu, on trouve la crainte qu'une explication scientifique des processus de création artistique pourrait dégrader la valeur de l'art, le mythe de l'inspiration, en somme détruire notre émerveillement. Sans faire appel à un argument scientifique, nous pouvons répondre à cette critique simplement en évoquant l'histoire, qui nous montre que la réduction du réel à une formulation mathématique ne signifie pas la perte de la curiosité et l'enchantement par le réel. C'est notamment le cas de la formalisation des lois de la nature par la physique et la biologie.

Du point de vue de l'histoire de l'IA, on s'aperçoit que les premiers systèmes sont appliqués à des domaines non artistiques comme la preuve automatique de théorèmes, le diagnostic médicale ou encore la traduction automatique. Les résultats

de ces recherches ont constitué un savoir-faire qui a fini par influencer les types de problèmes considérés par la suite⁴.

C'est surtout du point de vue de la construction même de systèmes intelligents que l'art perd du terrain en tant que domaine d'investigation. En plus des complications normales de la modélisation de l'expertise, il faut ajouter le fait que les connaissances mises en jeu dans la production et la critique d'œuvres d'art sont encore plus subjectives, ce qui rend plus délicate l'acquisition de connaissances et la mise au point du système. Voyons quelques-uns de ces problèmes de plus près.

Si on assimile la création d'une œuvre d'art à la résolution d'un problème, le problème est alors sous-contraint ou mal posé (*ill structured*) (Simon, 1973). Ainsi, plusieurs peintures peuvent satisfaire l'ensemble de contraintes de style et d'esthétique d'une époque; plusieurs solos peuvent être improvisés pour un morceau donné; etc. Dans tous les problèmes sous-contraints, la difficulté est de trouver les règles de préférences capables de résoudre les conflits parmi les nombreux choix. En musique tonale, par exemple, les règles d'harmonie de la plupart des styles sont très bien définies et cataloguées (Baudoin, 1990; Hindemith, 1968) En revanche, même les manuels d'harmonie les plus rigoureux affirment que le choix final passe par l'oreille (Schoenberg, 1943) [p. 3], ce qui invoque donc tout le vécu de l'individu, toute son expérience. L'accès aux connaissances impliquées dans le jugement esthétique de n'importe quelle forme d'art n'est donc pas évident. Il faut ajouter à cela que la tâche d'explicitation de connaissances est encore plus compliquée s'agissant d'activités (artistiques ou non) qui se déroulent en direct. Pour ces activités beaucoup d'actions sont "câblées", car que les contraintes de temps interdisent un raisonnement plus approfondi.

Le fait que les artistes sont loin d'être unanimes sur la façon de produire et d'évaluer une œuvre d'art rend l'évaluation du système plus laborieuse. En fait, plus on s'éloigne des applications mathématiques, plus on a de problèmes à évaluer les résultats d'un programme d'IA. C'est le cas par exemple des applications en traitement de

⁴Voir à ce propos la critique de Jean-Gabriel Ganascia à l'inflexion du projet initial de l'IA par la résolution de problèmes (Ganascia, 1990b) [pp. 55 à 83]

langage naturel. Des erreurs peuvent être détectées mais l'évaluation globale n'est pas simple. En ce qui concerne l'art, le moyen de minimiser ce problème est de se restreindre à des styles très bien connus, où l'évaluation est la plus consensuelle possible. Mais ceci implique un autre embarras : on aura du mal à construire un système qui crée du radicalement nouveau. Si l'on n'aime pas quelque chose on ne saura pas si c'est la faute du système ou bien si c'est la notre, à cause de notre difficulté à accepter les nouveaux concepts artistiques que le système propose. L'acceptation des nouvelles idées artistiques est toujours problématique, il suffit de penser aux nombreux artistes qui sont morts sans voir la reconnaissance de leurs œuvres.

Examinons de plus près la nature du raisonnement artistique au regard des outils de représentation de connaissances dont nous disposons en IA. Par exemple, la notion de vérité chère à la logique a un sens controversé en art. Quand on dit d'une œuvre d'art qu'elle est vraie, c'est une affaire de droit et non d'appréciation esthétique. Si on pense aux règles musicales (d'harmonie, par exemple), une partie de la créativité artistique tient justement à la désobéissance (Sloboda, 1985) [p. 55]. Savoir quand et à quelles règles désobéir est un vrai problème. S'il semble y avoir, du point de vue d'un observateur externe, un côté aléatoire dans la création artistique (Johnson-Laird, 1987), engendrer du "nouveau" n'est pas suffisant car les œuvres d'art doivent respecter aussi des contraintes esthétiques diverses (Boden, 1992; Rowe & Partridge, 1993). Si ce n'était pas le cas, les machines seraient aisément "créatives" car, mieux que quiconque, elles peuvent s'affranchir des *a priori* culturels pour explorer aléatoirement des espaces conceptuels. En effet, cette tension entre innover et répéter est certainement un des problèmes les plus importants et difficiles de la création artistique.

Il y a aussi les difficultés qui tiennent à la détermination des buts, ces derniers étant entendus au sens de la résolution de problèmes. Comme dans tout problème mal posé (Simon, 1973), l'artiste ne connaît que vaguement le profil qu'aura son œuvre à l'avance. Ce n'est qu'en cours de route qu'il comprend mieux ce qu'il veut créer. Autrement dit, l'artiste pose et résout le problème continuellement, ce qui implique des changements et "raffinements" permanents des buts. Ceci est d'autant plus vrai lorsqu'il s'agit d'une représentation en direct comme la représentation théâtrale ou l'improvisation en jazz, car les artistes, ne pouvant revenir en arrière dans le temps, sont obligés de s'adapter "au vol" aux événements imprévus, à leurs propres fautes, etc.

Mais ce problème des buts est vrai aussi pour le tâches "en différé". Kugel, en argumentant que certaines pensées musicales ne sont pas calculables par une machine de Turing, rappelle une histoire sur Beethoven (Kugel, 1992), dans laquelle ce dernier n'aurait fourni la version finale d'une sonate pour violon que le jour où elle devait être jouée au public, le 23 décembre 1806. Les musicologues disent que, contrairement à Mozart, Beethoven aimait bien ruminer ses compositions avant de les considérer finies (Roozendaal, 1990). Selon Kugel, on ne sait pas si Beethoven aurait encore apporté des modifications à la sonate si le jour de la présentation avait été repoussé.

B - CADRE DE LA THESE

Il ne serait pas raisonnable dans un projet de doctorat de vouloir traiter tous les domaines artistiques ni tout l'ensemble de techniques d'IA disponibles. Au contraire, restant fidèle à la vocation expérimentale de l'IA, il est souhaitable de choisir un problème bien précis et suffisamment complexe au regard du monde réel pour ensuite tester l'adéquation d'une approche de modélisation particulière ou d'un ensemble de techniques de construction de systèmes intelligents. Dans cette perspective, nous précisons, dans cette section, le problème concret auquel notre travail de thèse s'intéresse : la simulation de l'activité d'un bassiste de jazz jouant en direct. Nous examinons aussi le paradigme dans lequel notre travail s'insère, en l'occurrence la *résolution de problèmes* dans le sens de l'IA symbolique classique, et nous énonçons plus précisément les objectifs que nous nous fixons dans le cadre de ce travail de thèse.

a) De l'art au bassiste de jazz

Pourquoi la musique ? D'abord, nous avons acquis une certaine expertise dans le domaine musical aussi bien à titre personnel (Arcela & Ramalho, 1991) qu'au sein du LAFORIA à travers les travaux de Xavier Rodet, Francis Rousseaux, François Pachet et Boris Doval. D'ailleurs, la musique a traditionnellement été un champ d'investigation à l'Institut Blaise Pascal comme en témoignent les recherches de Pierre Cointe, Philippe

Gautron et Marc Chemillier et, indirectement, Patrick Greussay⁵. Le savoir-faire technique et la réflexion réunis au long des années par le biais de ces travaux nous ont été en effet très utiles.

Ensuite, la musique est un domaine dont les principes et concepts de base sont bien définis, au moins en ce qui concerne la musique tonale. L'existence même de ce que l'on appelle *théorie musicale* est révélatrice du degré de formalisation de la musique. Parmi les arts, la musique est la seule à proposer des cours de "théorie" (harmonie, solfège, rythme, etc.) aux élèves avant tout apprentissage d'instrument. En effet, au-delà des exemples allégoriques comme le jeu de dés musical de Mozart⁶, il y a un rapprochement entre la musique et les mathématiques. Néanmoins, bien qu'une partie des concepts musicaux se prête à la modélisation mathématique (Chemillier, 1990; Riotte, 1990), on n'est pas encore en mesure de construire des programmes qui composent de la musique ou qui la perçoivent dans toute sa complexité. A ce propos Laske (Laske, 1989) fait la distinction entre la "théorie musicale" et la "théorie de la composition" : la première concerne la musique existante, elle dit quoi faire, alors que la seconde concerne les musiques possibles, elle est censée dire comment faire. Connaître la première est certainement nécessaire mais pas suffisante pour maîtriser la seconde. En somme, la tâche de formalisation des concepts musicaux de base étant plus simple, on peut se concentrer plus rapidement sur le cœur du problème de la création et de la compréhension musicales.

Mais pourquoi la musique tonale ? Avant de justifier l'option pour le jazz il est important d'expliquer pourquoi choisir la musique tonale. En effet, le jazz a hérité de plusieurs éléments de la musique tonale occidentale. Quoique certains styles comme le *free* et même une partie du *bop*, aillent souvent au-delà du tonal, l'essentiel de l'harmonie et de la forme reste occidentale (Hodeir, 1981) [pp. 131-47], (Mehegan, 1984) [pp. 6-8]. Expliquer le choix pour la musique tonale est d'autant plus important que la

⁵ Le dossier "IA et Musique" du 23^e Bulletin de l'AFIA (Pachet & Assayag, 1995), [p. 22], refait le parcours historique des travaux en musique à l'IBP, ainsi qu'il présente ce qui se fait récemment en musique au LAFORIA.

⁶Dans ce jeu on compose des nouvelles pièces musicales en recombinaison des phrases mélodiques d'une pièce donnée. Ces phrases sont choisies à l'aide d'une matrice qui les indexe selon la valeur des dés et leur position dans le morceau (Schwanauer & Levitt, 1993) [pp. 533-8].

grande majorité de travaux en informatique musicale sur des systèmes d'aide à la composition ou de composition automatique est dédiée à la musique électronique contemporaine⁷. Particulièrement en France, la musique électronique contemporaine a été très développée à commencer par les travaux de Schaeffer, Barbaud, Xenakis et Boulez pour n'en citer que quelques-uns. La musique tonale est à notre avis un champ privilégié pour l'IA car, tout en gardant une complexité considérable, elle s'est enracinée dans la culture occidentale, ce qui a permis qu'un certain consensus se forme sur les concepts de bases, les méthodes de composition et le jugement esthétique. Ces caractéristiques facilitent aussi bien la validation du programme censé générer tel genre de musique que la construction même de ce programme (acquisition et représentation de connaissances). En fait, en musique tonale on sait précisément ce que veut dire une note, un accord, une gamme, un intervalle, une cadence, une modulation, etc. On sait aussi émettre des jugements fins sur un contrepoint, une harmonisation etc. En musique électronique contemporaine la validation et la construction du programme sont nettement plus compliquées, voire impossibles, à cause des idiosyncrasies de chaque compositeur. Le concept d'accord, par exemple, est très variable (Courtot, 1992).

Pourquoi le jazz? Dans le cadre de la musique tonale nous aurions pu opter pour la musique classique à travers de nombreux schémas de composition comme l'harmonisation de chorales, les contrepoints à plusieurs voix et espèces, les fugues, etc. (Baggi, 1974; Ebcioğlu, 1992; Rothgeb, 1993; Widmer, 1992). La première raison pour laquelle nous nous sommes penchés sur l'improvisation et l'accompagnement en jazz, est que le jazz est une musique vivante et encore en évolution. Les diverses formes de composition de la musique classique, bien qu'étant des chefs-d'œuvres de la musique occidentale, sont actuellement pratiquées essentiellement à titre pédagogique dans les conservatoires. L'opportunité de rencontrer des experts dont le souci est d'innover et de raffiner la pratique et les concepts autour de l'improvisation et l'accompagnement en jazz nous semblait intéressante. Par ailleurs, compte tenu des nombreuses études réalisées depuis des siècles sur la musique classique, les connaissances semblent avoir un caractère plutôt figé. Cela ne signifie certes pas que l'on sait écrire des programmes capables de composer des symphonies, des fugues ou des concertos. Mais, nous nous

⁷ En France on l'appelle plutôt "musique savante" ou "musique contemporaine".

intéressions en particulier à la composante “intuitive” et “empirique” liée au rôle central de l’écoute et de la pratique dans le jazz. Par exemple, la manipulation et la perception des rythmes, d’influence de l’Afrique noire, restent à être formalisée.

Mais quel jazz ? Nous nous sommes intéressés surtout au *be-bop*, car il est toujours un style de référence jazz, pratiqué jusqu’à aujourd’hui. En outre, les musiciens généralement débutent leurs études par le be-bop en jouant des standards.

Enfin, pourquoi le bassiste ? Un groupe de base de jazz se compose d’un ou plusieurs solistes et d’une section rythmique rassemblant un pianiste, un bassiste et un batteur (ces derniers sont aussi appelés accompagnateurs). Chaque jazzman, soliste ou accompagnateur, joue un rôle capital dans le groupe, et la perfection est très difficile à atteindre pour tous sans distinction. Néanmoins, comme le soliste a une plus grande liberté, la modélisation informatique de son activité pose plus de problèmes. C’est pour cette raison que, tout au début de la thèse, nous nous sommes intéressés à modéliser les connaissances de trois musiciens de la section rythmique pour créer un système capable d’accompagner un soliste humain (Ramalho & Ganascia, 1994a). Ce n’est que plus tard que nous nous sommes rendus compte de la difficulté de ce projet. Nous nous sommes alors restreints à l’instrumentiste dont nous avons le plus étudiés les connaissances jusqu’à présent, en l’occurrence le bassiste. Notre choix pour les tâches d’accompagnement en jazz a été renforcé par l’existence, au sein du LAFORIA, de travaux développés par François Pachet sur la modélisation des connaissances utilisées par les jazzmen. Pachet avait conçu un système censé simuler l’activité d’un pianiste et d’un bassiste accompagnant un soliste en temps réel (Pachet, 1987; Pachet, 1990). Les réflexions menées autour de ce système ont été en effet un point de départ très important dans notre travail de thèse.

b) Approche, hypothèses de travail et objectifs

Il nous a fallu définir précisément le problème artistique concret qui servira de base à notre travail. De même, nous avons besoin de mieux expliciter l’approche générale que nous adopterons. Il était en effet opportun, compte tenu de la complexité du problème que nous abordons, que nous choissions un angle d’attaque qui nous permette de mieux poser le problème, tout en évitant de nous restreindre d’emblée à une technique spécifique. De fait, la richesse, et aussi la difficulté, d’être guidé par un problème

concret est justement de pouvoir aller au-delà des frontières de telle ou telle technique, de pouvoir les assembler et les employer autrement.

Nous nous sommes donnés comme cap ce qu'il y a de plus traditionnel : la résolution de problèmes dans les *systèmes symboliques et physiques* ou simplement l'IA symbolique. Un système symbolique et physique est défini par Newell et Simon (Newell, 1980; Newell & Simon, 1976) comme étant une machine qui produit un ensemble de structures de symboles (expressions) qui évolue dans le temps. À travers la *désignation*, un tel système peut affecter un objet à une expression et à travers *l'interprétation*, le système peut identifier et exécuter une procédure associée à une expression. Les expressions peuvent être modifiées, effacées, créés et reproduites par l'application des procédures. La conséquence importante de l'introduction de tels systèmes est la fameuse *hypothèse des systèmes symboliques et physiques* :

The physical symbol system has the necessary and sufficient means for general intelligent action (Newell & Simon, 1976).

Le deuxième volet de cette hypothèse (la suffisance) est le moteur de la plupart des recherches en IA, car c'est sur elle que se basent les espoirs de construire des machines capables d'accomplir des tâches qui jusqu'à présent sont du ressort de l'être humain. Mais si les systèmes symboliques et physiques peuvent potentiellement avoir un comportement intelligent, comment feraient-ils pour y parvenir? La réponse est dans une autre hypothèse centrale en IA, celle de la *recherche heuristique* :

The solutions to problems are represented as symbols structures. A physical symbol system exercises its intelligence in problem solving by search — that is, by generating and progressively modifying symbol structures until it produces a solution (Newell & Simon, 1976).

Les deux éléments clés de la résolution de problèmes sont donc un test pour une classe de structures de symboles (but) et les générateurs de structures de symboles (opérateurs). Les systèmes symboliques physiques sont par principe capables de représenter l'espace de problèmes (l'ensemble de structures de symboles pouvant être généré) et d'avoir des générateurs.

Les méthodes de résolution de problèmes ont évolué depuis les premières expérimentations avec le GPS, comprenant aujourd'hui un vaste ensemble de techniques et modèles (McDermott, 1989; Wielinga, Schreiber & Breuker, 1992). Même des

chercheurs qui ont suivi une ligne différente comme Minsky, avec les agents, et Schank, avec le raisonnement à partir de cas, font de la résolution de problèmes. Ce qui reste central est l'idée initiale que les programmes doivent être conçus en supposant que l'activité humaine est toujours et complètement guidée par des buts (*goal-driven behavior*). Les motivations culturelles ou émotionnelles ainsi que les actions gratuites, comme papoter ou contempler, ne rentrent pas vraiment dans ce cadre (Dennett, 1993). Il s'agit donc d'une supposition forte car dans les actions il n'y a pas forcément un but précis, ou bien, il y a des buts concurrents, dont les interactions dépendent souvent du contexte (Agré, 1995; Clancey, 1993). C'est pourquoi quelques chercheurs (Johnson-Laird, 1987; Pressing, 1988) soutiennent que la résolution de problèmes ne serait pas adéquate pour les tâches de création musicale où les buts changent ou se raffinent au cours de la résolution; où les sous-divisions du problème ne sont pas toujours simples à établir; où il y a un certain "non-déterminisme" dans le comportement; etc.

Malgré cela, deux raisons nous conduisent à nous placer dans le cadre selon lequel jouer du jazz est résoudre des problèmes. Premièrement, nous voudrions pousser jusqu'au bout les deux hypothèses de travail en IA, à savoir l'hypothèse des systèmes symboliques et physiques et l'hypothèse de la recherche heuristique, car la résolution de problèmes reste le paradigme le plus probant jusqu'à présent en IA pour la plupart des domaines d'application étudiés. Comme ces hypothèses doivent être validées par une démarche expérimentale, et non pas être prises comme des dogmes (Dietrich, 1990), des domaines comme celui de la création musicale semblent être convenables. Deuxièmement, s'agissant de la création artistique, un domaine d'application encore obscur pour l'IA, nous espérons que le fonctionnement de notre programme sera plus facilement interprétable s'il s'inscrit dans un cadre bien connu et étudié.

Il existe une troisième hypothèse de travail centrale en IA et qui est étroitement liée aux deux citées. Il s'agit du *principe de la connaissance (the knowledge principle)* (Lenat & Feigenbaum, 1991), selon lequel plus on donne des connaissances aux machines plus elles peuvent agir intelligemment. Sans détailler ce que l'on entend par "connaissances", il faut dire que cette hypothèse s'est avérée vraie dans de nombreuses applications en IA jusqu'à présent. Néanmoins, elle reste une hypothèse. En effet, on voit aujourd'hui des machines très puissantes comme "Deep Thought" de chez IBM qui jouent très bien aux échecs en se basant énormément sur la recherche exhaustive. En

musique, plusieurs programmes de composition automatique, au lieu d'explorer cette voie dite *knowledge intensive*, font plutôt appel à des modèles mathématiques, physiques, bref non musicaux, qui vont des fractals (Monro, 1995), aux chaotiques (Little, 1996; Malt, 1996), en passant par l'utilisation des mesures de fluctuation du champ magnétique de la terre (Dodge, 1995), etc. Naturellement, ces programmes n'ont pas été conçus pour produire de la musique tonale. Nous nous plaçons ici dans cette troisième hypothèse de travail, car nous pensons que l'activité d'un jazzman est l'expression d'une expertise. La modélisation des connaissances liées à cette expertise suppose, par conséquent, une démarche d'acquisition de connaissances. A cet égard, nous renonçons à faire appel à des choix aléatoires, comme on fait dans la plupart des programmes de composition ou improvisation automatique. Faire appel à des choix aléatoires ou statistiques est, en quelque sorte, avouer que l'on n'est pas capable de saisir le comportement d'un phénomène. Malgré les difficultés d'accès et de formalisation des connaissances utilisées par les jazzmen, nous voulons entreprendre une démarche d'explicitation de ces connaissances afin de tester les trois hypothèses que nous avons rappelées dans cette section.

Il est important de souligner que ce n'est pas parce que nous nous plaçons dans un cadre d'IA symbolique, de résolution de problèmes et du principe de la connaissance, que nous voulons forcément adopter l'approche classique des systèmes à base de connaissances, à savoir "des règles de production, des heuristiques et un moteur d'inférence". C'est normalement à cela que l'on pense quand on évoque des hypothèses que nous venons de rappeler. Au contraire, nous gardons une ouverture à d'autres approches, notamment à celle du raisonnement à partir de cas sur laquelle, comme nous l'avons dit précédemment, nous avons travaillé dans le cadre de nos réflexions sur la problématique de la créativité (Cf. section A.b).

En conclusion de cette section, et pour que tout soit bien clair, rappelons que nous ne voulons ni révéler aux artistes (compositeurs et musiciens) de "nouveaux critères esthétiques" basés sur des concepts scientifiques ou technologiques ni, au moins dans un premier temps, offrir aux artistes de nouveaux outils technologiques pour leur création musicale. Notre objectif n'est pas non plus de faire de la simulation cognitive. Il y a des travaux en psychologie de la musique sur le problème de l'improvisation (Pressing, 1984; Pressing, 1988; Sloboda, 1985) qui peuvent nous être utiles, mais notre

système ne doit pas être jugé par sa plausibilité cognitive. Compte tenu de tout ce que nous avons évoqué dans cette section, nous voulons répondre aux questions suivantes.

- Quelle est la pertinence des trois hypothèses de travail énoncées et des techniques actuelles d'IA qui en découlent, vis-à-vis de la tâche musicale que nous allons traiter ?
- Quelles sont les connaissances et les mécanismes d'inférence qu'il faut donner à la machine pour qu'elle puisse créer des lignes de basse ?
- En quoi ces résultats peuvent-ils être généralisés au-delà du jazz et de l'art ?

C - NOTRE CONTRIBUTION

Dans cette section nous donnerons un aperçu de notre contribution. Nous commencerons par présenter très brièvement la tâche du bassiste de jazz et ensuite nous exposerons les solutions que nous avons apportées aux problèmes posés par cette tâche.

a) Brève description de la tâche du bassiste de jazz

La tâche d'un bassiste de jazz consiste à créer une mélodie appelée *ligne de basse*, en fonction d'une grille d'accord donnée et de ce que les autres musiciens du groupe jouent (Cf. figure B.1). Le problème du bassiste est de savoir quelle note jouer ensuite, et ce en respectant le tempo, car le déroulement du jeu se fait en direct. La grille d'accords⁸ contient une séquence temporelle d'accords chiffrés représentant la trame harmonique du morceau (que l'on peut aussi appeler "chanson"). C'est la grille, couramment le seul élément préexistant au jeu des musiciens, qui donne les indications principales de ce qui doit être joué. Selon l'accord les jazzmen savent comment une note va sonner. Une ligne de basse joue deux rôles principaux. D'une part, elle sert, avec toute la section rythmique, à maintenir la pulsation rythmique du morceau, assurant ainsi l'infrastructure du swing. D'autre part, elle sert, en compagnie de la partie du piano, à donner un support harmonique au soliste en soulignant les changements d'accords. Une mauvaise coordination d'un musicien par rapport aux autres gêne la performance globale du groupe. C'est une tâche difficile car la liberté de chaque musicien est très grande.

⁸Pour une vision plus complète et précise de la grille voir la figure 1.2.1

Grille:	... A7(b9)	Cmin7	F7	Fmin7	Bb7	Ebmaj7 ...
Basse:						
Solo:						

Figure B.1 - Tâche du bassiste de jazz

Il est important de remarquer que l'information fournie par la grille est insuffisante pour déterminer ce que les musiciens vont exactement jouer (Ramalho & Pachet, 1994). Certes, les informations supplémentaires de l'environnement sont importantes car elles restreignent un peu plus les choix des musiciens. Mais, elles engendrent d'autres complications concernant la façon dont tout s'articule, dont les conflits sont gérés, etc. Le problème fondamental est que, très souvent, les musiciens de jazz ne sont pas capables de justifier leurs choix de notes. La première raison est que, comme pour tous les experts, ils ont du mal à accéder à leurs connaissances puisqu'elles sont "câblées", ceci étant particulièrement vrai dans les activités se déroulant en temps réel. La deuxième raison dérive du caractère empirique de l'apprentissage du jazz fort basé sur l'écoute et la pratique (Ramalho & Ganascia, 1994a). C'est à cause de cette barrière dans l'explicitation de connaissances que pratiquement tout programme informatique essayant d'accomplir une tâche d'improvisation en jazz fait appel à des choix aléatoires (Cf. Chapitre 3).

b) Description de notre modèle

Pour un ordinateur, les notes d'une partition ou les accords d'une grille sont des symboles comme n'importe quels d'autres. Les jazzmen, néanmoins, perçoivent la musique d'une toute autre manière. À travers une démarche d'explicitation de connaissances auprès de quelques jazzmen, notamment Heriberto Paredes (pianiste et enseignant de l'IACP, Institut Art Culture et Perception, une des écoles de jazz à Paris) et Antoine Espagno (bassiste), nous avons essayé d'identifier les connaissances musicales impliquées dans l'improvisation et l'accompagnement en jazz. En donnant ces connaissances à la machine nous espérons qu'elle pourra disposer de critères pour interpréter les symboles représentant son environnement (grille, notes déjà jouées, etc.) et aussi choisir les symboles (notes) qui composent de bonnes lignes de basse. Ces connaissances sont nombreuses, concernant des tâches très variées comme l'analyse, la

synthèse, la planification, la reconnaissance de formes, etc. (Cf. chapitre 2). Citons les trois aspects de ces connaissances qui ont particulièrement attiré notre attention.

Le premier aspect est qu'au-delà des simples notes, un jazzman manipule et perçoit des *structures* (par ex. phrases mélodiques, enchaînement d'accords, sections, chorus, etc.) ainsi que des *traits musicaux* plus abstraits (par ex. densité, dissonance, contour mélodique, gamme, style, swing, contraste, etc.). De fait, la plupart du temps, les musiciens n'ont pas pu nous donner des justifications sur leur choix de chacune des notes jouées, prises individuellement. Leurs justifications concernent souvent un ensemble de notes (phrases) et sont basées sur des critères plus globaux comme "j'ai utilisé le mode dorien", "j'ai voulu souligner la reprise du thème par un contraste rythmique", "j'ai voulu augmenter la tension car le soliste jouait chromatique", etc. Bien entendu, chaque musicien a son propre "répertoire" de traits musicaux et aussi sa propre façon de les interpréter. De ces réponses nous avons tiré les conclusions que, d'une part, la "granularité" de la pensée d'un jazzman semble être plutôt "phrase par phrase" que "note par note", et que, d'autre part, le choix des notes passe par la satisfaction d'un certain nombre de critères liés aux traits musicaux (par ex. utiliser le mode dorien, créer un contraste rythmique, augmenter la tension). Mais puisqu'il s'agit de phrases mélodiques comment calculer leurs tailles ? En d'autres termes, comment déterminer continuellement le segment de la grille pour lequel le programme va calculer les notes d'une phrase ? La prise en compte des multiples traits musicaux pose aussi des problèmes. Comment représenter ces traits ? Comment les critères associés aux traits évoluent-ils pendant le déroulement du jeu d'un jazzman ? Comment les traits interagissent-ils ? Par exemple, quel est le lien entre la dissonance et le contour mélodique, entre le swing et la variation d'intensité des notes ? Comment traduire un ensemble quelconque de critères liés aux traits musicaux sous la forme de notes ? La grande majorité des programmes qui jouent du jazz, à quelques exceptions près (Baggi, 1992; Pachet, 1990), ne manipule pas ces traits musicaux de façon explicite car il est très difficile d'acquérir les connaissances qui concernent l'interaction mutuelle entre les traits musicaux ainsi que la "matérialisation" d'un ensemble de traits en termes de notes.

Le deuxième aspect important est que les règles de la théorie musicale sont très importantes mais elles n'englobent que partiellement le savoir-faire musical. Soit dans

les écoles de jazz, soit par des méthodes (Baker, 1980; Coker, 1970; Coolman, 1985), soit simplement en écoutant et imitant les grands maîtres, le jazzman acquiert les principaux concepts du jazz à travers des *exemples*, plutôt que par définitions ou règles. Il suffit de penser à des concepts comme le *swing* qui est un des éléments les plus caractéristiques du jazz pour lequel, pourtant, personne n'est capable de donner une définition précise. Pendant tout son parcours, un jazzman semble accumuler et créer toute une collection d'exemples de rythmes, de phrases mélodiques, d'enchaînement d'accords qui peuvent être réutilisés selon les circonstances. L'évidence la plus remarquable de ceci est donnée par les travaux des musicologues sur l'identification de catalogues de patterns mélodiques ou rythmiques (fragments mélodiques et rythmiques récurrents) utilisés par plusieurs jazzmen célèbres (Baker, 1980; Kernfeld, 1983; Owens, 1974; Smith, 1983). Bien entendu, jouer du jazz ne se limite pas à réutiliser des fragments mélodiques ou rythmiques, mais du point de vue de la construction d'un programme censé jouer du jazz, ces fragments sont très utiles. En effet, ces fragments codent "en extension" (au sens où l'on l'entend en IA) des connaissances difficiles à formaliser comme celles concernant le rythme, les accents, le son, les doigtés, etc. Actuellement, les programmes qui construisent une improvisation en réutilisant systématiquement des patterns mélodiques ou rythmiques préalablement stockés sont ceux qui produisent les résultats musicaux les plus satisfaisants (Baggi, 1992; Band-in-a-box, 1995; Hodgson, 1996b). Mais l'incorporation de fragments mélodiques ou rythmiques dans un programme soulève quelques questions. Quels sont les critères pour le choix d'un fragment ? Comment une bibliothèque de fragments devrait-elle être organisée ? Comment indexer ces fragments ?

Le troisième aspect est que le jeu d'un jazzman dépend de son *environnement* (le public et surtout les autres musiciens). L'influence mutuelle entre les musiciens est une caractéristique marquante du jazz, car, pour un jazzman, il n'est pas simplement question de bien jouer sa partie mais de le faire en fonction du groupe. Peu d'efforts ont néanmoins été faits pour incorporer l'environnement dans les programmes d'improvisation actuels car l'automatisation de l'écoute est très difficile. Elle implique en effet le traitement de différents aspects de la perception musicale : le suivi du tempo (Allen & Dannenberg, 1990; Desain & Honing, 1992), la quantification rythmique (Agon et al., 1994; Desain & Honing, 1992; Roozendaal, 1990), la détection des frontières des phrases (Lerhdal & Jackendoff, 1983; Narmour, 1989), etc. Une autre caractéristique liée à

l'environnement est qu'un jazzman joue "en direct", c'est-à-dire en respectant le tempo dans lequel les autres musiciens jouent. Le fait de jouer "en direct" a deux impacts sur le raisonnement d'un jazzman : le manque de temps pour réfléchir et l'interdiction de revenir en arrière. Le jazzman doit, continuellement et rapidement, s'adapter à de nouveaux événements. Malgré les travaux récents en planification et résolution de problèmes en temps réel (Musliner et al., 1995; Stroddiner, 1994), les contraintes de temps n'ont pratiquement été pas considérées par les programmes actuels.

Après avoir réalisé cette démarche d'explicitation de connaissances et après avoir analysé les programmes existants dans le domaine du jazz, nous avons décidé de prendre comme point de départ de notre programme le modèle d'*agent rationnel* tel qu'il s'est dégagé en IA (Ganascia, 1990c; McCarthy & Hayes, 1969; Newell, 1982). La notion d'agent nous a paru tout à fait convenable en tant que cadre général d'intégration de toutes ces connaissances explicitées auprès des jazzmen (Cf. chapitres 1 et 4). De fait, la notion d'agent rationnel nous permet d'expliquer le comportement d'une machine en lui prêtant des buts et un principe de rationalité : un agent rationnel choisit toujours l'action qui va satisfaire le mieux ses buts. Ainsi, l'agent rationnel a l'avantage de fixer un cadre général d'analyse et de conception des programmes sans que l'on ait à détailler leur structure interne ou la façon dont leurs connaissances y sont représentées. Par ailleurs, la notion d'agent rationnel nous permet de poser le problème de la construction de notre programme dans toute sa complexité et généralité. En effet, cette notion s'inscrit dans l'effort mené, notamment par Newell, pour proposer un cadre d'unification de théories cognitives et de systèmes intelligents (Laird, Newell & Rosebloom, 1987; Newell, 1990). Selon ce cadre, on doit s'intéresser aussi bien aux mécanismes de raisonnement qu'aux problèmes de perception, d'exécution et d'apprentissage.

Ensuite, nous avons essayé de trouver une formulation de la tâche d'improvisation ou d'accompagnement en termes de résolution de problèmes. Certains chercheurs (Johnson-Laird, 1992; Pressing, 1988) soutiennent qu'une telle formulation ne peut être faite adéquatement parce qu'il est extrêmement difficile d'établir des buts bien définis et figés (et donc un critère d'arrêt) pour la résolution. En effet, au départ du jeu, les musiciens semblent avoir uniquement une impression vague et des idées parfois contradictoires sur ce qu'ils veulent jouer. Ce n'est qu'au cours de l'improvisation que

les idées se précisent. Les critiques de ces chercheurs sont basées sur une formulation selon laquelle, étant donné un ensemble de mesures vides, le problème musical consiste à les remplir avec des notes qui satisfont certains critères. Comme ces critères sont justement difficiles à déterminer car ils se raffinent et changent avec le temps, le programme ne saura pas s'il a atteint le but. Nous proposons donc une formulation originale selon laquelle le problème de la génération d'une improvisation ou d'un accompagnement *par un agent rationnel* est triple (Cf. chapitre 4). Le premier problème est de déterminer le *segment* (laps de temps) pour lequel l'agent va calculer les notes. Le deuxième problème est justement d'avoir la spécification la plus précise, cohérente et complète possible des *critères* que les notes du segment d'improvisation doivent respecter. Le troisième problème est de concrétiser ces critères en termes de *notes*. Ces trois problèmes se posent continuellement tout au long du jeu de l'agent jazzman. À partir de ces réflexions nous avons eu l'idée de construire un agent jazzman (Ramalho & Ganascia, 1994b) dont le fonctionnement suit trois étapes principales. D'abord, l'agent détermine le *segment d'improvisation* selon des séquences d'accords connues (par. ex II-V, II-V-I, etc.) qui apparaissent sur la grille. Ensuite, il se fixe un certain nombre de critères (liés à un répertoire de traits musicaux) en fonction de ce qu'il a joué, de ce que les autres jazzmen ont joué, et des informations affichées sur la grille d'accords. Ces critères étant probablement contradictoires et incomplets, l'agent essaye de les combiner et d'en enlever les conflits. Enfin, quand les critères deviennent suffisamment clairs, ils évoquent chez notre agent un souvenir d'un fragment mélodique mémorisé dans le passé. En d'autres termes, une fois que les critères sont bien définis, l'agent cherche, dans ce que nous appelons la *Mémoire Musicale* (Ramalho & Ganascia, 1994a), un fragment mélodique de ligne de basse qui satisfait le mieux à ces critères. Ce modèle d'agent jazzman découle des trois résultats de notre travail d'explicitation de connaissances et de réflexion sur les programmes actuels que nous avons décrits au début de cette section. Le choix de segment tient à la supposition que les jazzmen progressent plutôt "phrase par phrase" (phrases une des longueurs variables et pouvant se chevaucher) que "note par note". En outre, les justifications que les jazzmen nous ont données sur leurs choix nous ont fait comprendre que le choix des notes devrait passer par un certain nombre de critères liés aux traits musicaux. Aussi, l'influence de l'environnement peut-elle être prise en compte pour la fixation de certains critères. Enfin, la constatation que des fragments mélodiques représentent en extension

énormément de connaissances des jazzmen nous a mené à voir un fragment comme la “matérialisation” même d’un ensemble de critères

Les fragments mélodiques que nous voulons stocker dans la Mémoire Musicale sont de vrais “cas” car, d’une part, ils correspondent à des passages réellement joués par quelqu’un et non à des connaissances d’ordre général et, d’autre part, ils peuvent être effectivement réutilisés dans des conditions similaires. Ainsi, nous avons fait appel aux techniques du raisonnement à partir de cas (Kolodner, 1993) pour l’organisation de la Mémoire Musicale (base de cas) et pour la récupération des fragments mélodiques (cas) de la Mémoire Musicale. En effet, le raisonnement à partir de cas est un outil conceptuel très adéquat à notre problème car il s’appuie sur l’expérience pour résoudre de nouveaux problèmes. Plus précisément, un cas est constitué de la description d’un problème et de sa solution. Quand un nouveau problème se présente, on récupère d’une base de cas le cas dont le problème est le plus similaire au nouveau problème et ensuite on adapte la solution contenue dans le cas récupéré au nouveau problème. Une des grandes difficultés du raisonnement à partir de cas est de savoir quelle est la description (ensemble d’index) la plus adéquate du problème pour que l’on puisse récupérer la bonne solution. La “partie solution” d’un cas de la Mémoire Musicale est un fragment mélodique de ligne de basse obtenu à partir de transcriptions d’enregistrements de Ron Carter (Aabersold, 1979). La “partie problème”, qui indexe le fragment, est composée des éléments suivants : les traits musicaux du fragment; la description du segment de la grille où le fragment a été joué (le schéma d’accord, la position dans la grille, la tonalité locale, etc.); les traits musicaux du fragment mélodique précédent et la description du segment de grille sur lequel ce dernier a été joué. Nous aurions pu étendre l’indexation du fragment mélodique à toutes les notes et tous les accords de la grille déjà joués ainsi qu’aux accords à venir. Toutefois, un trop grand nombre d’index peut biaiser incorrectement la récupération du cas de la base (Kolodner, 1993) [chap. 6]. En plus, ceci rendrait le calcul de la mesure de similarité trop coûteux car il faudrait faire appel à des algorithmes assez complexes d’appariement des notes (Rolland & Ganascia, 1997).

Maintenant, examinons chacune de ces trois étapes du raisonnement de notre agent jazzman : la détermination du segment d’improvisation; la fixation des critères portant sur les notes du segment et le calcul de ces notes.

La détermination du segment d'improvisation sur lequel le programme va travailler est très délicate car la solution qui consisterait à choisir des phrases de taille quelconque qui puissent s'enchaîner (ce que les jazzmen semblent faire) pose beaucoup de difficultés (Cf. section 3.4.2). Les programmes actuels choisissent des segments correspondant à des éléments facilement repérables comme un temps, la durée d'un accord, la durée d'une mesure, etc. Nous proposons une segmentation originale basée sur les *schémas d'accords* de type II-V, V-I, II-V-I, VI-II-V-I, etc. qui représentent des enchaînements d'accords typiques sur les grilles de jazz (Baudoin, 1990) [vol. 1, pp. 57-98]. Ce choix de segment permet, d'une part, de garder une certaine plausibilité musicale (les phrases de basse coïncident souvent avec les schémas d'accords) et, d'autre part, d'établir un bon compromis pour la "granularité" du raisonnement de l'agent (Cf. section 4.3.2). Un segment trop petit (par exemple un temps) rend la cohérence globale d'une improvisation plus difficile à atteindre, tandis qu'un segment trop grand (par ex. le morceau entier) nuit à la capacité du programme à "changer d'avis" pendant l'improvisation car tout est calculé en avance. C'est le compromis classique entre les niveaux local et global. La segmentation que nous proposons est faite automatiquement à travers une analyse harmonique particulière, exécutée à l'aide de règles de production et d'un moteur d'inférence de premier ordre en chaînage avant.

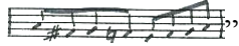
Pour pouvoir déterminer et manipuler les critères selon lesquels les fragments mélodiques seront récupérés, nous faisons appel à la notion de PACT (Potential ACTION) introduites par F. Pachet (Pachet, 1987; Pachet, 1990). Les PACTs (auparavant appelées "stratégies") sont des "actions potentielles" qui sont suscitées à un moment donné chez le jazzman concernant les notes qu'il va jouer dans un futur proche ou lointain. Voici quelques exemples de PACTs : "pendant cette mesure, jouer la gamme diatonique en direction ascendante", "jouer de plus en plus fort jusqu'à la fin du chorus d'improvisation", "jouer maintenant ce rythme ♩ ♩ ♩ ♩", "jouer beaucoup de notes et très syncopé la prochaine fois que l'on viendra sur ces accords", "jouer la dernière phrase en la transposant un ton plus haut", etc. Certaines PACTs peuvent donc être vues comme l'incarnation de ces "critères" (par ex. la PACT "... jouer la gamme diatonique en direction ascendante" fixe des critères concernant les traits musicaux "gamme" et "contour mélodique") avec une dimension temporelle car, comme toute action, une PACT a un début et une durée.

Maintenant que les PACTs ont été présentées, nous pouvons dire que la définition des critères selon lesquels les notes seront calculées se fait de la façon suivante : on “active” un certain nombre de PACTs, on sélectionne celles qui se superposent au segment d’improvisation courant et on “assemble” ces PACTs sélectionnées. Voyons plus en détail ce que signifient le processus d’activation et d’assemblage des PACTs.


Les PACTs sont *activées* par l’agent en fonction des informations contenues sur la grille, de ce que l’agent a déjà joué et ce que les autres musiciens jouent. Comme c’était le cas dans la proposition originelle de F. Pachet, nous utilisons des règles de production du type “si une situation *S* est reconnue, alors activer PACT *P*”. Par exemple, si l’agent est sur un accord altéré, alors le programme active la PACT “jouer chromatique”; si l’agent est au début de l’improvisation et si pendant les derniers segments il a joué une ligne basée sur la blanche, alors le programme active la PACT “jouer avec un rythme basé sur la noire”; et si le batteur joue doucement, alors c’est la PACT “jouer doucement” qui est activée. Les PACT qui seront prises en compte pour le calcul des notes d’un segment donné de la grille sont celles dont la durée de vie se superpose à ce segment. La dimension temporelle des PACTs est donc très importante. D’une part, elle permet de minimiser l’effet de la fragmentation du processus de raisonnement de l’agent jazzman qui calcule les notes segment par segment. En effet, les PACTs peuvent être activées pour durer assez longtemps et ainsi “influencer” le calcul des notes de plusieurs segments. D’autre part, puisque les PACTs peuvent avoir différentes durées de vie, nous pouvons simuler le fait que pendant l’improvisation il y a différentes “configurations” de critères pour le calcul des notes.

Nous avons dans le paragraphe précédent dit que certaines PACTs sont activées par rapport à “ce que les autres musiciens jouent”. En réalité, nous n’avons pas mis en œuvre la partie qui s’occuperait de l’écoute de l’environnement car, comme nous l’avons dit au début de cette section, cela est très difficile. Pour s’affranchir partiellement du problème de l’automatisation de l’écoute nous avons proposé la notion de *Scénario* qui contient la description symbolique, et donc interprétée a priori, des événements de l’environnement comme “le pianiste joue dans le mode dorien”, “le batteur joue de plus en plus fort depuis deux mesures”, “le soliste joue beaucoup de notes”, etc. (Cf. chapitre 4). Bien que le Scénario soit donné d’emblée au programme,

l'accès à un événement qu'il contient n'est possible qu'à partir de la date de début qui lui a été affectée (Ramalho, Rolland, & Ganascia, 1996).

Il reste à déterminer comment un ensemble de PACTs concernant un segment de la grille peut être “assemblé” pour que la requête qui sera posée à la Mémoire Musicale puisse être la plus complète et la moins conflictuelle possible. Nous avons travaillé sur la notion de PACTs pour essayer d'en dégager quelques propriétés qui pourraient nous permettre de définir une méthode de résolution pour l'*assemblage des PACTs*. Les deux propriétés centrales des PACTs que nous avons retenues sont la *jouabilité* et la *combinabilité*. Une PACT peut être plus ou moins *jouable* selon l'information qu'elle contient sur les notes à jouer. Par exemple, La PACT “jouer maintenant ce rythme ↓ ↓ ↘ ↓” est plus jouable que “jouer avec un rythme basé sur la noire”, ainsi que “jouer la gamme diatonique en direction ascendante” est plus jouable “jouer en direction descendante”. Un exemple de PACT (totalement) jouable est “jouer maintenant ces notes ”. La propriété de jouabilité d'une PACT est très importante car elle établit une relation d'ordre qui intervient dans la résolution de conflits entre deux PACTs incompatibles, la PACT la plus jouable étant préférée. Ainsi, entre les PACTs “jouer la gamme diatonique en direction ascendante” et “jouer en direction descendante” qui sont en conflit au niveau du trait “contour mélodique”, on préfère la première. D'autre part, selon la propriété de combinabilité, deux PACTs peuvent, s'il n'y a pas de conflits entre elles, se combiner pour engendrer une PACT plus jouable. Par exemple, la PACT “jouer de plus en plus fort jusqu'à la fin du chorus d'improvisation” combiné avec “swinguer d'ici deux mesures jusqu'à la fin” peut donner “jouer de plus en plus fort et avec swingue d'ici deux mesures jusqu'à la fin”. C'est cette propriété qui est au cœur de notre méthode de résolution de problèmes pour l'assemblage des PACTs (Ramalho & Ganascia, 1994b). L'état initial est un ensemble de PACTs et le but est une PACT jouable. Les PACTs sont assemblées, à l'aide de règles de production, par l'application successive d'opérateurs de résolution de conflits et de combinaison jusqu'à ce que l'on obtienne une PACT jouable.

L'obtention d'une PACT jouable, c'est-à-dire la concrétisation des critères en termes de notes, se fait précisément via la Mémoire Musicale à travers des mécanismes de raisonnement à partir de cas. Puisque les PACTs permettent une description aussi bien partielle (par ex. “jouer très dissonant”) que complète (par ex. “jouer maintenant

ces notes ) de ce que l'on va jouer, nous avons décidé de représenter aussi les fragments mélodiques de la Mémoire Musicale comme des PACTs jouables. Nous unifions donc la représentation des “traits musicaux” et des fragments mélodiques. Ainsi, lorsqu'une PACT en voie d'assemblage devient “suffisamment jouable” ou, simplement, lorsqu'elle est la seule PACT restante, le programme récupère de la Mémoire Musicale la PACT jouable la plus similaire à cette PACT en voie d'assemblage. Ensuite, le fragment mélodique associé à la PACT jouable récupérée est adapté (par des opérations de transposition, de changement d'amplitudes, d'ajout de notes, etc.) et affecté à la PACT en voie d'assemblage, qui, du coup, devient jouable. Par exemple, supposons que le programme soit en train de jouer le segment⁹ (Amin7(b5) D7) qui commence à la cinquième mesure de FEUILLES MORTES (Cf. figure B.2) et que la seule PACT restante en voie d'assemblage soit “jouer dans une tessiture basse, pas dissonant, moyennement syncopé, avec un rythme basé sur la blanche et en arpèges”. Cette PACT a été obtenue à partir de la combinaison des PACTs qui sont activées “par défaut” lorsqu'il s'agit du premier chorus d'un morceau joué dans un tempo moyen (entre 120 et 200 noires par minute). Alors, le programme recherche dans la base un fragment mélodique qui a ces propriétés (tessiture basse, pas dissonant...), qui a été joué sur un segment de grille semblable (II-V mineur où chaque accord dure 4 temps, etc.) et qui a été précédé par un fragment ayant des caractéristiques semblables à celles du fragment des mesures 3 et 4. Le fragment jugé le plus adéquat (Cf. figure B.3) est celui joué originellement sur Dmin7(b5) G7 aux mesures 13 et 14 de WHAT IS THIS THING CALLED LOVE (Cf. Annexe A).



Cherokee:78-81 Wittcl: 7

1 C min 7 F 7 Bb maj 7 Eb maj 7

5 A halfDim 7 D 7 G min 7 G min 7

Figure B.2 - Passage généré sur FEUILLES MORTES. Le nom du morceau ainsi que les numéros des mesures originelles de chaque fragment sont indiqués au-dessus des segments (Wittcl = WHAT IS THIS THING CALLED LOVE)

⁹Le programme d'affichage de partitions que nous utilisons indique les accords demi-diminués (par ex. Am7(b5)) par le suffixe “halfDim7”

Le tableau B.1 résume l'enchaînement de différentes étapes du fonctionnement de notre programme.

→↔☒	Déterminer le segment <i>S</i> de la grille selon les schémas d'accords;
→●☒	Activer des nouvelles PACTs en fonction des événements récents du Scénario, de ce que l'agent a joué jusqu'à présent et de la grille;
→✓☒	Sélectionner les PACTs (activées récemment ou auparavant) dont la durée de vie se superpose au segment <i>S</i> ;
→✓☒	Assembler les PACTs sélectionnées jusqu'à obtenir une PACT jouable.

Tableau B.1 - Boucle principale de notre programme

En réalité, le programme ne suit pas toujours les quatre pas de l'algorithme du tableau B.1 (Cf. chapitre 6). Il y a des changements en fonction de la disponibilité des ressources de temps. Comme le déroulement du jeu se fait en temps réel, l'agent est obligé de planifier en avance ce qu'il va jouer. Quand l'agent est trop en retard, c'est-à-dire quand l'écart de temps entre les notes planifiées et les notes exécutées est trop petit, l'étape d'activation de PACTs est réduite à l'activation d'une seule PACT "jouer fragment mélodique cliché" et l'assemblage est court-circuité. L'agent cherche alors directement dans la Mémoire Musicale un fragment mélodique qui est très récurrent (un pattern mélodique) chez Ron Carter. Il existe des techniques de résolution de problèmes en temps réel (Musliner et al., 1995; Strosdiner, 1994), qui pourraient être sans doute utilisées pour garantir que l'activation et l'assemblage des PACTs se fassent même quand le temps disponible est assez petit. Cependant, comme l'a fait F. Pachet (Pachet, 1990), nous avons préféré "jouer" avec le manque de temps pour changer le mode de raisonnement de l'agent, en supposant que le manque de temps est un des facteurs qui pousse les jazzmen à jouer des clichés.

Il faut signaler que, pour que le modèle que nous venons de présenter devienne opérationnel, nous avons dû mener un travail de formalisation très important de la notion de PACTs. La formulation de PACTs faite dans le programme de F. Pachet était encore incomplète et trop ad hoc. Nous avons défini toutes les inférences impliquant les PACTs (mesure de jouabilité, détection de conflits, combinaison, etc.) et nous avons inséré les PACTs au sein d'une méthode de résolution de problèmes bien précise. Par ailleurs, nous avons proposé une représentation plus adéquate des PACTs à l'aide d'un

langage hybride qui combine les langages de frames, les langages de classes et les graphes conceptuels (Cf. chapitre 5) .

L'importance du raisonnement à partir de cas dans le processus d'assemblage de PACTs est de garantir que, pour n'importe quel ensemble de critères associés aux différents traits musicaux (représentés par les PACTs en voie d'assemblage), l'on peut toujours trouver un fragment mélodique dans la Mémoire Musicale qui les satisfasse le mieux. Ceci est très important pour deux raisons.

La première raison est que l'agent doit être en mesure de produire des notes même si elles sont "sous-spécifiées", c'est-à-dire même avec peu de critères pour le calcul des notes. Le problème est que l'on n'est pas capable d'explicitement toutes les connaissances nécessaires pour avoir un ensemble "complet" de critères. Les musiciens en effet donnent souvent des justifications "partielles" sur les notes qu'ils ont jouées (par ex. "j'ai utilisé le mode dorien", "j'ai voulu augmenter la tension", etc.) (Cf. chapitre 2). Autrement dit, il n'y a pas de règles pour activer des PACTs concernant tous les traits musicaux d'un fragment de ligne de basse à chaque segment d'improvisation. La plupart du temps, les PACTs activées fixent à peine quelques critères. Néanmoins, avec une recherche dans la Mémoire Musicale notre programme peut toujours trouver un fragment mélodique qui respecte les critères existants, si peu nombreux soient-ils. Dans le cas extrême où il n'y a pas de PACTs activées (et donc pas de critères associés aux traits musicaux), le programme récupère le fragment de la Mémoire Musicale en se basant uniquement sur les caractéristiques du segment de la grille courant et du fragment qui a été joué auparavant.

La deuxième raison tient au fait que la concrétisation de divers traits en termes de notes pose de nombreuses difficultés (Cf. chapitre 3). D'abord, il n'existe pas d'unanimité dans la signification précise de certains traits. Par exemple, on peut énumérer au moins une douzaine d'exemples de fragments qui peuvent être considérés comme "syncopés". Ensuite, on ne connaît pas exactement l'influence mutuelle d'un ensemble de traits. Ainsi, on sait que l'utilisation de la gamme chromatique et l'absence de la tonique des accords rend un passage dissonant, mais il existe aussi des liens plus subtils. Par exemple, jouer dans une tessiture plus haute peut suggérer plus de dissonance dans certaines circonstances. Etant donné un ensemble de critères associés à divers traits, comment peut-on donc savoir s'il est cohérent et s'il existe une séquence

de notes qui satisfasse aux critères ? Si la réponse à cette question s'avère positive, dans quel ordre les traits doivent-ils être considérés pendant le calcul d'une telle séquence de notes ? Comme personne ne semble avoir de réponses précises à ces questions, les programmes actuels qui essaient de prendre en compte les traits musicaux dans leur raisonnement, réduisent le vocabulaire de traits et figent leur interaction. En plus, les critères associés à ces traits ne changent pas pendant le déroulement du morceau. Avec notre approche, nous sommes capables de traiter explicitement le nombre de traits le plus grand jamais considéré. Ceci est possible parce qu'au lieu de montrer comment achever une phrase mélodique "valable" à l'égard d'un certain nombre de critères, nous montrons ce qui constitue des exemples de phrases mélodiques "valables" et nous dotons le programme d'un mécanisme de récupération et d'adaptation de ces phrases en fonction de ces critères.

The figure shows two staves of musical notation in bass clef. The first staff is divided into two sections: 'Cherokee:78-81' and 'Wittcl: 7'. The first section has two measures with chords C min 7 and F 7. The second section has two measures with chords Bb maj7 and Eb maj7. The second staff is labeled 'Wittcl: 13-14' and has four measures with chords A halfDim7, D 7, G min 7, and G min 7. A question mark is placed above the final measure of the second staff.

Figure B.3 - Passage généré sur FEUILLES MORTES (complété de B.2 - version 1)

L'importance de l'étape d'activation et d'assemblage des PACTs dans la récupération des cas est qu'elle permet un raffinement et un enrichissement de la requête qui sera formulée à la Mémoire Musicale. Au lieu de simplement adresser la requête "cherchez un fragment mélodique joué sur tels accords, et dont le fragment précédent a tels traits musicaux et tels accords sous-jacents", nous (ré)interprétons la description du problème et nous adressons la requête "cherchez un fragment mélodique qui a une tessiture basse, qui n'est pas dissonant, qui est moyennement syncopé, qui a un rythme basé sur la blanche, qui a des arpèges, qui a été joué sur tels accords, et dont le fragment précédent a tels traits musicaux et tels accords sous-jacents". D'une part, ceci nous permet de compenser les réductions que nous avons faites dans l'indexation des cas, car dans la requête que le programme pose à la Mémoire Musicale, il y a implicitement l'influence de la grille dans sa totalité, de toute la ligne de basse jouée jusqu'à présent et de tous les événements du scénario. D'autre part, ceci nous donne plus de contrôle sur la récupération du cas, car le grand défaut d'une mesure numérique

de similarité est qu’il est difficile de maîtriser l’influence de chaque descripteur (index) des cas (les traits musicaux du fragment mélodique, le type de schéma d’accord, la tonalité locale, la position dans la grille, le tempo, les traits du fragment précédent, etc.) (Kolodner, 1993) [chap. 8]. L’étape d’activation et d’assemblage des PACTs nous permet donc de “biaiser” le choix du fragment. Ceci est d’autant plus utile que c’est à travers l’activation des PACTs que le programme peut tenir compte des événements du Scénario. Reprenant notre exemple de ligne de basse sur FEUILLES MORTES, supposons qu’au moment de calculer les notes du segment en question, le programme ait détecté des événements du scénario de type “le soliste commence à jouer chromatique”, alors le programme activerait la PACT “jouer très dissonant et en *stepwise*” et le fragment récupéré (Cf. figure B.4) serait celui joué par Ron Carter aux mesures 61 et 62 de STELLA BY STARLIGHT (Cf. Annexe A).

The figure displays two staves of musical notation in bass clef. The top staff, labeled 'Cherokee:78-81' and 'Wittcl: 7', contains measures 1 through 7. Above the staff, brackets group the measures with the following chords: C min 7 (measures 1-2), F 7 (measures 3-4), Bb maj7 (measures 5-6), and Eb maj7 (measures 7-8). The bottom staff, labeled 'Stella by starlight: 61-62', contains measures 5 through 6. Above the staff, brackets group the measures with the following chords: A half D min 7 (measures 5-6), D 7 (measures 7-8), G min 7 (measures 9-10), and G min 7 (measures 11-12). A large question mark is positioned above the final measure of the bottom staff.

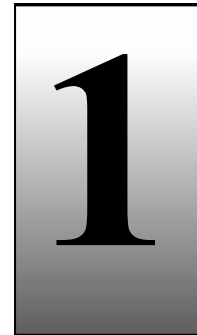
Figure B.4 - Passage généré sur FEUILLES MORTES (complété de B.2 - version 2)

La force de notre modèle est justement dans cette symbiose entre le raisonnement déductif (activation et assemblage des PACT) et le raisonnement à partir de cas (récupération et adaptation d’un fragment mélodique de la Mémoire Musicale). Notre modèle offre la possibilité au concepteur de représenter sous forme de règles toutes les connaissances qui peuvent être explicitées pour l’activation et assemblage des PACTs, tout en garantissant la résolution du problème quoiqu’il arrive via une recherche dans la Mémoire Musicale. Plus nous sommes capables d’expliciter des connaissances (même si elles sont incomplètes et un peu contradictoires), plus la requête à la Mémoire Musicale est précise.

Nous avons mis au point le système *ImPact*, qui a été programmé en Smalltalk-80 (Goldberg & Robson, 1983) en réutilisant largement la plate-forme de représentation de connaissances en musique tonale *MusES* (Pachet, Ramalho & Carrive, 1996). *ImPact* est un système assez important contenant plus de 130 classes et de 2500 méthodes, au sens de

la programmation par objet. *ImPact* contient aussi sept bases de règles de production comprenant plus de 70 règles en tout et une base de cas composée de 267 cas, tous obtenus à partir de lignes de basse jouées par Ron Carter (Aabersold, 1979). Pour donner une idée de l'ordre de grandeur, si l'on joue les cas actuels les uns après les autres dans un tempo de 180 la noire, ceci représenterait à peu près dix minutes de musique. Le système *ImPact* nous a permis de valider les idées exposées dans ce mémoire et de comparer nos résultats à ceux obtenus par d'autres systèmes. Les résultats musicaux que nous avons obtenus sont globalement aussi bons que ceux des deux "meilleurs" programmes pouvant générer des lignes de basse (Baggi, 1992; Band-in-a-box, 1995) (Cf. chapitre 7). Par ailleurs, il est facile d'ajouter des nouvelles connaissances dans notre système et ainsi de le faire évoluer. Ces connaissances peuvent être introduites à travers l'ajout de nouveaux cas dans la Mémoire Musicale et l'insertion de nouvelles règles d'activation de PACTs.

Pour conclure nous pouvons résumer nos trois principales contributions. La première est l'intégration d'un ensemble très important de *connaissances* et de modes de raisonnement au sein d'un modèle d'agent jazzman. Ces connaissances vont des simples réactions à l'environnement à l'analyse harmonique, en passant par des changements de comportement selon les ressources de temps disponibles, etc. En particulier, nous proposons une intégration, dans un modèle original de résolution de problèmes, des mécanismes de raisonnement déductif et de raisonnement à partir de cas. La deuxième contribution est l'enrichissement et la formalisation de la notion de *PACTs*. En partant d'une proposition encore incomplète des PACTs, nous avons entrepris un long travail d'acquisition d'un large vocabulaire (ontologie) de PACTs concernant la basse, d'insertion des PACTs dans une méthode de résolution de problèmes bien définie, de détermination formelle des opérations impliquant les PACTs. La troisième contribution est l'introduction même de la notion de *Mémoire Musicale*. Bien que des fragments mélodiques aient déjà été utilisés par d'autres programmes (Band-in-a-box, 1995; Hodgson, 1996a), la façon dont nous les indexons et contrôlons leur récupération est originale. L'originalité du processus de raffinement de la requête à la base de cas, s'étend d'ailleurs aux programmes de raisonnement à partir de cas appliqués à des domaines non-musicaux.



1. CARACTERISATION DE LA TACHE D'UN AGENT JAZZMAN

Dans ce chapitre nous faisons une analyse plus approfondie de la tâche d'un jazzman que nous avons brièvement présentée dans l'introduction de ce mémoire. Il ne s'agit pas de mener une analyse musicologique ou psychologique de l'activité d'un jazzman, mais de caractériser la tâche, les buts et les actions d'un programme jouant du jazz, ainsi que l'environnement dans le quel il doit accomplir telle tâche.

Pour ce faire, nous faisons appel à la notion d'agent rationnel (ou agent cognitif) (Ganascia, 1990c; McCarthy & Hayes, 1969; Newell, 1982) selon laquelle on peut expliquer le comportement d'une machine en lui prêtant des buts et un principe de rationalité : un agent rationnel choisit toujours l'action qui va satisfaire le mieux ses buts. La notion d'agent rationnel a l'avantage de fixer un cadre général d'analyse et de conception des programmes sans que l'on ait à détailler leur structure interne ou la façon dont leurs connaissances sont représentées. Par ailleurs, la notion d'agent rationnel nous permet de poser le problème de la construction de notre programme dans toute sa complexité et généralité. En effet, cette notion s'inscrit dans l'effort mené, notamment par Newell, pour proposer un cadre d'unification de théories cognitives et de systèmes intelligents (Laird, Newell & Rosebloom, 1987; Newell, 1990). Selon ce cadre, on doit s'intéresser aussi bien aux mécanismes de raisonnement qu'aux problèmes de perception, d'exécution et d'apprentissage.

Nous commençons ce chapitre en introduisant quelques concepts relatifs à l'agent rationnel en général pour, ensuite, nous pencher sur l'activité d'un agent jazzman. C'est à partir de ces concepts, surtout en ce qui concerne la structure perception-raisonnement-action, que nous analyserons les connaissances musicales chez

les jazzmen (Chapitre 2) et dans les programmes informatiques (Chapitre 3), et que nous présentons notre propre modèle d'un agent rationnel jazzman (chapitre 4).

1.1. LA NOTION D'AGENT RATIONNEL

1.1.1. Comportement et structure

Un *agent* est un système (un programme, un robot, etc.) qui *perçoit* son environnement à travers des capteurs (une camera, un clavier, un microphone, etc.) et qui *agit* sur cet environnement à l'aide d'effecteurs (un écran, une imprimante, des mains, etc.). L'*agent rationnel* est un agent qui suit un *principe de rationalité* selon lequel, pour chaque séquence de données perceptives (reçue par les capteurs), l'agent doit choisir une action qui maximise son succès vis-à-vis des buts à accomplir, et cela en faisant appel à des connaissances (Cf. figure 1.1.1). Le premier intérêt de la notion d'agent est qu'elle nous permet de rendre compte du comportement d'un programme ou d'un robot en se basant uniquement sur ses buts et ses connaissances. Ainsi, bien que la notion d'agent rationnel soit issue des systèmes symboliques physiques (Newell, 1980; Newell & Simon, 1976; Simon, 1981), elle est suffisamment générale pour comprendre aussi d'autres approches comme le connexionnisme, le raisonnement probabiliste, l'IA distribuée (multi-agent), etc. (Russel & Norvig, 1995) [pp. 31-52].

La figure 1.1.1 montre l'architecture de base d'un agent. Elle s'articule autour de trois modules : *perception*, *raisonnement* et *exécution*. Les trois composants principaux du module de raisonnement sont les *buts* que l'agent veut atteindre, l'ensemble d'*actions* que l'agent peut exécuter sur l'environnement et les *connaissances* qui permettent à l'agent de sélectionner l'action appropriée. Il s'agit d'une vision globale de l'agent. Selon l'environnement et la tâche à effectuer, cette architecture doit être détaillée.

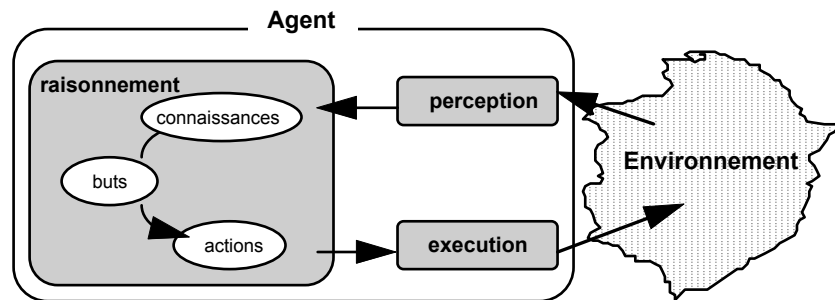


Figure 1.1.1 - Modèle général d'un agent rationnel

Un autre avantage de la notion d'agent rationnel est qu'elle permet de faire sortir l'IA du cadre assez réducteur du "logicisme" typique des années soixante et soixante-dix, où la modélisation est centrée sur le "raisonnement", prêtant peu d'attention à l'environnement et, par conséquent, à la perception, à l'exécution et à l'apprentissage. Quand on construit un agent qui va évoluer dans un environnement dynamique avec des contraintes de temps, comme celui d'un jazzman, la capacité de faire des inférences sophistiquées est aussi importante que la capacité de percevoir l'environnement, d'avoir des réactions rapides et de s'y adapter. ACT* (Anderson, 1983) et SOAR (Laird, Newell & Rosebloom, 1987) sont deux des premiers systèmes conçus comme des architectures d'unification de plusieurs manifestations de la cognition humaine. Partant de ces travaux, un effort particulier a été fait ces dernières années pour proposer des architectures plus spécialisées au cas où l'agent rationnel doit évoluer dans des environnements complexes et dynamiques (Georgeff & Rao, 1995; Hayes-Roth, 1995a; Hayes-Roth, 1995b; Russel, 1995).

Avant de construire un agent on doit étudier quatre éléments : les actions possibles, les données de perception possibles, les buts à atteindre et le type d'environnement dans lequel l'agent va opérer. Le tableau 1.1.1 montre quelques exemples d'agents.

1.1.2. Environnement

L'environnement où agit l'agent peut être classé selon cinq caractéristiques principales (Russel & Norvig, 1995).

- *Accessible ou inaccessible* : selon que les capteurs peuvent ou ne peuvent pas détecter tous les aspects de l'environnement;

- *Déterministe ou non-déterministe* : selon que le prochain état de l'environnement peut ou ne peut pas être déterminé à partir de l'état courant et de l'action sélectionnée par l'agent;
- *Épisodique ou non-épisodique* : l'environnement est épisodique s'il peut être divisé en "épisodes" et si les actions d'un agent dans l'épisode courant n'ont pas de conséquence dans les épisodes à venir;
- *Statique ou dynamique* : selon que l'environnement peut ou ne peut pas changer pendant que l'agent est en train de délibérer;
- *Discret ou continu* : selon qu'il existe un nombre réduit ou trop grand de données perceptives et d'actions.

Agent	Données perceptives	Actions	Buts	Environnement
Système de diagnostic médical	symptômes, résultats des examens, réponses du patient	poser des questions, demander des examens, prescrire des traitements	assurer la santé du patient, minimiser le coût pour la sécurité sociale	hôpital ou cabinet, patient
Système d'analyse d'images satellite	pixels avec différentes couleurs	imprimer ou afficher la catégorisation de la scène	trouver les catégorisations correctes	images de satellite
Système conducteur de taxi	images et sons de la route, mesure de vitesse	conduire, accélérer, ralentir, parler au passager	garantir sécurité, rapidité, confort, profits	routes, signaux, gens, voitures, immeubles, etc.
Tuteur interactif de la langue française	mots tapés au clavier d'un ordinateur	afficher exercices, suggestions, et corrections	améliorer les connaissances des élèves en langue française	ensemble d'élèves

Tableau 1.1.1 - Exemples de types d'agents et de leurs caractérisations — d'après (Russel & Norvig, 1995).

Dans le cas du système conducteur de taxi (Cf. tableau 1.1.1), l'environnement est non-accessible, non-déterministe, non-épisodique, dynamique et continu. Tandis que l'environnement du système d'analyse d'images satellite est accessible, déterministe, épisodique et discret. Ce qu'il faut retenir est que la difficulté de la conception d'un agent dépend des caractéristiques de l'environnement, l'architecture interne de l'agent n'étant qu'un reflet de la complexité de l'environnement (Simon, 1981) [pp. 8-17]. A cet égard il est plus facile de construire un système d'analyse d'images satellites que de conduite de taxi.

1.2. L'AGENT JAZZMAN

Nous avons présenté brièvement la notion d'agent rationnel en tant que cadre général pour la construction de programmes en IA. Voyons maintenant plus en détail la problématique spécifique de la construction d'un agent jazzman.

En simplifiant, la tâche d'un jazzman consiste à jouer sur un instrument donné des séquences de notes ou de rythmes, en fonction de ce qui est indiqué sur une grille d'accords et de ce que les autres musiciens jouent (Cf. tableau 1.2.1).

Agent	Données perceptives	Actions	Buts	Environnement
Système d'improvisation en jazz	grille d'accords, réactions du public, ce qui a été joué par tous jusqu'alors	jouer des notes et d'autres effets sonores	faire des improvisations intéressantes	scène
Système de composition musicale tonale	partition écrite jusqu'alors et notes jouées par le piano	écrire des notes sur une partition avec quelques pistes d'interprétation	composer un morceau intéressant	salle avec piano, crayon, gomme, partitions, etc.

Tableau 1.2.1 - Caractérisation d'un agent jazz et d'un agent compositeur

Mais plus précisément, que veut dire une grille d'accords ? Comment se passe l'interaction entre le jazzman et son environnement ? Quelles sont exactement les caractéristiques de la tâche ? Dans la suite de cette section nous précisons ces points.

1.2.1. Données perceptives

La *grille* est le premier élément de l'environnement qu'un jazzman perçoit. C'est la grille qui porte l'information du morceau sous la forme d'une séquence d'accords chiffrés représentant la trame harmonique (Cf. figure 1.2.1). Les accords sont écrits dans des "boîtes" correspondant chacune à une mesure, s'il y a plus d'un accord par mesure alors la "boîte" est divisée. La grille se lit de gauche à droite et du haut vers le bas. Normalement, avant de commencer à jouer, les musiciens choisissent la grille d'un morceau et se mettent d'accord sur le tempo et sur le *déroulement* de la grille (nombre de chorus, etc. — Cf. figure 1.2.4) et, éventuellement, sur quelques conventions rythmiques ou mélodiques. L'avantage de cette grille est que, à travers une notation concise, elle permet aux musiciens accompagnateurs de jouer même s'ils ne connaissent pas le thème. Néanmoins, quand les caractéristiques du thème sont importantes, on adopte une notation plus riche où le thème est affiché au-dessous des accords composant la grille, comme dans les *Real-Fake Books* (Sher, 1991). La grille est ce qu'il

Il y a de plus important pour les musiciens car, en indiquant les accords, elle indique implicitement comment les notes vont sonner, conférant ainsi une certaine sémantique à ces dernières. Toutefois, l'information de la grille est insuffisante pour déterminer ce qu'un jazzman va jouer mais elle donne des bonnes pistes.

E min7(b5)	A 7		C min 7	F 7	
F min 7	Bb 7		Eb maj7	Ab 7	
Bb maj7	Emin7(b5)	A7	D min7	G min7	C7
F maj7	G min7	C7	Amin7(b5)	D 7	
G 7	G7		C min7	C min7	
Eb min7	Eb min7	Ab 7	Bb maj7	Bb maj7	
E min7(b5)	A 7		D min7(b5)	G 7	
C min7(b5)	F 7		Bb maj7	Bb maj7	

Figure 1.2.1 - Grille de *STELLA BY STARLIGHT*

Pour mieux comprendre la tâche d'un agent jazzman et les conditions de son déroulement, il convient de faire quelques comparaisons entre l'improvisation et la composition en musique tonale. En effet, l'improvisation est un type de composition éphémère, où le compositeur est aussi instrumentiste; mais les aspects qui rapprochent ces deux activités musicales sont aussi nombreux que ceux qui les séparent. La grille illustre une première différence avec la composition. Ce dernier a beaucoup plus de liberté de création car, pour l'écriture d'une pièce, il doit définir la forme, la trame harmonique, la mélodie et ses contrepoints et accompagnements, le nombre et les types d'instruments, etc. En jazz, à l'exception du *free jazz*, on se base sur la grille ne s'occupant pratiquement plus ni de la forme ni de l'harmonie. En outre, la structure des morceaux est assez figée : elle est composée de phrases de quatre mesures, ce que l'on nomme la *carrure*. Il reste énormément de terrain aux jazzmen, ce dont témoignent les nombreuses versions (*takes*) d'improvisations sur un même morceau¹⁰, toutefois leur liberté est plus limitée que celle des compositeurs.

¹⁰Par exemple, dans le célèbre album "Now's the Time", les 13 titres correspondent seulement à 8 morceaux différents (Parker, 1957). Aussi, dans un *remastering* de "Tijuana Moods", à l'exception d'un morceau, tous les autres ont deux versions (Mingus, 1987).

Une deuxième donnée perceptive est simplement ce que le jazzman lui-même joue. Cette information prend plus d'importance en improvisation qu'en composition parce que les idées sont exposées dans un flux continu dans le temps. Pour garantir la cohérence de ce discours musical, le jazzman s'écoute, établissant ainsi une rétroaction.

Une troisième donnée perceptive importante vient des autres musiciens. Contrairement au compositeur, l'acte de création du jazzman est social. Il ne suffit pas de bien jouer sa partie mais de le faire par rapport aux autres musiciens, surtout pour les musiciens de la section rythmique. En outre, il y a aussi une certaine "complicité" qui s'établit entre les jazzmen et le public. Une bonne coordination entre les musiciens est difficile à atteindre. C'est pourquoi les musiciens de jazz se donnent la peine de répéter ensemble pendant des journées et des journées. Cette influence mutuelle des musiciens prend parfois la forme de questions-réponses comme les "trading four" (Walker, 1994), ou de dialogues plus subtils, courts et moins structurés. Toutefois, la plupart du temps il est quasiment impossible d'établir un lien clair de causalité entre les choix d'un musicien et ce que les autres musiciens jouent. L'influence se fait plutôt sentir dans l'ambiance musicale générale et même au-delà, dans l'état d'esprit et le plaisir de jouer ensemble, comme en témoignent leurs regards. A ce propos Bill Evans, dans la notice de *KIND OF BLUE*, un des albums de jazz les plus vendus au monde, écrit:

Group improvisation is a further challenge. Aside from the weighty technical problem of collective coherent thinking, there is the very human, even social need for sympathy from all members to bend for the common result. This most difficult problem, I think, is beautifully met and solved on this recording - (Davis, 1962)

En jazz, même la création qui déborde l'instant de la scène peut être collective. L'exemple historique le plus frappant est celui de la naissance du be-bop à partir des rencontres de Dizzy Gillespie, Charlie Christian, Thelonious Monk et Charlie Parker au *Minton's Playhouse*.

