https://bolprocessor.org/misc/texts/AFCET87.pdf

Actes du 6e congrès AFCET/INRIA, Antibes, 1987b: 353-366.

GRAMMAIRES DE GENERATION ET DE RECONNAISSANCE DE PHRASES RYTHMIQUES

Bernard Bel
Groupe Représentation et Traitement des Connaissances
Centre National
de la Recherche Scientifique
31, ch Joseph Aiguier
13402 MARSEILLE cedex 9

Résumé:

Un projet d'anthropologie cognitive sur les langages de percussion du nord de l'Inde met en interaction des musiciens experts, un analyste et un automate simulant la génération et la reconnaissance de compositions rythmiques. Les connaissances et les hypothèses sur chaque type de composition sont formulées à l'aide de règles de production. Les grammaires sont ensuite modifiées jusqu'à ce que l'adéquation entre l'ensemble des phrases générées et la pratique musicale soit jugée satisfaisante. Cet exposé décrit le formalisme de représentation et les algorithmes utilisés pour la synthèse et l'analyse des phrases, en s'appuyant sur des exemples tirés du répertoire musical.

Mots clés:

anthropologie cognitive - langages formels - reconnaissance de phrases - syntaxe - apprentissage - dérivation canonique - rythme

Abstract:

In a cognitive anthropological project dealing with drumming languages in North India, an interaction has been created between expert musicians, an analyst and an automaton simulating the generation and recognition of rhythmic compositions. Production rules are used to represent knowledge and hypotheses on each compositional type. Grammars are then modified in order to match the set of generated sentences with musical practice. This paper describes data representation and algorithms utilized for the synthesis and parsing of sentences, with the support of examples drawn from the musical repertoire.

Keywords:

cognitive anthropology - formal languages - sentence recognition - syntax - machine-learning - canonic derivation - rhythm

1. Introduction

Le domaine auquel cet article se réfère est celui du qa'ida, terme qui signifie 'règles' ou 'système' et qui désigne un type de composition utilisé

par les musiciens du nord de l'Inde. Ces pièces sont interprétées sur le tabla, instrument de percussion en deux parties frappées avec les mains. L'utilisation de grammaires génératives pour la représentation de langages rythmiques découle de l'importance des concepts de permutation, substitution et de contexte, notions faciles à traduire par des règles de production. L'expérience montre d'autre part que ce formalisme est accessible à un analyste non informaticien et même à certains musiciens [Kippen 86].

A partir d'une grammaire donnée, un automate (le Bol Processor) synthétise des phrases qui sont soumises à l'évaluation des musiciens. La manipulation de poids permet de rendre certaines règles plus probables que d'autres et d'explorer ainsi les frontières du langage. Après un certain temps de mise au point, il est possible que la grammaire ne produise que des variations correctes du qa'ida. Il faut alors s'assurer que le langage qu'elle génère contient toutes les variations possibles. Dans ce but, on demande au musicien de composer des variations qui sont soumises à un test d'appartenance (membership test). Toute évaluation incorrecte entraîne une nouvelle modification de la grammaire.

Nous avons présenté ailleurs [Bel 87b] une approche de 'laboratoire' basée sur l'apprentissage: segmentation, formulation de règles particulières, puis généralisation des grammaires par un raisonnement inductif. Dans le contexte expérimental, au contraire, l'analyste doit partir d'une grammaire aussi générale que possible, représentant sa connaissance (incomplète) du langage. Cette connaissance est en général décrite de manière procédurale, une phrase du langage pouvant être considérée comme une dérivation du symbole de départ faisant intervenir plusieurs grammaires transformationnelles. Cette idée est renforcée par l'utilisation d'un algorithme déterministe de test d'appartenance, qui utilise une dérivation canonique applicable à une sous-classe des langages contextuels. Il est toutefois nécessaire d'introduire du non-déterminisme lorsqu'une phrase est soumise à l'analyse sans information sur sa structure: le moteur d'inférences du Bol Processor fait appel pour cela à une collection de gabarits qui lui permettent d'insérer des symboles structurels dans la phrase.

A l'aide d'un exemple du répertoire musical, nous montrons comment une grammaire peut être modifiée pour tenir compte des contraintes d'accentuation et de contexte. Nous terminons par une technique de validation de règles qui fait intervenir un modèle de raisonnement déductif et des mécanismes de contrôle analogues à ceux de l'apprentissage.

Cette étude est menée en collaboration avec J.R. Kippen, ethnomusicologue au département d'Anthropologie Sociale de l'Université de Belfast.

2. Notation des compositions rythmiques

A partir d'un certain niveau d'expertise, l'enseignement du tabla se fait surtout par l'intermédiaire des bols, onomatopées qui représentent à la fois les sons de l'instrument et les actions qui permettent de les produire. C'est dire qu'il est possible d'analyser séparément le système de production sonore et le contenu du langage réduit à sa représentation écrite.

Il existe une application idempotente de l'alphabet des bols, que nous appelons miroir, et qui fait correspondre à un son ouvert sa version fermée obtenue en étouffant la résonance de l'instrument. Dans les exemples que nous traiterons plus loin, l'alphabet du langage et l'application miroir sont définis ainsi:

Le symbole "-" représente un silence. Les temps sont marqués par des espaces ou des tabulations. Les bols situés à l'intérieur d'un temps ont tous la même valeur rythmique.

