Modelling music with grammars: formal language representation in the Bol Processor

Bernard Bel, Jim Kippen

groupe
représentation
et traitement
des
connaissances

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

31, chemin Joseph Aiguier

F-13402 MARSEILLE CEDEX 9 (France)

MODELLING MUSIC WITH GRAMMARS: FORMAL LANGUAGE REPRESENTATION IN THE BOL PROCESSOR

Jim Kippen & Bernard Bel

Abstract

Improvisation in North Indian tabla drumming is similar to speech insofar as it is bound to an underlying system of rules determining correct sequences. The parallel is further reinforced by the fact that tabla music may be represented with an oral notation system used for its transmission and, occasionally, performance. Yet the rules are implicit and available only through the musicians' ability to play correct sequences and recognise incorrect ones. A linguistic model of tabla improvisation and evaluation derived from pattern languages and formal grammars has been implemented in the Bol Processor, a software system used in interactive fieldwork with expert musicians. The paper demonstrates the ability of the model to handle complex structures by taking real examples from the repertoire. It also questions the relevance of attempting to model irregularities encountered in actual performance.

Keywords

Formal grammars, patterns, membership test, rhythm, tabla, drumming, ethnomusicology

GRTC / 318 / octobre 1988

Jim Kippen & Bernard Bel

A. Marsden & A. Pople (eds.): Computer Representations and Models in Music, London, Academic Press, 1992, pp.207-38.

† This research has been partly funded by the Leverhulme Trust.

* Research Fellow
Department of Social Anthropology and
Ethnomusicology
The Queen's University of Belfast (U.K.)
E-mail: eihe4874@uk.ac.qub.v1 (JANET)

** Engineer in Computer Science
Groupe Représentation et Traitement des Connaissances
Centre National de la Recherche Scientifique
Marseille (France)
E-mail: grtc@frmop11.bitnet (EARN)

Modelling music with grammars: formal language representation in the Bol Processor †

Jim Kippen and Bernard Bel***

Introduction

The very nature of the music of the *tabla* (North Indian two-piece tuned drum set) suggests that linguistics may be an effective analytical approach to the investigation of its structure, and grammars the most appropriate models. An oral notation system using verbal symbols called *bol*s (from the Urdu/Hindi *bolna* 'to speak') is used for its transmission and, occasionally, performance: *dha, ti, ge, na, tirakita (trkt), dhee, tee, ta, ke* etc. Bols are quasi-onomatopoeic mnemonics that represent drum-strokes, although their aesthetic and technical 'meanings' (they have no semantic meaning per se) are nearly always context-bound with no one-to-one correlation between a stroke and its name (Kippen 1988a:xvi-xxiii). The equivalence of the verbal and musical representations is suggested by the fact that the word 'bol' refers both to a spoken syllable and a drum stroke.

A large part of the tabla repertoire involves improvisation. Any sequence of strokes may be considered as a finite string of symbols whose organisation is related to some implicit formal system. Automatic sequences (i.e. sequences generated by finite-state automata) share properties that place them somewhere between periodicity and chaos, yet closer to periodicity. Since both strict and approximate periodicity are essential features in music, it is realistic to think that strings of musical events may be appropriately represented with automata or formal grammars. In other words, the most fundamental reason for us to view formal language models as relevant models of music lies in properties that are intrinsic to music, rather than in analogical links between music and natural languages.

The following is an example of a compositional type known as *qa'ida*, the 'theme and variations' form *par excellence*. It should be read linearly from left to right, and each group represents a beat comprising four units (*trkt* is a compound stroke of two units).

dhatidhage	nadhatrkt	dhatidhage	dheenagena
dhatidhage	nadhatrkt	dhatidhage	teenakena
tatitake	natatrkt	tatitake	teenakena
dhatidhage	nadhatrkt	dhatidhage	dheenagena

Set in sixteen beats, this qa'ida comprises four four-beat statements of a basic 'sentence'. In accordance with the voiced/unvoiced structure of the metric cycle, the first and fourth statements are not inflected while part of the second and all of the third undergo transformations: *dha/ta, ge/ke, dhee/tee*. Variations are derived from the 'theme', and involve the repetition, permutation, or substitution of bols. Students are taught that changes occurring in the first half of the structure must be reflected in the second; variations, too, are subject to voiced/unvoiced transformations (see Kippen 1987:180-81), as can be seen in the following three variations (changes have been italicised; hyphens in the third variation represent silences of one unit, though effectively they elongate the preceding syllable):

† This research has been partly funded by the Leverhulme Trust.

* Research Fellow
Department of Social Anthropology and Ethnomusicology
The Queen's University of Belfast (U.K.)
E-mail: eihe4874@uk.ac.qub.v1 (JANET)

** Engineer in Computer Science
Groupe Représentation et Traitement des Connaissances
Centre National de la Recherche Scientifique
Marseille (France)
E-mail: grtc@frmop11.bitnet (EARN)

dhatidhage	nadhatrkt	dhatidhage	dheenagena
dhatrktdha	*tidhagena*	dhatidhage	teenakena
tatitake	natatrkt	tatitake	teenakena
dhatrktdha	*tidhagena*	dhatidhage	dheenagena
dhatidhatr	*ktdhatidha*	*trktdhage*	*nadhagena*
dhatidhage	nadhatrkt	dhatidhage	teenakena
tatitatr	*kttatita*	*trkttake*	*natakena*
dhatidhage	nadhatrkt	dhatidhage	dheenagena
dhagenadha	*trktdhage*	*nadhatrkt*	*dhatrktdha*
tidha-dha	*tidhagena*	dhatidhage	teenakena
takenata	*trkttake*	*natatrkt*	*tatrktta*
tidha-dha	*tidhagena*	dhatidhage	dheenagena

This paper traces the development of a formal-language representation for tabla music and its implementation in a computer system. The discussion will take as its starting point the musical examples given above. It should be noted that the grammatical models described below follow an approximate chronological order of development, even though many formal representations were evolved simultaneously.