1.2.2. Environnement

L'environnement d'un jazzman peut être caractérisé comme étant accessible (audible et visible), non-déterministe (car le jazzman ne sait pas ce que vont jouer les autres, ni comment le public va réagir), non-épisodique (car tout ce que l'on joue a une influence sur la suite), dynamique et continu avec des contraintes de temps importantes. A part le

fait qu'il soit accessible, cet environnement a toutes les caractéristiques des environnements les plus complexes.

Parmi ces caractéristiques, le déroulement en direct de l'improvisation mérite une réflexion plus approfondie. Cette caractéristique majeure de l'improvisation nous dévoile énormément de choses sur l'activité des jazzmen, leur pensée, leur apprentissage, leurs connaissances, etc. Du fait du jeu en direct, la première conséquence pour le jazzman est qu'il ne peut revenir en arrière. S'il commet des fautes, soit il continue simplement, soit il essaye de réparer dans la suite en tirant profit de la faute. La spontanéité typique du jazz réside en partie dans l'impossibilité de changer le passé. La seconde conséquence est que le jazzman n'a pas suffisamment de temps pour réfléchir quand il joue. Il ne peut se permettre de chercher "la meilleure idée", mais il acceptera plutôt la première bonne idée qui lui passe par la tête (ou par les doigts !). Par exemple, un bassiste jouant à la noire dans un tempo pas trop rapide de 240 noires par minute doit décider quelle note jouer à chaque quart de seconde. Ce qui est le plus étonnant est que, même dans des be-bops extrêmement rapides (83 ms environ par note), on peut toujours identifier la pensée créatrice volontaire des bons solistes. Il suffit d'écouter les solos de Charlie Parker et de Dizzy Gillespie dans KOKO (Parker, 1947) joué à plus de 350 la noire.

Vu ces contraintes de temps, on comprend le rôle de la grille qui, en figeant la forme et l'harmonie de base, restreint le degré de liberté du jazzman pour qu'il puisse se concentrer sur des questions mélodiques, stylistiques, sonores, etc. (Sloboda, 1985) [p.149].

Le compositeur, au contraire, peut prendre tout son temps et peut toujours changer d'avis. Il procède normalement en deux étapes: "inspiration et exécution" (Sloboda, 1985) [pp. 115-22]. D'abord, à partir d'une idée de base, il établit quelques esquisses de l'œuvre. Ensuite, il commence le travail plus conscient où il détaille note par note l'œuvre finale.

1.2.3. Le fonctionnement global

Les données de perception, les capteurs, et les effecteurs d'un agent jazzman sont présentés dans la figure 1.2.2. Le module de perception reçoit, à travers des

microphones, les données sonores venues des autres musiciens, du public et de soi-même. Il reçoit aussi, à travers des caméras des données visuelles concernant la grille, les autres musiciens et le public. Le module de raisonnement doit calculer les notes qui doivent être jouées et les transmettre au module d'exécution, qui, à son tour, envoie au synthétiseur chacune de ces notes au moment opportun.

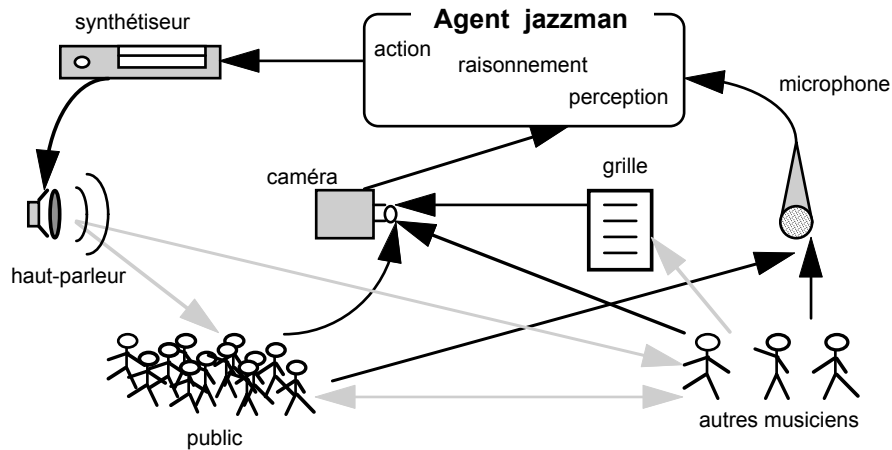


Figure 1.2.2 - Agent jazzman artificiel et son environnement

Dans le cas des agents rationnels évoluant dans des environnements dynamiques sous contraintes de temps, les trois modules de base travaillent en parallèle et établissent des communications entre eux de façon asynchrone. Le module de raisonnement (appelé aussi module de planification) planifie en avance un certain nombre de notes (une ou plusieurs) qui doivent être jouées par le module d'exécution le moment venu. Dans la figure 1.2.3, les musiciens et le module d'exécution de l'agent sont en train de jouer les dernières notes de la deuxième mesure, tandis que le module de raisonnement commence à planifier les notes qui devront être jouées quand l'exécution arrivera à la quatrième mesure. Ce schéma est tout à fait classique en ce qui concerne un agent qui évolue en temps réel dans un environnement dynamique (Ambros-Ingerson & Steel, 1988; Georgeff & Lansky, 1987).

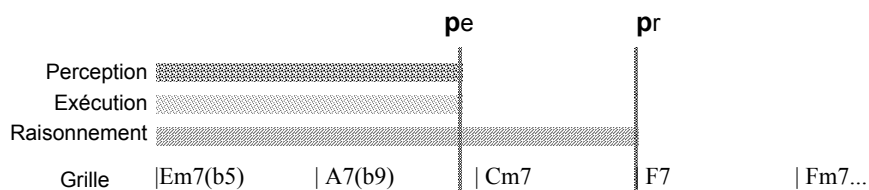


Figure 1.2.3 - Position des modules sur la grille

Nous appelons *écart raisonnement-exécution relatif* (ERE), l'écart existant, à un instant donné i , entre la position sur la grille du module de raisonnement (p_r) et celle du module d'exécution (p_e).

$$ERE(i) = p_{ri} - p_{ei} \quad (\text{unité de temps}) \quad (1.2.1)$$

L'ERE est donné en unité de temps (beat) puisque les positions sur la grille sont données en *unités de temps* (ou simplement *temps*). Néanmoins, l'ERE peut être exprimé en secondes en tenant compte du tempo. Le tempo étant fourni en nombre de temps (pulsations) par minute, nous pouvons donc définir l'*écart raisonnement-exécution absolu* (ERE^*) comme:

$$ERE^*(i) = \left(\frac{ERE_i}{tempo} \right) * 60 \quad (\text{secondes}) \quad (1.2.2)$$

Il est important d'expliciter l' ERE^* car il représente l'écart en temps réel entre les modules. Autrement dit, l' ERE^* est le temps dont le module de raisonnement dispose pour décider quoi jouer avant que l'exécution n'arrive à la position p_r sur la grille. D'une part, si ERE^* est trop petit, un raisonnement complexe demandant beaucoup de calcul ne peut être mené. D'autre part, s'il est trop grand, les événements plus récents ne pourront pas être pris en compte pendant le raisonnement, ce qui est gênant. Le compromis à trouver est d'autant plus difficile que le tempo est rapide. Dans un tempo lent, la situation est moins critique puisque l'on a suffisamment de temps pour raisonner (ERE^*), même pour un ERE petit.

1.2.4. Une formalisation de la tâche

La grille détermine un *chorus* (par ex. le chorus de STELLA BY STARLIGHT a 32 mesures) mais le déroulement du morceau se fait en plusieurs chorus. La figure 1.2.4 schématise ce déroulement pour un quartette de jazz typique : une section rythmique composée par le pianiste, le batteur et le bassiste et un saxophoniste (le soliste). Le soliste expose le thème dans le premier chorus. Ensuite, pendant plusieurs chorus intermédiaires le soliste improvise véritablement. Il arrive fréquemment que les accompagnateurs, à tour de rôle, deviennent solistes pendant quelques-uns de ces chorus intermédiaires. Enfin, le soliste expose de nouveau le thème.

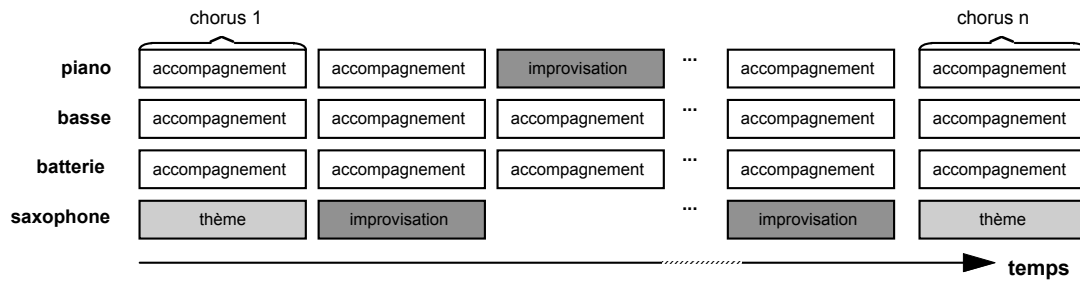


Figure 1.2.4 - Déroulement du morceau (les choruses)

Nous pouvons maintenant donner une première formalisation de la tâche d'un agent jazzman (accompagnateur ou soliste). On définit une note comme étant le n-tuplet suivant.

$$n = (\text{début}, \text{durée}, \text{hauteur}, \text{amplitude}, \text{timbre}) \quad (1.2.3)$$

L'information de timbre est nécessaire au contrôle du synthétiseur (Cf. figure 1.2.2). Nous considérons le silence comme une note ayant une amplitude égale à zéro et une hauteur et un timbre indéfinis.

Un *segment d'improvisation* $I(t,D)$ de durée D et de début t (en temps relatif ou absolu)¹¹, est un ensemble de notes ainsi défini :

$$I(t,D) = \{n_1, n_2, \dots, n_k, \dots, n_z\} \quad (1.2.4)$$

$$\text{tel que } \sum_{j=1, \dots, z} \text{durée}(n_j) = D,$$

$$\forall_i [\text{début}(n_{i+1}) = (\text{début}(n_i) + \text{durée}(n_i))] \text{ et}$$

$$\text{début}(n_1) = t$$

Une *improvisation* de durée D et début t peut alors être définie comme l'union de plusieurs segments d'improvisation :

$$I(t,D) = I(t_1, D_1) \cup I(t_2, D_2) \cup \dots \cup I(t_k, D_k) \cup \dots \cup I(t_z, D_z) \quad (1.2.5)$$

$$\text{tel que } \sum_{j=1, \dots, z} D_j = D, \quad \forall_i (t_{i+1} = t_i + D_i) \text{ et } t_1 = t$$

La tâche d'un agent jazzman est de planifier le prochain segment d'improvisation. Supposant que tous les musiciens ont commencé ensemble au temps t et que le module d'exécution de l'agent est sur la position p_e , à une durée D_e du début

¹¹En connaissant le tempo, on peut toujours calculer le temps relatif à partir du temps absolu et vice-versa

de l'improvisation ($p_e = t + D_e$), la détermination du prochain segment d'improvisation se fait en fonction de données suivantes :

- d'une grille G fournie au départ, représentant l'harmonie, et dont la durée totale du déroulement de tous les chorus détermine la longueur D de l'improvisation;
- les segments planifiés jusqu'à présent $I(t, D_e + ERE) = \{n_1, n_2, \dots, n_k\}$ par le module de raisonnement;
- des improvisations ou accompagnements $\{I(t, D_e)\}^x = (I^1(t, D_e), I^2(t, D_e), \dots, I^x(t, D_e))$ des autres x musiciens;
- des réactions du public $PUB((t, D_e))$

On appelle *correspondance perception-action* d'un agent la fonction qui définit les actions à prendre en réponse à ces données perceptives (Russel & Norvig, 1995). La correspondance perception-action de l'agent jazzman est formellement définie par la fonction *prochain segment* PS suivante:

$$I_{k+1} = PS(I(t, D_e + ERE), G, \{I(t, D_e)\}_x, PUB(t, D_e)) \quad (1.2.6)$$

En réalité, le public et les autres musiciens peuvent ne pas être présents. Un jazzman tout seul dans son appartement, par exemple, peut improviser. Un jazzman peut aussi jouer sans une grille, mémorisée ou affichée. Ainsi, dans le *free jazz*, bien que l'on suive un certain "parcours", il n'y a pas vraiment de grille. Toutefois, nous gardons la formulation la plus courante de l'improvisation et de l'accompagnement.

1.2.5. Quelques problèmes posés

L'intérêt de la formulation que nous venons de proposer est de permettre de poser, de façon plus précise, quelques problèmes sur la construction d'un agent jazzman indépendamment de l'approche choisie.

La première difficulté posée est la détermination de la *granularité* du raisonnement de l'agent, c'est-à-dire de la taille du segment d'improvisation. Combien de notes l'agent doit-il calculer : une note, les notes d'une mesure, les notes d'un accord ? Les segments doivent-ils être de taille fixe ou variable ? Le choix du segment est très délicat car il est lié au problème classique du contrôle entre le niveau local et le niveau global. Un segment trop court rend plus difficile la continuité globale de

l'improvisation, tandis qu'un segment trop grand implique un écart (ERE) trop important entre le raisonnement et la perception.

La deuxième difficulté est qu'il n'y a pas un calcul direct entre les paramètres de la fonction PN et l'improvisation à être créé. Un jazzman manifeste ses idées en respectant la grille et le tempo, et ayant une complicité avec les autres musiciens et le public, mais on n'en sait pas plus. Chacun de ces paramètres est un simple point de repère qui peut, éventuellement, suggérer telle ou telle action mais jamais déterminer quoi que se soit. En outre, on ignore la façon dont ces paramètres s'entremêlent. Par exemple, quelle importance doit prêter un agent jazzman à ce que les autres musiciens ont fait par rapport à la cohérence de son propre discours ? Qu'est-ce qui est le plus important : les accords courants de la grille ou ce que les autres musiciens sont en train de jouer ?

Une autre difficulté est qu'une correspondance perception-action idéale, où pour chaque donnée perceptive on peut spécifier une action à prendre, ne peut être établie à cause des caractéristiques de l'environnement. Un agent jazzman, comme d'ailleurs la plupart des agents, ne peut que rarement prendre la décision optimale parce qu'il n'a pas toutes les ressources de calcul pour le faire compte tenu de la complexité du problème et des contraintes de temps. Dans ces cas, plutôt qu'une "rationalité idéale", l'agent a une *rationalité limitée*; plutôt que chercher une solution optimale, il cherche une solution satisfaisante compte tenu de ses ressources de calcul et de la complexité de l'environnement (Simon, 1981) [pp. 28-36]. Par conséquent, il faut qu'il y ait un "filtrage" de l'information représentée par les paramètres de la fonction PN. De tout ce qu'un agent a écouté depuis le début de l'improvisation, qu'est-ce qui est pertinent pour l'action actuelle ? Les informations plus récentes ont-elles la même importance que celles plus lointaines dans le temps ? De même, qu'est-ce qu'il est pertinent de regarder dans le futur : la partie de la grille qui va encore se dérouler immédiatement ou toute la grille restante ?

Il y a enfin la difficulté de mesurer le succès d'un agent jazzman : comment savoir quand l'agent a bien joué ? Cette question se pose aussi bien pour le concepteur que pour l'agent lui-même car il faut qu'il puisse porter un jugement sur ce qu'il est en train de faire afin de décider s'il faut persister ou changer son comportement.

Dans la suite nous présentons des éléments de réponses à quelques-uns de ces problèmes en analysant, d'une part, les connaissances utilisées par les jazzmen (Chapitre 2) et, d'autre part, le fonctionnement des programmes existants (Chapitre 3).



2. CONNAISSANCES ET RAISONNEMENT CHEZ LES JAZZMEN

Comment les jazzmen improvisent-ils ? Quelles sont leurs connaissances ? Comment apprennent-ils ce métier ? Ce sont des questions difficiles pour lesquelles on n'a pas de réponse précise. Les psychologues, à peu d'exceptions près (Pressing, 1988; Sloboda, 1985), se concentrent surtout sur les problèmes de perception ne proposant, pour l'instant, aucun modèle suffisamment détaillé ou rigoureux de l'improvisation. Nous avons essayé de trouver des réponses à ces questions en privilégiant une démarche d'acquisition de connaissances auprès des experts musiciens. Le contact avec ces derniers n'a pas été facile parce qu'ils n'ont pas toujours une grande introspection sur leurs activités. Le travail le plus fructueux a été fait avec des enseignants de jazz, en particulier avec Heriberto Paredes de l'IACP.

Dans ce chapitre nous présentons un résumé des résultats de ce travail d'acquisition de connaissances. Par ailleurs, nous ajoutons quelques réflexions issues de l'étude de quelques œuvres de pédagogues et musicologues du jazz et des psychologues. Nous énumérons quelques-unes des principales connaissances acquises par les musiciens au long de leurs années de pratique. Soulignons que l'ordre dans lequel les connaissances sont présentées dans cette section ne correspond pas forcément à celui de l'apprentissage des musiciens, celui-ci se faisant plutôt en cycles où les connaissances sont consolidées, approfondies, remises en cause, personnalisées, etc. Les connaissances sont énumérées selon la façon dont un agent rationnel agit et perçoit l'environnement, décide de l'action à réaliser et fait évoluer ses connaissances.

2.1. HABILITE INSTRUMENTALE (*SKILL*)

Le jazzman est un instrumentiste et, comme tout instrumentiste, il doit avoir une maîtrise raisonnable de son instrument. Ceci signifie qu'il faut qu'il soit précis et rapide pour pouvoir jouer les notes correctement (hauteurs, intensités, placement dans le temps) en respectant le tempo. Pour un musicien, il est important d'acquérir un automatisme des membres du corps pour qu'il puisse employer ses ressources cognitives pour des aspects harmoniques, mélodiques, stylistiques, etc.

En outre, l'habileté instrumentale est très importante en jazz car la façon de jouer, de produire le son, est un facteur de création de même valeur que l'invention mélodique. Selon André Hodeir, compositeur et musicologue français, dont l'ouvrage cité est un des chefs-d'œuvres sur le jazz, "l'improvisateur de jazz ne crée qu'en fonction de l'instrument dont il joue" (Hodeir, 1981) [p. 143]. Le jazzman, en connaissant profondément son instrument, explore dans sa pensée créatrice toutes les possibilités. Ainsi, ce que l'on appelle *l'expressionnisme hot*, qui consiste à "faire chauffer" par l'utilisation de techniques comme le vibrato, les inflexions, les harmoniques, etc., est une des caractéristiques essentielles du jazz (ibidem) [pp. 204-12], de même, certains jazzmen, comme Miles Davis, sont reconnaissables dès les premières notes par leur timbre caractéristique.

L'architecture de l'instrument joue à cet égard un rôle non négligeable puisqu'elle influence le jazzman imposant des contraintes de doigté et de souffle. Certains choix, et certains "non-choix", trouvent leur explication dans ces contraintes. On n'a pas la même commodité pour jouer une phrase dans différentes tonalités et différents tempi. Ces limitations anatomiques contribuent indirectement à l'affirmation d'un style. Le travail du musicologue américain Owens, qui a catalogué les principaux patterns mélodiques utilisés par Charlie Parker, illustre bien ces propos. Chaque chapitre de sa thèse est concerné par les patterns d'une tonalité donnée car ils en dépendent. Alors, il n'y a rien de surprenant que Charlie Parker, qui jouait l'alto, ait préféré les tonalités de C, Bb et Eb.

2.2. ÉCOUTE

Certes, le jazzman perçoit le monde par les yeux car il voit entre autres les clins d'œil, les sourires et la désapprobation du public et des autres musiciens sur scène. Mais, c'est par l'oreille que la plupart de l'information sur son environnement lui parvient. Cette section se concentre sur cette dimension d'écoute. La première partie se penche sur le rôle de l'écoute dans la formation des jazzmen, tandis que les autres étudient son rôle pendant le déroulement du morceau.

2.2.1. Dans l'apprentissage

L'apprentissage du jazz est essentiellement basé sur l'écoute et la pratique, ce qui lui confère son caractère empirique. Hodeir, en se référant au rôle des disques, dit: "Les voies de l'enregistrement sont la seule approche objective du jazz, cette musique qui n'est nulle part et qu'on ne peut interroger en scrutant des partitions" [ibidem, p. 10]. Les jazzmen en général ont une base solide en théorie musicale, mais tous les grands pédagogues du jazz ainsi que les experts que nous avons rencontrés sont unanimes pour souligner l'approche essentiellement empirique du jazz. La méthode d'apprentissage IAI (imitation, assimilation et innovation) (Sabatella, 1996), majoritaire en jazz, exemplifie ce propos. On écoute, analyse et imite les grands noms du jazz pour ensuite développer son propre style (Baker, 1980; Coker, 1970; Sabatella, 1996; Sher, 1979). Cela met aussi en relief le rôle des exemples comme le matériau de base sur lequel, à travers l'expérimentation, le jazzman comprend comment accomplir sa tâche. Les jazzmen savent que c'est à travers les exemples qu'ils saisissent les concepts essentiels du jazz, plutôt que par des définitions. Le concept de swing en est l'évidence la plus frappante. En effet, le swing qui est un des éléments le plus caractéristique du jazz et qui le distingue de la musique tonale européenne (Hodeir, 1981) [pp. 179-91]. Malgré son importance, personne n'est capable d'en donner une définition précise.

2.2.2. Structures et multiples traits

Comment le jazzman écoute son environnement quand il est en train de jouer ? Le premier aspect de l'écoute concerne l'interprétation du son d'autres instruments et du sien en entités musicales discrètes, les notes dans la plupart.

Le deuxième aspect tient au fait que les musiciens, tous les styles confondus, ne perçoivent pas un morceau musical comme une séquence de notes non corrélées, mais plutôt comme des structures rythmiques et mélodiques. En fait, inconsciemment, ils segmentent un morceau en entités de granularité plus grande : des amas de notes (phrases, motifs, accords, etc.). Les psychologues ont obtenu des théories de pouvoir explicatif important sur ce sujet (Lerhdal & Jackendoff, 1983; Narmour, 1977; Narmour, 1989). On segmente une mélodie selon des principes de continuité et de changement rythmique ou mélodique (figure 2.2.1). Ce n'est qu'ayant ces amas de notes comme entité de base, que l'on peut comprendre et exprimer tout l'éventail de fonctions harmoniques et mélodiques comme l'attente, la résolution, la variation, la cohérence, etc. Ces amas de notes (phrases) ont une structure de caractère hiérarchique, bien que non parfaitement car il est souvent difficile d'établir des frontières absolues.

Les phrases et les relations entre elles comme : succession, concurrence et surtout répétition et variation sont un premier facteur très important de structuration musicale. Les relations partie au tout (*part of*) et contenance entre les composants d'un morceau en sont un deuxième (West, Howell & Cross, 1991). Par exemple, les morceaux en jazz ont une structure bien définie et hiérarchisée. La grande majorité des morceaux ont soit 32 mesures (les *standards*) soit 12 mesures (les *blues*) qui se composent généralement de phrases de quatre mesures (la carrure). Dans les *standards*, ces phrases s'associent deux à deux pour former les sections, un morceau ayant quatre sections (Cf. figure 2.3.1). La structuration la plus courante des sections est AABA, auquel cas, la section B est aussi appelée "le pont" (*bridge*). Mais il existe aussi ABA'B, ABAB' et ABCD (Baudoin, 1990) [vol. 1, pp. 13-14]. Par ailleurs, comme nous avons déjà expliqué (Cf. section 1.2.1), il existe aussi une structuration plus large qui est la division du morceau en chorus (nombre de grilles), chacun ayant une fonction : exposition du thème ou improvisation. Toute cette structure représente des points de repère très important pour l'auditeur et pour l'improvisateur. En réalité, on utilise toutes les structures dont nous avons parlé aussi bien pour analyser une mélodie existante que pour organiser le discours musical pendant la création.

Un groupe de notes peut être perçu selon plusieurs traits, points de vue, propriétés, concepts, etc. (nous retiendrons le mot *trait* pour la suite de ce document). Ces traits sont un "vocabulaire d'écoute" qui permet d'étendre la perception des notes

vers des niveaux plus abstraits que simplement les hauteurs, les durées et les intensités (ce que l'on appelle "le niveau de notes"). Voici quelques exemples de traits : tonalité, contour mélodique, gamme, *voicing*, tessiture, structure rythmique, accentuation, tension, dynamique, *syncopité*, densité, équilibre, température, contraste, style, sonorité, etc. (Cf. figure 2.2.1). Bien entendu, la plupart des traits n'ont de sens que sur une granularité plus grande qu'une note. On ne peut parler de contour mélodique sur une ou deux notes, par exemple. Des traits musicaux comme équilibre, température, développement concernent au moins un chorus. Ce sont ces traits qui, complétant l'information au niveau des notes, permettent à l'auditeur d'appréhender toute la richesse d'une improvisation, toute sa trame harmonique et mélodique, bref, les intentions de l'improvisateur. Ces traits musicaux sont entendus inconsciemment, toutefois, plus l'auditeur a des connaissances musicales, plus ce vocabulaire est riche. Chaque musicien a bien sûr son propre répertoire de traits et sa propre façon de les percevoir.

2.2.3. Anticipation et évaluation dynamique

À la vitesse à la quelle on joue, il est souvent très compliqué d'établir une bonne interaction entre les jazzmen, surtout quand il s'agit des interactions impromptues ou moins structurées que les "questions-réponses". Par exemple, quand un soliste "laisse des petits creux" pour que la section rythmique se manifeste, si la réponse n'est pas rapide l'effet est raté. Pour s'en sortir les musiciens anticipent, plutôt inconsciemment, les uns sur les actions des autres. Ils font des suppositions à court terme sur les actions à venir de type "je parie qu'il va laisser de nouveau ce petit creux dans 4 temps" ou "je crois qu'il va continuer à jouer de plus en plus haut". Plus on connaît un musicien et son style, plus l'anticipation est judicieuse. Les répétitions sont des moments privilégiés pour acquérir ce type de connaissance.

Un des experts qui ont collaboré avec nous, Heriberto Paredes, en parlant des plans d'improvisation, a dit : "il n'y a que les débutants qui se font un plan précis avant de commencer à improviser. Pour les bons musiciens, le plan ne se dégage qu'en pleine improvisation". Cela signifie simplement que les jazzmen s'écoutent et jugent continuellement ce qu'ils viennent de jouer. Ce n'est qu'ainsi qu'ils sont en mesure de décider de continuer à développer une idée, d'en chercher une autre, de faire plus

attention à tel ou tel aspect, etc. Par exemple, c'est ainsi que le jazzman peut se rendre compte qu'une erreur s'avère intéressante et mérite être exploitée; ou qu'une idée bien approfondie n'a pas bien sonné au bout du compte. La créativité et la cohérence d'une improvisation ou d'un accompagnement ne peuvent avoir lieu sans ces connaissances d'évaluation dynamique.

Figure 2.2.1 - Fragment de ligne de basse de Ron Carter au début de *STELLA BY STARLIGHT* (Aabersold, 1979) dont la description a été enrichie par la mise en relief des groupements et de quelques traits musicaux

2.3. FONDEMENTS EN THEORIE MUSICALE

La théorie musicale n'est pas à même d'expliquer comment un compositeur ou un improvisateur crée (Laske, 1989). Néanmoins, la théorie musicale est un outil clé car elle définit les concepts musicaux fondamentaux et dévoile les possibilités de création. Si on fait un parallèle avec le langage, la théorie musicale établit le lexique, les catégories

syntaxique et la grammaire, sans lesquels toute création littéraire serait difficilement concevable mais qui n'expliquent pas le processus de création.

Les concepts de base qu'un musicien, de jazz ou pas, doit connaître sont très nombreux. Ce sont des connaissances relativement complexes mais très bien définies et documentées et qui font objet d'une quasi-unanimité. Il existe des concepts autour de la notion de note : durée, hauteur, intensité, altération, tessiture, etc. D'autres concernant un ensemble de notes : intervalle, accords, gamme, mode, tonalité, inversion d'accord, fonctions des accords, temps, tempo, mesure, rythme, syncope, mélodie, modulation, dynamique, cadence, contrepoint, etc. Dans cette section, nous nous restreignons à un petit nombre de ces concepts en insistant sur comment ces connaissances sont utilisées par un jazzman pour interpréter son environnement musical (la grille et ce que ses partenaires et lui-même jouent).

2.3.1. La grille d'accords

Contrairement à ce que l'on pourrait penser, un jazzman voit et entend beaucoup de choses dans une grille d'accords. De fait, les connaissances musicales du jazzman lui permettent d'analyser la grille pour en tirer beaucoup d'informations utiles pour l'improvisation ou l'accompagnement. On peut grouper ces connaissances en trois aspects.

D'abord, il y a l'analyse de la grille comme un tout, de sa structure. Comme présenté, dans la section précédente, la grille a une structure bien définie : mesure, carrure, section, chorus, etc. Le jazzman a besoin d'avoir conscience de la structure pour organiser son discours, pour anticiper sur ce que les autres musiciens vont jouer, pour la souligner pour le public et les autres musiciens, etc. Il sait par exemple, que "le pont" est un bon endroit pour faire des changements dans la structure rythmique (*two-beat*, *four-beat*, etc.). Il sait aussi que les accords de fin de section (*turnaround*) et fin de chorus (*turnback*) sont particulièrement importants pour les accompagnateurs en ce qui concerne l'annonce de ce qui va venir.

Ensuite, chaque accord séparément fournit d'autres indications importantes. Il n'y a que trois choix pour les notes d'une improvisation ou d'un accompagnement : note de l'accord, note d'une gamme ou note chromatique. C'est vis-à-vis d'un accord

particulier que l'on peut choisir quelle technique employer. En effet, les connaissances de la théorie musicale permettent au jazzman d'analyser un accord pour savoir quelles sont les notes propres à l'accord, les possibles tonalités locales et par conséquent les gammes dont il peut se servir. Plus généralement, c'est l'accord et quelques notions de métrique qui donnent une certaine sémantique aux choix des notes. Par exemple, sur l'accord A7 (deuxième mesure du fragment présenté dans la figure 2.2.1), le Ré sonne plus dissonant que le La que lui précède car le La est une note de l'accord et le Ré ne l'est pas. De même, le Ré dont nous parlons paraîtra moins dissonant que le Sol dièse de la première mesure car, contrairement au Ré, le Sol dièse ne fait pas partir de la tonalité de Ré. Les choix de certaines notes peuvent aussi mettre en relief la fonction d'un accord. En jouant La et Mi bémol sur un accord F7 par exemple, cela crée une tension qui renforce sa fonction de dominante. Ce type de technique est particulièrement utilisé par le bassiste.

Enfin, il faut considérer les accords en groupe et les rapports entre eux. On ne joue pas les notes sur un accord comme si celui-ci était le seul, mais en préparant le passage adéquat vers l'accord suivant. Pour un bassiste, il est important de souligner les changements d'accords et pour cela il se sert de techniques comme l'utilisation des notes de rapprochement (*leading tones* — les notes Ré et Sol dièse dont nous avons parlé dans le paragraphe précédent en sont des exemples). En outre, l'information de l'accord suivant permet d'évaluer correctement la tonalité locale et la fonction de l'accord courant. Par exemple, si le F 7 est suivi de Bb, Bb min, C, G min ou G, alors probablement la tonalité est respectivement Si bémol majeur, Si bémol mineur, Do blues, Sol mineur ou Sol majeur (emprunt modal).

Si on va au-delà de deux accords, on constate que, de la même façon que les notes, les accords se regroupent pour former des entités sémantiquement plus riches et de granularité plus grande. Avant de commencer à jouer, les premières choses qu'un jazzman identifie par une brève analyse harmonique sont les schémas d'accords de type II-V, V-I, II-V-I, IV-III-II-V-I, VI-II-V-I, etc. Ces schémas sont abondamment catalogués (Baudoin, 1990) [vol. 1, pp. 57-98] et parfois ils ont même de noms (Christophe et Anatole, respectivement pour les deux dernières). En plus de préciser la tonalité, et donc les gammes pertinentes, ces schémas représentent des situations prototypes très utiles. Comme montre la figure 2.3.1, les sous-séquences "Emin7(b5)

A7”, “Dmin7(b5) G7” et “Cmin7(b5) F7” sont identiques, à des transpositions près, car elles sont de type II-V mineur. Alors, on peut construire des phrases musicales qui sonnent bien pour les II-V mineurs, et plus tard les réutiliser dans différentes tonalités. Ces schémas d’accords sont à la base de l’écoute et de l’analyse d’improvisations des grands maîtres du jazz. Aussi sont-ils utiles à construction des improvisations car ils structurent davantage la grille et en font ressortir des enchaînements typiques (Baker, 1980; Coker, 1970; Mehegan, 1984).

Section A	E min7(b5) II ————— V(Mj) A 7	C min 7 II ————— V(Mj) F 7		
	F min 7 II ————— V Bb 7	Eb maj7 I(Mj) Ab 7		
Section B	Bb maj7	Emin7(b5) II ————— V A 7	D min7 I(m)	G min7 II ————— V C 7
	F maj7 I(Mj)	G min7 II ————— V(Mj) C 7	Amin7(b5) II ————— V(m) D 7	
Section C	G 7 V ————— G7	C min7 I(m) C min7		
	Eb min7	Eb min7 II ————— V(Mj) Ab 7	Bb maj7	Bb maj7
Section D	E min7(b5) II ————— V(m) A 7	D min7(b5) II ————— V(m) G 7		
	C min7(b5) II ————— V(m) F 7	Bb maj7	Bb maj7	

Figure 2.3.1 - Grille de STELLA BY STARLIGHT dont la description a été enrichie par la mise en relief des sections et des schémas d’accords (Mj = majeur, m = mineur)

2.3.2. La construction même de l’improvisation

Voyons maintenant en quoi la théorie musicale est utile pour interpréter l’environnement, c’est-à-dire, ce que le jazzman et ses partenaires jouent.

La théorie musicale n’enseigne rien sur comment obtenir une interaction adéquate entre les musiciens, mais en donnant les concepts musicaux de base elle permet une écoute plus fine et ample.

Quant à l’interprétation que le jazzman fait de ce qu’il a joué pour la suite il existe quelques indications, quoiqu’elles soient plutôt du ressort de la “théorie musicale appliquée”. Pour la construction d’une ligne de basse, il y a par exemple quelques

techniques (Cf. figure 2.2.1) : l'arpège (*chordal-based*), où l'on joue uniquement les notes de l'accord; le déplacement en *stepwise* (tons voisins) où on avance en intervalles de secondes en faisant usage des notes de l'accord plus les notes de la gamme locale; et le chromatisme, un cas particulier de la technique précédente où on fait appel à la gamme chromatique. Par ailleurs, chacune de ces techniques peut avoir une structure rythmique différente : basée sur la blanche (*two-beat* ou *two-feel*) ou sur la noire (*four-beat*, *four-feel* ou encore *walking*). Selon la stratégie et le rythme adoptés les qualités harmoniques et mélodiques varieront. Il importe d'affirmer que jamais un bassiste n'emploie uniquement une de ces techniques pendant le morceau entier. Comme on peut le suspecter, il n'y a pas de règles pour le dosage. Cela fait partie en réalité d'un problème beaucoup plus général qui est celui du rapport mélodie-harmonie : comment construire une improvisation qui souligne les accords tout en ayant une cohérence interne en tant que mélodie (Hodeir, 1981) [pp. 136-7].

Il y a un certain nombre de suggestions concernant les contours mélodiques, pour assurer une bonne fluidité et continuité de l'improvisation ou l'accompagnement. Ainsi, la règle de contrepoint "il faut compenser un saut (plus grand qu'une quinte) par un changement de direction", connue depuis des siècles est toujours utile. D'autres existent comme "il est intéressant pour la basse de faire un *drop* après une montée douce" (Cf. deuxième mesure de la figure 2.2.1).

Il est important de faire maintenant des remarques sur la nature de ces règles ou techniques de construction de lignes de basse. Prenons une règle très connue : "Jouez la tonique au premier temps et après n'importe quelle note de l'accord". Cette règle pourtant très élémentaire marche toujours, indépendamment de la situation. Bien entendu, aucun bon bassiste va l'utiliser tout le temps. Cette "règle" est très révélatrice. D'une part, elle met évidence le fait que l'improvisation est un problème sous-contraint, où les solutions sont nombreuses et la difficulté est de savoir quelles sont les meilleures solutions à un moment donné. D'autre part, elle témoigne du caractère incomplet, incertain et contradictoire des règles d'improvisation. Incomplet car elle ne dit pas exactement quoi faire après la tonique (quelles notes et dans quel ordre et rythme). C'est souvent le cas pour le bassiste qui raisonne en termes de notes "cibles" (*landmarks*), en l'occurrence les toniques des accords, laissant le reste en second plan. Incertain car le contexte n'est pas défini et, toutefois, on sait qu'elle n'est pas toujours

appliquée. Enfin, contradictoire car il y a au moins une autre demi-douzaine des règles différentes qui peuvent être appliquées à la même situation. En jazz les règles, quand elles existent, sont plutôt des conseils, des heuristiques. D'ailleurs, même en musique classique c'est souvent comme ça (Thomas, 1985), (Sloboda, 1985) [p. 55]. Cela pose évidemment des problèmes de représentation car il faut pouvoir traiter les contradictions et le caractère partiel et heuristique des connaissances.

2.4. AU-DELA DU “NOTE PAR NOTE” ET DU “LA NOTE A CAUSE DE LA NOTE”

Au même titre que la perception et l'analyse, le processus de création se base sur des regroupements des notes et sur des traits musicaux associés à ces regroupements.

2.4.1. Patterns

Le mot improvisation, du latin *improvisus*, signifie imprévu, composée sur-le-champ et sans préparation. Mais en jazz, ce n'est pas tout à fait le cas. Il y a beaucoup de travail préalable pour soutenir le jazzman quand il est sur scène. Outre les connaissances que nous avons évoquées sur l'habileté instrumentale, l'écoute et la théorie musicale, les jazzmen réutilisent une collection de ce que l'on appelle les *patterns*, les formules ou les *licks*, c'est-à-dire des fragments rythmiques ou mélodiques préférés. La figure 2.4.1, montre quatre exemples de patterns mélodiques employés par Miles Davis sur les II-V majeurs. Rien que pour ce schéma d'accord Baker (Baker, 1980) en a sélectionné soixante-trois utilisés par Miles Davis.



Figure 2.4.1 - Cinq patterns mélodiques de Miles Davis sur |Dmin7 |G7 | (Baker, 1980).

La présence des patterns en tant qu'élément de la pensée des jazzmen confirme ce que nous avons remarqué sur le rôle structurel et des phrases et des schémas d'accords. Les patterns sont composés d'une ou plusieurs phrases mélodiques et sont souvent mémorisées selon des schémas d'accords puisque ça augmente la possibilité de réutilisation, compte tenu du degré de généralité de ces schémas.

Tout mène à croire que l'utilisation des patterns en jazz s'inscrit dans une longue tradition d'utilisation de "formules" dans certaines activités d'improvisation artistique qui remonte à la poésie d'Homère, en passant par le chant grégorien, les troubadours, etc. (Lord, 1964; Parry, 1930; Parry, 1932; Smith, 1983). Une formule serait, selon Pressing, un "*group of words regularly employed under the same metrical conditions to express a given essential idea; it has melodic, metric, syntactic and acoustic dimensions*" (Pressing, 1988) [p. 146]. Ce qu'il y a en commun entre toutes ces manifestations artistiques est le fait qu'elles se déroulent en direct. Faute de temps pour tout inventer, les formules viennent à la rescousse de l'improvisateur puisqu'étant connues en avance, elles sont immédiatement disponibles à la réutilisation.

La mémorisation d'une bonne collection des patterns fait partie de l'apprentissage des jazzmen. Soit les jazzmen les "piquent" à quelqu'un d'autre (ce qui est légitime puisque ça fait partie de la méthode IAI — imitation, assimilation et innovation), soit ils les repèrent dans des nombreux catalogues du genre comme celui de Coker (Coker, 1970), soit enfin ils se créent leurs propres patterns. On ne sait réellement pas comment ces nouveaux patterns sont créés. Un des volets de l'innovation est là. Quoiqu'il en soit, le fait est que les patterns sont intimement liés à l'expérience, au vécu d'un jazzman car pendant tout son parcours un jazzman ne cesse d'apprendre et de créer des nouveaux patterns, et surtout de les tester pour mieux comprendre quand y faire appel. Dans cette perspective, chaque jazzman (jouant un instrument monophonique) a sa collection personnelle de fragments mélodiques ou rythmiques préférés. C'est pourquoi il y a tout un courant de musicologues qui s'intéressent à les cataloguer, ainsi que ses variations et les circonstances où ils sont utilisés, comme une manière de saisir le style et la pensée d'un jazzman. Ainsi, on a identifié les patterns de Charlie Parker (Owens, 1974), John Coltrane (Kernfeld, 1983), Bill Evans (Smith, 1983), Miles Davis (Baker, 1980), Lester Young (Rottlieb, 1991) , entre autres.

Il faut signaler que la façon dont les musiciens et les musicologues utilisent le mot *pattern* ne veut pas forcément dire quelque chose d'abstrait ou de général. À part le fait d'être récurrent, un *pattern* mélodique peut être identique à n'importe quel autre fragment mélodique. En outre, certains *patterns* ne sont pas toujours "transposables", par exemple. Ils sont utilisés toujours dans une tonalité particulière, comme montrent les travaux des musicologues cités dans le paragraphe précédent.

Si les connaissances de théorie musicale supposent un type de raisonnement déductif, dans l'utilisation des *patterns* est le raisonnement analogique qui s'impose. Devant une situation, il s'agit de penser à un *pattern* qui marche bien dans des situations similaires, et éventuellement l'adapter à la nouvelle situation. Beaucoup de concepts et techniques de jazz, comme rythme, style et swing, sont difficiles à saisir formellement par des règles telles que celles de la musique classique. Dans ce sens, les *patterns* sont des exemples représentant des connaissances en extension.

Avant de finir cette section, il importe de rappeler qu'il n'y a pas de "grammaire" qui indique que tel *pattern* suit un tel autre. Selon les études de ses musicologues, un solo ou un accompagnement n'est pas composé d'un *pattern* après l'autre. On y trouve des zones indéfinies où des *patterns* n'apparaissent pas clairement et aussi des zones où on explore les possibilités *pattern* en le développant et variant. Ces *patterns* ne sont donc guère figés. Owens par exemple, classe les *patterns* de Charlie Parker dans des hiérarchies ayant 4 niveaux de profondeur (Owens, 1974), soulignant que les *patterns* portent en eux-mêmes des variations.

2.4.2. Traits musicaux

A plusieurs reprises nous avons demandé aux jazzmen que nous avons rencontrés, les bassistes en particulier, pourquoi ils avaient choisi telle note (ou telles notes) en particulier. Quand la réponse n'était pas "je sais pas trop expliquer", elle était de type "parce qu'elle(s) appartient à telle gamme", "parce que ça augmente la tension", "parce que ça donne un contour mélodique plus suave", "parce j'aime la sonorité de l'instrument dans cette tessiture", etc. Dans ces cas, quand nous leur demandions si la note (ou les notes) ne pouvait pas être une autre (ou d'autres) appartenant à la même gamme ou au même accord ou à la même tessiture, etc., la réponse était "si, bien sur".

A notre avis, ces réponses des jazzmen suggèrent que, dans le processus de création, un jazzman fait appel à divers traits musicaux au même titre que dans le processus de perception (Cf. section 2.2). Ces traits semblent fonctionner comme des contraintes qui guident le choix des notes. A condition que des propriétés globales soient satisfaites, on peut choisir n'importe quelle note. Nous ne sommes pas les seuls à faire ce constat. Par exemple, Holland (Holland, 1989) relate l'étonnante économie de vocabulaire qui ont les musiciens quand ils commentent ce qu'ils ont joué. Avec quelques traits, ils arrivent à expliquer leurs choix principaux et à comprendre ce qu'un autre a fait. Holland a développé un système d'enseignement d'harmonie où l'élève part d'un morceau qu'il aime et le "recompose" en relaxant ou ajoutant des contraintes associées à ces traits. Ainsi, l'élève se rend compte de l'importance de chaque trait. Chez les pédagogues de jazz, cette vision des choses fait aussi écho. Dans une sorte de formulaire employé pour commenter le solo de Miles Davis dans STRAIGHT NO CHASER (Davis, 1958), David Baker (Baker, 1980) énumère quelques traits (Cf. tableau 2.4.1) qui viennent s'ajouter à ceux que nous avons déjà cités dans la section 2.2.2.

Trait	Valeurs (conjonction)
Tune type	blues, be-bop, standard
Key	F Major
Tessitura	medium to high
Dramatic devices	vibrato, slurs, glissandi, alternate fingerings
Scale preferences	major, blues, diminished, chromatic
Developmental techniques	single climax, simple to complex, chord referential, vertical and horizontal, use of quotes
Rhythmic practice	reiterative
Melody	folk-like, riff-like, bebop, bluesy, wide expressively
Rhythmic patterns	...transcription
Melodic patterns (II V)	...transcription
Solo transcription	...transcription

Tableau 2.4.1 - Description d'un solo de Miles Davis par David Baker

En réalité, il nous semble qu'il y ait une relation entre le manque de temps et cette pensée "abstraite" ou partielle d'un ensemble de notes à travers ses traits. N'ayant pas suffisamment de temps pour réfléchir, le jazzman s'efforce de garantir la réalisation de quelques propriétés à travers son choix de notes. Bien que les jazzmen que nous connaissons soient d'accord là-dessus et que d'autres chercheurs partagent notre point de vue (Baggi, 1992; Hodgson, 1996a; Levitt, 1993), nous ne pouvons pas prouver la plausibilité de cette affirmation, mais peu importe. En revanche, ce qui nous paraît beaucoup plus important, vis-à-vis de la conception de notre agent jazzman, est qu'il est

impossible d'expliquer complètement les choix des musiciens. Il faut être donc capable de manipuler des descriptions partielles de notes parce que ce sont ces connaissances que nous pouvons acquérir auprès des jazzmen.

Bien entendu, dans beaucoup de situations un jazzman veut choisir précisément une note et pas n'importe laquelle. Par exemple, un soliste peut jouer une note très longue bien précise pour qu'elle ait des couleurs différentes à chaque changement d'accord (Miles Davis en est un spécialiste). Un bassiste fait un choix assez précis pour les notes de rapprochement, les notes pédales, les notes "cibles", etc. Mais même dans ces cas, il y a souvent d'autres intentions derrière la note : les différentes couleurs de la note longue pour le soliste, la création d'une tension particulière pour la note pédale jouée par le bassiste, etc. Ce "niveau d'abstraction" dans la pensée du jazzman est sans doute un élément d'explication au fait que nombreuses improvisations ou nombreux accompagnements sont possibles pour une grille donnée.

2.5. PRATIQUE ET APPRENTISSAGE

Voyant le parcours de n'importe quel grand maître on se rend compte de l'importance de la pratique dans la formation des jazzmen. Selon Hodeir, "c'est par assimilation intuitive plus que par réflexion qu'il (le jazzman) s'approprie l'essentiel des ses acquisitions" (Hodeir, 1981) [p. 24]. À notre avis, deux raisons majeures concourent sans doute à cela.

La première raison concerne le besoin d'une utilisation immédiate des connaissances. La pratique a comme fonction de développer l'habileté instrumentale, de consolider les connaissances de la théorie musicale, ainsi que l'utilisation des fragments mélodiques ou rythmiques connus (patterns), pour que lorsque le jazzman en ait besoin tout soit "au bout des doigts". Par exemple, il faut qu'en regardant un accord un bassiste sache automatiquement, par exemple, quelles sont ces notes et les gammes qu'il suggère, pour qu'il puisse concentrer ses ressources cognitives sur d'autres décisions.

La seconde, et plus importante, raison tient au fait que des connaissances, notamment celles qui font la différence entre le génie et le musicien médiocre, dépassent largement le cadre de la théorie musicale et ne peuvent être acquises que par la voie de la pratique. Parmi ces connaissances, il y a tout ce qui concerne l'interaction

avec l'environnement. Par exemple, c'est sur scène et pendant les répétitions que l'on saisit le style de chaque musicien, et que l'on apprend à anticiper ce qu'ils vont jouer. Aussi, l'évaluation de l'humeur du public, qui peut jouer un rôle dans le déroulement de l'improvisation (Sabatella, 1996) [p. 37], échappe-t-elle à tout apprentissage théorique. Un autre type de connaissance concerne l'aspect rythmique, qui malgré son rôle privilégié en jazz, est très mal formalisé. Cela est dû certainement au fait que le rythme est plutôt quelque chose d'intuitif et de corporel (Hodeir, 1981) [pp. 189-91]. Nous pourrions continuer cette liste en citant le développement de l'expressivité à travers le traitement de la matière sonore, le processus de raffinement et extension du vocabulaire des traits musicaux ou encore les diverses heuristiques qui permettent le jazzman de proposer des solutions aux plus grands problèmes de la création en jazz exprimés par les dialectiques : répétition *vs.* innovation, mélodie *vs.* harmonie, personnel *vs.* collectif; etc.

Revenons à la méthode d'apprentissage IAI qui réunit un peu tout ce que nous avons présenté dans ce chapitre. On ne sait pas comment l'innovation se produit, mais les pédagogues de jazz croient que les phases d'imitation et assimilation y sont pour beaucoup. L'écoute plus l'imitation fournit au jazzman des exemples mais c'est à travers l'assimilation qu'il est véritablement en mesure de comprendre pourquoi une idée musicale (une phrase, un son, un rythme, etc.), exprimée par quelqu'un ou par lui-même, sonne bien ou mauvais dans une situation donnée. Alors, pour assimiler les concepts, il lui faut de la théorie et de l'écoute pour bien analyser ce qui se passe mais il lui faut surtout de l'expérimentation. La pratique est donc au centre de la démarche du jazzman.

C'est n'est pas inutile de rappeler que cette démarche d'apprentissage est individuelle et vise donner un caractère personnel aux connaissances que nous venons de citer. Si dans n'importe quel domaine les connaissances ne sont jamais exactement les mêmes chez différents êtres humains, en art, cette personnalisation est poussée aux limites, car l'innovation passe aussi par l'affirmation d'un style individuel.

2.6. CONCLUSIONS

Nous avons pu tirer beaucoup de leçons du travail que nous venons de présenter. Parmi les nombreuses choses que nous avons comprises, trois aspects des connaissances des jazzmen nous ont particulièrement frappé, car ils ont une incidence directe sur les questions que nous avons posées à la fin du chapitre 1.

En premier lieu, aussi bien pour la compréhension de ce qu'ils écoutent que pour ce qu'ils créent, les jazzmen s'appuient, consciemment ou pas, sur des structures musicales groupant plusieurs entités musicales (notes, accords, etc.). Ceci nous donne d'abord une importante piste pour la détermination du segment d'improvisation. Notre agent jazzman doit progresser en calculant des phrases mélodiques, plutôt que note par note, ces phrases devant une relation quelconque avec les structures de la grille (schémas d'accords, mesures, sections, etc.). Une autre piste importante concerne le filtrage des données de l'environnement à considérer pour le calcul du segment d'improvisation. L'agent peut plus facilement interpréter ces structures musicales qu'une séquence de notes ou d'accords non-corrélés. Bien que la notion de structure nous fasse mieux comprendre comment concevoir notre agent jazzman, la question de comment segmenter l'improvisation pour pouvoir avancer dans la grille reste ouverte car plusieurs structures se superposent.

En deuxième lieu, nous avons vu que les jazzmen font appel à un grand vocabulaire de "traits musicaux" à la fois pour analyser ce qu'ils écoutent et pour guider leur choix de notes. De même que l'agent peut avoir une vue plus "abstraite" des accords et des notes en les groupant, il peut interpréter les notes et accords selon leurs multiples traits et ainsi minimiser le problème du filtrage des informations cité plus haut. L'idée centrale autour du concept de traits musicaux est que notes ne sont qu'un moyen d'exprimer des idées musicales, de "raconter une histoire"¹². Faisant une analogie extrême, penser le contraire consisterait à imaginer que l'écriture d'un roman se réduirait au choix des mots. Nous pensons donc que le choix des notes doit passer d'abord par un choix des propriétés (selon les traits) que ces notes doivent satisfaire. Malheureusement, l'incorporation de la notion de traits soulève des problèmes

¹²Quand une improvisation est réussie, on dit que "it tells a story".

importants. Comment représenter ces traits ? Quelle est l'influence mutuelle de ces traits ? Comment prendre en compte un ensemble quelconque de traits pour calculer de façon cohérente les notes ?

Dans la figure 2.6.1 nous donnons une illustration “pictographique” de descriptions supplémentaires (structures et traits) aux symboles initialement affichés dans la figure B.2.

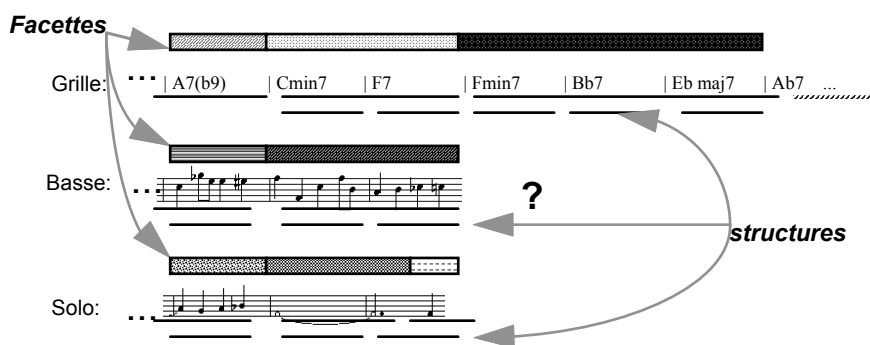


Figure 2.6.1 - Tâche du bassiste de jazz revisitée : descriptions plus riches

En dernier lieu, nous avons remarqué le caractère empirique de l'apprentissage des jazzmen essentiellement basé sur l'écoute et la pratique. Les règles de la théorie musicale sont très importantes mais elles n'englobent que partiellement le savoir-faire musical. Soit dans les écoles de jazz, soit par des méthodes (Baker, 1980; Coker, 1970; Coolman, 1985), soit simplement en écoutant et imitant les grands maîtres, le jazzman acquiert les principaux concepts du jazz à travers des exemples, plutôt que par définitions ou règles. Pendant tout son parcours, un jazzman semble accumuler et créer tout une collection d'exemples de rythmes, de phrases mélodiques, d'enchaînement d'accords qui peuvent être réutilisés selon les circonstances. L'évidence la plus remarquable de ceci est l'identification de catalogues de patterns mélodiques ou rythmiques (fragments mélodiques et rythmiques récurrents) utilisés par plusieurs jazzmen célèbres (Baker, 1980; Kernfeld, 1983; Owens, 1974; Smith, 1983). Bien entendu, jouer du jazz ne se limite pas à réutiliser de fragments mélodiques ou rythmiques, mais du point de vue technique de la construction d'un agent jazzman, ces fragments peuvent être très utiles. En effet, ils codent “en extension”, au sens où on l'entend en IA, des connaissances difficiles à formaliser comme celles liées au rythme et aux accents d'intensité. Néanmoins, la réutilisation de ces fragments par l'agent pose un problème

central : quels sont les critères pour le choix d'un fragment ? Comment une bibliothèque de fragments devrait-elle être organisée ? Puisque ces fragments sont de vrais "cas", dans le sens où ils correspondent à des passages réellement joués par quelqu'un et non à des connaissances d'ordre général, et puisqu'ils peuvent être effectivement réutilisés dans des conditions similaires, le raisonnement à partir de cas (Kolodner, 1993) peut être un outil conceptuel adéquat pour la construction d'une base de cas composé de ces fragments (Ramalho & Ganascia, 1994a). Le critère de réutilisation serait alors la similarité entre le segment d'improvisation actuel et celui ou le fragment joué précédemment. Mais, que veut-t-on dire par une situation similaire ? Nous revenons à la question centrale du raisonnement à partir de cas : comment indexer les cas ?

La conclusion la plus générale que nous pouvons tirer de ce chapitre est qu'il faut pouvoir donner à la machine suffisamment de connaissances pour qu'elle puisse interpréter pertinemment son environnement (les accords de la grille, ce qui a été déjà joué) et décider quels notes choisir pour constituer son improvisation ou son accompagnement. Ces connaissances concernent des tâches très variées comme l'analyse, la synthèse, la planification, la reconnaissance de formes et le contrôle moteur, et elles ont différentes natures : déclaratives (par ex. la théorie musicale en générale), procédurales (par ex. les doigtés), temporelles (par ex. l'anticipation), etc. De même, les mécanismes d'inférence sont divers : déductives (par ex. l'analyse harmonique), analogiques (par ex. la réutilisation des patterns), etc. Si, d'une part ces connaissances apportent des réponses aux questions que nous nous sommes posées sur la construction d'un agent jazzman, d'autre part, leur intégration dans un programme pose un problème. En effet, quoique nous ayons mis en relief la manipulation et perception de structures, de traits musicaux et de fragments mélodiques ou rythmiques, toutes les connaissances doivent être considérées. Par exemple, avoir une collection de fragments mélodiques ne sert à rien si l'agent ne sait pas faire une analyse harmonique de la grille pour voir où ils peuvent être éventuellement réutilisés. C'est ce problème de l'intégration de ces connaissances qui a motivé le plus notre travail. Le modèle que nous présentons par la suite (chapitre 4) essaye d'y apporter une réponse.



3. CONNAISSANCES ET RAISONNEMENT DANS LES PROGRAMMES JOUANT DU JAZZ

Dans ce chapitre, nous analysons les programmes existants dans le domaine de l'improvisation et l'accompagnement en jazz. Notre analyse vise à répondre aux trois questions suivantes :

- Dans quelle mesure les connaissances des jazzmen, identifiées dans le chapitre précédent, sont-elles incorporées dans de tels programmes et de quelle façon les sont-elles ?
- Dans quelle mesure et comment l'environnement des jazzmen est-il pris en compte par ces programmes ?
- Quel est l'impact de la prise en compte de telles connaissances et de tel l'environnement sur les résultats musicaux ?

Nous analysons les programmes en prenant l'agent rationnel comme référence. Ainsi, nous commençons par donner une vision panoramique et succincte des programmes pour expliciter leurs tâches spécifiques. Ensuite, nous décrivons les aspects de perception, d'environnement, d'action et de raisonnement des programmes. Nous concluons le chapitre par un bref examen des résultats musicaux obtenus. Il est important de souligner que, à notre connaissance, aucun article n'a abordé la totalité des programmes décrits ici, ni proposé une grille d'analyse semblable à celle que nous employons.

3.1. PANORAMA DES PROGRAMMES

Il existe plusieurs programmes essayant d'automatiser des "tâches de composition" en musique tonale : harmonisation de chorales à quatre voix (Ebcioglu, 1992; Pachet & Roy, 1994; Thomas, 1985), mélodie et accompagnement dans le style classique romantique (Cope, 1991), contrepoint à deux voix de diverses espèces (Balzer & Wegscheider, 1992; Goldman, Gang & Rosenschein, 1995; Widmer, 1992), composition de mélodies en général (Marsella & Schmidt, 1992; Riecken, 1992; Vincinanza & Prietula, 1989). Bien que nous puissions tirer des leçons de ces différents programmes, nous étudierons uniquement les programmes de jazz afin de pouvoir mieux centrer notre analyse sur les aspects spécifiques à l'improvisation.

3.1.1. Tâches

Le tableau 3.1.1 énumère les programmes les plus connus en fonction de leur tâche. Le but de ces programmes est de produire, à partir d'un type de grille, des improvisations ou des accompagnements intéressants dans un style donné.

programme	tâche	style	type de morceau
Ames & Domino 92 (Cybernetic Composer)	acc. (basse, piano, batterie) et imp. (réécriture du thème)	be-bop, latin jazz, rock et ragtime	blues, standard, rock, rag time, bossa nova
Baggi 89 (NeurSwing)	acc. (basse, piano, batterie)	be-bop	blues et standard
Brown & Sidley 93	imp.	?	standard
Band-in-a-box 91	acc. (basse, piano, batterie)	très nombreux	toute grille en 4/4
Fry 91 (Flavors Band)	imp. et acc. (tous instruments) selon le programme donné	tout style pouvant être programmé	standard, blues, etc.
Giomi & Ligabue 89	imp.	be-bop	blues
Hidaka et al. 95	acc. (réécriture de parties de basse et batterie données)	?	standard
Hodgson 96	imp. (saxophone)	be-bop	blues et standard
Horowitz 95	imp. (trompette)	nouvelle orléans	nouvelle orléans
Johnson-Laird 91	acc. (basse)	?	blues
Levitt 78,93	acc (bass, piano)	nouvelle orléans, be- bop, rock, ragtime	blues et standard
Pachet 87, 90b	acc. (basse, piano)	be-bop	blues et standard
Pennycook et al. 93	imp.	be-bop	blues et standard
Spector & Alpern 94,95	imp. (trading four)	be-bop	blues et standard
Ulrich 77	imp.	?	standard
Walker 94 (ImprovisationBuilder)	imp. et acc. (trading four)	?	standard

Tableau 3.1.1 - Présentation des programmes (acc. = accompagnement; imp. = improvisation; ? = pas clair)

Parmi ces programmes, *ImprovisationBuilder* et *Spector*¹³ ont des tâches un peu différentes des autres car ils se concentrent sur les *trading fours*, des situations de type “question-réponse” où chaque musicien doit, à tour de rôle, improviser pendant 4 mesures. En plus de répondre à un soliste humain, *ImprovisationBuilder* accompagne ce soliste lorsque ce dernier improvise. Deux autres programmes, *Hidaka* et *Flavors Band*, ont des natures un peu particulières. *Hidaka* ne génère pas un accompagnement, mais simplement modifie un accompagnement déjà existant en fonction de l’environnement. *Flavors Band* n’est pas véritablement un programme de génération automatique d’improvisation ou d’accompagnement. C’est un langage de programmation spécialisé en musique tonale, et en particulier en jazz, qui permet à un utilisateur de spécifier avec plus ou moins de précision ce qu’il veut que la machine joue, et ceci pour chaque morceau.

3.1.2. Approches

Les programmes que nous analyserons dans les prochaines sections de ce chapitre peuvent être regroupés en sept approches différentes.

La première approche (très répandue au-delà du jazz) repose sur l’utilisation des grammaires dans lesquelles des probabilités sont associées aux règles, c’est-à-dire aux branches d’une *probabilistic derivational tree* (Cf. figure 3.1.1). *Cybernetic Composer*, *Giomi* et *Johnson-Laird* utilisent de telles grammaires d’abord pour générer le rythme (début et durées des notes) et ensuite pour lui assigner des hauteurs.

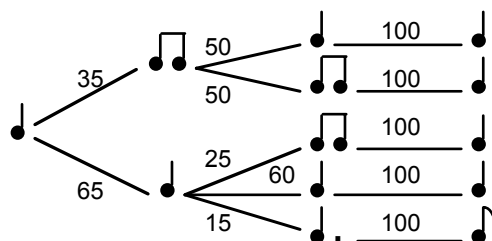


Figure 3.1.1 - Exemple de grammaire définissant le rythme d’une mesure de ligne de basse (les pondérations sont données en pourcentage).

¹³Pour alléger la lecture, nous nous référerons aux systèmes présentés dans le tableau 3.1. par leurs noms, quand ils existent, ou par le nom du premier auteur

La deuxième approche est très similaire à la première mais il s’agit d’utiliser non pas des grammaires mais des heuristiques pour borner les choix aléatoires. Cette approche est adoptée dans `Brown` et `ImprovisationBuilder`. Par exemple, pour les durées des notes d’un segment de deux mesures, `Brown` utilise l’heuristique suivante : “préférer les notes courtes au début de la phrase (jusqu’au cinquième temps) et les notes longues ensuite”.

La troisième est l’approche connexionniste au sens large, mise en œuvre en `NeurSwing` et `Horowitz`. Dans `NeurSwing`, un utilisateur ajuste les valeurs de trois traits musicaux (*hot/cool*, *dissonance/consonance*, *free/as-is-ness*) qui constituent les entrées d’un réseau de neurones. Ce réseau active de façon cohérente plusieurs traits de sortie (densité, appoggiature, intensité, *voicing*, etc.) qui “paramétrisent” les procédures qui seront appelées pour générer les notes. `Horowitz` suit le même schéma, selon lequel les traits constituent des contraintes sur le choix de notes. Mais, au lieu d’un réseau de neurones, `Horowitz` associe un “agent” (au sens IA distribuée) à chaque trait et laisse les agents interagir pour combiner les traits.

La quatrième approche, adoptée par `Levitt`, est la programmation par contraintes. À partir du morceau pris comme un tout, `Levitt` fait des raffinements progressifs du rythme et des hauteurs en fonction de contraintes associées à un nombre réduit de traits musicaux. `Levitt` associe des valeurs par défaut pour les traits non spécifiés.

Dans la cinquième approche, `Pachet` propose une représentation des actions des jazzmen (par ex. “jouer des arpèges pendant cette mesure” ou “jouer beaucoup de notes à partir de la prochaine section jusqu’à la fin de l’improvisation”) associées à différents traits musicaux. Pour chaque segment, ces “actions potentielles”, appelées *stratégies*, sont activées (suscitées) à l’aide de règles de production selon les accords de la grille et ce que le programme a déjà joué. Une fois que les éventuels conflits entre les stratégies activées ont été enlevés, ces stratégies sont “concrétisées” en termes de notes par un ensemble de procédures.

La sixième approche, que l’on pourrait appeler “transformationnelle”, vise à transformer une mélodie donnée à travers l’application d’opérateurs (`Spector`, `ImprovisationBuilder`, `Hidaka`). Les transformations employées dans `Spector` et

ImprovisationBuilder sur des phrases de quatre mesures sont par exemple *extend*, *trunc*, *invert*, *retrograde*, *rotate*, *diminute*, etc. Spector choisit les transformations selon un degré d'adéquation (*fitness*) qui est calculé à partir de critères comme *tonal-novelty-balance*, *rhythmic coherence*, etc. Hidaka utilise des transformations beaucoup plus simples (par ex. changement d'intensité, de dissonance et de densité), mais ces transformations sont appliquées en fonction des "intentions" courantes du soliste qu'il accompagne (Cf. section 3.2.2).

La dernière approche consiste à réutiliser systématiquement des patterns mélodiques ou rythmiques en les enchaînant les uns après les autres, un pattern étant éventuellement sujet à des modifications. Ces patterns sont des "successions de degrés de gammes" (utilisés dans Ulrich) ou des fragments mélodiques ou rythmiques (utilisés dans Hodgson, NeurSwing et Band-in-a-box). Dans ces programmes, en plus d'une dose d'aléatoire, la réutilisation d'un pattern se base sur différents critères, le plus important étant l'accord sur lequel il sera joué.

Dans la suite, nous ne faisons pas une description très précise du fonctionnement de ces programmes ni une énumération des remarques bien connues à leur propos. Nous préférons adopter une grille d'analyse un peu différente, basée sur des critères indépendants des approches citées dans cette partie.

3.2. DONNEES PERCEPTIVES ET ENVIRONNEMENT

De toutes les données visuelles de l'environnement (Cf. figure 1.2.2), seulement la grille est considérée par les programmes car elle peut être donnée sous le format de texte sans perte d'information.

3.2.1. Problèmes de l'écoute

Le problème du décodage du son de plusieurs instruments polyphoniques en termes de notes est loin d'être résolu. La solution envisageable actuellement est d'extraire les informations des notes, des phrases, et d'autres traits musicaux à partir d'instruments portant une sortie MIDI (Loy, 1985; Moog, 1986). L'information MIDI spécifie déjà les hauteurs, intensités, débuts et fins de notes. Toutefois, même avec ces données MIDI, le problème de la perception musicale en temps réel reste un problème complexe,

constituant en soi un domaine de recherche à part entière qui intéresse aussi bien des psychologues que des informaticiens. Déjà au niveau le plus élémentaire qui est celui du temps, des nombreux problèmes se posent comme : le suivi du tempo (*beat induction*) (Allen & Dannenberg, 1990; Desain & Honing, 1994; Large, 1995), la détection de la métrique (Desain & Honing, 1992; Longuet-Higgins & Lee, 1984; Povel & Essens, 1981) et la quantification rythmique (Agon et al., 1994; Desain & Honing, 1992; Roozendaal, 1990). En outre, il y a les difficultés de détection des frontières de phrases et d'extraction de traits plus abstraits comme la dissonance, le style, etc. (Lerhdal & Jackendoff, 1983; Narmour, 1989; Rowe, 1993).

Dans le domaine de ce que l'on appelle "les systèmes musicaux interactifs", quelques programmes ont résolu avec un succès relatif ces problèmes. Les programmes dits de suivi de partition ont comme tâche de jouer avec un ou plusieurs musiciens humains. Un programme de ce type connaît préalablement ce que lui-même et les autres musiciens vont jouer (Dannenberg, 1984; Grubb & Dannenberg, 1994; Vercoe, 1985). Ces programmes ne sont donc pas directement applicables au jazz car, d'une part, on ne connaît pas ce que les musiciens vont jouer et, d'autre part, on a besoin de décrire les notes identifiées en termes de traits musicaux plus abstraits pour pouvoir comprendre les intentions des musiciens que l'on s'efforce d'accompagner.

Le programme "référence" en matière d'automatisation de la perception est Cypher (Rowe, 1993). Il propose des solutions intéressantes pour la plupart des problèmes de perception cités plus haut. Nous ne l'avons pas cité plus tôt car ce n'est pas un programme conçu pour le jazz mais plutôt pour l'*interactive live performance* en musique électronique contemporaine. Cypher applique un ensemble de transformations à ce que le musicien va jouer afin d'établir une interaction particulière. Les liens entre les transformations et ce que le musicien joue sont programmés préalablement par le musicien lui-même.

3.2.2. Données auditives effectivement incorporées

A part Cypher, seulement trois programmes (Pennycook, Hidaka et ImprovisationBuilder) sont en mesure actuellement d'écouter un environnement comme celui du jazz. Le fait que les mécanismes de perception soient compliqués à

automatiser explique les simplifications faites dans ces trois programmes en ce qui concerne le raisonnement et l'action.

Dans ces programmes, les caractéristiques perceptives de base proposées par `Cypher` ont été étendues dans deux directions. `Pennycook` peut détecter des événements plus fins, comme un *motif* qu'un soliste joue à un moment donné. `Hidaka` inclut la perception de traits plus globaux et à long terme (excitation, tension, importance de l'accord, substitution d'accord et reprise du thème), obtenus à partir des caractéristiques perceptives de base (tessiture d'une note, densité de notes, notes de tension, note d'accords, note de la gamme, note du thème, etc.). Ainsi, si depuis un certain temps le soliste joue beaucoup de notes, avec dissonances, très fort et dans une tessiture haute, alors `Hidaka` peut conclure que le soliste est très "excité".

Tous les autres programmes ne prennent en compte de l'environnement que la grille, et éventuellement le thème associé. Ceci est vrai même pour les programmes qui génèrent des parties pour plus d'un instrument, comme `Cybernetic Composer`, `NeurSwing`, `Flavors Band`, car les instruments ne s'écoutent pas. En réalité, dans le cas de `NeurSwing` la section rythmique est vue comme un seul instrument, ce qui permet de mieux coordonner les instruments les uns par rapport aux autres.

Pour conclure, rappelons qu'aucun programme n'est capable actuellement d'anticiper ce que les jazzmen humains vont jouer, ni d'évaluer dynamiquement ce qu'il joue lui-même. `Cypher` propose une esquisse d'auto-évaluation mais qui est trop simple par rapport à celle que nous avons présentée dans la section 2.2.3.

3.2.3. Prise en compte globale de l'environnement

Outre les données perceptives, les composantes de l'environnement ont une influence directe sur le raisonnement de l'agent. Comme nous l'avons exposé précédemment (Cf. section 1.2.2), l'environnement d'un jazzman est accessible, non-déterministe, non-épisodique, dynamique et continu avec des contraintes de temps importantes. Ces caractéristiques de l'environnement sont généralement prises en compte par les programmes à l'exception du fait que l'improvisation se déroule en direct.

En effet, au lieu de calculer les notes au fur et à mesure qu'ils avancent sur la grille, certains programmes (Levitt, Ulrich et Cybernetic Composer) calculent toutes les notes à jouer jusqu'à la dernière mesure en une seule fois et peuvent faire autant de "retours arrière" qui soient nécessaires. Ce procédé est radicalement opposé à l'idée même d'improvisation. Improviser est construire quelque chose graduellement (ce qui a été joué ne peut plus être changé), en étant toujours capable de changer la façon de jouer en fonction l'environnement ou de ce que l'on vient de faire soi-même.

Par ailleurs, certains programmes ne tiennent pas compte des contraintes de temps. Le programme peut "tourner" toute une journée pour enfin produire une improvisation. Si l'on ignore le manque de temps, un des vecteurs de formation d'un jazzman, on ne voit pas le besoin d'explicitier et modéliser des connaissances heuristiques employées en jazz. Actuellement, aucun programme à part *Pachet* ne présente de changement de stratégie lié au manque de temps. En d'autres termes, le calcul des notes à jouer est mené de la même façon pour quel que soit le temps disponible. Dans *Pachet*, au contraire, lorsque ERE^* est trop petit (Cf. équation 1.2.2), on raccourcit le raisonnement en déclenchant des procédures plus simples qui produiront des phrases mélodiques de type cliché. En résolution de problèmes en temps réel, des nombreuses approches ont été développées pour tenir compte du manque de temps (Lesser, Pavlin & Durfee, 1988; Musliner, Durfee & Shin, 1993; Strosdiner, 1994) mais, pour l'instant, la répercussion de ces travaux dans les applications musicales est faible.

3.3. ACTION

L'action d'un jazzman est de jouer, de produire un son à l'aide de ses doigts et de ses poumons pour les instruments à vent. Ceci a deux implications importantes pour la construction d'un agent qui joue du jazz.

Premièrement, si l'on veut vraiment obtenir des bons résultats musicaux, il est souhaitable de ne pas restreindre l'action de l'agent à l'écriture, sur une portée, de notes ayant un début, une durée et une hauteur. En effet, avec de telles réductions, un agent ne pourra jamais "swinguer" même s'il a conçu une belle improvisation ou un bel accompagnement. Bien entendu, quelques simplifications, notamment sur les modes de jeux et l'articulation, s'imposent car un agent jazzman fait usage d'un synthétiseur.

Néanmoins, il reste encore quelques aspects pouvant être traités par l'ordinateur : l'intensité des notes, le tempo et peut-être quelques changements de son. Cela peut contribuer au swing puisque pour swinguer il faut, entre autres, tenir compte des accents (variation d'intensité des notes) et des changements de tempo (*ahead beat* et *behind beat*). Le changement de tempo n'est traité que par le programme `NeurSwing`. Quant aux intensités des notes, elles ne sont considérées que par un petit groupe de programmes existants mais les variations d'intensité sont toujours globales : on joue *toutes* les notes plus fort ou plus doucement. Les seuls programmes à s'occuper des accents des notes sont `Band-in-a-box` (pour toute la section rythmique) et `NeurSwing` (pour le piano et la batterie).

Deuxièmement, il serait intéressant de tenir compte des contraintes anatomiques. Cela n'a pas encore été fait parce qu'il est difficile de formaliser telles contraintes (Cordier, 1995; Sayegh, 1989). Toutefois, il y a des aspects plus simples pouvant être traités par l'ordinateur : les transpositions de hauteur et de tempo. Une phrase ne peut être jouée facilement dans n'importe quelle tonalité ni à n'importe quel tempo. Même si notre agent en a les possibilités, puisqu'il s'agit d'une machine, il ne doit pas le faire sous risque de compromettre la plausibilité musicale. Une ligne de basse jouée à 80 noires par minutes va difficilement "sonner bien" à 250. Aucun des programmes présentés dans ce chapitre ne semble prendre en compte ces deux aspects de tonalité et de tempo.

3.4. RAISONNEMENT

Dans cette section, nous nous concentrons sur le raisonnement impliqué dans les choix des actions qu'un agent jazzman doit effectuer vis-à-vis de son environnement.