Les compositions utilisent souvent des répétitions de motifs, ainsi qu'une répétition composée avec l'application miroir. Nous utilisons pour noter ces structures un système de parenthèses et de pointeurs, ainsi qu'un marqueur de genre (ouvert/fermé). Par exemple:

- (= dha te te dha) (= - -) (: dha te te dha) (: - -) (: dha te te dha)

(= dha ge dhi na ghi na) * (: ta ke ti na ki na)

Une parenthèse qui contient "=" est appelée parenthèse maître. Celles qui contiennent ":" sont appelées parenthèses esclaves, et contiennent un pointeur vers une parenthèse maître située à leur gauche. Le contenu de la parenthèse qui suit le symbole "*" est une séquence de sons fermés. Plusieurs niveaux de parenthèses peuvent être imbriqués. Le moteur d'inférences remet à jour après chaque réécriture les pointeurs des parenthèses esclaves en tenant compte des modifications de structure.

La notation des variations de tempo, qui ne sont pas utilisées dans l'exemple cité en annexe, est explicitée ailleurs [Bel 87b].

3. Grammaires et classes de langages

L'analyste peut programmer des grammaires de type 0 (langages récursivement énumérables) sur le Bol Processor. Toutefois, pour que le test d'appartenance des phrases soit algorithmique, il faut que ces langages soient aussi récursifs, c'est à dire reconnaissables par machine de Turing [Kain 81 p.111]. Cette classe contient celle des langages contextuels (context-sensitive, type 1).

Les règles de type 1 sont de la forme monotone croissante (length increasing):

LCR --> LDR avec |D| ≥ |C|

Les grammaires ainsi formées sont faiblement équivalentes à des grammaires strictement contextuelles [Salomaa 73], dans lesquelles |C| = 1. Les règles strictement de type 0 utilisées dans les grammaires de langages rythmiques sont toujours de la forme:

A ... A (N fois) --> AN

dans laquelle AN représente une variable qui sera dérivée ainsi:

AN => ... => a1...aN

où les symboles ai sont des terminaux du langage. Ces dérivations sont donc équivalentes à celles des grammaires monotones croissantes.

Les langages sont en général décrits par dérivations successives dans plusieurs grammaires transformationnelles sans règle d'effacement (phrase-structure transformational grammars), description qui reflète la division en plusieurs étapes du processus de synthèse ou de reconnaissance en plusieurs étapes. Nous définissons ces grammaires comme suit:

Une grammaire transformationnelle sans règle d'effacement G est un quintuplet ordonné (VE,VN,VT,VD,F) tel que:

$V = VE \cup VN \cup VT \cup VD$, $VT \neq \emptyset$, $VD \neq \emptyset$
{VE,VN,VT,VD} est une partition de V,
et F est un ensemble fini de couples ordonnés (LCR, LDR) tel que:
$C \in (VN \cup VT \cup VD)+$, $L \in V*$, $R \in V*$ et $D \in (VN \cup VT)+$

VD est l'alphabet de départ, VT l'alphabet terminal, VN l'alphabet intermédiaire (les variables), VE l'alphabet extérieur à G et F l'ensemble des règles de production.

L'alphabet de départ d'une grammaire est nécessairement inclus dans la réunion des alphabets terminaux des grammaires précédentes, sauf pour la première où VD = {S}. Si le langage est décrit par n grammaires, l'alphabet terminal de Gn est inclus dans l'alphabet terminal du langage. Nous convenons d'autre part que l'alphabet extérieur de G1 est vide et que tout symbole de l'alphabet extérieur d'une grammaire Gi est un terminal d'une grammaire Gj avec j < i.

Les symboles terminaux qui ne représentent pas des bols (voir 2) sont appelés symboles structurels:

les parenthèses et pointeurs de répétition: (=) (:)
le symbole de miroir: *
les entiers (1..99) qui représentent la vitesse courante
des symboles spéciaux choisis parmi: {+ : ; = }

4. Génération de phrases

La génération de phrases utilise l'algorithme suivant:

```
Procédure GENERER
début
n := 1
tant qu'il existe une grammaire Gn, répéter:
      début
      tant qu'une règle est appliquable, répéter:
            début
            choisir une règle appliquable
            choisir une position de dérivation
            si le membre gauche de la règle est reconnu
                  alors remplacer l'occurrence trouvée par le membre
droit
            fin
      n := n+1
      fin
```

fin

4.1 Le choix des règles

Dans les grammaires ORD, la règle choisie est en priorité la règle précédemment appliquée, à défaut la première règle candidate dans l'ordre croissant. Dans les grammaires LIN et RND, le choix d'une règle se fait par un tirage pondéré par les poids des règles candidates.

4.2 La position de dérivation

Si la règle comporte l'instruction LEFT, ou si elle appartient à une grammaire LIN, la position de dérivation est la position la plus à gauche qui permet de faire coïncider la partie gauche de la règle avec la phrase en cours de dérivation. Dans tous les autres cas la position est aléatoire.

5. Reconnaissance de phrases (grammaires LIN)

L'algorithme déterministe de test d'appartenance est le suivant:

(1) Etant donnée une grammaire transformationnelle G, permuter les membres gauche et droit de toutes les règles de G. Soit G' la grammaire ainsi obtenue (grammaire duale [Salomaa 73]).

(2) Sachant que le langage est généré par l'application des grammaires transformationnelles G1, ... Gn dans cet ordre, on effectue la dérivation canonique à droite de la chaîne analysée par G'n, ... G'1.