Pattern grammars

Variations obey strict rules of construction, but these rules are rarely expressed formally by traditional musicians; rather, an implicit model is transmitted by means of sequences of positive instances of the 'language'. Negative instances composed by their students during the course of training are rejected or corrected. In a performance, a musician normally creates between six and ten variations for any qa'ida. These explore only a tiny subset of all possible variations, and thus it is unrealistic to attempt to elicit a complete set as this could run to many thousands of correct pieces. Therefore, when beginning this research we considered that a more prudent approach would be to construct a descriptive generalisation based on observed recurrent sequences of bols and patterns. For instance, it may be seen in the above variations that *dhatidhagedheenagena* occurred periodically and may thus be attributed a label A8 (the numbers attached to labels denote the lengths (in time units) of the terminal strings they represent). Furthermore, the inner structure of each variation may be represented as a *pattern* (a string of constants and variables — see Bel 1988:1-2). It is possible to preserve these structures because derivations in pattern languages can be represented in a special grammatical format. For instance, a production rule that we notate

A —> (=B) (:B)

implies that both derivations of B must be identical. In this pattern format, (=) denotes a referential sequence and (:) a copy. Several levels of brackets can also be embedded to represent complex patterns. An asterisk '*' indicates that the bracketed expression immediately following it should be derived as a string of unvoiced bols. Thus the initial formal representation we developed, as applied to the first variation of the qa'ida, was written:

(=dhatidhage	nadhatrkt	dhatidhage	dheenagena)
(=dhatrktdha	tidhagena)	(=dhatidhage	teenakena)
*(:tatitake	natatrkt	tatitake	teenakena)
(:dhatrktdha	tidhagena)	(=dhatidhage	dheenagena)

This was then reduced to the more concise expression

(=A16) (=V8) (=A'8) *(:A16) (:V8) (=A8)

in which A16, A8, and A'8 were fixed patterns, and V8 a permutation of bols.

A *pattern grammar* which described the three variations presented was:

GRAM#1

[1] S —> (=A16) (=V8) (=A'8) *(:A16) (:V8) (=A8)
[2] S —> (=V16) (=A'16) *(:V16) (=A16)
[3] S —> (=V24) (=A'8) *(:V24) (=A8)
[4] A16 —> dhatidhagenadhatrktdhatidhagedheenagena
[5] A'16 —> dhatidhagenadhatrktdhatidhageteenakena
[6] A8 —> dhatidhagedheenagena
[7] A'8 —> dhatidhageteenakena

in which the first three rules indicated options for the transformation of a starting symbol S. This grammar was implemented in a knowledge-based system called the *Bol Processor* (BP). The computer software comprises an editor designed to represent bols, variables, and conventional symbols (brackets, asterisk, etc,) as tokens. The dependancy (=B)(:B) is taken care of by pointers so that any modification in (=B) is reflected identically in (:B). A special table is also defined that indicates the voiced/unvoiced mapping. The 'active' element of the BP is an *inference engine* that uses an enumerative/stochastic process to generate sentences derived from the grammar, and a *membership algorithm* to check whether or not a sentence entered into the editor is consistent with the grammar. In the generative process (synthesis or *modus ponens*) the BP is able either to enumerate all sentences of the language or to generate one sentence randomly. For modus ponens, the order in which rules are placed (e.g. as shown in GRAM#1) is not important. This type of representation in which the specification of the problem ('infer any sentence from this set of rules') can clearly be separated from the method of solution ('select candidate rules and rewrite the current work string') is known as *declarative*.

The derivation of sentences may be viewed as a number of distinct stages involving several transformational grammars (*subgrammars*). Breaking a process into several subprocesses is basically a *procedural* approach. The word 'transformational' is borrowed from formal language theory (Bel 1987a:356), not linguistics. In a subgrammar there is more than one starting symbol, and symbols that are terminal to that subgrammar may become the starting symbols of the next subgrammar(s) to be applied, and so on. Subgrammars subsequent to GRAM#1 should describe acceptable derivations of the terminal symbols V8, V16, and V24.

Context-sensitive derivations in synthesis

Based on observations of collected data, and our understanding of the often incomplete and ambiguous statements of musicians, we hypothesized that V8, V16, and V24 were strings of 'words' which could be listed as follows:

V3 = dhagena
V2 = trkt
V1 = dha, ti, -

To describe all possible sequences we wrote:

```
V8      —> V V V V V V V V
V16     —> V V V V V V V V V V V V V V V V
V24     —> V V V V V V V V V V V V V V V V V V V V V V V V
V       —> V1
V V     —> V2
V V V   —> V3
```

It appeared that any arrangement of V2 and V3 was possible, but that the choice of derivations for V1 was highly context-sensitive. Three conditions were established based on both aesthetic and technical (i.e. fingering) criteria:

(1) there should be no more than two consecutive *dha*;
(2) there should be no more than two consecutive '-';
(3) *ti* should not appear before or after *ti* or *trkt*.

Therefore a subgrammar describing acceptable derivations of V1 was written:

```
dha V1 dha    —> dha ti dha
dha V1 -      —> dha ti -
dha V1 )      —> dha ti )
dha V1 V1     —> dha ti V1
na V1 dha     —> na ti dha
na V1 -       —> na ti -
na V1 )       —> na ti )
na V1 V1      —> na ti V1
- V1 dha      —> - ti dha
- V1 -        —> - ti -
- V1 )        —> - ti )
- V1 V1       —> - ti V1
(= V1 dha     —> (= ti dha
(= V1 -       —> (= ti -
(= V1 V1      —> (= ti V1

kt V1 tr      —> kt dha tr
kt V1 ti      —> kt dha ti
kt V1 -       —> kt dha -
kt V1 )       —> kt dha )
kt V1 V1      —> kt dha V1
na V1 tr      —> na dha tr
na V1 ti      —> na dha ti
na V1 -       —> na dha -
na V1 )       —> na dha )
na V1 V1      —> na dha V1
ti V1 tr      —> ti dha tr
ti V1 ti      —> ti dha ti
ti V1 -       —> ti dha -
ti V1 )       —> ti dha )
ti V1 V1      —> ti dha V1
- V1 tr       —> - dha tr
- V1 ti       —> - dha ti
- V1 -        —> - dha -
- V1 )        —> - dha )
- V1 V1       —> - dha V1
(= V1 tr      —> (= dha tr
(= V1 ti      —> (= dha ti
(= V1 -       —> (= dha -
(= V1 V1      —> (= dha V1

dha V1 tr     —> dha - tr
dha V1 dha    —> dha - dha
dha V1 ti     —> dha - ti
dha V1 )      —> dha - )
dha V1 V1     —> dha - V1
kt V1 tr      —> kt - tr
kt V1 dha     —> kt - dha
kt V1 ti      —> kt - ti
kt V1 )       —> kt - )
kt V1 V1      —> kt - V1
na V1 tr      —> na - tr
na V1 dha     —> na - dha
na V1 ti      —> na - ti
na V1 )       —> na - )
na V1 V1      —> na - V1
ti V1 tr      —> ti - tr
ti V1 dha     —> ti - dha
ti V1 ti      —> ti - ti
ti V1 )       —> ti - )
ti V1 V1      —> ti - V1
(= V1 tr      —> (= - tr
(= V1 dha     —> (= - dha
(= V1 ti      —> (= - ti
(= V1 V1      —> (= - V1
```