3.4.1. Fondements en théorie musicale

Bien que les concepts fondamentaux de la théorie musicale tonale (par ex. note, accord, intervalle, tonalité, gamme, etc.) soient bien définis, certains sont complexes et leur manipulation par des machines pose des problèmes. Il est donc important d'avoir une représentation adéquate de ces concepts, car ceci a un impact direct sur la facilité avec laquelle des inférences peuvent être appliquées. Par exemple, les mélodies de `Spector`

sont représentées d'une façon assez "pauvre" par une liste de 64 numéros, chacun correspondant à une hauteur et ayant comme durée une double-croche (les silences sont représentés par -1). De même, un accord Cmaj7 est représenté dans *Ulrich* par C (4 7 11), où les numéros correspondent à la distance en nombre de demi-tons de la tonique. Comme ces représentations ne font pas de référence explicite à la notion d'intervalle, la notion d'enharmoine est absente ou complexe à obtenir. Du coup, il devient beaucoup plus difficile de programmer les activités qu'un agent jazzman doit accomplir, notamment l'analyse harmonique, l'analyse des traits musicaux, la réutilisation de patterns, etc.

En fait, la représentation de connaissances musicales est un domaine à part entière. Il y existe des systèmes conçus comme des plates-formes générales de base de représentation de connaissances musicales comme *Pla* (Schottstaedt, 1983), *Flavors Band* (Fry, 1991), *Common Music* (Taube, 1991), *MODE* (Pope, 1991a), *MusES* (Pachet, 1994), etc. Ces plates-formes peuvent être utilisées pour construire des systèmes d'aide à la composition ou de composition et improvisation automatiques. *Flavors Band* et *MusES* sont des systèmes spécialement conçus pour la musique tonale et le jazz. Ils témoignent d'une tendance, amorcée il y a plus de dix ans avec *Pla* (Schottstaedt, 1983) et *Formes* (Rodet & Cointe, 1984), très répandue pour la représentation d'entités musicales : les langages à objets (Pope, 1991b).

Pour clore cette section nous aimerions parler de l'analyse harmonique fonctionnelle. Celle-ci sert à déterminer des tonalités locales et globales, si possible de façon hiérarchique, à identifier les fonctions des accords et à indiquer les schémas d'accords (Cf. Section 2.3.1). L'automatisation de cette analyse pose des nombreuses difficultés (Baggi, 1974; Mouton & Pachet, 1995; Steedman, 1984; Winograd, 1993). Ainsi que la représentation de connaissances, l'analyse harmonique est en soi un thème de recherche. Cependant, parmi les programmes que nous analysons dans cette section, plusieurs proposent des solutions simplifiées qui sont suffisantes pour la construction d'une improvisation ou d'un accompagnement.

3.4.2. Granularité du raisonnement

Examinons maintenant comment les programmes répondent aux questions concernant le choix des segments d'improvisation et le "filtrage" des informations de

l'environnement formulées à la fin du chapitre 1. Le contrôle entre le niveau local et le niveau global des décisions de l'agent dépend beaucoup d'un choix judicieux de ces segments et de la façon de considérer le passé et le futur.'

D'après ce que nous avons exposé dans le premier chapitre, le segment "naturel" chez les musiciens (jouant un instrument monophonique) est une phrase mélodique. Il serait donc souhaitable qu'un programme avance sur la grille en calculant des phrases mélodiques, de taille quelconque, qui puissent s'enchaîner. Le problème est que, contrairement au langage naturel où il y a une ponctuation explicite, on ne sait pas comment déterminer la frontière d'une phrase mélodique (Lerhdal & Jackendoff, 1983; Narmour, 1989; Rowe, 1993). Les phrases se chevauchent très souvent. En plus, entre deux phrases, il y a parfois des "zones de silence" dont la taille on ne sait pas calculer car ce silence peut correspondre, par exemple, au temps qu'il faut au jazzman pourquoi pour qu'il décide ce qu'il va jouer ensuite.

A cause de ces difficultés de segmentation selon les phrases, les programmes actuels choisissent a priori un segment selon des éléments facilement repérables comme un temps, une note, un accord, une mesure, etc. (Cf. figure 3.4.1). Une fois le segment déterminé, les programmes calculent son contenu (les notes). A cet égard, une fragmentation selon laquelle les segments ont des durées variables nous paraît plus "plausible" musicalement. Ainsi, le choix d'un accord comme segment est plus intéressant (Band-in-a-box, Hodgson, Pachel) que le choix d'une mesure (Giomi) ou d'un temps (Johnson-Laird). Il est important de souligner que les schémas d'accords n'ont pas encore été utilisés comme segment d'improvisation, et ce malgré le fait qu'ils sont sans doute les segments qui "collent" le mieux avec les phrases mélodiques. De fait, ces schémas représentent des situations stéréotypées pour lesquelles la réutilisation de patterns mélodiques est assez naturelle (Cf. section 2.3.1 et 2.4.1).

Outre sa "plausibilité" musicale, le choix des segments peut être vue selon un compromis entre le niveau global et le niveau local. Certains programmes calculent une note à la fois (Johnson-Laird) ou des notes à chaque temps (Hidaka), ce qui représentent des segments trop petits. Le problème est qu'une granularité trop fine nuit à la continuité et à la cohérence globale d'une improvisation ou d'un accompagnement. En effet, il est très laborieux de réaliser une propriété particulière (dissonance, densité,

contraste, etc.) sur une note unique ainsi que de faire référence à un motif utilisé dans le passé. D'autres programmes, au contraire, prennent des segments trop grands comme le morceau entier (Levitt, *Cybernetic Composer*). À chaque accord de la grille, ils associent une note ayant la même durée mais sans une hauteur définie. Ensuite, ils divisent ces notes initiales de façon récursive à l'aide des règles de réécriture ou de contraintes (par ex. une blanche donne deux noires). Enfin, ils affectent à chaque note restante une hauteur. Les choix peuvent être bien contrôlés, mais ce procédé évacue tout l'aspect dynamique et imprévu de l'improvisation.

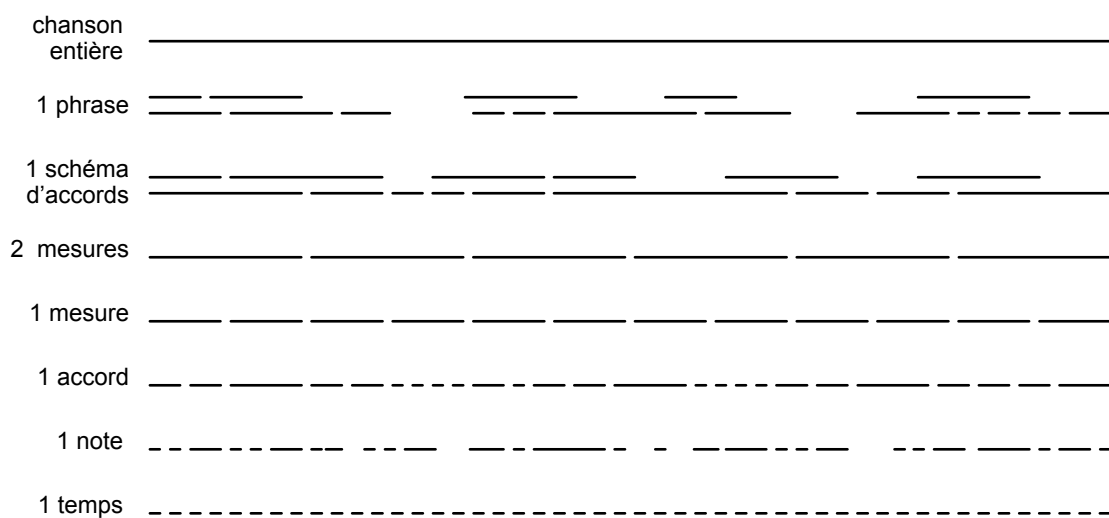


Figure 3.4.1 - Quelques exemples de segments d'improvisation

Le problème de la détermination des segments de raisonnement est étudié depuis longtemps en robotique, planification et, plus généralement, dans tous les domaines où l'on modélise un agent qui évolue dans un environnement dynamique (Tate, Hendler & Drummond, 1990; Vere, 1983). Une solution consiste à choisir des segments ni trop courts, ni trop longs, pour que l'agent puisse planifier des actions à plus long terme tout en gardant la capacité de réagir à l'environnement. A ce titre, des segments intermédiaires comme une mesure (Giomi), deux mesures (Brown), un accord (Band-in-a-box, Pachel) ou un schéma d'accords nous semblent plus appropriés. Il est intéressant de signaler que *NeurSwing* et Hodgson préconisent une sorte de double segmentation où on choisit des patterns rythmiques de durées variables et ensuite on spécifie hauteur et amplitude note par note.

Examinons maintenant la question du filtrage des informations de l'environnement. De la grille, la majorité des programmes considèrent uniquement les accords qui se superposent au segment d'improvisation courant et laissent de côté les accords précédents et subséquents à ce segment. Certains programmes considèrent plus d'information comme le premier accord subséquent ou l'intervalle entre l'accord courant et le subséquent (*Band-in-a-box*, Hodgson et *NeurSwing*). Ceci est malgré tout réducteur car un jazzman ne pense pas uniquement au futur immédiat (l'accord subséquent), d'autant plus que la structure de la grille le pousse justement à réfléchir plus loin. Le deuxième élément de l'environnement est ce que l'agent a déjà joué. D'après ce que nous avons pu comprendre, la plupart des programmes se restreignent à la dernière note jouée. L'exception notable est Hodgson qui essaie de tenir compte des traits musicaux du pattern joué précédemment. Cette problématique de filtrage du contexte, c'est-à-dire des données perceptives de l'environnement, n'est pas un privilège de la musique. Le besoin d'oublier le passé et d'anticiper le futur est toujours présent dans la simulation d'activités se déroulant en direct (Dojat et al., 1996; Hayes-Roth et al., 1992).

Malheureusement, un bon choix du segment ou de filtrage du contexte n'est pas toujours suffisant pour garantir une cohérence entre les segments. En planification, selon la stratégie du "moindre engagement", lorsque l'on est dans un segment de raisonnement courant, on peut décider partiellement de ce que l'on veut réaliser dans les segments futurs (Tate, Drabble & Kirby, 1994; Tate, Hendler & Drummond, 1990). On attend alors la dernière minute pour détailler les actions planifiées auparavant. *NeurSwing* utilise indirectement une telle stratégie à travers quelques variables, comme *hot/coll*, *dissonance/consonance*, *free/as-is-ness*, qui gardent l'état global du déroulement de l'activité de l'agent et qui, en conséquence, influencent plusieurs segments.

3.4.3. Structures

Pendant l'exposition du thème, le jazzman joue des choses simples pour mettre en relief le thème et préparer le début de l'improvisation. Par contre, pendant l'improvisation, il peut jouer dissonant, syncopé, bref de façon plus élaborée. La détection des fractionnements (mesures, carrures, sections, chorus, etc.), qui structurent les morceaux et par conséquent l'improvisation et l'accompagnement, n'est pas difficile car il s'agit

de structures bien définies. Néanmoins, seulement quelques programmes prennent en compte ces structures pour modifier la façon de jouer ou lui donner une cohérence. *Band-in-a-box* tient compte des chorus, et du début et de la fin de grille, tandis que *Fry*, *Giomi*, *Ames* et *Brown* font usage des sections de la grille (AABA, ABAC, etc.) et de la carrure pour construire les phrases mélodiques qui composent l'improvisation.

D'autres facteurs de structuration du discours musical, comme les répétitions et les variations, ne sont pas traités par les programmes existants. Le problème principal n'est pas tant celui de savoir comment générer des variations ou répétitions d'un fragment joué, mais celui de savoir *quand* le faire. Par exemple, le langage proposé par *Pachet* permet de susciter des "stratégies" comme "jouer la phrase transposée d'un ton" mais les conditions selon lesquelles une telle "stratégie" devait être suscitée ne sont pas claires. On rejoint ici le problème de l'évaluation dynamique (Cf. Section 2.2.3). Il faut en effet évaluer ce que l'on vient de faire pour décider si on le répète ou le varie ou si on change complètement. Les connaissances permettant une telle évaluation sont extrêmement difficiles à acquérir auprès des musiciens.

3.4.4. Traits musicaux

Les divers traits qui font partie du vocabulaire d'écoute et de création des musiciens est comparable à la notion de vue ou de représentation multiple en IA. Les traits permettent de spécifier les propriétés de la musique que l'on est en train d'écouter ou de la musique que l'on veut créer. Ceci est fondamental car, très souvent, les musiciens ne peuvent justifier leur choix que partiellement (Cf. Section 2.4.2). On peut alors se demander quels sont les problèmes qui ont empêché qu'une telle notion n'ait pas été davantage considérée par les programmes actuels.

Le premier problème concerne le manque d'unanimité dans la signification précise de certains traits. Par exemple, comment calculer le degré de dissonance du passage de la figure 3.4.2 ? Ceci implique certes une *analyse* des notes par rapport à l'accord sous-jacent, à la tonalité locale et à la métrique, mais il n'existe pas de définition précise de ce calcul. La manipulation des traits est d'autant plus délicate si, au lieu de la perception, on pense à la *création*: calculer des notes ayant des propriétés associées à un trait. Par exemple, il existe d'innombrables exemples de fragments

mélodiques ayant la propriété d'être dissonants. Comment choisir le fragment le plus adéquat ?

Le deuxième problème est que, à peu d'exceptions près, l'on ne connaît pas quelle est l'influence mutuelle d'un ensemble de traits. Ainsi, on sait que l'utilisation de la gamme chromatique et l'absence de la tonique des accords rend un passage dissonant, mais il existe aussi des liens plus subtils. Par exemple, jouer dans une tessiture plus haute suggère parfois plus de dissonance. Ou bien, quel est le lien entre le swing, la dynamique, la densité et le contour mélodique ? Ou encore entre la densité, la dissonance et le style de ligne de basse ? Les deux problèmes évoqués ici se posent moins pour l'analyse car on peut disséquer un passage selon plusieurs traits sans avoir forcément à traiter l'interaction entre les traits. En création musicale, la situation est toutefois différente. De fait, étant donné un ensemble de propriétés associées à divers traits, comment peut-on savoir si cet ensemble est cohérent et si l'on peut vraiment trouver une séquence de notes qui satisfasse aux propriétés ? Si la réponse à cette question s'avère positive, comment effectivement calculer une telle séquence de notes ? Et dans quel ordre les traits doivent-ils être considérés ? Supposons que l'on commence par jouer des arpèges (style de ligne de basse). S'il faut que le passage soit dissonant, on peut par exemple introduire des notes de passage chromatiques. Toutefois, si on veut aussi une densité faible, il faut tout recommencer. En somme, plus on prend en compte de traits, plus il est difficile de contrôler leur interaction et de concevoir un algorithme capable de les matérialiser en notes.

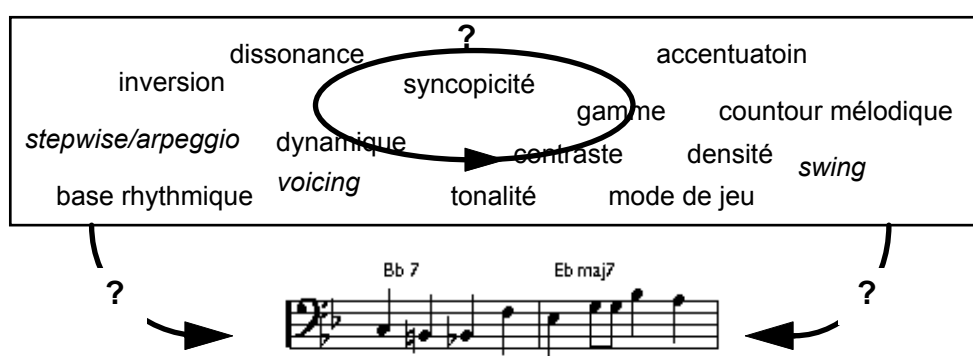


Figure 3.4.2 - Concrétisation des traits en termes de notes

Examinons le fonctionnement des quatre programmes (*NeurSwing*, Horowitz, Levitt et Pachet) qui tiennent explicitement compte des traits musicaux dans leur raisonnement.

Dans *NeurSwing*, les traits sont pris en compte grâce à un réseau de neurones qui a en entrée trois traits non-orthogonaux (*hot/cool*, *dissonance/consonance*, *change/as-is-ness*) dont les valeurs sont ajustées par un utilisateur à l'aide de trois boutons. Ce réseau contrôle la cohérence de l'interaction entre quelques traits "de sortie". Ce sont ces traits de sortie qui pondéreront les probabilités utilisées dans les procédures qui seront appelés, dans un ordre fixe, pour la construction de l'improvisation. Pour la basse, les traits "de sortie" sont cinq : le style de la ligne (arpège ou *stepwise*), le contour mélodique (ascendant ou descendant), l'intensité (amplitude des notes), l'utilisation d'appoggiatures (des croches fonctionnant comme notes de passage) et le changement de tempo (*behind/ahead beat*). Un positionnement du premier bouton proche du "hot" augmente la probabilité que la basse joue en *stepwise* ascendant, fort, avec des appoggiatures et *ahead beat*. Le deuxième bouton en position "consonant" diminue la possibilité d'utilisation d'appoggiatures. Quant au troisième bouton, s'il est plus proche de "as-is-ness", la basse jouera probablement des arpèges. La façon dont *NeurSwing* manipule les traits musicaux nous semble très intéressante et relativement simple, mais le nombre de traits est assez réduit par rapport à ce que l'on peut dire d'une ligne de basse (Cf. section 5.1.1).

Horowitz fait appel à des techniques d'IA distribuée pour concevoir un réseau d'agents utilisé aussi bien pour analyser les traits musicaux d'un passage d'improvisation donnée que pour créer de nouvelles improvisations à partir de traits. Chaque agent est associé à un trait et peut interagir avec d'autres agents. Le calcul de notes se fait en fonction des valeurs des traits stockées dans les agents à un moment donné. Malheureusement, l'auteur ne donne pas suffisamment d'explications sur la façon dont les traits interagissent et les notes sont calculées. Ceci est dû au fait que la partie du programme qui s'occupe de la génération d'improvisations n'est pas encore opérationnelle.

À partir d'un ensemble de contraintes associées à quelques traits, Levitt utilise la programmation par contraintes pour générer les notes. Ces contraintes concernent le morceau entier (Cf. section 3.4.2) et sont spécifiées par un utilisateur. Pour les autres

contraintes (traits) non spécifiées, le programme utilise des valeurs par défaut. Il est néanmoins difficile d'évaluer la portée de cette approche car le répertoire de traits musicaux est très réduit. Pour la basse, par exemple, les traits musicaux sont le style de ligne, la tessiture et l'inversion. De plus, il n'y a de traits portant sur le rythme, toutes les notes étant des noires.

Dans *Pachet*, la notion de stratégie peut servir à représenter les différents traits musicaux que l'on veut retrouver dans les notes du segment courant. Par exemple, la stratégie "jouer en direction ascendante et dans la gamme majeure" concerne respectivement les traits "contour mélodique" et "gamme". Pour chaque segment d'improvisation, des stratégies sont activées selon des données perceptives notamment le contenu de la grille et ce que le programme vient de jouer à l'aide de règles de production de type "si situation X alors activer stratégie Y". Par exemple, si l'intervalle entre l'accord courant et l'accord suivant est plus grand qu'une quarte et l'accord courant a une durée courte, alors on active la stratégie "jouer chromatique". Aussi, à travers un module de perception très rudimentaire qui détecte, le programme peut-il aussi activer des stratégies en fonction de ce qu'un soliste joue. Ainsi, si le soliste joue beaucoup de notes la basse va en faire autant. Ensuite, une fois les stratégies activées, on résout les conflits éventuels entre les stratégies activées, en préférant certaines stratégies plutôt que d'autres. Une fois les conflits éliminés, les stratégies restantes sont "concrétisées" en termes de notes par un ensemble de procédures et des règles de productions. La différence de cette approche d'utilisation des traits musicaux par rapport à *NeurSwing*, *Horowitz* et *Levitt*, est que c'est le programme qui spécifie les propriétés (associées aux traits) que les notes doivent avoir, et ce de façon continue car les stratégies sont suscitées à chaque segment. Pour les autres programmes cités dans cette section, les traits donnés au départ vont influencer le choix des notes de tous les segments d'improvisation, du début à la fin, sauf si l'utilisateur change les paramètres du programme pendant qu'il tourne (c'est le cas de *NeurSwing*). *Pachet*, au contraire, essaye de modéliser le fait que pendant l'improvisation le jazzman peut changer d'avis et vouloir un passage plus dissonant et un autre moins dissonant par exemple. Le problème de ce programme est que le nombre de traits explicitement manipulés est assez réduit (à peu près équivalent à celui de *NeurSwing*). En outre, la représentation des stratégies est très dépendante du moteur d'inférence pour lequel les règles de

production ont été écrites. De plus, les processus de résolution de conflit entre les stratégies et de “concrétisation” des stratégies en notes ne sont pas bien formalisés.

3.4.5. Réutilisation de patterns

Parmi les programmes que nous analysons, la réutilisation de patterns se fait de façon systématique (Hodgson, Ulrich, NeurSwing) ou sporadique (Spector et Flavors Band). Nous restreignons notre analyse à ceux qui font usage systématique des patterns. Remarquons que les approches basées sur les grammaires ont du mal à utiliser les patterns longs (à partir d’une mesure). En effet, la complexité qui résulte de l’augmentation du vocabulaire est grande, à moins que l’on puisse définir des classes de patterns et faire appel à des algorithmes d’appariement puissants. Pour les programmes qui réutilisent des patterns, il nous paraît important d’examiner les questions suivantes : Quelle est la nature des patterns ? Comment sont-ils représentés et indexés ? Quels sont les critères de choix d’un pattern ? Comment un pattern choisi est-il adapté au contexte de réutilisation ?

Tout d’abord il faut préciser que bien que tous les auteurs parlent de “patterns”, les fragments mélodiques ou rythmiques réutilisés ne sont pas forcément des patterns, au sens où ils sont les plus récurrents. Par exemple, dans *Band-in-a-box*, un utilisateur humain peut entrer n’importe quel fragment mélodique pour enrichir la “bibliothèque de patterns” sans avoir à prouver qu’il s’agit d’un “pattern”. Par ailleurs, pour que les programmes que nous étudions dans cette section fonctionnent, ils n’ont pas besoin d’avoir une bibliothèque de fragments mélodiques ou rythmiques contenant uniquement les fragments “très courants” (patterns).

La réutilisation de patterns permet de résoudre certains des problèmes discutés précédemment dans ce chapitre car ces patterns portent en eux des connaissances qui échappent à une formalisation par des règles : le rythme, le style, les accents d’intensité, les contraintes anatomiques de doigté, etc. Plus les patterns sont “épisodiques” au sens où Tulving le définit (Tulving, 1984), c’est-à-dire plus ils correspondent à des transcriptions de vrais accompagnements ou improvisations, plus ils portent de connaissances. Par ailleurs, en codant les dimensions de rythme, de hauteur et d’intensité ensemble, on saisit aussi leur interaction qui est, comme nous venons d’en discuter (Cf. section 3.4.4), difficile à formaliser. Dans cette perspective, les patterns

d'Ulrich sont très pauvres car ils sont simplement des degrés d'une gamme abstraite, sans aucune référence ni aux hauteurs ni aux durées originelles. Les patterns rythmiques utilisés dans *NeurSwing* (16 pour le piano et 22 pour la batterie) sont plus riches mais ils ne font aucune référence aux intensités. Hodgson utilise 67 patterns mélodiques, tandis que *Band-in-a-box* fait usage d'un nombre encore plus grand, mais ces patterns mélodiques ne contiennent pas non plus d'information sur les amplitudes des notes.

En ce qui concerne les accords sous-jacents aux patterns mélodiques, tous les patterns mélodiques de *Band-in-a-box* correspondent à des fragments joués sur un accord isolé. La plupart des patterns de Hodgson doivent être aussi réutilisés accord par accord, mais il semble que Hodgson contient aussi des patterns qui peuvent être utilisés sur le schéma d'accord de type II-V.

Voyons maintenant comment les patterns sont indexés. Nous pensons qu'il serait souhaitable d'indexer les patterns par un vocabulaire important de traits musicaux mais pour l'instant ce n'est pas le cas. Cela nous paraît "naturel", car, d'une part, les traits sont justement des descriptions supplémentaires d'un ensemble de notes, et, d'autre part, les patterns représentent la concrétisation des traits en notes. *NeurSwing* stocke les patterns rythmiques de piano et batterie en fonction de leur taille (durée) et leur densité. Les index employés par Hodgson, dont les patterns sont mélodiques, prennent en compte en plus de la durée et de la densité, des traits musicaux comme le contour mélodique et la dissonance. L'indexation des patterns mélodiques dans *Band-in-a-box* est un peu particulière car il ne s'agit pas de décrire le pattern mais plutôt d'indiquer dans quelles conditions il peut être réutilisé, c'est-à-dire d'indiquer les contraintes portant sur la réutilisation des patterns. C'est pourquoi, dans *Band-in-a-box*, les index sont appelés "masques". Les masques pour les patterns de basse sont affichés dans le tableau 3.4.1. Quand la valeur du masque est "nil", il n'existe pas de restrictions à la réutilisation du pattern mélodique.

Examinons la façon dont un pattern est choisi dans ces trois programmes. Le choix est fondamentalement aléatoire borné par les traits qui servent d'indexation aux patterns. Dans *Band-in-a-box*, le "poids" du pattern a importance particulière. La façon précise dont toutes les informations sont combinées n'est pas présentée dans *Band-in-a-box* ni dans Hodgson pour des raisons "commerciales"; le premier est un logiciel commercialisé depuis longtemps et le deuxième le sera prochainement.

masque	valeurs
Poids (priorité)	1 à 6
Mesure	nil, 1, 2, ..., 7 ou 8 (si mesure = x, le pattern sera réutilisé uniquement aux mesures multiples de x)
Début de l'accord	nil, 1, 2, 3 ou 4 (temps, dans une mesure de 4 temps, où l'accord commence)
Degré de l'accord	nil, I, II, III, IV, V, VI ou VII
Type d'accord	nil, mineur, majeur, dominant, diminué ou demi diminué
Éventail de toniques	nil ou deux hauteurs (par ex. Do à Fa, ou Ré à Mi)
Intervalle avec le prochain accord	nil ou un intervalle (par ex. quinte)
"Push"	0 à 100% (fréquence selon laquelle la première note sera jouée en avance — <i>ahead beat</i>)
Delta "push"	0 à 100% pourcentage d'avance par rapport à un temps

Tableau 3.4.1 - Masques de réutilisation des patterns de *Band-in-a-box*

Pour terminer, il nous paraît important d'aborder la question de l'adaptation des patterns stockés à l'égard du segment d'improvisation courant. A part des modifications triviales comme le déplacement dans le temps, le changement de tempo et les transpositions, aucune autre modification n'est faite. Notons cependant que Hodgson change quelques notes pour assouplir les enchaînements des patterns. *NeurSwing* peut ajouter des notes (appogiatures) dans la ligne de basse ainsi que changer la quinte en quinte diminuée dans les lignes de basse et dans les accords du piano.

3.5. RESULTATS

L'évaluation d'un programme qui génère de la musique est difficile. Cependant, si l'on se place dans une tâche et un style particulier, il est possible de le faire avec assez de finesse. Plus précisément, l'évaluation d'un programme qui génère des improvisations ou des accompagnements en jazz peut se faire selon plusieurs points de vue. La première chose est de vérifier si le programme joue correctement, c'est-à-dire s'il ne fait pas de fautes d'harmonie. Tous les programmes discutés dans ce chapitre jouent correctement. La qualité musicale de ce qui est joué peut être évaluée selon plusieurs critères, le swing étant un des plus importants. Comme nous l'avons vu, certains programmes étudiés ne s'occupent pas vraiment de jouer mais plutôt d'écrire des notes sur une partition. Cela dit, même en se restreignant à la partition, il y a encore beaucoup de critères d'évaluation de la qualité musicale comme la richesse rythmique, harmonique et mélodique, la cohérence, l'adéquation de quelques passages, etc.

Après avoir analysé les exemples musicaux donnés dans divers articles, avoir exécuté certains programmes et écouté des enregistrement disponibles, nous sommes

arrivés à un jugement sur la qualité musicale des programmes que nous analysons dans ce chapitre. Le programme d'improvisation automatique le plus impressionnant actuellement est `Hodgson`. `Band-in-a-box` et `NeurSwing` sont indiscutablement les meilleurs programmes en accompagnement automatique selon pratiquement tous les critères annoncés ci-dessus. `Band-in-a-box` est un programme qui a fait ses preuves puisque cela fait plus de dix ans qu'il est commercialisé. Les autres programmes ont beaucoup de "défauts" surtout en ce qui concerne le rythme et les accents (donc de swing) car il est difficile d'explicitement des connaissances pour ces aspects de la pensée musicale. Par exemple, les lignes de basse de `Johnson-Laird`, `Pachet` et `Levitt` n'ont que des noires et celles de `Cybernetic Composer` n'ont que des noires ou ont un triolet dans le premier temps de la mesure.

Un dernier critère d'évaluation un peu orthogonal aux précédents est l'interaction du programme avec l'environnement. Il existe aujourd'hui deux programmes pouvant le faire : `Hidaka` et `ImprovisationBuilder`. Aucune évaluation de la capacité d'`ImprovisationBuilder` de saisir les intentions du soliste et de réagir convenablement a été rapportée. En revanche, d'après les solistes qui ont joué avec `Hidaka`, ce programme se montre capable de bien "comprendre quelques-unes de leurs intentions".

3.6. CONCLUSIONS

La construction d'un agent rationnel ayant toutes les aptitudes et connaissances que nous avons présentées dans le chapitre 2 est un projet qui implique de nombreux problèmes, chacun étant suffisamment complexe pour justifier une recherche. Cela explique pourquoi un tel agent n'a pas encore vu le jour. Parmi les programmes qui ont un certain succès, certains s'attaquent plus à la perception (`Hidaka`, `Pennycook`, `ImprovisationBuilder`) et d'autres à l'improvisation ou l'accompagnement proprement dits (`NeurSwing`, `Hodgson`).

L'intelligence des programmes n'est pas celle des hommes. L'artificialité, dont il est question en intelligence artificielle, nous autorise à doter les machines de mécanismes de raisonnement qui n'ont guère de plausibilité psychologique et à leur donner des connaissances qui ne sont pas similaires à celles des êtres humains. Selon

Ganascia, “L’intelligence artificielle est une discipline qui permet de prêter une intelligence aux machines pour mieux les maîtriser” (Ganascia, 1993) [couverture]. Néanmoins, parmi les programmes analysés dans ce chapitre, ceux qui ont incorporé le maximum de connaissances des jazzmen produisent les meilleurs résultats musicaux. Parmi toutes les connaissances incorporées, ce qui a été peu exploré et qui nous paraît très important est la prise en compte des traits musicaux et la réutilisation extensive des fragments mélodiques et rythmiques. Ceci pourrait expliquer en partie les réussites de *Band-in-a-box*, *NeurSwing* et *Hodgson*.

Deux autres programmes dont les résultats sont très convaincants confirment notre analyse. Le premier est le célèbre programme de composition de Cope (Cope, 1991) qui réutilise abondamment les “signatures” de compositeurs comme Mozart. Les signatures sont des fragments mélodiques généralisés en termes de suite d’intervalles. L’autre est un programme d’apprentissage d’interprétation (rubato, dynamique, etc.) de partitions de piano classique développé par Widmer (Widmer, 1992). Son succès s’explique en grande partie par le fait que le programme tient compte des structures et des traits musicaux pour induire des règles d’interprétation.

4

4. VUE D'ENSEMBLE DE NOTRE MODELE

Le modèle général d'agent jazzman que nous avons présenté dans le chapitre 1, est certes bien utile en tant que cadre d'analyse et de développement de programmes visant à jouer du jazz. En effet, le modèle de l'agent nous a permis de mieux comprendre les connaissances et le raisonnement des jazzmen et des programmes informatiques jouant du jazz. Mais, pour rendre ce modèle général opérationnel, il est nécessaire de le détailler davantage, ainsi que de le simplifier sous certains aspects. Dans ce chapitre, nous décrirons le fonctionnement de notre modèle d'agent jazzman et les simplifications que nous avons dû apporter au modèle général. Nous montrerons quelles connaissances ont été incorporées à partir des trois activités principales d'un agent : perception, exécution et raisonnement. La présentation de la notion de PACTs, de la Mémoire Musicale et du processus d'improvisation sera faite de façon intuitive par souci de compréhensibilité. Ces sujets seront l'objet d'une étude plus détaillée et formelle dans les chapitres 5 et 6 respectivement.

4.1. PERCEPTION

Comme tous les autres programmes, nous ne tenons pas compte des données visuelles de l'environnement provenant du public ou des musiciens, à l'exception de la grille qui peut être représentée facilement sans perte d'information. Quant à l'écoute, nous avons essayé de trouver un compromis pour prendre en compte l'environnement sans pour autant attaquer directement les grands problèmes posés par l'automatisation de la perception.

4.1.1. La grille

La grille que nous fournissons d'entrée à notre agent bassiste est très complète. Tout d'abord, contrairement à beaucoup de programmes, nous pouvons spécifier toutes sortes d'accords, dont tous ceux qui sont utilisés dans les *Real/Fake Books* (Sher, 1979; Sher, 1991). Ceci est possible parce que nous réutilisons la plate-forme *MUSES* (Pachet, 1994) de représentation de connaissances en musique tonale (Cf. section 6.1). Ensuite, la grille telle que nous l'utilisons n'est pas simplement une suite d'accords. Elle contient d'autres informations comme : le style (blues, standard, etc.); la structure du chorus (AABA, ABAB', ...); le nombre de chorus et le tempo typique, etc. La grille fait aussi référence au thème du morceau, auquel on associe encore d'autres informations comme son style (be-bop, swing, cool, hot, etc.).

4.1.2. La notion de Scénario

Comme nous l'avons exposé auparavant (Cf. section 3.2.1), l'automatisation de l'écoute pose énormément de problèmes. Les programmes existants se concentrent sur l'écoute soit sur le raisonnement. Nous ne voulions pas nous consacrer uniquement à la perception mais nous savions qu'en la négligeant, nous rejeterions les connaissances impliquées dans l'interaction avec l'environnement. Nous avons contourné le problème de l'automatisation de l'écoute grâce à la notion de *Scénario* (Ramalho, Rolland & Ganascia, 1996). Le *Scénario* contient la description symbolique, déjà interprétée, des actions des musiciens et du public qui auront lieu pendant le déroulement du morceau. On fait appel à un utilisateur expert qui écoute un enregistrement préalable d'un groupe de jazz et qui le décrit à l'exception de la basse. Quelques exemples d'*événements* décrits dans le Scénario sont : "le pianiste joue dans le mode dorien", "le batteur joue de plus en plus fort depuis deux mesures", "le soliste joue beaucoup de notes", etc. Une fois finie cette description, le scénario peut être donné conjointement avec la grille à notre agent bassiste pour qu'il commence à jouer.

Le langage que nous utilisons pour décrire les événements du Scénario est inspiré du système Cypher de Rowe (Rowe, 1993) qui, en temps réel, est capable de percevoir toute une série de traits musicaux de base (tessiture, intensité, contour, densité, etc.) ainsi que l'évolution de leurs valeurs dans le temps. Le type le plus courant d'événement est ainsi décrit:


```
EvenBase = (début, durée, auteur, type, valeur, variation) (4.1.1)
  où, auteur = {pianiste, batteur, soliste, public}
      type = {intensité, densité, syncopité, tessiture, gamme}
      valeur = {faible, fort, moyen, etc.}
      variation = {positive, aucune, négative}
```

Les deux premiers exemples cités plus haut sont décrits de la façon suivante: EvBase₁ = (auteur = pianiste, type = gamme, valeur = mode dorien); EvBase₂ = (durée = 16, auteur = batteur, type = intensité, valeur = fort, variation = positive).

En nous inspirant des travaux de Pennycook et son équipe (Pennycook, Stammen & Reynolds, 1993; Stammen & Pennycook, 1993) nous avons étendu le langage de description des événements musicaux de l'environnement pour prendre en compte aussi des épisodes de type "le soliste est en train de jouer tel motif connu", "le batteur joue tel riff", etc. Ces événements sont représentés par

```
EvenMotif = (début, durée, auteur, transcription) (4.1.2)
```

Nous avons aussi voulu intégrer dans le Scénario des descriptions de l'état musical général de l'environnement comme proposé par Hidaka (Hidaka, Goto & Muraoka, 1995). Néanmoins, les cinq traits musicaux globaux employés par Hidaka (excitation, tension, importance de l'accord, substitution d'accord et reprise du thème) nous sont apparus soit trop "pointus" soit redondants. Aussi, en avons nous proposé cinq autres inspirés de NeurSwing et de quelques réflexions de Hodeir sur les tensions centrales du jazz (Hodeir, 1981). Les deux premiers traits qui sont employés pour décrire l'état global de ce que les musiciens jouent sont les suivants : Premièrement, la *température*, qui correspond au *hot/cool* de NeurSwing où "ça chauffe" quand on joue fort, syncopé, avec beaucoup de notes et assez haut (en tessiture). Deuxièmement, la *couleur* qui correspond au *dissonant/consonant* de NeurSwing où "c'est coloré" quand on joue haut, dissonant, avec substitution d'accords, chromatique, et avec des inflexions du son. Ensuite, les trois traits musicaux globaux restants sont la *technique de développement*, la *monotonie* et *l'individualisme* qui correspondent respectivement aux tensions harmonie/mélodie, innovation/répétition et personnel/collectif. Bien que ces trois dernières soient plutôt applicables à un soliste, nous les avons généralisées au groupe dans la mesure où il interagit avec le soliste. Dans le Scénario, quoiqu'il s'agisse des états globaux nous les décrivons comme des événements de la façon suivante :

```
EvenGlobal = (début, durée, type, valeur) (4.1.3)
```

Puisque nous avons mis de côté le problème de la perception pour passer directement à une description accessible de l’environnement, pourquoi ne pas prendre en compte les réactions du public et d’autres données visuelles venant des musiciens eux-mêmes et de l’environnement en général? Outre les événements musicaux, l’utilisateur humain, dont nous avons parlé, peut, à son gré, imaginer des événements liés à l’auditoire. Jusqu’à présent il y en a de deux sortes : ceux qui encouragent ou désapprouvent l’agent comme, respectivement, “le public commence à applaudir” ou “le public dit qu’il n’en peut plus”; et ceux qui peuvent le distraire ou attirer son attention sur quelque chose comme “la police arrive” ou “le batteur fait un signe pour que tu l’écoutes”. Ces événements, qui se résument à “quelqu’un faisant quelque chose”, sont représentés par le n-tuplet suivant :

$$\text{EvenPublic} = (\text{début}, \text{durée}, \text{auteur}, \text{action}) \quad (4.1.4)$$

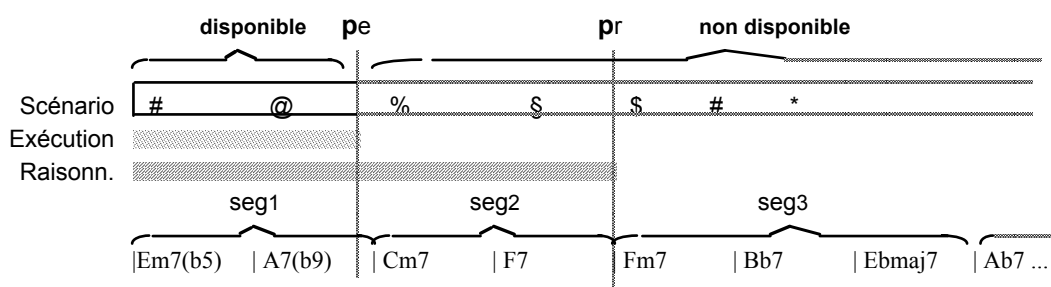


Figure 4.1.1 - Disponibilité des données du scénario

Ce qui est important de souligner est que les événements du scénario ne sont disponibles qu’au temps prévu (Cf. figure 4.1.1). On ne triche pas car les seuls événements du scénario qui peuvent être pris en compte par le module de raisonnement sont ceux dont le début est inférieur ou égal à p_e (position courante du module d’exécution — Cf. section 1.2.3).

La notion de Scénario, à travers une représentation riche des données perceptives, nous permet de concevoir un module de raisonnement qui tient compte de l’environnement sans faire face directement aux problèmes posés par la perception. Cependant, toute simplification a son prix. Le grand défaut du Scénario est qu’il est construit à partir d’un enregistrement préalable d’un groupe de jazz. Cela signifie qu’au lieu d’une interaction directe avec des musiciens humains, notre agent bassiste interagit avec un enregistrement qui est par définition figé. Quoique l’agent ne sache pas ce qui va se passer, l’évolution de l’environnement est donnée une fois pour toutes. Du coup,

on ne peut espérer qu'une véritable interaction entre l'agent et l'environnement ait lieu puisqu'il n'y a pas de rétroaction. L'environnement influence l'agent mais l'agent ne peut exercer d'influence sur l'environnement.

4.1.3. Le module de perception

Chaque module de l'agent a son comportement. Normalement la tâche du module de perception se déroule en deux temps (Cf. Section 3.2.1). D'abord il extrait l'information sous forme de notes ou de phrases (ce qui présuppose la détection du tempo et de la métrique, la quantification temporelle des notes, la détection des frontières des phrases, etc.). Ensuite, les valeurs de quelques traits musicaux peuvent être estimées (par ex. dissonance, densité, etc.). Si un scénario est donné avec toute cette information, la tâche du module de perception est énormément simplifiée.

Ce module doit néanmoins répondre encore à deux types de requêtes de la part du module de raisonnement. Le premier type de requête vise à obtenir des informations sur la grille : quels sont les accords qui tombent dans un tel laps de temps; quelle est la position de tel accord par rapport aux sections de la grille; quel accord suit tel autre; etc. Cela ne pose pas de problème puisque la grille est donnée en entier au module de perception dès le début. Le deuxième type de requête concerne l'extraction du scénario des événements qui tombent dans un intervalle de temps donné. Ces événements, une fois identifiés, sont envoyés aux modules de raisonnement et d'exécution (Cf. figure 1.1.1).

4.2. EXECUTION

Nous avons critiqué les faiblesses de la partie "action" (module d'exécution) des programmes existants, spécialement en ce qui concerne la prise en compte de quelques éléments qui favorisent le swing et des contraintes de vitesse et transposition. Dans notre modèle, une bonne partie de ces problèmes sont déjà résolus par le module de raisonnement. Les notes que le module d'exécution reçoit ont, en plus des débuts, durées et hauteurs, l'information concernant l'amplitude, le timbre et le changement de tempo.

La fonction principale du module d'exécution est de faire jouer par le synthétiseur, les notes fournies par le module de raisonnement à leur début, hauteur, amplitude et timbre spécifiés. La conception d'un tel module implique la construction d'un programme appelé ordonnateur MIDI (*MIDI scheduler*) capable d'établir, selon les protocoles d'interface MIDI, la communication avec le synthétiseur. Nous ne nous attarderons pas sur ce point parce que les problèmes techniques que cela pose n'ont pas d'importance conceptuelle dans le contexte de cette thèse. En outre, l'ordonnateur MIDI que nous utilisons a été développé par Bill Walker (Walker, 1994).

Outre cette tâche de service, notre module d'exécution doit apporter éventuellement quelques modifications aux notes avant de les jouer effectivement.

Le premier type de changement concerne les durées et débuts de notes. En jazz, on ne joue pas comme on écrit. Une partie du phénomène du swing est lié au fait que l'on joue, à peu près, suivant une décomposition ternaire du temps tandis que l'écriture est binaire (Baudoin, 1990) [p. 78-80]. Nous avons introduit, à l'aide de quelques règles simples, ce changement du binaire au ternaire. La figure 4.2.1 montre quelques situations typiques. En réalité, comme le suggère Hodeir (Hodeir, 1995), nous varions la durée de chaque croche en fonction du tempo. Respectivement, les deux notes durent à peu près 0,6 et 0,4 de la noire dans les tempos rapides et 0,8 et 0,2 dans les tempos lents.

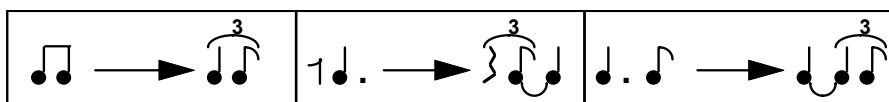


Figure 4.2.1 - Exemples de changement binaire-ternaire.

Un deuxième type de changement tente de combler le manque de réaction de l'agent dû au décalage du raisonnement par rapport à l'exécution ERE (Cf. équation 1.2.1). Par exemple, dans la figure 4.1.1, les événements “#” et “@” ont sans doute été pris en compte pour le calcul des notes du deuxième segment, mais pas les événements “%” et “§”. De même, les événements “\$,” “#” et “*” ne seront pas directement pris en compte dans le raisonnement concernant le troisième segment. Pour cette raison, il est nécessaire de doter le module d'exécution de réactions à l'environnement, ce qui correspond à l'idée “d'arc réflexe” proposée dans les modèles récents d'agent rationnel évoluant sous des contraintes de temps importantes (Georgeff & Rao, 1995; Hayes-Roth,

1995a). L'arc réflexe établit une communication directe entre le module de perception et le module d'exécution, en court-circuitant le raisonnement. Bien entendu, chez les agents qui ont besoin d'établir cette communication inhabituelle, la tâche du module d'exécution qui en découle ne doit pas demander beaucoup de calcul. Dans notre cas, nous n'avons mis en œuvre qu'un type de modification qui peut sûrement être réalisé en respectant les contraintes de temps : le changement d'intensité globale (*loudness*). Le besoin d'un changement d'intensité est détecté à l'aide de règles de production très simples. Ces règles visent à garantir, d'une part, que le bassiste ne joue pas plus fort que le soliste ni que le pianiste et, d'autre part, qu'il joue dès que possible à la même intensité que le batteur.

Un troisième type de changement vise à améliorer le passage (ou "collage") entre segments d'improvisation. La fonction principale des dernières notes d'une phrase mélodique de basse jouée sur un accord donné est de préparer la transition vers la phrase suivante. Il se trouve que lorsque le module de raisonnement calcule les notes du segment courant, il ne connaît pas encore les premières notes du prochain segment et, pour cette raison, le choix des dernières notes du segment courant peut ne pas être adéquat. Alors, le module de raisonnement calcule les notes du segment courant, les envoie au module d'exécution et essaie de garantir une bonne transition par un choix approprié des premières notes du prochain segment. Le problème est qu'il ne réussit pas toujours, car les premières notes jouées sur un accord donné sont soumises à d'autres contraintes. Puisque les notes du segment précédent ont été déjà envoyées au module d'exécution, c'est lui qui doit les changer dès qu'il recevra les notes du prochain segment. Plus précisément, le module d'exécution change la dernière note du segment selon un certain nombre de règles simples comme celle qui vise à éviter que la dernière note d'un segment soit la même que la première note du segment subséquent. La figure 4.2.2 montre un exemple où le module d'exécution a changé les dernières notes du segment 1 et 2 (respectivement Fa bémol et La bémol) selon la règle que nous venons de citer. Si ces changements n'avaient pas eu lieu, le passage de ligne de basse serait celui de la figure 4.2.3.

The image shows a musical score for a bass line, divided into four segments. Segment 1 (measures 41-44) contains chords C min 7, F 7, Bb maj7, and Eb maj7. Segment 2 (measures 45-48) contains chords A halfDim7, D 7, G min 7, and G min 7. The final note of each segment is adjusted to align with the start of the next segment.

Figure 4.2.2 - Passage de ligne de basse où le module d'exécution ajuste la dernière note des segments

The image shows a musical score for a bass line, identical to Figure 4.2.2, but without the adjustment of the final notes. The segments are labeled 'segment 1' through 'segment 4' with their respective chords: C min 7, F 7, Bb maj7, Eb maj7, A halfDim7, D 7, G min 7, and G min 7.

Figure 4.2.3 - Passage de ligne de basse sans ajustement de la dernière note des segments

Un dernier type de changement est celui du tempo. Selon les événements du Scénario, l'agent décide de jouer *behind beat* ou *ahead beat*. Comme dans *NeurSwing*, plus ça “chauffe” (événement de type “température haute”), plus on joue *behind beat*. Ces changements, bien que simples, contribuent au swing (Collier & Collier, 1994).

4.3. RAISONNEMENT

4.3.1. Improvisation vue comme une résolution de problèmes

La tâche d'improvisation ou d'accompagnement peut-elle être assimilée à une activité de résolution de problèmes ?

Quelques chercheurs (Johnson-Laird, 1992; Pressing, 1988) qui se sont penchés sur cette question soutiennent que de telles activités ne peuvent être formalisées de façon adéquate dans un cadre de résolution de problèmes. La difficulté centrale est d'établir des buts bien définis et figés (et donc un critère d'arrêt) pour la résolution. En effet, au départ de l'improvisation, les musiciens semblent avoir uniquement une impression

vague et des idées parfois contradictoires sur ce qu'ils veulent jouer. Ce n'est qu'au cours de l'improvisation que les idées se précisent. Certains chercheurs ont effectivement eu du mal à obtenir de bons résultats musicaux en explorant ce cadre de formalisation. C'est notamment le cas du système de composition de Vincinanza (Vincinanza & Prietula, 1989) basé sur SOAR (Laird, Newell & Rosebloom, 1987). À notre avis, les critiques exprimées par Johnson-Laird et Pressing sont basées sur une formulation assez naïve de ce qu'est un problème musical : étant donné un ensemble de mesures vides, le problème musical consiste à les remplir avec des notes qui satisfont certains critères. Comme ces critères sont justement difficiles à déterminer car ils se raffinent et changent avec le temps, le programme ne saura jamais s'il a atteint le but.

Nous soutenons qu'il est possible de formaliser l'activité de création musicale en termes de résolution de problèmes sous la réserve que l'on formule le problème autrement. Nous proposons donc une formulation originale selon laquelle le problème de la génération d'une improvisation ou d'un accompagnement *par un agent rationnel* est triple. Le premier problème est de déterminer le *segment* (laps de temps) pour lequel l'agent va calculer les notes. Le deuxième problème est justement d'avoir la spécification la plus précise, cohérente et complète possible des *critères* que les notes du segment d'improvisation doivent respecter. Le troisième problème est de concrétiser ces critères en termes de *notes*. Ces trois problèmes se posent continuellement tout au long du jeu de l'agent jazzman. Autrement dit, on doit construire un agent rationnel qui sache bien poser le problème et ensuite le résoudre. Ce n'est pas une conclusion étonnante puisque la réalisation d'une œuvre d'art commence normalement par la détermination des critères que l'œuvre doit respecter.

À partir de ces réflexions nous avons eu l'idée de construire un agent jazzman (Ramalho & Ganascia, 1994b) dont le fonctionnement suit trois étapes principales. D'abord, l'agent détermine le *segment d'improvisation* selon des schémas d'accords qui apparaissent sur la grille. Ensuite, il se fixe un certain nombre de critères (liés à un répertoire de traits musicaux) concernant les notes à jouer en fonction de son état interne (Humeur — Cf. section 4.3.5) et les données perceptives suivantes : les événements récents du Scénario, les accords de la grille (surtout les accords qui composent le segment d'improvisation courant) et la ligne de basse jouée jusqu'à présent (en particulier les notes du dernier segment). Ces critères étant, le cas échéant,

contradictoires et incomplets, l'agent essaye de les combiner et d'en enlever les conflits. Enfin, quand les critères deviennent suffisamment clairs, ils évoquent chez notre agent un souvenir d'un fragment mélodique mémorisé dans le passé. En d'autres termes, une fois que les critères sont bien définis et cohérents, l'agent cherche, dans ce que nous appelons la *Mémoire Musicale* (Ramalho & Ganascia, 1994a), un fragment mélodique de ligne de basse qui satisfait le mieux à ces critères. Ce modèle d'agent jazzman suit donc les principales conclusions auxquelles nous sommes arrivés à partir de notre travail d'explicitation de connaissances et de réflexion sur les programmes actuels décrits dans les chapitres précédents. Notre choix de segmentation découle de l'idée que les jazzmen progressent plutôt "phrase par phrase" que "note par note", et qu'une segmentation en schémas d'accords est, parmi les segmentations actuellement faisables, celle qui "colle" le mieux avec les phrases (Cf. sections 3.4.2 et 4.3.2). En outre, les justifications que les jazzmen nous ont données sur leurs choix nous ont fait comprendre que le choix des notes devrait passer par un certain nombre de critères liés aux traits musicaux. Aussi, l'influence de l'environnement peut-elle être prise en compte pour la fixation de certains critères. Enfin, la constatation que de vrais (au sens épisodique) fragments mélodiques représentent, en extension, énormément de connaissances des jazzmen nous ont amenés à voir un fragment comme la "matérialisation" même d'un ensemble de critères.

Dans la suite de ce chapitre nous approfondissons davantage chacune de ces trois grandes étapes du raisonnement de notre agent : la détermination du segment d'improvisation; la fixation des critères portant sur les notes du segment et le calcul de ces notes. Par souci de clarté, nous remettons néanmoins la présentation formelle de ces étapes aux chapitres 5 et 6.

4.3.2. Segmentation de la grille et granularité du raisonnement

Comme nous l'avons exposé précédemment (Cf. section 3.4.2), la détermination du segment d'improvisation sur lequel le programme va travailler est très délicate. D'une part, la solution la plus "plausible" qui consisterait à choisir des phrases de taille quelconque qui puissent s'enchaîner, et ce indépendamment de "points de repère" sur la grille, pose beaucoup trop de difficultés. D'autre part, il faut que le segment établisse un bon compromis pour la "granularité" du raisonnement de l'agent.

Nous proposons une segmentation originale basée sur les *schémas d'accords* de type II-V, V-I, II-V-I, VI-II-V-I, etc. Autrement dit, notre agent génère sa ligne de basse par segments d'improvisation qui correspondent aux schémas d'accords de la grille en question. Ce choix particulier a, comme tout autre choix, des avantages et des inconvénients.

Le premier avantage est la plausibilité musicale. Les schémas d'accords jouent un rôle important dans la pensée d'un jazzman, soit en fournissant une analyse structurelle de la grille (Baudoin, 1990; Mouton & Pachet, 1995), soit en permettant une vue stéréotypée des séquences d'accords connues, soit enfin, en servant d'élément d'indexation de phrases mélodiques ou rythmiques. La plupart des patterns de solistes identifiés par les musicologues (Kernfeld, 1983; Owens, 1974), ou proposés dans des méthodes de jazz (Baker, 1980; Coker, 1970) sont en effet énumérés en fonction des schémas d'accords sous-jacents. Néanmoins, ce choix s'appuie sur l'hypothèse selon laquelle les phrases, l'élément structurel principal d'une improvisation ou d'un accompagnement, sont ancrées à des structures harmoniques de type schéma d'accords. Évidemment, cette hypothèse n'est qu'une simplification car les phrases, surtout celles d'un soliste, débordent souvent les frontières des schémas d'accords, y compris pour le jeu du bassiste. Toutefois, selon nos analyses des corpus de basse, à peu près 80 % du temps, les phrases jouées sur un schéma d'accords forment une unité. L'autre avantage est que, tout comme le choix de segments de taille égale à une ou deux mesures par exemple, les tailles des schémas sont un compromis raisonnable de "granularité" (Cf. section 3.4.2). Les schémas d'accord que notre programme est capable de repérer durent en moyenne sept temps (presque deux mesures de quatre temps).

Nous avons conçu trois bases de règles qui, couplées à un moteur d'inférence de premier ordre en chaînage avant, permettent l'identification automatique de ces schémas à partir d'une analyse harmonique particulière. La reconnaissance des schémas se fait en trois étapes. Premièrement, on identifie tous les schémas possibles d'un lexique donné. Ce lexique, comme nous expliquerons plus tard, dépend directement des fragments mélodiques que l'agent connaît, c'est-à-dire des fragments stockés préalablement dans la Mémoire Musicale de l'agent (Cf. section 6.3). Deuxièmement, on élimine les conflits entre les schémas qui se superposent à l'aide de quelques règles de préférence concernant les types de schéma et leurs tailles. Troisièmement, on

"remplit les trous" entre les schémas, en considérant les accords restants comme un type spécial de schéma : le schéma d'un unique accord. La segmentation minimale est celle où chaque accord constitue un segment.

Selon le lexique, il y a plusieurs manières de segmenter une grille. La figure 4.3.1 montre une segmentation possible de STELLA BY STARLIGHT dont la grille originelle est présentée dans la figure 1.2.1.

1	E min7(b5)	A 7	C min 7	F 7
5	F min 7	Bb 7	Eb maj7	Ab 7
9	Bb maj7	Emin7(b5) A7	D min7	G min7 C7
13	F maj7	G min7 C7	Amin7(b5)	D 7
17	G 7	G7	C min7	C min7
21	Eb min7	Eb min7 Ab 7	Bb maj7	Bb maj7
25	E min7(b5)	A 7	D min7(b5)	G 7
29	C min7(b5)	F 7	Bb maj7	Bb maj7

Figure 4.3.1 - Exemple d'une segmentation possible de STELLA BY STARLIGHT

4.3.3. Fixation des critères concernant les notes à jouer

Pour pouvoir déterminer et manipuler les critères selon lesquels les notes d'un segment de grille donné seront calculées, nous avons repris la notion de "stratégie" proposée par F. Pachet (Cf. section 3.4.4) (Pachet, 1987; Pachet, 1990). Nous avons préféré désigner cette notion par le mot "PACT" (Potential ACTION) parce que le mot "stratégie" avait déjà beaucoup d'autres connotations et que les stratégies étaient définies comme des *actions potentielles* qui sont suscitées à un moment donné chez le jazzman concernant les caractéristiques des notes qu'il va jouer dans un futur proche ou lointain. Dans la suite de cette section nous allons approfondir davantage la notion de PACTs et montrer comment nous l'avons enrichie pour pouvoir définir un cadre d'utilisation qui répondait à nos besoins.

4.3.3.1. *Activation et sélection des PACTs*

L'idée de favoriser une représentation explicite des actions musicales confère aux PACTs quelques caractéristiques intéressantes. Les PACTs permettent une description partielle ou complète de ce que le musicien a l'intention de jouer. En effet, les PACTs peuvent représenter des actions au niveau des notes (PACTs e, f et k du tableau 4.3.1) jusqu'à des niveaux plus abstraits (PACTs a, b, c, d, h, i, j et l). Ceci correspond à la supposition que, selon la situation, le musicien peut avoir des idées plus ou moins précises de ce qu'il va jouer. Il est utile de souligner que, parmi les PACTs qui décrivent précisément les notes à jouer, certaines le font en fonction d'un fragment mélodique déjà joué, comme l'illustre la PACT "e" du tableau 4.3.1. Ceci est important car les répétitions et les variations d'un motif ou d'une phrase sont des techniques courantes de structuration du discours musical. Pour nous, l'intérêt plus immédiat de la notion de PACTs est que certaines PACTs peuvent être vues comme l'incarnation des "critères" que nous voulons manipuler associés à un début et une durée. Par exemple, les PACTs a, b, c, d, h, i, j et l fixent des critères concernant, respectivement, les traits musicaux suivants : gamme utilisée et contour mélodique, variation d'intensité, inversion, style de la ligne (arpège), densité et "syncopacité", swing, contraste et contour mélodique.


<p>a - "pendant cette mesure, jouer la gamme majeure en direction descendante"</p> <p>b - "jouer de plus en plus fort jusqu'à la fin du chorus d'improvisation";</p> <p>c - "jouer la tonique au premier temps jusqu'à la fin"</p> <p>d - "jouer un arpège sur cet accord"</p> <p>e - "jouer la dernière phrase en la transposant un ton plus haut"</p> <p>f - "jouer maintenant ces notes </p> <p>g - "jouer maintenant ce rythme ↓ ↓ ↓× ↓ "</p> <p>h - "jouer beaucoup de notes et très syncopé la prochaine fois que l'on viendra sur ces accords"</p> <p>i - "swinguer d'ici deux mesures jusqu'à la fin"</p> <p>j - "jouer différemment cette section"</p> <p>k - "ne pas jouer à partir de maintenant jusqu'au début du bridge"</p> <p>l - "jouer dans la direction ascendante pendant ces deux mesures"</p>

Tableau 4.3.1 - Exemples de PACTs

Comme nous l'avons déjà exposé dans la section 3.4.4, pour chaque segment d'improvisation (un segment étant égal à un accord la plupart du temps), le raisonnement du programme de F. Pachet suit trois étapes. D'abord, toutes les PACTs concernant le segment sont *activées*. Ensuite, parmi les PACTs activées, le programme *élimine* celles qui sont en *conflit*. Enfin, l'ensemble de PACTs restant est *concrétisé* en termes de notes. Ce type de "cycle d'inférence" est donc cohérent avec le mode de fonctionnement d'un agent jazzman tel que nous l'avons proposé : segmentation, fixation de critères (les PACTs) et calcul des notes à partir des critères. Néanmoins, ce modèle d'utilisation des PACTs présente des insuffisances. Dans la suite de cette section, nous allons formuler quelques critiques à cet égard et montrer les solutions et les améliorations que nous avons apportées à l'activation des PACTs. Dans la prochaine section, nous ferons de même pour la résolution de conflits et le calcul des notes à partir des PACTs.

Le mécanisme d'inférence utilisé dans le système de F. Pachet pour l'activation des PACTs à savoir les règles de production de type "si une situation *s* est reconnue, alors activer PACT *p*" couplées à un moteur en chaînage avant, nous semble très pertinent. Outre la facilité de mise en œuvre (Watterman & Hayes-Roth, 1978), l'emploi de règles de production est bien adapté lorsqu'il s'agit de réagir à un environnement dynamique (Laird, Newell & Rosebloom, 1987; Russel & Norvig, 1995). Dans notre programme, nous avons simplement élargi l'éventail de données perceptives selon lesquelles les PACTs peuvent être activées. En ce qui concerne la grille, en plus d'activer des PACTs selon les accords sous-jacents au segment d'improvisation courant (par ex. si l'agent est sur un accord altéré, alors activer la PACT "jouer chromatique"), nous en activons aussi selon la structure du morceau (par ex. si on est au début de l'improvisation et si jusqu'à présent on joue avec le rythme basé sur la blanche, alors activer la PACT "jouer avec le rythme basé sur la noire (*walking*)"), le thème du morceau (par ex. s'il y a des creux dans le thème, alors activer PACT "jouer plus de notes pendant les creux du thème"), le style de la grille (par ex. s'il s'agit d'une ballade, alors activer PACT "jouer avec peu de notes et syncopé"), etc. Nous activons aussi des PACTs en fonction de ce que l'agent a déjà joué (par ex. si l'agent joue dans la direction ascendante en *stepwise* depuis deux mesures et si la tessiture est déjà haute, alors activer PACT "faire un *drop*"), et ce de façon plus généralisée. Dans le programme de F. Pachet l'analyse se restreint uniquement à la dernière note jouée.

Enfin, nous activons des PACTs selon les derniers événements du scénario (par ex. si le batteur joue doucement, alors activer PACT “jouer doucement”, ou si le soliste utilise des chromatismes, alors “jouer de façon dissonante”).

Les PACTs ont, comme toute action, une dimension temporelle (début et durée) et qu'elles soient définies comme des actions potentielles concernant les notes qui seront jouées dans un futur *proche* ou *lointain*. Cette caractéristique a été peu explorée dans le programme de F. Pachet. Toutes les PACTs étaient activées avec un laps de temps égal à celui du segment d'improvisation courant. Cela signifie qu'à chaque segment il ne restait plus aucune “idée” (critère) du segment d'avant et que le programme n'a jamais “d'idées” concernant le futur lointain.

Nous avons pensé qu'il était très important de tirer davantage profit de la dimension temporelle des PACTs, et ce pour deux raisons. La première raison tient au fait qu'il est utile d'activer, à un instant donné, des PACTs qui ne seront prises en compte que plus tard (par ex. PACTs h et i du tableau). Indépendamment de sa plausibilité cognitive (les idées d'un jazzman ne se restreignent vraisemblablement pas au futur immédiat), ce procédé peut atténuer le problème de filtrage du passé (Cf. section 1.2.5). En effet, au lieu d'essayer d'identifier ce qui est pertinent parmi tout ce que l'agent a déjà joué, il est plus simple “de fixer des marques” pour le futur en fonction de ce que l'agent est en train de faire. Par exemple, supposons que l'agent veuille créer un contraste rythmique pour le “pont” (troisième section d'un morceau de structure AABA). Quelques mesures avant le pont, il active les PACTs “jouer avec très peu de notes jusqu'au début du pont” et “jouer avec plus de notes pendant le pont”. Quand il sera au début du pont, il retrouvera naturellement la PACT préalablement activée. La deuxième raison est que l'activation de PACTs ayant une longue durée peut minimiser l'effet de la “fragmentation” du processus de raisonnement de l'agent jazzman. De fait, dans la mesure où des PACTs sont activées pour durer assez longtemps (PACTs b, c, i et k), elles peuvent “influencer” le calcul des notes de plusieurs segments d'improvisation, rendant le passage plus homogène.

Afin d'obtenir cette “flexibilité” dans l'activation des PACTs, il nous a fallu introduire une étape supplémentaire que nous avons appelée *sélection* des PACTs pertinentes vis-à-vis du segment courant (Ramalho & Ganascia, 1994b). À chaque segment, de nouvelles PACTs sont activées et sont “envoyées” à la mémoire de travail

de l'agent où elles rejoignent d'autres PACTs activées précédemment. Ensuite, l'agent sélectionne parmi ces PACTs (activées récemment ou auparavant), celles dont la durée de vie se superpose au segment en question. Ce sont les PACTs sélectionnées qui sont considérées comme critères pour le calcul des notes du segment, c'est-à-dire pour la recherche d'un fragment mélodique dans la Mémoire Musicale. Dans l'illustration de la figure 4.3.2, les cinq PACTs pertinentes pour le segment sont les 74, 75, 76, parmi les quatre activées récemment, et les PACTs 21 et 2, parmi les quatre plus anciennes, c'est-à-dire celles dont la date de création est plus ancienne.

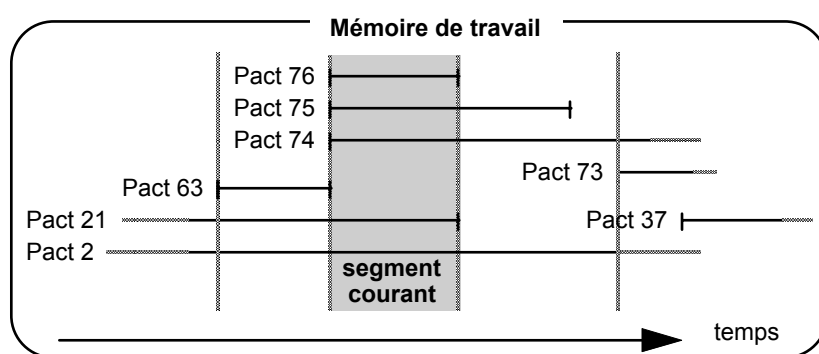


Figure 4.3.2 - Sélection des PACTs activées

4.3.3.2. Assemblage des PACTs

Une caractéristique de l'activation des PACTs est qu'elle génère très souvent des PACTs "incompatibles" (en conflit). Par exemple, les PACTs *a* et *l* sont incompatibles au niveau du trait "contour mélodique" pendant une mesure, tandis que les PACTs *a* et *d* le sont au niveau du style de ligne de basse (arpège ou *stepwise*). Les PACTs *f* et *g* sont incompatibles au niveau du rythme. Les conflits entre les PACTs s'expliquent surtout par le fait que chaque PACT est activée selon des données perceptives différentes et relativement indépendantes les unes des autres (événements du Scénario, accords sous-jacents, ce que l'agent vient de jouer, position sur la grille, etc.). De toutes façons, il n'est pas surprenant que les règles d'activation, c'est-à-dire les "règles" qui fixent les critères des notes à jouer, soient un peu contradictoires (Cf. section 2.3.2). Au-delà du jazz, dans des disciplines comme le contrepoint ou l'harmonisation des chorales, quand quelqu'un veut construire un programme avec les "règles" identifiées par les méthodes de composition importantes (Fux, 1725; Schoenberg, 1943), il se rend compte qu'il reste

encore des ambiguïtés et des contradictions (Schottstaedt, 1984; Thomas, 1985). Pour ces raisons, nous avons adopté l'idée de F. Pachet d'activer toutes les PACTs possibles (en fonction des données perceptives) et ensuite seulement de résoudre les éventuels conflits. L'intérêt de cette stratégie d'activation est que, si nous sommes capables d'explicitier les "règles" fixant les critères sur les notes à jouer, nous pouvons les exprimer facilement pour activer des PACTs, et ce, indépendamment des conflits éventuels entre ces connaissances.

Dans le programme de F. Pachet, les conflits entre les PACTs activées étaient résolus de la façon suivante : on examinait les PACTs deux à deux et, lorsqu'un conflit était détecté, on éliminait une d'entre elles. Malheureusement, pour choisir la PACT à éliminer, des méthodes employées étaient plutôt ad hoc. Une PACT était définie comme ayant un *début*, une *durée*, un *nom*, un *niveau* (bas ou haut) et un *type* (règle, paquet, objet ou fait). Les PACTs de niveau bas étaient celles qui pouvaient vraiment produire des notes (par ex. "jouer un arpège" ou "jouer la dernière phrase transposée d'un ton") tandis celles de niveau haut ne pouvaient qu'influencer le calcul des notes (par ex. "jouer fort" ou "jouer dans une tessiture haute"). Plus précisément, une fois que les notes avaient été calculées, elles subissaient des "transformations" selon l'information contenue dans les PACTs de niveau haut. Par exemple, les amplitudes des notes produites en fonction de la PACT "jouer un arpège" peuvent être augmentées à cause de la présence de la PACT "jouer fort". Parmi les PACTs activées, il ne pouvait donc y avoir qu'une PACT de niveau bas. Lorsqu'il y en avait deux, le programme choisissait aléatoirement une d'entre elles. Quant à la résolution de conflits entre les PACTs de niveau haut, elle était traitée cas par cas de façon ad hoc. Par exemple, entre "jouer en direction ascendante" et "jouer en direction descendante", le programme préférait garder la première.

Nous avons détecté une autre faiblesse dans la façon dont les PACTs étaient utilisées dans le programme de F. Pachet. Si les PACTs sont censées être des actions, on pourrait bien penser à les "combiner" (celles qui sont compatibles, naturellement) avant de commencer à déterminer les notes à jouer. L'intérêt de la combinaison est qu'une action combinée avec une autre peut impliquer une troisième action. En combinant les PACTs, on pourrait donc avoir plus de critères pour le choix des notes. Par exemple, "jouer assez dissonant" avec "jouer des arpèges" pourrait donner lieu à

“jouer assez dissonant, avec des arpèges, et en évitant la tonique au premier temps”. En effet, pour un bassiste, une façon de jouer “des arpèges assez dissonants” est d’éviter la tonique au premier temps. Cette idée de combinaison était absente du programme de F. Pachet. Dans le langage initial employé pour représenter les PACTs, il est impossible d’exprimer les PACTs a et h du tableau 4.3.1 car elles concernent plus d’un trait musical à la fois.

Pour répondre à ces lacunes nous avons essayé de trouver une formulation, en termes de résolution de problèmes, de l’étape qui mène des PACTs activées aux notes. Ainsi, nous avons défini deux propriétés des PACTs : *jouabilité* et *combinabilité*.

Une PACT peut être plus ou moins *jouable*, suivant que l’information qu’elle contient est plus ou moins suffisante pour que l’on détermine les notes à jouer. La PACT f du tableau 4.3.1 est plus jouable que la PACT g, qui elle-même est plus jouable que la PACT d, qui est plus jouable que la PACT c, qui enfin est plus jouable que la PACT j. La propriété de jouabilité d’une PACT est très importante car elle établit une relation d’ordre qui intervient dans la résolution de conflits entre deux PACTs incompatibles, la PACT la plus jouable pouvant être préférée. Une PACT est (*complètement*) *jouable* si elle indique, directement ou indirectement, toutes les notes à jouer pour la durée de son laps de temps, chaque note ayant tous ses attributs bien spécifiés (Cf. Section 1.2.4, formule 1.2.3). Les PACTs e, f et k du tableau 4.3.1 sont jouables. D’autre part, selon leur compatibilité, deux PACTs peuvent se *combiner* pour générer une PACT plus jouable. Par exemple, la PACT l peut se combiner avec la PACT d générant, dans un contexte de G majeur, les notes sol, si, ré. De même la combinaison des PACTs g et c et des PACTs b et i donne respectivement “jouer ce rythme en plaçant la tonique au premier temps” et “jouer de plus en plus fort et avec swing d’ici deux mesures jusqu’à la fin du chorus d’improvisation”. En somme, à chaque combinaison, on “enrichit la description de la PACT” et on donne plus de critères pour le choix des notes à jouer.

C’est l’idée qu’au fur et à mesure des combinaisons on peut produire une PACT jouable, qui nous a permis de proposer une méthode de résolution de problèmes que nous appelons méthode *d’assemblage des PACTs*. Les PACTs qui ont été sélectionnées pour le segment courant composent l’état initial de l’espace du problème (appelé espace d’assemblage). L’état final (le but) est une PACT (totalement) jouable. Cela signifie

que, par le même processus d'assemblage des PACTs, notre programme raffine les critères (PACTs activées) et les concrétise sous forme de notes (PACT jouable). Plus précisément, un nouvel état de l'espace d'assemblage peut être atteint par l'application de trois opérateurs : enlève, combine et propage (Cf. figure 4.3.3). Le choix des opérateurs obéit à une stratégie opportuniste, extension du principe de rationalité de l'agent, selon laquelle on cherche le chemin le plus court pour atteindre le but. Il n'y a pas de "retour arrière" (*backtracking*). Tout le processus d'assemblage est réalisé à l'aide d'une base de règles de production et d'un moteur d'inférence en chaînage avant.

L'opérateur *enlève* est utilisé pour résoudre des conflits entre deux PACTs incompatibles. Dans ce cas, celle qui est plus jouable est préférée. Par exemple, entre "jouer un arpège ascendant" et "jouer dans la direction descendante", on préfère la première PACT. Quand la jouabilité n'est pas suffisante pour départager deux PACTs, on regarde leur date de création (le moment où une PACT a été activée). Les PACTs plus "jeunes" sont préférées car elles peuvent représenter une réaction urgente à l'environnement. Par exemple, entre la PACT "jouer très doucement" activée au début du jeu et la PACT "jouer très fort" activée récemment, on préfère la dernière. Si ceci n'est toujours pas suffisant, on tranche selon la durée de vie : on choisit la PACT qui va durer plus longtemps car elle peut, a priori, offrir une plus grande homogénéité de la ligne de basse.

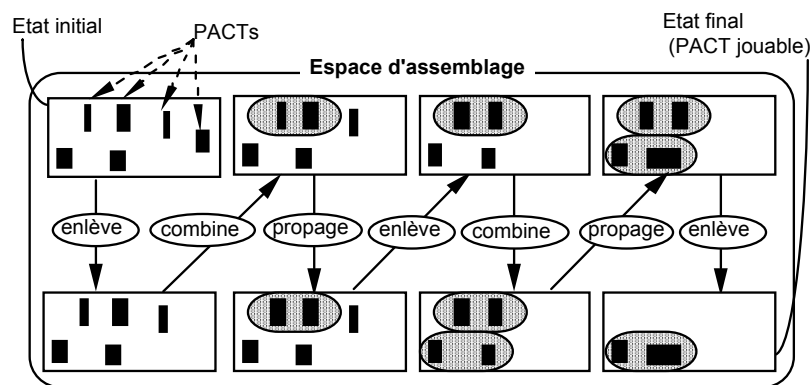
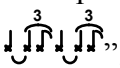


Figure 4.3.3 - Illustration graphique de l'assemblage des PACTs

L'opérateur *combine* fusionne l'information contenue dans deux PACTs compatibles dans une nouvelle PACT. C'est un opérateur très simple une fois que l'on connaît la compatibilité des PACTs en question. Par exemple, la combinaison des PACTs "jouer avec un style rythmique basé sur la croche" et "jouer avec une faible

dissonance”, donne “jouer avec un style rythmique basé sur la croche et avec une faible dissonance”.