(3) La phrase appartient au langage si la dérivation se termine avec le symbole de départ S.

Si un mot u est dérivé d'un mot v par dérivation canonique à droite dans une grammaire sans règle d'effacement, alors v est dérivé de u par dérivation canonique à gauche dans la grammaire duale [Hart 76 p.245].

L'intérêt de cet algorithme est multiple: il est économique en espace mémoire et le nombre de réécritures est inférieur à la longueur de la chaîne analysée. La seule difficulté est de définir la dérivation canonique, ainsi que le domaine d'application de l'algorithme, c'est à dire la classe des langages dont il reconnaît toute phrase.

5.1 Dérivation canonique contextuelle à droite: définition

Soit G une grammaire contextuelle. La dérivation de G :

$$W0 \Rightarrow W1 \Rightarrow \ldots \Rightarrow Wn$$

est canonique contextuelle à droite (context-sensitive rightmost canonic) si:

$\forall\ i \in [0, n-1]$, $Wi = XiLiCiRiYi$ et $Wi+1 = XiLiDiRiYi$ en appliquant la règle fi:

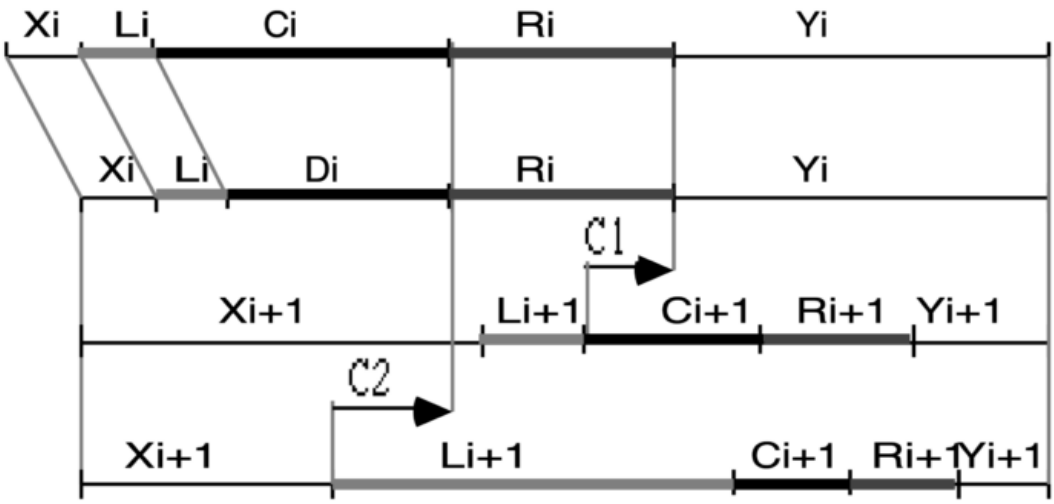
$LiCiRi \rightarrow LiDiRi$,

et l'une au moins des conditions suivantes est réalisée :

$$|Ci+1Ri+1Yi+1| > |Yi| \qquad (C1)$$

$$|Li+1Ci+1Ri+1Yi+1| > |RiYi| \qquad (C2)$$

La figure ci-dessous illustre les conditions (C1) et (C2) que nous allons justifier.



Supposons que les conditions (C1) et (C2) ne soient pas respectées. La règle fi+1 aurait pu être appliquée avant fi puisque Li+1Ci+1Ri+1 serait une sous-chaîne de RiYi. D'autre part, l'application de la règle fi+1 ne modifierait que la chaîne Yi sans toucher au contexte Ri. Dans ce cas, l'ordre de fi et de fi+1 pourrait être inversé. Cette inversion serait justifiée puisque les symboles réécrits par fi+1 se trouvent à droite de ceux réécrits par fi.

Pour une grammaire strictement contextuelle, |Ci| = |Ci+1| = 1, et la condition (C1) peut s'écrire |Ri+1Yi+1| ≥ |Yi|, ce qui revient à la définition de Hart [Hart 80 p.82]. D'autre part, si (C1) ou (C2) est réalisée, alors la condition |Li+1Ci+1Ri+1Yi+1| > |Yi|, qui définit la dérivation canonique à droite d'une grammaire de type 0 sans règle d'effacement, est aussi réalisée [Hart 76 p.246].

5.2 Critères de sélection des règles

Dans le Bol Processor, nous avons défini des critères qui permettent d'ordonner les règles afin de réaliser en analyse la dérivation canonique à droite. Si deux règles sont candidates:

```
      LiCiRi      -->  LiDiRi                  (fi)
et    L'iC'iR'i   -->  L'iD'iR'i               (f'i)
avec  XiLiCiRiYi  =  X'iL'iC'iR'iY'i  =  Wi
```

(fi) sera choisie en priorité sur (f 'i) si l'un des critères suivants, dans l'ordre, est vérifié:

```
|XiLiCi|  > |X'iL'iC'i|            (D1)
|XiLiCiRi|  > |X'iL'iC'iR'i|       (D2)
|LiCiRi|  > |L'iC'iR'i|            (D3)
i > i'                             (D4)
```

Nous avons montré [Bel 87a] que les critères (D2) et (D3) pouvaient être ignorés sous deux conditions. La première constitue la règle des chunks:

Le membre droit d'une règle fi ne peut être une sous-chaîne de celui d'une règle fj telle que j < i.