In this subgrammar all possible left and right contexts were accounted for. Brackets generated by GRAM#1 were retained to mark the beginning and end of a string. V1 was also considered to be a right context since it was obvious that derivations of V8, V16, and V24 could have resulted in strings of consecutive V1s. Apart from the additional bulk and complexity, the main defect of such a subgrammar was that it had both left and right contexts: therefore, the membership test of a sentence, although decidable, was costly in computation time. Evidently this kind of description was as unsatisfactory as it was user-unfriendly.

In the search for an alternative formal expression capable of satisfying the set of three simple criteria listed above, we introduced *negative contexts*. For example, the first condition was written:

```
#dha V1  —>  #dha dha
```

meaning 'V1 may be rewritten as *dha* unless it is preceded by *dha*'. Since this type of rule invoked the left context exclusively, it followed that strings should be constructed from left to right — more particularly, for any rule applied, the *leftmost* occurrence of V1 should be considered. To indicate this, an instruction 'LEFT' was placed in the rule. The derivations of V1 yielding *dha* and '-' were then:

```
[1]   LEFT #dha V1  —>  #dha dha
[2]   LEFT #- V1    —>  #- -
```

Derivations yielding ***ti***, on the other hand, were too complex to represent in a single production rule. We chose not to implement conjunctive expressions in negative contexts, such as

```
[#kt and #ti] V1 #tr      —>  [#kt and #ti] ti #tr
```

which expresses the third condition. Here it should be noted that *tr* cannot be a left context as it is always followed by *kt*, and ***ti*** cannot appear as a right context since V1s are derived from left to right. Instead we opted to represent positive contexts. Using positive contexts to express the third condition resulted in rules:

```
[3]   LEFT   dha V1 #tr  —>  dha ti #tr
[4]   LEFT   na V1 #tr   —>  na ti #tr
[5]   LEFT   - V1 #tr    —>  - ti #tr
[6]          (= V1 #tr   —>  (= ti #tr
```

A new problem arose because any negative context could be instantiated with V2 or V3. For example, a feasible derivation of *dhagena V1 V2* was *dhagenati V2* (using rule 4) and then *dhagenatitrkt*, which was incorrect but nevertheless possible since the derivation of V2 was context-free. Therefore, as a general procedure it was clear that any variable or terminal used as a right context should not be transformed in the same subgrammar in which it appeared as a context. And so we arrived at the more concise grammar listed below. For the sake of clarity the first six rules were written in two distinct subgrammars: GRAM#2 which is context-free, and GRAM#3 which is length-decreasing (type 0).

```
GRAM#2
[1]   V8    —>  V V V V V V V V
[2]   V16   —>  V V V V V V V V V V V V V V V V
[3]   V24   —>  V V V V V V V V V V V V V V V V V V V V V V V V V V V V
```

```
GRAM#3
[1]   V      —>  V1
[2]   V V    —>  trkt
[3]   V V V  —>  V3
```

```
GRAM#4
[1]          V3          —>  dhagena
[2]   LEFT   #dha V1     —>  #dha dha
[3]   LEFT   #- V1       —>  #- -
[4]   LEFT   dha V1 #tr  —>  dha ti #tr
```

```
[5]    LEFT  na V1 #tr   —>  na ti #tr
[6]    LEFT  - V1 #tr    —>  - ti #tr
[7]          (= V1 #tr   —>  (= ti #tr
```

Membership test (1): RND grammars

The new grammar worked in synthesis, but for it to be considered complete and consistent the parsing of new sentences had to be taken into account. In order to achieve rapid membership tests, the parsing had to be deterministic and bottom-up (data-driven) because a deterministic procedure cannot backtrack should an unacceptable state be reached. The main advantage of such a description was that it could easily be performed by hand. Bottom-up parsing implied that subgrammars were to be considered in reverse order (4, 3, 2, 1). The premises and conclusions of the rules were then swapped. This produced a *dual grammar* where instead of a single arrow (—>), a double arrow (<—>) was used to notate the possibility of a derivation in both directions. The single arrow was retained for rules relevant only in synthesis or analysis (e.g. in the 'absorption' subgrammar below, under *Constructing stress-sensitive grammars*).

A string was accepted if its derivation by the dual grammar yielded a starting symbol. Evidently, many derivations were possible, and so when several rules were candidates (i.e. their conclusions were substrings of the work string) there had to be a decision procedure to determine which was to be prioritised. Therefore in subgrammars where the order in which symbols were to be rewritten in the work string was not specified, the rule selected was the one appearing nearest the bottom of the list of rules. Such grammars were known as RND, or 'random', subgrammars. The following example shows the successful parsing of *-ti-trktdhatrkt* in which rules used in the derivation have been specified in the left margin:

```
                    (= -ti-trktdhatrkt)
Step  1   G#4 [6]   (= - V1 -trktdhatrkt)
…     2   G#4 [3]   (= - V1 V1 trktdhatrkt)
…     3   G#4 [3]   (= V1 V1 V1 trktdhatrkt)
…     4   G#4 [2]   (= V1 V1 V1 trkt V1 trkt)
…     5   G#3 [2]   (= V1 V1 V1 trkt V1 V V)
…     6   G#3 [2]   (= V1 V1 V1 V V V1 V V)
…     7   G#3 [1]   (= V1 V1 V1 V V V V V)
…     8   G#3 [1]   (= V1 V1 V V V V V V)
…     9   G#3 [1]   (= V1 V V V V V V V)
…     10  G#3 [1]   (= V V V V V V V V)
…     11  G#2 [1]   (= V8)                    ... successful.
```

Given the candidate rule, there were several occurrences (steps 2, 5, 7, 8, and 9) of the conclusion in the premise. Each time, the rightmost occurrence was selected.