L’opérateur *propage*, un peu plus complexe, est applicable à une seule PACT lorsque celle-ci contient des informations qui peuvent être “propagées” pour la rendre plus jouable, afin “d’enrichir” sa description. C’est typiquement le cas des PACTs générées par combinaison : les informations des deux PACTs peuvent, ensemble, rendre possible la spécification d’autres traits. Dans l’exemple de combinaison cité plus haut (“jouer avec un style rythmique basé sur la croche et avec une faible dissonance”) l’opérateur “propage” pourrait être appliqué pour spécifier le trait musical concernant une technique particulière de construction de ligne de basse (répéter les notes) qui “découle” de ces deux exigences (style rythmique basé sur la croche et faible dissonance). Ainsi, après la propagation, la PACT deviendrait “jouer avec un style rythmique basé sur la croche, avec faible dissonance et en utilisant des notes répétées”. De même, la combinaison des PACTs “jouer assez dissonant” et “jouer des arpèges” pourrait via une propagation donner lieu à la PACT “jouer assez dissonant, avec des arpèges, et en évitant la tonique au premier temps”. C’est à travers cet opérateur que les procédures de calcul des notes ou d’autres traits intermédiaires sont déclenchées. Par exemple, si l’on connaît le style rythmique, la densité et la “syncopité”, comme dans la PACT “jouer avec un style rythmique basé sur la noire et très syncopé, pendant cette mesure de 4 temps”, alors on pourrait déclencher le calcul du fragment rythmique et obtenir une PACT “jouer ”.

Dans le prochain chapitre nous présenterons le langage qui permet la représentation de notre répertoire de PACTs ainsi que la formalisation de toutes les opérations les impliquant (activation, combinaison, jouabilité, etc.).

En conclusion de cette section, nous pouvons résumer les différentes étapes du cycle d’inférence de notre agent. Comme l’illustre la figure 4.3.4, les trois étapes de notre formulation initiale (segmentation, fixation des critères et calcul des notes) se transforment en quatre étapes. En premier lieu, l’agent détermine un segment adéquat selon les schémas d’accords composant la grille. En deuxième lieu, il active un nombre variable de PACTs en fonction de différentes données perceptives. Ces PACTs fonctionnent comme des critères (liés à un répertoire de traits musicaux) concernant le

choix des notes d'un segment proche ou lointain dans le futur. En troisième lieu, les PACTs qui se superposent au segment courant sont sélectionnées. En dernier lieu, les PACTs sélectionnées sont assemblées de façon à combiner le maximum de critères, tout en évitant les conflits. À chaque fois que deux PACTs se combinent, elles donnent lieu à une PACT plus jouable et à la fin du processus d'assemblage on trouve une PACT (totalement) jouable.

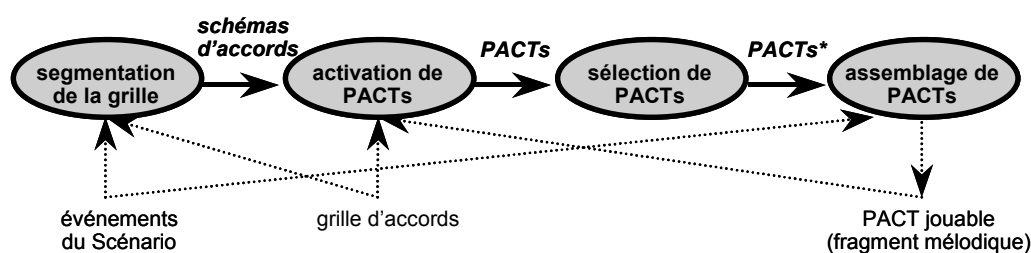


Figure 4.3.4 - Étapes du raisonnement de notre agent. Les données perceptives (entrées) sont la grille, les événements du Scénario et les PACTs déjà jouées.

4.3.4. La notion de Mémoire Musicale : une base de cas pour des fragments mélodiques de ligne de basse

Comme nous l'avons discuté dans la fin du chapitre 2, rapprocher le raisonnement à partir de cas de la réutilisation de fragments mélodiques ou rythmiques en jazz va de soi. En effet, ces fragments sont de vrais cas, au sens épisodique du terme, et ils peuvent être réutilisés dans des conditions similaires. Le raisonnement à partir de cas (Kolodner, 1993; Schank & Riesbeck, 1989) est justement une technique de raisonnement adéquate pour des domaines où l'expérience codée sous forme de cas vécus dans le passé joue un rôle important. Plus précisément, on stocke dans une base de cas des problèmes concrets déjà rencontrés dans le passé, ainsi que les solutions apportées. Devant un nouveau problème, on cherche dans cette base le problème le plus similaire au problème actuel et on essaie d'adapter l'ancienne solution au problème actuel. Ainsi, après notre première démarche d'acquisition de connaissances, nous avons eu l'idée de concevoir une base de cas, que nous avons appelée Mémoire Musicale (Ramalho & Ganascia, 1994a), contenant des fragments mélodiques de lignes de basse obtenus à partir de transcriptions d'enregistrements de bassistes de jazz. Ces fragments correspondent à des passages joués sur des schémas d'accords, puisque c'est sur ces structures que notre agent segmente la grille. Puisque les PACTs permettent aussi une

description complète de notes à jouer, nous avons décidé de représenter les fragments mélodiques de la Mémoire Musicale comme des PACTs jouables. Nous unifions ainsi la représentation des “traits musicaux” et des fragments mélodiques.

Néanmoins, pour rendre opérationnelle cette Mémoire Musicale il nous faut répondre à un certain nombre de questions. Quel est le rôle exact d’une telle Mémoire Musicale ? Autrement dit, quand l’agent devrait-il chercher des cas dans la Mémoire Musicale ? Comment indexer un cas pour pouvoir le retrouver, rapidement et avec exactitude, lorsqu’une situation similaire à celle où il a été joué se présente ? Comment prendre en compte les interférences d’un environnement dynamique dans la récupération des cas ?


Dans les deux prochaines sections nous exposerons les réponses que nous avons apportées à ces questions. Nous indiquerons précisément comment le raisonnement à partir de cas et la méthode de résolution de problèmes pour l’assemblage des PACTs s’intègrent. Nous montrerons qu’une telle méthode ne peut être opérationnelle que parce que nous faisons systématiquement appel à une base de cas contenant des fragments mélodiques des lignes de basse stockés sous forme de PACTs jouables. En effet, il est impossible de trouver des règles qui puissent garantir la génération d’une PACT jouable, pour n’importe quel ensemble de PACTs composant l’espace d’assemblage. Nous exposerons aussi la façon dont nous indexons les cas et l’importance du processus d’activation et d’assemblage des PACTs en tant qu’une étape de raffinement et d’enrichissement de la requête qui sera faite à la base de cas.

4.3.4.1. *L’évolution du rôle de la Mémoire Musicale*

L’hypothèse de base que nous avons initialement adoptée à l’égard du processus d’assemblage de PACTs est que l’on peut toujours générer une PACT jouable à l’aide de règles de production. Malheureusement, ceci ne peut pas être toujours garanti car une caractéristique des processus d’activation et de sélection des PACTs que nous venons de présenter est qu’ils ne donnent aucune indication sur la quantité et le contenu des PACTs qui composeront initialement l’espace d’assemblage. Très fréquemment, il y a un nombre trop réduit de PACTs, voire il n’existe aucune PACT. Supposons qu’après toutes les combinaisons la dernière PACT de l’espace soit “jouer avec un style rythmique basé sur la noire et très syncopé, pendant cette mesure de 4 temps”;

admettons que l'on puisse calculer la structure rythmique, comment achever le calcul des autres paramètres des notes ? Il est fréquent que la dernière PACT de l'espace d'assemblage n'ait pas une description suffisamment complète pour devenir jouable (par ex. "jouer en direction ascendante et avec peu de notes" ou "jouer avec une forte dissonance et dans une tessiture basse", "jouer très fort", etc.). Ceci est dû au fait que l'on est incapable d'explicitier toutes les connaissances nécessaires pour avoir un ensemble "complet" de critères. Les musiciens donnent souvent des justifications "partielles" sur les notes qu'ils ont jouées (par ex. "j'ai utilisé le mode dorien", "j'ai voulu augmenter la tension", etc.) (Cf. section 2.4.2). Autrement dit, il n'y a pas de règles pour activer, à chaque segment d'improvisation, des PACTs concernant tous les traits musicaux d'un fragment de ligne de basse. Certes, la flexibilité du cadre d'utilisation des PACTs que nous présentons est justement dans le fait qu'il est possible de spécifier *de façon partielle* les notes à jouer. Mais, ceci implique que l'agent soit en mesure de produire des notes même si la spécification est "partielle", ce qui peut être très problématique.

A partir de cette constatation nous avons pensé utiliser la Mémoire Musicale pour pallier la difficulté relative au manque de critères. Dans la méthode d'assemblage de PACTs que nous avons proposée initialement (Ramalho & Ganascia, 1994b), il y avait un opérateur en plus de ceux que nous avons présentés dans la section précédente. Cet opérateur, appelé *ajout*, était appliqué lorsque toutes les combinaisons entre PACTs pendant l'assemblage ne produisaient pas une PACT contenant suffisamment de critères pour le calcul des notes. L'application de cet opérateur consistait à chercher dans la Mémoire Musicale un fragment mélodique qui respectait le mieux les critères représentés par cette dernière PACT de l'espace d'assemblage. Une fois récupéré, ce fragment subissait des adaptations et ensuite il était "affecté" à la PACT en question, en la rendant ainsi jouable. Avec ce procédé, notre programme pouvait toujours trouver, à condition d'avoir une Mémoire Musicale relativement riche, un fragment mélodique qui respecte le plus possible les critères existants, si peu nombreux soient-ils. Plus précisément, étant donné que les fragments mélodiques de la Mémoire Musicale sont représentés en tant que PACTs jouables, l'opération qui consiste à trouver un fragment qui satisfait le mieux un certain nombre de critères contenus dans une PACT en voie d'assemblage est équivalent à une opération de mesure de similarité entre cette PACT et la PACT jouable stockée dans la Mémoire Musicale. En effet, les PACTs jouables

contiennent implicitement toutes les “actions potentielles” (les critères) qui ont conduit à l’achèvement du calcul de ses notes. Par exemple, à partir des notes d’une PACT comme “jouer ”, dont l’accord sous-jacent est un F, on peut facilement déterminer tous les autres traits musicaux associés : le style rythmique (basé sur la noire), la densité (moyenne), le contour mélodique (ascendant), la dissonance (faible), l’inversion (première), la “syncopicit  ” (aucune), etc. Par exemple, supposons qu’   la fin de l’assemblage on se trouve avec une PACT de type “jouer avec un rythme bas   sur la noire”. Malgr   l’apparent manque de crit  re, il suffit de trouver, dans la M  moire Musicale, une PACT jouable (fragment m  lodique) qui ait une densit   tr  s faible et qui ait   t   jou  e dans un segment de grille similaire (dur  e, accords, sch  ma d’accord, style de grille, position dans la grille, tonalit   locale, etc.). La figure 4.3.5 montre un fragment de ligne de basse jou   par Ron Carter sur un II-V mineur qui pourrait   tre r  cup  r   de la M  moire Musicale de l’agent    ces fins.

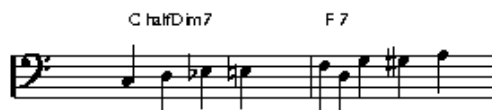


Figure 4.3.5 - Fragment de ligne de basse jou  s par Ron Carter sur un II-V mineur aux mesures 29-30 de *STELLA BY STARLIGHT* (Cf. Annexe A).

Le r  le que nous avons initialement donn      la M  moire Musicale est donc de fournir une PACT jouable lorsque l’agent n’a pas suffisamment de crit  res pour d  terminer les notes. Dans le cas contraire, les notes sont d  termin  es via des r  gles de production. L’id  e de ne faire appel    la M  moire Musicale que de temps en temps nous a paru int  ressante car on ne peut imaginer qu’un jazzman construise une improvisation ou un accompagnement en “collant” syst  matiquement des fragments m  lodiques m  moris  s l’un apr  s l’autre (Ramalho & Ganascia, 1994a). Comme nous l’avons d  j   expos      la fin de la section 2.4.1,    certains moments, le jazzman r  utilise des fragments m  lodiques qu’il a r  pertori  s mais ceci ne semble pas se faire de mani  re syst  matique.

Nous avons r  alis   quelques tests afin de valider l’hypoth  se selon laquelle nous pouvions toujours calculer les notes via des r  gles de production, pourvu que les crit  res (repr  sent  s par les PACTs en voie d’assemblage) soient suffisamment nombreux. Nous avons alors mis en   uvre un mod  le selon lequel nous compl  tons

l'information manquante dans la dernière PACT de l'espace d'assemblage par des critères par défaut (par ex. densité moyenne, dissonance moyenne, première inversion, arpège, "syncopité" moyenne, etc.). Après quelques mois de travail, les résultats musicaux obtenus étaient toujours très décevants : dans les lignes de basse il n'y avait que des noires et que des arpèges, et toujours avec la tonique en premier temps (Cf. figure 4.3.6). Le problème principal était que nous avons beaucoup de mal à prendre en compte tous les critères associés à un grand répertoire de traits musicaux (contour mélodique, gamme, tessiture, structure rythmique, accentuation, tension, dynamique, "syncopité", densité, etc.). Cela tenait au fait que nous n'arrivions pas à expliciter les connaissances impliquées dans la plupart des calculs notamment celui du rythme et celui des notes elles-mêmes. Par exemple, les connaissances sur le rythme et sur l'accentuation des notes ne sont formalisées nulle part sous la forme de règles, ou de contraintes. Il fallait soit ajouter des règles totalement arbitraires, soit figer les configurations de critères possibles, soit encore diminuer leur nombre. Autrement dit, il fallait réduire le répertoire de PACTs, ce que nous jugions incohérent avec notre objectif d'essayer d'incorporer le maximum de connaissances utilisées par les jazzmen.

La conclusion de ces premières expérimentations est que le "manque de critères" n'est pas le seul problème à affronter pour la détermination de notes via de règles de production. Même s'il y a beaucoup de critères, on n'est toujours pas capable de calculer les notes car on ne sait pas respecter tous les critères à la fois. Les difficultés que nous avons trouvées sont ni plus ni moins celles de la manipulation de divers traits musicaux en vue d'en calculer des notes (Cf. section 3.4.4). En fait, il est extrêmement difficile d'expliciter en termes de règles les connaissances nécessaires à la prise en compte de toutes les interactions entre les traits. Afin de résoudre ce problème, nous avons décidé d'utiliser davantage la Mémoire Musicale. En effet, comme nous l'avons présenté auparavant (Cf. chapitre 2), les fragments mélodiques ou rythmiques peuvent être vus comme l'achèvement d'un ensemble de critères car, de façon implicite, ils "codent" en eux-mêmes l'interaction entre différents critères. La recherche dans la Mémoire Musicale suit le même principe indépendamment de nombre de critères. On cherche toujours le fragment qui globalement les satisfait le mieux. Supposons qu'à la fin de l'assemblage on se trouve avec la PACT "jouer beaucoup de notes, avec des arpèges, légèrement dissonant, dans une direction horizontale, dans une tessiture moyenne, très fort, pas en première inversion et assez syncopé". La figure 4.3.7 montre

un fragment mélodique qui peut être récupéré selon les critères de cette PACT. Notons que, pour le premier accord, la tonique du fragment récupéré est sur le premier temps, ce qui est en contradiction avec le critère “pas en première inversion”. Le fragment peut être adapté en changeant par exemple la première note par un Sol3.

The image displays a musical score for the piece 'STELLA BY STARLIGHT'. It consists of seven staves of music, each representing a four-measure phrase. Above each staff, the corresponding chord progression is written. The chords are as follows:

- Staff 1: E halfDm7, A 7, C min 7, F 7
- Staff 2: F min 7, Bb 7, Eb maj7, Ab 7
- Staff 3: Bb maj7, E halfDm7A 7, D min 7, G min 7 C 7
- Staff 4: F maj7, G min 7 C 7, A halfDm7, D 7
- Staff 5: G 7, G 7, C min 7, C min 7
- Staff 6: Eb min 7, Eb min 7 Ab 7, Bb maj7, Bb maj7
- Staff 7: E halfDm7, A 7, D halfDm7, G 7
- Staff 8: C halfDm7, F 7, Bb maj7, Bb maj7

Figure 4.3.6 - STELLA BY STARLIGHT : Utilisation des seules règles de production pour calculer les notes

L'intérêt majeur d'une utilisation étendue de la Mémoire Musicale est de garantir que, pour n'importe quel ensemble de critères associés aux différents traits musicaux (représentées par une PACT en voie d'assemblage), nous pouvons obtenir une PACT jouable. Ainsi, d'un rôle plutôt “marginal”, où elle n'intervenait que lorsque “l'information manque”, la Mémoire Musicale passe à un rôle central : elle fournit un fragment mélodique à chaque segment d'improvisation. En d'autres termes, notre agent produit les lignes de basse en “collant”, les uns après les autres, des fragments mélodiques préalablement stockés, ces fragments devant être adaptés (par transposition, remplacement de notes, etc.) à un segment d'improvisation courant. C'est d'ailleurs le même mode de fonctionnement que celui de Band-in-a-box et Hodgson (Band-in-a-box,

1995; Hodgson, 1996a). Naturellement, il reste toujours la possibilité qu'une PACT déjà jouable soit activée, en rendant inutile l'appel à la Mémoire Musicale. En effet, des PACTs comme "jouer le fragment précédent en le transposant d'un ton" ou "jouer ces notes" peuvent être activées. Ce dernier type de PACT est activé selon certaines "marques" que nous attachons à la grille. Il arrive que les jazzmen se mettent d'accord sur un certain nombre de conventions rythmiques ou mélodiques. Quand ils sont à un endroit précis de la grille ou quand ils entendent un signal, alors ils commencent à jouer comme prévu. A l'exception de ces situations plutôt rares, la détermination des notes se fait toujours à l'aide de la Mémoire Musicale.



Figure 4.3.7 - Fragment de ligne de basse joués par Ron Carter sur un II-V mineur aux mesures 25-26 de *STELLA BY STARLIGHT* (Cf. Annexe A).

Revenons maintenant sur le fonctionnement de l'opérateur "propage". Toute PACT de l'espace d'assemblage peut avoir sa description enrichie pour devenir plus jouable, si elle contient les informations nécessaires. Les propagations simples, pour lesquelles on a les connaissances qu'il faut, sont réalisées à l'aide de règles. Ainsi, la PACT "jouer avec des arpèges et de façon dissonante" peut, après le déclenchement d'une propagation, devenir "jouer avec des arpèges, de façon dissonante et en évitant la tonique au premier temps", car, à partir des valeurs du style de la ligne (arpège) et de la dissonance, on peut stipuler l'inversion de l'accord. De même, "jouer avec un style rythmique basé sur la croche et avec faible dissonance" devient "jouer avec un style rythmique basé sur la croche, avec faible dissonance et en utilisant des notes répétées". Néanmoins, le calcul des notes se fait *toujours* via la Mémoire Musicale. A ce sujet, il faut bien préciser les conditions de déclenchement. Comme nous l'avons montré, on peut toujours rendre jouable n'importe quelle PACT en déclenchant une propagation qui va chercher les notes dans la Mémoire Musicale. Il faut donc contrôler ce déclenchement, faute de quoi les PACTs composant l'état initial de l'espace d'assemblage deviendront chacune jouable précipitamment, avant même que les combinaisons et les résolutions de conflits entre elles ne commencent. Il est évident que nous avons intérêt à considérer toute l'information contenue dans la PACTs, sinon nous n'aurions pas besoin de les avoir toutes activées. Pour cette raison, nous avons introduit

un *seuil de propagation*, qui indique l'information minimale nécessaire pour le déclenchement de la propagation du calcul de notes (par ex. le calcul des notes n'est autorisé que lorsque l'on connaît les valeurs du style de la ligne, de la densité de notes, de la dissonance, de la tessiture, etc.). Normalement, nous fixons ce seuil très haut de manière à repousser le plus loin possible le déclenchement de la propagation du calcul de notes, et, ainsi, garantir qu'une bonne partie des combinaisons entre les PACTs s'effectue.

La conséquence la plus importante de ce contrôle strict de la propagation vers les notes, est que l'assemblage proprement dit échoue la plupart du temps. On obtient rarement une PACT déjà suffisamment jouable pour que l'on autorise la récupération d'une PACT jouable dans la Mémoire Musicale. Dans ces situations, on prend la PACT qui ressort de l'assemblage, on "baisse le seuil de propagation" et on déclenche le calcul des notes, c'est-à-dire la récupération d'un fragment mélodique dans la Mémoire Musicale. C'est une étape de "post-assemblage" qui est toutefois en accord avec les principes de l'assemblage. Cette étape peut être nécessaire indépendamment de la valeur du "seuil", car il se peut qu'aucune PACT n'ait été sélectionnée concernant le segment courant. Dans ce cas rarissime, les notes sont obtenues de la même façon, sauf que la recherche dans la Mémoire Musicale est basée uniquement sur les caractéristiques du segment de la grille courant et du fragment qui a été joué auparavant.

Pour illustrer le mécanisme de propagation que nous venons d'exposer, supposons que l'espace initial d'assemblage soit composé des cinq PACTs suivantes : p_1 = "jouer dans la direction ascendante"; p_2 = "jouer assez dissonant"; p_3 = "jouer des arpèges"; p_4 = "ne pas jouer syncopé" et p_5 = "jouer avec le rythme basé sur la noire". Supposons que l'on combine d'abord p_1 et p_2 (l'ordre dans cet exemple n'a pas d'importance) et ensuite p_{1+2} et p_3 . Grâce aux informations provenues de p_2 et p_3 , le calcul de l'inversion de l'accord se déclenche et est réalisé par une règle de production simple. La PACT p_{1+2+3} deviendra "jouer dans la direction ascendante, assez dissonant, avec des arpèges et en évitant la tonique au premier temps". Ensuite, p_4 et p_5 se combinent et enfin p_{4+5} rejoint p_{1+2+3} . On obtient alors $p_{1+2+3+4+5}$ = "jouer dans la direction ascendante, assez dissonant, avec des arpèges, en évitant la tonique au premier temps, pas syncopé et avec le rythme basé sur la noire". Imaginons que le segment de la

grille soit un II-V majeur durant huit temps. Alors, s'il est permis, on peut par exemple récupérer le fragment mélodique affiché dans la figure 4.3.8.



Figure 4.3.8 - Fragment de ligne de basse joué par Ron Carter sur *STELLA BY STARLIGHT* aux mesures 67-68 (Cf. Annexe A).

4.3.4.2. *Indexation des cas*

Un des problèmes les plus difficiles et centraux dans la recherche en raisonnement à partir de cas est celui de *l'indexation* des cas. Il s'agit d'affecter à un cas un ensemble d'index qui, a priori, permet que le cas soit pertinemment récupéré lorsqu'une situation ou problème similaire se présente. Comme nous montrons par la suite, ce problème est d'autant plus délicat dans modélisation des activités qui se déroulent dans le temps, comme celle d'un jazzman, car le nombre d'index possibles est très important. Dans cette section nous approfondissons cette problématique et nous présentons la solution originale que nous avons y apportée.

Pour mieux comprendre cette problématique, regardons de plus près le fonctionnement du raisonnement à partir de cas. Un système à base de cas résout des nouveaux problèmes en adaptant les solutions déjà employées pour solutionner d'autres problèmes. C'est pourquoi dans un cas on représente le problème, la solution et, éventuellement, la qualité (*outcome*) de la solution. Dans la Mémoire Musicale, la composante "solution" de chaque cas correspond à un pattern mélodique tandis que le problème est désigné par plusieurs éléments qui nous présentons par la suite. Nous ne représentons pas l'évaluation, qui est naturellement problématique en musique (et plus généralement dans les arts), d'autant plus que l'évaluation esthétique concerne davantage le morceau (la chanson) comme un tout qu'un petit fragment mélodique.

L'approche des systèmes à base de cas peut être résumée par "le cycle des quatre RE's" (Aamodt & Plaza, 1994) : *retrieve*, *reuse*, *revise* et *retain* (ça marche aussi en français — Cf. figure 4.3.9). Une description initiale du problème (ou situation) donné constitue le "nouveau cas". En premier lieu, ce nouveau cas (appelé techniquement *cas cible*) est utilisé pour récupérer un des cas préalablement stockés

dans la base (appelés techniquement *cas sources*). Le cas source le plus similaire au cas cible est choisi. En deuxième lieu, le cas source récupéré est “combiné” avec le cas cible pour générer un cas solution. Autrement dit, la solution contenue dans le cas source récupéré est adaptée au problème représenté dans le cas cible. En troisième lieu, le succès du cas solution est vérifié et si elle ne convient pas, le cas est révisé. En dernier lieu, le nouveau cas produit est retenu pour être réutilisé pour des problèmes futurs. Une bonne partie des systèmes à base de cas ne font pas ce cycle entier, car l’étape de révision est extrêmement difficile et requiert beaucoup de connaissances du domaine (Aamodt & Plaza, 1994).

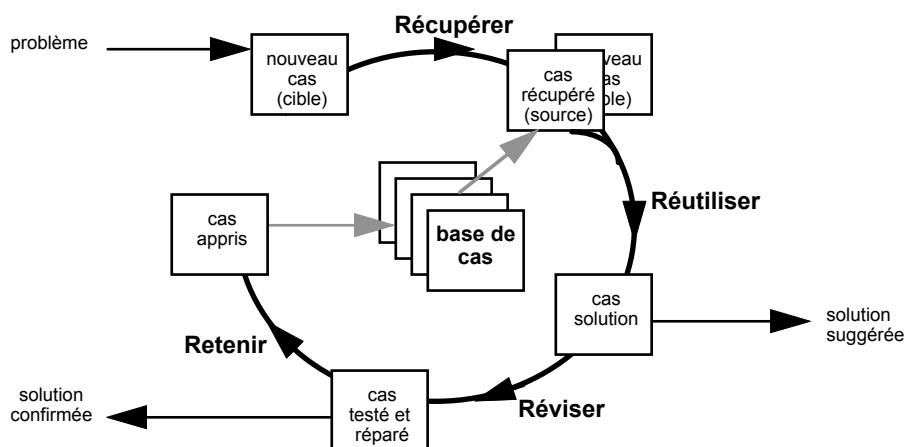


Figure 4.3.9 - Le cycle du raisonnement à partir de cas

Ce “cycle de quatre RE’s” permet que l’on pose la problématique de *l’interprétation de la situation (situation interpretation or situation assessment)* (Kolodner, 1993) [chap. 10]. L’interprétation de la situation (IS) vise à décrire un problème (ou situation) “en état brut” dans le format des cas existant dans la base. Il s’agit précisément d’identifier les caractéristiques “les plus pertinentes” d’un problème (ou situation) afin de déterminer l’ensemble d’index selon lequel le cas sera décrit. L’IS s’applique donc en amont du cycle, où on doit traduire le problème donné dans un cas cible de façon à favoriser, à travers une “requête pertinente”, la récupération du cas source le plus approprié. L’IS se fait aussi lors de l’ajout d’un nouveau cas à la base, et ce, soit en aval du cycle, soit pendant la construction de la base de cas quand celle-ci est créée préalablement. Il est nécessaire de savoir quels sont les index les plus pertinents pour que le nouveau cas ajouté soit bien discriminé des autres déjà stockés. L’IS est très difficile à réaliser car les caractéristiques d’un problème peuvent être nombreuses et

L'on ne sait pas forcément comment choisir les index les plus pertinents. Malgré son importance dans le raisonnement à partir de cas l'interprétation de la situation a été très peu étudiée jusqu'à aujourd'hui, à l'exception notamment des systèmes CASEY (Koton, 1988), JUDGE (Bain, 1989) et ANON (Owens, 1989).

Il ne faut pas confondre l'IS avec la problématique de la constitution du *vocabulaire d'indexation*, à savoir l'ensemble d'index qui caractérisent les cas de la base (Kolodner, 1993) [chap. 6]. Par exemple, si l'on veut représenter des cas correspondant à des disputes entre deux personnes (le système JUDGE), un vocabulaire d'indexation pourrait être le suivant : objet disputé et, pour chaque personne, les buts, les noms, les ages, les sexes, les tailles, les poids, les professions, les liens familiaux. L'IS consiste à déterminer pour un cas particulier quels sont les index à considérer. Si, par exemple, il existe des liens familiaux entre les personnes, la profession importe sans doute peu. Par contre, l'âge doit être pris en compte pour mieux caractériser s'il s'agit d'une simple dispute entre enfants ou une dispute plus sérieuse entre adultes.

En revenant au jazz, quels sont les index intéressants pour stocker les patterns mélodiques dans la Mémoire Musicale ? La réponse la plus naturelle serait de choisir les index caractérisant l'environnement de l'agent jazzman. Nous savons aussi que le passé joue un rôle d'indexation, car une des principales caractéristiques de la musique est justement de créer continuellement des attentes et de les résoudre ou de les frustrer par la suite (Narmour, 1977; Narmour, 1989). L'ennui est qu'il y a trop de d'éléments (Cf. section 1.2.5) dans l'environnement et il très difficile de "filtrer" ce qui est important. Les accords sur lesquels le pattern a été joué sont des index très pertinents mais ceux qui viennent après et avant ne seraient-ils pas intéressants ? La grille entière avec toute sa structure (sections, chorus, mesures, etc.) ne serait-elle pas également à considérer ? Toutes les notes jouées par l'agent depuis le début ont-elles la même importance ? Faudrait-il utiliser comme index tout ce que les autres jazzmen ont déjà joué ?

Toutes ces questions s'insèrent dans une problématique plus générale concernant l'IS dans le domaine appelé *raisonnement à partir de cas continu*, où on modélise des phénomènes qui se déroulent dans le temps, comme l'activité d'un robot autonome, la propagation d'un incendie de forêt, l'évolution de santé d'un patient, etc. (Ram & Santamaria, 1993). Le choix des index est plus critique dans ces domaines parce que l'on ne peut songer inclure tous les événements ou paramètres constituant l'historique du

phénomène. Si tout est pris en compte, la mesure de similarité entre les cas devient extrêmement coûteuse et, la recherche de cas, inefficace. Mais ce qui est pire, on n'est plus sûr de récupérer le bon cas car les données non pertinentes biaisent incorrectement la mesure de similarité entre le cas cible et les cas sources (Kolodner, 1993) [chap. 6]. Pour s'en sortir, les chercheurs fixent une taille (fenêtre) maximale du passé à considérer, essaient d'établir des classes d'événements du phénomène pour faciliter leur mesure de similarité et utilisent des algorithmes assez sophistiqués d'appariement entre ces événements (Ram & Santamaria, 1993; Rougegrez-Loriette, 1994).

Dans le cas de la musique, l'utilisation d'algorithmes d'appariement des notes et des accords (Dannenberg, 1984; Rolland & Ganascia, 1996a; Stammen & Pennycook, 1993) rendrait le calcul de la similarité excessivement coûteux. En outre, d'après notre travail d'acquisition de connaissances, nous pensons que les traits musicaux et les structures musicales sont des index plus pertinents que les notes elles-mêmes. Ces "descriptions supplémentaires" (traits et structures), étant plus simples que les notes elles-mêmes, ont l'avantage de rendre la mesure de similarité moins complexe et de minimiser les risques que l'on biaise incorrectement la mesure de similarité. Malheureusement, il ne suffit pas de faire appel aux traits et structures, puisque le problème de la taille du passé ou du futur à considérer demeure. Par conséquent, nous avons dû aussi réduire le contexte que nous voulons considérer pour indexer les cas. Dans la Mémoire Musicale, nous indexons un pattern par trois groupes d'éléments : les traits musicaux du pattern; la description du segment de la grille sur lequel le pattern a été joué (schéma d'accord, position dans la grille, tonalité locale, accords, etc.); et les traits musicaux du pattern joué juste avant (ainsi que le segment de grille de ce dernier) (Cf. figure 5.5.6). Nous ne tenons pas compte des événements du Scénario, car l'acquisition des cas avec ce type d'indexation serait trop difficile (Cf. section 5.5.2).

Toutefois, nous compensons toutes ces réductions par l'élaboration d'une requête (cas cible) plus riche à la base de cas, à l'aide de l'activation et de l'assemblage des PACTs. Au lieu de simplement adresser la requête "cherchez un fragment mélodique joué sur tels accords, et dont le fragment précédent a tels traits musicaux et tels accords sous-jacents", nous (ré)interprétons la description du problème et nous adressons la requête "cherchez un fragment mélodique qui a une tessiture basse, qui n'est pas dissonant, qui est moyennement syncopé, qui a un rythme basé sur la blanche,

qui a des arpèges, qui a été joué sur tels accords, et dont le fragment précédent a tels traits musicaux et tels accords sous-jacents”. Au fond, c’est à cela que sert toute l’étape d’activation et d’assemblage des PACTs. Derrière cette requête, il y a implicitement l’influence de la grille dans sa totalité, de la toute la ligne de basse jouée jusqu’alors et de tous les événements du scénario, à travers les PACTs qui ont été activées jusqu’à présent (Cf. figure 4.3.10). En outre, ceci nous donne plus de contrôle sur la récupération du cas, car le grand défaut d’une mesure globale de similarité est qu’il est difficile de maîtriser l’influence de chaque descripteur (index) des cas (les traits musicaux du fragment mélodique, le type de schéma d’accord, la tonalité locale, la position dans la grille, le tempo, les traits du fragment précédent, etc.) (Kolodner, 1993) [chap. 8]. L’étape d’activation et d’assemblage des PACTs nous permet donc de “biaiser” le choix du fragment. Ceci est d’autant plus utile que c’est à travers l’activation des PACTs que le programme peut tenir compte des événements du Scénario.

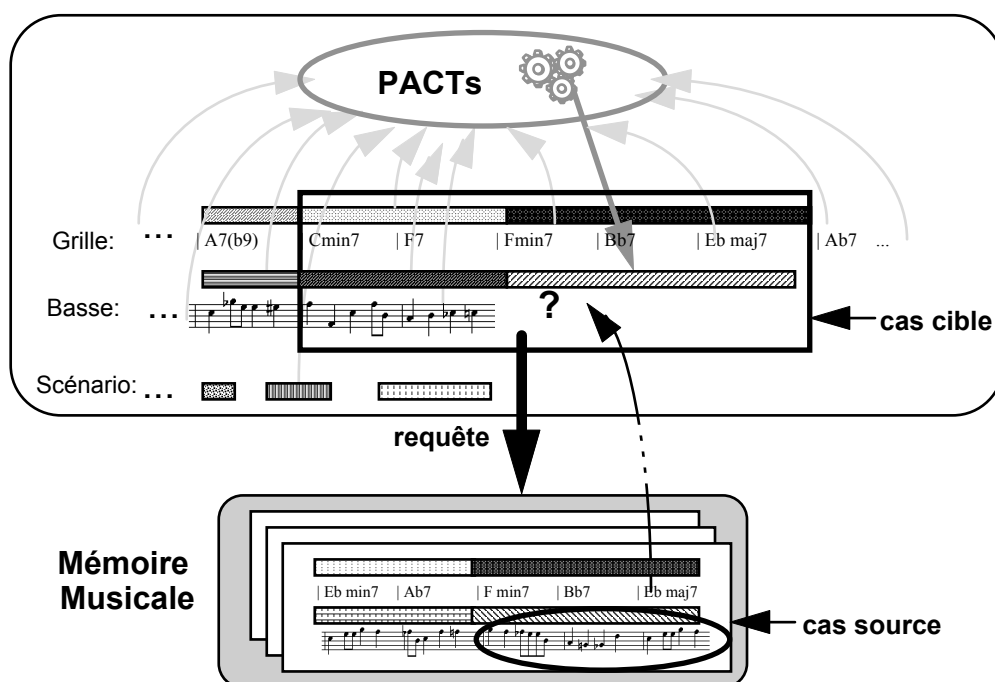


Figure 4.3.10- Activation et assemblage des PACTs en tant que raffinement de la requête de la base de cas

Plus on est capable d’expliciter des connaissances (même si elles sont incomplètes et un peu contradictoires) pour activer des PACTs et pour propager

l'information à l'intérieur d'une PACT, plus la requête à la Mémoire Musicale est précise. Toutefois, si on n'a aucune connaissance de ce genre, cela n'empêche pas le programme de bien trouver un cas. Nous arrivons ici au centre de notre contribution et à la force de notre modèle : la symbiose entre le raisonnement déductif et le raisonnement à partir de cas. Le modèle offre la possibilité au concepteur de représenter sous forme de règles toutes les connaissances qui peuvent être ainsi explicitées, tout en garantissant la résolution du problème quoi qu'il arrive. Naturellement, à condition d'avoir un nombre minimal de cas dans la base (Cf. section 6.3.5).

4.3.5. L'humeur de l'agent

Après quelques expérimentations nous avons ressenti le besoin d'explorer davantage la notion d'état interne de notre agent. Outre la mémoire de travail, tout à fait habituelle dans l'architecture d'un agent rationnel (Laird, Newell & Rosembloom, 1987; Newell, 1990), nous avons associé à l'état interne ce que nous appelons *l'Humeur*. Cette Humeur est censée représenter, de façon très simplifiée, la disposition d'esprit de l'agent à travers trois variables : urgence (le manque de temps), concentration et confiance en soi.

$$\text{Humeur} = (\text{urgence}, \text{concentration}, \text{confiance}) \quad (4.3.1)$$

L'intérêt de la notion d'Humeur dans notre modèle est multiple. D'abord, à travers l'Humeur on est en mesure de mieux contrôler l'interaction avec l'environnement, notamment en ce qui concerne le temps et quelques événements du scénario. Ensuite, l'Humeur aide à minimiser le problème de la segmentation du raisonnement puisqu'il s'agit toujours d'une mesure plus globale et à long terme. Enfin, les changements de valeurs des variables de l'Humeur, nous offrent la possibilité de "bouleverser" le raisonnement, la perception et l'action de l'agent de manière à pouvoir rendre son comportement plus ambigu. En effet, l'Humeur peut inhiber ou provoquer l'activation de certaines PACTs, ainsi que, dans le cas de "urgence", changer le mode de raisonnement. Voyons plus précisément dans la suite chacun de ces aspects.

La variable de l'Humeur qui modifie plus radicalement le comportement de notre agent bassiste est sans doute "l'urgence". Elle peut atteindre trois valeurs (trèsEnAvance, normal et trèsEnRetard) selon la valeur courante de ERE* (Cf. section 1.2.3), les seuils dépendant bien sûr de la machine sur lequel le programme tourne. Si l'agent est "très en retard", l'étape d'activation de PACTs est réduite à l'activation

d'une seule PACT "jouer pattern cliché", conforme à la suggestion de F. Pachet (Pachet, 1987; Pachet, 1990). Cette PACT renforce le choix dans la Mémoire Musicale d'un pattern dont les propriétés sont les plus courantes. L'assemblage est aussi court-circuité car, quand il est très pressé, l'agent ne considère pas les anciennes PACTs qui sont pertinentes vis-à-vis du segment courant. L'assemblage échouant forcément, on récupère dans la Mémoire Musicale la PACT jouable adéquate.

Les deux variables restantes fonctionnent comme des accumulateurs. Elles subissent des incréments ou décréments selon les événements du scénario. La "concentration" peut être perturbée par un événement comme "la police arrive" et ou renforcée par quelque chose de type "le soliste fait un signe pour attirer l'attention de la section rythmique". Quand la "concentration" est très basse l'agent ignore les événements du Scénario (sauf, naturellement, ceux qui attirent son attention). De même, lorsque l'agent n'est pas très concentré, on active des PACTs concernant des choses simples à jouer comme "jouer peu de notes", "jouer des arpèges", "jouer pas syncopé", etc. Quant à la "confiance", elle est également sensible à l'environnement. Par exemple, si "le public applaudit", spécialement à la fin des chorus, l'agent renforce sa "confiance" et si "le public dit qu'il n'en peut plus" ou simplement n'applaudit pas aux bons moments, la "confiance" faiblit. Pendant que la "confiance" est grande l'agent continue à jouer pareil (on active des PACTs qui ont les mêmes caractéristiques de celles jouées récemment). Si la "confiance" faiblit alors, l'agent change ce qu'il est en train de faire. Le tableau 4.3.2 résume, de façon plus globale, les liens entre les variables de l'Humeur et quelques dilemmes, parmi plusieurs, de l'improvisation et de l'accompagnement en jazz.

	urgence (↑)	concentration (↑)	confiance(↑)
homogénéité / changement			+ / -
innovation / cliché	- / +		
collectif / personnel	- / +	+ / -	
complexité / simplicité		+ / -	

Tableau 4.3.2- Rapports entre l'Humeur et quelques dilemmes du jazz

Nous ne pouvons justifier la plausibilité musicale de tout ce qui est autour de la notion d'Humeur. Les liens entre les événements et les variables de l'Humeur et entre celles-ci et l'activation des PACTs et le changement du mode de raisonnement de l'agent sont arbitraires. Nous avons introduit la notion de l'Humeur à cause de besoins techniques, bien que notre agent puisse accomplir sa tâche sans ce raffinement. En

réalité, tout le monde sait que dans n'importe quelle tâche de la conception d'un programme d'IA, il y a toujours le biais du concepteur. Soit pendant l'acquisition de connaissances, soit dans la désignation du formalisme de représentation de connaissances et des mécanismes d'inférences, il ne peut que minimiser les choix qui ne reflètent pas les connaissances de l'expert que l'on veut modéliser. Mais cela n'est pas bien grave car, *l'artificialité*, dont il est question en IA, nous donne la liberté d'utiliser toutes les techniques et modèles disponibles à condition que cela améliore les performances du programme. Dans les mots de Simon : "*artificiality connotes perceptual similarity but essential difference, resemblance from without rather than within*" (Simon, 1981) [p. 17].



5. VUE DETAILLEE DES PACTs

Dans ce chapitre nous exposerons une formalisation de la notion de PACTs. Nous commencerons par montrer le vocabulaire de PACTs que nous avons acquis pour la basse. Ensuite, nous présentons le langage que nous avons conçu pour représenter ces PACTs. Enfin, nous présenterons formellement les opérations impliquant les PACTs, à savoir l'activation d'une PACT, la détermination du degré de jouabilité d'une PACT par rapport à une autre, les mesures de compatibilité et de similarité entre deux PACTs, la combinaison entre deux PACT, etc.

5.1. CONSIDERATIONS PRELIMINAIRES SUR LA REPRESENTATION DES PACTs

Dans la première partie de cette section, nous introduirons un vocabulaire de "PACTs de basse" que nous avons acquis et que nous aimerions représenter. Dans la deuxième partie, nous montrerons comment à partir d'une étude des trois langages de représentations (les graphes conceptuels, les langages de frames et les langages de classes) nous sommes arrivés à un langage hybride qui nous permet de représenter les PACTs et de définir formellement leurs propriétés.

5.1.1. Le répertoire PACTs de basse

Le vocabulaire des PACTs de basse, est constitué de toutes les descriptions possibles d'un fragment de ligne de basse. Ces descriptions vont du niveau des notes jusqu'à des traits musicaux tout en passant par d'autres descriptions comme des transformations sur les notes déjà jouées. Pour identifier ces descriptions, nous nous sommes servis de plusieurs sources : les PACTs déjà identifiées par Pachet, des discussions avec des

bassistes, la lecture de manuels de jazz et les programmes existants. L'élaboration d'un tel vocabulaire est un travail d'acquisition de connaissances avec toutes les difficultés connues, puisqu'il dépend du style de musique et de l'instrument. Bien entendu, il n'est pas envisageable de constituer un vocabulaire complet. Chaque musicien se forge ses propres "points de vue" sur ce qu'il écoute et joue. En outre, certains traits sont très flous. Par exemple, on ne sait pas exactement ce que Coolman (Coolman, 1985) a en tête quand il considère comme "lyrique" une des trente-sept lignes de basse (Jimmy Blanton, Ray Brown, Paul Chambers, Ron Carter, etc.) qu'il analyse.

Nous avons identifié trois groupes de descriptions. Nous avons identifié d'abord celles correspondant aux traits musicaux d'un fragment de ligne de basse, à savoir : *pitch contour*, *tessitura*, *scale*, *dissonance*, *bass line construction style*, *inversion*, *repeated notes*, *rhythm style*, *density*, *syncopation*, *amplitude contour*, *loudness*, *leading note*, *pull down*, *drop*, *swing*, *contrast*, *balance*, *accentuation*, *musical style*, *classiness*, *slide*, *ghost notes*, *bend*, *hammer on*, *pull off*, *dead notes*, etc.¹⁴. A cela s'ajoute un deuxième groupe de descriptions "plus proches des notes", à savoir *melodic fragment* (séquence de notes), *rhythmic fragment* (séquence de notes sans hauteur), *pitch list* (séquence de notes sans durée), *enveloppe* (les amplitudes des notes), etc. Enfin, le dernier groupe est composé d'un type spécial de description où un fragment de ligne de basse est conçu par rapport à un autre. C'est le cas des répétitions, des transpositions, ou d'autres variations. On peut dire, par exemple, qu'un fragment mélodique est le même qu'un tel autre à une transposition près, ou qu'un fragment mélodique a le même rythme qu'un tel autre. Nous appelons *descriptions standards* celles des deux premiers groupes où l'on parle, d'une façon "objective", des propriétés souhaitées ou déjà réalisées d'un fragment de ligne de basse. Nous appelons *descriptions transformationnelles* celles du dernier groupe où on se réfère à un fragment de ligne de basse de façon "relative" car il est vu en tant que transformation d'un autre.

Pour diverses raisons nous avons dû, en pratique, restreindre davantage ce vocabulaire. L'utilisation de PACTs liées aux descriptions transformationnelles est très délicate, car elle pose le problème de l'évaluation dynamique de ce que l'agent vient de

¹⁴Nous préférons employer l'Anglais pour rendre plus homogène la présentation de ces descriptions, certaines d'entre elles n'ayant pas de traduction "parlante".

jouer (Cf. sections 3.2.1 et 3.4.3). L'agent doit avoir une raison pour activer une PACT de type "jouer avec le même rythme de la dernière phrase (fragment)". La raison pourrait être simplement que l'agent ait jugé le rythme du dernier fragment mélodique intéressant. Quoique nous ayons implanté ce type de PACT dans le système, nous n'en avons guère fait usage dans la première version du modèle car il est bien difficile de doter l'agent d'une telle capacité de jugement.

En ce qui concerne les descriptions standards, nous avons dû aussi les limiter. Certaines descriptions, comme "contraste" et "équilibre", s'appliquent à une ligne complète de basse plutôt qu'à un fragment. Nous avons aussi écarté, dans un premier temps, ce qui concerne le son (*slide, ghost notes, dead notes, bend, hammer on, pull off*), et ceci pour deux raisons. Premièrement, pour prendre en compte ces nuances de l'exécution de l'instrument, on doit avoir un programme capable d'exercer un contrôle très fin sur le changement de timbre du synthétiseur (Cf. figure 1.2.2) et il ne serait pas raisonnable de vouloir construire un tel programme en plus du travail de thèse. Deuxièmement, l'acquisition des cas (exemples de fragments de lignes de basses qui composent la Mémoire Musicale) serait infaisable, puisqu'il serait nécessaire de disposer d'un programme capable d'extraire ce type d'information à partir des enregistrements sonores de groupes de jazz.

5.1.2. Quel formalisme ?

Il existe trois grandes familles de formalismes de représentation de connaissances en IA : la logique, les procédures, les langages à héritage ou à association (dont les réseaux sémantiques, les frames, les graphes conceptuels, les langages de classes, etc.)¹⁵. Nous ne voulons pas entrer dans le débat sur quel est "le meilleur" langage car il n'y a pas de langage universel qui possède toutes les qualités nécessaires à la construction d'un système à base de connaissances. Ce n'est que vis-à-vis d'un problème précis que l'on sait quel est le langage le plus adéquat en ce qui concerne l'expressivité, l'efficacité, la modularité, la réutilisabilité, la lisibilité, etc. (Masini et al., 1989; Niwa, Sasaki & Ihara,

¹⁵Nous ne tenons pas à décrire ces formalismes qui sont bien étudiés dans les ouvrages généraux comme (Charniak & McDermott, 1985; Rich & Knight, 1983; Winston, 1992) et surtout (Brachman & Levesque, 1985).

1984; Rich & Knight, 1983; Winograd, 1975). Par ailleurs, si le problème en question est complexe, on va plutôt utiliser un langage hybride intégrant plusieurs formalismes et styles de programmation. En effet, les langages cités plus haut sont rarement employés à “l’état pur”, les chercheurs en IA préférant en général développer leur propre langage en fonction de leurs besoins.

Nous n’étions pas sûr d’avoir besoin de l’expressivité de la logique de premier ordre et nous ne voulions pas avoir des problèmes d’efficacité posés par un tel langage, car notre agent est censé évoluer sous des contraintes de temps importantes. D’autre part, nous ne voulions pas procéder à des manipulations syntaxiques d’expressions mais manipuler une entité bien définie : la PACT. En effet, nous savions ce que les PACTs étaient censées représenter ainsi que les types d’inférences les impliquant. C’est pourquoi, nous nous sommes centrés sur les langages à héritage, les procédures (la deuxième famille de langages de représentation de connaissances) pouvant être incorporées dans des tels langages, notamment les frames (Winograd, 1975). Pour notre formalisation des PACTs, nous avons étudié plusieurs langages: les graphes conceptuels (Sowa, 1984), les réseaux sémantiques (Quillian, 1968) et ce que Masini (Masini et al., 1989) appelle “les langages à objets”, à savoir, les langages de frames (Minsky, 1975); les langages de classes (comme Simula (Birtwhistle et al., 1973) et Smalltalk-80 (Goldberg & Robson, 1983)); et les langages d’acteurs (Hewitt, Bishop & Steiger, 1973).

Les graphes conceptuels sont un formalisme de représentation de connaissances qui nous était très familier, car ils ont fait l’objet de plusieurs travaux dans l’équipe ACASA aussi bien en acquisition de connaissances (Aïmeur, 1994; Faron & Kieu, 1993), qu’en apprentissage (Bournaud, 1996) . Les graphes conceptuels nous ont été très utiles dans la mesure où ils nous ont permis de mieux organiser les descriptions standards. Dans ce type de langage, on est obligé de rendre explicite les relations entre les attributs, en l’occurrence “composé de” et “caractérisé par” comme le montre la figure 5.1.1¹⁶. Un fragment mélodique peut être projeté sur trois axes : de temps (*rhythmicFragment*), de hauteurs (*pitchList*) et d’amplitudes (*envelope*)¹⁷.

¹⁶En réalité les relations s’établissent entre des *concepts*, un concept étant le triplet attribut, sélecteur, valeur.

¹⁷Les divers effets sonores pourraient être regroupés autour d’une projection sur l’axe du timbre.

L'arbre résultant montre de façon plus intelligible ce que, vis-à-vis des accords sous-jacents, on sait dire sur un fragment rythmique, une séquence de hauteurs, une enveloppe d'amplitudes et ce que l'on ne peut dire que si les trois dimensions sont réunies. Comme on le verra plus en détail par la suite, une telle organisation, que nous appelons désormais *hiérarchie des descriptions standards* (HDS), est d'une importance capitale lorsque l'on doit décider si une PACT est plus jouable qu'une autre et si une PACT peut être combinée avec une autre. Par exemple, une PACT dont la valeur de `rhythmicFragment` est connue, est plus jouable qu'une autre où seule la valeur de `density` est connue car le premier attribut est "hiérarchiquement plus jouable" que le deuxième.

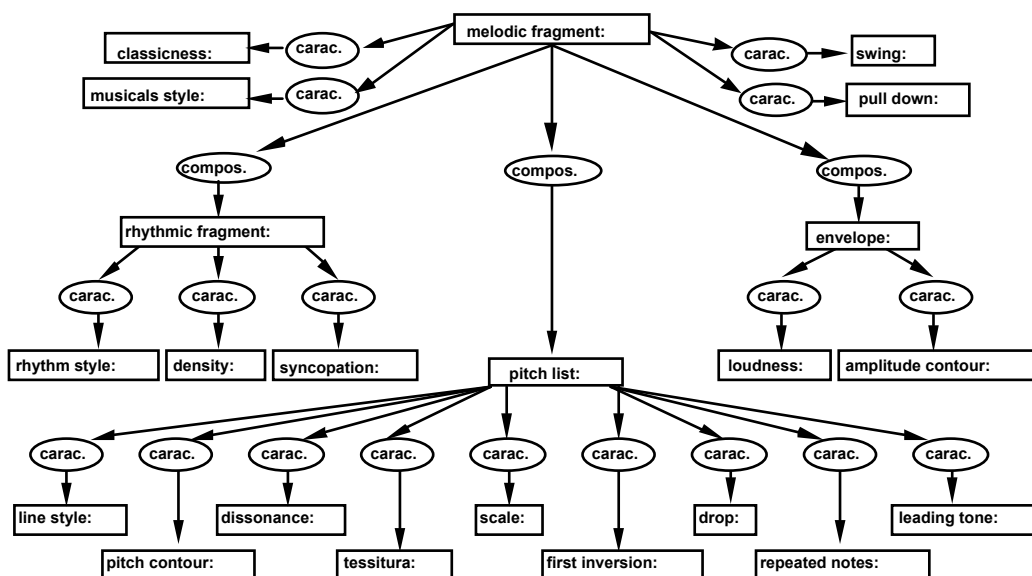


Figure 5.1.1- Descriptions standards représentées par un graphe conceptuel (comp. = composé de; carac. = caractérisé par).

Néanmoins, une représentation des PACTs par des graphes conceptuels présente un inconvénient majeur : ils ne sont pas très modulaires. Si on veut représenter un segment de mélodie comme il faut, il est nécessaire d'ajouter encore beaucoup de nœuds et de relations au graphe de la figure 5.1.1. Par exemple, un fragment mélodique est composé de notes, qui à leur tour sont caractérisées par une durée, un début, une hauteur, une amplitude et un timbre. Une hauteur est caractérisée par une octave et une classe de hauteur (*pitch-class*). En plus, ces notes peuvent être dans une position particulière de la grille par rapport aux chorus, à la section ou aux mesures. Comme tout cela doit être représenté dans un seul graphe, non seulement la conception du

programme qui manipule le graphe devient plus complexe mais aussi des problèmes d'efficacité apparaissent. C'est d'autant plus vrai lorsque l'on doit représenter des ensembles d'objets (comme les notes, les accords, etc.) car les graphes conceptuels sont plus adaptés à une représentation en intention qu'en extension.

Ces difficultés ne se posent pas pour les langages de frames (ou de classes) puisqu'une la valeur d'un attribut peut être un frame (objet) aussi. On a toute la facilité de définir des frames "note", "accord", "mélodie", "hauteur", "grille", etc. Toutefois, dans les frames, les relations entre les attributs ne sont pas explicites comme dans les graphes conceptuels, les dépendances entre les descriptions étant gérées par le concepteur au niveau des procédures attachées (*demons*). Néanmoins, comme un attribut d'un frame est aussi un frame, rien n'empêche que l'on redéfinisse le frame "attribut" pour y ajouter ces relations (Cf. figure 5.2.2).

Malheureusement, les frames ne sont pas tout à fait adéquats pour représenter les PACTs. La première raison tient au fait qu'avec des procédures attachées aux attributs, il est compliqué de mettre en œuvre des opérations associés aux PACTs, comme combinaison et activation ainsi que mesure de compatibilité, de similarité et de jouabilité. La deuxième raison est que le contrôle des inférences dans les frames est assez délicat. Par exemple, la valeur d'un attribut peut être obtenue des trois façons différentes : par lecture directe de la facette "valeur" ou via les facettes "défaut" et "si-besoin". Si le frame en question n'a rien associé à ces facettes alors par héritage (multiple) on va chercher la valeur plus haut (Winston, 1992) [chap. 9]. A ce titre, un langage de classe ayant un héritage simple et pouvant définir les procédures en tant que méthodes au niveau de la classe est plus adéquat. La dernière et la plus importante des raisons de l'inadéquation aussi bien des graphes que des frames est conceptuelle. Les PACTs n'ont été pas conçues ni pour gérer des bases de connaissances ni pour faire de la classification, "tâches" pour lesquelles ces formalismes servent en principe. Les PACTs sont censées représenter des actions plus ou moins achevées. Activer, combiner, mesurer la jouabilité et la combinabilité des PACTs sont des opérations assez étrangères à ces formalismes. Même en ce qui concerne le stockage et la récupération de PACTs dans la Mémoire Musicale, il ne s'agit pas d'un problème de classification mais simplement de rendre l'accès plus rapide et fiable (Kolodner, 1993) [chap. 6 et 8].

Au vue de cette dernière critique, nous avons pensé représenter les PACTs, dans le cadre des langages d'acteurs, comme des multi-agents. Cependant, l'utilisation d'un langage d'acteurs ne serait vraiment utile que si nous n'avions pas établi un contrôle "non-centralisé" pour les inférences associées aux PACTs. Ce ne sont pas les PACTs elles-mêmes qui doivent décider de s'activer ou qui doivent interagir librement pour "faire émerger" une PACT jouable. Au contraire, les PACTs sont activées et assemblées par des mécanismes de contrôle "externes", correspondant à des bases de règles de production.

Il nous restait les langages de classes qui sont adéquats du point de vue conceptuel, car ils sont très généraux en ce qui concernent les domaines d'application. En outre, avec ces langages nous pouvions nous affranchir des problèmes évoqués pour les graphes conceptuels (modularité) et pour les frames (procédures et contrôle). Cependant, dans les langages de classe on ne représente pas les attributs (variables d'instances) en tant que classes, ce qui nous empêchait d'introduire les facettes et, du coup, les relations entre les attributs.

Nous avons finalement conçu un langage hybride à mi chemin entre les frames et les classes (inspiré de Smalltalk-80) et qui incorpore quelques idées des graphes conceptuels. Dans ce langage, les PACTs sont représentées comme des frames avec héritage simple, ayant des méthodes définies dans la classe au lieu des procédures attachées, et ayant des facettes associées aux attributs. Autrement dit, les PACTs sont des classes dont les variables d'instance ont été "réifiées" pour contenir des facettes. Quelques tentatives d'associer des facettes aux variables d'instance d'une classe, ont été déjà faites (Ferber, 1989; Jung, Mohr & Napoli, 1988; Nebel, 1985; Scheneider, 1988) D'une façon générale, les langages hybrides sont courantes en IA parce que les besoins de chaque chercheur sont divers. On peut citer par exemple KRL et LOOPS (Stefik & Bobrow, 1986), ou encore, en France, MERING (Ferber, 1985) et YAFOOL (Ducornau, 1988) parmi des dizaines d'autres. A cause des particularités des PACTs, nous avons finalement trouvé plus simple de développer notre propre langage, au lieu de réutiliser un des langages cités.

Puisque notre langage est hybride, il convient de clarifier la terminologie que nous adoptons dans la suite du document. Nous employons les mots "classe" et "instance" provenant des langages de classes au lieu du mot "frame" pour éviter

l'ambiguïté due au fait qu'un frame peut être aussi bien une classe représentant un ensemble d'objets qu'un objet en particulier. Nous gardons néanmoins les mots "attributs" et "facette" provenant des langages de frames.

5.2. LES PACTs STANDARDS

La figure 5.2.1 montre les liens d'héritage les différents types de PACTs : `StandardPact` incorpore les descriptions standards; `TransformingPact` incorpore les transformationnelles; `AccomplishedPact` est utilisé pour représenter les cas de la Mémoire Musicale; et `ToBeDiscretizedPact` est une sorte de PACT "intermédiaire". Les seules PACTs censées participer à l'assemblage sont les PACTs standards et les PACTs transformationnelles. Dans cette section, nous nous intéressons particulièrement aux PACTs standards, les plus complexes dans notre modèle. Dans la version actuelle de notre programme, ces derniers PACTs sont les seules à être vraiment impliquées dans le processus d'assemblage.

5.2.1. Représentation

Toute PACT est une sorte d'objet temporel. A ce titre, toute PACT possède un attribut `lapse`. Il existe trois différentes théories en IA sur la représentation du temps selon les primitives suivantes : intervalles (Allen, 1983; Allen, 1984), points (McDermott, 1982), et événements (Kowalski & Sergot, 1986). L'équivalence entre ces trois théories a été prouvée (Tsang & Aitken, 1991) : le choix de la primitive dépendant uniquement du modèle de résolution de problèmes. Nous avons adopté l'intervalle de temps comme primitive, c'est-à-dire un laps de temps ayant un début et une durée.

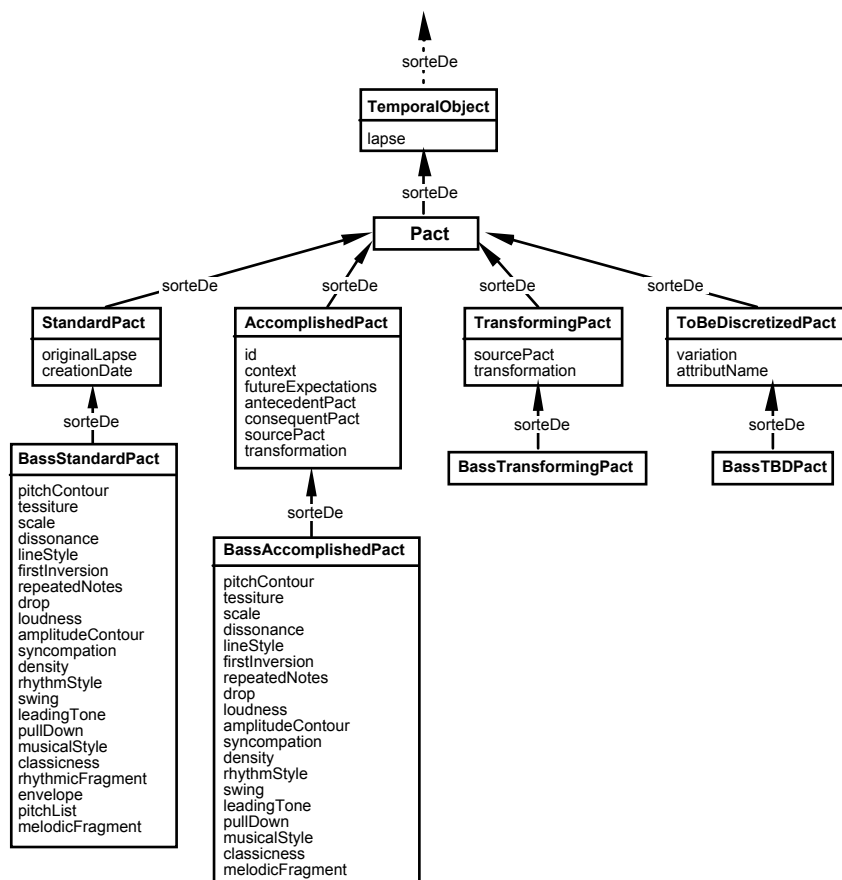


Figure 5.2.1 - Classes de PACTs (sorteDe = relation d’héritage entre classe de frames)

Parmi les sous-classes de la classe PACT, on trouve la StandardPact qui est une PACT contenant les descriptions standards dont nous avons parlé précédemment. Cette classe StandardPact a seulement deux attributs (originalLapse et creationDate). Comme nous expliquons plus tard, nous faisons appel à ces attributs pendant l’assemblage pour résoudre des conflits entre deux PACTs. StandardPact est une “classe abstraite” car elle n’a jamais d’instance. Une classe abstraite est utile dans la mesure où elle définit une partie des opérations et propriétés de ses sous-classes. Dans notre cas, cela signifie que nous pouvons très facilement, grâce à l’héritage, ajouter des nouvelles sous-classes de PACTs standards comme pour le piano, la batterie, la trompette, etc.

BassStandardPact est le type de PACT auquel nous faisons appel pour incorporer concrètement les descriptions standards d’un fragment de ligne de basse. La liste d’attributs est la même que celle de graphes conceptuels (Cf. figure 5.1.1).

Dans les langages de frames, les attributs sont aussi des frames, les facettes étant les attributs des attributs. C'est pourquoi nous avons défini la classe `Attribut` (Cf. figure 5.2.2). Cette classe contient tous les facettes normalement définies dans les langages des frames, sauf celles auxquelles sont attachées les procédures, à savoir `whenRead`, `whenWritten`, `whenRequested`, et `whenAdded` (nous les avons marquées en gris dans la figure 5.2.2). La facette `default`, qui normalement fournit une valeur par défaut lorsque qu'il n'y a rien associé à la facette `value`, n'est pas non plus représentée car aucune classe de notre système n'en a pas besoin¹⁸.

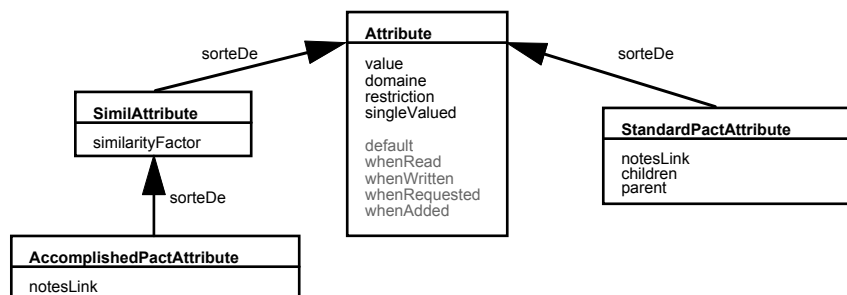


Figure 5.2.2 - Réification des attributs

Nous avons dû aussi créer d'autres sous-classes d'`Attribut`. Dans cette section nous parlerons de `StandardPactAttribute`, spécialement conçue pour représenter les attributs des PACT standards¹⁹. Les facettes `parent` et `children` reconstruisent la hiérarchie de descriptions standards (HDS) explicitée dans la représentation par des graphes conceptuels. Après quelques études, nous avons réalisé qu'en réalité nous n'avons pas besoin de garder les relations "composé de" et "caractérisé par", mais simplement les rapports hiérarchiques entre les nœuds. La HDS simplifiée (Cf. figure 5.2.3) nous suffit largement pour pouvoir réaliser les calculs (compatibilité, jouabilité, etc.) nécessaires à l'assemblage des PACTs standards. La facette `notesLink` est censée indiquer le lien direct avec les notes (souhaités ou connues): `primary`, `secondary` et `no`. Comme nous verrons par la suite, la

¹⁸En réalité, dans les premières versions de notre programme nous avons fait appel aux procédures attachées à l'aspect `whenWritten` et aux valeurs par défaut pour la représentation des PACTs standards, mais actuellement ces aspects sont obsolètes.

¹⁹Par "PACTs standards" nous entendons les instances des classes (par ex. `BaseStandardPact`) dont la super-classe est `StandardPact`.

caractérisation de ce type de lien est utile pour les inférences que l'on fait sur les PACTs standards. On considère, à cet égard, que les attributs `lapse`, `originalLapse`, `creationDate` n'ont pas de lien direct avec les notes. Les attributs `melodicFragment`, `rhythmicFragment`, `pitchList` et `envelope` ont un lien que l'on appelle "primaire" car il s'agit soit des notes elles-mêmes, soit de leurs projections simples sur les axes des durées, des hauteurs et des amplitudes, respectivement. Les autres attributs, qui correspondent à ce que nous avons appelé les traits musicaux, entretiennent un lien "secondaire" avec le fragment mélodique.

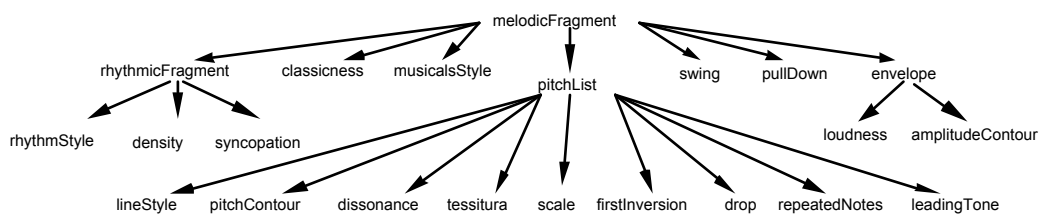


Figure 5.2.3 - Hiérarchie de descriptions standards (HDS) des PACTs standards de basse

La figure 5.2.4, montre les attributs et les facettes de la `BassStandardPact`. Il faudrait préciser que tous les attributs de `BassStandardPact` sont monovalués dans un sens un peu particulier car `melodicFragment`, `rhythmicFragment`, `pitchList` et `envelope` ont tout de même des listes d'objets comme domaine. Nous disons qu'ils sont monovalués pour signaler que, contrairement aux attributs multivalués, on n'a pas le droit d'ajouter ni de retirer un élément de ces listes. Cela signifie que la valeur de ces attributs est prise en tant qu'un tout indissociable correspondant à la durée de la PACT. Rappelons aussi que certaines valeurs de ces attributs de la `BassStandardPact` peuvent être des entités assez élaborées comme `MelodicFragment`.

5.2.2. Notation et autres notions de base

Avant de commencer à exposer les opérations impliquant les PACTs (standards ou pas), il est utile de définir quelques notations.

Nous adoptons la notation `pact.attribut.facette` pour la lecture d'une valeur d'une facette et `pact.attribut.facette <- nouvelleValeur` pour l'affectation d'une nouvelle valeur. Ce sont les notations de base des langages de

frames. Dans le cas particulier de la facette `value`, on adopte une notation plus succincte : `pact.attribut` pour la lecture et `pact.attribut <- nouvelleValeur` pour l'écriture.

<p><i>lapse</i> domain: Lapse restriction: notesLink: no children: () parent: () singleValued: true</p> <p><i>originalLapse</i> domain: Lapse restriction: notesLink: no children: () parent: () singleValued: true</p> <p><i>creationDate</i> domain: Date restriction: notesLink: no children: () parent: () singleValued: true</p> <p><i>pitchContour</i> domain: QualitativePitchContour restriction: (veryDesc, desc, hor, asc, veryAsc) notesLink: secondary children: () parent: (pitchList) singleValued: true</p> <p><i>tessiture</i> domain: QualitativeTessitura restriction: (low, medium, high) notesLink: secondary children: () parent: (pitchList) singleValued: true</p> <p><i>scale</i> domain: MusicalScale restriction: notesLink: secondary children: () parent: (pitchList) singleValued: true</p> <p><i>dissonance</i> domain: QualitativeIntensity restriction: (low, medium, high) notesLink: secondary children: () parent: (pitchList) singleValued: true</p> <p><i>lineStyle</i> domain: Symbol restriction: (arpeggio, stepwise) notesLink: secondary children: () parent: (pitchList) singleValued: true</p> <p><i>firstInversion</i> domain: Boolean notesLink: secondary</p>	<p>children: () parent: (pitchList) singleValued: true</p> <p><i>repeatedNotes</i> domain: Boolean notesLink: secondary children: () parent: (pitchList) singleValued: true</p> <p><i>drop</i> domain: Boolean notesLink: secondary children: () parent: (pitchList) singleValued: true</p> <p><i>leadingNote</i> domain: Boolean notesLink: secondary children: () parent: (pitchList) singleValued: true</p> <p><i>loudness</i> domain: QualitativeLoudness restriction: (pp, p, mf, f, ff) notesLink: secondary children: () parent: (envelope) dimensions: (amplitude) singleValued: true</p> <p><i>amplitudeContour</i> domain: QualitativeAmplitudeContour restriction: (veryDesc, desc, hor, asc, veryAsc) notesLink: secondary children: () parent: (envelope) singleValued: true</p> <p><i>syncopation</i> domain: QualitativeIntensity restriction: (low, medium, high) notesLink: secondary children: () parent: (rhythmicPattern) singleValued: true</p> <p><i>density</i> domain: QualitativeIntensity restriction: (low, medium, high) notesLink: secondary children: () parent: (rhythmicPattern) singleValued: true</p> <p><i>rhythmStyle</i> domain: QualitativeRhythmStyle restriction: (full, half, quarter, eighth) notesLink: secondary children: () parent: (rhythmicPattern) singleValued: true</p> <p><i>swing</i> domain: QualitativeIntensity</p>	<p>restriction: (low, medium, high) notesLink: secondary children: () parent: (melodicPattern) singleValued: true</p> <p><i>musicalStyle</i> domain: Symbol restriction: (bebop, swing, hot, cool, free) notesLink: secondary children: () parent: (melodicPattern) singleValued: true</p> <p><i>pullDown</i> domain: Boolean notesLink: secondary children: () parent: (melodicPattern) singleValued: true</p> <p><i>classiness</i> domain: QualitativeIntensity restriction: (low, medium, high) notesLink: secondary children: () parent: (melodicPattern) singleValued: true</p> <p><i>envelope</i> domain: Envelope restriction: notesLink: primary children: (amplitudeContour, loudness) parent: (melodicPattern) singleValued: true</p> <p><i>pitchList</i> domain: Liste d'hauteurs restriction: mi2 - mi5 notesLink: primary children: (pitchContour, tessiture, scale, dissonance, lineStyle, firstInversion, repeatedNotes, drop, leadingNote) parent: (melodicPattern) singleValued: true</p> <p><i>rhythmicFragment</i> domain: RhythmicFragment restriction: notesLink: primary children: (syncopation, density, rhythmStyle) parent: (melodicPattern) singleValued: true</p> <p><i>melodicFragment</i> domain: MelodicFragment restriction: notesLink: primary children: (swing, style, pullDown, classiness, rhythmicPattern, pitchList) parent: (melodicPattern) singleValued: true</p>
--	--	--

Figure 5.2.4 - Attributs et facettes de la *BassStandardPact*

Il est important, pour la suite, de bien définir ce que l'on entend par la spécialisation d'un attribut. Généralement, on dit qu'un attribut est spécialisé s'il a une valeur associée. Dans le cas des PACTs, pour que les attributs dont la valeur est un ensemble d'objets temporels soient spécialisés, il faut aussi que le laps de temps (début et durée) de cet ensemble d'objets soit égal au laps de temps de la PACT. Plus précisément,

$$\begin{aligned} \text{spécialisé}(p, a) \text{ ssi} & \quad (5.2.1) \\ & ((\neg \text{listeOrdonnéeEnTemps}(a) \wedge \neg(p.a = \text{nil})) \vee \\ & (\text{listeOrdonnéeEnTemps}(a) \wedge \neg(p.a = \text{nil}) \wedge (p.\text{lapse} = p.a.\text{lapse}))) \end{aligned}$$

Où, dans le cas de la `BassStandardPact`,

$$\begin{aligned} & \text{listeOrdonnéeEnTemps}(a) \\ \text{ssi} & [(p.a.\text{domain} = \text{MelodicFragment}) \vee \\ & (p.a.\text{domain} = \text{RhythmicFragment}) \vee \\ & (p.a.\text{domain} = \text{Enveloppe})] \end{aligned}$$

Nous définissons A^* comme l'ensemble d'attributs liés aux notes d'une PACT standard p .

$$A^* = \{a / p.a.\text{notesLink} \bullet \text{no}\} \quad (5.2.2)$$

L'ensemble S^* est défini comme une restriction de A^* aux attributs spécialisés.

$$S^* = \{a / (\text{spécialisé}(p, a_i) \wedge \neg(p.a_i.\text{notesLink} = \text{no}))\} \quad (5.2.3)$$

5.2.3. Activation et segmentation

Toutes les opérations impliquant les PACTs que nous présentons dans ce chapitre concernent des *PACTs activées*. Activer une PACT correspond simplement à générer une instance d'une classe particulière de façon à ce que l'instance ait certain nombre d'attributs spécialisés. Quant aux PACTs standards de basse, on active une PACT p en créant une instance de `BassStandardPact`²⁰ et en affectant des valeurs à l'attribut `lapse` et, au moins, à une des attributs "liés aux notes".

$$\begin{aligned} \text{activée}(p) & \quad (5.2.4) \\ \text{ssi} & (\text{spécialisé}(p.\text{lapse}) \wedge \neg \text{vide}(S^*)) \end{aligned}$$

Parmi les PACTs standards qui sont dans la mémoire de travail de l'agent, on sélectionne celles qui se superposent au segment courant pour constituer l'état initial de l'espace d'assemblage (Cf. section 4.3.3.1). Visant à faciliter le processus d'assemblage, on fait une "copie" des toutes les PACTs standards sélectionnées. Ensuite, on leur applique une opération de *segmentation* pour que leur attribut `lapse` ait la même valeur et pour qu'elles ne contiennent que l'information concernant le segment au niveau de leurs attributs de type "liste ordonnée en temps" (`melodicFragment`, `enveloppe` et `rhythmicFragment`). Précisément, la segmentation d'une PACT standard p concernant un segment de laps de temps lp est faite de la façon suivante :

²⁰La valeur de `creationDate` est affectée automatiquement par la procédure de création de l'instance.

```

fonction segmenter (p, lp) : une PACT segmenté
p.originalLapse <- p.lapse.
p.lapse <- lp.
Pour tout a tel que spécialisé (p, a)  $\wedge$  listeOrdonnéeEnTemps (a), faire
  p.a <- segmenterListeOrdonnéeEnTemps (p.a, lp)
retourner p

```

Tableau 5.2.1 - Fonction segmenter

Il est néanmoins important de présenter la segmentation parce que toutes les comparaisons entre des PACTs standards (concernant leur combinabilité et jouabilité) que nous exposons par la suite, se font entre deux PACTs impliquées dans l'assemblage, et donc ayant un même laps de temps.