La seconde est une condition suffisante (mais non nécessaire) pour que (D1) et (D4) respectent les dépendances de contexte dans le graphe syntaxique contextuel [Révész 85 p.181, Bel 87a]:

Un contexte à droite ne peut contenir que des symboles de l'alphabet extérieur à la grammaire.

Cette règle est une justification de l'usage des grammaires transformationnelles, où la notion d'alphabet extérieur est reconnue (voir 3).

6. Domaine d'application de l'algorithme déterministe

Nous allons montrer comment modifier une grammaire avec chevauchement de motifs, et qui ne génère pas un langage reconnu par l'algorithme décrit en (5):

LIN			
S	<->	A A A A A A A A A A	(f1)
A	<->	A1	(f2)
A A	<->	A2	(f3)
A A A	<->	A3	(f4)
A1	<->	-	(f5)
A2	<->	dha -	(f6)
A2	<->	- dha	(f7)
A3	<->	te te dha	(f8)
A3	<->	dha te te	(f9)

La règle des chunks impose l'ordre (f5,f6). La phrase correcte suivante:

te te dha te te dha - dha te te

sera dérivée, selon les critères (D1) et (D4), de la manière suivante:

te te dha te te dha - dha te te => te te dha te te dha - A3 => te te dha te te dha - A A A =>

te te dha te te A2 A A A => te te dha te te A A A A A => te te A3 A A A A A =>

te te A A A A A A A A ... échec

La solution, pour que l'algorithme déterministe soit appliquable, consiste à examiner toutes les configurations possibles de concaténation de deux motifs avec leur segmentation correcte:

- | -, - | dha -, - | - dha, - | te te dha, - | dha te te

dha - | -, dha - | dha -, dha - | - dha, dha - | te te dha, dha - | dha te te

- dha | -, - dha | dha -, - dha | - dha, - dha | te te dha, - dha | dha te te

te te dha | -, te te dha | dha -, te te dha | - dha, te te dha | te te dha, te te dha | dha te te

dha te te | -, dha te te | dha -, dha te te | - dha, dha te te | te te dha, dha te te | dha te te

La concaténation soulignée est celle qui sera mal segmentée et conduira à un échec lors de l'analyse. Il faut donc ajouter les règles:

A A A A	<->	A4	(f10)
A4	<->	te te dha -	(f11)

et recommencer le processus si nécessaire avec le nouveau mot. Cette manière de procéder est très proche de celle du musicologue qui segmente les phrases en mots simples puis définit des mots composés représentant des motifs plus larges, et peut aussi être réalisée par logiciel [Bel 87b].

Le domaine d'application de l'algorithme déterministe de test d'appartenance peut maintenant être défini formellement:

Les grammaires d'un langage L qui permettent d'utiliser l'algorithme déterministe possèdent les propriétés suivantes :

(1) Toute règle est de la forme : Li Ci Ri --> Li Di Ri (fi)
avec |Ci| ≥ 1 et |Di| ≥ 1
(2) Soit M = {Di membre droit de (fi) avec Di inclus dans VT+}.
(M est l'ensemble des mots)
∃ u,v ∈ M tels que xuvy ∈ L avec x,y ∈ M*,
Si ∃ v' ∈ M tel que uv = u'v' avec u' ∈ VT+ et |v'| > |v|
alors ∃ w ∈ M tel que w = uv

Si u' ∈ de M, la création du mot composé w lève l'ambiguïté entre uv et u'v'. Partant d'une grammaire ne possédant pas la propriété (2), on peut ajouter des règles jusqu'à ce que la propriété soit vérifiée. Ce problème est lié à celui de la segmentation [Bel 87b].

7. Reconnaissance de phrases (grammaires RND et ORD)

L'algorithme ci-dessous, qui donne priorité à l'ordre des règles (critère D4) sur la position de dérivation (critère D1) remplace la dérivation canonique contextuelle à droite dans les grammaires RND et ORD:

```
Procédure REECRIRE(i)
début
chercher le membre droit de fi dans la phrase à partir de la droite
```

```
remplacer l'occurrence trouvée par le membre gauche de fi
fin

Procédure SATURER(i)
début
tant que la règle fi est applicable, répéter:
      début
      REECRIRE(i)
      test := 1
      fin
fin

Procédure ANALYSER
début
répéter:
      début
      i := numéro de la dernière règle
      test := 0
      répéter:
            début
            SATURER(i)
            i := i - 1
            fin
      jusqu'à ce que (i=0) ou (test=1)
      fin
jusqu'à ce que (test=0)
fin
```

Cet algorithme plus rapide est utilisable si la double condition (suffisante) est respectée: tous les contextes gauches et droits des règles n'utilisent que des symboles de l'alphabet extérieur, et les motifs des membres droits ne se chevauchent pas (voir 7). La règle des chunks doit aussi toujours être observée.

8. Exemple de qai‘da: structure et vocabulaire

Nous allons montrer, à partir d'un exemple réel, comment formuler des règles de production aussi générales que possible, puis comment modifier la grammaire pour tenir compte des contraintes d'accentuation qui sont mises en évidence par l'expérimentation. Le qai‘da en question est interprété sur une mesure à 16 temps, chaque temps étant divisé en 6 parties égales. Les exemples de variations donnés en annexe sont tabulés à raison de 4 temps (24 bols) par ligne. L'exemple [1] est le prototype de la composition, que l'on peut formuler comme suit en utilisant les conventions de représentation définies au paragraphe (2):

```
(= KP) CB * (: KP) CK
```

d'où l'on tire les règles de production des motifs fixes KP, CB et CK:

```
KP    <-> dhin--dhagenadha--dhagenadhatigegenakadhinedhinaghina
CK    <-> tagetirakitadhin--dhagenadhatigegenakadhinedhinaghina
etc. voir  grammaire n°5 en annexe
```

La même grammaire définit des variantes de CB et CK auxquelles sont attachés des poids plus faibles: <10> au lieu de <100>.