The parsing algorithm was dependent upon the order in which rules appeared in each subgrammar. For instance, if the work string had contained *trktdhagena*, then rule 2 of G#4 would have yielded *trkt V1 gena* and consequently *gena* would never have been transformed. Since *dha* was a substring of *dhagena*, the rule producing *dhagena* had to be considered first. This reflected a simple pattern recognition strategy that 'large chunks should be recognised first', or precisely:

'Chunk' rule

> For any pair of rules p_i —> q_i and p_j —> q_j in the same subgrammar, if q_i is a substring of q_j then $i < j$.

Consequently the positions of rules 1 and 2 in G#4 needed to be swapped. (It should be noted that the grammar given in Kippen (1987:185,191) was written before the introduction of the chunk rule.)

Although V1s were derived from left to right in synthesis, they were not necessarily recognised from right to left in analysis. This was due to the fact that in the RND parsing algorithm the order of rules had priority over the position of the derivation. The shortcoming of this algorithm was evident in the parsing of (= --*dhatrktdhatrkt*), which, according to our initial hypothesis at least, was an incorrect sentence:

```
          (= --dhatrktdhatrkt)
G#4 [3]   (= V1 -dhatrktdhatrkt)
G#4 [3]   (= V1 V1 dhatrktdhatrkt)
G#4 [2]   (= V1 V1 dhatrkt V1 trkt)
G#4 [2]   (= V1 V1 V1 trkt V1 trkt)
G#3 [2]   (= V1 V1 V1 trkt V1 V V)
G#3 [2]   (= V1 V1 V1 V V V1 V V)
G#3 [1]   (= V1 V1 V1 V V V V V)
G#3 [1]   (= V1 V1 V V V V V V)
G#3 [1]   (= V1 V V V V V V V)
G#3 [1]   (= V V V V V V V V)
G#2 [1]   (= V8)                    ... successful.
```

In fact, the RND parsing algorithm yielded incorrect membership tests in most context-sensitive grammars. Yet a simple technique was soon developed to determine the construction of sentences from left to right with constraints on left and right contexts. This we termed the *sliding marker* method. A sliding marker grammar equivalent to the previous one was written:

```
GRAM#2

RND
[1]     V8    <—> V V V V V V V V
[2]     V16   <—> V V V V V V V V V V V V V V V V
[3]     V24   <—> V V V V V V V V V V V V V V V V V V V V V V V V V V V V

GRAM#3

RND
[1]     LEFT          V       <—> V1
[2]     LEFT          V V     <—> trkt
[3]     LEFT    V V V         <—> V3
[4]             (= #M         <—> (= M #M

GRAM#4

RND
[1]     M V3            <—> dhagena M
[2]     M trkt          <—> trkt M
[3]     #dha M V1       <—> #dha dha M
[4]     #- M V1         <—> #- - M
[5]     dha M V1 #tr    <—> dha ti M #tr
[6]     na M V1 #tr     <—> na ti M #tr
[7]     - M V1 #tr      <—> - ti M #tr
[8]     (= M V1 #tr     <—> (= ti M #tr
[9]     #M M )          <—> #M )
```

In synthesis, a sliding marker M was created by rule G#3 [4]. Note that the negative context #M prevented the rule from being self-embedding, i.e. from generating an infinite number of Ms. This marker was erased (in synthesis) or created (in analysis) by rule G#4 [9]. So, using the sliding marker method, a derivation sequence yielding *dhatrktdhatidhatrkt* was the following:

```
          (= V8)
G#2 [1]   (= V V V V V V V V)
G#3 [1]   (= V1 V V V V V V V)
G#3 [2]   (= V1 tr kt V V V V V)
G#3 [1]   (= V1 tr kt V1 V V V V)
G#3 [1]   (= V1 tr kt V1 V1 V V V)
G#3 [1]   (= V1 tr kt V1 V1 V1 V V)
G#3 [2]   (= V1 tr kt V1 V1 V1 tr kt)
G#3 [4]   (= M V1 tr kt V1 V1 V1 tr kt)
G#4 [3]   (= dha M tr kt V1 V1 V1 tr kt)
G#4 [2]   (= dha tr kt M V1 V1 V1 tr kt)
G#4 [3]   (= dha tr kt dha M V1 V1 tr kt)
G#4 [5]   (= dha tr kt dha ti M V1 tr kt)
G#4 [3]   (= dha tr kt dha ti dha M tr kt)
G#4 [2]   (= dha tr kt dha ti dha tr kt M)
G#4 [8]   (= dha tr kt dha ti dha tr kt)
```

In analysis, the same sequence was followed in reverse order.

Membership test (2): LIN grammars

The derivation just shown in synthesis mode is known as a *leftmost* derivation: a leftmost derivation forces the rewriting of the symbol appearing at the leftmost position of the work string. Similarly, a *rightmost* derivation was used to parse sequences. The definition of a leftmost/rightmost derivation applies well to context-free grammars, i.e. grammars in which the premise of each rule contains only one variable. Yet in context-sensitive grammars the problem consists in deciding whether or not the string of symbols to be rewritten includes the context. A general format for context-sensitive rules is

L C R —> L C' R

where L and R are left and right contexts respectively, and C is a string of variables rewritten as C'. Hart (1980:82) defined a canonic context-sensitive leftmost derivation in *strictly length-increasing* grammars, i.e. where C contains only one variable. Bel (1987b:7ff) has adapted Hart's definition to rightmost derivations in length-decreasing grammars. Appendix 1 gives the formal definition of this derivation along with a list of tests that a bottom-up parser must perform in order to select the next rule to be applied. In the BP fast parsing was achieved as a result of a simplification of this algorithm, but at the cost of some restrictions on grammars. Two have already been encountered: the partial ordering of rules (the 'chunk' rule), and the restriction on transforming any symbol already functioning as a context within the same subgrammar. A third restriction on overlapping patterns has been demonstrated in Bel (1987a:358-59) and need not concern us here.