5.2.4. Jouabilité

La propriété de jouabilité d'une PACT standard est très importante pour deux raisons. D'abord, elle constitue le critère d'arrêt de l'assemblage des PACTs, le but étant justement une PACT jouable. Ensuite, elle établit un ordre partiel qui intervient dans la résolution de conflits entre deux PACTs incompatibles, la PACT la plus jouable étant préférée.

Pour savoir si une PACT standard est (complètement) jouable, il suffit de vérifier si elle contient toutes les notes concernant sa durée de vie. Etant donnée une PACT standard p

$$\text{jouable}(p) \text{ ssi } \text{spécialisé}(p, \text{melodicFragment}) \quad (5.2.5)$$

Il est cependant plus problématique de déterminer laquelle parmi deux PACTs standards est la plus jouable. Le cas le plus simple est celui où une PACT est "strictement plus jouable" qu'une autre. Formellement, étant données deux PACTs standards p_1 et p_2 , dont l'ensemble d'attributs spécialisés liés aux notes est respectivement S_1^* et S_2^* .

$$\text{strictementPlusJouable}(p_1, p_2) \quad (5.2.6)$$

$$\text{ssi } \forall_{a_2 \in S_2^*} \exists_{a_1 \in S_1^*} a_1(\text{ascendant}(a_1, a_2))$$

Où $\text{ascendant}(a, b)$ indique que, dans la HDS, l'attribut "a1" est parmi les parents de "a2".

L'exemple le plus évident d'une *jouabilité supérieure stricte* (où le prédicat `strictementPlusJouable` est satisfait) est celle où p_1 est jouable. En effet, comme l'attribut `melodicFragment` (qui est spécialisé puisque p_1 est jouable) est la racine de la HDS, tout attribut spécialisé de p_2 est forcément un descendant de `melodicFragment`, sauf si p_2 elle-même est jouable (dans ce dernier cas, on ne peut rien décider). Autre exemple de jouabilité supérieure stricte est le cas des PACTs $p_1 = [\text{rhythmicFragment} = \downarrow \downarrow \uparrow \downarrow]$ et $p_2 = [\text{density} = \text{high}, \text{syncopation} = \text{low}]$, car les deux attributs `density` et `syncopation` sont descendants de `rhythmicFragment` (voir la HDS figure 5.2.3).

Bien entendu, la plupart du temps on ne peut trancher sur quelle est la plus jouable entre deux PACTs standards. Par exemple,

- $p_1 = [\text{syncopation} = \text{high}]$ et $p_2 = [\text{density} = \text{high}, \text{syncopation} = \text{low}]$;
- $p_1 = [\text{rhythmicFragment} = \downarrow \downarrow \uparrow \downarrow, \text{pitchContour} = \text{asc}]$ et $p_2 = [\text{density} = \text{high}, \text{pitchList} = (\text{la}, \text{do}\#, \text{ré}, \text{mi})]$,

La jouabilité supérieure stricte étant indécidable, nous avons introduit la mesure de *jouabilité supérieure approximative*, basée sur le nombre d'attributs spécialisés.

$$\begin{aligned} \text{approximativementPlusJouable}(p_1, p_2) & \qquad (5.2.7) \\ \text{ssi } \text{Card}(S_1^*) & > \text{Card}(S_2^*) \end{aligned}$$

Pour alors savoir si une PACT standard est plus jouable qu'une autre, on teste d'abord la jouabilité stricte et si on ne peut trancher on teste la jouabilité approximative. Plus précisément, étant données deux PACTs standards p_1 et p_2 ,

$$\begin{aligned} \text{plusJouable}(p_1, p_2) & \qquad (5.2.8) \\ \text{ssi } (\text{strictementPlusJouable}(p_1, p_2) \vee & \\ \text{approximativementPlusJouable}(p_1, p_2)) & \end{aligned}$$

5.2.5. Préférence

Pendant l'assemblage, lorsque que deux PACTs (standards) ne peuvent se combiner, on doit appliquer l'opérateur "enlève" pour éliminer l'une d'entre elles (Cf. section 4.3.3.2). Nous optons pour une PACT plutôt qu'une autre selon deux critères. Le premier est la jouabilité : on préfère la PACT la plus jouable pour être plus proche du but. Le deuxième critère est temporel. Au cas où la jouabilité n'est pas suffisante, nous préférons la PACT dont la date de création est la plus récente et, si cela ne suffit pas, la

PACT qui va durer le plus longtemps. Il s’agit d’un double souci “d’urgence” et de “continuité”. Plus la date de création d’une PACT standard est récente, plus cette PACT est censée représenter fidèlement ce qui se passe dans l’environnement. Par exemple, entre $p_1 = [\text{loudness} = \text{ff}, \text{creationDate} = 1234]$ et $p_2 = [\text{loudness} = \text{mf}, \text{creationDate} = 200]$, on préfère la première. En ce qui concerne la continuité, nous préférons la PACT qui va encore durer longtemps car cela contribue à ce que la ligne de basse soit plus homogène. Par exemple, on préfère la deuxième entre $p_1 = [\text{loudness} = \text{ff}, \text{lapse} = (16,8)]$ et $p_2 = [\text{loudness} = \text{mf}, \text{lapse} = (16,24)]$.

```
préféréeParRapportAuTemps (p1, p2) (5.2.9)
  ssi (p1.creationDate(p1, p2) ∨ finitAprès(p1.lapse, p2.lapse))
```

La préférence entre deux PACTs standards p_1 et p_2 est alors faite à partir de deux critères introduits selon la fonction ci-dessous. Dans le cas rarissime où on ne peut départager les deux PACTs, on choisit arbitrairement p_1 .

fonction <code>factPréféréeEntre</code> (p_1, p_2): la PACT préférée	
si <code>plusJouable</code> (p_1, p_2)	alors retourner p_1 .
si <code>plusJouable</code> (p_2, p_1)	alors retourner p_2 .
si <code>préféréeParRapportAuTemps</code> (p_1, p_2)	alors retourner p_1 .
si <code>préféréeParRapportAuTemps</code> (p_2, p_1)	alors retourner p_2 .
retourner p_1	

Tableau 5.2.2 - Fonction `factPréféréeEntre`

5.2.6. Combinabilité

Nous avons jusqu’ici parlé “d’incompatibilité” entre deux PACTs standards parce que c’est un mot évocateur. Cependant, le plus juste serait de parler de “combinabilité”. Il ne suffit pas de vérifier si les PACTs contiennent des informations *conflictuelles* mais aussi de vérifier si elles contiennent des informations *redondantes*, car on n’a pas intérêt à combiner des informations redondantes. En effet, la question que l’on se pose avant d’appliquer l’opérateur “combine” est la suivante : peut-on avoir une PACT plus jouable en combinant telle et telle PACTs ? A cet égard, les PACTs $p_1 = [\text{syncopation} = \text{low}]$ et $p_2 = [\text{syncopation} = \text{high}]$ sont aussi “non combinables” que $p_1 = [\text{syncopation} = \text{high}]$ et $p_3 = [\text{syncopation} = \text{high}]$, car elles concernent toutes les mêmes attributs (`syncopation`). De même, la PACT $p_4 = [\text{rhythmicFragment} = \downarrow \downarrow \uparrow \downarrow]$ ne peut être combinée ni avec $p_5 = [\text{density} = \text{veryLow}]$, ni avec $p_6 = [\text{density} = \text{medium}]$. De fait, sachant que `rhythmicFragment` est “hiérarchiquement plus jouable” (voir HDS) que `density`, on peut dire que la PACT p_4 est en “conflit

d'intégrité" avec p_5 car la densité affichée ne correspond pas à la densité intrinsèque du fragment rythmique. D'autre part, la densité affichée par la PACT p_6 correspond à celle du fragment rythmique affiché en p_4 . Toutefois, il serait redondant de combiner p_4 et p_6 .

5.2.6.1. Les trois cas "canoniques"

Il y a trois cas "canoniques" de combinabilité entre deux PACTs (standards). Avant de les étudier, il faut introduire quelques prédicats. On dit qu'une PACT standard p_1 a un *conflit ou redondance directe* avec une PACT p_2 au niveau de l'attribut a ($a \in A^*$), si a est spécialisé dans les deux PACTs.

$$\begin{aligned} & \text{conflitOuRedondanceDirecte}(p_1, p_2, a) && (5.2.10) \\ & \text{ssi } (\text{spécialisé}(p_1, a) \wedge \text{spécialisé}(p_2, a)) \end{aligned}$$

On dit qu'une PACT standard p_1 a un *conflit ou redondance hiérarchique* avec une PACT p_2 au niveau de l'attribut "a" ($a \in A^*$), si parmi les attributs spécialisés de p_2 il y en a un qui est ascendant de l'attribut a dans la HDS.

$$\begin{aligned} & \text{conflitOuRedondanceHiérarchique}(p_1, p_2, a) && (5.2.11) \\ & \text{ssi } (\text{spécialisé}(p_1, a) \wedge (\exists_{b \in S_2^*} b \text{ ascendant}(b, a))) \end{aligned}$$

Le premier cas "canonique" est celui de la *couverture directe totale* entre les descriptions contenues dans les deux PACTs p_1 et p_2 .

$$\begin{aligned} & \text{couvertureDirecteTotale}(p_1, p_2) && (5.2.12) \\ & \text{ssi } \forall_{a \in A^*} a(\text{conflitOuRedondanceDirecte}(p_1, p_2, a)) \\ & \text{ou simplement} \\ & \text{couvertureDirecteTotale}(p_1, p_2) \text{ ssi } S_1^* \sqsupseteq S_2^* \end{aligned}$$

On n'a pas intérêt à combiner deux PACTs standards ayant une couverture directe totale, car l'une n'exprime alors rien de plus de ce que l'autre exprime. C'est le cas de l'exemple des PACTs p_1 , p_2 et p_3 donné au début de cette section. D'autres exemples de couverture directe totale sont:

- $p_x = [\text{syncopation} = \text{high}]$ et $p_z = [\text{density} = \text{high}, \text{syncopation} = \text{low}]$;
- $p_x = [\text{melodicFragment} = \text{$

Le deuxième cas “canonique” concerne deux PACTs standards ayant une *couverture hiérarchique totale*. Les PACTs entretenant ce type de relation ne sont pas combinables.

$$\text{couvertureHiérarchiqueTotale}(p_1, p_2) \quad (5.2.13) \\ \text{ssi } \forall_{a \in S_1^*} a(\text{conflitOuRedondanceHiérarchique}(p_1, p_2, a))$$

ou simplement

$$\text{couvertureHiérarchiqueTotale}(p_1, p_2) \\ \text{ssi } \text{strictementPlusJouable}(p_1, p_2)$$

Dans ce type de situation, les attributs spécialisés ne sont pas les mêmes, mais ils sont redondants ou conflictuels par rapport à la HDS. Autrement dit, pour tout attribut spécialisé d’une PACT il y a toujours un attribut spécialisé plus haut dans la HDS dans l’autre PACT. C’est l’exemple des PACTs p_4 , p_5 et p_6 donné au début de cette section. Le cas extrême de couverture hiérarchique totale est celui où une des PACTs est jouable. Telle PACT a une relation de couverture hiérarchique totale avec n’importe quelle autre PACT non jouable. D’autres exemples de couverture hiérarchique totale sont :

- $p_1 = [\text{density} = \text{high}, \text{syncopation} = \text{low}]$ et $p_2 = [\text{rhythmicFragment} = \downarrow \downarrow \overline{\downarrow} \downarrow]$;
- $p_1 = [\text{density} = \text{high}, \text{pitchList} = (\text{la}, \text{do}\#, \text{ré}, \text{mi})]$ et $p_2 = [\text{pitchContour} = \text{asc}]$.

Le dernier cas “canonique” est celui où la couverture directe entre deux PACTs est nulle. Les informations contenues dans ces PACTs sont complémentaires.

$$\text{couvertureDirecteNulle}(p_1, p_2) \text{ ssi } (S_1^* \cap S_2^* = \emptyset) \quad (5.2.14)$$

Dans le cas où deux PACTs ont une couverture nulle et n’ont aucun conflit ou redondance hiérarchique, elles peuvent se combiner. Quelques exemples de ce type de situation sont :

- $p_1 = [\text{density} = \text{high}, \text{syncopation} = \text{low}]$ et $p_2 = [\text{rhythmicFragment} = \downarrow \downarrow \overline{\downarrow} \downarrow]$;
- $p_1 = [\text{swing} = \text{high}, \text{scale} = \text{Major}]$ et $p_2 = [\text{pitchContour} = \text{asc}]$.

La définition de couverture nulle est malheureusement trop restrictive, car elle exclue des situations comme $p_1 = [\text{density} = \text{high}, \text{syncopation} = \text{low}, \text{pitchContour} = \text{asc}]$ et $p_2 = [\text{swing} = \text{high}, \text{scale} = \text{Major}, \text{pitchContour} = \text{asc}]$ à cause de l’intersection

au niveau de l'attribut `pitchContour`. Et pourtant il serait souhaitable que ces deux PACTs se combinent. Pour éviter ce problème, nous étendons la définition de la couverture nulle.

$$\begin{aligned} & \text{couvertureDirecteNullePlus}(p_1, p_2) && (5.2.15) \\ & \text{ssi } ((S_1^* \cap S_2^* = \emptyset) \vee (\forall_{a \in (S_1^* \cap S_2^*)} a(p_1.a=p_2.a))) \end{aligned}$$

Permettre de combiner deux PACTs ayant une couverture nulle (s'il n'y a pas des conflits ou redondances hiérarchiques) est une simplification. On peut estimer que certains attributs, qui n'entretiennent pas un rapport d'ascendance dans la HDS, sont assez indépendants les uns des autres (par ex. `loudness`, `pitchContour`, `scale`, `density` et `syncopation`). Néanmoins, les attributs d'un même niveau dans la HDS ne sont pas vraiment orthogonaux, comme nous l'avons discuté à propos de l'interaction entre les traits musicaux (section 3.4.4). Par exemple, `scale` et `dissonance` sont très liés, et `density` et `rhythmStyle`, aussi, ainsi que plusieurs autres attributs. Le problème est que certains liens sont subtils, inconnus ou dépendants du contexte, bref, difficile à formaliser. Nous avons donc fait l'hypothèse que les attributs d'un même niveau dans la HDS sont indépendants. Cette simplification est, toutefois, compensée par le fait que c'est à l'aide de la Mémoire Musicale que l'on concrétise les divers attributs (représentant les traits musicaux) en termes de notes. Par ce procédé, nous sommes sûrs de toujours trouver un fragment mélodique de basse musicalement plausible, pour n'importe quelle configuration d'attributs spécialisés, fussent-ils en parfait accord ou en contradiction (Cf. Section 4.3.4.1). Cela peut être expliqué par deux raisons. D'abord, la Mémoire Musicale ne contient que des fragments mélodiques joués par des vrais musiciens et, donc, par définition, plausibles. Ensuite, la récupération d'un fragment mélodique est basé sur une mesure de similarité *globale* entre une PACT en voie d'assemblage et une PACT accomplie préalablement stockée (Cf. section 5.5.3). En d'autres termes, il n'est pas nécessaire que tous les attributs spécialisés d'une PACT aient les mêmes valeurs que ceux de l'autre PACT pour que les PACTs soient considérées similaires. Par exemple, on ne trouvera jamais un fragment de ligne de basse qui soit peu dissonant et qui, à la fois, utilise la gamme chromatique : soit l'un, soit l'autre selon les autres propriétés recherchées (densité, rythme, tessiture, etc.).

5.2.6.2. *Les autres cas*

Très souvent deux PACTs ne rentrent pas dans ces trois cas “canoniques” étudiés ci-dessus. Ainsi, prenons les deux PACTs suivantes :

- p1 = [density = high, syncopation = low, pitchContour = desc]
- p2 = [dissonance = low, pitchContour = asc]

Ces PACTs ne peuvent-elles être combinées car elles présentent un conflit au niveau de l’attribut `pitchContour`. Comme p1 est plus jouable p2, p1 sera préférée et l’information concernant la dissonance sera perdue. C’est bien dommage car plus on a d’information à la fin de l’assemblage, plus précise est la requête que l’on peut poser à la base de cas (Cf. Section 4.3.4.2). Dans ce sens, il serait souhaitable, par exemple, de permettre un type de combinaison un peu particulière qui produirait $p_{1+2} = [\text{density} = \text{high}, \text{syncopation} = \text{low}, \text{pitchContour} = \text{desc}, \text{dissonance} = \text{low}]$, où on “transfère” vers la PACT plus jouable p1 l’information sur la dissonance. Il faut donc pouvoir traiter ces cas “non canoniques” de combinabilité. D’autant plus qu’il y a un sérieux problème de contrôle de l’ordre des opératoins de combinaison des PACTs. Supposons que l’on ait les trois PACTs suivantes :

- p5 = [density = high, syncopation = low, pitchContour = desc]
- p6 = [dissonance = low]
- p7 = [pitchContour = asc].

Si p6 et p7 sont examinées d’abord, on va déclencher l’opérateur “combine” et elles seront combinées, ce qui mène à la situation de conflit citée plus haut (p1 contre p2). En revanche, si c’est p5 et p7 qui sont examinées, l’opérateur “enlève” va être appliqué sur p7, p5 étant plus jouable que p7. Dans ce dernier cas, on ne perdrait pas l’information sur la dissonance car p5 et p6 peuvent se combiner, $p_{5+6} = [\text{density} = \text{high}, \text{syncopation} = \text{low}, \text{pitchContour} = \text{desc}, \text{dissonance} = \text{low}]$.

Un autre exemple intéressant est celui de PACTs suivantes :

- p10 = [rhythmicFragment = $\downarrow \downarrow \downarrow \downarrow$, loudness = pp, pitchContour = desc]
- p11 = [density = high, scale = Chromatic, envelope = ((60,1), (80,5), (80,9))].

Il y a un “conflit ou redondance hiérarchique” de l’attribut *density* par rapport à l’attribut *rhythmicFragment* et de l’attribut *loudness* par rapport à l’attribut *envelope*. Néanmoins, il serait souhaitable d’obtenir par combinaison une PACT comme $p_{10+11} = [\text{rhythmicFragment} = \downarrow \downarrow \downarrow \downarrow \downarrow, \text{pitchContour} = \text{desc}, \text{scale} = \text{Chromatic}, \text{envelope} = ((60,1), (80,5), (80,9))]$, où on enlève les conflits et redondances hiérarchiques tout en maximisant la combinaison d’informations.

Il n’est pas difficile de trouver des heuristiques pour ordonner les combinaisons en évitant certaines situations comme celle de l’exemple “ p_5, p_6 et p_7 ” que nous venons de présenter. Il suffit de privilégier l’application de l’opérateur “enlève”. Néanmoins, si les PACTs ont beaucoup d’attributs spécialisés, parce que par exemple des combinaisons se sont déjà produites, on n’a plus d’heuristiques simples pour éviter une “perte d’information”. C’est pourquoi, nous avons décidé d’accepter ces types de combinaisons “non canoniques”, tout en imposant une façon particulière de les combiner qui respecte leurs degrés de jouabilité. C’est la PACT plus jouable qui doit devenir encore plus jouable. Par exemple, reprenant les deux PACTs p_1 et p_2 , on doit obtenir une combinaison $p_{1+2} = [\text{density} = \text{high}, \text{syncopation} = \text{low}, \text{pitchContour} = \text{desc}, \text{dissonance} = \text{low}]$ et pas $p_{1+2} = [\text{density} = \text{high}, \text{syncopation} = \text{low}, \text{pitchContour} = \text{asc}, \text{dissonance} = \text{low}]$, car p_1 est plus jouable que p_2 . Dans ces genres de situations, nous disons que la combinabilité est *partielle*.

La fonction qui calcule la combinabilité des deux PACTs standards p_1 et p_2 est affichée dans le tableau 5.2.3, le prédicat *pasDeConflictOrRedondanceHiérarchique* étant ainsi défini:

$$\begin{aligned} \text{pasDeConflictOrRedondanceHiérarchique}(p_1, p_2) \text{ ssi} & \quad (5.2.16) \\ \neg (\forall_{a \in S_1^*} (\text{conflitOuRedondanceHiérarchique}(p_1, p_2, a)) \wedge & \\ (\forall_{b \in S_2^*} (\text{conflitOuRedondanceHiérarchique}(p_2, p_1, b))) &) \end{aligned}$$

fonction combinabilité (p_1, p_2) : totale, partielle ou aucune
si ($\text{couvertureDirecteTotale}(p_1, p_2) \vee \text{couvertureDirecteTotale}(p_1, p_2) \vee$
 $\text{couvertureHiérarchiqueTotale}(p_1, p_2) \vee$
 $\text{couvertureHiérarchiqueTotale}(p_1, p_2)$)
alors retourner aucune.
si ($\text{couvertureDirecteNullePlus}(p_1, p_2) \wedge \text{pasDeConflictOrRedondanceHiérarchique}(p_1, p_2)$)
alors retourner totale.
retourner partielle

Tableau 5.2.3 - Fonction combinabilité

5.2.7. Combinaison

La combinaison entre deux PACTs standards (combinabilité totale ou partielle) s'effectue en deux étapes (Cf. tableau 5.2.4). Dans la première étape, la plus importante, il s'agit d'affecter les valeurs associées aux attributs d'une des PACTs aux mêmes attributs de l'autre. Les attributs en question sont ceux dont la liaison avec les notes est "primaire" ou "secondaire". Dans la première étape, on s'occupe des attributs relatifs au temps, à savoir `originalLapse` et `creationDate`.

```

fonction combiner (p1, p2) : une PACT p1+2
  si combinabilité (p1, p2) = totale
    alors combinaisonTotaleEntre (p1, p2) .
      ajusterParamètresDeTemps (p1, p2) .
    retourner p1
  si combinabilité (p1, p2) = partielle
    alors si plusJouable (p1, p2)
      alors combinaisonPartielleEntre (p1, p2)
        ajusterParamètresDeTemps (p1, p2) .
      retourner p1
    sinon combinaisonPartielleEntre (p2, p1) .
      ajusterParamètresDeTemps (p2, p1) .
    retourner p2

```

Tableau 5.2.4 - Fonction combiner

Lorsque la combinabilité est totale, peu importe la direction du transfert d'information entre les deux PACTs. Etant donné deux PACTs standards p_1 et p_2 , dont la combinabilité est totale, l'algorithme qui effectue une combinaison est le suivant :

```

fonction combinaisonTotaleEntre (p1, p2)
  Pour tout attribut  $b \in S_2$  faire,  $p_1.b \leftarrow p_2.b$ 

```

Tableau 5.2.5 - Fonction combinaisonTotaleEntre

Lorsque la combinabilité est partielle, le transfert d'information doit être dirigé vers la PACT la plus jouable. On transfère toutes les valeurs associées aux attributs de la PACT moins jouable vers la plus jouable à condition qu'il n'y ait ni de conflit, ni de redondance directe ou hiérarchique. Etant donné deux PACTs standards p_1 et p_2 , dont la combinabilité est partielle, p_1 étant la plus jouable des deux PACTs, l'opération de combinaison est exécutée à l'aide de la fonction suivante :


```

fonction combinaisonPartielleEntre ( $p_1, p_2$ )
1 Pour tout attribut  $a \in S_2^*$  faire
2 si  $\neg$  (spécialisé ( $p_1, a$ )  $\vee$ 
3   conflitOuRedondanceHiérarchique ( $p_2, p_1, a$ ))
4 alors  $p_1.a \leftarrow p_2.a$ .
5   si conflitOuRedondanceHiérarchique ( $p_1, p_2, a$ )
6     alors remplaceSousArbre ( $p_1, p_2, a$ )

```

Tableau 5.2.6 - Fonction combinaisonPartielleEntre

Supposons que les deux PACTs qui doivent se combiner (combinabilité partielle) soient $p_1 = [\text{rhythmicFragment} = \downarrow \downarrow \uparrow \downarrow, \text{loudness} = \text{pp}, \text{pitchContour} = \text{desc}, \text{scale} = \text{Chromatic}, \text{tessitura} = \text{high}]$ et $p_2 = [\text{density} = \text{medium}, \text{scale} = \text{Major}, \text{repeatedNotes} = \text{true}, \text{envelope} = ((60,1), (80,5), (80,9))]$. La PACT p_1 est plus jouable que p_2 par le critère de “jouabilité supérieure approximative” (Cf. formule 5.2.7). Les éléments de S_2^* sont density , scale , repeatedNotes et envelope . La valeur de density ne peut être transférée vers p_1 , car elle est hiérarchiquement redondante avec l’attribut rhythmicFragment de p_1 (ligne 3). La valeur de scale ne peut être transférée, car elle est directement conflictuelle avec l’attribut scale de p_1 (ligne 2). La valeur de repeatedNotes est transférée vers p_1 sans problèmes (ligne 4). La valeur de envelope est aussi transférée vers p_1 mais l’attribut loudness de p_1 est hiérarchiquement conflictuel avec envelope de p_2 (ligne 5). C’est la situation inverse des attributs density , où c’est un attribut de p_2 qui est hiérarchiquement conflictuel ou redondant. Quoique p_1 soit globalement plus jouable que p_2 , on autorise ce type de transferts car envelope est hiérarchiquement plus jouable que loudness . Néanmoins, pour maintenir la cohérence, on remplace toute la description qui, dans p_2 , concerne envelope , c’est-à-dire les attributs de la HDS dont envelope est un des parents. La PACT résultante de la combinaison est alors $p_{1+2} = [\text{rhythmicFragment} = \downarrow \downarrow \uparrow \downarrow, \text{pitchContour} = \text{desc}, \text{scale} = \text{Chromatic}, \text{repeatedNotes} = \text{true}, \text{envelope} = ((60,1), (80,5), (80,9))]$.

Etant données deux PACTs standards p_1 et p_2 ayant un ensemble A d’attributs, la fonction qui fait le remplacement du sous-arbre dont la racine est l’attribut a est ainsi définie :

```

fonction remplaceSousArbre ( $p_1, p_2, a$ )
Pour tout attribut  $c \in A$ , tel que descendant ( $c, a$ ) faire
   $p_1.c \leftarrow p_2.c$ .

```

Tableau 5.2.7 - Fonction remplaceSousArbre

L'algorithme *combinaisonPartielleEntre* est en réalité l'algorithme général de la combinaison entre deux PACTs. Il peut être employé aussi bien pour des combinaisons totales que partielles. Par ailleurs, si l'on l'applique à deux PACTs ayant une combinabilité nulle, aucun transfert d'information ne se produira.

Dans la deuxième étape de la combinaison, il s'agit d'ajuster les valeurs de *creationDate* et *originalLapse*. Comme nous avons discuté dans la section 5.3.3, ces deux attributs interviennent dans le calcul de préférence entre deux PACTs non combinables. Par conséquent, les valeurs de ces attributs doivent aussi subir des changements. L'idée derrière un tel ajustement est d'améliorer les chances de la PACT résultante d'être préférée en cas de conflit (Cf. section 5.2.5). C'est pourquoi parmi les deux dates de création, nous gardons la plus récente, et, parmi les deux durées de vie, nous gardons celle dont la fin est la plus lointaine (Cf. tableau 5.2.8).

```

fonction ajusterParamètresDeTemps (p1, p2) : modifie PACT p1
  si (p2.creationDate < p1.creationDate)
    alors p1.creationDate <- p2.creationDate.
  si (finitAprès(p2.originalLapse, p1.originalLapse))
    alors p1.originalLapse <- p2.originalLapse

```

Tableau 5.2.8 - Fonction *ajusterParamètresDeTemps*

5.2.8. Propagation

La dernière opération concernant les PACTs standards est la propagation. Comme discuté précédemment (Cf. sections 4.3.3.2 et 4.3.4.1), la propagation sert à enrichir la description d'une PACT standard par la spécialisation des nouveaux attributs à partir de l'information des attributs déjà spécialisés. Dans ce sens, toute propagation a un *attribut visé*, qui est celui auquel propagation compte affecter une valeur. Une propagation a deux étapes : d'abord, on vérifie si l'information contenue dans les attributs déjà spécialisés autorise le calcul la valeur de l'attribut visé et, ensuite, on déclenche le calcul de la valeur qui sera affectée à l'attribut visé. Nous avons montré (Cf. section 4.3.4.1) que le calcul de la valeur à affecter pourrait être mis en œuvre via des règles de production quand il s'agit des calculs simples où les attributs visés sont des traits musicaux comme *repeatedNotes*, *firstInversion* ou *pullDown* (Cf. Tableau 5.2.9). Par contre, quant l'attribut visé est *melodicFragment*, c'est-à-dire quand on veut déterminer les notes elles-mêmes, le calcul est fait via la Mémoire Musicale (Cf. tableau 5.2.10).

Nous avons conçu une base de règles de production qui fait la *gestion des déclenchements* des propagations. Cette base de règles joue en fait un rôle double, car elle met aussi en œuvre les calculs simples comme celui du tableau 5.2.9. A ces fins, nous utilisons le moteur d'inférence d'ordre un en chaînage avant qui est capable de manipuler des objets Smalltalk conçu au LAFORIA (Pachet, 1996). La règle ci-dessous montre comment le déclenchement du calcul de l'attribut visé `repeatedNotes` est réalisé. La commande `Modified` indique au moteur d'inférence qu'un objet a été changé, pour qu'il puisse de nouveau être "filtré"²¹, car le changement fait sur la PACT peut déclencher d'autres propagations. On s'arrête quand plus aucune règle est déclenchable.

<p>repeatedNotes Pour toute PACT standard de basse (BassStandardPact) p SI ¬ spécialisé(p,repeatedNotes). spécialisé(p,density). spécialisé(p,dissonance). p.density = high. p.dissonance = low. ALORS p.repeatedNotes <- true. Modified p.</p>

Tableau 5.2.9 - Règle *repeatedNotes*

Deux autres règles servent à déclencher le calcul des notes (attribut visé `melodicFragment`) via la Mémoire Musicale selon le seuil de propagation (Cf. section 4.3.4.1). La règle montrée au tableau 5.2.10 est en est un exemple.

<p>melodicFragmentHigh Pour toute PACT standard de basse p, et pour un seuil de propagation s SI haut(s). ¬ spécialisé(p,melodicFragment). toutsLesAttributsLienSecondaireSontSpécialiséesSaufMusicalStyleEtClassiness(p). ALORS calculerNotes(p, MemoireMusicale)</p>
--

Tableau 5.2.10 - Règle *melodicFragmentHigh*

Tout ce qui a été défini jusqu'à la section précédente concernant la jouabilité et combinabilité des PACTs standards est défini au niveau de la classe `StandardPact`.

²¹Le moteur utilise un réseau RETE pour le filtrage des faits.

Si l'on veut introduire un nouveau genre de PACT standard (pour le piano, batterie, etc.), il suffit de définir la HDS et bien initialiser les facettes des attributs. Toutefois, la propagation est tout à fait dépendante du domaine, de la sémantique de chaque attribut. C'est pourquoi, pour chaque nouvelle sous-classe de `StandardPact`, une base de gestion de propagation doit être conçue.

5.3. LES PACTS TRANSFORMATIONNELLES

Le deuxième classe de PACT est la `TransformingPact`. Ce genre de PACT représente ce que nous avons appelé les descriptions transformationnelles (Cf. section 5.1) à l'aide de deux attributs : `sourcePact`, auquel est affectée la PACT qui va subir la transformation, et `transformation` (Cf. figure 5.2.1).

Outre les problèmes que nous avons évoqués dans la première section de ce chapitre, à savoir la difficulté d'expliciter les connaissances nécessaires pour l'activation de ce genre de PACT, il n'est pas évident d'établir une ontologie semblable à celle des descriptions standards (la HDS), pour pouvoir décider quand une PACT transformationnelle est plus jouable qu'une autre ou incompatible avec une autre. D'abord, il faudrait trouver un répertoire de "transformations primitives" (comme celles que Schank a identifiées pour les dépendances conceptuelles (Schank, 1972) car le nombre possible de variations d'un fragment mélodique est très grand. Ensuite, il serait nécessaire établir des relations entre ces transformations pour savoir dans quel ordre elles doivent être appliquées. Enfin, il faudrait pouvoir décider de la jouabilité de telles PACTs. Par exemple, une "transposition" ou une "répétition" d'un fragment mélodique est toujours jouable tandis qu'une transformation comme "répéter le rythme et changer le reste" ne l'est pas. Pour ces raisons nous avons décidé de ne pas nous occuper de ce type de PACT dans la première version de notre programme afin d'évaluer ce que notre programme peut déjà faire avec des représentations plus simples.

5.4. LES PACTS "A DECOUPER"

De la façon dont `BassStandardPact` est définie, on n'est pas capable d'exprimer certaines actions comme "jouer de plus en plus dissonant" ou "jouer de moins en moins de notes", bien que l'on puisse dire "jouer de très dissonant" ou "jouer avec peu notes".

Cela est dû à notre intention de décrire surtout un fragment mélodique (achevé ou à achever). En effet, il est plus pertinent de parler de variation de dissonance ou de densité lorsqu’il s’agit de toute une section, un chorus, voire de l’improvisation entière, que lorsqu’il s’agit d’un fragment mélodique de courte durée. Ceci n’est pas le cas par exemple de `pitchContour` ou `amplitudeContour`, représentant respectivement des variations en hauteur et amplitude, qui peuvent être identifiées dans un fragment mélodique.

Bien que notre programme joue fragment par fragment nous avons besoin d’activer des PACT de type “jouer de plus en plus dissonant pendant le chorus d’improvisation” qui ont une durée de vie plus grande que les fragments que nous utilisons. Pour étendre le langage, une première solution serait d’avoir une fonction en tant que valeur de certains attributs comme `dissonance` ou `density`. Ainsi, nous pourrions représenter aussi bien des valeurs fixes que des variations. Mais ce type de solution rend nettement plus compliqué la détermination de la combinabilité de deux PACTs et leur combinaison proprement dite. D’autant plus que les valeurs des ces attributs sont qualitatives. Par conséquent, nous avons opté pour la définition d’un quatrième type de PACT : `ToBeDiscretizedPact`. Les PACTs à découper ont deux attributs : `attributName`, auquel le nom d’un des attributs d’une PACT standard est affecté et `variation` qui contient un segment de droite défini par les coordonnées (début-pact, valeur-initiale-d’attributName) et (fin-pact, valeur-finale-d’attributName) (Cf. figure 5.4.1). Contrairement aux PACTs standards, les PACTs à découper “traînent” dans la mémoire de travail de l’agent sans jamais être impliquées directement dans l’assemblage. A chaque nouveau segment d’improvisation, s’il y a une PACT à découper qui se superpose à tel segment, alors on génère une PACT standard dont la valeur de l’attribut en question (`attributName`) est calculée à partir d’une simple discretisation de la valeur de l’attribut `variation`.

attributName domain: Symbol restriction: (dissonance, density, syncopation) singleValued: true	variation domain: Edge restriction: singleValued: true
--	--

Figure 5.4.1- Attributs et facettes de la *BasstBDPact*.

Selon l’instrument, on doit définir quels sont les attributs qui peuvent être variés (par ex. on suppose que la valeur de l’attribut `scale` ne varie pas, car il n’est pas courant que l’on entende dire “jouer de plus en plus dans le mode dorien”), et dans quelle

échelle de temps la variation se produit (à l'intérieur d'un segment ou dans une granularité plus grande). C'est pourquoi il faut ajouter des sous-classes de `ToBeDiscretizedPact`. Pour la basse (`BassTBDPact`), les attributs sont définis comme le montre la figure 5.4.1.

Quant aux opérations impliquant une PACT à découper, parler de jouabilité, de combinabilité et de tout autre genre de calcul intervenant pendant l'assemblage n'a pas de sens. L'activation d'une PACT à découper est faite par la création d'une instance d'une sous-classe de `ToBeDiscretizedPact` et la spécialisation de tous les attributs, puisqu'il n'y a pas d'attribut de type "séquence temporelle" (voir formule 5.2.4). Le seul détail sur l'activation est qu'en pratique on n'affecte pas directement la valeur de la variation. Celle-ci est calculée automatiquement à partir des valeurs qualitatives initiales et finales de la variation et du laps de temps de la PACT. Par exemple, en disant que l'on veut une densité variant de très faible à très fort pendant une durée de deux mesures, la variation est considérée comme "très ascendante".

L'autre opération associée à ce genre de PACT est justement son découpage. Le découpage d'une PACT à découper p , selon un laps de temps lp (laps du segment courant), génère une PACT standard à l'aide de la fonction discrétiser ci-dessous.

```

fonction discrétiser (p, lp) : a PACT standard
  standp <- nouvelleInstance(BassStandardPact).
  standp.lapse <- lp.
  standp.attName <- valeurDiscret(p, lp.startBeat, p.attName.domain)
  retourner standp

```

Tableau 5.4.1 - Fonction discrétiser

5.5. LES PACTS DE LA MÉMOIRE MUSICALE

Comme nous l'avons déjà dit, la base de cas (Mémoire Musicale) de notre agent est constituée de fragments mélodiques joués par un bassiste humain, ces fragments étant stockés sous la forme de PACTs jouables (Cf. section 4.3.4.1). Ces PACTs pourraient être représentés par des PACTs standards mais nous avons préféré concevoir un troisième type de PACT, appelée *PACT accomplie*, pour les représenter (Cf. figure 5.2.1). Les raisons pour un tel choix sont multiples. Les PACTs de la Mémoire Musicale, comme elles sont jouables, ont à la fois un "comportement" (ensemble d'opérations associées) et une "structure" (ensemble d'attributs) différentes des PACTs

standards en voie d'assemblage. En effet, les PACTs de la Mémoire Musicale ne sont pas impliquées dans des opérations comme mesure de jouabilité, de préférence et de combinabilité, ainsi que combinaison et propagation. D'autre part, ces PACTs sont impliquées dans des opérations, comme des mesures de similarité ou ce que nous appelons d'anti-propagations (Cf. section 5.5.2), qui ne concernent pas les PACTs standards en voie d'assemblage. Les PACTs de la Mémoire Musicale servent uniquement à stocker et à indexer les fragments de façon à rendre rapide et adéquate leur récupération (Cf. section 4.3.4.1). Par ailleurs, ces PACTs ont aussi une structure particulière car nous avons besoin d'y représenter le contexte (accords, schéma d'accord, position dans la grille, etc.) dans lequel le fragment a été joué. De fait, ces données du contexte sont prises en compte pendant la récupération d'un fragment mélodique de la Mémoire Musicale (Cf. section 4.3.4). Enfin, ces PACTs ont à la fois les attributs des PACTs standards et ceux des PACTs transformationnelles, car un fragment mélodique peut être décrit aussi bien par ses traits que par des transformations concernant un autre fragment.

En somme, compte tenu de ces raisons, l'utilisation d'un nouveau type de PACT procure plus de modularité à la représentation et simplifie la tâche de conception. Naturellement, nous pouvions, en faisant usage de l'héritage multiple, concevoir les PACTs accomplis en tant que sous-classes de `StandardPact` et `TransformingPact`. Toutefois, l'héritage multiple pose souvent plus de problème, au niveau du contrôle et de la conception, qu'il n'en résout.

<p><i>lapse</i> domain: Lapse restriction: singleValued: true similarityFactor: 0 notesLink: no</p> <p><i>id</i> domain: NaturalNumber restriction: singleValued: true similarityFactor: 0 notesLink: no</p> <p><i>context</i> domain: LocalContext restriction: similarityFactor: 1 singleValued: true notesLink: no</p> <p><i>transformation</i> domain: Transformation restriction: similarityFactor: 0 singleValued: true notesLink: no</p> <p><i>sourcePact</i> domain: AccomplishedPact restriction: similarityFactor: 0 singleValued: true notesLink: no</p>	<p><i>futureExpectations</i> domain: restriction: similarityFactor: 0 singleValued: true notesLink: no</p> <p><i>antecedentPact</i> domain: AccomplishedPact restriction: similarityFactor: 0 singleValued: true notesLink: no</p> <p><i>consequentPact</i> domain: AccomplishedPact restriction: similarityFactor: 0 singleValued: true notesLink: no</p> <p><i>pitchContour</i> similarityFactor: 2</p> <p><i>tessiture</i> similarityFactor: 1</p> <p><i>scale</i> similarityFactor: 1</p> <p><i>dissonance</i> similarityFactor: 1</p> <p><i>lineStyle</i> similarityFactor: 4</p>	<p><i>firstInversion</i> similarityFactor: 1</p> <p><i>repeatedNotes</i> similarityFactor: 1</p> <p><i>drop</i> similarityFactor: 1</p> <p><i>leadingNote</i> similarityFactor: 1</p> <p><i>loudness</i> similarityFactor: 1</p> <p><i>amplitudeContour</i> similarityFactor: 1</p> <p><i>syncopation</i> similarityFactor: 1</p> <p><i>density</i> similarityFactor: 1</p> <p><i>rhythmStyle</i> similarityFactor: 2</p> <p><i>swing</i> similarityFactor: 1</p> <p><i>musicalStyle</i> similarityFactor: 1</p> <p><i>pullDown</i> similarityFactor: 1</p> <p><i>classiness</i> similarityFactor: 1</p> <p><i>melodicFragment</i> similarityFactor: 0</p>
---	--	--

Figure 5.5.1- Attributs et facettes de la *BassAccomplishedPact*. A partir de l'attribut *pitchContour* nous avons simplifié les descriptions car elles sont identiques à celles de la *BassStandardPact* (Cf. figure 5.2.4).

5.5.1. Représentation

Nous avons créé deux classes supplémentaires d'attributs (Cf. figure 5.2.2). La première est *SimilAttribut*, qui est utilisée par toutes les entités impliquées dans des mesures de similarité. La facette *similarityFactor* est utile pour la pondération de la mesure de similarité entre deux PACTs lors de la récupération d'un cas source de la Mémoire Musicale. La deuxième est *AccomplishedPactAttribute* qui est le type d'attribut des PACTs accomplies. En plus de *similarityFactor*, cette classe d'attribut a la facette *notesLink*, dont nous avons déjà parlé lors de la présentation de la classe *StandardPactAttribute*. La figure 5.5.1 montre les attributs de *BassAccomplishedPact*. Voyons précisément la signification des attributs dont nous avons pas encore parlé (ceux qui sont partagés par les PACTs standards et transformationnelles n'ayant plus besoin d'être examinés).

Il y a d'abord l'attribut *context*, auquel on attache une instance de la classe *LocalContext* qui contient des informations concernant la grille (tempo, style et nom de la grille, version de la mélodie —par ex. enregistré à New York en 1964, nom du joueur) et en particulier le segment de la grille sur lequel le fragment mélodique a été joué (Cf. figure 5.5.2).

<p>tempo domain: QualitativeTempo restriction: verySlow(40)-veryFast(360) singleValued: true similarityFactor: 6</p> <p>gridStyle domain: Symbol restriction: (Standard, Blues) singleValued: true similarityFactor: 1</p>	<p>gridName domain: String restriction: singleValued: true similarityFactor: 1</p> <p>melodyVersion domain: String restriction: singleValued: true similarityFactor</p>	<p>performer domain: String restriction: singleValued: true similarityFactor: 0</p> <p>currentChunk domain: ChordChunk restriction: singleValued: true similarityFactor: 20</p>
--	---	---

Figure 5.5.2 - Attributs et facettes de LocalContext.

Dans un contexte local, l'entité dont la représentation est la plus riche est le schéma d'accord. Le schéma d'accord est représenté par la classe ChordChunk (Cf. figure 5.5.3), qui a les attributs suivants: *shape* (la forme du schéma); *rhythmicStructure* (les durées de chaque accord); *tonality* (la tonalité locale); *backwardResolution* (l'intervalle entre les toniques de l'accord juste avant le schéma et du premier accord du schéma); *forwardResolution* (l'intervalle entre les toniques du dernier accord du schéma et du premier accord juste après le schéma); *position* (la position du schéma dans un chorus); *section* (la section où se trouve le schéma) et *chorus* (le chorus où se trouve le schéma); *chords* (les accords). Le tableau 5.5.1 illustre quelques schémas de la grille de STELLA BY STARLIGHT représentés comme des instances de ChordChunk (Cf. figure 5.5.3). Tous ces paramètres sont calculés par le programme pendant la phase de segmentation de la grille (Cf. section 6.3).

<p>lapse domain: Lapse restriction: singleValued: true similarityFactor: 0</p> <p>shape domain: Symbol restriction: singleValued: true similarityFactor: 5</p> <p>rhythmicStructure domain: Array restriction: singleValued: true similarityFactor: 4</p> <p>tonality domain: Scale</p>	<p>restriction: singleValued: true similarityFactor: 2</p> <p>backwardResolution domain: MusicalInterval restriction: singleValued: true similarityFactor: 1</p> <p>forwardResolution domain: MusicalInterval restriction: singleValued: true similarityFactor: 1</p> <p>position domain: Symbol restriction: (beginning, other, turnaround, turnback)</p>	<p>singleValued: true similarityFactor: 1</p> <p>chorus domain: Symbol restriction: (theme, improvisation) singleValued: true similarityFactor: 1</p> <p>section domain: Number restriction: (1,2,3,4) singleValued: true similarityFactor: 1</p> <p>chords domain: liste de PlayableChords restriction: singleValued: false similarityFactor: 0</p>
--	---	--

Figure 5.5.3- Attributs et facettes de ChordChunk.

Deux autres attributs importants d'une PACT accomplie sont *consequentPact* et *antecedentPact*. Ceci tient au fait qu'un cas correspond précisément à un couple de PACTs accomplies : la *PACT conséquente* correspondant au fragment mélodique que nous voulons récupérer plus tard de la Mémoire Musicale et la *PACT antécédente* correspondant au fragment joué juste avant (Cf. section 4.3.4.2). Le rôle de la PACT

antécédente est d'améliorer l'indexation à travers l'élargissement de la représentation du contexte dans lequel la PACT consécutive a été jouée.

attributs	cc10-11*	cc24	cc33-34
lapse**	37-(8)-45	89-(8)-97	129-(8)-137
shape	MinorTwoFiveOne	Major	MinorTwoFive
rhythmicStructure	(2 2 4)	(8)	(4 4)
tonality	D HarmonicMinorScale	Bb MajorScale***	D HarmonicMinorScale
backwardResolution	Augmented Fourth	Major Second	Augmented Fourth
forwardResolution	Perfect Fourth	Augmented Fourth	Minor Third
position	other	turnaround	beginning
chorus	theme	theme	improvisation
section	2	3	1
chords	Emin7(b5) A7 Dmin7	Bbmaj7 Bbmaj7	Emin7(b5) A7

Tableau 5.5.1 - Quelques schémas d'accords de STELLA BY STARLIGHT. (*les numéros indiquent les mesures de la grille; ** le laps de temps est affiché comme début-(durée)-fin; ***tonalité la plus plausible)

L'attribut `id` sert simplement à indexer les PACTs qui composent un cas lors de la sauvegarde de la base de cas. Quant à l'attribut `futureExpectations`, il contient quelques informations sur ce qui a été joué après le segment correspondant à la PACT consécutive. Dans la version actuelle du programme, nous avons réduit ses informations simplement à la première note jouée après la PACT consécutive. Connaître telle note est une information très utile lors de l'adaptation du fragment mélodique récupéré, notamment en ce qui concerne la transposition (Cf. section 6.6.2).

5.5.2. Acquisition de cas et anti-propagation

La construction de la base de cas est une étape très importante dans la conception d'un système de raisonnement à partir de cas. Dans notre système, où les cas sont des fragments mélodiques représentés sous la forme de PACTs accomplies, l'acquisition de cas s'effectue en trois étapes. La première étape est la *saisie des partitions et de grilles*, où il est question de prendre des transcriptions des lignes de basse, comme celles faites par Aabersold (Aabersold, 1979) et des grilles rencontrées dans les *real/fake books* (Sher, 1979; Sher, 1991), et de les mettre sous un format compréhensible pour la machine. C'est un travail répétitif et long, mais nécessaire. Il faut signaler que nous avons essayé de faire une saisie automatique, soit à partir des fichiers MIDI, soit à partir des enregistrements sonores. Malheureusement, les lignes de basse des vrais jazzmen ne sont pas disponibles sous forme de fichier MIDI. On n'est pas non plus capable de décoder avec exactitude des sons polyphoniques des albums de jazz pour les mettre

sous un format MIDI. Ainsi, nous avons dû effectuer une saisie manuelle qui, en plus d’être fastidieuse, ne permet pas de tenir en compte les amplitudes et les durées exactes des notes. Toutes les notes composant les fragments stockés ont la même amplitude et ont des durées standards (noire, croche, etc.). C’est un grave handicap pour notre système car nous avons perdu les informations qui caractérisent le mieux le swing.

La deuxième étape consiste à choisir quels fragments mélodiques nous allons capturer. Bien sûr, comme le segment d’improvisation de notre agent est égal à un schéma d’accord, notre acquisition des cas se centre sur les fragments mélodiques joués sur des schémas d’accords. Mais, puisqu’il y a plusieurs façons de segmenter une grille (Cf. section 6.3), quels fragments choisir et combien ? Par exemple, la figure 5.5.4 montre quelques fragments de ligne de basse que l’on peut obtenir à partir des quatre premières mesures de STELLA BY STARLIGHT jouée par Ron Carter.

Figure 5.5.4 - Fragments possibles de ligne de basse à être acquis

Le seul critère que nous avons employé pour le choix des fragments a été la “complétude” de la base de cas (Smyth & Keane, 1995b). Nous avons essayé de ne pas avoir de fragments trop similaires et d’avoir un nombre minimal de fragments pour chaque type de schéma dans la Mémoire Musicale. Contrairement à ce qui Hodgson (Hodgson, 1996b) a fait, nous n’avons donc pas utilisé des critères de “typicalité” pour le choix des fragments mélodiques. Les fragments mélodiques utilisés dans le programme de Hodgson correspondent à des patterns, au sens où ce sont des phrases mélodiques récurrentes. Notre base de cas contient aussi bien des phrases du type pattern que celles qui ne le sont pas. Nous utilisons la distinction entre un pattern et un fragment mélodique peu récurrent (l’aide de l’attribut `classiness`) dans la stratégie de raisonnement de l’agent. Par exemple, quand l’agent a peu de temps pour raisonner, il joue des patterns (clichés) (Cf. section 6.2).

La dernière étape est la représentation d'un couple de fragments mélodiques choisi en forme de PACTs accomplies. Pour représenter les cas nous avons défini la classe `Case` (Cf. figure 5.5.5).

<i>id</i> domain: Number restriction: singleValued: true similarityFactor: 0	<i>antecedentPact</i> domain: AccomplishedPact restriction: singleValued: true similarityFactor: 1	<i>consequentPact</i> domain: AccomplishedPact restriction: singleValued: true similarityFactor: 4
--	--	--

Figure 5.5.5 - Attributs et facettes de `Case`

Avec la fonction `acquérirCas` (Cf. tableau 5.5.2), les cas sont créés automatiquement à partir de la grille, de la ligne de basse et de deux laps de temps (`lp1` et `lp2`) correspondant, respectivement, à la PACT antécédente et à la PACT conséquent.

```

fonction acquérirCas(lp1, lp2, grille, melodie) : un cas cible
  cas <- nouvelleInstance(Case).
  cas.antecedentPact <- nouvelleInstance(AccomplishedPact).
  cas.consequentPact <- nouvelleInstance(AccomplishedPact).
  cas.antecedentPact.context <- calculerContext(lp1, grille).
  cas.consequentPact.context <- calculerContext(lp2, grille).
  cas.antecedentPact.noteSequence <- notesDansLaps(lp1, melodie).
  cas.consequentPact.noteSequence <- notesDansLaps(lp2, melodie).
  anti-propager(cas.antecedentPact).
  anti-propager(cas.consequentPact).
  retourner cas.

```

Tableau 5.5.2 - Fonction `createCase`

De cette fonction, nous voulons commenter seulement la notion de *anti-propagation* (Cf. tableau 5.5.3), une opération typique des PACTs accomplies (jouables). La propagation chez les PACTs standards en voie d'assemblage vise à spécialiser des attributs de plus en plus jouables (de plus en plus haut dans la HDS) jusqu'à calculer le fragment mélodique. L'anti-propagation, tout au contraire, consiste à calculer la valeur des attributs "plus abstraits" (les traits musicaux) d'une PACT accomplie en fonction de la valeur de l'attribut `melodicFragment` (la racine de la HDS). La figure 5.5.6 montre un exemple de cas obtenu à partir des quatre premières mesures de la ligne de basse affichée précédemment dans la figure 5.5.4. Ce sont les mêmes fragments dont nous avons déjà eu l'opportunité d'analyser les traits musicaux (Cf. section 2.2.2, figure 2.2.1).

```

anti-propager (pact)
Pour tout attribut a tel que (pact.a.notesLink = secondary) faire
pact.a <- calculerTrait(a, pact.melodicFragment, pact.context.currentChunk)

```

Tableau 5.5.3 - Fonction anti-propager

Nous n'allons pas rentrer dans les détails des calculs de chaque trait d'un fragment mélodique, néanmoins il faut souligner que certains sont loin d'être triviaux. Ainsi, les calculs des valeurs des attributs *syncopation*, *dissonance*, *lineStyle*, *scale*, *density*, entre autres, font intervenir des connaissances non consensuelles en musique tonale. Pour pouvoir les mener à bien nous avons dû faire une étude statistique des valeurs des traits musicaux de toutes les phrases (fragments) de ligne de basse, ce qui a exigé beaucoup de travail car au fur et à mesure que l'acquisition des cas progressait on devait réajuster les paramètres. Par exemple, on a fixé que si le ratio entre les notes qui tombent dans le temps et celles de contretemps est plus grand que 0,5 la "syncopité" est grande; s'il se trouve entre 0,28 et 0,5., la "syncopité" est moyenne; et s'il est plus petit que 0,28 alors elle est petite. Quant au calcul de la dissonance, il est plus complexe, car il faut tenir compte des trois éléments : la proportion entre les notes appartenant aux accords sous-jacents et celles qui n'y appartiennent pas; le placement de ces notes (temps et contretemps); et la présence d'inversions (tonique ne tombant pas au premier temps). Bien entendu, même en ayant des seuils "statistiquement plausibles" et des algorithmes convenables, une vérification et une correction humaine pour certains attributs, notamment *dissonance*, *scale* et *lineStyle*, est toujours nécessaire. D'autant plus qu'en réalité, le calcul des attributs *swing*, *classiness* et *musicalStyle* n'a pas pu être automatisé. Ce sont les raisons pour lesquelles nous avons conçu une interface d'acquisition permettant d'affecter manuellement les valeurs des attributs (Cf. figure 5.5.6).

The screenshot displays the Stella software interface for Case #2. It is divided into several sections:

- Case #2**: Contains navigation buttons (open, browse, modify, misc).
- Antecedent Pact** and **Consequent Pact**: Two columns of parameters for the musical cases.

Antecedent Pact	Consequent Pact
loudness: #mforte	loudness: #mforte
amplContour: #nor	amplContour: #nor
tessitura: #medium	tessitura: #medium
pitchContour: #nor	pitchContour: #asc
dissonance: #medium	dissonance: #high
scale: _____	scale: C ChromaticScale
density: #medium	density: #medium
rhytStyle: #quarterBased	rhytStyle: #quarterBased
syncopation: #medium	syncopation: #low
lineStyle: #arpeggio	lineStyle: #stepwise
first inversion: #true	first inversion: #true
repeated notes: #false	repeated notes: #false
leading tone: #true	leading tone: #true
pull down: #false	pull down: #false
drop: #true	drop: #false
musStyle: _____	musStyle: _____
classiness: _____	classiness: _____
swing: _____	swing: _____
lapse: (1)-(8)-9	lapse: (9)-(8)-17
- Antecedent Pact Context** and **Consequent Pact Context**: Two columns of contextual parameters.

Antecedent Pact Context	Consequent Pact Context
shape: #MinorTwoFive	shape: #MajorTwoFive
rhyStruct: #(4 4)	rhyStruct: #(4 4)
tonality: D HarmonicMinorScale	tonality: Bb MajorScale
bacwdRes: Augmented fourth	bacwdRes: Minor third
forwdRes: Minor third	forwdRes: unisson
position: #beginning	position: #other
section: 1	section: 1
chorus: #theme	chorus: #theme
chords: E halfDim7, A 7	chords: C min 7, F 7
version: Stella by Starlight	version: Stella by Starlight
performer: Ron Carter	performer: Ron Carter
gridName: Stella by Starlight	gridName: Stella by Starlight
gridStyle: standard	gridStyle: standard
tempo: 180	tempo: 180
- antecedent + consequent of Case #2**: A musical score for the antecedent, showing a bass line with chords E halfDim7, A 7, C min 7, and F 7.

Figure 5.5.6 - Cas obtenu à partir d'une ligne de basse de Ron Carter sur *STELLA BY STARLIGHT* (Aabersold, 1979)

5.5.3. Similarité

Pouvoir mesurer la similarité entre deux cas (cible-source) est central dans le raisonnement à partir de cas, puisqu'on cherche toujours le cas le plus similaire à un problème donné en espérant que la solution associée au cas récupéré est la plus adaptée (Cf. section 4.3.4.2). Avant de présenter le calcul de similarité des cas composant la Mémoire Musicale, il convient d'expliquer comment on représente une PACT standard en tant que PACT accomplie. De fait, le premier pas dans la récupération d'un cas de la Mémoire Musicale, est la "traduction" de la PACT standard, pour laquelle on cherche à spécialiser l'attribut `melodicFragment`, dans le cas cible (Cf. figure 5.5.7).

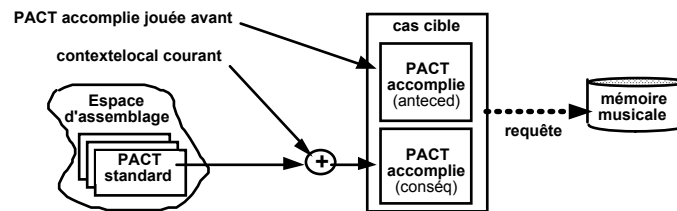


Figure 5.5.7 - Passage de la PACT standard vers le cas cible

Etant donné un contexte local *ctx* (composé du schéma d'accord courant, le tempo courant, le style de la grille, etc.), on peut obtenir une PACT accomplie à partir d'une PACT standard *p* à l'aide de la fonction `commeUnePactAccomplie` (Cf. tableau 5.5.4). C'est une fonction très simple car il s'agit essentiellement de "transférer" les valeurs des attributs ayant un lien secondaire avec les notes vers la PACT accomplie (qui en effet n'est pas encore "accomplie" au vrai sens du terme!).

```

fonction commeUnePactAccomplie (p, ctx) : une PACT accomplie
  pactAcc <- nouvelleInstance(BassAccomplishedPact).
  pactAcc.lapse <- p.lapse
  pactAcc.context <- ctx
  Pour tout attribut a tel que (pactAcc.a.notesLink = secondary) faire
    pactAcc.a <- p.a.
  retourner pactAcc

```

Tableau 5.5.4 - Fonction `commeUnePactAccomplie`

Il y a plusieurs techniques de mesure de similarité entre deux cas : plus proches voisins, induction, *template retrieval*, etc. (Watson, 1996). Un des grands avantages de la technique de classification de type "plus proches voisins" est qu'elle permet l'utilisation d'une fonction de similarité explicite. Du coup, elle favorise l'introduction de connaissances du domaine (Aamodt & Plaza, 1994; Aha, Kliber & Albert, 1991). Dans les approches opposées, la mesure de similarité est étroitement liée à représentation des cas et l'organisation de la base. En effet, pour les cas dont l'indexation est basée sur des réseaux d'indexation (*common feature network*) (Kolodner, 1983), la mesure de similarité est implicite (Porter, 1988) : en parcourant le réseau on s'approche du cas le plus similaire²². Pour ces approches, il a été montré que l'introduction de connaissances du domaine dans l'étape de récupération des cas est très problématique dans des domaines plus complexes, notamment la conception "non-routinière" (Cunningham et al., 1993; Wolverton & Hayes-Roth, 1984).

²²Ce type d'organisation des cas dans la base est discuté en détail dans la section 6.5.1

Pour ces raisons, nous avons adopté la technique de classification de type “plus proche voisin”, qui d’ailleurs est la plus couramment employée (Watson, 1996). Dans ce type de mesure de similarité, on associe à chaque “caractéristique” (*feature*) qui décrit le cas un poids w représentant l’importance de la caractéristique. La similarité entre un cas cible C et un cas source S est calculée par la formule suivante (Kolodner, 1993) [p. 355] :

$$simil(C,S)=\frac{\sum_{i=1}^n w_i \times sim(f_i^C, f_i^S)}{\sum_{i=1}^n w_i} \quad (5.5.1)$$

où w_i est le poids associé à la caractéristique i , f_i^C et f_i^S sont respectivement les valeurs de la caractéristique i dans les cas cible et source, et sim est la fonction primitive de mesure de similarité entre deux caractéristiques

Dans notre programme, toutes les caractéristiques décrivant les cas correspondent à des attributs. Les entités impliquées dans des mesures de similarité, à savoir des instances de `Case`, `BassAccomplishedPact`, `LocalContext` et `ChordChunk`, ont toutes la facette `similarityFactor`, à laquelle nous associons les poids de chaque attribut (Cf. figure 5.2.2) . Etant données deux instances c_1 et c_2 d’un de ces quatre genres de classes, ayant n attributs (a_1, \dots, a_n), on peut exprimer la fonction de similarité donnée par Kolodner de la façon suivante :

$$similarité(c_1,c_2)=\frac{\sum_{i=1}^n (c_1.a_i.similarityFactor \times similarité(c_1.a_i,c_2.a_i))}{\sum_{i=1}^n c_1.a_i.similarityFactor} \quad (5.5.2)$$

Cette mesure est d’abord appliquée au cas lui-même, et successivement aux PACTs (antécédente et conséquente), aux contextes locaux et aux schémas d’accords. Il est intéressant de voir qu’à l’intérieur des PACTs accomplies, la fonction de similarité associée à chaque attribut ne rend pas uniquement des valeurs 0 ou 1 (sauf pour les attributs booléens, naturellement). Par exemple, la similarité des valeurs qualitatives `veryLow` et `high` est nulle, tandis qu’entre `veryLow` et `low` la similarité vaut 0.5. Une fonction de similarité particulièrement complexe concerne les schémas d’accords contenus dans la description de chaque contexte local. Par exemple, la similarité entre deux intervalles (dans `forwardResolution` et `backwardResolution`) tient vraiment compte de la “taille” des intervalles : une quarte juste est plus proche d’une tierce mineure que d’une septième majeure.

Il est important de rappeler que la similarité entre deux PACTs accomplies ne tient pas compte des notes du fragment mais uniquement de ses traits musicaux (Cf.

section 4.3.4.2). Autrement dit, les seuls attributs considérés dans l'équation 5.5.2 sont ceux qui ont un lien secondaire avec les notes (`notesLink = secondary`).

La méthode des plus proches voisins est que le réglage des poids des caractéristiques représentant les cas s'avère souvent difficile (Aamodt & Plaza, 1994). Actuellement, on commence à étudier des méthodes d'analyse des bases de cas qui proposent automatiquement des valeurs de poids pour les attributs à partir des études statistiques ou de l'emploi de techniques d'induction de type "arbre de décision" (Quinlan, 1986) pour identifier les caractéristiques les plus discriminantes (Wettschereck & Aha, 1995). Néanmoins, ces méthodes sont plutôt applicables dans les domaines où les attributs ont une "sémantique" assez floue. Pour les attributs de PACTs, nous connaissons la signification de chacun des attributs, car nous les avons introduits après une démarche importante d'explicitation de connaissances. Par conséquent, nous avons préféré affecter manuellement les poids et les raffiner après expérimentation (les figures 5.5.1, 5.5.2, 5.5.3 et 5.5.5 montrent les valeurs actuelles des poids). Par exemple, on sait que le fait de jouer des arpèges ou en *stepwise* est une caractéristique discriminante d'une ligne de basse, c'est pourquoi nous avons donné plus d'importance à l'attribut `lineStyle`. De même, le tempo (dans le contexte local) est très important car une ligne peut changer considérablement selon le tempo. Le poids associé à l'attribut `chorus` dans `ChordChunk` a été augmenté pour mettre en relief le fait que la façon de jouer pendant l'improvisation est différente de celle de l'exposition du thème.



6. VUE DETAILLEE DU PROCESSUS D'IMPROVISATION

Nous avons donné, dans le chapitre 4, une vision intuitive du fonctionnement de notre modèle d'agent jazzman et, dans le chapitre 5, nous avons mieux formalisé la notion des PACTs. Dans ce chapitre, nous continuons cette démarche de formalisation du modèle. Nous commencerons par présenter l'architecture de notre agent et la boucle principale du programme. Dans cette première partie, nous exposerons aussi la gestion du temps libre pour le raisonnement. Ensuite, chacune de trois phases du déroulement de l'improvisation (segmentation de la grille, sélection de PACTs et assemblage de PACTs) sera étudiée avec un accent sur les bases de règles impliquées. Enfin nous examinerons, en détail, comment la Mémoire Musicale s'organise et comment les fragments de ligne de basse qu'elle contient sont effectivement récupérés.

6.1. MUS_{ES} : UNE PLATE-FORME DE REPRESENTATION DE CONNAISSANCES EN MUSIQUE TONALE

Avant d'expliquer plus précisément le fonctionnement de notre modèle d'agent jazzman, il est important de signaler que nous réutilisons comme base de notre programme la plate-forme MUS_{ES} (Pachet, 1994; Pachet, Ramalho & Carrive, 1996), conçue pour représenter toutes les entités fondamentales de la théorie musicale. MUS_{ES} est une des seules plates-formes actuelles qui répondent aux exigences que nous avons posées à propos de la représentation de connaissances en musique tonale (Cf. Section 3.4.1). Par exemple, parmi les nombreuses classes implantées dans MUS_{ES}, on peut citer les suivantes :

- les *pitch-classes* (par ex. La, Sib, Fa###, etc.) qui mettent en œuvre toute l'algèbre non triviale des altérations et permettent de tenir compte des enharmoniques;
- les *intervalles* (par ex. tierce mineure, quarte augmentée, neuvième majeure) qui permettent entre autres le calcul de l'intervalle entre deux notes et des extrémités d'un intervalle étant donnée une note;
- les *accords* (par ex. Em7(b5), Fmaj7(aug11), etc.) qui peuvent représenter tout l'éventail d'accords utilisés en jazz et effectuer des opérations non triviales comme la détermination des tonalités possibles d'un accord;
- les *gammes* (par ex. Do# mineur harmonique, Ré dorien) offrant la possibilité de repérer facilement l'appartenance de notes et d'accords à une tonalité;
- les *notes*, les *mélodies* et d'autres objets temporels qui mettent en œuvre toute sorte d'opérations et comparaisons temporelles comme les primitives d'Allen (Allen, 1983).

6.2. ARCHITECTURE ET BOUCLES PRINCIPALES

La figure 6.2.1 montre l'architecture de notre modèle d'agent, conçue à partir de l'architecture générale d'agent rationnel (Cf. chapitre 1) selon les modifications et raffinements que nous avons introduits dans le chapitre 4. La fonction *jouer*, boucle principale du programme, détermine le comportement de notre agent (Cf. tableau 6.2.1) et l'articulation entre ses trois modules. Les deux entrées du programme sont la grille d'accords, à laquelle éventuellement on attache le thème, et le Scénario qui décrit les événements associés aux autres musiciens et au public, le Scénario étant donné facultativement. Avant d'entrer dans la boucle principale, c'est-à-dire avant que l'agent commence réellement à générer la ligne de basse, tous les composants (mémoires, Humeur et d'autres variables internes) sont initialisés (ligne 1) et la grille est complètement segmentée (ligne 2). Techniquement parlant, segmenter la grille au fur et à mesure que l'accompagnement avance ne pose pas de problèmes supplémentaires, comme nous verrons dans la prochaine section. Nous avons décidé de la segmenter une fois pour toutes simplement pour que l'agent dispose d'un peu plus de temps libre pour réfléchir pendant l'acte même de l'accompagnement.

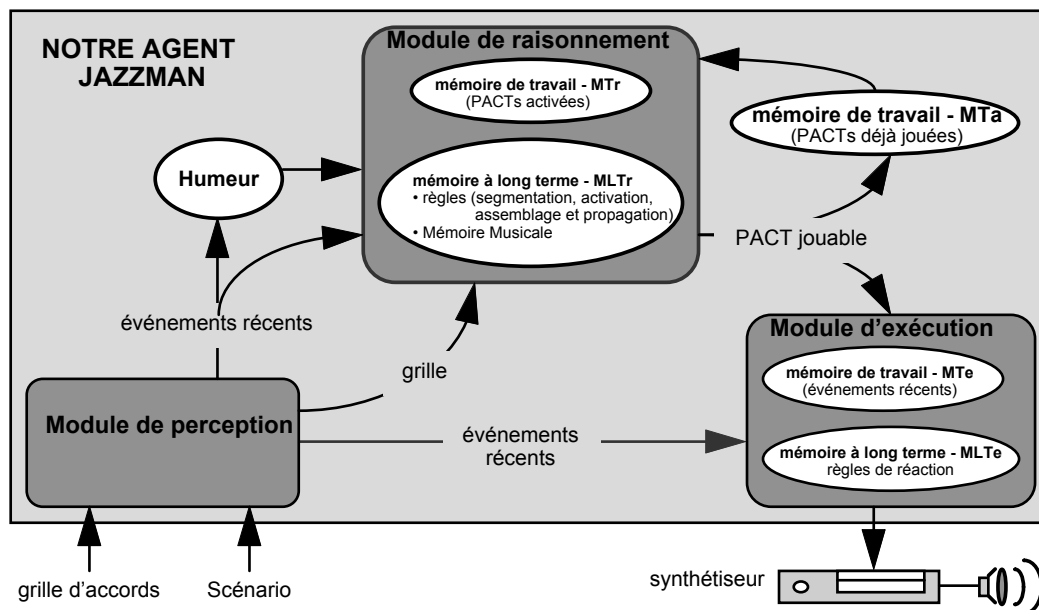


Figure 6.2.1 - Notre modèle d'agent jazzman

```

fonction jouer
1 initialiser.
2 grille <- lireGrille(ModuleDePerception)
3 schémas <- segmenterGrilleEnSchémasDAccords (grille).
4 Pour tout schéma s de schémas faire
5   Tant que raisonnementEstTrèsEnAvance (s) faire
6     attendre un peu.
7   evs <- lireEvénementsRécents ().
8   si raisonnementEstTrèsEnRetard (s)
9     alors pactJouable <- calculerBoutDImprovisationPressé (s) .
10    sinon pactJouable <- calculerBoutDImprovisation (s, evs, grille).
11    stocker pactJouable dans la mémoire de travail MTA
12    exécuter (pactJouable, ModuleDExécution)
    
```

Tableau 6.2.1 - Fonction jouer (boucle principale)

Une fois tous les schémas d'accords identifiés, on entre dans la boucle principale du programme (ligne 4), où il s'agit de calculer, pour chaque schéma donné, les notes à jouer. La première étape dans la boucle (ligne 5) consiste à vérifier si le raisonnement est trop en avance par rapport à l'exécution. Cela se fait en comparant la valeur courante de ERE (Cf. section 1.2.3) avec les seuils préfixés de la variable "urgence" de l'Humeur (Cf. tableau 6.2.2). Dans notre cas, le raisonnement est trop en avance avec un ERE plus grand que 3 mesures. Si l'avance est trop grande, l'agent attend, car plus l'avance est grande moins récents sont les événements du Scénario considérés (Cf. section 4.1.2)

```

fonction raisonnementEstTrèsEnAvance (s) : un booléen
  ERE <- (début (s) - positionCourante (ModuleDExécution) ).
  metreAJourLUrgence (ERE, Humeur)
  si Humeur.urgence = trèsEnAvance
    alors retourner vrai
    sinon retourner faux

```

*Tableau 6.2.2 - Fonction **raisonnementEstTrèsEnAvance***

Dans la deuxième étape (ligne 7), il est question de “s’emparer” des derniers événements disponibles du scénario. Comme nous l’avons déjà discuté (Cf. section 4.1.2), les seuls événements du scénario disponibles sont ceux qui commencent avant la position courante du module d’exécution. Parmi tous les événements disponibles, l’agent ne prend que les plus récents, c’est-à-dire ceux qui sont devenus disponibles depuis le dernier appel de la fonction `lireEvénementsRécents` (Cf. tableau 6.2.3).

```

fonction lireEvénementsRécents () : liste d’événements
  DernièrePosition <- moment de la dernière lecture du Scénario.
  evs <- élémentsEntre(DernièrePosition, positionCourante (ModuleDExécution) ,
    lireScénario (ModuleDePerception) ).
  DernièrePosition <- positionCourante (ModuleDExécution) .
  retourner evs

```

*Tableau 6.2.3 - Fonction **lireEvénementsRécents***

Dans la troisième étape, on vérifie si le raisonnement est trop en retard par rapport à l’exécution. C’est un calcul identique à celui de la fonction `raisonnementEstTrèsEnAvance`, sauf que dans ce cas l’objectif est de savoir s’il reste suffisamment de temps (ERE^*) pour le processus d’activation et d’assemblage des PACTs. Bien entendu, cela dépend de la machine sur laquelle le programme tourne. Pour un Macintosh LC 475, le raisonnement est très en retard quand $ERE^* < 1200$ ms. S’il reste suffisamment de temps, alors le raisonnement suit son cours normal (ligne 10), sinon il est “raccourci” (ligne 9). Le raccourcissement (Cf. tableau 6.2.4) se fait de la façon suivante : l’agent active une seule PACT et tout de suite, en court-circuitant l’assemblage, la rend jouable en récupérant un fragment de ligne basse de la Mémoire Musicale. La PACT activée est “jouer un cliché”, ce qui force la récupération d’un fragment mélodique récurrent (*lick*).

```

fonction calculerBoutDImprovisationPressé (s) : une PACT jouable
  pact <- nouvelleInstance(StandardPact).
  pact.classicness <- high.
  calculerLesNotes (pact, s, MTa, MémoireMusicale)
retourner pact.