Les variations de type S1V remplacent KP par une chaîne de motifs E24 qui doit être répétée en miroir dans la deuxième partie. Les variations S2V couvrent deux mesures complètes. Les structures de ces variations sont définies par les règles 3 et 4 de la grammaire n°2:

```
S2V   <-> (= KP CB) (= E24) CB * (: KP CB) (: E24) CK
S1V   <-> (= E24) CB * (: E24) CK
```

Les autres règles de la grammaire n°2 (voir annexe) servent à définir des répétitions ou des miroirs utilisés par certaines variantes de S1V et S2V.

Pour l'alphabet intermédiaire, nous utilisons en général la convention suivante: An, Bn, etc., représentent des variables dont l'instanciation est une chaîne de symboles terminaux de longueur n. Les mots (ou motifs déplaçables) ont été dans un premier temps listés comme suit:

```
A3    <-> dhin--
A3    <-> dha--
A3    <-> dhagena
A4    <-> tirakita
A6    <-> dhagenadhin--
A6    <-> dhagenadha--
A6    <-> dha-dha-dha-
A6    <-> dha-ta-dha-
A6    <-> dhinedhinedhine
A6    <-> dhinedha-dhine
A6    <-> tagetirakita
A6    <-> dhinedhinaghina
A6    <-> tinetinakina
A6    <-> dhatigegenaka
A12   <-> dhatigegenakatinetinakina
```

9. Exemple (suite): permutations

La grammaire n°3 doit transformer les variables B24, B12 ou B6 (symboles terminaux de la grammaire n°2) en séquences de variables A3, A4, A6 et A12 (symboles de départ de la grammaire n°4). Il est intéressant de retracer le processus qui a permis de la formaliser. La première grammaire envisagée était la suivante:

```
LIN
B6  <-> A A A A A A
B12 <-> A A A A A A A A A A A A
B24 <-> A A A A A A A A A A A A A A A A A A A A A A A A
A A A <-> A3
A A A A <-> A4
A A A A A A <-> A6
A A A A A A A A A A A A <-> A12
```

Cette grammaire est correcte au sens où toute suite de symboles A3, A4, et A6 est une dérivation de B24 si la somme des indices égale 24. Toutefois, certaines dérivations de B24 aboutissent à un cul-de-sac, comme par exemple:

```
A3 A6 A3 A4 A3 A4 A A
```

Il est facile de constater qu'il ne reste jamais plus de deux A isolés. Pour les éliminer, l'analyste réalise une 'passerelle' dans l'arbre des dérivations en reliant tous les culs-de-sac à un chemin favorable. Il suffit pour cela d'insérer une nouvelle grammaire entre n°3 et n°4:

```
ORD
A A         --> A2
A3 A        --> A4
#A3 A       --> A #A3   [Contexte négatif A3]
A6 A2       --> A4 A4
#A6 A2      --> A2 #A6  [Contexte négatif A6]
```

?Ces règles permettent de déplacer vers la gauche les A isolés puis de les absorber. Dans le cas présent, l'élimination est toujours possible. Les règles ci-dessus ne sont pas utilisées en analyse (flèche simple "-->").

10. Accentuation

L'expérimentation montre qu'un grand nombre de variations synthétisées par la grammaire précédente ne sont pas acceptables. C'est le cas de l'exemple [5] cité en annexe. En fait, la plupart des mots ne peuvent pas être permutés arbitrairement: cette composition étant interprétée avec un tempo très rapide, l'accentuation doit tomber sur un temps ou un demi-temps. (Chaque temps est divisé en six unités.) De nouvelles variables doivent être choisies pour désigner les mots: chacune d'elles représente maintenant une suite d'événements qui débute sur un temps fort. Dans le cas particulier de dhagenadhin--, l'accentuation peut tomber sur dha aussi bien que sur dhin. Ce mot est donc représenté par "A3 A3" et non plus par A6. Certains motifs de six unités, d'autre part, obéissent à des contraintes particulières de contexte. Les mots correspondants sont désignés par la variable D6.

La grammaire n°4 devient:

```
RND
A3    <-> dhin--
A3    <-> dha--
A3    <-> dhagena
A3 A3 <-> dhagenadhin--
A3 A3 <-> dhagenadha--
A4    <-> tirakita
D6    <-> dha-dha-dha-
D6    <-> dha-ta-dha-
D6    <-> dhinedha-dhine
A6    <-> dhinedhinedhine
A6    <-> tage...
etc...  Autres règles inchangées
```

Il reste à réécrire la grammaire n°3 en tenant compte de l'accentuation. C'est une grammaire linéaire à gauche qui permet d'exprimer cette contrainte. Dans un premier temps, on écrit toutes les règles de production qui permettent de découper B24 en une suite de A3, A4, A6 et D6. On élimine ensuite celles qui produisent des placements incorrects:

```
LIN
B24   <-> A3 B21
B24   <-> A4 B20
B24   <-> A6 B18
B24   <-> D6 B18
B21   <-> A3 B18
B21   <-> A4 B17  éliminée: accentuation sur le 8ème symbole
B21   <-> A6 B15
B21   <-> D6 B15  éliminée: accentuation incorrecte pour D6
etc...
```

Un nouveau problème se pose en synthèse: dans de nombreux cas, le Bol Processor produit des séquences "A4 A4 A4 A4 A4 A4" qui sont jugées inacceptables en raison de leur répétitivité. Ce problème est résolu en introduisant le contexte négatif #A4 dans la règle 27:

```
[27]  #A4 B12  <->  #A4 A4 B8
```

de sorte qu'on ne puisse produire (ni accepter) plus de trois A4 consécutifs.