A subgrammar in which a context-sensitive rightmost derivation was used in membership tests was called a LIN grammar. This term was adopted in view of the fact that the LIN membership algorithm was needed whenever right-linear subgrammars were used. It was generally unnecessary to impose leftmost derivations in synthesis: when needed, these were implemented with sliding markers or *right-linear-left-context* rules (see below). Effectively, a LIN grammar in synthesis was identical to a RND grammar in which all production rules were prefixed with LEFT. Consequently, the LIN version of the grammar became:

```
GRAM#2

RND
[1]   V8    <—> V V V V V V V V
[2]   V16   <—> V V V V V V V V V V V V V V V V
[3]   V24   <—> V V V V V V V V V V V V V V V V V V V V V V V V
```

```
GRAM#3

LIN
[1]   V       <—> V1
[2]   V V     <—> trkt
[3]   V V V   <—> V3
```

```
GRAM#4

LIN
[1]   V3          <—> dhagena
[3]   #dha V1     <—> #dha dha
[4]   #- V1       <—> #- -
[5]   dha V1 #tr  <—> dha ti #tr
[6]   na V1 #tr   <—> na ti #tr
[7]   - V1 #tr    <—> - ti #tr
[8]   (= V1 #tr   <—> (= ti #tr
```

In this grammar, the sliding marker M is no longer necessary because the analysis is forced to rightmost derivations by the instruction LIN. It may also be noticed that rule 2 in GRAM#4 has become unnecessary.

Fixed patterns

By linking the original pattern subgrammar (GRAM#1) to GRAM#2, 3, and 4, a grammar was obtained from which a large set of variations could be generated. Unfortunately, however, the chunk rule was violated during parsing because some sequences of bols occurring in fixed patterns A16, A8, etc. were recognised and transformed by GRAM#4, 3, and 2. For this reason the four rules defining fixed patterns had to be re-ordered and placed in a fifth subgrammar:

```
GRAM#5

ORD
[1] A8    <—> dhatidhagedheenagena
[2] A'8   <—> dhatidhageteenakena
[3] A16   <—> dhatidhagenadhatrktdhatidhagedheenagena
[4] A'16  <—> dhatidhagenadhatrktdhatidhageteenakena
```

Here, ORD indicated that in synthesis the rules could be applied in order (top to bottom) as no alternative options were possible. During analysis ORD subgrammars were treated identically to RND subgrammars.

Bol density

The vast majority of variations maintain the same bol density established in the theme of the qa‘ida. However, during performance some musicians vary the bol density from variation to variation, or even within one variation. To indicate bol density in the BP’s editor, a slash followed by an integer was typed to specify the number of bols between two tabulations (beat markers). When the tempo changed within the variation itself, the new (integer) value appeared as a terminal symbol. For instance, in the following improvisation (a maverick variation in view of the grammars listed above describing this qa‘ida)

dhatidhage	nadhatrkt	dhatidhage	dheenatrkt
dhadhatrkt	dhatidha-	dhatidhage	teena-ta
teena-ta	titakena	tatitake	teenakena
/8 dhatidhagenadhatrkt	dhatidhagedheenagena	gena-dhatidhagena	dhatidhagedheenagena

the fourth line was executed at double speed. Like terminals, density markers were manipulated by production rules. Consider, for instance, GRAM#2 [26] in appendix 2:

GRAM#2 [26] S16 <—> /8+ A16 O16 ; /4

in which the fourth line of the example above has been defined. ‘+’ and ‘;’ are context markers; the sequence derived from A16 O16 (32 bols) is played at a density of eight bols per beat, following which the density marker is reset to four bols per beat.

Templates

It will be remembered that additional information, such as brackets and asterisks, was incorporated into grammars in order to define the structure of patterns as well as to provide contexts in generation. It followed, then, that sentences could only be parsed if they were entered into the editor complete with structural information. Yet in view of the fact that the BP had to perform membership tests on large amounts of data comprising examples to which it was not always easy to assign a structure, this limitation was held to be unacceptable. In response to this, the inference engine of the BP was modified to generate templates from a grammar, i.e. a list of possible structures in which each dot represented a terminal symbol of one unit. Templates were enumerated and stored in the grammar file. For instance

[1] (=.....)(=.....)(=.....)*(:.....)(:.....)(=.....)
[2] (=.....)(=.....)*(:.....)(:.....)
[3] (=.....)(=.....)*(:.....)(=.....)

were the three generated by the grammar so far constructed for the qa‘ida in question. Bol density markers also appeared in templates. (See appendix 2: templates 4, 5 etc. include sequences of six bols per beat, and 3, 6 etc. have sequences of eight bols per beat.)

This development allowed us to dispense with structural information when entering data. Consequently, in analysis the BP took a new sentence and superimposed it on each template in strict order. A membership test was performed each time the sentence matched a template, so

allowing for any structural ambiguity to be assessed. It was important (especially when inferring weights, see below under *The probabilistic model*) that the first template producing a successful parsing was the most *specific*, i.e. the one containing the largest number of brackets. This reflected the view that patterns fitting a production rule in the pattern subgrammar were not incidental. Thus, templates had to be ordered from the most specific to most general. This was achieved automatically once the rules that produced them were ordered accordingly. In appendix 3, it may be seen that rules GRAM#2 [4-5] are more specific than [7-8] although they may incidentally have produced the same strings. It may also be noticed that the partial ordering of rules from generic to specific is not identical to the ordering imposed by the 'chunk' rule, although it never contradicts with it.

Structural markers and wildcards

The templates shown in appendix 2 contain symbols ('+' and ';') that played a specific structural role as contexts that indicated precise metric positions in the sentence. For instance, '+' was used to mark the beginning of a line of four beats (see templates 1 to 10). However, if it could be established that structures existed where two lines merged and indivisible chunks of bols could span the end of one line and the beginning of the next, then the marker was suppressed (see, for example, templates 11 to 16 which were generated as derivations of L24 M8 in GRAM#2 [5] and [17]). Furthermore, '+' was used as a context in all production rules defining cadential patterns (see GRAM#6). The offset from a position marked '+' was determined by the number of wildcards '?' (metavariables), each of which could be instantiated as a single variable or terminal (but *not* as a structural marker). For instance, in the following rule

GRAM#6 [5] LEFT + ? ? ? ? ? ? ? ? A8-2 <—> + ? ? ? ? ? ? ? ? dhatidhagedheena

the variable A8-2 could only be rewritten as *dhatidhagedheena* if it was located eight units to the right of a '+' marker.

In the grammar shown in appendix 2, ';' has been used to mark the final closing bracket of the work string: in effect, the end of a variation. A combination of metavariables and negative context is notated '#?'. Used as a left context, '#?' would mark the very beginning of a work string. This symbol denotes the absence of any context, be it a terminal, variable, or structural marker (bracket, equals, colon, semi-colon, asterisk, integer, plus, etc.). As structural markers were automatically inserted into input sentences during template matching in membership tests, there were few restrictions on their use in production rules. However, they had to be employed with great precision because their inconsistent use resulted in too many distinct templates (including, presumably, some that were unnecessary) and a much slower membership test.