```

Tableau 6.2.4 - Fonction *calculerBoutDImprovisationPressé*

Il existe des techniques de résolution de problèmes en temps réel (comme *approximative reasoning*, *anytime algorithms*, *procedural reasoning*, etc.) (Musliner et al., 1995; Strosdiner, 1994), qui pourraient être sans doute utilisées pour garantir que l'activation et l'assemblage des PACTs se fassent, au moins partiellement, pour des valeurs de ERE* encore plus petites. Cependant, comme l'a fait Pachet (Pachet, 1990), nous avons préféré "jouer" avec le manque de temps pour changer le mode de raisonnement de l'agent, en supposant que le manque de temps est un des facteurs responsables du fait que les jazzmen jouent souvent des clichés. Il est intéressant de signaler que jouer des "clichés" (licks) lorsque le programme est trop en retard s'insère dans une approche plus générale et très récente en planification appelée "stratégie de subterfuge" (Musliner, Durfee & Shin, 1993). Supposons qu'un robot doive prendre des pièces de différentes tailles dans un tapis roulant pour les mettre dans une boîte sur une table, tout en désactivant une alarme qui sonne de temps en temps (Hendler, Austin & Musliner, 1993). La tâche de planification consiste à optimiser l'espace libre de la boîte. Si l'alarme sonne quand le robot est en train de réfléchir sur l'endroit où mettre la pièce déjà saisie, alors il la met sur la table, il désactive l'alarme et il la reprend. Si, avant de la placer dans la boîte, une nouvelle pièce arrive par le tapis, alors, il pose l'ancienne pièce encore sur la table, prend la nouvelle pièce, la pose sur la table aussi, reprend l'ancienne pièce, la met dans la boîte et fait de même pour la nouvelle pièce. Le subterfuge du robot est de mettre une pièce sur la table, car pendant ce temps-là il peut s'occuper d'autres choses, tout en laissant "tourner en parallèle" le calcul de la meilleure position de la pièce dans la boîte. Jouer un cliché connu, qui ne demande pas d'efforts majeurs de raisonnement, est le "subterfuge" que l'agent a pour gagner un peu de temps et se remettre de nouveau à activer et à assembler des PACTs.

Le calcul habituel (non raccourci) des notes d'un schéma d'accord donné se fait de la manière suivante (Cf. tableau 6.2.5). D'abord, l'Humeur est mise à jour à partir des événements récents, car il peut influencer l'étape d'activation des nouvelles PACTs. Ensuite les PACTs sont activées et transférées vers la mémoire de travail du module de raisonnement. De plus, on sélectionne parmi toutes les PACTs récemment ou

précédemment activées, toutes celles qui superposent le segment (schéma) donné. Enfin, ces dernières PACTs sont assemblées jusqu'à générer une unique PACT jouable.

```
fonction calculerBoutDImprovisation (s, evs, grille) : une PACT jouable
mettreAJourLHumeur (evs, Humeur) .
nouvPacts <- activerDesNouvellesPacts (s, grille, MTa, Humeur) .
transférer nouvPacts vers la mémoire de travail MTr.
espaceDAssemblage <- sélectionnerPacts (s, MTr)
pactJouable <- assemblerPacts (s, espaceDAssemblage, MTa, MémoireMusicale) .
retourner pactJouable .
```

Tableau 6.2.5 - Fonction calculerBoutDImprovisation

Une fois la PACT jouable calculée, elle est stockée dans la mémoire de travail de l'agent (MTa) (ligne 11, tableau 6.2.1) et envoyée au module d'exécution (ligne 12) qui la joue, éventuellement avec des modifications de dernière heure, à l'aide du synthétiseur le moment venu.

6.3. SEGMENTATION DE LA GRILLE

La segmentation automatique de la grille se fait en quatre phases (Cf. tableau 6.3.1). Pour chacune des trois premières phases (identification des schémas, résolution des conflits et "remplissage des trous"), nous faisons appel à une base de règles car, comme nous le décrirons par la suite, il s'agit typiquement d'une activité qui peut être bien modélisée par des systèmes classiques de règles de production (Watterman & Hayes-Roth, 1978). Ces trois premières étapes déterminent les schémas (classe *ChordChunk*) en ce qui concerne leur placement dans le temps (*lapse*), leur forme (*shape*); leur structure rythmique (*rhythmicStructure*) et leur tonalité locale (*tonality*) . La dernière étape consiste simplement à compléter leur description, c'est-à-dire trouver les valeurs des attributs *backwardResolution*, *forwardResolution*, *position*, etc. (Cf. figure 5.5.4).

```
fonction segmenterGrilleEnSchémasDAccords (grille) : les schémas qui segmentent la grille
schémas <- identifierTousLesSchémasConnus (grille, MémoireMusicale) .
résoudreConflitsEntreSchémas (schémas)
remplirLesTrous (schémas, grille) .
compléterLaDescription (schémas, grille)
retourner schémas
```

Tableau 6.3.1 - Fonction segmenterGrilleEnSchémasDAccords

6.3.1. Une analyse harmonique particulière

L'identification de tels schémas requiert des connaissances d'analyse harmonique fonctionnelle de la grille (Cf. Section 3.4.1). Nous nous sommes basés sur l'approche du système d'analyse harmonique conçu autour de MusES (Mouton & Pachet, 1995), qui a été appliqué aux grilles d'accords de jazz avec succès. Cependant, nous avons dû concevoir des bases de règles originales, car la tâche de segmentation de la grille est différente de l'analyse harmonique fonctionnelle conventionnelle. En effet, du fait que nous nous intéressons uniquement à identifier les schémas d'accords pour qu'ils servent de support à la génération d'un segment de ligne de basse, trois différences se font remarquer.

Premièrement, nous n'avons pas le souci de trouver les tonalités de façon hiérarchique. Il nous suffit d'identifier les schémas pour pouvoir découper la grille de façon non équivoque. L'analyse traditionnelle pourrait nous être utile pour indiquer les sections (AABA, ABAB, etc.) et le style d'une grille (blues, standard, etc.) mais toute cette information est déjà donnée à l'agent avec la grille. Deuxièmement, outre l'identification la forme (II-V, II-V-I, etc.) du schéma et de sa tonalité locale, nous avons besoin de prendre en compte la structure rythmique des schémas (Cf. section 5.5.1). Par exemple, jouer sur un II-V, dont les accords durent chacun huit unités de temps, n'a rien à voir avec le cas où les mêmes accords durent chacun deux unités de temps, bien qu'il s'agisse de la même forme de schéma. Les patterns mélodiques qu'un jazzman se forge sont catalogués selon la forme mais aussi selon la structure rythmique du schéma (Baker, 1980; Coker, 1970; Owens, 1974). Troisièmement, nous avons voulu que l'identification de tels schémas soit faite non d'une façon "absolue" mais relative à un lexique donné (forme plus structure rythmique des schémas). Ce lexique correspond exactement à l'ensemble de schémas qui sont sous-jacents aux fragments mélodiques de la Mémoire Musicale, c'est-à-dire, tous les types de schémas attachés à l'attribut `context` des PACTs accomplies conséquentes. Cette dépendance avec la Mémoire Musicale est nécessaire pour garantir que l'agent ne segmente pas la grille dans des schémas pour lesquels il ne connaît pas de fragments mélodiques.

6.3.2. Détection des "chord chunks"

L'identification de tous les schémas du lexique qui apparaissent dans la grille se fait à travers une base de règles, dont la base de faits est constituée initialement de l'ensemble d'accords de la grille et de la Mémoire Musicale. Chaque règle déclenchée ajoute un nouveau schéma à la base de faits. À la fin, on récupère tous les schémas ajoutés. Nous associons à cette base un mécanisme de contrôle qui garantit qu'une règle n'est déclenchée qu'une fois. Ce contrôle est fait déclarativement à l'aide de "métarègles" (Watterman & Hayes-Roth, 1978). La règle *deuxCinqMajeur* du tableau 6.3.2 illustre comment un II-V majeur est détecté. Dans la première prémisse on s'assure que l'accord a_2 succède à a_1 . On vérifie ensuite que a_1 est mineur (puisque qu'un accord "demi-diminué" est aussi mineur, on s'assure que a_1 n'en est pas un) et que a_2 est dominant. On teste si l'intervalle entre les toniques de chaque accord est égal à une quarte. Enfin, on vérifie que le lexique contient un schéma de forme un II-V majeur et de structure rythmique les durées de chaque accord. Quand la règle est déclenchée, on crée une instance de schéma, en l'initialisant correctement avec la tonalité locale qui en découle (majeur avec le premier degré égal à la quarte de la tonique de l'accord a_2), et on l'ajoute à la base de faits.

<p>deuxCinqMajeur Pour tout accord a_1 et a_2, une Mémoire Musicale mm et une grille d'accord cg SI succède(a_2, a_1). mineur(a_1). \neg demiDiminué(a_1). dominant(a_2). intervalleEntreToniques(a_1, a_2) = quarte reconnaitSchéma($mm, MajorTwoFive, a_1, a_2$). ALORS $ns \leftarrow$ créerNouveauSchéma($cg, MajorTwoFive, durée(a_1), durée(a_2)$). $ns.tonality \leftarrow$ gammeMajeure(quarte(tonique(a_2))). Add ns.</p>
--

Tableau 6.3.2 - Règle *deuxCinqMajeur*

Les schémas reconnus par cette base de règles sont les suivants : II-V majeur, II-V (mineur), II-V-I majeur, II-V-I mineur, V-I majeur, V-I mineur, VI-II-V-I majeur (Anatole), IV-IV-III-VI-II-V-I majeur (Christophe), II-SubV-I majeur et II-SubV-I mineur. Ces schémas sont appelés *schémas composés*, pour souligner qu'ils contiennent au moins deux accords, en opposition à des schémas un peu particuliers (d'un seul accord) présentés dans la section 6.3.4.

Pour mieux illustrer le processus de segmentation, reprenons la grille de STELLA BY STARLIGHT, dont la segmentation finale a été montrée précédemment dans la figure 4.3.1. Tous les schémas détectés dans cette première phase sont indiqués dans la figure 6.3.1. Le lexique utilisé correspond à la Mémoire Musicale la plus complète que nous avons actuellement. Il est intéressant de signaler que, malgré la richesse de cette Mémoire Musicale (approximativement 270 cas), certains schémas en sont absents et du coup, ils ne sont pas identifiés dans la grille. Par exemple, ni le schéma “V-I mineur”, situé entre les mesures 10 et 11 (Ab7 Dmin7), ni le schéma “II-V majeur”, situé aux mesures 21 et 22 (Ebmin7 Ab7), ne sont reconnus. Il se trouve que le lexique contient des schémas “V-I mineur” et “II-V majeur”, mais pas avec la même structure rythmique, à savoir {2,4} et {6,2} respectivement. En effet, un V-I mineur (mesures 18-19) et des II-V majeurs (mesures 3-4, 5-6, 12-12, 14-14), ayant des structures rythmiques plus courantes, sont reconnus.

1	E min7(b5)	A 7	C min 7	F 7
5	F min 7	Bb 7	Eb maj7	Ab 7
9	Bb maj7	Emin7(b5) A7	D min7	G min7 C7
13	F maj7	G min7 C7	Amin7(b5)	D 7
17	G 7	G7	C min7	C min7
21	Eb min7	Eb min7 Ab 7	Bb maj7	Bb maj7
25	E min7(b5)	A 7	D min7(b5)	G 7
29	C min7(b5)	F 7	Bb maj7	Bb maj7

Figure 6.3.1 - STELLA BY STARLIGHT : identification des schémas d'un lexique donné

Il est utile de rappeler qu'à l'exemple de Mouton et Pachet, nous tenons compte de la circularité de la grille (Pachet, Ramalho & Carrive, 1996). S'il y a des schémas qui commencent dans un chorus et finissent dans un autre, ils sont reconnus. Par exemple, si on programme l'agent pour jouer le deuxième chorus de FEUILLES MORTES (Cf. figures 6.3.3 ou 6.3.4), un schéma VI-II-V-I (Gmin7 Cmin7 F7 Bbmaj7) est détecté entre la dernière mesure du premier chorus et la troisième du chorus suivant.

6.3.3. Résolution des conflits

La deuxième phase de la segmentation est la résolution de conflits entre les différents schémas identifiés qui se superposent. L'heuristique générale de cette étape est de couvrir le maximum possible de la grille, c'est-à-dire d'avoir un minimum d'accords "isolés", et de préférer les schémas les plus longs possibles.

La base de faits de la base de règles employée dans cette étape est composée des schémas d'accords (instances de `ChordChunk`) identifiés précédemment plus la grille d'accords elle-même. Au fur et à mesure que les règles se déclenchent, des schémas sont enlevés de la base de faits. Nous contrôlons le déclenchement des règles en les groupant en trois paquets et en ordonnant ces paquets. Dans le premier paquet, il y a des règles traitant des intersections assez particulières comme entre "E7 Amin7" (un V-I mineur) et "Amin7 Dmin7 G7 Cmaj7" (un VI-II-V-I) qui se chevauchent au niveau de l'accord "Amin7". Dans une telle situation, nous éliminons le "Amin7 Dmin7 G7 Cmaj7" bien qu'il soit un schéma long (4 accords), car nous préférons une segmentation "E7 Amin7" plus "Dmin7 G7 Cmaj7", plutôt que "E7" plus "Amin7 Dmin7 G7 Cmaj7" qui laisserait "E7" isolé. La règle `cinqUnMineurContreSixDeuxCinqUn` du tableau 6.3.3 résout justement le conflit entre un V-I mineur et un VI-II-V-I.

<p>cinqUnMineurContreSixDeuxCinqUn Pour tout schéma d'accord s_1 and s_2 SI $s_1.shape = \text{MinorFiveOne}.$ $s_2.shape = \text{SixTwoFiveOne}.$ $superpose(s_1, s_2)$ $dernier(s_1.chords) = premier(s_2.chords).$ ALORS Remove $s_2.$</p>
--

Tableau 6.3.3 - Règle `cinqUnMineurContreSixDeuxCinqUn`

Le deuxième paquet a une demi-douzaine de règles concernant des cas plus courants comme le conflit entre un II-V mineur et un V-I majeur, et vice versa. Nous préférons les II-V dans ces situations. Par exemple, le conflit entre le V-I mineur (Cmin7(b5) F7) des mesures 29 et 30 de *STELLA BY STARLIGHT* (Cf. figure 6.3.1) et le V-I majeur (F7 Bbmaj7) des mesures 30 et 31 est résolu par l'élimination V-I (F7 Bbmaj7) de la base de faits. Quant au dernier paquet, il ne contient que la règle *subsume* (Cf. tableau 6.3.4), qui d'ailleurs est la plus générale et la plus utilisée parmi toutes les règles de résolution de conflit.

<p>subsume Pour tout schéma d'accord s_1 and s_2 et une grille cg SI $s_1 \neq s_2$. contient (s_1, s_2). neSuperposeAucunAutreSchéma (cg, s_1). nestPasSuperposeParAucunAutreSchéma (cg, s_1). ALORS Remove s_2.</p>
--

Tableau 6.3.4 - Règle subsume

En reprenant la segmentation de STELLA BY STARLIGHT, la résolution des conflits est simple car la plupart des schémas conflictuels sont enlevés par la simple subsumption (Cf. figure 6.3.2). Les exceptions sont le conflit entre les schémas des mesures 29-30 et 30-31, dont nous avons déjà parlé, et celui entre les schémas des mesures 5-6 et 6-7, où nous préférons le II-V (Fmin7 Bb7). Ce dernier cas est un peu "tordu" car les deux schémas sont majeurs, mais de toutes les façons, l'un ou l'autre serait subsumé par le schéma des mesures 5 à 7. Nous pourrions très facilement éviter ce type de cas, mais cette base de règles est aussi utilisée pour la tâche d'acquisition de cas pour la Mémoire Musicale où nous avons besoin d'identifier *tous* les schémas possibles.

1	E min7(b5)	A 7	C min 7	F 7
5	F min 7	Bb 7	Eb maj7	Ab 7
9	Bb maj7	Emin7(b5) A7	D min7	G min7 C7
13	F maj7	G min7 C7	Amin7(b5)	D 7
17	G 7	G7	C min7	C min7
21	Eb min7	Eb min7 Ab 7	Bb maj7	Bb maj7
25	E min7(b5)	A 7	D min7(b5)	G 7
29	C min7(b5)	F 7	Bb maj7	Bb maj7

Figure 6.3.2 - STELLA BY STARLIGHT : les schémas après résolution des conflits (les "trous " sont signalés par des ellipses)

6.3.4. Remplissage des “trous”

La troisième phase de la segmentation est dédiée à ce que nous appelons le remplissage des zones correspondant à des accords qui n’ont pas pu être assimilés à un schéma composé du lexique.

Dans la première phase de la segmentation, le programme identifie des *vrais* schémas d’accords au sens où ils représentent des enchaînements typiques d’accords (les schémas composés). Néanmoins, un accord isolé peut être considéré comme une situation prototype et donc un type particulier de schéma appelé *schéma simple*. Des études (Kernfeld, 1983; Owens, 1974) montrent que les jazzmen réutilisent des fragments mélodiques associés à quelques catégories d’accords (isolés) comme les majeurs, les dominants, les diminués, etc. Tenant compte de cet élargissement du concept de schéma d’accord, le but du remplissage est de reconnaître chaque accord non inclus précédemment dans un schéma (composé) comme un schéma simple. La seule spécificité de ce processus est que nous voulons tenir compte de la structure rythmique du schéma, en l’occurrence de la durée de l’accord de schéma simple. Il se peut qu’un schéma simple de type “dominant” qui dure huit temps (par. ex l’accord G7 mesures 17 et 18 de STELLA BY STARLIGHT), ne soit pas présent dans le lexique. A ce moment-là, on découpe l’accord en deux accords identiques chacun durant quatre temps. On continue jusqu’à ce que le schéma simple soit reconnu.

<p>schémaValide Pour tout laps de temps <code>lp</code>, une grille <code>cg</code> et une Mémoire Musicale <code>mm</code> SI <code>schémaSimpleValide (mm, accordsDansLaps (cg, lp)) .</code> ALORS <code>ns <- créerSchémaSimple (cg, lp) .</code> Add ns . Remove lp.</p>

Tableau 6.3.5 - Règle *schémaValide*

La base de faits de la base de règles de remplissage des “trous” est composée des laps de temps (instances de `Lapse`) représentant les “trous”, de la grille et de la Mémoire Musicale. Au fur et à mesure que l’on assimile un schéma simple à un accord, le laps de temps correspondant est enlevé de la base de faits et le schéma reconnu y est ajouté comme le montre l’application de la règle *schémaValide* (Cf. tableau 6.3.5). En revanche, si l’accord n’est pas reconnu (Cf. règle *schémaNonValide*, tableau 6.3.6), le

laps de temps est découpé en deux et le test de validité est relancé. Dans STELLA BY STARLIGHT, il y a quatre “trous” laissés par la deuxième phase de segmentation (Cf. figure 6.3.2). Tous les schémas simples sont reconnus car cette segmentation est faite avec la Mémoire Musicale au complet.

Il n'est pas inutile d'ajouter que bien que nous ayons montré l'analyse de la grille entière, tout a été programmé de telle sorte qu'il est possible d'adresser des requêtes comme “quelles sont les schémas existants dans tel laps de temps de la grille?”, ce qui nous permettrait de segmenter la grille graduellement si nécessaire.

<p>schémaNonValide Pour tout laps de temps lp, une grille cg et une Mémoire Musicale mm SI \neg schémaSimpleValide (mm, accordsDansLaps (cg, lp)). divisible (cg, lp). ALORS $newLps \leftarrow$ divisionsDUnSchémaSimple (cg, lp). Add premier ($newLps$). Add deuxième ($newLps$). Remove lp.</p>

Tableau 6.3.6 - règle schémaNonValide

6.3.5. Influence et taille minimale de la Mémoire Musicale

Comme on peut le déduire de la dernière étape de la segmentation, le segment minimal est l'accord. Cela signifie donc qu'il faut que la Mémoire Musicale ait au moins un fragment mélodique joué sur un accord de chacune de cinq différentes catégories :

- majeur - Eb, Ebmaj7, Eb6, Ebmaj7(9), etc.;
- mineur - Fm, Fm7, Fm7(9), Fm7(9 11), etc.;
- dominant - Bb7, Bb7(9), Bb7(b9 b13), Bb 7 (#11), Bbsus4(7), Bb alt., etc.;
- demi-diminué - Fm7(b5);
- diminué - A°, A°(b13), etc.

Puisque la segmentation tient compte des structures rythmiques, il est nécessaire d'avoir des fragments de différentes durées, à savoir 1, 2, 3, 4 et, si possible, 8. Cela fait déjà une mémoire d'au moins 20 cas. Bien entendu, comme l'idée de base du raisonnement à partir de cas est la réutilisation, ce nombre peut être réduit si nous faisons appel à des mécanismes d'adaptation des cas. Certes, il serait facile d'adapter des passages joués sur un accord vers un autre lorsque la durée du fragment mélodique

est courte (un à deux temps) : on peut remplacer une quinte par une quinte diminuée ou une tierce mineure par une tierce majeure, etc. Mais ce n'est pas toujours si simple. Il suffit de penser à des passages plutôt chromatiques où le bassiste crée des tensions particulières. C'est pourquoi, nous avons choisi de ne pas faire ce type d'adaptation dans la version actuelle de notre programme.

Plus riche est la Mémoire Musicale, plus longs et diversifiés sont les schémas détectés. Les figures 6.3.3 et 6.3.4 montrent la segmentation de FEUILLES MORTES en utilisant, respectivement, la Mémoire Musicale la plus complète que nous ayons actuellement, et la Mémoire Musicale constituée seulement des fragments mélodiques joués en STELLA BY STARLIGHT. Remarquez que les schémas II-V-I mineurs (mesures 5-8, 13-16 et 17-20) sont tous coupés car il n'y en a pas dans STELLA BY STARLIGHT.

1	Cm7	F 7	Bb maj7	Eb maj7
5	Amin7(b5)	D 7	G min 7	G min 7
9	Cm7	F 7	Bb maj7	Eb maj7
13	Amin7(b5)	D 7	G min 7	G min 7
17	Amin7(b5)	D 7	G min 7	G min 7
21	Cm7	F 7	Bb maj7	Eb maj7
25	Amin7(b5)	D 7	Gmin7 C 7	Fmin7 Bb7
29	Eb maj7	Amin7(b5) Daug5 7	G min 7	G min 7

Figure 6.3.3 - Segmentation de FEUILLES MORTES : Mémoire Musicale plus complète

1	Cm7	F 7	Bb maj7	Eb maj7
5	Amin7(b5)	D 7	G min 7	G min 7
9	Cm7	F 7	Bb maj7	Eb maj7
13	Amin7(b5)	D 7	G min 7	G min 7
17	Amin7(b5)	D 7	G min 7	G min 7
21	Cm7	F 7	Bb maj7	Eb maj7
25	Amin7(b5)	D 7	Gmin7 C 7	Fmin7 Bb7
29	Eb maj7	Amin7(b5) Daug5 7	G min 7	G min 7

Figure 6.3.4 - Segmentation de FEUILLES MORTES basée sur les schémas de STELLA BY STARLIGHT

6.4 LA DETERMINATION DES PACTS A ETRE ASSEMBLEES

Dans cette section, nous exposerons le fonctionnement des bases de règles responsables du changement de l'Humeur et de l'activation et de l'assemblage des PACT, les étapes principales de la fonction `calculerBoutDImprovisation` (Cf. tableau 6.2.5). Nous montrons quels sont les faits filtrés par les règles, quel type d'action elles provoquent et quels sont les mécanismes de contrôle de leur déclenchement.

6.4.1. Mise à jour de l'Humeur

La première variable "urgence" de l'Humeur est automatiquement mise à jour en fonction de ERE dans toutes les occasions où il est nécessaire (fonctions `raisonnementEstTrèsEnRetard` et `raisonnementEstTrèsEnAvance`). Les deux autres variables de l'Humeur, à savoir "concentration" et "confiance" subissent des incréments et des décréments en fonction des événements récents du scénario. Ces changements sont mis en œuvre par une petite base de règles dont les faits sont de tels événements plus l'humeur elle-même. Chaque règle de cette base ne peut être déclenchée qu'une fois.

Les tableaux 6.4.1 et 6.4.2 montrent deux règles `policeArrive` et `publicApplaudit` qui provoquent, respectivement, un changement dans la valeur de "concentration" et de "confiance". Ce sont donc des règles très simples simulant des réactions à l'environnement qui est justement un des cadres typiques d'utilisations des règles de production (Laird, Newell & Rosebloom, 1987).

<p>policeArrive Pour tout événement <code>ev</code> de type <code>EvenPublic</code> et une humeur <code>h</code> SI auteur (<code>ev</code>) = <code>police</code>. action (<code>ev</code>) = <code>arriver</code>. ALORS diminuer (<code>h</code>, <code>concentration</code>)</p>

Tableau 6.4.1 - Règle `policeArrive`

<p>publicApplaudit Pour tout événement <code>ev</code> de type <code>EvenPublic</code> et une humeur <code>h</code> SI auteur (<code>ev</code>) = <code>public</code>. action (<code>ev</code>) = <code>applaudir</code>. ALORS augmenter (<code>h</code>, <code>confiance</code>)</p>

Tableau 6.4.2 - Règle `publicApplaudit`

6.4.2. Activation des nouvelles PACTs

Le mécanisme d'activation des PACTs est réalisé à l'aide d'une base de règles très semblable à celle de changement de l'Humeur (concentration et confiance). La seule grande différence est dans le nombre et la nature des faits "filtrés" par les prémisses des règles, l'activation des PACT se faisant à partir des données suivantes : le schéma courant, la grille (les accords, la forme, la structure, le thème associé), toutes les PACTs déjà jouées (MTa), l'Humeur et le tempo courant (Cf. tableau 6.4.3). Les PACT activées sont ajoutées à la base de faits. Voyons quelques exemples de règles d'activation de PACTs parmi la trentaine existante.

<p>fonction activerDesNouvellesPacts (<code>s</code>, <code>grille</code>, <code>MTa</code>, <code>Humeur</code>) : nouvelles PACTs activées lancerBase (<code>BaseDActivation</code>, <code>s</code>, <code>grille</code>, <code>MTa</code>, <code>Humeur</code>, <code>tempo</code> (<code>ModuleDExecution</code>)) <code>pacts</code> <- <code>recueillirPactsContenuesDansLaBaseDeFaits</code> (<code>BaseDActivation</code>). retourner <code>pacts</code></p>
--

Tableau 6.4.3 - Fonction `activerDesNouvellesPacts`

La règle `rythmePlusDenseAPendantLImprovisation` montre une activation de PACT à partir des PACTs déjà joués et de la structure du morceau (les chorus). Si l'agent est au début du premier chorus d'improvisation et si les deux dernières PACTs jouées ont le style rythmique basé sur la blanche ou sur la ronde, alors on active une PACT qui va durer toute l'improvisation et dont le "style rythmique" sera basé sur la noire.

rythmePlusDensePendantLImprovisation

Pour un schéma courant *s*, une grille *cg* et un ensemble de PACTs jouables *MTa*

SI

```
débutPremièreImprovisation (s, cg) .
dernièrePact (MTa) .rhythmicStyle • halfBased
dernièresNPactsOnLeMême (2, rhythmicStyle, MTa)
```

ALORS

```
pact <- nouvelleInstance (BassStandardPact)
pact.lapse <- dIciALaFinDeLImprovisation (s, cg)
pact.rhythmicStyle <- quarterBased
Add pact
```

Tableau 6.4.4 - Règle rythmePlusDenseAuDebutDImprovisation

La règle du tableau 6.4.5 montre une activation basée sur le schéma d'accord courant, où la structure de la grille joue aussi un rôle important. Il s'agit d'activer une PACT ayant des caractéristiques similaires (style rythmique, dissonance, contour mélodique, etc.) que celle jouée juste avant, si le schéma courant a la même forme que le schéma d'avant et si le chorus n'est pas d'improvisation. Ce type d'activation est intéressant dans la mesure où il vise à combler le manque de PACTs transformationnelles qui pourraient éventuellement être activées pour souligner une répétition de schéma d'accord dans la grille. Par exemple, si les accords courants sont "Dm7 G7" précédées par "Em7 A7" (deux II-V majeurs), la PACT transformationnelle "jouer la phrase d'avant transposée d'un ton" pourrait être activée.

presqueRépétitionSelonLaRépétitionDuSchéma

Pour un schéma courant *s*, une grille *cg* et un ensemble de PACTs jouables *MTa*

SI

```
schémaPrécédent (s, cg) .shape = s.shape.
chorusCourant (s, cg) = expositionThème
```

ALORS

```
p <- nouvelleInstance (BassStandardPact)
p.lapse <- s.lapse
transférerLesValeursDesPricipauxTraits (dernièrePact (MTa), pact) .
Add p
```

Tableau 6.4.5 - Règle presqueRépétitionSelonLaRépétitionDuSchéma

Les événements du Scénario servent eux aussi à activer des PACTs, dont certaines sont très peu jouables (par ex. règle intensitéBatteur) tandis que d'autres le sont davantage (par ex. ambianceChaude).

<p>intensitéBatteur Pour tout événement <i>ev</i> de type <i>EvBase</i> un schéma courant <i>s</i>, une grille <i>cg</i> et un ensemble de PACTs jouables <i>MTa</i></p> <p>SI auteur(<i>ev</i>) = batteur. type(<i>ev</i>) = intensité. valeur(<i>ev</i>) = ff. variation(<i>ev</i>) = aucune.</p> <p>ALORS <i>p</i> <- nouvelleInstance (BassStandardPact) <i>p.lapse</i> <- lapseDeDuréeIndéterminéeCommencantA (<i>s.début</i>) . <i>p.loudness</i> <- ff. Add <i>p</i></p>

Tableau 6.4.6 - Règle *intensitéBatteur*

<p>ambianceChaude Pour tout événement <i>ev</i> de type <i>EvenGlobal</i> un schéma courant <i>s</i>, une grille <i>cg</i> et un ensemble de PACTs jouables <i>MTa</i></p> <p>SI type(<i>ev</i>) = température. valeur(<i>ev</i>) = hot.</p> <p>ALORS <i>p</i> <- nouvelleInstance (BassStandardPact) <i>p.lapse</i> <- <i>s.lapse</i>. <i>p.loudness</i> <- ff <i>p.swing</i> <- high. <i>p.pullDown</i> <- true. <i>p.density</i> <- high. <i>p.syncopation</i> <- high. <i>p.tessitura</i> <- high. Add <i>p</i></p>

Tableau 6.4.7- Règle *ambianceChaude*

Techniquement parlant, la mise en place de cette base de règles ne pose pas de problèmes majeurs. Ainsi que pour la base concernant les changements de l’Humeur, le mécanisme de contrôle est extrêmement simple (les règles ne peuvent être déclenchées qu’une fois) car les modifications apportées par le déclenchement d’une règle n’ont pas d’implication pour le déclenchement des règles subséquentes. La grande difficulté est dans la construction même de cette base. Certes, il est plus difficile pour les musiciens de justifier leurs choix “note par note” qu’en termes plus abstraits comme “j’ai essayé de coller au batteur” (règle *intensitéBatteur*) ou “je voulais jouer plus de notes pour créer un contraste qui souligne le début du chorus de l’improvisation” (règle *rythmePlusDenseAuDebutDImprovisation*). Mais, il est évident que les règles que nous codons ne sont pas consensuelles, ni complètes, ni toujours applicables. Tout d’abord, compte tenu de la démarche d’apprentissage du jazz, il est tout à fait normal que chaque musicien ait ses “règles” que l’on pourrait aussi appeler stratégies, astuces, habitudes, etc. Ensuite, on ne peut songer à avoir *toutes* les “règles” utilisées par la

plupart les musiciens. En plus de demander un travail d'acquisition énorme, l'identification d'un répertoire "complet" de ces "règles" ne peut être garantie, car l'expert n'a pas forcément l'introspection nécessaire pour faire ressortir toutes les connaissances auxquelles il fait appel lorsqu'il joue. Enfin, bien que la plupart des musiciens que nous avons rencontrés reconnaît la "plausibilité musicale" des règles d'activation de PACTs que nous avons introduites, ils avouent ne pas les utiliser toujours dans les mêmes conditions.

En réalité, ces difficultés d'acquisition de connaissances sont classiques. Dans les méthodes d'acquisition conçues pour les systèmes experts de deuxième génération, le paradigme du "transfert" des connaissances de l'expert vers la machine a été remplacé par "l'accord de l'expert sur le modèle que le cognitif est en train d'élaborer" (Le Roux, 1994) [section 1.3]. Pour nous, ce qui est le plus important n'est pas tant d'exprimer toutes les règles et ce de façon tout à fait plausible. Le plus important, pour nous, est d'avoir conçu un langage avec lequel il est possible d'exprimer ces règles.

6.4.3. Sélection des PACTs à être assemblées

Une fois que les PACTs récemment activées ont été envoyées à la mémoire de travail du module de raisonnement (MTr), l'agent fait la sélection des PACTs concernant le schéma courant (Cf. tableau 6.4.8). D'abord, chaque PACT à découper est examinée et (Cf. fonction `discrétiser`, section 5.4) celles qui superposent le segment défini par le schéma d'accord courant sont discrétisées. La discrétisation génère des PACTs standards qui seront incluses dans la mémoire. De façon similaire, l'agent examine chaque PACT standard pour la segmenter (Cf. fonction `segmenter`, section 5.2.3), en faisant une copie avant pour garder intacte la PACT originelle, qui peut être impliquée dans des segments d'improvisation futurs. Au passage, on en profite pour enlever toutes les PACTs (à découper et standards) qui ont déjà "expiré", c'est-à-dire dont la durée finit avant le début du schéma courant.

```

fonction sélectionnerPacts (s, MTr) : PACTs contenant l'espace l'état initial de l'espace d'assemblage
Pour toute PACT à découper tbdp dans MTr faire
  si superposeEntièrement(tbdp.lapse, s.lapse)
    alors standardPact <- discrétiser (tbdp, s.lapse)
    ajouter (standardPact, MTr).
  sinon si (tbdp.lapse.end • s.lapse.begin)
    alors enlever (tbdp, MTr).
espaceDAssemblage <- vide.
Pour toute PACT standard p dans MTr faire
  si superposeEntièrement(p.lapse, s.lapse)
    alors pChoisie <- segmenter (copie (p), s.lapse)
    ajouter (pChoisie, espaceDAssemblage).
  sinon si (p.lapse.end • s.lapse.begin)
    alors enlever (p, MTr).
retourner espaceDAssemblage

```

Tableau 6.4.8 - Fonction sélectionnerPacts

Il est important de souligner que nous testons, pour les deux genres de PACTs composant MTr, une superposition particulière, selon laquelle les PACTs qui “superposent partiellement” le segment défini par le schéma courant ne sont pas sélectionnés. Ainsi, les PACTs N et K de la figure 6.4.1 ne seraient sélectionnés.

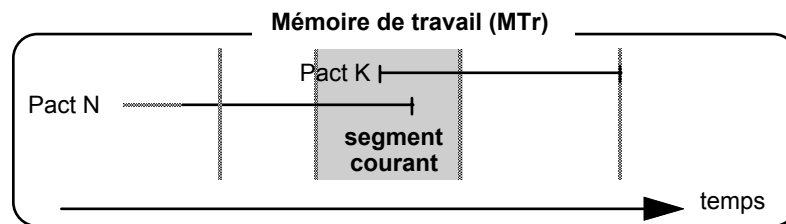


Figure 6.4.1- PACTs ne pouvant être sélectionnés

6.5. ASSEMBLAGE DES PACTS

L'assemblage des PACTs sélectionnés est fait à l'aide de la fonction `assemblerPacts` (Cf. tableau 6.5.1). D'abord, l'exécution de la base de règles d'assemblage est lancée en ayant ces PACTs comme les faits à être filtrés. Si l'assemblage échoue, c'est-à-dire si une PACT jouable ne ressort pas de ce processus, alors on “provoque” une propagation visant à calculer les notes de la PACT par un appel à la Mémoire Musicale. Chaque règle déclenchée ajoute et/ou enlève des PACTs dans la base de faits. C'est pourquoi, contrairement aux bases des règles présentées dans la dernière section, le déclenchement d'une règle peut rendre d'autres règles déclenchables ou pas.

```

fonction assemblerPacts (s, espaceDAssemblage, MTa, MémoireMusicale) : une pact jouable
  lancerBase (BaseDAssemblage, espaceDAssemblage) .
  pact <- pactRésultante (BaseDAssemblage) .
  si jouable (pact)
    alors retourner pact .
    sinon calculerLesNotes (pact, s, MTa, MémoireMusicale)
      retourner pact .

```

Tableau 6.5.1 - Fonction assemblerPacts

La base de règles d'assemblage met en œuvre les trois opérateurs d'assemblage (enlève, combine et propage) à travers six règles organisées dans un ordre qui reflète leur priorité de déclenchement. La première règle détecte simplement que le but a été atteint (règle `pactJouable`). S'il y a une PACT jouable parmi toutes les PACTs composant la base de faits, alors on force l'arrêt du moteur d'inférence même indépendamment du fait qu'il y ait d'autres règles déclenchables.

```

pactJouable
Pour toute PACT standard p.
SI
  jouable (p)
ALORS
  PactRésultante <- p.
  CritèreD'ArrêtForcé <- vrai

```

Tableau 6.5.2 - Règle pactJouable

La deuxième et la troisième règle gèrent l'application de l'opérateur "enlève" lorsque deux PACTs ne peuvent être combinées. Dans la première de ces règles (Cf. tableau 6.5.3) on examine deux PACTs ayant le même ensemble d'attributs spécialisés. Il est très important de donner la priorité à telle règle pour pouvoir favoriser les PACTs qui ont été activées comme une réaction à l'environnement. Ainsi, parmi deux PACTs $p_1 = [\text{loudness} = \text{fff}, \text{creationDate} = 10]$ et $p_2 = [\text{loudness} = \text{p}, \text{creationDate} = 182]$, on préfère p_2 car elle est plus récente (Cf. section 5.2.5). Dans la deuxième de ces règles, on teste simplement le cas plus général de la non-combinabilité. Il est toujours préférable d'éliminer des PACTs non-combinables avant de commencer à combiner celles qui le sont. En effet, l'application prioritaire de l'opérateur "enlève" minimise l'effet de l'ordre de combinaison des PACTs, évitant des "conflits" futurs entre les PACTs (Cf. section 5.2.6).

<p>courvertureDirecteTotaleMutuelle Pour toute PACT standard p_1 et p_2 . SI $p_1 \neq p_2$ $\text{attributsSpécialisés}(p_1) = \text{attributsSpécialisés}(p_2)$. $\text{pactPréféréeEntre}(p_1, p_2) = p_1$ ALORS Remove p_2</p>
--

Tableau 6.5.3 - Règle courvertureDirecteTotaleMutuelle

La quatrième et la cinquième règle s’appliquent à des PACTs combinables partiellement ou totalement. Le tableau 6.5.4 montre celle qui s’occupe de la combinaison partielle. La seule différence avec la règle de combinaison totale est qu’il n’y est pas nécessaire de tester la préférence entre les PACTs.

<p>combinabilitéPartielle Pour toute pact standard p_1 et p_2 . SI $p_1 \neq p_2$ $\text{combinabilité}(p_1, p_2) = \text{partielle}$ $\text{pactPréféréeEntre}(p_1, p_2) = p_1$ ALORS $p_{1+2} \leftarrow \text{combiner}(p_1, p_2)$. propager (p_{1+2}) Remove p_1 Remove p_2 Add p_{1+2}</p>
--

Tableau 6.5.4- Règle combinabilitéPartielle

Il est intéressant de souligner que nous n’avons pas mis en œuvre l’opérateur “propager” de façon explicite. Au lieu d’avoir un prédicat pour tester si une PACT contient des informations à propager et, le cas échéant, les propager, nous avons opté pour appliquer l’opérateur “propage” à chaque fois qu’une combinaison se produit pour une question d’efficacité (Cf. tableau 6.5.4). Comme nous contrôlons l’activation de PACTs, nous savons que toutes les PACTs composant l’état initial de l’espace d’assemblage ne peuvent devenir “propageables” qu’après une combinaison. En outre, les tests concernant les conditions de déclenchement des propagations sont déjà inclus dans la façon dont nous mettons en œuvre ces propagations elles-mêmes (Cf. section 5.2.8).

La dernière règle (*dernièrePact*) dit simplement que, s’il n’y a qu’une PACT dans la base de faits et elle n’est toujours pas jouable, on arrête l’assemblage. En réalité, il n’est pas possible d’exprimer la notion de “dernière PACT” dans un langage de premier ordre, car cela exigerait une quantification de quantificateur. Pour pallier ce

problème nous employons un mécanisme de contrôle qui tient compte de l'ordre des règles. Autrement dit, bien que la règle `dernièrePact` soit toujours déclenchable, elle ne l'est que lorsqu'il n'y a aucune autre règle déclenchable.

dernièrePact Pour toute pact standard p. SI \neg jouable (p) ALORS PactRésultante <- p. CritèreDArretForcé <- vrai

Tableau 6.5.5- Règle dernièrePact

6.6. UTILISATION DE LA MEMOIRE MUSICALE

L'algorithme général de récupération et de réutilisation d'un cas pour un problème P est donné dans le tableau 6.6.1. Dans cette section, nous présenterons précisément chacune de ces étapes en ce qui concerne la récupération et la réutilisation des fragments de lignes de basse stockés dans la Mémoire Musicale de notre agent jazzman.

-
- ➔☞☒ Décire le problème donné P en tant qu'un cas cible CC;
 - ➔☞☒ Chercher les N premiers cas les plus similaires au cas CC, parmi les cas (sources) potentiellement pertinents de la base;
 - ➔✓☒ Choisir parmi les N cas pertinents identifiés, le meilleur cas MC vis-à-vis de CC;
 - ➔✓☒ Adapter MC en fonction de CC;
 - ➔✗☒ Retourner MC adapté.
-

Tableau 6.6.1 - L'algorithme général de récupération et de réutilisation d'un cas (adapté de (Kolodner, 1993) [p. 287])

6.6.1. Organisation de la Mémoire Musicale

La récupération d'un cas d'une base de cas dépend de l'organisation de la mémoire de cas, c'est-à-dire de la façon dont les cas sont disposés. Dans le raisonnement à partir de cas il y a deux des types classiques d'organisation (Kolodner, 1993) [chap. 8]. Le premier type est la *mémoire plate*, où les cas sont rangés dans une liste. L'algorithme de recherche des cas les plus similaires se fait en comparant de façon séquentielle le cas cible donné à chaque cas source de ladite liste. Ce type d'organisation est normalement

employé lorsque l'on fait appel à une technique de mesure de similarité selon les plus proches voisins (Cf. section 5.5.3). Le deuxième type est la *mémoire hiérarchisée*, où les cas sont stockés selon les nœuds d'un réseau orienté sans cycle, appelé *réseau de discrimination*. Chacun des nœuds correspond à une partition de mémoire, car chaque nœud "regroupe" tous les cas qui sont caractérisés par le même ensemble de propriétés. La recherche des cas les plus similaires consiste à choisir et suivre le fils le plus similaire au niveau de chaque nœud jusqu'à ce que l'on arrive aux feuilles. Il existe différentes techniques de construction du réseau d'indexation d'une mémoire hiérarchique, la plus courante étant *le réseau de caractéristiques communes* (Plaza & Lopez de Mantaras, 1990; Richter & Weiss, 1991; Veloso, 1994). La figure 6.6.1 montre comment, dans le système MEDIATOR (Kolodner & Simpson, 1989), on représente cinq cas (Korea, Panama, Orange1, Orange2 et Candy) en assemblant dans les nœuds les caractéristiques (descriptions) communs aux cas.

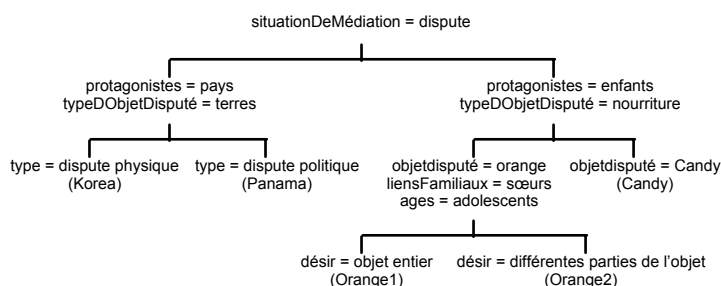


Figure 6.6.1 - Exemple d'indexation de cinq cas de disputes

Ces deux types de mémoires ont chacun ses avantages et inconvénients. Dans les mémoires plates, on est sûr de trouver le cas le plus similaire, mais le temps requis pour la récupération d'un cas grandit de façon linéaire selon le nombre de cas (Aha, Kliber & Albert, 1991). La mémoire hiérarchisée, à son tour, permet un accès très rapide aux cas, mais elle ne garantit pas que l'on trouve le meilleur cas. Ceci résulte du fait que le réseau d'indexation fixe un ordre statique de tests, les descriptions "plus importantes" correspondant aux premiers nœuds visités du réseau. Si la requête (le cas cible) n'est pas complète ou si l'importance des descriptions change d'une requête à l'autre, la récupération du meilleur cas échoue vraisemblablement (Kolodner, 1993) [pp. 303-9]. La technique de *réseaux de discrimination redondants*, un couplage de plusieurs réseaux d'indexation, chacun imposant un ordre différent de tests, tente de résoudre ce problème. Néanmoins, la redondance introduite entraîne un surcoût du temps de

récupération des cas (Kolodner, 1993) [ibidem]. La combinaison des deux techniques (réseau de caractéristiques communes et réseaux de discrimination redondants) préconisée par Kolodner et Schank dans ce qu'ils appellent la "Mémoire Dynamique" (Kolodner, 1983; Schank, 1982), bien qu'elle minimise ce problème, ne le résout pas complètement (Maurice-Demourieux, Lâasari & Levallet, 1993). Outre les avantages et inconvénients liés au *trade-off* "efficacité contre précision", central dans la récupération des cas, le problème de l'introduction de connaissances dans l'évaluation de la similarité entre deux cas du domaine se pose. A cet égard, comme nous avons déjà discuté dans la section 5.5.3, les mémoires hiérarchisées sont moins adéquates que les mémoires plates (Porter, 1988). De fait, dans les mémoires plates, on fait explicitement appel à une fonction de mesure de similarité explicite comme celle de plus proche voisins, tandis que dans les mémoires hiérarchisées, la similarité est implicitement définie à travers le parcours du réseau d'indexation.

Revenant à la Mémoire Musicale, son organisation est du genre mémoire plate, et ce pour deux raisons. Premièrement, parce que l'introduction des connaissances du domaine dans le calcul de similarité se fait plus aisément. Deuxièmement, parce que l'on ne sait jamais quels attributs de la PACT en voie assemblage (qui va composer le cas cible) sont spécialisés. La requête que l'on pose à la Mémoire Musicale est donc très souvent incomplète et très variable dans son contenu.

La récupération d'un cas se fait normalement en deux étapes : appariement partiel (*partial matching*) et ordonnancement (*ranking*) (Kolodner, 1993) [chap. 9]. Dans la première étape (ligne 2 de l'algorithme du tableau 6.6.1), on vise à identifier les cas qui sont potentiellement intéressants vis-à-vis du problème afin de restreindre davantage la recherche dans la base de cas. Pour cette étape, on fait appel à des critères simples qui peuvent, de manière efficace, être testés sur un grand nombre de cas de la mémoire. Dans la deuxième étape (ligne 3 de l'algorithme du tableau 6.6.1), on emploie des critères supplémentaires et plus raffinés pour donner un ordre de priorité (*ranking*) aux cas choisis dans la première étape. Dans notre programme, les cas sont déjà appariés, au vrai sens du terme, car nous utilisons une représentation de type "attribut-valeur". Néanmoins, nous avons la possibilité de réaliser un premier filtrage des cas pertinents. En effet, notre programme cherche à récupérer un fragment de ligne de basse jouée sur des schémas ayant la même forme (*shape*) et la même structure rythmique

(*rhythmicStructure*) du schéma (cible) donné. Comme nous avons discuté auparavant (Cf. section 6.3.5), nous ne faisons pas d'adaptation des fragments par rapport à la forme et la structure rythmique du schéma. La figure 6.6.2 montre comment nous organisons la Mémoire Musicale. Elle n'est donc pas tout à fait plate, car un petit arbre d'indexation groupe les cas selon les descriptions de la forme et de la structure rythmique du schéma sous-jacent à la PACT *conséquente* de chaque cas. Dans chaque partition, les cas sont rangés dans une liste simple.

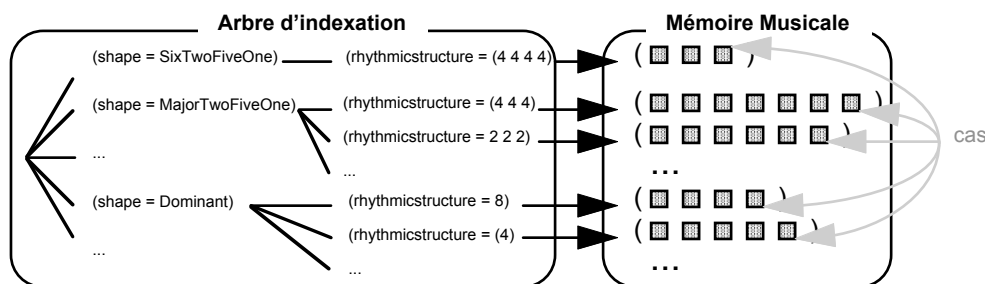


Figure 6.6.2 - Organisation de la Mémoire Musicale

Pour terminer cette section, il est souhaitable de rappeler qu'il existe des techniques visant à rendre plus efficace l'accès aux mémoires plates, soit en utilisant des algorithmes de recherche parallèle (Kolodner, 1988) ou les *dynamic template trees* (DTT) (Grolimund & Ganascia, 1996). Ces techniques permettent de rendre hiérarchique une mémoire, tout en gardant une fonction de mesure de similarité explicite. Quoique cette dernière technique (DTT) soit tout à fait adaptée à notre problème (la "parallélisation" proposée étant conçue pour une machine en particulier), nous ne l'avons pas encore employée, car le nombre de cas de chaque catégorie de schéma (forme plus structure rythmique) ne dépasse pas, pour l'instant, la trentaine. Lorsque des nouveaux cas seront introduits dans la Mémoire Musicale, une organisation de type DTT sera nécessaire.

6.6.2. Récupération et réutilisation des cas

Maintenant que l'organisation de la Mémoire Musicale a été présentée, nous pouvons exposer comment la récupération du cas source le plus convenable est faite. La fonction `calculerLesNotes` montre "l'adaptation" de l'algorithme général du tableau 6.6.1 à notre programme. Pour notre agent, il s'agit de récupérer une PACT jouable de la Mémoire Musicale et d'adapter le fragment mélodique associé vis-à-vis du problème

représenté par une PACT standard p en voie d'assemblage plus la description du schéma d'accord courant s .

```

fonction calculerLesNotes (p, s, MTa, MémoireMusicale)
1 cible <- casCibleAPartirDe (p, s, MTa) .
2 listeDeCas <- lesCasLesPlusSimilaires (cible, NombreMax,
                                     lesCasPertinents (cible, MémoireMusicale))
3 casChoisi <- leMeilleurCas (cible, listeDeCas).
4 T <- ensemble de transformations identifiées pour casChoisi .
5 adapter (casChoisi.consequentPact, T)
6 transférerInformations (casChoisi.consequentPact, p)
7 retourner p

```

Tableau 6.6.2 - Fonction *calculerLesNotes*

Le premier pas de l'algorithme (ligne 1, tableau 6.6.2) consiste à décrire la PACT donnée en tant que cible. Ceci est fait par la fonction *casCibleAPartirDe* (Cf. tableau 6.6.3). La PACT antécédente du cas cible est obtenue à partir de la mémoire de travail de l'agent MTa , tandis que la PACT conséquente est définie, à l'aide de la fonction *commeUnePactAccomplie* (Cf. tableau 5.5.2), à partir de la PACT standard p donnée plus la description du contexte correspondant au schéma d'accord courant s .

```

fonction casCibleAPartirDe (p, s, MTa) : un cas cible
  cas <- nouvelleInstance (Case) .
  cas.antecedentPact <- dernierElement (MTa)
  cas.consequentPact <- commeUnePactAccomplie (p, contextLocal (s))
retourner cas

```

Tableau 6.6.3 - Fonction *casCibleAPartirDe*

Le deuxième pas de l'algorithme (ligne 2, tableau 6.6.2) concerne la récupération d'un certain nombre de cas potentiellement pertinents vis-à-vis du cas cible donné. Dans notre programme, les cas potentiellement pertinents sont trouvés à travers un simple parcours (fonction *lesCasPertinents*) de l'arbre d'indexation selon les valeurs de forme et de structure rythmique du schéma d'accord du cas cible. Une fois les cas source trouvés, nous choisissons les n ($n = \text{NombreMax}$) les plus similaires au cas cible (fonction *lesCasLesPlusSimilaires*).

Le troisième pas de l'algorithme de récupération et de réutilisation des cas (ligne 3, tableau 6.6.2) comporte le choix du meilleur cas. L'hypothèse centrale de travail du raisonnement à partir de cas est qu'en trouvant le cas *le plus similaire* à un problème donné, la solution associée au cas récupéré est la plus convenable. Toutefois, des recherches récentes menées par Smyth et Keane (Smyth & Keane, 1994; Smyth & Keane,

1995a) montrent qu'en ce qui concerne les tâches de conception, la similarité n'est pas le bon critère pour la récupération des cas. Il se trouve qu'un cas peut être le plus similaire à un problème donné (cas cible) mais que le coût d'adaptation de la solution récupérée soit trop grand. Supposons que l'on veuille concevoir un véhicule pour travailler à l'intérieur des mines. On peut, par exemple, récupérer de la base de cas un projet de véhicule satisfaisant aux nombreuses propriétés affichées dans le cas cible (charge maximale, nombre de vitesses, système de phares, type de siège, capacité de se déplacer sur des rails, etc.) sauf à la taille qui est trop grande. Le problème est qu'une diminution de la taille implique une diminution de la place du moteur, qui implique un changement du moteur, faute de quoi la charge maximale ne serait pas respectée. Le coût d'adaptation vers une taille plus petite est, par exemple, plus grand que celui de l'adaptation de système de phares, du type de sièges ou du nombre de vitesses.

Dans notre cadre d'application, le problème se pose également. Nous pouvons retrouver un fragment mélodique qui a les propriétés requises (en ce qui concerne, par exemple, la consonance, la densité, le style de ligne, etc.) mais qui nécessite une transposition, parce que les accords sous-jacents (la tonalité) ne sont pas les mêmes que ceux du cas cible. Il se peut qu'une fois la transposition réalisée, des notes tombent hors de la tessiture de la basse. L'opération qui consiste à remettre ces notes dans la tessiture est compliquée et peut remettre en cause les propriétés (dissonance, style de la ligne, etc.) pour lesquelles le cas source a été récupéré.

Pour résoudre ce problème, Smyth et Keane ont proposé une stratégie de récupération basée sur l'adaptabilité des cas vis-à-vis du problème donné (*adaptation-guided retrieval*). Dans une première étape, ils choisissent tous les cas (source) qui peuvent être adaptés au problème donné à travers un ensemble ordonné de transformations. Le meilleur cas est celui dont la somme des coûts associés à chaque transformation est la plus petite.

Quant à nous, nous n'avons pu adopter que partiellement la méthode préconisée par Smyth et Keane parce que le problème auquel nous nous attaquons est, à deux égards, différent de ceux traités par ces deux chercheurs. Premièrement, nous ne savons pas faire toutes les adaptations. Dans l'application étudiée par ces deux chercheurs, le contrôle de certains types de véhicules autonomes (*coil cars*) dans un environnement industriel, l'adaptation des cas pour l'ensemble des tâches données n'a pas posé de

problèmes particuliers. Dans notre application, certaines adaptations sont simples à mettre en œuvre, comme le déplacement dans le temps, le changement d'intensité, la transposition en hauteur, etc. Toutefois, plusieurs adaptations, comme le changement du contour mélodique, du style de la ligne, du style rythmique, en plus d'être difficiles à réaliser, changent de manière tellement profonde le caractère du fragment mélodique récupéré, qu'il est préférable de chercher un autre fragment dans la base. En outre, certaines transformations peuvent "annuler" l'effet d'autres si elles ne sont pas appliquées dans le bon ordre. Or, comme nous avons eu déjà l'opportunité d'en discuter (Cf. section 5.3), l'ordre avec lequel des transformations peuvent être appliquées à un fragment mélodique n'est pas facile à déterminer. Pour nous, il n'y a donc pas de sens à récupérer des cas en se basant uniquement sur leur adaptabilité, car nous ne sommes pas capables de les adapter. Deuxièmement, contrairement à certains problèmes de conception, nous pouvons nous permettre de ne pas adapter totalement les solutions récupérées. Ainsi, jouer des arpèges quand l'agent voulait jouer du *stepwise* est moins grave qu'utiliser un câble en acier de mauvaise épaisseur dans la conception d'un ascenseur. Les contraintes imposées sur l'improvisation ou l'accompagnement sont effectivement moins restrictives.

Nous avons alors défini la notion de réutilisabilité d'un cas source cs par rapport à un cas cible cc de la façon suivante:

$$\text{réutilisabilité}(cs,cc) = \frac{(k_s \times \text{similarité}(cs,cc)) + (k_a \times \text{adaptabilité}(cs,cc))}{k_s + k_a} \quad (6.6.1)$$

où, k_s et k_a sont, respectivement, appelés coefficient de similarité et coefficient d'adaptabilité.

La réutilisabilité étant définie, le meilleur cas (ligne 3, tableau 6.6.2) est celui qui a la plus grande adaptabilité vis-à-vis du cas cible.

Le calcul de l'adaptabilité d'un cas source particulier vis-à-vis d'un problème donné est illustré dans le tableau 6.6.4. Tout commence par la détermination de l'ensemble des transformations devant être appliquées à la PACT conséquente du cas pour qu'elle soit en conformité avec la PACT conséquente du cas cible. Ensuite, on calcule l'adaptabilité globale du cas en fonction de l'adaptabilité de chaque transformation.

```

fonction adaptabilité(source,cible): une valeur numérique entre 0 et 1
T <- transformationsDAdaptation (source.consequentPact,cible.consequentPact) .
adaptGlobale <- 0.
Pour chaque transformation t dans T faire
  adaptGlobale<- adaptGlobale + aptabilitéDeLaTransformation(t) .
retourner adaptGlobale / taille(T)

```

Tableau 6.6.4 - Fonction adaptabilité

L'ensemble des transformations d'adaptation est déterminé à l'aide de la fonction `transformationsDAdaptation` ci-dessous. On vérifie que chaque attribut spécialisé de la PACT (conséquence) cible a la même valeur chez la PACT (conséquence) source. Lorsque cela n'est pas vrai, on détermine la transformation qu'il est nécessaire d'appliquer sur les notes de la PACT source (attribut `melodicFragment`). Par exemple, si la valeur de l'attribut `density` est `high` dans PACT cible tandis qu'elle est `low` dans la PACT source, alors une transformation consistant à ajouter des notes dans le fragment mélodique de la PACT source devient nécessaire. L'ordonnement toujours nécessaire des transformations identifiées ne pose pas de problèmes car les transformations que nous traitons sont peu nombreuses et assez orthogonales. Les transformations réalisées actuellement sont : la transposition en hauteur, le changement d'enveloppe d'amplitudes et le déplacement dans le temps. D'autres transformations, comme le changement de densité, de dissonance et d'inversion sont envisageables mais n'ont pas été mises en œuvre dans la version actuelle de notre programme.

```

fonction transformationsDAdaptation(pactSource,pactCible): transformations
T <- vide.
Pour tout attribut a spécialisé de pactCible faire
  Si pactCible.a • pactSource.a
    alors t <- calculerTransformation(a,pactCible,pactSource)
    ajouter t dans T.
ordonnerTransformations(T) .
retourner T

```

Tableau 6.6.5 - Fonction transformationsDAdaptation

Ce qu'il est important de souligner est que chaque transformation déterminée correspond à une instance précise d'une classe de transformation. Par exemple, la transformation "transposition" peut avoir différentes valeurs : une quinte ascendante, une octave descendante, etc. La figure 6.6.3 montre la classe `Transposition` avec ses attributs respectifs : `interval`, qui donne l'intervalle de transposition; `reduction`, qui indique quelles notes sont hors de la tessiture; et `adaptability`, qui désigne la valeur d'adaptabilité de cette transformation. L'adaptabilité d'une transposition dépend du fait qu'il y ait des notes hors de la tessiture, ou trop de notes dans une tessiture

permise mais trop haute (au-delà du Mi 4), ou encore que la continuité avec le fragment mélodique joué précédemment soit rompue. En somme, ce n'est qu'à l'égard d'une transformation particulière que nous pouvons mesurer son impact sur l'adaptabilité du cas. Il est impossible de dire que la transformation dans l'absolu rend un cas quelconque plus ou moins adaptable. Il existe une grande différence entre notre stratégie de récupération de cas et celle employée par Smyth et Keane.

<i>interval</i> domain: MusicalInterval restriction: singleValued: true	<i>reduction</i> domain: liste de notes restriction: singleValued: true	<i>adaptability</i> domain: Number restriction: [0..1] singleValued: true
--	--	--

Figure 6.6.3 - Classe *Transposition*

L'avant dernier pas de l'algorithme de récupération et de réutilisation des cas (ligne 5, tableau 6.6.2) concerne l'adaptation proprement dite du cas source choisi, plus précisément de sa PACT consécutrice. Cette opération est triviale dans notre modèle car il s'agit uniquement d'appliquer les transformations qui ont été déjà parfaitement calculées et ordonnées lors de l'estimation de l'adaptabilité du cas source choisi. Il y a, néanmoins, un calcul supplémentaire concernant la mise à jour de la description de la PACT en fonction de la transformation appliquée (Cf. tableau 6.6.6). Par exemple, si le fragment mélodique contenu dans la PACT a été transposé, alors il faut recalculer la valeur de l'attribut tessiture. De même, s'il y a eu un changement dans les amplitudes des notes, alors il faut réévaluer l'intensité globale (*loudness*).

```

fonction adapter (pactChoisie, T)
Pour toute transformation t de T faire,
  appliquer (t, pactChoisie) .
  actualiserLaDescription (t, pactChoisi)

```

Tableau 6.6.6- Fonction *adapter*

Le dernier pas de l'algorithme (ligne 6, tableau 6.6.2) sert simplement à transférer la description de la PACT consécutrice du cas source adapté vers la PACT standard (pour laquelle l'agent cherchait à calculer les notes). Ceci est fait par la fonction `transférerInformations` ci-dessous.

```

fonction transférerInformations (pactSourceChoisi, pactStandard)
  Pour tout attribut a spécialisé de pactSourceChoisi faire
    pactStandard.a <- pactSourceChoisi.a

```

Tableau 6.6.7 - Fonction *transférerInformations*

Pour conclure ce chapitre, reprenons la grille de FEUILLES MORTES dont la segmentation basée sur les schémas d'accord rencontrés dans STELLA BY STARLIGHT a été montrée dans la section 6.3.5. Poursuivant dans cette voie, la figure 6.6.4 montre le premier chorus de la ligne de basse que notre programme a généré sur FEUILLES MORTES en réutilisant les fragments mélodiques obtenus justement de la ligne de basse de Ron Carter sur STELLA BY STARLIGHT (Cf. Annexe A). Il importe de signaler que cette ligne a été générée sans les étapes d'activation et d'assemblage de PACTs. Les cas réutilisés ont été récupérés uniquement en fonction du schéma d'accord courant et de la PACT jouée précédemment. Cette ligne de basse donne un exemple intéressant où le critère adaptabilité a empêché le choix d'un cas jugé le plus similaire. Il s'agit du fragment (cas 51 de la base) joué dans les mesures 15 et 17 (accords Gmin7 Gmin7). Le cas 51, joué originellement dans les mesures 83 et 84 de STELLA BY STARLIGHT, a été préféré (Cf. tableau 6.6.8) au cas 10, joué originalement dans les mesures 19 et 20 de STELLA BY STARLIGHT, parce qu'il était plus adapté. Le cas 10 n'a pas été jugé très adapté car sa transposition entraînerait une "rupture" du point de vue du contour mélodique avec le fragment joué précédemment. La réutilisation du cas 51, au contraire, procure une homogénéité plus grande, malgré le fait qu'il soit considéré globalement moins similaire que le cas 10. Deux mesures plus tard (19-20), on trouve le même type de schéma d'accord (Gmin7 Gmin7). Cependant, cette fois-ci, c'est le cas 10 qui emporte sur le cas 51 puisque compte tenu du contour mélodique du fragment précédent, l'adaptabilité du cas 51 était excellente. D'ailleurs cette liaison entre le cas 16, dont les mesures originelles sont 29 et 30, et le cas 10 est un des passages musicalement les plus intéressants de ce chorus.

Figure 6.6.4 - Premier chorus de FEUILLES MORTES généré à partir de fragments de la ligne de basse jouée par Ron Carter sur les trois premiers chorus de STELLA BY STARLIGHT (Cf. Annexe A). Les mesures originaires de chaque fragment sont indiquées en dessus des accords

cas	mesures d'origine	similarité	adaptabilité	réutilisabilité
#10	83-84	0.636515	0.833333	0.715242
#51	19-20	0.595119	1	0.757071

Tableau 6.6.8 - Adaptabilité et similarité des cas 10 et 51 vis-à-vis du schéma d'accord (Mineur8) des mesures 15 et 16 de FEUILLES MORTES



7. IMPLEMENTATION ET EXPERIMENTATIONS

L'implémentation de notre modèle a certes été nécessaire pour sa validation, mais aussi pour la compréhension, le raffinement et le changement du modèle lui-même. La plupart des décisions que nous avons prises, ainsi que l'entendement que nous avons aujourd'hui des avantages et inconvénients des techniques que nous préconisons et utilisons, n'ont été possibles qu'après la mise en œuvre de toute une série de prototypes de notre programme. À titre d'indication, notons que le programme est actuellement plus petit qu'il ne l'était, il y a un an. Ainsi, nous avons fait l'expérience du caractère expérimental de l'IA.

Dans ce chapitre, nous commençons par présenter brièvement l'implémentation du modèle. Ensuite, nous exposons quelques expérimentations ayant un double objectif : l'appréciation de la pertinence des choix faits dans le modèle et l'évaluation empirique des résultats musicaux de notre programme vis-à-vis des programmes existants. Nous concluons en exposant les principales faiblesses d'`ImPact` par rapport aux musiciens humains.

7.1. LE SYSTEME IMPACT

Le modèle d'agent jazzman que nous avons introduit dans les chapitres précédents a été implémenté dans un système que nous appelons `ImPact` (Improvisant avec des PACTs). Dans cette section nous donnons quelques précisions sur cette implémentation.

7.1.1. Données générales

Le système `ImPact` a été programmé en Smalltalk-80 (VisualWorks 2.0) (Goldberg & Robson, 1983), un langage de programmation par objets. Le programme fonctionne sur toutes les plates-formes où Smalltalk-80 est disponible (SparcStations, Macintosh, PC, etc.). Toutefois, seuls le Macintosh et PC ont un *driver* MIDI, interface assurant la communication entre un ordinateur et un synthétiseur via le protocole MIDI. Pour le Macintosh, nous utilisons le *driver* MIDI développé par Bill Walker du CERL Group, Université d'Illinois (Walker et al., 1992). Sur une Sparc 10, le système `ImPact` dans sa configuration actuelle est capable de jouer à un tempo d'au moins 600 noires par minute, ce qui est nettement au-dessus des contraintes du temps réel, car, en jazz, on ne joue pas plus vite que 400 noires par minutes.

`ImPact` a été construit en réutilisant le système `MusES`, une plate-forme de représentation de connaissances en musique tonale, conçue au LAFORIA (Pachet, 1994). En effet, `MusES` se montre facilement réutilisable comme en témoignent d'autres systèmes développés au LAFORIA (Mouton & Pachet, 1995; Pachet & Roy, 1994; Rolland & Ganascia, 1996b) et ailleurs (Holland, 1994). Nous avons aussi fait usage de l'éditeur de partition de `MusES`. Il est important de signaler que nous n'avons pas réutilisé `MusES` "passivement". Nous lui avons aussi apporté une contribution importante, notamment en ce qui concerne les objets temporels (Pachet, 1990). En effet, nous avons défini les trois éléments qui sont à la base du modèle temporel de `MusES`, à savoir la classe `Lapse` définissant un intervalle de temps, la classe `TemporalObject` définissant un objet dont une des variables d'instance est `lapse`, et la classe `TemporalCollection` représentant un ensemble ordonné d'objets temporels.

En plus des `PACTs` et de toutes les autres classes présentées dans le chapitre 5, il y a beaucoup d'autres objets : depuis les classes représentant l'agent et tous ses composants (modules, mémoires, etc.), aux classes conçues pour tester le programme et pour le stockage des cas, en passant par les composants de l'environnement (grille, Scénario, événements, etc.). `ImPact` contient plus de 130 classes et 2500 méthodes représentant les propriétés et opérations associées à ces classes. Si on y ajoute les classes et méthodes de `MusES`, notre système a en tout plus de 220 classes auxquelles sont associés 3900 méthodes.

7.1.2. Bases de règles et de cas

En ce qui concerne les règles de production, nous avons utilisé un moteur d'inférence de premier ordre en chaînage avant écrit en Smalltalk : le système NéOpus (Pachet, 1992; Pachet, 1996). L'intégration de NéOpus au sein de notre programme a été aisée car dans NéOpus les faits correspondent à des objets Smalltalk et les prémisses et actions des règles à des expressions Smalltalk quelconques. Nous avons construit 7 bases de règles qui contiennent en tout 84 règles : 25 pour la segmentation de la grille; 35 pour l'activation des PACTs; 7 pour l'assemblage des PACTs; 6 pour la gestion du déclenchement des propagations; 8 pour la mise à jour de l'Humeur de l'agent; et 3 pour les réactions du module d'exécution. Nous avons aussi conçu quelques métabases de règles, en explorant la possibilité qu'offre NéOpus d'avoir un contrôle déclaratif du déclenchement des règles. Comme nous avons présenté dans le chapitre 6, nous avons besoin de plusieurs types de contrôle : déclenchement des règles par ordre de priorité, règles qui ne sont déclenchées qu'une fois, interruption abrupte du moteur d'inférence, etc.

En ce qui concerne la base de cas, celle que nous avons construite contient 267 cas composés à partir de 354 fragments mélodiques. Ces cas ont été obtenus à partir des lignes de basse jouées par Ron Carter (Aabersold, 1979) sur les trois premiers chorus de STELLA BY STARLIGHT (96 cas) et de WHAT IS THIS THING CALLED LOVE (99 cas). Pour compléter la base de cas de façon à avoir un certain nombre de cas pour chaque type de schéma d'accords, nous avons aussi acquis des fragments joués par Ron Carter sur CHEROKEE (23 cas), THE SONG IS YOU (23 cas), BODY AND SOUL (17 cas), et IT'S YOU OR NO ONE (12 cas). L'acquisition des cas a représenté à peu près huit mois de travail, car il nous a fallu programmer toute l'interface de saisie et toutes les classes d'objets impliquées, programmer les "anti-propagations" (Cf. section 5.5.2) pour aider la saisie, saisir les partitions, choisir les cas et les décrire.

7.1.3. Les interfaces

Pour faciliter l'acquisition des cas (Cf. section 5.5.2), nous avons implémenté quelques interfaces notamment celle concernant la description d'un cas et celle visant à aider l'accès et la manipulation des cas d'une base de cas (Cf. figure 7.1.1). Le concepteur de la base de cas peut facilement compléter ou corriger la description des fragments sous

Nous avons également mis en place une panoplie d'outils permettant aussi bien d'évaluer la possibilité d'utilisation des cas d'une base vis-à-vis d'une grille, que de garder la trace de l'utilisation réelle des cas (la requête posée, le cas source récupéré, les adaptations apportées, etc.) lors d'une séance d'accompagnement du système *ImPact* (Cf. figure 7.1.2)

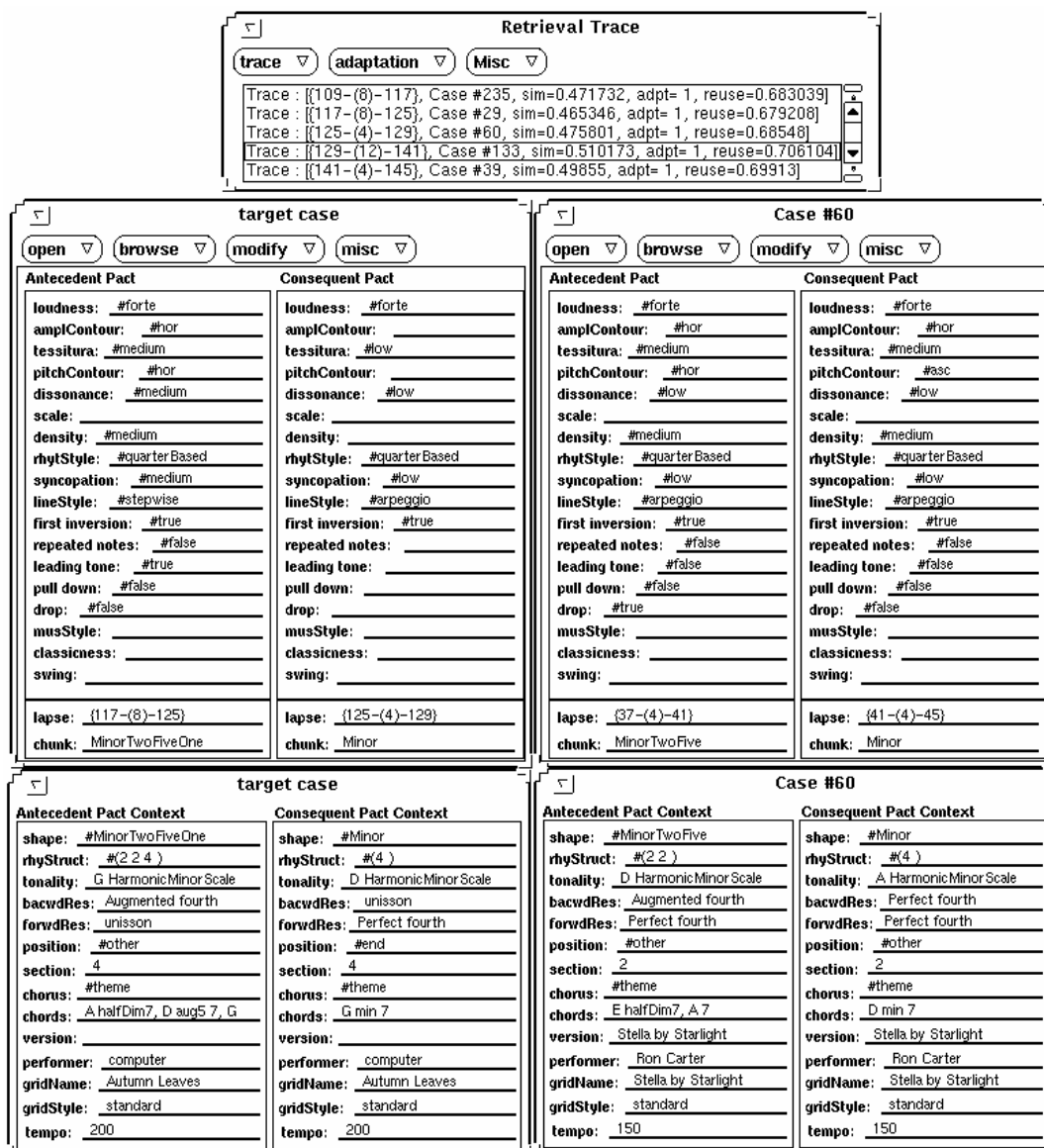


Figure 7.1.2 - Interface gérant la trace des cas utilisés. (haut-centre). Un cas cible (gauche) et un cas source récupéré (droite) sont montrés.

Nous avons aussi conçu une interface pour gérer *ImPact*, donner ses entrées, réaliser divers tests et, surtout, stocker les valeurs des paramètres du programme (la grille donnée, la Mémoire Musicale utilisée, les facteurs de similarité, etc. — Cf. figure 7.1.3).