?11. Gabarits (templates)

Une phrase n'est analysable que si elle est représentée à l'aide des parenthèses, pointeurs et marqueurs de genre (symboles structurels, voir 3). Il est possible toutefois d'insérer ces symboles en plaçant la phrase non structurée (telle que le musicologue l'a entrée) sur un gabarit (template). Les points représentent les emplacements (slots) des symboles terminaux:

```
(= (= ......) * (: ......) .............) ........................
etc.
```

Le Bol Processor est doté d'un dispositif permettant de générer tous les gabarits d'une grammaire donnée (voir en annexe la 'grammaire' n°6). Pour analyser une phrase, il suffit donc de l'entrer sans symbole structurel. Le moteur d'inférences lance l'analyse sur tous les gabarits compatibles.

12. Validation de règles et généralisation

La validation de règles peut se résumer ainsi: on remet à zéro les poids de toutes les règles des grammaires transformationnelles. Puis on soumet un échantillonnage représentatif de phrases (correctes) au test d'appartenance en faisant en sorte que, chaque fois qu'une règle est utilisée, son poids est incrémenté. Il est intéressant de reconsidérer les règles dont les poids restent nuls après l'analyse de nombreuses variations, car elles peuvent mettre en évidence des possibilités inexplorées par le musicien. On peut, pour statuer sur leur validité, revenir à la synthèse en leur donnant un poids

fort.

Cette méthode, qui permet, à partir d'une grammaire supposée contenir tout le langage, d'éliminer les dérivations conduisant à des phrases incorrectes, présuppose une certaine indépendance des règles: dans l'exemple ci-dessus, tous les motifs reconnus par la grammaire n°4 (voir annexe) devraient correspondre à des noms de variables différents. La grammaire n°3 contiendrait les règles produisant toutes les permutations possibles de ces variables, et la validation permettrait ensuite d'éliminer celles qui sont illicites.

Que l'on procède par création (apprentissage et raisonnement inductif) ou par validation de règles (raisonnement déductif), il est nécessaire, après chaque modification, d'unifier les variables qui conduisent aux mêmes dérivations, puis de supprimer les règles que l'unification a rendues redondantes [Bel 87b]. Ces deux techniques se rejoignent aussi sur le principe que toute généralisation ou particularisation de la grammaire doit pouvoir être remise en question si ses effets sont contredits par l'expérience. La modélisation de ces méthodes d'analyse, qui permettent au musicologue de gérer systématiquement sa connaissance du langage et les hypothèses qui ont servi à l'élaborer, constitue aujourd'hui l'essentiel du travail théorique sur ce projet.

Bibliographie

[Bel 87a] B. Bel: Les grammaires et le moteur d'inférences du Bol Processor; Groupe Représentation et Traitement des Connaissances, CNRS (Marseille), document 237, 1987

[Bel 87b] B. Bel: Grammaires de langages rythmiques; Mémoire de Mathématiques et Informatique; Groupe Intelligence Artificielle, Faculté des Sciences de Luminy (Marseille),1987

[Hart 76] J.M. Hart: Right and Left Parses in Phrase-Structure Grammars; Information & Control 32, pp.242-262, 1976

[Hart 80] J.M. Hart: Derivation Structures for Strictly Context-Sensitive Grammars; Information & Control 45, pp.68-89, 1980

[Kain 81] R. Kain: Automata Theory: Machines and Languages; Krieger (Malabar), 1981

[Kippen 86] J.R. Kippen: An Ethnomusicological Approach to the Analysis of Musical Cognition; Music Perception, University of California (San Diego), à paraître

[Révész 85] G.E. Révész: Introduction to Formal Languages; McGraw-Hill (Singapore), 1985

[Salomaa 73] A. Salomaa: Formal Languages; Academic Press (New York), 1973

Annexe: exemple de qa'ida (style de Lucknow)

L'alphabet terminal de la grammaire n°5 et les symboles structurels de la grammaire n°2 sont explicités au paragraphe (2). Les nombres entre crochets <...> représentent les poids des règles.