A 'complete' grammar for this qa'ida

Until now we have discussed the construction of a grammar based on three quite regular variations of a qa'ida. In this way it has been possible to demonstrate the principal stages in the development of formal representations for tabla music. However, it should not be thought that all improvisations based on drum themes are equally regular. On the one hand, tabla teachers usually demonstrate simple and systematic variations to their students because their aim is to pass on the essence of a system. Students remember sets of 'fixed improvisations' and use them as models on which to base their own improvisatory efforts. On the other hand, performance situations 'offer musicians greater licence to explore new channels of creativity

and to stretch the limits of musical acceptability' (Kippen 1988c:30). The variation given above under *Bol density* was taken from just such a performance: clearly there is no duplication of the material in the two halves, there is a change in density within the variation itself, and the usual cadential ending to the first half is modified. Nevertheless, this piece was recognised by knowledgeable listeners to be a masterful improvisation, and so we must conclude that if our models are intended to be truly representative of tabla playing, then they too must be able to cope with the complexity and structural variety contained in performances from a range of different social contexts. Therefore, any grammar should be seen only as a hypothesis of musical structure that is 'complete' for the data entered up to that point but which makes no claims to 'completeness' with regard to as yet unexamined data. Experience has shown that a grammar rarely remains unchanged following the processing of a new set of examples.

The grammar shown in appendix 2 was written to account for a variety of very complex structures explored during one performance of this qa'ida by the expert musician Ustad Afaq Husain Khan of Lucknow. Brief comments restricted to its salient features are as follows: GRAM#2 is truly right-linear and sets all structural markers (line markers, brackets, asterisks, and tempo markers). These are used only as contexts in subsequent subgrammars, and so when the system is requested to generate templates it need only detail the language generated by GRAM#1 and 2, the alphabet of which is {L16, L14, L12, L24, M16, M14, M40, O8, M18, etc.}. Therefore, for each of these variables the inference engine need only derive the *first* string of terminals using subgrammars 3, 4, and 5 and write into the template a sequence of dots representing the lengths of the variables. GRAM#3 defines various subdivisions of L16, L14, etc. Its terminal alphabet comprises V30, V28, etc. (the starting symbols of GRAM#4) and all variables A16, A'16, A16-2, etc. that denote cadential patterns defined/recognised by GRAM#6. A16 represents a voiced pattern of sixteen units whereas A'16 is its partly unvoiced transformation (see rules 18 and 19 in GRAM#6). A16-2 is a truncated A16 where the last two bols have been omitted. The same principles apply to A8 and A6. GRAM#4 generates a string of Vs that are translated to bols in various contexts in GRAM#5. Both these subgrammars are LIN: the work string is transformed from left to right in synthesis, and from right to left in analysis. GRAM#5 [2] defines the leftmost gap that is always the second bol in any section of the sentence. This satisfies two hypothetical conditions: (1) any section of a sentence may not begin with a gap, and (2) a gap is structurally linked to the bol immediately preceding it. The ordering of rules in GRAM#5 has been determined by the chunk rule.

The parsing of the example given above under *Bol density* may be found in Bel (1987b:19-20). The test was positive when the sentence was matched against template 12. Templates 3 and 6 also matched the sentence but in both cases the parsing led to a dead end. The total matching of 21 templates took 1'43" on the Apple IIc. However, this is unrepresentative because membership tests were performed considerably more quickly with regard to the vast majority of grammars. For instance, a grammar for a different qa'ida that generated six templates may be found in Kippen and Bel (1989), in which the parsing of an input sentence was completed in eleven seconds. It should perhaps be mentioned here that the qa'ida we have dealt with in this paper, a composition often referred to as the 'King of qa'idas' by the traditional tabla players of Dehli where it originated (personal communication: Ustad Inam Ali Khan, New Delhi, April 1984), offers probably the widest scope of all for improvisation, and the modelling of the performance by Ustad Afaq Husain Khan has resulted in the most complex grammar encountered to date. The grammar would have been substantially more complex had each and every variation been included in the model (the improvisation extended to an exceptional total of twenty-nine variations for this qa'ida). As suggested in appendix 2, structures contained in the other variations would have been defined initially in GRAM#1 [3] and further additional rules.

Constructing context-sensitive grammars

In practice, GRAM#5 of the 'complete' grammar was easy to construct once the structural markers had been defined consistently in GRAM#2 and typical divisions identified in

GRAM#3. However, at one stage GRAM#5 [40] was not included in the grammar when a (correct) V10 sequence *dhagedheenatidhagedheenage* was being analysed. The derivation was:

(= dhagedheenatidhagedheenage)
G#5[21] (= dhagedheenatidhage V V V)
G#5[9] (= dhagedheenati V V V V V)
G#5[25] (= V V V V ti V V V V V) … *failed*

The shortest unrecognised suffix was *tidhagedheenage,* and therefore a rule

V V V V V V <—> tidhagedheenage

had to be added to the grammar. When the updated grammar was tried in synthesis mode, a derivation of V8 resulted in *dhatitidhagedheenage*, which was considered to be incorrect because of the two adjacent ***ti***. Therefore we used the negative context ***#ti*** to yield the final rule:

[40] #ti V V V V V V <—> #ti tidhagedheenage

If other incorrect left contexts had been found in productions using this rule, then the derivation would have been listed with all possible positive contexts:

dha V V V V V V <—> dha tidhagedheenage
na V V V V V V <—> na tidhagedheenage
etc...

(See, for instance, rules 12, 13, 14 in the same subgrammar; note also how rule 12 reflects a newly encountered derivation where ***kt*** and ***ti*** are juxtaposed so disproving a previous hypothesis.) The addition of contextual constraints when a negative example is found may be called a *specialisation* process, whereas adding a new rule or suppressing a context is a *generalisation* process. The method described here could be automated as it bears some resemblance to grammatical inference techniques that use the right formal derivatives of a sample sequence (Fu & Booth 1975).

Constructing stress-sensitive grammars

We now consider another qa‘ida, this time of the Ajrara tradition (see Kippen 1988a:xi).