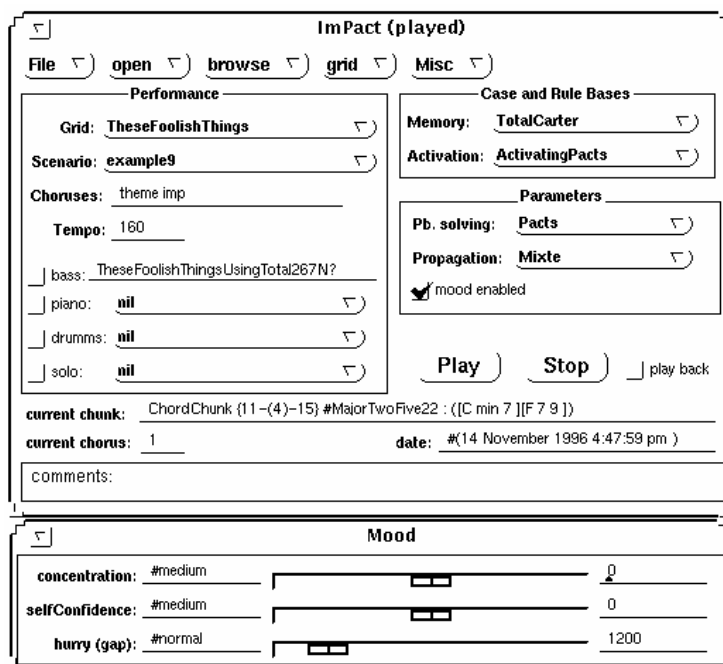


Figure 7.1.3 - Interface principale d'*ImPact*. Dans la fenêtre du haut on choisit, entre autres, les entrées du programme (grille, Scénario, déroulement de la grille, tempo), une Mémoire Musicale et une base d'activation de PACTs. La partie inférieure de la première fenêtre ainsi que la deuxième fenêtre affichent, respectivement, la position courante du programme et l'état l'Humeur.

7.2. EXPERIMENTATIONS

Dans cette section nous décrivons, à travers quelques exemples de lignes de basse produites par notre système, les conséquences musicales de choix faits dans notre modèle. En particulier, nous examinons l'importance de la segmentation de la grille en schémas d'accords, l'impact de l'ajout de cas, l'effet de l'étape d'activation et d'assemblage des PACTs dans la récupération des cas, la pertinence du critère d'adaptabilité dans la récupération des cas et l'influence du scénario et de l'Humeur. Nous faisons ensuite quelques comparaisons empiriques entre notre programme et les deux programmes qui, à notre avis, génèrent les meilleures lignes de basse

actuellement : `Band-in-a-box` et `NeurSwing`. Nous ajoutons enfin, une petite comparaison entre `ImPact` et les bassistes humains.

Nous avons fait plusieurs expérimentations sur un corpus constitué de plus d'une trentaine de grilles d'accords, la plupart des *standards* et des *blues*. Pour analyser les lignes de basse engendrées, nous avons demandé l'aide d'un des bassistes qui a collaboré avec nous (Antoine Espagno). Nous l'avons fait écouter, jouer et commenter les lignes de basse générées par `ImPact`.

7.2.1. Évaluation empirique des différents aspects du modèle

Examinons maintenant les conséquences musicales de certains aspects de notre modèle.

7.2.1.1. Ajout de cas

L'ensemble des attributs d'un cas, avec ses valeurs possibles, définissent un *espace de description*. Puisque les adaptations ne peuvent pas être systématiquement réalisées, il est souhaitable que les cas de la base couvrent le mieux possible l'espace de solutions, c'est-à-dire qu'il y ait au moins un cas assez similaire à chaque description possible. Dans notre application, nous décrivons actuellement les cas à l'aide de 26 attributs (traits musicaux et caractéristiques des segments de la grille des fragments mélodiques antécédents et conséquents) chacun pouvant en moyenne avoir 4 valeurs différentes. L'espace de descriptions a une taille d'à peu près $4^{26} \approx 4,5 \times 10^{15}$. La Mémoire Musicale contient actuellement 267 cas, un nombre donc insuffisant. Cela dit, tous les attributs n'ont pas la même importance à l'égard de la réutilisation des cas. En outre, nous pouvons réaliser quelques adaptations des fragments mélodiques récupérés de la Mémoire Musicale. C'est pourquoi, malgré le nombre insuffisant de cas, `ImPact` a pu produire des lignes de basse intéressantes comme nous le montrerons dans la suite de ce chapitre. Il nous paraît important de souligner que nous avons constaté, pendant la construction de la Mémoire Musicale, que plus nous ajoutons de cas, plus les résultats musicaux s'amélioraient.

Afin d'illustrer les conséquences musicales de l'ajout des cas, examinons l'exemple de la figure 7.2.1 qui montre deux versions de ligne de basse générées par `ImPact` sur le premier chorus de ALL OF ME à un tempo de 160 à la noire. La version

de gauche a été produite uniquement à partir de 96 cas obtenus de la ligne de basse jouée par Ron Carter sur *STELLA BY STARLIGHT* (Cf. Annexe A). En réalité, parmi ces 96 cas, seulement 40 sont *potentiellement utilisables*, car ils correspondent à des schémas d'accords selon lesquels la grille a été segmentée. Par exemple, les fragments mélodiques joués sur des II-V mineurs ne peuvent, a priori, être réutilisés car ce type de schéma n'a pas été identifié dans *ALL OF ME* pendant la segmentation. En ce qui concerne la version de droite, la Mémoire Musicale utilisée pour sa construction contient 267 cas, dont 125 sont potentiellement utilisables.

The figure consists of two side-by-side screenshots of a music software interface. Both windows have a menu bar with 'File', 'Insert notes', 'Global Edit', 'Browse', 'MIDI', 'Refresh', and 'Various'. The left window is titled 'AllofMeUsingStella96N1 by: computer' and shows a bass line with 40 potential cases. The right window is titled 'AllofMeUsingTotal267N1 by: computer' and shows a bass line with 125 potential cases. Both windows display musical notation with chord symbols above the notes.

Figure 7.2.1 - Deux versions de ligne de basse sur le premier chorus de *ALL OF ME*. Les lignes de basse de gauche et de droite ont été générées à partir d'une Mémoire Musicale contenant respectivement 40 cas et 125 cas potentiellement utilisables

Lorsque le tempo est lent, le programme active normalement la PACT “jouer avec rythme basé sur la blanche pendant le premier chorus”. Malgré l’activation de cette PACT, il n’y a pas de passages joués sur la blanche dans la version de gauche car il n’existe pas de fragments mélodiques en *STELLA BY STARLIGHT* ayant cette caractéristique. Par contre, dans la version de droite nous retrouvons les passages basés

sur la blanche en accord avec la PACT activée. Remarquons que dans cette dernière version, il n'y a pas exclusivement des passages basés sur la blanche. Ceci est dû au fait que dans la Mémoire Musicale actuelle (utilisée pour la version de droite), il n'y a pas encore de fragment mélodique basé sur la blanche pour les schémas d'accords II-V-I majeur, tels qu'ils sont rencontrés dans mesures 15-18 et 29-31 d'ALL OF ME.

En somme, plus on ajoute de cas, plus on peut espérer que de meilleures lignes de basse soient générées, car les fragments mélodiques récupérés seront plus en conformité avec les requêtes faites à la Mémoire Musicale.

7.2.1.2. *Granularité*

Pour la segmentation de la grille nous avons fait appel à des schémas d'accords car ils semblaient être musicalement plausibles (les phrases de basses coïncident assez souvent avec les schémas) et ils avaient une taille moyenne qui représentait un bon compromis de granularité (Cf. section 4.3.2). Pour tester ce choix, nous avons fait une série d'expérimentations opposant, d'une part, une segmentation basée sur les schémas d'accord et, d'autre part, une segmentation faite accord par accord. L'accord est en effet le segment minimal de notre programme. Ces expérimentations ont montré que les lignes de basse produites à partir d'une segmentation en schémas d'accords sont meilleures mais la différence n'est pas aussi importante que nous nous le pensions.

L'influence de la segmentation est illustrée par l'exemple de la figure 7.2.2 qui montre deux versions de lignes de basse jouées sur les deux premiers chorus (chaque chorus ayant 12 mesures) de BLUESETTE à un tempo de 200 à la noire et sans événement dans le Scénario. La version de gauche a été générée sur une segmentation accord par accord tandis que pour la version de droite, la segmentation a été basée sur les schémas d'accords.

Ce qui nous a d'abord surpris dans les lignes construites accord par accord est qu'il y avait toujours des "répétitions" de phrases plus longues qu'un segment (un accord). Par exemple, dans la ligne de basse de gauche de BLUESETTE, le passage des mesures 5-6 est répété, à des transpositions près, dans les mesures 7-8 et 9-10. De même, le passage de la mesure 15 est répété, avec des transpositions, aux mesures 16, 18 et 22. Cela est surprenant car il est difficile de tenir compte du niveau plus global

avec des choix très locaux, c'est-à-dire des choix confinés à des segments de taille très courte (Cf. section 4.3.2). Nous pensions, en effet, que les lignes générées à partir d'une segmentation en schémas d'accords seraient nettement plus homogènes et qu'elles auraient beaucoup plus de phrases récurrentes que les lignes conçues accord par accord. La différence à cet égard n'est pas vraiment sensible. Ces dernières lignes ont autant de phrases récurrentes que celles basées sur les schémas d'accords.

Figure 7.2.2 - Deux versions de ligne de basse sur deux chorus de BLUESETTE. La segmentation de la grille a été minimale (accord par accord) pour l'exemple de gauche, tandis que celle de l'exemple de droite a été basée sur les schémas d'accords

Il est important de souligner que le phénomène de répétition de phrases qui a lieu dans les deux versions de ligne de basse de BLUESETTE est renforcé par le fait que nous n'avons ajouté aucun événement au scénario. La grande majorité de PACTs impliquées dans l'assemblage, et donc dans la requête à la Mémoire Musicale, est composée de PACTs activées "par défaut" selon la structure de la grille (par exemple "si le tempo est moyennement rapide et l'agent commence à jouer le premier chorus d'exposition du thème, alors jouer avec un rythme basé sur la noire et peu syncopé pendant le premier chorus", "si, indépendamment du tempo, l'agent commence à jouer le premier chorus d'exposition du thème, alors jouer des arpèges, dans une tessiture plutôt basse et avec une faible dissonance", etc.). Puisque ces PACTs durent tout un

chorus, elles favorisent, à l'intérieur du chorus, la récupération de cas ayant les mêmes caractéristiques.

Pour ce qui est de la qualité globale des lignes de basse, celles construites sur des schémas d'accords sont meilleures, mais pas autant que nous imaginions. Un problème critique des approches basées sur la réutilisation systématique de fragments mélodiques est justement la transition (ou le "collage") entre les fragments. Or, puisque qu'une segmentation de type "accord par accord" génère des segments plus petits et plus nombreux, on peut s'attendre à ce qu'il y ait plus de problèmes dans les transitions. En effet, reprenant BLUESETTE, on peut remarquer que les transitions "intra-schémas" comme celles entre le Ab7 et Dbmaj7 (mesures 6-7), entre le Gb7 et le Cbmaj7 (mesures 8-9), entre le Bb7 et le Ebmaj7 (mesures 16-17) et entre le Dbmin7 et Gb7 (mesure 20) sont globalement meilleures dans la ligne de basse de droite. À l'exception de celle entre le Db7 et le Cmin7 (mesures 11-12), les transitions "inter-schémas" de la version de droite sont meilleures. C'est notamment le cas de celles entre le Dbmaj7 et Dbmin7 (mesures 7-8), entre le C7 et le Fmin7 (mesures 15-16), et surtout celle entre le D7 et Gmin7 (mesure 14-15).

Parmi toutes les transitions que nous venons d'indiquer, celle entre le D7 et Gmin7 (mesures 14-15) montre une situation où la segmentation en schémas d'accords peut être particulièrement intéressante. Le fragment de la mesure 14 correspond à celui joué par Ron Carter à la sixième mesure de BODY AND SOUL (Cf. annexe A). Ce fragment a été originellement joué sur un II-V mineur (Cm7(b5) F7) suivi par un II-V mineur (Bbm7 Eb7) exactement comme dans la grille de BLUESETTE. C'est certainement à cause de cette similarité que ce fragment a été récupéré. Il est très courant de trouver des enchaînements de schémas d'accords comme les deux II-V's des mesures 14 et 15 dans Bluesette. C'est pourquoi on peut espérer que souvent, les transitions "inter-schémas" soient bonnes.

7.2.1.3. Adaptabilité

La récupération d'un cas source de la Mémoire Musicale ne se fait pas uniquement selon la similarité avec le cas cible (requête) donné. Comme nous l'exposons dans la section 6.6.2, les cas sont récupérés selon un critère de réutilisabilité qui tient compte à la fois de la similarité et de l'adaptabilité vis-à-vis du segment d'improvisation courant.

Dans les grilles utilisées pour les expérimentations nous avons constaté qu'en moyenne 80% des cas jugés les plus réutilisables étaient aussi ceux qui ont la meilleure valeur de similarité. Ce qui montre que la similarité est un bon critère. Toutefois, la similarité demeure insuffisante en tant que critère de récupération car une fois sur cinq la non prise en compte de l'adaptabilité pourrait engendrer de mauvais résultats musicaux.

Examinons un exemple où l'importance de l'adaptabilité se révèle évidente. La figure 7.2.3 montre deux versions de ligne de basse générées par ImPaCt sur les deux premiers chorus (16 mesures) de GIANT STEPS à un tempo de 160 à la noire. Pour la version de gauche, la récupération se base uniquement sur la similarité, tandis que pour celle de droite, c'est la réutilisabilité (similarité plus adaptabilité) qui a été adoptée. La grille est segmentée de façon identique pour les deux versions et les cas disponibles sont aussi les mêmes.

Figure 7.2.3 - Deux versions de ligne de basse sur deux chorus de GIANT STEPS. Les cas composant les lignes de basse de droite et de gauche ont été respectivement choisis selon une mesure de similarité et de réutilisabilité.

Le premier segment dans lequel le choix des cas diverge est celui de la fin de la première mesure (D7). Le fragment mélodique de la version de droite prépare mieux l'accord G que celui de la version de gauche, où il y a un saut de neuvième descendante qui n'est pas heureux. Pour le deuxième fragment différent (mesure 5), les choix faits dans les deux versions sont musicalement équivalents. En ce qui concerne les fragments des mesures 8-9, 12-13 et 13-14, les fragments joués dans la version de gauche n'ont pas été retenus dans celle de droite car, soit ils impliquaient une transposition ou remplacement du Mi bémol qui tombe hors de la tessiture de la basse (fragment des mesures 8-9 et 13-14), soit ils provoquaient des sauts trop importants dans les transitions avec les fragments antécédents (fragment des mesures 8-9 et 12-13). C'est aussi pour éviter des sauts inopportuns que le choix des cas est différent dans les mesures 24-25 et 28-29.

7.2.1.4. *Activation et assemblage des PACTs*

Nous avons mené des expérimentations pour vérifier l'importance de l'activation et de l'assemblage des PACTs comme une étape d'enrichissement et de raffinement de la requête posée à la Mémoire Musicale. Pour cela, nous avons comparé des lignes de basse générées par *ImPact* dans sa version normale, appelée version "CBR-PACTs" (raisonnement à partir de cas avec des PACTs), avec celles d'une version où les fragments sont récupérés uniquement en fonction des caractéristiques du segment de la grille courant et précédent et des traits musicaux du fragment mélodique précédent (appelée version "CBR-Naïf").

Avant de mettre au point toute la base de règles d'activation des PACTs, nous avons généré quelques exemples avec une version CBR-Naïf. Puisque nous pensions qu'il s'agissaient de lignes de basse de qualité "acceptable", nous les avons données à Antoine Espagno pour qu'il puisse les jouer et les évaluer. Sa réponse fut : "Les lignes de basse de votre programme ne sont pas logiques. Pas cohérentes du tout". C'est une critique qui nous a étonnés, car nous attendions plutôt des remarques comme "ce n'est pas intéressant", "il n'y a rien de nouveau", "c'est monotone", etc. D'après lui, les lignes CBR-Naïf n'étaient pas "logiques" car, d'une part, le programme enchaînait assez souvent des fragments qui avaient peu à voir l'un avec l'autre au niveau de certains traits musicaux. D'autre part, on ne se rendait pas compte de ce que le

programme cherchait à accomplir. Comme on dirait en jazz, le programme en version CBR-Naïf ne “racontait pas une histoire”. En effet, il est, en général, souhaitable qu’une “progression” cohérente et graduelle de l’improvisation et de l’accompagnement s’établisse : on commence en jouant des choses simples pendant l’exposition du thème; ensuite on joue de façon de plus en plus complexe en atteignant le point culminant quelque part pendant les chorus d’improvisation; et, enfin on retourne au simple (thème). Par simple nous voulons dire moins dissonant, peu dense, moins fort, en première inversion, etc. C’est justement ça qui n’apparaissait pas dans les lignes CBR-Naïf, car la similarité avec caractéristiques du segment de la grille courant et précédent et des traits musicaux du fragment mélodique précédent n’était pas suffisante pour *contrôler* le choix des fragments.

Prenons l’exemple d’une ligne produite avec la version CBR-Naïf à un tempo de 140 à la noire sur les deux premiers chorus de FEUILLES MORTES (Cf. figure 7.2.4). Une des premières remarques détaillées qu’Antoine Espagno nous a faite est qu’un bassiste doit *toujours* commencer la grille en jouant la tonique au premier temps (première inversion). Or, dans la ligne de la figure 7.2.4, `ImPact` commence (mesures 1-3) en jouant une phrase qui est trop dissonante (ou “non-logique”) pour un début de grille. Ne serait-ce que sur le premier accord, la première note est un si bémol (la septième de Cm7 !) et ni la tonique ni la tierce ne sont jouées. Pour le F7 qui suit, `ImPact` joue de façon chromatique. C’est un passage intéressant mais il devrait apparaître plus tard. En outre, des alternances “peu logiques” (comme dirait Antoine Espagno) entre des passages très denses et assez syncopés avec des passages peu denses et peu syncopés se prolongent tout au long du premier chorus. Ainsi, les fragments des mesures 4 et 12, joués respectivement à la ronde et à la blanche ne sont pas très opportuns : ils seraient mieux placés au début de la carrure. Pour ce qui est des changements musicaux entre les deux types de chorus (thème et improvisation), on identifie une légère différence. Ceci est dû au fait que l’attribut `chorus` est utilisé pour décrire les schémas d’accords (ChordChunk - Cf. figure 5.5.3), ce qui augmente la mesure de similarité des fragments joués sur le même type de chorus. Mais, cet unique attribut n’est pas suffisant pour permettre un choix cohérent des cas.

The figure consists of two side-by-side screenshots of a music software interface. Both windows have a title bar that reads "AutumnUsingTotalCarter267N2 by computer". Each window has a menu bar with options: File, Insert notes, Global Edit, Browse, MIDI, Refresh, and Various. The left window displays a bass line starting at measure 1 and ending at measure 28. The right window displays a bass line starting at measure 33 and ending at measure 61. Both windows show musical notation in bass clef with various chord symbols written above the notes. The chord symbols include C min 7, F 7, Bb maj7, Eb maj7, A halfD im7, D 7, G min 7, G min 7, C 7 9, F min 7, Bb 7, and A halfD im7 aug 7.

Figure 7.2.4 - Ligne de basse sur deux choros de FEUILLES MORTES sans activation et assemblage de PACTs

La figure 7.2.5 montre une autre version (CBR-PACTs) de ligne de basse jouée sur FEUILLES MORTES jouée dans le même tempo que celle de la version CBR-Naïf. La différence avec cette dernière version est étonnante du point de vue de la cohérence. Nous avons, en effet, incorporé toutes les règles qu'Antoine Espagno nous avait suggérées comme "jouer la tonique au début de la grille", "jouer plus simple pendant le premier choris et ensuite jouer de plus en plus complexe jusqu'au retour au thème", etc. en tant que règles d'activation de PACTs. L'activation de ces PACTs permet un contrôle plus précis des choix des cas. Naturellement, la "cohérence" de cette ligne de basse, surtout au premier choris, est loin d'être parfaite car il y a encore trop peu de cas dans la Mémoire Musicale. Par conséquent, ImPact ne peut toujours choisir un fragment qui satisfait aux critères représentés par les PACTs activées. Par exemple, dans les schémas de type II-V-I des mesures 1 à 3, 9 à 11 et 21 à 23 de FEUILLES MORTES les fragments mélodiques sont basés sur la noire bien qu'une PACT "jouer sur la blanche" ait été activée.

The image displays two side-by-side screenshots of a music software interface titled "AutumnLeavesUsingTotal267N5 by computer". Each screenshot shows a bass line in bass clef with various chords indicated above the notes. The left screenshot shows measures 1-29, and the right screenshot shows measures 33-61. The chords are: C min 7, F 7, Bb maj7, Eb maj7, A halfDm7, D 7, G min 7, G min 7, C min 7, F 7, Bb maj7, Eb maj7, A halfDm7, D 7, G min 7, G min 7, A halfDm7, D 7, G min 7, G min 7, C min 7, F 7, Bb maj7, Eb maj7, A halfDm7, D 7, G min 7, C 7 9, F min 7, Bb 7, Eb maj7, A halfDm7 aug 7, G min 7, G min 7.

Figure 7.2.5 - Ligne de basse sur deux choruses de *FEUILLES MORTES* avec activation et assemblage de PACTs

Puisque nous parlons de l'influence de l'activation et de l'assemblage des PACTs dans les lignes de basse produites, il est important de signaler des aspects positifs et négatifs de notre modèle.

L'aspect positif est la facilité avec laquelle nous avons pu ajouter des règles d'activation de PACTs et ainsi faire évoluer le système. Si l'on est capable d'identifier les règles, leur ajout se fait facilement car il n'est pas nécessaire de vérifier si les nouvelles règles vont engendrer des conflits avec celles déjà existantes. En effet, comme nous l'avons présenté dans la section 4.3, on active une PACT en fonction des données perceptives et sans avoir besoin de connaître les PACTs activées auparavant. Les conflits sont gérés durant l'assemblage.

L'aspect négatif et, certainement, le grand défaut de notre modèle est que l'on n'est jamais sûr qu'un critère, lié à une PACT activée, sera respecté. Autrement dit, il n'y a aucune garantie qu'un fragment récupéré aura une propriété musicale donnée. Il

existe quatre raisons à cela. La première est qu'il se peut que la PACT, représentant le ou les critères que l'on veut fixer, soit "mangée" par les autres PACTs pendant le processus d'assemblage. Il est tout fait courant qu'une PACT importante peu jouable ou très ancienne (date de création lointaine), comme "jouer peu de notes car le soliste en joue beaucoup", ne "survive" pas à la résolution de conflits de l'assemblage. Nous avons, par exemple, été amenés à employer quelques "ruses" pour minimiser ce problème. Au lieu d'activer, au début du premier chorus d'une chanson lente, une seule PACT de type "jouer peu de notes, peu dissonant, peu syncopé, pas très fort, etc.", nous activons plusieurs PACTs : "jouer peu de notes", "jouer peu dissonant", "jouer peu syncopé", etc. Si nous ne faisons pas ainsi, lorsque le programme activerait une PACT de type "jouer dissonant", parce qu'il devrait réagir à l'environnement ou qu'il aurait trouvé des accords altérés sur la grille, cette PACT ne survivrait pas face à la première PACT très jouable ("jouer peu de notes, peu dissonant,..."). En activant les PACTs séparément, quand les deux PACTs "jouer dissonant" et "jouer peu dissonant" s'affronteront, c'est la plus jeune qui l'emportera. Dans certains cas, comme celui que nous venons de citer, quelques "ruses" peuvent aider mais le problème est difficile à résoudre en général.

La deuxième raison tient à la récupération même d'un cas. Même si une PACT a pu "survivre" à l'assemblage, rien ne dit que le critère qu'elle représente sera respecté par le fragment mélodique récupéré. Comme nous l'avons montré dans la section 7.2.1 et dans les exemples de FEUILLES MORTES de la figure 7.2.5, s'il n'y a pas de fragments respectant un critère (par ex. jouer en se basant sur la blanche) et si notre programme ne sait pas les adapter, alors ce critère ne sera pas pris en compte.

La troisième raison concerne aussi la récupération des cas. Puisque nous employons une mesure *globale* de similarité, on ne peut être sûr qu'un tel critère sera considéré au détriment des autres. Bien entendu, il y a les poids associés (`similarityFactor` Cf. section 5.5.3) à chaque attribut (traits) des PACTs mais cela ne suffit pas toujours. Ainsi, considérons le calcul des notes du segment correspondant aux mesures 51 et 52 (accord mineur durant huit temps) de la ligne de la figure 7.2.5. Normalement au début du chorus d'improvisation, la "PACT à découper" (Cf. section 5.4) "jouer de plus en plus syncopé" est activée. À partir du milieu du chorus, cette PACT à découper engendre la PACT standard "jouer très syncopé". Cette dernière se

combine, à travers le processus d'assemblage, avec d'autres PACTs produisant la PACT "jouer très dissonant, très syncopé, en *stepwise* et avec un rythme basé sur la noire pendant le segment des mesures 51 et 52". Malheureusement, la PACT "jouer très syncopé" n'arrive pas à influencer suffisamment le choix des cas. Le fragment récupéré, originellement joué par Ron Carter aux mesures 83 et 84 de *STELLA BY STARLIGHT* (Cf. annexe A), n'est pas syncopé. Le programme l'a choisi parce que vis-à-vis des autres critères ce fragment est pertinent (il est dissonant, basé sur la noire et *stepwise*). Il y a bien des fragments syncopés et joués sur un accord mineur durant huit temps dans la Mémoire Musical, comme le fragment originellement joué par Ron Carter aux mesures 3 et 4 de *WHAT IS THIS THING CALLED LOVE* (Cf. Annexe A). Toutefois, ce fragment n'est pas préféré car, bien qu'il soit assez syncopé et soit joué en *stepwise*, son rythme est basé sur la blanche et il est moyennement dissonant.

La dernière raison est liée au fait que nous tenons compte aussi de l'adaptabilité des cas pendant leur récupération. Comme nous l'avons précédemment examiné, il se peut qu'un cas source soit très similaire au cas cible donné mais qu'il ne soit pas adapté au segment d'improvisation courant.

En somme, bien que l'activation de PACTs raffine la requête posée à la Mémoire Musicale, le contrôle du choix des cas demeure difficile.

7.2.1.5. *Scénario*

Nous avons utilisé diverses grilles et Scénarios pour la mise au point et validation d'ImPact. D'une manière générale, ImPact s'est montré capable de changer de façon très satisfaisante ce qu'il était en train de jouer en fonction des événements du Scénario. Néanmoins, les réactions d'ImPact face à l'environnement sont parfois très tardives. Pour illustrer ces propos, examinons les deux versions de lignes de basse jouées sur *ALL OF ME* de la figure 7.2.6. Ces lignes ont été générées au même tempo et avec la même configuration de la Mémoire Musicale que la version droite de la figure 7.2.1, mais elles tiennent compte de quelques événements musicaux du Scénario.

La version gauche de la figure 7.2.6 a été "perturbée" par l'événement H "on joue *hot*" qui commence vers le milieu de la mesure 8 et finit vers le milieu de la mesure 21. Cet événement provoque l'activation de PACT "jouer beaucoup de notes,

syncopé, fort, avec swing, avec des *pull-downs* et dans une tessiture plutôt haute” (Cf. tableau 6.4.7). Comme on peut remarquer, l’activation de cette PACT “assez jouable” bouleverse la ligne de basse qui serait jouée sans sa présence (Cf. figure 7.2.1 - droite). Il s’agit d’une démonstration très intéressante de la capacité d’ImPact à adapter son comportement à l’environnement parce que, d’une part, la nouvelle ligne de basse est musicalement cohérente avec la nouvelle “ambiance” musicale, la réutilisation du fragment des mesures 19-20 étant particulièrement réussie. D’autre part, la réaction d’ImPact est presque immédiate. Sur une Sparc 10 ayant 64 Mo de mémoire vive, ImPact tourne très vite, ce qui permet que la taille maximale de ERE (Cf. section 1.2.3) soit fixée à deux temps et que les événements assez récents puissent être considérés avant le déclenchement de l’activation des PACTs. C’est ce qui a permis à ImPact de prendre en compte l’événement H très rapidement (à partir de la mesure 9). Toutefois, pour ce qui est de la fin de cet événement, la réaction d’ImPact est trop lente. Il aurait dû se remettre à jouer “calmement” vers la mesure 22 ou 23, mais il ne le fait qu’à la mesure 25. Le problème est qu’une fois l’activation des PACTs engagée, le cycle d’inférence suit son cours normal jusqu’à ce que les notes d’un segment d’improvisation donné soient toutes déterminées. Dans cette période, ImPact est insensible aux événements qui sont entre train d’avoir lieu. Comme la fin de l’événement H a lieu au début du segment des mesures 21 à 24, il ne peut être pris en compte que dans le prochain cycle d’inférence. Plus les segments sont longs, plus il y a le risque qu’ImPact “ignore” l’environnement. C’est un défaut que nous avons prévu (Cf. section 4.2), et dont les remèdes ne sont pas simples à mettre en œuvre car il n’y pas de théorie générale de révision de plan (McDermott, 1992). Naturellement, avec quelques techniques de re-planification où des étapes de perception sont intercalées avec celles de raisonnement (Georgeff & Lansky, 1987; Kaebling, 1987), il est possible de minimiser ce problème. Néanmoins, pour les réactions vraiment rapides, comme “jouer dans les creux laissés par le soliste”, il est nécessaire de doter la machine d’une capacité à anticiper sur ce que les autres musiciens vont jouer (Cf. section 2.2.3). C’est une perspective de recherche intéressante, d’autant plus que l’anticipation n’a pratiquement pas encore été considérée dans les projets de programmes qui jouent du jazz.

The figure displays two side-by-side screenshots of a music software interface, likely a digital audio workstation (DAW), showing the bass line notation for the first chorus of the song "All of Me".

The left window is titled "AllofMeUsingTotal267HOT by: computer" and the right window is titled "AllofMeUsingTotal267Diss2 by: computer". Both windows show a sequence of chords and notes across 29 measures.

The chords shown in both windows are: C6/9, C6/9, E7, E7, A7, A7, D min 7, D min 7, E7, E7, A min 7, A min 7, D7 9 11 13, D7 9 11 13, D min 7, G7, C6/9, C6/9, E7, E7, A7, A7, D min 7, D min 7, F6/9, F min 6/9, E min 7, A7 9, D min 7, G7 9 11 13, C6/9, C6/9.

The right window has a circled "D" at measure 14 and a circled "H" at measure 28, indicating specific events.

Figure 7.2.6 - Deux lignes de basse jouées sur le premier chorus de *ALL OF ME* en réaction à des événements H "on joue hot" et D "le pianiste joue très consonant"

Avant les expérimentations nous pensions qu'ImPact réagirait bien aux événements plus globaux, comme "on joue *hot*", car ils provoquent l'activation de PACTs "assez jouables". Mais, nous ne croyions pas que les réactions seraient perceptibles lorsqu'il s'agit d'événements qui impliquent l'activation de PACTs "peu jouables". Comme nous avons exposé dans la section précédente, nous avons peur que les PACTs activées par réaction à l'environnement soient "mangées" par d'autres, ou bien, que ces premières n'arrivent pas à influencer la récupération du fragment mélodique de la Mémoire Musicale. Bien que ce type de problème se produise de temps en temps, les résultats ont été meilleurs que nous ne les espérions. À gauche de la figure 7.2.6 se trouve une version de ligne de basse dont la génération a été influencée par l'événement D "le pianiste joue très consonant" qui commence à la mesure 14 et finit vers le milieu de la mesure 28. Pour ce type d'événement, la PACT "jouer dissonant et en *stepwise*" est activée pour créer un contraste avec le pianiste. À partir de la mesure 15, la réaction d'ImPact se fait sentir. On peut remarquer qu'ImPact préfère quelques

fois, notamment aux segments des mesures 18-19 et 26-27, jouer plus dissonant en détriment de la PACT “jouer sur la blanche pendant le premier chorus”

Comme nous venons de l’illustrer, *ImPact* s’est montré capable d’être sensible à son environnement et de réagir très souvent convenablement. La plupart du temps, toutefois, le Scénario a le grand défaut d’être figé, et étant figé depuis le départ. Comme nous l’avions prévu (Cf. section 4.1.2), on ne peut pas espérer qu’une véritable interaction entre *ImPact* et l’environnement ait lieu. Malgré cela, le Scénario nous a été extrêmement utile pour pouvoir, sans avoir à faire à tous les grands problèmes de l’automatisation de l’écoute, mettre au point et tester des mécanismes d’adaptation de comportement de notre agent face à l’environnement.

7.2.1.6. Humeur

Toutes les lignes de basse que nous avons montrées jusqu’ici ont été produites sans tenir compte des valeurs de l’Humeur : presse (urgence), concentration, et confiance. Examinons dans cette section l’influence que peut avoir les deux dernières variables de l’Humeur dans les résultats musicaux : la “concentration” et la “confiance”. Ces deux variables changent de valeur uniquement selon les événements non-musicaux du Scénario, c’est-à-dire les actions du public et les signes visuels des autres musiciens (Cf. sections 4.3.5 et 6.4.1).

La figure 7.2.7 montre une version de ligne de basse jouée sur ALL OF ME dans le même tempo que celles présentées dans ce chapitre. Tout comme celle de la figure 7.2.6 (gauche), cette nouvelle version est “perturbée” par l’événement H “on joue *hot*” vers le milieu de la mesure 8, provoquant l’activation de PACT “jouer beaucoup de notes, syncopé, fort, avec swing, avec des *pull-downs* et dans une tessiture plutôt haute” à partir de la mesure 9.

Figure 7.2.7 - Ligne de basse jouée sur le premier chorus d'ALL OF ME en réaction à des événements H "on joue hot", P "le public siffle" et M "un musicien dit: "n'aie pas peur"".

Supposons que cette nouvelle version soit jouée pour un public "plus averti" qui au début de la mesure 13 manifeste son mécontentement (événement P "le public siffle") contre une ambiance "hot" trop précoce. Cet événement P implique une diminution de la "confiance" de l'agent et entraîne un changement de la façon dont l'agent joue à travers l'activation de la PACT "jouer un peu différemment par rapport au fragment mélodique des mesures 12-13". Compte tenu des valeurs des traits musicaux du fragment (PACT jouable) en question (mesures 12-13), cette nouvelle PACT activée est donc "jouer sur la blanche, un peu dissonant, en *stepwise*, en direction ascendante, etc.". Cette nouvelle PACT, étant plus jouable que celle activée en réaction à l'événement H, impose un changement à partir du segment de la mesure 14, de la ligne de basse qui serait jouée si le public ne s'était pas manifesté (Cf. figure 7.2.6 - gauche). Dans le segment subséquent (mesures 15-18), ImPact choisit un fragment qui n'est pas vraiment basé sur la blanche, bien qu'il ne soit pas non plus "hot", car dans la Mémoire Musicale il n'y a pas de cas basés sur la blanche pour les schémas II-V-I. Le

public arrête de siffler vers la mesure 17. Néanmoins, *ImPact* ne se remet pas à jouer “hot” à partir du segment tout de suite après (mesures 19-20), car sa “confiance” est restée très basse. C’est alors que, dans la mesure 20, un autre événement (M) de type “un musicien dit : n’aie pas peur” intervient, en remettant la valeur de la “confiance” d’*ImPact* au normal. À partir de la mesure 21, *ImPact* peut à nouveau réagir à ce que les autres musiciens jouent, car la seule PACT qui sera activée est celle correspondant à l’événement H. *ImPact* joue “hot” jusqu’à la fin, car cette fois-ci, contrairement à l’exemple de la figure 7.2.6 (gauche), l’événement H se prolonge jusqu’au bout (peut-être parce que les musiciens étaient un peu fâchés contre intervention du public ?).

Il est certes “amusant” d’inventer des scénarii comme celui que nous venons de décrire. Mais, musicalement parlant, nous ne pouvons vraiment dire que l’ajout des variables “confiance” et “concentration”, ainsi que de l’intervention du public, génère de meilleures lignes de basse. Ces éléments n’auraient vraiment un intérêt musical que si nous pensions à doter *ImPact* d’une interface permettant une vraie interaction en directe avec un public ou des musiciens. C’est d’ailleurs, la principale contribution que les expérimentations avec le scénario nous ont apportée. Elles nous ont montrés qu’il serait intéressant dans le futur de construire une interface qui remplacerait le Scénario, ne serait-ce que pour être utilisée par des musiciens, l’interaction avec public étant encore plus compliquée. Ainsi, si un musicien souhaitait jouer “hot”, alors il appuierait, avec le pied par exemple, sur un bouton pour indiquer son intention à *ImPact*. De même, s’il n’aime pas ce qu’*ImPact* est en train de jouer, il pourrait le lui faire savoir.

7.2.1.7. *Le manque de temps*

Lorsque *ImPact* manque du temps pour raisonner, il court-circuite l’activation et l’assemblage des PACTs afin de chercher directement un fragment mélodique récurrent (un cliché, un pattern) dans la Mémoire Musicale. Deux raisons nous ont conduits à utiliser une telle stratégie de raccourcissement. D’abord, nous voulions permettre au module de raisonnement de travailler “très proche” (ERE petit — Cf. section 1.2.3) de celui d’exécution, tout en garantissant que lorsque le module d’exécution aura terminé de jouer un fragment, le prochain soit déjà déterminé. Ensuite, puisque dans ces situations critiques *ImPact* cherche un cliché, nous pensions que la ligne de basse générée serait plus semblable à celles jouées par les vrais jazzmen (Cf. section 6.2).

Nous avons eu beaucoup de mal à évaluer cette stratégie de raccourcissement du raisonnement de notre agent. En premier lieu, il nous a fallu choisir un seuil “minimal” et un seuil “maximal” de ERE très proches pour “forcer” le manque de temps. En effet, comme nous l’avons dit dans la section précédente, *ImPact* est suffisamment efficace pour qu’il n’ait jamais besoin de raccourcir son raisonnement. En deuxième lieu, ce phénomène de manque de temps est difficile à contrôler, car chaque cycle d’inférence d’*ImPact* prend un temps différent selon le nombre de PACTs activées et le nombre de cas appartenant à la catégorie (forme plus structure rythmique) du schéma d’accord courant. Les raccourcissements sont donc asynchrones. En dernier lieu, l’ensemble de fragments clichés que nous avons identifié a été obtenu à partir d’un corpus restreint (à peu près 10 minutes de lignes de basse) constitué uniquement par ce qu’un bassiste en particulier (Ron Carter) a joué. Juste en écoutant *ImPact*, il est difficile de reconnaître s’il joue davantage de clichés lorsqu’il a moins de temps pour raisonner. Peut-être que quelqu’un qui connaît bien le travail de Ron Carter pourrait effectivement identifier ces clichés.

Pour mener les expérimentations concernant le manque de temps nous avons dû classer les fragments mélodiques contenus dans la Mémoire Musicale selon leur degré de récurrence (*classiness*) (Cf. section 5.5.2). Cette classification demande un travail très laborieux, car il faut identifier et compter les occurrences *approximatives* d’un fragment mélodique donné dans le corpus. C’est pourquoi nous avons fait appel au programme *Imprology* développé au LAFORIA par Rolland (Rolland & Ganascia, 1996a; Rolland & Ganascia, 1997) pour l’extraction de patterns dans les solos de jazz, en particulier ceux de Charlie Parker. Ce programme nous a donc fourni de façon plus rapide et, surtout, plus fiable les fragments les plus récurrents dans la Mémoire Musicale par rapport à un corpus constitué d’une douzaine de chorus de 5 différentes grilles jouées par Ron Carter.

D’après les tests que nous avons faits, il nous semble évident que la stratégie de raccourcissement n’a rien apporté musicalement aux lignes de basse. Au contraire, lorsqu’un raccourcissement se produit, le contrôle du choix des fragments mélodiques diminue car la requête à la Mémoire Musicale est moins complète. Afin d’illustrer ces faits, comparons la ligne de la figure 7.2.5 (dorénavant appelée “ligne originelle”) avec une nouvelle version présentée dans la figure 7.2.8 (dorénavant appelée “nouvelle

ligne”). Les segments marqués sur cette dernière figure sont ceux où le raisonnement à subi un raccourcissement.

Figure 7.2.8 - Nouvelle version de ligne de basse jouée sur FEUILLES MORTES. Les segments marqués sont ceux où le raisonnement à subi un raccourcissement

Certains raccourcissements ne changent en rien la nouvelle ligne par rapport à l’originelle, car les fragments mélodiques joués originellement sont par hasard très récurrents. Ainsi, le fragment mélodique joué dans les mesures 4 et 20 de la nouvelle ligne apparaît une douzaine de fois dans le corpus et celui des mesures 30-31 et 62-63, une demi-douzaine de fois. D’autres raccourcissements impliquent le choix de nouveaux fragments mélodiques qui sont tout à fait adéquats à l’égard du contexte musical. C’est le cas des fragments mélodiques des mesures 9-11, 39-40, 47-48 et 59. Toutefois, certains choix occasionnés par le raccourcissement du raisonnement ne sont pas tout à fait adéquats (mesures 5-6 et 25-26), ou ils sont carrément mauvais (mesures 17-18 et 36). En effet, les fragments mélodiques mesures 17-18 et 36 nuisent au développement et homogénéité de la ligne. Le fait curieux est que, tandis que le fragment des mesures 5-6 et 25-26 est récurrent, car il apparaît 17 fois dans le corpus,

celui des mesures 17-18 ne l'est pas. Ce dernier a été choisi car le cas jugé le plus similaire à la requête (cas cible) donnée, en l'occurrence celui des mesures 5-6 et 25-26, ne serait pas très adapté. Soit la première note serait un La 3, et entraînerait une montée trop importante de la ligne, soit elle serait un La 2, et impliquerait alors un saut important avec la dernière note (Sol dièse 3) de la mesure précédente. Cela illustre que le raccourcissement augmente la possibilité d'un choix non adéquat puisque le seul attribut spécialisé de la PACT conséquent du cas cible sera la `classiness`.

En somme, compte tenu que `ImPact` fonctionne beaucoup plus vite que nous ne l'avions prévu, il nous semble inopportun de limiter son temps de réflexion si les résultats musicaux ainsi produits ne sont pas meilleurs.

7.2.1.8. Bilan

Les expérimentations que nous avons menées ont pu justifier l'intérêt de différents éléments de notre modèle, à savoir la segmentation de la grille en schémas d'accords, l'activation et l'assemblage des PACTs comme raffinement de la requête posée à la base de cas, la prise en compte de l'adaptabilité dans la récupération des cas. En outre, nous avons constaté que l'ajout de cas et des règles d'activation de PACTs peut se faire facilement et peut contribuer à améliorer la qualité des lignes de basse générés par `ImPact`.

Néanmoins, la stratégie de raccourcissement du raisonnement et l'utilisation des variables "concentration" et "confiance" n'ont pas donné de meilleurs résultats musicaux. En outre, nous avons vérifié que, bien qu'`ImPact` se soit montré capable de s'adapter de façon plutôt satisfaisante à l'environnement, un véritable dialogue avec ce dernier ne peut se réaliser qu'à travers le développement du module de perception ou les interfaces interactives spéciales.

Nous avons enfin identifié un des problèmes les plus importants dans notre modèle : le manque de garantie qu'un critère musical correspondant à une PACT précise soit pris en compte dans la récupération du fragment mélodique de la Mémoire Musicale.

7.2.2. Comparaison avec `Band-in-a-box` et `NeurSwing`

Dans la suite, nous faisons quelques comparaisons entre les lignes de basse d'`ImPact`, de `Band-in-a-box` et de `NeurSwing`. Les réflexions que nous présentons sont basées sur l'étude d'une demi-douzaine (`NeurSwing`) et une douzaine (`Band-in-a-box`) de grilles d'accords, chacune contenant au moins trois chorus. Il faut noter qu'il ne s'agit pas d'une comparaison générale entre ces programmes mais uniquement entre les lignes de basse qu'ils produisent. `Band-in-a-box` et `NeurSwing` sont capables de générer en temps réel des accompagnements de toute la section rythmique (basse, piano et batterie), tandis qu'`ImPact` s'occupe uniquement de la basse. En outre, `ImPact` et `NeurSwing` sont dédiés au be-bop, tandis que `Band-in-a-box` aborde une vingtaine d'autres styles musicaux. Une autre remarque importante est que les comparaisons que nous décrivons reflètent un état particulier des programmes. Chacun des trois programmes peut évoluer, soit par l'ajout de nouveaux fragments mélodiques (`Band-in-a-box` et `ImPact`) ou rythmiques (`NeurSwing`), soit enfin par l'ajout de règles (`ImPact`), soit par d'autres raffinements.

Le tableau 7.2.1 énumère, en ordre alphabétique, huit aspects ou critères selon lesquels nous avons comparé les lignes de basse de ces trois programmes. Nous allons les détailler par la suite, mais disons tout de suite qu'il ne s'agit pas d'une liste extensive d'aspects pouvant concerner une ligne de basse. Par ailleurs, ces aspects ne sont pas tout à fait orthogonaux et, surtout, ils ne sont très certainement pas consensuels. Toutefois, nous avons trouvé qu'il serait utile de bien fixer les critères de notre analyse pour qu'elle soit la plus claire et la plus objective possible. Afin d'illustrer la discussion du tableau 7.2.1, nous montrons dans les figures 7.2.9 et 7.2.10 deux versions de ligne de basse jouées respectivement²³ par `Band-in-a-box` et `NeurSwing`. Examinons maintenant chacun des aspects énumérés dans le tableau 7.2.1.

²³ Malheureusement, nous avons eu du mal à transcrire le restant de la ligne de `NeurSwing`.

	Band-in-a-box	ImPact	NeurSwing
1) développement	**	**	*
2) diversité	*	**	***
3) fluidité des changements d'accords	**	**	***
4) richesse mélodique	***	***	*
5) richesse rythmique	**	***	*
6) rendu auditif (timbre et accentuation)	***	**	***
7) sens harmonique	**	***	*
8) swing	**	**	**

Tableau 7.2.1 - Tableau comparatif des lignes de basse de *Band-in-a-box*, *ImPact* et *NeurSwing* (***) = bien, ** = passable, * = médiocre)

7.2.2.1. Développement

Nous avons déjà parlé de l'aspect "développement" dans la section 7.2.1.4. Il s'agit de la capacité de la ligne de basse à exhiber un certain parcours, dont le plus courant est le "simple-complexe-simple". Autrement dit, il est souhaitable que le programme fasse des changements dans la ligne au fur et à mesure qu'il avance dans la grille. Il est, par exemple, souhaitable qu'il joue différemment le chorus d'improvisation par rapport à celui d'exposition du thème et que, si possible, des changements aient également lieu à l'intérieur même d'un chorus selon sa structure (sections).

Les lignes de basse de *Band-in-a-box* présentent un très bon développement en ce concerne le changement de chorus, mais le développement est moins intéressant l'intérieur d'un chorus. On peut remarquer dans la figure 7.2.9 qu'à l'intérieur d'un chorus, notamment le deuxième, les mêmes fragments mélodiques sont utilisés indistinctement. *ImPact* joue aussi bien que *Band-in-a-box* à cet égard. Nous espérons que les lignes d'*ImPact* aient un meilleur développement pour deux raisons. D'une part, les cas sont indexés, entre autres, par le chorus, la position dans la grille et la section où ils ont été joués (Cf. figure 5.5.3 et tableau 5.5.1). D'autre part, nous activons des PACTs expressément pour améliorer le développement (Cf. section 7.2.1.4). Néanmoins, puisqu'il lui manque encore des cas dans la Mémoire Musicale, il n'est pas toujours en mesure de choisir ceux qui seraient les plus adéquats vis-à-vis du développement. Par exemple, les fragments basés sur la noire des mesures 1-3, 9-11 et 21-23, bien qu'ils soient excellents du point de vue du contour mélodique, sont mal placés par rapport aux fragments qui les suivent. Quant aux lignes de *NeurSwing*, leur développement n'est pas perceptible. En les écoutant, nous avons l'impression qu'elles

n'évoluent que lorsqu'un utilisateur externe change les valeurs des trois boutons qui contrôlent NeurSwing.

Figure 7.2.9 - Ligne de basse jouée par Band-in-a-box sur les deux premiers choros de FEUILLES MORTES.

7.2.2.2. Diversité

Par diversité nous entendons le degré de “variation” ou de “surprise” d’une ligne de basse. Moins la ligne est “prévisible”, plus grande est sa diversité. Naturellement, il est toujours délicat d’établir le bon degré de diversité, car il faut aussi qu’il y ait des répétitions pour garantir une certaine homogénéité et un certain développement dans la ligne. La dialectique changement-répétition est une des questions les plus centrales et les plus difficiles en musique.

Le plus grand défaut des lignes de Band-in-a-box est justement d’être trop répétitives ou prévisibles. Ceci est bien connu par les utilisateurs de Band-in-a-box qui ont l’habitude de dire que dès les premières mesures il est facile de le reconnaître dans certains styles, dont le jazz. Ainsi, seulement deux fragments mélodiques sont joués sur

les accords mineurs du deuxième chorus de la ligne de la figure 7.2.9. Le fragment de la mesure 33 se répète de façon identique ou approximativement (léger changement à la dernière note), 4 fois et celui de la mesure 39, se répète 5 fois. Le passage des mesures 39-41 est particulièrement mauvais car il y a une répétition triple de ce dernier fragment. Ce phénomène de répétition se produit sur pratiquement tous les accords de la grille mais il est plus gênant dans le deuxième chorus, notamment sur ceux de dominante, car pendant l'exposition du thème on peut se permettre de répéter davantage. Ce qui est plus grave dans les lignes de `Band-in-a-box` est le fait que l'on retrouve les mêmes fragments dans d'autres grilles de jazz. Ceci s'explique assez facilement car `Band-in-a-box` n'a visiblement pas un nombre important de fragments mélodiques pour le jazz. Nous n'avons pas les données précises, mais il semblerait qu'ils soient environ au nombre de quarante. En outre, puisque la segmentation de la grille chez `Band-in-a-box` se fait accord par accord, il est plus probable qu'il retrouve des situations (segments) semblables même dans des grilles différentes .

The figure displays two screenshots of the 'autumn NeurSwing' software interface. The left screenshot shows the first chorus of the piece, with measures 1 through 29. The right screenshot shows the second chorus, with measures 33 through 61. The interface includes a menu bar with options like 'File', 'Insert notes', 'Global Edit', 'Browse', 'MIDI', 'Refresh', and 'Various'. The notation is in bass clef with various chords and melodic lines.

Figure 7.2.10 - Ligne de basse jouée par `NeurSwing` sur le premier chorus et partie du deuxième chorus de *FEUILLES MORTES*. Les boutons *hot/cool*, *dissonance/consonance* et *free/as-is-ness* sont tous à 0.5 (moitié).

La diversité des lignes d'ImPact est clairement plus grande que celles de Band-in-a-box. Le fait d'avoir un nombre plus grand de fragments mélodiques stockés et de segmenter la grille en schémas d'accords minimisent le degré de répétition, du moins entre différentes grilles. Par exemple, parmi les grilles que nous avons présentées dans ce chapitre, le schéma II-V mineur où chaque accord dure huit temps n'apparaît que dans ALL OF ME. Cela dit, les lignes d'ImPact sont encore très répétitives notamment lorsqu'il s'agit de jouer sur la blanche. Ainsi, le fragment des mesures 5 et 6 de la figure 7.2.5 est répété trois fois dans le même chorus. Même dans les passages joués sur la noire, il se produit parfois des répétitions un peu abusives, comme la triple répétition (non-consécutive) du fragment des mesures 12-13 de BLUESETTE (Cf. figure 7.2.2).

Les lignes de NeurSwing sont celles qui présentent la plus grande diversité du fait de plusieurs "surprises harmoniques". Ces "surprises" sont des phrases très inhabituelles créées grâce à la liberté que NeurSwing prend par rapport à la grille. Par exemple, dans la grille de Feuilles MORTES (Cf. figure 7.2.10), on peut identifier la phrase "pianistique" de la mesure 2, celle "guitarristique" de la mesure 34, et encore celle de la mesure 13-14. En outre, il y a des choix de notes qui sont très audacieux, notamment pour la première note de l'accord. Cette "prise de risque" harmonique est équilibrée par un rythme simple et un contour mélodique pauvre, en garantissant une certaine homogénéité à la ligne.

7.2.2.3. *Fluidité*

Comme on dit, il faut que la ligne de basse "coule", que l'on ne sente pas qu'il a des ruptures (sauts) à chaque fois que l'accord de la grille change, et que l'on puisse anticiper sur l'accord suivant. Pour cela, il est en général bon que la dernière note jouée sur un accord soit assez proche de la première note du prochain accord, l'exception étant notamment les situations où le saut entre ces deux notes est d'un intervalle consonant, comme l'octave et la quinte juste. En plus, il est souhaitable que ces deux notes ne soient pas exactement les mêmes car ceci ne contribuerait pas à créer l'attente nécessaire vis-à-vis de l'accord suivant. C'est pourquoi, les bassistes font souvent usage du *leading tone*, la note juste au-dessus ou au-dessous de la première note de l'accord subséquent. Plus les transitions sont bien préparées, plus on a la sensation que la ligne est fluide. Naturellement, la fluidité dépend aussi de ce qui se joue pendant la durée

d'un accord. En général, il n'est pas intéressant qu'il y ait des longs passages avec beaucoup de sauts, sauf évidemment lorsqu'il s'agit de l'exposition du thème, des morceaux lents, etc.

A cet égard, la fluidité des lignes de *NeurSwing* est excellente. Les transitions entre les accords sont très bonnes et la ligne, lorsque *NeurSwing* joue en *stepwise* (comme c'est le cas de FEUILLES MORTES), est évidemment très fluide. Quant aux lignes de *Band-in-a-box*, elles n'ont pas une très bonne fluidité. Par exemple, dans FEUILLES MORTES, les transitions²⁴ 34-35, 41-42 et 62-63 pourraient être meilleures et les transitions 39-40, 47-48, 49-50, 51-52 devraient absolument l'être. Ceci peut s'expliquer par le fait que *Band-in-a-box* joue toujours la tonique au premier temps. Du moment qu'une autre note d'accord peut-être choisie au premier temps, les possibilités d'une bonne transition augmentent sensiblement. Dans les lignes d'*ImPact* les transitions entre les accords à l'intérieur d'un fragment sont évidemment bien faites, elles étant définies par Ron Carter. Mais elles ne le sont pas toujours pour la transition entre les fragments (inter-schémas). Dans FEUILLES MORTES, les transitions sont bonnes à l'exception des 16-17 et 58-59 qui pourraient être améliorées et des 26-27 et 44-45 qui sont vraiment mauvaises. Dans ALL OF ME de la figure 7.2.1 (droite), les transitions 2-3 et 12-13 ne sont pas tout à fait réussies. Toute la difficulté d'*ImPact* et *Band-in-a-box* d'obtenir des lignes fluides tient du fait qu'il est difficile de "coller" des phrases préalablement stockées. *NeurSwing* s'en sort mieux car il construit la ligne note par note.

7.2.2.4. *Richesse mélodique*

Par richesse mélodique nous entendons la diversité de contours mélodiques : des montées longues, courtes, descentes longues, des *drops* ou des sauts d'octave, des arpèges, des passages en *stepwise*, etc.

Visiblement ce qui contribue à la fluidité des lignes de *NeurSwing*, nuit à a richesse mélodique dans pratiquement tous les exemples que nous avons écoutés. Dans FEUILLES MORTES, tout est joué en *stepwise* et certains passages sont vraiment plats

²⁴ Les transitions entre respectivement le dernier accord et le premier accord des mesures citées

(par ex. mesures 1, 4-5, 10, 18-19, 22-23 et 26). C'est assez ennuyeux. *Band-in-a-box* produit des lignes de basse ayant des contours mélodiques plus intéressants. Par exemple, les contours des mesures 33-44 sont diversifiés et s'enchaînent bien. Toutefois, la richesse mélodique des lignes de *Band-in-a-box* n'est pas excellente car elle souffre du fait qu'il y a trop souvent de répétitions des fragments. Une certaine tendance à une montée lente pendant les carrures du deuxième chorus de FEUILLES MORTES en témoigne. Les lignes générées par *ImPact* sont du point de vue mélodique un peu plus riches que celles de *Band-in-a-box*. Dans FEUILLES MORTES, par exemple, il y a différentes montées et descentes, l'emploi de la technique de notes répétées (mesures 37-39, 57-59), des *drops* ou sauts d'octave (mesures 42, 54 et 60) et un bon équilibre entre arpèges et passages en *stepwise*.

7.2.2.5. *Richesse rythmique*

La richesse rythmique d'une ligne dépend de la diversité de figures rythmiques utilisées.

Les lignes de *NeurSwing* ont des rythmes assez pauvres, car *NeurSwing* ne produit que des lignes basées sur la noire, où il ajoute de temps en temps des appoggiatures. Il n'y a ni silences, ni notes blanches, ni notes pointées, etc. C'est sans doute pourquoi, *NeurSwing* ne se débrouille pas très bien dans les tempos lents. *Band-in-a-box* s'en sort nettement mieux, surtout dans le chorus où il joue des blanches. En outre, dans les lignes de *Band-in-a-box*, on trouve des *pull-downs* (par ex. mesures 53 et 56 de FEUILLES MORTES) et des double-croches très intéressantes (par ex. mesures 27-28). Comme pour la richesse mélodique, la rythmique souffre du fait qu'il y a beaucoup trop de répétitions de phrases dans *Band-in-a-box*. Les lignes d'*ImPact* sont, quant à elles, rythmiquement les plus riches des trois programmes. Elles comprennent toutes les figures utilisées dans *Band-in-a-box* plus d'autres. Les exemples les plus frappants sont ceux des mesures 17-24 de la figure 7.2.6 (gauche) et des mesures 21-28 de la figure 7.2.7, où *ImPact* tente de jouer "hot".

7.2.2.6. *Rendu auditif*

Par rendu auditif nous entendons le timbre, l'accentuation et tous les autres aspects liées à l'écoute même des lignes de basse générées par ces systèmes.

`Band-in-a-box` et `NeurSwing` se révèlent meilleurs qu'`ImPact` à cet égard. `NeurSwing` apporte vraiment beaucoup de soin au timbre. C'est pourquoi il a été spécialement programmé pour un synthétiseur en particulier. Le plus frappant quand on écoute `Band-in-a-box` est l'accentuation de chaque note. Le résultat sonore de ces deux programmes est très bon, bien qu'ils ne puissent pas reproduire exactement les modes de jeux (glissandos, notes fantômes, *hammer on*, *pull off*, etc.) des bassistes humains. Comme nous l'avons dit dans la section 5.5.2, nous n'avons pas pu incorporé l'information d'amplitude dans les fragments mélodiques que nous avons acquis pour constituer la Mémoire Musicale. C'est qui explique que les résultats sonores soient moins bons que ceux des deux autres programmes.

7.2.2.7. *Sens harmonique*

Une ligne de basse a du sens harmonique si elle offre un bon équilibre entre deux exigences. D'un côté, le besoin de souligner clairement l'harmonie représentée par les accords de la grille. En effet, un des rôles principaux de la basse est justement de donner un support harmonique aux solistes. D'un autre côté, la nécessité de "prendre des risques" par rapport à l'harmonie pour enrichir la ligne. En général, plus le musicien est expérimenté, plus il sait quels sont les risques qu'il peut prendre, quelles notes n'appartenant pas à l'accord sous-jacent il peut jouer, etc.

Les lignes de basse de `Band-in-a-box` sont très correctes mais pauvres harmoniquement. Les toniques des accords sont toujours jouées au premier temps. Certes, il y a des passages (courts) un peu chromatiques ou dissonants, mais il s'agit toujours des mêmes fragments mélodiques comme celui de la mesure 38 de la figure 7.2.9. On n'entend pas vraiment des passages un peu inhabituels du point de vue harmonique. On pourrait dire qu'à l'égard de l'harmonie `Band-in-a-box` joue comme un musicien débutant. C'est tout à fait acceptable et correct mais c'est souvent élémentaire.

`NeurSwing` s'oppose totalement à `Band-in-a-box`. Les lignes de `NeurSwing` sont d'une énorme diversité harmonique et c'est cela qui fait à la fois sa vertu, comme nous l'avons exposé dans la section 7.2.2.2, et son défaut. Par exemple, `NeurSwing` ne joue la tonique au premier temps qu'une fois sur quatre en moyenne, sauf, évidemment, si le bouton "dissonant/consonant" est positionné au plus consonant possible. Dans ce

cas, il joue toujours la tonique, puis la quinte, puis la tierce et de nouveau la quinte. Ne pas jouer fréquemment la tonique au premier temps n'est pas grave, si la note choisie est la tierce ou la quinte de l'accord. Mais, à force de prendre des risques *NeurSwing* fait des fautes ou joue des phrases qui ne sont pas "convenables" pour la basse. Prenons par exemple la ligne de la figure 7.2.10. Les passages des mesures 4, 10, 12, 14, 20 et 34 sonnent faux, compte tenu du style visé. Dans la mesure 10, par exemple, *NeurSwing* commence avec la quatrième augmentée, et après avoir joué la tierce, il revient encore deux fois sur la quatrième augmentée. Plus grave encore, dans la mesure 25, il y a une faute d'harmonie. On ne peut jouer la tierce majeure sur un accord demi-diminué, sauf en tant que note de passage ou si la tierce mineure a été déjà jouée. Commencer avec la tierce majeure et revenir dessus n'est pas correct. Selon, Antoine Espagno, il vaut mieux jouer des choses simples, comme les lignes de *Band-in-a-box*, que prendre des risques trop qui conduisent à des fautes.

ImPact génère de lignes qui sont plus complexes harmoniquement que celles de *Band-in-a-box*, tout en évitant de prendre certains risques. En réalité, *ImPact* lui-même ne vérifie pas le sens harmonique de chaque note d'un fragment mélodique qu'il récupère. Si jamais il y a des fautes, elles ne seront pas détectées. Il se trouve simplement que les exemples de lignes de basse à partir desquels nous avons construit la Mémoire Musicale ne contiennent pas de fautes.

7.2.2.8. *Swing*

Il est difficile de parler de swing car, non seulement il s'agit d'un aspect musical dont la perception est très personnelle, mais de plus nous ne pouvons pas l'illustrer à l'aide de partitions comme nous l'avons fait jusqu'à présent.

En écoutant les lignes de basse générées par ces trois programmes, nous n'avons pas eu la nette impression que l'un d'entre eux swinguait vraiment, bien que ce qu'ils jouaient ne soit pas "carré". Nous trouvons que *NeurSwing* swingue un peu plus que les autres mais la différence n'est pas grande.

7.2.2.9. *Bilan*

Il est certes facile de remarquer la différence ce que `ImPact`, `Band-in-a-box` et `NeurSwing` jouent et les lignes générées par d'autres programmes comme ceux de Levitt (Levitt, 1993), Johnson-Laird (Johnson-Laird, 1991) ou Ames (Ames & Domino, 1992). Toutefois, la différence entre ces trois premiers programmes est assez serrée. Antoine Espagno souligne que notre programme joue la basse globalement mieux que les autres deux. D'après nos critères musicaux *personnels*, nous croyons aussi que `ImPact` joue légèrement mieux. Néanmoins, scientifiquement parlant, il est extrêmement difficile de définir une méthode d'évaluation et des critères suffisamment précis et objectifs pour prouver qu'un des ces programmes joue mieux, ou un peu mieux, que les autres. Une telle évaluation est d'autant plus délicate qu'un programme peut être meilleur pour un aspect et mauvais pour un autre, en l'occurrence c'est souvent ce qui se passe. Un jugement global n'est pas évident à établir car l'importance relative de chaque aspect n'est formalisée nulle part. Par exemple, Antoine Espagno trouve que les "fautes d'harmonie" de `NeurSwing` le discrédite par rapport aux deux autres programmes. Mais, c'est une opinion tout à fait personnelle. Chaque musicien prête une importance différente aux aspects du tableau 7.2.1.

Les comparaisons que nous avons faites nous ont montré les principales faiblesses et vertus musicales d'`ImPact` et les directions d'un travail futur. Nous croyons que le développement et la diversité des lignes de basse produites par `ImPact` peuvent être améliorées par l'ajout de nouveaux cas dans la Mémoire Musicale, surtout des cas joués sur la blanche qui ne représentent à l'heure actuelle que 12 % du total. En ce qui concerne la fluidité, encore insatisfaisante, des lignes d'`ImPact`, elle pourrait être améliorée en augmentant la capacité du module d'exécution à changer les dernières notes des fragments mélodiques calculés (Cf. section 4.2). Pour l'instant les changements se limitent au cas où la dernière note d'un accord est égale à la première de l'accord suivant. Ce problème reflète, en réalité, un besoin que nous avons ressenti de donner à `ImPact` plus de connaissances au niveau des notes. Le raisonnement d'`ImPact` se situe, en effet, plus au niveau des traits musicaux d'un fragment mélodique qu'au niveau des notes qui composent ce fragment. Pour ce qui est de la faiblesse du rendu auditif, en particulier des accentuations, nous sommes en train de mettre en place un projet, en collaboration avec Boris Doval du LIMSI, pour la saisie automatique des

lignes de basse dans un enregistrement monophonique. Cela nous permettra de compléter la description des cas de la Mémoire Musicale en y ajoutant des informations précises sur les amplitudes des notes.

7.2.3. Comparaison avec des musiciens humains

La figure 7.2.11 montre les deux premiers chorus d'une ligne de basse jouée par Tyrone Wheeler sur FEUILLES MORTES. Un jour nous avons donné cette ligne à Antoine Espagno pour qu'il l'évalue sans lui dire qu'il s'agissait d'une ligne jouée par un être humain. Lorsqu'il l'a fini de la jouer, il a déclaré : "le meilleur programme est sans aucun doute celui qui a joué cette ligne". Sous tous les points de vue énumérés dans la section précédente, cette ligne de basse est, en effet, meilleure que celles produites par les trois programmes étudiés. Elle a un développement clair mais non fastidieux et elle a une grande diversité avec quelques surprises simples et heureuses (par ex. l'harmonique — Mi 4 — joué à la fin de la mesure 31). C'est aussi une ligne fluide et très riche du point de vue de la mélodie et du rythme. Bien entendu le sens harmonique, le swing et le son sont très bons.

Nous avons montré les défauts et les vertus des lignes de basse d'ImPact à l'égard de celles de *Band-in-a-box* et de *NeurSwing*. Nous avons aussi conclu que ces trois programmes jouent aussi bien les uns que les autres. Dans cette section il s'agit d'examiner plus en détail le "niveau musical" des lignes de basse d'ImPact vis-à-vis de ce que les bassistes de jazz sont capables de jouer.

ImPact joue à toute évidence mieux qu'un musicien débutant. Les musiciens qui l'ont écouté trouvent qu'il joue correctement et que de temps en temps il sort des bonnes phrases. Ainsi, Antoine Espagno a dit, à plusieurs reprises, qu'il aurait joué les mêmes phrases qu'ImPact dans tel ou tel moment de la grille (par ex. début du deuxième chorus de FEUILLES MORTES — figure 7.2.5). Personnellement, ImPact nous a beaucoup surpris. Nous pensions qu'il jouerait moins bien. Cela dit, ImPact est actuellement loin de jouer aussi bien qu'un bassiste expérimenté comme Tyrone Wheeler. Outre les imperfections signalées précédemment et d'autres défauts (il ne joue pas bien dans les tempos inférieurs à 120 à la noire, il ne joue pas aux temps ternaires, etc.) ImPact a un grand problème : il est incapable de se rendre compte qu'il a joué bien ou mauvais. Par conséquent, il ne peut décider de développer ou d'éviter une telle

phrase, ce qui nuit au développement et à la diversité des lignes de basse générées. Cette capacité de porter dynamiquement un jugement sur ce que l'on vient de jouer est très importante en jazz, comme nous l'avons déjà souligné dans la section 2.2. Au-delà des mauvaises conséquences qu'un tel manque d'auto-évaluation peut engendrer dans un morceau en particulier, le plus grave est que sans savoir s'auto-évaluer un programme ne pourra vraisemblablement pas développer un style, évoluer et innover. Bien que l'on ne sache pas très bien ce que ça veut dire "innover", on peut difficilement imaginer que les parcours des musiciens qui ont poussé très loin les frontières du jazz soient dus au hasard. À notre avis, la capacité de porter un jugement sur ce que l'on a fait soi-même et ce que les autres jazzmen ont fait jusqu'à une époque donnée est partie indissociable de la démarche d'évolution d'un jazzman. De temps en temps *ImPact* joue de phrases intéressantes ou bien placées, mais il ne s'en rend pas compte. C'est pourquoi, même si nous le faisons fonctionner un million de fois, à la fin il n'aura rien compris de nouveau sur le jazz ou sur sa propre façon de jouer.

The image displays two screenshots of a music software interface for the piece "Autumn Leaves" by Tyrone Wheeler. The interface includes a menu bar with options: File, Insert notes, Global Edit, Browse, MIDI, Refresh, and Various. The main area shows the bass line for the first two choruses, with measures 1-29 on the left and measures 33-61 on the right. Each measure is accompanied by a chord symbol, such as C min 7, F 7, Bb maj7, Eb maj7, A half Dim7, D 7, G min 7, G min 7, C 7 9, F min 7, Bb 7, Eb maj7, A half Dim7 aug 7, and G min 7.

Figure 7.2.11 - Ligne de basse jouée sur les deux premiers choros de *FEUILLES MORTES* par Tyrone Wheeler (AABERSOLD, 1993).

Cette critique peut naturellement s'adresser à tous les autres programmes actuellement conçus pour jouer du jazz. Même au-delà de la musique, les programmes produisant des œuvres d'arts ont du mal à évaluer ce qu'ils ont fait ou de reconnaître une œuvre d'art "réussie" ou "créative" (Elton, 1993). Les contraintes ou les règles de choix reflétant une préférence esthétique sont données préalablement et elles n'évoluent guère. Une évaluation a posteriori s'impose. En effet, les programmes ne peuvent prévoir exactement ce qu'ils vont créer, soit parce qu'il y a des choix aléatoires, soit que le contrôle des détails de l'œuvre générée (notes dans les cas de la musique) n'est pas suffisamment contraint, soit encore que les choix du programme impliquent des raisonnements très complexes. Afin que le programme soit capable d'évoluer, de raffiner ou de changer sa façon de créer, il serait souhaitable qu'il ait plus de "conscience" de ce qu'il fait.

CONCLUSIONS ET PERSPECTIVES

RESULTATS ET CONTRIBUTIONS

Nous avons présenté dans ce mémoire un travail dont le but était de mieux comprendre les possibilités de construction d'une "machine à jouer du jazz", un problème dont nous avons montré toute la difficulté. À travers le jazz, nous espérons examiner certaines hypothèses de travail centrales en IA et, surtout, nous investir dans la problématique de la simulation des comportements "créatifs" sur des machines.

Pour ce faire, nous avons conçu un modèle informatique original d'agent rationnel que nous avons appliqué à la tâche d'accompagnement d'un bassiste de jazz. Selon ce modèle la tâche d'improvisation ou d'accompagnement d'un agent jazzman se déroule en trois étapes qui se succèdent continuellement du début à la fin de la grille d'accords donnée. Premièrement, à travers une analyse harmonique particulière, il établit le "segment" de la grille pour lequel l'agent va calculer les notes. Deuxièmement, il détermine un ensemble de PACTs (actions potentielles comme "jouer peu dissonant", "jouer beaucoup de notes", "jouer tessiture très haute") concernant les notes du segment. Les PACTs peuvent, en effet, être vues comme des "consignes musicales" portant sur un ou plusieurs "traits musicaux" (par ex. dissonance, densité, tessiture, etc.) et ayant un début et une durée. Elles sont d'abord activées selon les données perceptives de l'agent (la grille, ce que les autres musiciens jouent, ce que l'agent vient de jouer, etc.). Ensuite, celles qui se superposent au segment en question sont assemblées afin d'obtenir l'ensemble le plus complet et le moins contradictoire de PACTs. Troisièmement, il récupère de la Mémoire Musicale, contenant des fragments de lignes de basse jouées par un bassiste humain, le fragment qui respecte le mieux les consignes (PACTs) données. Dans les premières étapes nous faisons appel à un raisonnement déductif, implémenté à travers des règles de production couplées à un moteur d'inférence d'ordre un en chaînage avant. Dans la dernière étape, nous nous servons de techniques de raisonnement à partir de cas.

Afin de rendre opérationnel ce modèle, nous avons implémenté le système *ImPact*, ce qui a d'ailleurs représenté un effort non négligeable de programmation. Nous avons pu grâce à *ImPact* évaluer et raffiner les différents aspects du modèle, ainsi que comparer les résultats musicaux obtenus avec ceux des programmes actuels. Les résultats musicaux que nous avons obtenus sont globalement aussi bons que ceux des deux "meilleurs" programmes pouvant générer des lignes de basse. En outre, nous avons constaté qu'il était facile d'ajouter des nouvelles connaissances sous la forme de règles ou de cas faisant ainsi évoluer le système. Ceci étant, il y a encore un grand écart entre les lignes de basse d'*ImPact* et celles d'un bassiste expérimenté, que le simple ajout de règles ou de cas ne suffira pas à combler. C'est pourquoi plusieurs perspectives de recherche s'ouvrent pour le futur. Nous les énumérons à la fin de ce chapitre.

Plus précisément, le travail que nous avons réalisé a apporté trois contributions principales. La première contribution, et aussi la plus générale, est notre modèle en tant que tel. Ce modèle permet d'intégrer de façon originale et assez simple de nombreux types de *connaissances* et modes de raisonnement allant de simples réactions à l'environnement jusqu'à l'analyse harmonique. L'intégration la plus originale est celle du raisonnement à partir de cas et le raisonnement déductif. Notre travail apporte donc un éclairage sur une controverse célèbre concernant l'adéquation du raisonnement "guidé par des règles" par opposition à celui "guidé par exemples" dans la simulation des activités de composition musicale au sens large (Laske, 1989; Smoliar, 1990). Nous montrons l'importance de combiner ces deux modes de raisonnement. Ce besoin d'intégration est d'ailleurs ressenti de plus en plus dans la modélisation de différentes tâches de conception, comme en témoignent notamment les travaux récents de Strube (Strube et al., 1995) ou les chercheurs travaillant sur le projet allemand FABEL (Gebhardt et al., 1996).

La deuxième contribution concerne la notion de *Mémoire Musicale*. L'idée de réutiliser des fragments mélodiques ou rythmiques est utilisée dans d'autres programmes (Baggi, 1992; Band-in-a-box, 1995; Hodgson, 1996). Toutefois, la richesse avec laquelle nous indexons ces fragments et la finesse des mécanismes qui contrôlent leur récupération constitue une réelle contribution dans le domaine de l'informatique musicale. En outre, nous pensons que pour ce qui est du problème spécifique de "l'interprétation de situation" (Cf. section 4.3.4.2), notre solution peut, au-delà de la

musique, avoir un intérêt dans le domaine du “raisonnement à partir de cas continu” (Ram & Santamaria, 1993). Nous décrivons cette possible extension dans la prochaine section.

La troisième contribution, cette fois-ci dans le domaine de la représentation des connaissances musicales, est relative à l’enrichissement et à la formalisation de la notion de PACTs. A partir de la formulation initiale de F. Pachet (Pachet, 1987; Pachet, 1990), nous avons mené un travail considérable sur l’élargissement de la notion de PACTs, la détermination d’une ontologie de PACTs concernant la basse, l’insertion des PACTs dans une méthode de résolution de problèmes bien définie, et la formalisation des opérations les impliquant.

QUELQUES REFLEXIONS

Dans l’introduction de ce mémoire, nous avons posé un certain nombre de questions qui concernaient, d’une manière plus générale, l’importance de la recherche en IA dans des domaines artistiques (Cf. section A.b, page 4) ainsi que notre problématique spécifique (Cf. section B.b, page 15). Nous ne sommes pas en mesure de répondre à toutes ces questions, certaines d’entre elles définissant en effet tout un champ de recherche. Néanmoins, nous pouvons y apporter quelques éléments de réponses.