GRAM#1 [1] RND [Familles de variations]

```
GRAM#1 [2] <100>  LEFT  S <-> S2V   [double]
GRAM#1 [3] <100>  LEFT  S <-> S1V   [simple]
--------------------------------------------------------------------------------
GRAM#2 [1]              RND         [Structures]
GRAM#2 [2] <1>          LEFT  E24 <-> KP
GRAM#2 [3] <100>  LEFT  S2V <-> (= KP CB ) (= E24 ) CB *(: KP CB ) (: E24 ) CK
GRAM#2 [4] <100>  LEFT  S1V <-> (= E24 ) CB *(: E24 ) CK
GRAM#2 [5] <100>  LEFT  E24 <-> B24
GRAM#2 [6] <33>   LEFT  E24 <-> E12 E12
GRAM#2 [7] <33>   LEFT  E24 <-> (= E12 ) (: E12 )
GRAM#2 [8] <33>   LEFT  E24 <-> (= E12 ) *(: E12 )
GRAM#2 [9] <100>  LEFT  E12 <-> B12
GRAM#2 [10] <50>  LEFT  E12 <-> (= B6 ) *(: B6 )
--------------------------------------------------------------------------------
GRAM#3 [1]  LIN         [Permutations]
GRAM#3 [2] <100>        B6 <-> A6
GRAM#3 [3] <100>        B24 <-> A6 B18
GRAM#3 [4] <2>          B24 <-> D6 B18
GRAM#3 [5] <20>         B24 <-> A4 B20
GRAM#3 [6] <100>        B24 <-> A3 B21
GRAM#3 [7] <100>        B21 <-> A6 B15
GRAM#3 [8] <100>        B21 <-> A3 B18
GRAM#3 [9] <100>        B20 <-> A4 B16
GRAM#3 [10] <100>       B20 <-> A6 B14
GRAM#3 [11] <100>       B18 <-> A6 B12
GRAM#3 [12] <5>         B18 <-> D6 B12
GRAM#3 [13] <100>       B18 <-> A4 B14
GRAM#3 [14] <100>       B18 <-> A3 B15
GRAM#3 [15] <100>       B16 <-> A4 B12
GRAM#3 [16] <100>       B16 <-> A6 B10
GRAM#3 [17] <100>       B15 <-> A6 B9
GRAM#3 [18] <100>       B15 <-> A3 B12
GRAM#3 [19] <100>       B14 <-> A4 B10
GRAM#3 [20] <100>       B14 <-> A6 B8
GRAM#3 [21] <100>       B12 <-> A12
GRAM#3 [22] <100>       B12 <-> A6 B6
GRAM#3 [23] <5>         B12 <-> D6 B6
GRAM#3 [24] <5>         B12 <-> A6 D6
GRAM#3 [25] <100>       B12 <-> A3 B9
GRAM#3 [26] <100>       B10 <-> A6 A4
GRAM#3 [27] <100>       #A4 B12 <-> #A4 A4 B8   [Contexte négatif A4]
GRAM#3 [28] <100>       ) B12 <-> ) A4 B8
GRAM#3 [29] <5>         B10 <-> A4 D6
GRAM#3 [30] <100>       B9 <-> A3 B6
GRAM#3 [31] <5>         B9 <-> A3 D6
GRAM#3 [32] <100>       B9 <-> A6 A3
GRAM#3 [33] <100>       B8 <-> A4 A4
GRAM#3 [34] <100>       B6 <-> A3 A3
--------------------------------------------------------------------------------
GRAM#4 [1] RND          [Mots ou motifs déplaçables]
GRAM#4 [2] <30>   A3 <-> dhin--
GRAM#4 [3] <30>   A3 <-> dha--
GRAM#4 [4] <30>   A3 <-> dhagena
GRAM#4 [5] <100>  A3 A3 <-> dhagenadhin--
```

```
GRAM#4 [6] <100>  A3 A3 <-> dhagenadha--
GRAM#4 [7] <100>  A4 <-> tirakita
GRAM#4 [8] <50>   D6 <-> dha-dha-dha-
GRAM#4 [9] <50>   D6 <-> dha-ta-dha-
GRAM#4 [10] <10>  A6 <-> dhinedhinedhine
GRAM#4 [11] <100>       D6 <-> dhinedha-dhine
GRAM#4 [12] <100>       A6 <-> tagetirakita
GRAM#4 [13] <100>       A6 <-> dhinedhinaghina
GRAM#4 [14] <10>  A6 <-> tinetinakina
GRAM#4 [15] <100>       A6 <-> dhatigegenaka
GRAM#4 [16] <100>       A12 <-> dhatigegenakatinetinakina
--------------------------------------------------------------------------------
GRAM#5 [1]  RND         [Motifs fixes]
GRAM#5 [2] <100>  LEFT  KP <-> dhin--dhagenadha-
-dhagenadhatigegenakadhinedhinaghina
GRAM#5 [3] <100>  LEFT  CK <-> tagetirakitadhin-
-dhagenadhatigegenakadhinedhinaghina
GRAM#5 [4] <10>   LEFT  CK <-> tagetirakitaghina-
dhagenadhatigegenakadhinedhinaghina
GRAM#5 [5] <100>  LEFT  CB <-> tagetirakitadhin-
-dhagenadhatigegenakatinetinakina
GRAM#5 [6] <10>   LEFT  CB <-> tagetirakitaghina-
dhagenadhatigegenakatinetinakina
--------------------------------------------------------------------------------
```