Theme:

dhin--dhagena	dha--dhagena	dhatigegenaka	dheenedheenagena
tagetirakita	dhin--dhagena	dhatigegenaka	teeneteenakena
tin--takena	ta--takena	tatikekenaka	teeneteenakena
tagetirakita	dhin--dhagena	dhatigegenaka	dheenedheenagena

A few variations:

dhin--dhagena	dha--dhagena	dhatigegenaka	dheenedheenagena
tagetirakita	dhin--dhagena	dhatigegenaka	teeneteenakena
dheenedheenagena	*teeneteenakena*	*tirakitatira*	*kitatirakita*
tagetirakita	dhin--dhagena	dhatigegenaka	teeneteenakena
tin--takena	ta--takena	tatikekenaka	teeneteenakena
taketirakita	tin--takena	tatikekenaka	teeneteenakena
dheenedheenagena	*teeneteenakena*	*tirakitatira*	*kitatirakita*
tagetirakita	dhin--dhagena	dhatigegenaka	dheenedheenagena

dhin--dhagena	dha--dhagena	dhatigegenaka	dheenedheenagena
tagetirakita	dhin--dhagena	dhatigegenaka	teeneteenakena
dhin--dhagena	*dha-dha-dha-*	*dhagenadheen--*	*dhagenadha--*
tagetirakita	dhin--dhagena	dhatigegenaka	teeneteenakena
tin--takena	ta--takena	tatikekenaka	teeneteenakena
taketirakita	tin--takena	tatikekenaka	teeneteenakena
dhin--dhagena	*dha-dha-dha-*	*dhagenadheen--*	*dhagenadha--*
tagetirakita	dhin--dhagena	dhatigegenaka	dheenedheenagena

dhin--dhagena	dha--dhagena	dhatigegenaka	dheenedheenagena
tagetirakita	dhin--dhagena	dhatigegenaka	teeneteenakena
dheenedheenagena	*dheenedha-dheene*	*dhatigegenaka*	*teeneteenakena*
tagetirakita	dhin--dhagena	dhatigegenaka	teeneteenakena
tin--takena	ta--takena	tatikekenaka	teeneteenakena
taketirakita	tin--takena	tatikekenaka	teeneteenakena
dheenedheenagena	*dheenedha-dheene*	*dhatigegenaka*	*teeneteenakena*
tagetirakita	dhin--dhagena	dhatigegenaka	dheenedheenagena

Our observations based on several samples of variations (again from performances and demonstrations by Ustad Afaq Husain Khan of Lucknow) suggested that variable lines (shown in italics) were constructed with chunks of bols of lengths 3, 4, and 6 in permutations that we presumed to be context-free. This view was reinforced by the fact that no technical (i.e. fingering) difficulties were encoutered when chunks were arranged in any order. The significant units were listed in the following *lexical* rules:

```
RND
A3       <—>  dhin--
A3       <—>  dha--
A3       <—>  dhagena
A4       <—>  tirakita
A3 A3    <—>  dhagenadhin--
A3 A3    <—>  dhagenadha--
A6       <—>  dha-dha-dha-
A6       <—>  dha-ta-dha-
A6       <—>  dheenedheenedheene
A6       <—>  dheenedha-dheene
A6       <—>  tagetirakita
A6       <—>  dheenedheenagena
A6       <—>  teeneteenakena
A6       <—>  dhatigegenaka
```

In view of their frequent occurrence in examples, *dhagenadhin--* and *dhagenadha--* were identified specifically so as to increase the likelihood that they would be generated as blocks. A grammar for defining all possible sequences in variable lines of 6, 12 or 24 units was:

```
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
```

This grammar was correct in the sense that any sequence of A3, A4, and A6 was a derivation of B24 only if the sum of its metric values was 24. Yet incomplete derivations such as A3 A6 A3 A4 A3 A3 A A were also possible. Obviously not more than two isolated As were found in such sequences and so these were eliminated (or 'absorbed') in the following subgrammar where rules were taken in order (single arrows here indicate that these rules were ignored in analysis mode):

```
ORD
A A       —> A2
A3 A      —> A4
#A3 A     —> A #A3
A6 A2     —> A4 A4
#A6 A2    —> A2 #A6
```

Some of the pieces generated by this grammar displayed irregularities in the accentuation. For instance,

dhin--tiraki ta**dha**gena**dha**ti gegenaka**ti**ra kita**ti**rakita

imposed a rhythm counter to the natural stresses of the beat and half-beat, and was therefore virtually impossible to recite or perform at speeds normally employed by musicians (c. mm=108-120, i.e.up to twelve bols per second). In a four-beat string comprising 24 units, primary accents fall on beats and half-beats: 1, 4, 7, 10, 13, 16, 19, and 22. A cursory analysis of variations created by musicians showed that in addition to these divisions they employed hemiolic rhythmic patterns beginning on units 1, 7, and 13. This produces a series

of secondary stresses on units 5, 9, 11, 15, 17, 21. The following is a list of possible starting positions for the blocks defined above:

A3: 1, 4, 7, 10, 13, 16, 19, 22
A4: 1, 5, 7, 9, 11, 13, 15, 17, 21
A6: 1, 4, 7, 10, 13, 16, 19
tagetirakita: 1, 4, 5, 7, 9, 10, 11, 13, 15, 16, 17, 19

The exceptional status of ***ta****ge****ti****rakita* is due to the fact that it was accentuated in two different ways. Therefore it was labelled with a new variable: C6.

We developed a way to define derivations of B24, B12, and B6 in a systematic way that took into account acceptable starting positions. The grammar was right-linear:

```
LIN
B24  <—> A3 B21          ...(A3 in starting position: 24 - (3+21) +1 = 1)
B24  <—> A4 B20
B24  <—> C6 B18
B24  <—> A6 B18
B21  <—> A3 B18          ...(A3 in starting position:  24 - (3+18) +1 = 4)
B21  <—> A4 B17          ...(cancelled: A4 in starting position 4)
B21  <—> A6 B15
B21  <—> C6 B15
B20  <—> A3 B17          ...(cancelled: A3 in starting position 5)
B20  <—> A4 B16
B20  <—> A6 B14          ...(cancelled: A6 in starting position 5)
B20  <—> C6 B14
etc...
```

This grammar could have generated a string A4 A4 A4 A4 A4 A4 whose only derivation would have been an unbroken series of *tirakita*s that musicians would almost certainly have assessed as incorrect. The solution to this problem may be found in subgrammar 3 of the final grammar for this qa'ida, given in appendix 3, where it may be seen that rules 28 to 31 were attributed left contexts. We term this kind of subgrammar *right-linear-context-sensitive*. In the version given here, the grammar shows a number of improvements over that published in Bel (1987a).