Généralité du modèle

Une première question importante concerne le degré de généralité de notre modèle. Ce qui nous semble essentiel dans notre modèle est l’idée que c’est l’artiste qui à la foi détermine les consignes que l’œuvre doit respecter et que concrétise l’œuvre. Autrement dit, l’artiste, jouissant de toute sa liberté, pose et résout le problème. Lorsque notre agent active des PACTs, il est en train de poser le problème car il “spécifie” les consignes auxquels l’œuvre doit respecter. Ensuite, il essaye de trouver une solution correspondante à partir de son expérience codée sous la forme de cas vécus. Notre modèle permet de représenter et concrétiser sous forme de notes ces consignes, même s’elles sont partielles, un peu contradictoires ou très complexes. Avec toutes les adaptations dues, nous pensons que ce modèle pourrait être appliqué à d’autres tâches artistiques et de conception. La seule condition est que des cas jouent un rôle important

et qu'il existe des connaissances permettant de mieux "spécifier" l'œuvre à créer ou la conception à réaliser.

Dans le domaine du raisonnement à partir de cas, notre approche a des possibilités d'extension intéressantes concernant la problématique de "l'interprétation de situation", en particulier pour le "raisonnement à partir de cas continu". Dès que le déroulement dans le temps du phénomène à modéliser doit être considéré (par ex. l'activité d'un robot autonome, la propagation d'un incendie de forêt ou l'évolution de la santé d'un patient), le choix d'index pertinents devient critique. En effet, il est impossible de prendre en compte la description complète du passé. Les mécanismes d'activation puis d'assemblage des PACTs peuvent être utiles à cet égard car ils permettent de tenir compte indirectement de tout le passé grâce à la dimension temporelle des PACTs. Par ailleurs, indépendamment de la question du déroulement du temps, les processus d'activation et d'assemblage des PACTs permettent de raffiner la requête posée à la base de cas. Chaque PACT représente une consigne supplémentaire sur la solution à être récupérée de la base. Pour les tâches où il existe des connaissances permettant de spécifier, même de façon partielle, les caractéristiques de la solution recherchée, le cadre que nous proposons peut être appliqué.

Dans le domaine musical, nous pensons que le modèle que nous proposons peut s'étendre très facilement à d'autres instruments (batterie, piano, percussion, cuivres, guitare, etc.) et à d'autres styles musicaux (bossa nova, reggae, salsa, rock, etc.). De fait, lorsqu'il s'agit d'activités musicales se déroulant en direct et ayant une grille d'accords comme guide, notre modèle peut s'appliquer directement. D'ailleurs, la représentation des PACTs ainsi que toutes les autres composantes de notre programme ont été conçues avec le souci préalable de faciliter une réutilisation et adaptation à d'autres instruments et styles. Naturellement, un travail d'acquisition de cas et d'acquisition de connaissances, visant l'identification des "traits musicaux" pertinents et les règles d'activation de PACTs, serait nécessaire pour chaque instrument dans chacun des styles.

Hypothèses de travail revisitées

D'autres questions concernent la pertinence des trois hypothèses de travail (Cf. section B.b) qui nous ont guidés pendant ce projet de thèse.

Nous avons pu constater que jouer du jazz est un véritable métier. C'est pourquoi *le principe de la connaissance* s'est avéré pertinent non seulement pour notre projet mais aussi pour les autres programmes conçus pour jouer du jazz. Les programmes qui ont pu incorporer le plus de connaissances sont ceux qui produisent les meilleurs résultats.

Contrairement à certaines idées (Johnson-Laird, 1991; Pressing, 1988), nous avons pu assimiler les tâches d'improvisation et d'accompagnement à un modèle de *résolution de problèmes*, que nous avons ensuite implémenté et testé. Les difficultés inhérentes qui écartaient a priori les techniques de résolution de problèmes, à savoir le manque de buts précis et figés et le nombre important de choix possibles, ont donc pu être contournées.

Pour ce qui est de l'*hypothèse de systèmes symboliques et physiques*, notre travail renforce son deuxième volet : ces systèmes disposent des moyens nécessaires pour exhiber un comportement intelligent. D'autre part, comme nous l'espérons, l'adoption d'un cadre symbolique puis de celui de la résolution de problèmes a contribué à une meilleure compréhensibilité des principes sous-jacents à notre programme. Nous pouvons toutefois critiquer un certain manque de clarté lié aux mesures numériques de similarité et d'adaptabilité employées pour la récupération des cas. Parfois, il n'est pas évident d'expliquer le choix spécifique d'un cas compte tenu de nombre important d'attributs décrivant les cas et des pondérations associées à ces attributs. Comme nous l'avons exposé dans la section 7.2.1.4, nous ne sommes en effet jamais sûr qu'un critère musical (représenté par un attribut) en particulier sera respecté.

Créativité

Bien que notre travail soit surtout centré sur la modélisation des connaissances mises en jeu dans le jazz, la problématique de la construction des programmes créatifs en demeure la toile de fond. Il est donc important de comprendre quel est l'éclairage que notre travail peut lui apporter.

Notre modèle renforce l'opinion que la simulation de "comportements créatifs" sur une machine passe par la réutilisation d'idées, et plus précisément des épisodes vécus dans le passé. C'est le principe du modèle de Schank (Ganascia, 1990; Schank, 1990). *ImPact* crée des lignes de basse qui n'ont été jamais jouées (note par note)

auparavant et qui sont acceptables du point de vue esthétique. À cet égard, *ImPact* est doté d'une capacité de création musicale. Néanmoins, nous ne pouvons pas vraiment dire qu'*ImPact* est "créatif", au sens où l'on l'entend dans le milieu artistique, car ses lignes de basse n'apportent aucune innovation à ce que les bassistes de jazz jouent actuellement. Cette constatation, ainsi que toutes les réflexions que nous avons menées pendant notre travail sur le jazz, nous conduisent à soulever deux questions qui ont été jusqu'à présent négligées dans tous les "modèles de créativité" (AAAI, 1993; Dartnall, 1994; Hofstadter, 1993; Rowe & Partridge, 1993; Schank, 1990). Ces questions montrent que la problématique de la créativité est beaucoup plus complexe que ces modèles ne le laissent croire.

La première concerne "l'échelle de temps de la créativité". Dans le contexte du jazz, ceci revient à se demander si la "créativité" se manifeste dans une phrase mélodique, un morceau, un concert, une tournée ou la vie entière. Il est vraisemblable que les jazzmen ne cherchent pas à jouer des phrases "innovatrices" ou "inattendues" à tous les instants (Sloboda, 1985) [p. 149], d'autant que ceci pourrait gêner les autres musiciens. En outre, bien qu'ici ou là on puisse entendre des "idées innovatrices", elles semblent être inscrites dans une démarche plus longue d'expérimentation, de réflexion, de raffinement, d'exploration, etc. Si l'on veut construire un "programme créatif", il est donc important de savoir quand il doit manifester sa "créativité" et de savoir la faire progresser. Cette dimension temporelle est toutefois absente des "modèles de créativité" existants. Ils supposent que le programme doit toujours trouver une "solution créative" à chaque problème posé sans proposer une perspective d'évolution globale. On pourra noter l'exception des travaux de Lenat sur AM et Eurisko (Lenat & Brown, 1984). Ceci n'a rien d'étonnant car, comme Phil Agre (Agre, 1995) a bien souligné, il est difficile en IA de proposer des modèles pour accomplir des tâches à long terme (par ex. vivre, éduquer un enfant, faire une thèse de doctorat, etc.).

La deuxième question est de savoir s'il est possible de concevoir des programmes qui génèrent des "solutions créatives", sans pour autant aborder la problématique de savoir ce qu'est une "solution créative". Actuellement, pour éviter d'affronter cette problématique majeure, les modèles de créativité proposent une définition "opératoire" de la créativité : c'est le processus de génération qui détermine si la solution est créative. Ainsi, selon Schank, un programme a un comportement créatif

lorsqu'il adapte une solution connue face à un problème nouveau. De même, Boden affirme qu'un programme est créatif lorsque, pour résoudre un problème, il doit effectuer un changement dans son "espace conceptuel", c'est-à-dire remettre en cause une règle, relaxer une contrainte, etc. Cependant, rien ne garantit que l'adaptation d'une solution connue ou la remise en cause d'une règle mène forcément à une solution "créative". La solution peut se révéler tout à fait médiocre.

La problématique de la reconnaissance de ce qui est créatif est intimement liée à la question de la dimension temporelle de la créativité que nous avons évoquée dans cette section. En effet, si le programme ne sait pas évaluer la solution générée, comment pourrait-il entreprendre une vraie démarche de création ? Sans cette capacité d'évaluation, le programme ne peut proposer des solutions créatives que *par hasard*. C'est ce qui se passe actuellement avec *ImPact* et les autres programmes de jazz, ou avec les générateurs d'œuvres d'arts en général : ils ne sont pas *conscients* de ce qu'ils créent (Cf. section 7.2.3). Bien entendu, il s'agit d'une problématique extrêmement difficile. Qualifier une œuvre d'art d'innovatrice présuppose, d'une part, une certaine connaissance de ce qui a été créé jusqu'à l'heure actuelle et, d'autre part, une maîtrise des possibilités d'expression propres à la forme d'art en question. Cependant, nous croyons que cette tâche peut être automatisée lorsqu'il s'agit d'un instrument et d'un style particulier. De fait, il n'existe pas en IA d'algorithme général de reconnaissance d'une "solution intelligente", mais il est possible de définir un certain nombre de critères pour évaluer la qualité de la solution pour une tâche bien précise. Ainsi, il est souhaitable qu'une preuve d'un théorème soit correcte, courte, intelligible, etc. Nous pensons que la "reconnaissance de la créativité" est un problème incontournable et qu'il constitue le plus grand défi pour la construction de "machines créatives".

PERSPECTIVES DE RECHERCHE

Outre les améliorations du système *ImPact* énumérées dans le dernier chapitre de ce mémoire, les travaux que nous avons développés pendant ces quatre années de thèse nous ouvrent plusieurs perspectives de travail. Ces différentes perspectives peuvent être groupées dans quatre axes de recherche que nous décrivons par la suite.

Extension à d'autres styles et à d'autres instruments

Parmi les perspectives les plus naturelles, nous comptons appliquer le même modèle à d'autres styles et à d'autres instruments. L'idée d'étendre *ImPact* à divers styles, instruments et combinaisons d'instruments, nous paraît très importante à deux égards. D'une part, ceci nous permettra de valider la généralité du modèle que nous proposons et de montrer quelles sont ses faiblesses et ses points forts. D'autre part, nous pourrons, en faisant jouer plusieurs instruments ensemble (par ex. basse, batterie et piano), étudier l'interaction entre les agents responsables de chaque instrument. Puisque chaque agent peut à tout moment informer les autres agents de ce qu'il est en train de faire, nous n'avons pas besoin d'un Scénario figé ni d'un vrai module de perception. Nous pourrons alors traiter directement les problématiques des modalités de communication et des conflits personnel-collectif.

Les possibilités de styles, d'instruments et de formations instrumentales (groupe d'instruments) sont très nombreuses, ce qui nous permet d'envisager plusieurs années de recherche sur ce thème. Actuellement, le projet de conception de deux systèmes jouant respectivement de la guitare dans le style de bossa nova et plusieurs instruments de percussion dans divers styles musicaux est en cours avec l'Université de Pernambuco au Brésil.

Interaction avec les musiciens humains

Nous nous sommes surtout intéressés aux mécanismes de raisonnement de l'agent musical. Nous avons réussi à éviter les problèmes liés à l'automatisation de la perception en utilisant la notion de Scénario. Le Scénario a été une simplification utile mais si nous voulons une interaction avec des musiciens humains, il va nous falloir travailler dans deux directions.

D'abord, nous comptons développer des interfaces permettant aux musiciens d'indiquer à *ImPact*, d'une part, les caractéristiques (selon un répertoire limité de traits musicaux) de ce qu'ils jouent ou ont l'intention de jouer et, d'autre part, leur jugement sur ce qu'*ImPact* est en train de jouer. Puisque les mains du musicien sont toujours occupées, il faut concevoir des interfaces pouvant être actionnée par les pieds, par le

regard, ou encore par des signaux musicaux (*cues*). Des interfaces de ce type sont discutées en détail dans une édition récente de *La Recherche* (La_Recherche, 1996).

Ensuite, nous pensons travailler sur l'automatisation même de l'écoute. On sait qu'il s'agit d'un problème très difficile mais qui heureusement a été partiellement résolu par d'autres chercheurs (Hidaka, Goto & Muraoka, 1995; Pennycook, Stammen & Reynolds, 1993; Rowe, 1993; Walker, 1994). Parmi les divers aspects de la perception le problème de l'anticipation nous intéresse en particulier il a été très peu étudié jusqu'à présent, et ce malgré les différents travaux réalisés en robotique (McDermott, 1992) ou dans d'autres activités (Meyer, 1996)

L'aide à l'arrangement

Une autre perspective est l'utilisation d'ImPact en tant qu'outil d'aide à l'arrangement. Le système *Band-in-a-box* est utilisé pour éviter que l'on ait à fournir au séquenceur les parties de la section rythmique lorsqu'elles sont simples. Les parties de la section rythmique sont donc générées *automatiquement* et ensuite, avec des modifications éventuelles, l'arrangeur ajoute *à la main* les parties des cuivres, des cordes, etc. Le problème est que l'on a peu de contrôle sur ce que *Band-in-a-box* va produire. En plus de rentrer les fragments mélodiques de sa préférence, les seules choses que l'arrangeur peut faire est d'établir le tempo et le nombre de chorus, et de déterminer les sections de la grille que *Band-in-a-box* doit souligner basculant le style entre la blanche et la noire. À l'exception de ce dernier choix, l'arrangeur ne peut donner de critères fins sur les traits musicaux des notes qui vont être jouées. Par exemple, il ne peut spécifier que la basse ou le piano joue de façon plus dissonante, plus syncopé, etc. dans un passage donné (mesure, section, chorus, etc.).

Tout ce contrôle pourrait se faire très facilement avec ImPact. L'arrangeur n'aurait qu'à choisir un certain nombre de "consignes" (PACTs) et les "placer" sur la grille (leur donner un début et une durée), avant qu'ImPact ne commence à jouer. Les PACTs fournies par l'arrangeur seraient naturellement prises en compte pendant le raisonnement. Pour en être sûr, nous pouvons imposer que ces PACTs soient les seules à être activées durant leur laps de temps respectifs. L'arrangeur pourrait ainsi se constituer une bibliothèque de "consignes", certaines pouvant éventuellement être appliquées à divers morceaux et d'autres à un morceau spécifique. Bien entendu, outre

la possibilité de donner *en avance* des consignes (PACTs), nous pourrions permettre à l'arrangeur d'écrire ses propres règles d'activation de PACTs et d'entrer ses propres cas.

Auto-évaluation et apprentissage

Le dernier thème de recherche vise à doter *ImPact* de la capacité d'évaluer ce qu'il a joué et, en conséquence, de faire évoluer sa façon de jouer. Nous aurions pu compléter la boucle du raisonnement à partir de cas en ajoutant la PACT qui vient d'être jouée à la base. L'opération d'ajout est en soi triviale à réaliser. Le problème est de savoir *quand* ajouter une PACT jouée. Il serait redondant d'ajouter, par exemple, un fragment qui a été simplement transposé. En outre, comme nous l'avons examiné dans la section 7.2.3, il serait souhaitable d'avoir un jugement de la qualité musical du fragment à être ajouté. C'est pour ces raisons que nos travaux sur cet axe de recherche se dérouleront en deux étapes.

La première étape consiste donc à construire un programme pouvant émettre un jugement esthétique sur une ligne de basse. Deux raisons nous mènent à croire que ceci est faisable. Premièrement, si les humains sont capables de le faire, s'il existe une expertise pouvant être modélisée, rien ne nous empêche d'essayer de concevoir un système qui effectue la même tâche. Nous retrouvons l'hypothèse des systèmes symboliques et physiques. De fait, certaines des analyses faites dans la section 7.2.2 peuvent être automatisées. Deuxièmement, il ne s'agit pas de concevoir un programme capable d'avoir un jugement esthétique "universel", d'autant plus qu'un tel jugement n'existe probablement pas. Le jugement reflétera forcément une vision esthétique personnelle.

La seconde étape est d'intégrer ce programme d'évaluation esthétique dans *ImPact*. Le jugement de la qualité musicale des fragments utilisés renforcera ou affaiblira leur chance d'être utilisés à nouveau dans des conditions similaires. De plus, si un fragment "intéressant" a subi d'énormes modifications dues au processus d'adaptation, il pourrait être ajouté à la base, en impliquant "l'oubli" d'un des cas existant dans la base. Les critères d'effacement d'un cas de base peuvent être sa "qualité musicale", stipulée par le programme d'évaluation esthétique, ou son "taux

d'utilisation". Les cas peu réutilisés seraient plus facilement oubliés. *ImPact* pourrait ainsi faire évoluer sa façon de jouer.

Nous concluons ce mémoire en disant qu'à travers ce travail de thèse nous espérons avoir contribué à réaffirmer une certaine vision de la recherche en IA. Nous croyons à l'importance d'une démarche empirique qui se situe au-delà des frontières des techniques établies pour être plus ouverte aux problèmes proches de la complexité du monde réel. Ces problèmes représentent en effet des situations privilégiées où la généralité des techniques courantes peut être vérifiée et le besoin de nouvelles techniques peut se faire sentir.

ANNEXE A - LIGNES DE BASSE JOUEES PAR RON CARTER

(AABERSOLD, 1979)

Stella by Starlight by: Ron Carter

File	Insert notes	Global Edit	Browse	MIDI	Refresh	Various
------	--------------	-------------	--------	------	---------	---------

The musical score is presented in a software interface window. It features a title bar with the text "Stella by Starlight by: Ron Carter". Below the title bar is a menu bar with seven items: "File", "Insert notes", "Global Edit", "Browse", "MIDI", "Refresh", and "Various". The main area of the window contains a musical score for the bass clef. The score is organized into eight systems, each starting with a measure number (1, 5, 9, 13, 17, 21, 25, 29). Above each system, specific chords are indicated: E halfDdim7, A 7, C min 7, F 7; F min 7, Bb 7, Eb maj7, Ab 7; Bb maj7, E halfDdim7/A 7, D min 7, G min 7 C 7; F maj7, G min 7 C 7, A halfDdim7, D 7; G 7, G 7, C min 7, C min 7; Eb min 7, Eb min 7 Ab 7, Bb maj7, Bb maj7; E halfDdim7, A 7, D halfDdim7, G 7; C halfDdim7, F 7, Bb maj7, Bb maj7. The notation includes eighth and quarter notes, rests, and accidentals (sharps and flats) on a five-line staff.

Figure AA.1- STELLA BY STARLIGHT par Ron Carter - chorus 1

Stella by Starlight by: Ron Carter

File	Insert notes	Global Edit	Browse	MIDI	Refresh	Various
------	--------------	-------------	--------	------	---------	---------

The image displays a musical score for the bass line of 'Stella by Starlight' by Ron Carter. The score is presented in a software interface with a menu bar at the top containing options: File, Insert notes, Global Edit, Browse, MIDI, Refresh, and Various. The music is written in bass clef with a key signature of two flats (Bb and Eb). The score is divided into eight systems, each starting with a measure number and followed by four measures of music with corresponding chord symbols above them. The chord progressions are as follows:

- System 1 (Measures 33-36): E halfDim7, A 7, C min 7, F 7
- System 2 (Measures 37-40): F min 7, Bb 7, Eb maj7, Ab 7
- System 3 (Measures 41-44): Bb maj7, E halfDim7 A 7, D min 7, G min 7 C 7
- System 4 (Measures 45-48): F maj7, G min 7 C 7, A halfDim7, D 7
- System 5 (Measures 49-52): G 7, G 7, C min 7, C min 7
- System 6 (Measures 53-56): Eb min 7, Eb min 7 Ab 7, Bb maj7, Bb maj7
- System 7 (Measures 57-60): E halfDim7, A 7, D halfDim7, G 7
- System 8 (Measures 61-64): C halfDim7, F 7, Bb maj7, Bb maj7

Figure AA.2- STELLA BY STARLIGHT jouée par Ron Carter - chorus 2

Stella by Starlight by: Ron Carter

File	Insert notes	Global Edit	Browse	MIDI	Refresh	Various
------	--------------	-------------	--------	------	---------	---------

The image displays a musical score for the bass line of 'Stella by Starlight' by Ron Carter. The score is presented in a software interface with a menu bar at the top. The music is written in bass clef with a key signature of one flat (Bb). The score is divided into measures, with chord progressions indicated above the staff. The measures shown are 65 through 93, with some measures containing multiple chords. The chords are: 65: E halfDim7, A 7, C min 7, F 7; 69: F min 7, Bb 7, Eb maj7, Ab 7; 73: Bb maj7, E halfDim7 A 7, D min 7, G min 7, C 7; 77: F maj7, G min 7, C 7, A halfDim7, D 7; 81: G 7, G 7, C min 7, C min 7; 85: Eb min 7, Eb min 7, Ab 7, Bb maj7, Bb maj7; 89: E halfDim7, A 7, D halfDim7, G 7; 93: C halfDim7, F 7, Bb maj7, Bb maj7.

Figure AA.3- STELLA BY STARLIGHT jouée par Ron Carter - chorus 3

What Is This Thing Called Love by: Ron Carter

File	Insert notes	Global Edit	Browse	MIDI	Refresh	Various
------	--------------	-------------	--------	------	---------	---------

The image displays a musical score for the bass line of the chorus of "What Is This Thing Called Love" by Ron Carter. The score is presented in a software interface with a menu bar at the top containing options: File, Insert notes, Global Edit, Browse, MIDI, Refresh, and Various. The music is written on a single bass clef staff with a key signature of one flat (Bb) and a 4/4 time signature. The score is divided into eight measures, each with a measure number (1, 5, 9, 13, 17, 21, 25, 29) and a corresponding chord progression written above the staff. The chords are: Measure 1: C7 aug9, C7 aug9, F min 7, F min 7; Measure 5: D halfDim7, G7 aug9, C, C; Measure 9: C7 aug9, C7 aug9, F min 7, F min 7; Measure 13: D halfDim7, G7 aug9, C, C; Measure 17: C min 7, F7, Bb, Bb; Measure 21: Ab7, Ab7, G7, G7; Measure 25: C7 aug9, C7 aug9, F min 7, F min 7; Measure 29: D halfDim7, G7 aug9, C, C. The notation includes eighth and quarter notes, rests, and accidentals (flats and naturals).

Figure A.A.4 - WHAT IS THIS THING CALLED LOVE jouée par Ron Carter - chorus 1

What Is This Thing Called Love by: Ron Carter

File Insert notes Global Edit Browse MIDI Refresh Various

33 C 7 aug⁹ C 7 aug⁹ F min 7 F min 7

37 D half D im 7 G 7 aug⁹ C C

41 C 7 aug⁹ C 7 aug⁹ F min 7 F min 7

45 D half D im 7 G 7 aug⁹ C C

49 C min 7 F 7 Bb Bb

53 Ab 7 Ab 7 G 7 G 7

57 C 7 aug⁹ C 7 aug⁹ F min 7 F min 7

61 D half D im 7 G 7 aug⁹ C C

Figure A.A.4 - WHAT IS THIS THING CALLED LOVE jouée par Ron Carter - chorus 2

What Is This Thing Called Love by: Ron Carter

File	Insert notes	Global Edit	Browse	MIDI	Refresh	Various
------	--------------	-------------	--------	------	---------	---------

65 C 7 aug⁹ C 7 aug⁹ F min 7 F min 7

69 D halfDim7 G 7 aug⁹ C C

73 C 7 aug⁹ C 7 aug⁹ F min 7 F min 7

77 D halfDim7 G 7 aug⁹ C C

81 F min 7 F 7 Bb Bb

85 Ab 7 Ab 7 G 7 G 7

89 C 7 aug⁹ C 7 aug⁹ F min 7 F min 7

93 D halfDim7 G 7 aug⁹ C C

Figure A.A.5 - WHAT IS THIS THING CALLED LOVE jouée par Ron Carter - chorus 3

Cherokee by: Ron Carter

File	Insert notes	Global Edit	Browse	MIDI	Refresh	Various
------	--------------	-------------	--------	------	---------	---------

The musical score is presented in a software interface. It features a menu bar with options: File, Insert notes, Global Edit, Browse, MIDI, Refresh, and Various. Below the menu is a musical score for the first chorus of 'Cherokee' by Ron Carter. The score is written in bass clef with a key signature of two flats (Bb and Eb). It consists of eight lines of music, each starting with a measure number (1, 5, 9, 13, 17, 21, 25, 29). Chords are indicated above the notes: Bb, Eb, F min 7, Bb 7, Ab 7, C 7, G 7 aug9, and C min 7. The score includes various musical notations such as eighth and sixteenth notes, rests, and accidentals.

Figure A.A.6 - CHEROKEE jouée par Ron Carter - chorus 1

Cherokee by: Ron Carter

File	Insert notes	Global Edit	Browse	MIDI	Refresh	Various
------	--------------	-------------	--------	------	---------	---------

The image displays a musical score for the piece "Cherokee" by Ron Carter. The score is presented in a software interface with a menu bar at the top containing options: File, Insert notes, Global Edit, Browse, MIDI, Refresh, and Various. The music is written in bass clef with a key signature of one flat (Bb). The score is divided into eight systems, each starting with a measure number and a chord symbol above the staff. The chords and measure numbers are as follows:

- System 1: Measure 33, Chord C# min 7
- System 2: Measure 37, Chord B min 7
- System 3: Measure 41, Chord A min 7
- System 4: Measure 45, Chord G min 7
- System 5: Measure 49, Chord Bb
- System 6: Measure 53, Chord Eb
- System 7: Measure 57, Chord Bb
- System 8: Measure 61, Chord C min 7

Chord symbols for measures 34-36, 38-40, 42-44, 46-48, 50-52, 54-56, 58-60, and 62-64 are also present but not explicitly labeled with text in the image. The notation includes eighth and quarter notes, rests, and accidentals.

Figure A.A.7 - CHEROKEE jouée par Ron Carter - chorus 2

Cherokee by: Ron Carter

File	Insert notes	Global Edit	Browse	MIDI	Refresh	Various
------	--------------	-------------	--------	------	---------	---------

65 Bb Bb F min 7 Bb 7

69 Eb Eb Ab 7 Ab 7

73 Bb Bb C 7 C 7

77 C min 7 G 7 aug9 C min 7 F 7

81 Bb Bb F min 7 Bb 7

85 Eb Eb Ab 7 Ab 7

89 Bb Bb C 7 C 7

93 C min 7 F 7 Bb Bb

Figure A.A.8- CHEROKEE jouée par Ron Carter - chorus 3

Cherokee by: Ron Carter

File	Insert notes	Global Edit	Browse	MIDI	Refresh	Various
------	--------------	-------------	--------	------	---------	---------

The image displays a musical score for the piece "Cherokee" by Ron Carter, specifically the chorus section. The score is presented in a software interface with a menu bar at the top containing options: File, Insert notes, Global Edit, Browse, MIDI, Refresh, and Various. The music is written in bass clef with a key signature of two flats (Bb and Eb). The score consists of eight staves, each representing a four-measure phrase. Measure numbers are indicated at the beginning of each staff: 97, 101, 105, 109, 113, 117, 121, and 125. Chord symbols are placed above the notes to indicate the harmonic structure. The chords used are: C# min 7, F# 7, B, A, B min 7, E 7, A, A min 7, D 7, G, G min 7, C 7, C min 7, F 7, Bb, Bb, F min 7, Bb 7, Eb, Eb, Ab 7, Ab 7, Bb, Bb, C 7, C 7, C min 7, F 7, Bb, and Bb.

Figure A.A.9- CHEROKEE jouée par Ron Carter - chorus 4

Body And Soul by: Ron Carter

File	Insert notes	Global Edit	Browse	MIDI	Refresh	Various
------	--------------	-------------	--------	------	---------	---------

The musical score is presented in a software interface. At the top, there is a title bar "Body And Soul by: Ron Carter" and a menu bar with options: File, Insert notes, Global Edit, Browse, MIDI, Refresh, and Various. Below the menu is a scrollable area containing eight staves of music. Each staff begins with a measure number (1, 5, 9, 13, 17, 21, 25, 29) and contains a sequence of notes with chord symbols written above them. The chords are: Eb min 7, Bb 7, Eb min 7, Ab 7, Db maj7, Gb 7, F min 7, Bb 7 dim9; Eb min 7, C halfDim7, F 7 aug9, Bb min 7, Eb min #b 7, Db maj7, Bb 7 dim9; Eb min 7, Bb 7, Eb min 7, Ab 7, Db maj7, Gb 7, F min 7, Bb 7 dim9; Eb min 7, C halfDim7, F 7 aug9, Bb min 7, Eb min #b 7, Db maj7, Emin 7A 7; D maj7, Emin 7, F# min 7, G min 7, F# min 7, Emin 7A 7, Db maj7; D min 7, G7, C, A 7 dim9, D min 7, G7, C7, B7, Bb 7 dim9; Eb min 7, Bb 7, Eb min 7, Ab 7, Db maj7, Gb 7, F min 7, Bb 7 dim9; Eb min 7, C halfDim7, F7, Bb min 7, Eb min #b 7, Db maj7, Bb 7.

Figure A.A.9- BODY AND SOUL jouée par Ron Carter - chorus 1

Body And Soul by: Ron Carter

File	Insert notes	Global Edit	Browse	MIDI	Refresh	Various
------	--------------	-------------	--------	------	---------	---------

The image displays a digital music score for the bass part of 'Body And Soul' by Ron Carter. The score is presented in a software interface with a menu bar at the top containing options: File, Insert notes, Global Edit, Browse, MIDI, Refresh, and Various. The music is written on a single bass clef staff with a key signature of two flats (Bb and Eb). The score is divided into measures, with chord progressions indicated above the staff. The measures shown are 33, 37, 41, 45, 49, 53, 57, and 61. The chord progressions are as follows:

- Measures 33-36: Eb min 7, Bb 7, Eb min 7, Ab 7, Db maj7, Gb 7, F min 7, Bb 7 dim9
- Measures 37-40: Eb min 7, C halfDim7, F 7 aug9, Bb min 7, Eb min #b 7, Db maj7, Bb 7 dim9
- Measures 41-44: Eb min 7, Bb 7, Eb min 7, Ab 7, Db maj7, Gb 7, F min 7, Bb 7 dim9
- Measures 45-48: Eb min 7, C halfDim7, F 7 aug9, Bb min 7, Eb min #b 7, Db maj7, Emin 7A 7
- Measures 49-52: D maj7, Emin 7, F# min 7, G min 7, F# min #7, Emin 7A 7, Db maj7
- Measures 53-56: D min 7, G7, C, A 7 dim9, D min 7, G7, C7, B7, Bb 7 dim9
- Measures 57-60: Eb min 7, Bb 7, Eb min 7, Ab 7, Db maj7, Gb 7, F min 7, Bb 7 dim9
- Measures 61-64: Eb min 7, C halfDim7, F 7, Bb min 7, Eb min #b 7, Db maj7, Bb 7

Figure A.A.9- BODY AND SOUL jouée par Ron Carter - chorus 2

ANNEXE B - GLOSSAIRE MUSICAL²⁵

²⁵Adapté et inspiré des glossaires fournis dans (Hodeir, 1981; Hucher, 1996)

Accord : Emission simultanée de plusieurs sons superposés. L'accord est à la base de l'harmonie (v. système tonal). Il soutient la mélodie.

Bridge : Phrase centrale d'un thème. Dans les thèmes de 32 mesures groupées en 4 sections de type AABA, le bridge est la phrase de la section "B".

Chanson : De l'anglais *song*. Morceau.

Chorus : L'ensemble d'accords de la grille correspondant au thème principal. Les chorus des standards ont souvent 32 mesures et ceux des blues ont 12 mesures. Le déroulement d'une chanson se fait à plusieurs chorus : un chorus d'exposition du thème, plusieurs chorus d'improvisation et un dernier chorus d'exposition du thème.

Chromatique : Qui procède par demi-tons consécutifs (opposé à diatonique).

Contrepoint : Technique d'écriture musicale consistant à superposer des lignes mélodiques.

Diatonique : Qui procède par tons et demi-tons consécutifs (opposé à chromatique)

Gammes : Série conjointe de notes comprises dans l'intervalle d'une octave. Les gammes les plus courantes sont pentatoniques (cinq sons), heptatoniques (sept sons), et chromatiques (douze sons). Les gammes diatoniques, majeures ou mineures, sont des gammes heptatoniques.

Intervalle : Distance (harmonique ou mélodique) entre deux notes. La seconde est l'intervalle compris entre la tonique et le II^e degré, la septième entre la tonique et le VII^e degré, etc.

Leading tone : On dit de la dernière note de l'accord courant quand elle est à un intervalle d'un demi-ton de la première note de l'accord suivant.

Mélodie : Série de sons dont la succession constitue une structure perceptible, une forme achevée.

Mesures : Cycles rythmiques constituant la base d'un rythme. Ils sont composés de différents temps ordonnés entre eux.

Mode : Chacune des dispositions particulières d'une gamme.

Modulation : Passage d'une tonalité à une autre.

Polyphonie : Musique où l'on combine plusieurs voix.

Riff : Figure mélodico-rythmique destinée à être répétée.

Section Rythmique : Ensemble d'instruments qui "marquent le rythme": piano, guitare, contrebasse, batterie.

Song (Chant, chanson) : Se dit d'un thème de jazz emprunté au répertoire de Tin Pan Alley (quartier des éditeurs new-yorkais de musique légère), par opposition aux airs traditionnels et aux blues.

Structure Harmonique : Série d'accords joués en accompagnement d'un thème mélodique.

Syncopé : prolongation sur un temps fort d'un élément joué dans un temps faible.

Système Tonal : Ensemble des lois et conventions qui régissent l'organisation des tonalités et leurs rapports.

Tempo : Vitesse d'exécution. Notez en général pr nombre de noires par minute.

Temps : Unité de durée.

Tessiture : Echelle de sons émis par une voix ou par un instrument.

Thème : En jazz, le thème est un motif mélodique, rythmique et harmonique dont le déploiement forme une unité, qui peut donner lieu à toutes sortes d'interprétations.

Tonalité : Organisation de l'ensemble de notes selon une gamme.

Variation : Modification d'un thème: invention d'une paraphrase ou même de phrases nouvelles se rattachant au thème par un dénominateur commun (harmonie, rythme ou mélodie).

REFERENCES BIBLIOGRAPHIQUES ET DISCOGRAPHIQUES

- AAAI (1993). Workshop on Artificial Intelligence & Creativity. Stanford: The AAAI Press.
- Aabersold, J. (1979). *Play-along: Paying Dues*. New Albany: Janey Aabersold Pub.
- Aabersold, J. (1993). *Maiden Voyage. Play-a-Long Recording vol. 54.* . New Albany: Janey Aabersold Pub.
- Aamodt, A., & Plaza, E. (1994). Case-Based Reasoning: Fundamental Issues, Methodological Variations, and system Approaches. *Artificial Intelligence Communications*, 7(1), 39-59.
- Agon, C., Assayag, G., Fineberg, J., & Rueda, C. (1994). Kant: a Critique of Pure Quantification. In *International Computer Music Conference*, Aarhus: International Computer Music Association.
- Agre, P. (1995). The Concept of Planning. Séminaires en Intelligence Artificielle, organisées par M. Pitrat, 6 novembre 1995.
- Aha, D., Kliber, D., & Albert, M. (1991). Instance-Based Learning Algorithms. *Machine Learning*, 6, 37-66.
- Aïmeur, E. (1994) METIS: un système et une méthode d'explicitation de taxonomies destinés à l'identification de structures conceptuelles. Thèse de doctorat, Université Paris VI.
- Allen, J. (1983). Maintaining Knowledge About Temporal intervals. *Communications of the ACM*, 16(1), 832-43.
- Allen, J. (1984). Towards a general theory of action and time. *Artificial Intelligence*, 23, 123-54.
- Allen, P., & Dannenberg, R. (1990). Tracking Musical Beats in Real Time. In *International Computer Music Conference*, (pp. 140-3). Glasgow: International Computer Music Association.
- Ambros-Ingerson, J., & Steel, S. (1988). Integrating Planning, Execution and Monitoring. In *Sixth National Conference on Artificial Intelligence*, (pp. 83-8). The AAAI Press.
- Ames, C., & Domino, M. (1992). Cybernetic Composer: an overview. In M. Balaban, K. Ebicioglu, & O. Laske (Eds.), *Understanding Music with AI: Perspectives on Music Cognition* (pp. 186-205). Menlo Park: The AAAI Press.

- Anderson, J. (1983). *The Architecture of Cognition*. Massachusetts: Harvard University Press.
- Arcela, A., & Ramalho, G. (1991). A Formal Composition System based on the theory of Time Trees. In *International Computer Music Conference*, (pp. 246-9). Montreal: International Computer Music Association.
- Baggi, D. (1974) *Realization of the Unfigured Bass by Digital Computer*. Doctoral Dissertation, University of California (Berkeley).
- Baggi, D. (1992). NeurSwing: An Intelligent Workbench for the Investigation of Swing in Jazz. In D. Baggi (Eds.), *Computer-Generated Music* (pp. 79-93). IEEE Computer Society Press.
- Bain, W. (1989). Judge. In C. Riesbeck & R. Schank (Eds.), *Inside Case-Based Reasoning* (pp. 93-140). Northvale, NJ: Erlbaum.
- Baker, D. (1980). *Miles Davis Trumpet*. Giants of Jazz Series. Lebanon: Studio 224 Ed.
- Balzer, K., & Wegscheider, K. (1992). Counterpoint and Geometry. In *Third Workshop on Artificial Intelligence and Music, ECAI-92*, (pp. 16-23). Vienna
- Band-in-a-box (1995). Band-in-a-box Pro 6.0. Canada: PG Music Inc.
- Baudoin, P. (1990). *Jazz: mode d'emploi, 2 vols*. Paris: Editions Outre Mesure.
- Birtwhistle, G., Dahl, O., Myhrhaug, B., & Nygaard. (1973). *SIMULA begin*. New York: Petrocelli Charter.
- Boden, M. (1992). *The Creative Mind: Myths and Mechanisms*. London: Abacus.
- Bournaud, I. (1996) *Regroupement conceptuel pour l'organisation de connaissances*. Thèse de doctorat, Université Paris VI.
- Brachman, R., & Levesque, H. (1985). *Readings in Knowledge Representation*. Los Altos: Morgan Kaufmann.
- Breuker, J. (1993). Modeling Artificial Legal Reasoning. In *Knowledge Acquisition for Knowledge-Based Systems*, (pp. 66-78). Toulouse: Springer-Verlag.
- Brown, D., & Sidley, S. (1993). The Expression of Aesthetic Principles as Syntactic Structures and Heuristic Preferences and Constraints in a Computer Program that Composes Jazz Improvisations. In *AAAI Spring Symposium Workshop on Artificial Intelligence & Creativity*, (pp. 133-6). Stanford: The AAAI Press.
- Bundy, A. (1990). What Kind of Field is AI? In D. Partridge & Y. Wilks (Eds.), *The Foundations of Artificial Intelligence: a sourcebook* (pp. 215-22). Cambridge: Cambridge University Press.

- Charniak, E., & McDermott, D. (1985). *Introduction to Artificial Intelligence*. London: Addison-Wesley.
- Chemillier, M. (1990) *Structure et méthodes algébriques en informatique musicale*. Thèse de doctorat, Paris VI.
- Clancey, W. (1993). Situated Action: a Neuropsychological Interpretation Response to Vera and Simon. *Cognitive Science*, 17(1), 87-116.
- Cohen, H. (1981). *On the Modeling of Creative Behavior* (Rand Paper No. 6681). Rand Corporation, Santa Monica.
- Coker, J. (1970). *Patterns for Jazz*. Lebanon: Studio Publications/Recordings.
- Collier, G., & Collier, J. (1994). An Exploration of the Use of Tempo in Jazz. *Music Perception*, 11(3), 219-42.
- Coolman, T. (1985). *The Bass Tradition*. New Albany: Jamey Aabersold.
- Cope, D. (1991). *Computers and Musical Style*. Oxford: Oxford University Press.
- Cordier, M.-O. (1995). Doigtage intelligent d'une partition de guitare. *Dossier IA et Musique - Bulletin de l'AFIA*, 23(octobre), 46-8.
- Corruble, V. (1996) *Possibilités et limitations d'une approche inductive pour la découverte en médecine*. Thèse de doctorat, Université Paris VI.
- Corruble, V., & Ganascia, J.-G. (1994a). Discovery of the Causes of Leprosy: a Computational analysis. In *National Conference on Artificial Intelligence - AAAI'96*, Detroit
- Corruble, V., & Ganascia, J.-G. (1994b). Using formal induction techniques to aid research on human leukemia. *Blood Cells*, 19.
- Courtot, F. (1992). Logical Representation and Induction for Computer Assisted Composition. In M. Balaban, K. Ebicioglu, & O. Laske (Eds.), *Understanding Music with AI: Perspectives on Music Cognition* (pp. 157-81). Menlo Park: The AAAI Press.
- Cunningham, P., Smyth, B., Fin, D., & Cahill, E. (1993). Retrieval Issues in Real-World CBR Applications: How far can we go with discrimination nets? In *IJCAI Workshop on Re-Use of Designs*, Chambéry
- Dannenberg, R. (1984). An On-Line Algorithm for Real-Time Accompaniment. In *International Computer Music Conference*, (pp. 193-8). IRCAM, Paris: International Computer Music Association.
- Dartnall, T. (1994). *Artificial Intelligence and Creativity: an Interdisciplinary Approach*. Dordrecht: Kluwer Academic Pub.

- Davey, A. (1978). *Discourse Production: A Computer Model of Some Aspects of a Speaker*. Edinburgh: Edinburgh University Press.
- Davis, M. (1958). *Milestones*. Columbia.
- Davis, M. (1962). *Kind of Blue*. CBS.
- Dennett, D. (1993). Allen Newell, *Unified Theories of Cognition* (Book Review). *Artificial Intelligence*, 59, 285-94.
- Desain, P., & Honing, H. (1992). *Music, Mind and Machine: Studies in Computer Music, Music Cognition and Artificial Intelligence*. Amsterdam: Thesis Publishers.
- Desain, P., & Honing, H. (1994). Advanced Issues in Beat Induction Modeling: Syncopation, Tempo and Timing. In *International Computer Music Conference*, (pp. 92-4). Aarhus: International Computer Music Association.
- Dietrich, E. (1990). Programs in Search for Intelligent Machines: the mistaken foundation of AI. In D. Partridge & Y. Wilks (Eds.), *The Foundations of Artificial Intelligence: a sourcebook* (pp. 223-33). Cambridge: Cambridge University Press.
- Dodge, C. (1995). Cité par L'éditeur Stephen Pope. *Computer Music Journal* (Vol. 19, pp. 1).
- Dojat, M., Harf, A., Touchard, D., Laforest, H., Lemaire, F., & Brochard, L. (1996). Evaluation of a Knowledge-Based System Providing Ventilatory Management and Decision for Extubation. *American Journal of Respiratory and Critical Care Medicine*(153), 997-1004.
- Ducornau, R. (1988). *YAFOOL version 3.22. Manuel de référence*. Montrouge: SEMA.METRA.
- Ebcioğlu, K. (1992). An expert System for Harmonizing Chorales in the Style of J.S. Bach. In M. Balaban, K. Ebcioğlu, & O. Laske (Eds.), *Understanding Music with AI: Perspectives on Music Cognition* (pp. 294-333). Menlon Park: The AAAI Press.
- Elton, M. (1993). *Toward Artificial Creativity* (Cognitive Science Research Paper, CSRP No. 228). The University of Sussex.
- Faron, C., & Kieu, Q. (1993). SATELIT: un outil d'explicitation de connaissances hypermédia. In *Journées Françaises d'Acquisition de Connaissances - JAC'93*, Strasbourg.

- Ferber, J. (1985) *MERING: un langage d'acteurs pour la représentation de connaissances*. Thèse de Docteur Ingénieur, Université Paris VI.
- Ferber, J. (1989). Systèmes experts et approches orientées objet. In *6èmes JISEA*, (pp. 525-42). Avignon
- Florens, C., & Cadoz, C. (1991). The Physical Model, Modelisation and Simulation Systems of the Instrumental Universe. In G. De Poli, A. Picciali, & C. Roads (Eds.), *Representations of Musical Signals* Massachusetts: MIT Press.
- Fry, C. (1991). Flavors Band: A Language for Specifying Musical Style. In S. Pope (Eds.), *The Well-Tempered Object: Musical Applications of Object-Oriented Software Technology* (pp. 49-63). Massachusetts: MIT Press.
- Fux, J. (1725). *Gradus ad Parnassum (Steps to Parnassus)*. New York: W. W. Norton.
- Ganascia, J.-G. (1990a). Commentary on Schank's Paper. In Y. Kodratoff & R. Michalski (Eds.), *Machine Learning: An Artificial Intelligence Approach Vol. III* Los Altos: Morgan Kaufmann.
- Ganascia, J.-G. (1990b). *L'âme-machine: les enjeux de l'intelligence artificielle*. Science Ouverte. Paris: Seuil.
- Ganascia, J.-G. (1990c). L'hypothèse du "Knowledge Level": théorie et pratique. In G. Vergnaud (Eds.), *Les sciences cognitives en débat* Paris: Editions du CNRS.
- Ganascia, J.-G. (1993). *L'intelligence artificielle*. Dominos. Paris: Flammarion.
- Gebhardt, F., Voss, A., Gräther, W., & Schmidt-Belz, B. (1996). *Reasoning with Complex Cases*. Boston: Kluwer.
- Georgeff, M., & Lansky, A. (1987). Reactive Reasoning and Planning. In J. Allen, J. Hendler, & A. Tate (Eds.), *Readings in Planning* (pp. 729-34). Los Altos: Morgan Kaufmann.
- Georgeff, M., & Rao (1995). The Semantics of Intention Maintenance for Rational Agents. In *International Joint Conference on Artificial Intelligence '95*, (pp. 704-10). Montreal
- Giomi, F., & Ligabue, M. (1991). Computational Generation and Study of Jazz Music. *Interface*, 20(1), 47-63.
- Goldberg, A., & Robson, D. (1983). *Smalltalk-80: The Language and its Implementation*. London: Addison-Wesley.
- Goldman, C., Gang, D., & Rosenschein, J. (1995). NetNeg: A Hybrid System Architecture for Composing Polyphonic Music. In *Fourth Workshop on Artificial Intelligence and Music, IJCAI-95*, (pp. 16-23). Montreal

- Grolimund, S., & Ganascia, J.-G. (1996). Speeding-up Nearest Neighbour Memories: The Template Tree Case Memory Organization. In *International Conference on Machine Learning*, (pp. 225-33). Bari: Morgan Kaufmann.
- Grubb, L., & Dannenberg, R. (1994). Automated Accompaniment of Musical Ensembles. In *Twelfth National Conference on Artificial Intelligence*, (pp. 94-9). The AAAI Press.
- Hayes-Roth, B. (1995a). An Architecture for Adaptive Intelligent Systems. *Artificial Intelligence*, 72, 329-65.
- Hayes-Roth, B. (1995b). On Building Integrated Cognitive Agents: a review of Newell's "Unified Theories of Cognition". *Artificial Intelligence*, 59, 213-20.
- Hayes-Roth, B., Washington, R., Ash, D., Hewett, R., Collinot, A., Vina, A., & Seiveur, A. (1992). Guardian: a Prototype Intelligent Agent for Intensive-Care Monitoring. *Artificial Intelligence in Medicine*, 4, 165-85.
- Hendler, J., Austin, T., & Musliner, D. (1993). Tutorial on Real-Time Intelligent Planning and Control. In *12th National Conference on Artificial Intelligence*, Seattle: AAAI Press.
- Hewitt, C., Bishop, P., & Steiger, R. (1973). A Universal Modular ACTOR Formalism for Artificial Intelligence. In *3rd International Joint Conference on Artificial Intelligence*, (pp. 167-82). Washington, D.C.
- Hidaka, I., Goto, M., & Muraoka, Y. (1995). An Automatic Jazz Accompaniment System Reacting to Solo. In *International Computer Music Conference*, (pp. 167-70). Banff: International Computer Music Association.
- Hiller, L., & Leonard, I. (1959). *Experimental Music*. New York: McGraw-Hill.
- Hindemith, P. (1968). *Craft of Musical Composition*. London: Schott.
- Hodeir, A. (1981). *Hommes et Problèmes du Jazz*. Paris: Parenthèses.
- Hodeir, A. (1995). Deux temps à la recherche. *Musurgia*, 2(3), 35-42.
- Hodgson, P. (1996a). Charlie Parker is alive! Communication faite en visites au LAFORIA en juin 1994 et mars 1996
- Hodgson, P. (1996b). Modelling Cognition in Creative Musical Improvisation (unpublished poster). In *International Conference on Music Perception and Cognition*, Montreal
- Hofstadter, D. (1986). *Metamagical Thema*. London: Penguin.

- Hofstadter, D. (1993). How could a copycat ever be creative? In *AAAI Spring Symposium Workshop on Artificial Intelligence & Creativity*, (pp. 1-10). Stanford: The AAAI Press.
- Holland, S. (1989) *Artificial Intelligence, Education and Music: the use of artificial intelligence to encourage and facilitate music composition by novices*. Ph.D. Thesis, Open University, London.
- Holland, S. (1994). Learning About Harmony Space: An Overview. In M. Smith, A. Smaill, & G. Wiggins (Eds.), *Music Education: an Artificial Intelligence Perspective* (pp. 41-55). London: Springer-Verlag.
- Horowitz, D. (1995). Representing Musical Knowledge in Jazz Improvisation System. In *Fourth Workshop on Artificial Intelligence and Music, IJCAI-95*, (pp. 16-23). Montreal
- Hucher, P. (1996). *Le Jazz*. Dominos. Paris: Flammarion.
- Johnson-Laird, P. (1987). Reasoning, imagining and creating. *Bulletin of the British Psychological Society*, 40, 121-9.
- Johnson-Laird, P. (1991). Jazz improvisation: a theory at the computational level. In P. Howell, R. West, & I. Cross (Eds.), *Representing Musical Structure* (pp. 291-325). London: Academic Press.
- Johnson-Laird, P. (1992). *The Computer and the Mind*. London: Fontana.
- Jung, S., Mohr, R., & Napoli, A. (1988). ORPHEE, a 0++ System. In T. Hervé, V. Rialle, & C. Roche (Eds.), *Artificial Intelligence and Cognitive Sciences* (pp. 96-106). New York: Manchester University Press.
- Kaebling, L. (1987). An Architecture for Intelligent Reactive Systems. In J. Allen, J. Hendler, & A. Tate (Eds.), *Readings in Planning* (pp. 713-26). Los Altos: Morgan Kaufmann.
- Kernfeld, B. (1983). Two Coltranes. *Annual Review of Jazz Studies*, 2, 7-66.
- Kolodner, J. (1983). Maintaining Organization in Dynamic Long-Term Memory. *Cognitive Science*, 7(4), 243-280.
- Kolodner, J. (1988). Retrieving Events from a Case Memory: A parallel implementation. In *Ninth Annual Conference of the Cognitive Science Society*, Northvale: Erlbaum.
- Kolodner, J. (1993). *Case-Based Reasoning*. San Mateo: Morgan Kaufmann.
- Kolodner, J., & Simpson, R. (1989). The MEDIATOR: Analysis of an earlier an case-based reasoner. *Cognitive Science*, 13(4), 507-49.

- Koton, P. (1988). Reasoning About Evidence in Causal Explanation. In *Sixth National Conference on Artificial Intelligence*, Cambridge: AAAI Press.
- Kowalski, R., & Sergot, M. (1986). A Logic-Based Calculus of Events. *New Generation Computing*, 4, 67-95.
- Kugel, P. (1992). Beyond Computational Musicology. In M. Balaban, K. Ebicioglu, & O. Laske (Eds.), *Understanding Music with AI: Perspectives on Music Cognition* (pp. 31-48). Menlo Park: The AAAI Press.
- Laird, J., Newell, A., & Rosebloom, P. (1987). SOAR: An Architecture of General Intelligence. *Artificial Intelligence*, 33, 1-64.
- Langley, P., Simon, H., Bradshaw, G., & Zytkow, J. (1987). *Scientific Discovery: Computational Explorations of the Creative Processes*. Cambridge: The MIT Press.
- La_Recherche (1996). L'ordinateur à doigt et à l'œil. *La Recherche*, 285.
- Large, E. (1995). Beat Tracking with a Nonlinear Oscillator. In *Fourth Workshop on Artificial Intelligence and Music, IJCAI-95*, (pp. 24-31). Montreal
- Laske, O. (1989). Composition Theory: an Enrichment of Music Theory. *Interface*, 18(1-2), 45-59.
- Laurière, J.-L. (1987). *Résolution de problèmes par l'homme et la machine*. Paris: Editions Eyrolles.
- Le Roux, B. (1994) *Eléments d'une approche constructive de la modélisation et de la réutilisation en acquisition des connaissances*. Thèse de Doctorat, Université Paris VI.
- Lenat, D., & Brown, S. (1984). Why AM and EURISKO Appear to Work? *Artificial Intelligence*, 23, 269-94.
- Lenat, D., & Feigenbaum, E. (1991). On the Thresholds of Knowledge. *Artificial Intelligence*, 47(1-3), 185-250.
- Lerhdal, F., & Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. Massachusetts: The MIT Press.
- Lesser, V., Pavlin, J., & Durfee, E. (1988). Approximate Processing in Real-Time Problem Solving. *AI Review*, spring, 49-61.
- Levitt, D. (1983) *A Melody Description System for Jazz Improvisation*. M.Sc. Thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

- Levitt, D. (1993). A Representation for Musical Dialects. In S. Schwanauer & D. Levitt (Eds.), *Machine Models of Music* (pp. 455-69). Massachusetts: The MIT Press.
- Little, D. (1996). Composing with Chaos: Applications of a New Science for Music. In *Journées d'Informatique Musicale -JIM'96*, Ile de Tatihou, Normandie
- Longuet-Higgins, C., & Lee, C. (1984). The Rhythmic Interpretation of Monophonic Music. *Music Perception*, 1(4), 424-41.
- Lord, A. (1964). *The Singer of Tales*. Cambridge: Harvard University Press.
- Loy, G. (1985). Musicians Make a Standard: The MIDI Phenomenon. In C. Roads (Eds.), *The Music Machine* (pp. 181-98). Massachusetts: The MIT Press.
- Malt, M. (1996). Lambda3.99: chaos et composition musicale. In *Journées d'Informatique Musicale -JIM'96*, (pp. 44-54). Ile de Tatihou, Normandie
- Marsella, S., & Schmidt, C. (1992). On the Application of Problem Reduction Search to Automated Composition. In M. Balaban, K. Ebicioglu, & O. Laske (Eds.), *Understanding Music with AI: Perspectives on Music Cognition* (pp. 238-333). Menlo Park: The AAAI Press.
- Masini, G., Napoli, A., Colnet, D., Léonard, D., & Tombre, K. (1989). *Les langages à objet: langages de classes, langages de frmaes, langages d'acteurs*. Paris: InterEditions.
- Masterman, M. (1971). Computerized Haiku. In J. Reichardt (Eds.), *Cybernetic, Art and Ideas* (pp. 175-83). London: Studio Vista.
- Maurice-Demourieux, M., Lâasari, B., & Levallet, C. (1993). Le raisonnement à partir de cas: panorama et modelisation dynamique. In *Seminaire "Raisonnement à partir de cas"*, (pp. 53-67). Rapport interne LAFORIA 93/42.
- McCarthy, J., & Hayes, P. (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 6, 463-502.
- McDermott, J. (1989). Preliminary Steps Toward a Taxonomy of Problem-Solving Methods. In S. Marcus (Eds.), *Automation of Knowledge Acquisition for Expert Systems* (pp. 225-55). London: Academic Press.
- McDermott, D. (1992). Robot Planning. *AI Magazine*, Summer, 55-79.
- Mehegan, J. (1984). *Jazz Improvisation*. New York: AMSCO Music Pub.
- Meyer, C., Ganascia, J.-G. (1996). *SAGACE: Solution Algorithmique Génétique pour l'Anticipation de Comportements Evolutifs*. LAFORIA.
- Mingus, C. (1987). *New Tijuana Moods*. RCA.

- Minsky, M. (1975). A Framework for Representing Knowledge. In P. Winston (Eds.), *The Psychology of Computer Vision* (pp. 211-77). New York: McGraw-Hill.
- Mitchell, A. (1993). *Analogy-Making as Perception*. Cambridge: The MIT Press.
- Monro, G. (1995). Fractal Interpolation Waveforms. *Computer Music Journal*, 19(1), 88-98.
- Moog, B. (1986). MIDI: Musical Instrument Digital Interface. *Journal of the Audio Engineering Society*, 34(5), 394-404.
- Mouton, R., & Pachet, F. (1995). The Symbolic vs. Numeric Controversy in Automatic Analysis of Music. In *Fourth Workshop on Artificial Intelligence and Music, IJCAI-95*, (pp. 32-40). Montreal
- Musliner, D., Durfee, E., & Shin, K. (1993). CIRCA: A Cooperative Intelligent Real-Time Control Architecture. *IEEE Trans. Systems, Man, and Cybernetics*, 23(6), 1561-74.
- Musliner, D., Hendler, J., Agrawala, A., Durfee, E., Strosnider, J., & Paul, C. (1995). The Challenge for Real-Time AI. *Computer*, January, 58-66.
- Narmour, E. (1977). *Beyond Schenkerism*. Chicago: University of Chicago Press.
- Narmour, E. (1989). *The Analysis and Cognition of Basic Melodic Structures: the Implication-Realization Model*. Chicago: University of Chicago Press.
- Nebel, B. (1985). How Well Does a Vanilla Loop Fit into a Frame? *Data & Knowledge Engineering*, 1(2), 181-94.
- Newell, A. (1980). Physical Symbol Systems. *Cognitive Science*, 4, 135-83.
- Newell, A. (1982). The Knowledge Level. *Artificial Intelligence*, 18, 87-127.
- Newell, A. (1990). *Unified Theories of Cognition*. Cambridge: Harvard University Press.
- Newell, A., Shaw, J., & Simon, H. (1962). The Processes of Creative Thinking. In H. G. et al. (Eds.), *Contemporary Approaches to Creative Thinking* New York: Atherton Pub.
- Newell, A., & Simon, H. (1976). Computer Science as Empirical Inquiry: Symbols and Search. *Communications of ACM*, 19(3), 113-26.
- Niwa, S., Sasaki, K., & Ihara, H. (1984). An Experimental Comparison of Knowledge Representation Schemes. *The AI Magazine*, 5(2), 29-36.
- Owens, C. (1989). Integrating Feature Extraction and Memory Search. In *Eleventh Annual Conference of the Cognitive Science Society*, Northvale: Erlbaum.

- Owens, T. (1974) *Charlie Parker: Techniques of Improvisation*, 2 vols. Ph.D. Thesis, University of Los Angeles California.
- Pachet, F. (1987). *Utilisation d'un Système Expert pour Simuler les Comportements d'un Pianiste et d'un Bassiste Accompagnant un Improvisateur de Jazz* (Rapport de DEA No. LAFORIA, Université Paris VI.
- Pachet, F. (1990). Representing Knowledge Used by Jazz Musicians. In *International Computer Music Conference*, (pp. 285-8). International Computer Music Association.
- Pachet, F. (1992) *Representation de connaissances par objets et règles: le système NéOpus*. Thèse de doctorat, Université Paris VI.
- Pachet, F. (1994). The MusES system: an environment for experimenting with knowledge representation techniques in tonal harmony. In *First Brazilian Symposium on Computer Music - SBC&M '94*, (pp. 195-201). Caxambu: Computer Science Brazilian Society (SBC).
- Pachet, F. (1996). On the Embeddability of Production Rules in Oriented-Object Languages. *Journal of Oriented-Object Programming*, 8(4), 19-24.
- Pachet, F., & Assayag, G. (1995). Dossier IA et Musique. *Bulletin de l'AFIA*, 23 (octobre), 30-48.
- Pachet, F., Ramalho, G., & Carrive, J. (1996). Representing Temporal Musical Objects and Reasoning in the MusES System. *to appear in Journal of New Music Research*, 3.
- Pachet, F., & Roy, P. (1994). Mixing constraints and objects: a case study in automatic harmonization. In *TOOLS Europe*, (pp. 119-26). Versailles
- Parker, C. (1947). *The Savoy Recordings*, vol. 1. Savoy.
- Parker, C. (1957). *Now's the Time*. Verve Records.
- Parry, M. (1930). Studies in the Epic Technique of Oral Verse-making: Homer and Homeric Style, Part 1. *Harvard Studies in Classical Philology*, 41, 73-147.
- Parry, M. (1932). Studies in the Epic Technique of Oral Verse-making: Homer and Homeric Style, Part 2. *Harvard Studies in Classical Philology*, 43, 1-50.
- Partridge, D. (1990). What is an AI program? In D. Partridge & Y. Wilks (Eds.), *The Foundations of Artificial Intelligence: a sourcebook* (pp. 112-8). Cambridge: Cambridge University Press.
- Partridge, D., & Wilks, Y. (1990). *The Foundations of Artificial Intelligence: a sourcebook*. Cambridge: Cambridge University Press.

- Pennycook, B., Stammen, D., & Reynolds, D. (1993). Toward a Computer Model of Jazz Improviser. In *International Computer Music Conference*, (pp. 228-31). Tokyo: International Computer Music Association.
- Plaza, E., & Lopez de Mantaras, R. (1990). A Case-Based Apprentice that Learns from Fuzzy Examples. In *First International Meeting on Advances in Learning (IMAL)*, (pp. 159-74). Les Arcs
- Pope, S. (1991a). Introduction to MODE: The Musical Object Development Environment. In S. Pope (Eds.), *The Well-Tempered Object: Musical Applications of Object-Oriented Software Technology* (pp. 83-106). Massachusetts: MIT Press.
- Pope, S. (1991b). *The Well-Tempered Object: Musical Applications of Object-Oriented Software Technology*. Massachusetts: MIT Press.
- Porter, B. (1988). Similarity assessment: computation vs. representation. In *Case-Based Reasoning*, (pp. 82-4). Morgan Kaufmann.
- Povel, D., & Essens, P. (1981). Perception of Temporal Patterns. *Music Perception*, 2(4), 309-20.
- Pressing, J. (1984). Cognitive Processes in Improvisation. In W. Crozier & A. Chapman (Eds.), *Cognitive Processes in the Perception of Art* (pp. 345-63). Amsterdam: North-Holland.
- Pressing, J. (1988). Improvisation: Methods and Models. In J. Sloboda (Eds.), *Generative Processes in Music: The Psychology of Performance, Improvisation and Composition* (pp. 129-78). Oxford: Oxford Science Publications.
- Quillian, M. (1968). Semantic Memory. In M. Minsky (Eds.), *Semantic Information Processing* (pp. 227-70). Cambridge: The MIT Press.
- Quinlan, J. (1986). Induction of Decision Trees. *Machine Learning*, 1(1), 81-106.
- Ram, A., & Santamaria, J. C. (1993). Continuous Case-Based Reasoning. In *AAAI Workshop on Case-Based Reasoning*, (pp. 86-93). Washington: The AAAI Press.
- Ramalho, G., & Ganascia, J.-G. (1994a). The Role of Musical Memory in Creativity and Learning: a Study of Jazz Performance. In M. Smith, A. Smaill, & G. Wiggins (Eds.), *Music Education: an Artificial Intelligence Perspective* (pp. 143-56). London: Springer-Verlag.

- Ramalho, G., & Ganascia, J.-G. (1994b). Simulating Creativity in Jazz Performance. In *Twelfth National Conference on Artificial Intelligence*, (pp. 108-13). Seattle: The AAAI Press.
- Ramalho, G., & Pachet, F. (1994). What is Needed to Bridge the Gap Between Real Book and Real Jazz Performance? In *Fourth International Conference on Music Perception and Cognition*, (pp. 349-50). Liège: Escom.
- Ramalho, G., Rolland, P.-Y., & Ganascia, J.-G. (1996). An Artificially Intelligent Jazz Performer. *submitted to the Journal of New Music Research*.
- Rich, E., & Knight, K. (1983). *Artificial Intelligence*. (2nd ed.). New York: McGraw-Hill.
- Richter, A., & Weiss, S. (1991). Similarity, Uncertainty and Case-Based Reasoning. In R. Boyer (Eds.), *Automated Reasoning: Essays in honor of Woody Bledsoe* (pp. 249-265). Kluwer Academic Press.
- Riecken, R. (1992). Wolfgang: A system Using Emoting Potential to Manage Musical Design. In M. Balaban, K. Ebicioglu, & O. Laske (Eds.), *Understanding Music with AI: Perspectives on Music Cognition* (pp. 206-36). Menlo Park: The AAAI Press.
- Riotte, A. (1990). Formalisation des échelles de hauteurs en analyse et en composition. In *Colloque International "Musique et Assistance Informatique"*, (pp. 247-260). Marseille: LIM.
- Rodet, X., & Cointe, P. (1984). FORMES: composition and scheduling of processes. *Computer Music Journal*, 8(3), 32-50.
- Rolland, P.-Y., & Ganascia, J. G. (1996a). Automated Identification of Prominent Motives Jazz Solos Corpuses. In *International Conference on Music Perception and Cognition*, (pp. 491-5). Montreal
- Rolland, P.-Y., & Ganascia, J. G. (1996b). Automated Motive-Oriented Analysis of Musical Corpuses: a Jazz Case Study. In *International Computer Music Conference*, (pp. 240-3). Honk Kong
- Rolland, P.-Y., & Ganascia, J. G. (1997). Musical Pattern Extraction and Similarity Assessment (to appear). *Contemporary Music Review*.
- Roosendaal, R. (1990). Generative Processes in Music: musical composition. In *Colloque International "Musique et Assistance Informatique"*, (pp. 207-22). Marseille: LIM.

- Rothgeb, J. (1993). Simulating Musical Skills by Digital Computer. In S. Schwanauer & D. Levitt (Eds.), *Machine Models of Music* (pp. 157-64). Massachusetts: The MIT Press.
- Rottlieb, L. (1991). Who So Sad, Pres? In L. Porter (Eds.), *A Lester Young Reader* (pp. 211-23). Washington: Smithsonian Institute Press.
- Rougegrez-Loriette, S. (1994) *Prédiction de processus à partir de comportements observés: le système REBECAS*. Thèse de Doctorat, Université Paris VI.
- Rowe, J., & Partridge, D. (1993). Creativity: a survey of AI approaches. *Artificial Intelligence Review*, 7, 43-70.
- Rowe, R. (1993). *Interactive Music Systems*. Massachusetts: The MIT Press.
- Rumelhart, D. (1975). Notes on a Schema of Stories. In Collins (Eds.), *Representation and Understanding: Studies in Cognitive Science* London: Academic Press.
- Russel, S. (1995). Rationality and Intelligence. In *International Joint Conference on Artificial Intelligence '95*, (pp. 950-7). Montreal
- Russel, S., & Norvig, P. (1995). *Artificial Intelligence: a Modern Approach*. Englewood Cliffs: Prentice-Hamm, Inc.
- Sabatella, M. (1996). *A Jazz Improvisation Primer*. Distribué sur le web: <http://www.fornet.org/~marc/primer>.
- Sayegh, S. (1989). Fingering for String Instruments with Optimum Path Paradigm. *Computer Music Journal*, 13(3).
- Schank, R. (1972). Conceptual Dependency: A Theory of Natural Language Understanding. *Cognitive Psychology*, 3(4).
- Schank, R. (1982). *Dynamic Memory: A Theory of Learning in Computers and People*. Cambridge: Cambridge University Press.
- Schank, R. (1990). Explanations, Machine Learning and Creativity. In Y. Kodratoff & R. Michalski (Eds.), *Machine Learning: An Artificial Intelligence Approach IV* (pp. 31-48). California: Morgan Kaufmann.
- Schank, R., & Riesbeck, C. (1989). *Inside Case-base Reasoning*. Northvale, NJ: Erlbaum.
- Schneider, F. (1988). An Implementation of Frames in Smalltalk. In T. Hervé, V. Rialle, & C. Roche (Eds.), *Artificial Intelligence and Cognitive Sciences* (pp. 139-46). New York: Manchester University Press.
- Schoenberg, A. (1943). *Models for beginners in Composition*. New York: Schirmer.

- Schottstaedt, B. (1983). Pla: A Composer's Idea of Language. *Computer Music Journal*, 7(1), 11-20.
- Schottstaedt, B. (1984). *Automatic Species Counterpoint* (Technical Report No. STAN-M-19). CCRMA, Stanford University.
- Schwanauer, S., & Levitt, D. (1993). *Machine Models of Music*. Massachusetts: The MIT Press.
- Sher, C. (1979). *The Improviser's Bass Method*. Petaluma: Sher Music.
- Sher, C. (1991). *The New Real Book (vol. 1 and 2)*. Berkeley: Sher Music.
- Simon, H. (1973). The Structure of Ill Structured Problems. *Artificial Intelligence*, 4, 181-201.
- Simon, H. (1981). *The Sciences of the Artificial*. Cambridge: The MIT Press.
- Simon, H. (1995). Explaining the Ineffable: AI on the Topics of Intuition, Insight and Inspiration. In *International Joint Conference on Artificial Intelligence '95*, (pp. 939-48). Montreal
- Sloboda, J. (1985). *The Musical Mind: the Cognitive Psychology of Music*. Oxford: Oxford University Press.
- Smith, G. (1983) *Homer, Gregory, and Bill Evans? The Theory of Formulaic Composition in the Context of Jazz Piano Improvisation*. Ph.D. Thesis, Harvard University.
- Smoliar, S. (1990). The Mental Process of Musical Composition. In *Colloque International "Musique et Assistance Informatique"*, (pp. 171-8). Marseille: LIM.
- Smyth, B., & Keane, M. (1994). Retrieving Adaptable Cases: The Role of Adaptation Knowledge in Case Retrieval. In *Topics in Case-Based Reasoning* (pp. 209-20). Springer Verlag.
- Smyth, B., & Keane, M. (1995a). Experiments on Adaptation-Guided Retrieval in Case-Based Design. In *First International Conference on Case-Based Reasoning*,
- Smyth, B., & Keane, M. (1995b). Remembering to Forget: A Competence-Preserving Case Deletion Policy for Case-Based Reasoning Systems. In *International Joint Conference on Artificial Intelligence '95*, (pp. 377-83). Montreal
- Sowa, J. (1984). *Conceptual Structures: Information Processing in Mind and Machine*. London: Addison-Wesley.

- Spector, L., & Alpern, A. (1994). Criticism, Culture and the Automatic Generation of Artworks. In *Twelfth National Conference on Artificial Intelligence*, (pp. 3-8). The AAAI Press.
- Spector, L., & Alpern, A. (1995). Induction and Recapitulation of Deep Musical Structure. In *Fourth Workshop on Artificial Intelligence and Music, IJCAI-95*, (pp. 41-8). Montreal
- Stammen, D., & Pennycook, B. (1993). Real-time Recognition of Melodic Fragments Using the Dynamic Timewarp Algorithm. In *International Computer Music Conference*, Tokyo: International Computer Music Association.
- Steedman, M. (1984). A Generative Grammar for Jazz Chord Sequences. *Music Perception*, 1(2), 52-77.
- Stefik, M., & Bobrow, D. (1986). Object-Oriented Programming: Themes and Variations. *AI Magazine*, 6(4), 40-62.
- Strosdiner, P. (1994). A Structured View of Real-Time Problem Solving. *The AI Review*, 15(2), 45-66.
- Strube, G., Enzinger, A., Janetzko, D., & Knauff, M. (1995). Knowledge Engineering for CBR systems from a Cognitive Science Perspective. In *First International Conference on Case-Based Reasoning*, (pp. 458-68). Lisboa: Springer-Verlag.
- Tate, A., Drabble, B., & Kirby, R. (1994). O-Plan2: an Open Architecture for Command, Planning and Control. In M. Zweeben & M. Fox (Eds.), *Intelligent Scheduling* San Mateo: Morgan Kaufmann.
- Tate, A., Hendler, J., & Drummond, M. (1990). A Review of AI Planning. In J. Allen, J. Hendler, & A. Tate (Eds.), *Readings in Planning* (pp. 26-49). San Mateo: Morgan Kaufmann.
- Taube, R. (1991). Common Music: A Music Composition Language in Common Lisp and Clos. *Computer Music Journal*, 15(2), 21-32.
- Thomas, M. (1985). Vivace: a rule based AI system for composition. In *International Computer Music Conference*, (pp. 267-74). Simon Fraser University: International Computer Music Association.
- Tsang, C., & Aitken, M. (1991). Harmonizing Music as a Discipline of Constraint Logic Programming. In *International Computer Music Conference*, Montreal: International Computer Music Association.
- Tulving, E. (1984). How Many Memory Systems Are There? *American Psychologist*, April, 384-97.

- Turing, A. (1950). Computer Machinery and Intelligence. *Mind*, 59, 433-60.
- Ulrich, W. (1977). The Analysis and Synthesis of Jazz by Computer. In *Fifth International Joint Conference on Artificial Intelligence*, (pp. 865-72). Massachusetts.
- Veloso, M. (1994). *Planning and Learning by Analogical Reasoning*. Berlin: Springer-Verlag.
- Vercoe, B. (1985). The Synthetic Performer in the Context of Live Performance. In *International Computer Music Conference*, (pp. 25-31). Simon Fraser University: International Computer Music Association.
- Vere, S. (1983). Planning in Time: windows and Duration for Activities and Goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(3), 246-67.
- Vincinanza, S., & Prietula, M. (1989). A Computational Model of Musical Creativity. In *Second Workshop on Artificial Intelligence and Music, IJCAI-89*, (pp. 21-25). Detroit.
- Walker, W. (1994) *A Conversation-Based Framework for Musical Improvisation*. Ph.D. Thesis, University of Illinois at Urbana-Champaign.
- Walker, W., Hebel, K., Martirano, S., & Scaletti, C. (1992). ImprovisationBuilder: Improvisation as conversation. In *International Conference on Computer Music*, San Jose: International Computer Music Association.
- Watson, I. (1996). Case-Based Reasoning: A Review. In *2nd UK CBR Workshop*, (pp. 71-88).
- Watterman, D., & Hayes-Roth, F. (1978). *Pattern-Directed Inference Systems*. New York: Academic Press.
- West, R., Howell, P., & Cross, I. (1991). Musical Structure and Knowledge Representation. In P. Howell, R. West, & I. Cross (Eds.), *Representing Musical Structure* (pp. 1-30). London: Academic Press.
- Wettschereck, D., & Aha, D. (1995). Weighting Features. In *First International Conference on Case-Based Reasoning*, (pp. 347-58). Sesimbra.
- Widmer, G. (1992). A Knowledge Intensive Approach to Machine Learning in Tonal Music. In M. Balaban, K. Ebicioglu, & O. Laske (Eds.), *Understanding Music with AI: Perspectives on Music Cognition* (pp. 490-507). Menlo Park: The AAAI Press.
- Wielinga, B., Schreiber, T., & Breuker, J. (1992). KADS: a Modeling Approach to Knowledge Engineering. *Knowledge Acquisition*, 4(1), 5-53.

- Winograd, T. (1975). Frame Representation and the Declarative/Procedural Controversy. In D. Bobrow & A. Collins (Eds.), *Representation and Understanding: Studies in Cognitive Science* (pp. 185-210). New York: Academic Press.
- Winograd, T. (1993). Linguistic and Computer Analysis of Tonal Harmony. In S. Schwanauer & D. Levitt (Eds.), *Machine Models of Music* (pp. 113-53). Massachusetts: The MIT Press.
- Winston, P. (1992). *Artificial Intelligence*. (3rd ed.). London: Addison-Wesley.
- Wolverton, M., & Hayes-Roth, B. (1984). Retrieving Semantically Distant Analogies with Knowledge-Directed Spreading Activation. In *Twelfth National Conference on Artificial Intelligence*, (pp. 56-61). Seattle: The AAAI Press.
- Zucker, J.-D., Corruble, V., Thomas, J., & Ramalho, G. (1994). DICE: a Discovery Environment integrating Inductive Bias. In *AAAI Workshop on Knowledge Discovery in Database*, (pp. 275-86). Seattle: AAAI Press