Les gabarits (templates) générés par cette grammaire:

```
GRAM#6 [1]              TEM         [templates]
GRAM#6 [2] (=.............................................)
(=.......................) .......................
*(:.............................................)
(:.......................) .......................
GRAM#6 [3] (=.............................................) (=............
(=......) *(:......) ) ......................
*(:.............................................) (:............ (=......)
*(:......) )......................
GRAM#6 [4] (=.............................................) (= (=......)
*(:......) ............) ......................
*(:.............................................) (: (=......) *(:......)
............)......................
GRAM#6 [5] (=.............................................) (= (=......)
*(:......) (=......) *(:......))......................
*(:.............................................) (: (=......) *(:......)
(=......) *(:......) ) ......................
GRAM#6 [6] (=.............................................) (=
(=............) (:............) ) ......................
*(:.............................................) (: (=............)
(:............) ) ......................
GRAM#6 [7] (=.............................................) (= (= (=......)
*(:......) ) (: (=......) *(:......) ) ) ......................
*(:.............................................) (: (= (=......)
*(:......) ) (: (=......) *(:......) ) ) ......................
GRAM#6 [8] (=.............................................) (=
(=............) *(:............) ) ......................
```

```
*(:.....) (: (=.....)
*(:.....) ).....
GRAM#6 [9] (=.....) (= (= (=......)
*(:......) ) *(: (=......) *(:......) ) ) .....
*(:.....) (: (= (=......)
*(:......) ) *(: (=......) *(:......) ) ) .....
GRAM#6 [10] (=.....) .....
*(:.....) .....
GRAM#6 [11] (=..... (=......) *(:......) ) .....
*(:..... (=......) *(:......)) .....
GRAM#6 [12] (= (=......) *(:......) .....) ..... *(:
(=......) *(:......) .....) .....
GRAM#6 [13] (= (=......) *(:......) (=......) *(:......) )
..... *(: (=......) *(:......)(=......) *(:......))
.....
GRAM#6 [14] (= (=.....) (:.....) ) ..... *(:
(=.....) (:.....) ) .....
GRAM#6 [15] (= (= (=......) *(:......) ) (: (=......) *(:......) ) )
..... *(: (= (=......)*(:......) ) (: (=......) *(:......)
) ) .....
GRAM#6 [16] (= (=.....) *(:.....) ) ..... *(:
(=.....) *(:.....)) .....
GRAM#6 [17] (= (= (=......) *(:......) ) *(: (=......) *(:......) ) )
..... *(: (= (=......) *(:......) ) *(: (=......)
*(:......) ) ) .....
?Exemples de variations
```

Prototype du qa'ida:

```
[1]   dhin--dhagena     dha--dhagena      dhatigegenaka     dhinedhinaghina
      tagetirakita      dhin--dhagena     dhatigegenaka     tinetinakina
      tin--takena ta--takena  tatikekenaka      tinetinakina
      tagetirakita      dhin--dhagena     dhatigegenaka     dhinedhinaghina

soit: (=dhin--dhagena   dha--dhagena      dhatigegenaka     dhinedhinaghina )
      tagetirakita      dhin--dhagena     dhatigegenaka     tinetinakina
      *(:tin--takena    ta--takena  tatikekenaka      tinetinakina )
      tagetirakita      dhin--dhagena     dhatigegenaka     dhinedhinaghina
```

Variation simple :

```
[2]   dhin--dhatige     genakadhin--      tirakitatira      kitatirakita
(=E24)
      tagetirakita      dhin--dhagena     dhatigegenaka     tinetinakina
      CB
      tin--tatike       kenakatin--       tirakitatira      kitatirakita
*(:E24)
      tagetirakita      dhin--dhagena     dhatigegenaka     dhinedhinaghina
      CK

soit: (=dhin--dhatige   genakadhin--      tirakitatira      kitatirakita )
      tagetirakita      dhin--dhagena     dhatigegenaka     tinetinakina
      *(:tin--tatike    kenakatin--       tirakitatira      kitatirakita )
```

tagetirakita dhin--dhagena dhatigegenaka dhinedhinaghina

Variations doubles:

[3] dhin--dhagena dha--dhagena dhatigegenaka dhinedhinaghina
(=KP
tagetirakita dhin--dhagena dhatigegenaka tinetinakina
CB)
dhin--dhagena dha-dha-dha- dhatigegenaka tinetinakina
(=E24)
tagetirakita dhin--dhagena dhatigegenaka tinetinakina
CB
tin--takena ta--takena tatikekenaka tinetinakina *(:KP
taketirakita tin--takena tatikekenaka tinetinakina
CB)
dhin--dhagena dha-dha-dha- dhatigegenaka tinetinakina
(:E24)
tagetirakita dhin--dhagena dhatigegenaka dhinedhinaghina
CK

[4] dhin--dhagena dha--dhagena dhatigegenaka dhinedhinaghina
tagetirakita dhin--dhagena dhatigegenaka tinetinakina
dhinedhinaghina dhinedha-dhine dhatigegenaka tinetinakina
tagetirakita dhin--dhagena dhatigegenaka tinetinakina
tin--takena ta--takena tatikekenaka tinetinakina
taketirakita tin--takena tatikekenaka tinetinakina
dhinedhinaghina dhinedha-dhine dhatigegenaka tinetinakina
tagetirakita dhin--dhagena dhatigegenaka dhinedhinaghina

[5] dhin--dhagena dha--dhagena dhatigegenaka dhinedhinaghina
tagetirakita dhin--dhagena dhatigegenaka tinetinakina
dhin--tiraki tadhagenadhati gegenakatira kitatirakita
tagetirakita dhin--dhagena dhatigegenaka tinetinakina
tin--takena ta--takena tatikekenaka tinetinakina
taketirakita tin--takena tatikekenaka tinetinakina
dhin--tiraki tadhagenadhati gegenakatira kitatirakita
tagetirakita dhin--dhagena dhatigegenaka dhinedhinaghina

(Variation incorrecte: l'accentuation de dha est mal placée)