The probabilistic model

It should be noted that the grammars discussed in this paper so far can only claim the status of analytical models in view of the fact that they were shown to have begun as initial hypotheses and to have developed as a result of our own intuitions, albeit based on considerable practical experience of tabla and extensive analyses of the music. It is beyond the scope of this paper to enter into a detailed discussion of theoretical and methodological aspects of this research: these have been covered extensively in Kippen (1985, 1987, 1988b, 1988c). However, it should be emphasized that a prominent aim of the research has been to create a human-computer interaction where musicians themselves respond to the output of BP grammars, and grammars are in turn modified to account for the input of musicians. Thus, in theory at least, analytical control over the models lies with the musicians, and it is they, not us, who are the sole arbiters of the correctness of computer-generated data.

A number of problems were encountered during actual interactions. One was that sometimes a grammar would reach a point of stagnation where computer-generated variations were judged to be neither very good nor incorrect. Consequently there was no simple way of refining or

improving the model. We felt that a solution lay in attributing to each production rule a coefficient of likelihood (or weight) where the probability that certain generative paths would be chosen in preference to others could be examined.

The probabilistic model that has been implemented on the BP is derived from probabilistic grammars/automata as defined by Booth & Thompson (1973), the difference being that a weight — within the range [0,255] — rather than a probability is attached to every rule. The rule probability is computed as follows: if the weight is zero then the probability is zero; if the weight is positive then the inference engine calculates the sum of weights of all *candidate* rules, and the rule probability is the ratio of its own weight to the sum. Candidate rules are those whose premise is a substring of the work string. Consider, for example, GRAM#2 in appendix 2

```
GRAM#2 [2]    <100>  S64   <—>  L16 + S48
...etc.
GRAM#2 [12]   <100>  S16   <—>  O16
GRAM#2 [13]   <100>  S50   <—>  V10  A'8-2  * (= N18 +) + O16
...etc.
GRAM#2 [16]   <100>  S52   <—>  M20 + S32
GRAM#2 [17]   <100>  S40   <—>  M8 + S32
GRAM#2 [18]   <100>  S32   <—>  * (=+ N16 +) + S16
...etc.
GRAM#2 [25]   <100>  S34   <—>  * (= N18 +) + S16
GRAM#2 [26]   <5>    S16   <—>  /8+ A16  O16 ; /4
```

and a work string containing S16. The sum of the weights of the two candidate rules is 100+5 = 105. The probability of rule 26 is therefore 5/105 = 0.048. The advantage of weights over probabilities is that they do not presuppose the sum of the coefficients of all candidate rules to be 1 — a condition that can only be satisfied in context-free grammars.

Weights (and their associated probabilities) have been used in modus ponens to direct the BP's production along paths more likely to be followed by musicians. In some context-free grammars — those that fulfilled the *consistency* condition expressed by Booth & Thompson (1973:442) — they were used to compute a *probabilistic sentence function*, i.e. a coefficient representing the likelihood of occurrence of each sentence in the language. Even in cases where the likelihood sentence function was not strictly probabilistic, it was used as an approximate assessment of the consistency of the sentence with the grammar: at least, any sentence that required a rule with weight <0> during its parsing was assigned likelihood zero. Comparing the likelihoods obtained from parsing a sentence on different templates helped to determine decisions regarding ambiguous structures.

Another remarkable feature of consistent grammars is that rule probabilities can be inferred from a set of sentences (Maryanski & Booth 1977:525). The algorithm implemented in the BP is more powerful than the one devised by Maryanski and Booth, since the latter required the choice of a sample set in which *all* rules had been used. Given a grammar and a subset of the language that this grammar generates (for instance a sample sequence taken from a performance of an expert musician), rule weights may be inferred as follows: let all weights be reset to zero; then analyse every sentence and increment by one unit the weights of all rules used in the derivation. The weights displayed in the grammar in appendix 3 were inferred from the analysis of thirteen examples. Consequently, it produced variations that bore some resemblance to those in the sample sequence. Rules that were not used in this process, for instance GRAM#3 [11] or GRAM#1 [2], were scrutinised to see whether they were incorrect or whether they pointed to fragments of the language that had not yet been explored. To test this, their weights were set to a high value so that the BP was forced to generate sentences that either had never been assessed or at least had not been considered by the informant at the time.

Using weighted rules resulted in a marked improvement in the quality of the generated music. This went a long way towards solving the problem of musical credibility encountered in earlier experiments, a problem that arose from the complete randomness of the generative process and the stagnation of grammars.

Further developments

A more general theoretical model of BP grammars, called BP2, is being developed and implemented on a Macintosh computer. In this new version, potentially any symbolic/numeric musical code (e.g. MIDI code) can be used as a terminal alphabet. The concept of voiced/unvoiced transformations has been generalised to any user-defined non-erasing homomorphism — for instance tonal transposition can be easily represented. Several homomorphisms can be defined on the same alphabet.

The main advantage of BP2 is its ability to represent the time structures of polyphonic music: simultaneous events are listed in sets, each event may in turn be a set of sub-events. A program generating MIDI code sequences from BP polyphonic representations is being studied.

A free copy of BP2 and the grammars described in this paper are available in Binhex format on the E-mail network. ***[Alan, plese set this bit as a footnote]***

Conclusion

In this paper, we have attempted to show that formal models can be manipulated to represent highly elaborated musical concepts that underlie the art of improvisation in tabla music. Grammars are evidently easier to work on if the music is regular: i.e. if it conforms to idealised models such as those transmitted to students in teaching situations. On the other hand, the grammar shown in appendix 2 shows the extraordinary rise in complexity when attempts are made to account for the irregularities encountered in actual performances. The unpredictability of improvisation in this context serves only to compound the problem, as there is no guarantee that even a complex and reasonably comprehensive grammar will be adequate to deal with further samples. Technically, however, the formal representations discussed here can be adapted to express any structure likely to be played by a tabla musician, whatever its degree of complexity.

The major limitation of the models, then, lies not in the formal representations themselves but in the transfer of knowledge from informants to the computer. Knowledge acquisition has proved to be an important consideration in artificial intelligence, and one that has produced no satisfactory answers so far. The problem consists in the fact that expert systems like the BP represent knowledge at a low (non-hierarchical) theoretical level, and so it is impossible to separate general analytical statements from specific instances of facts. A discussion of the shortcomings of the knowledge-acquisition strategy in the BP may be found in Kippen & Bel (1989).

References cited

Bel, Bernard

'Grammaires de génération et de reconnaissance de phrases rythmiques', *6ème congrès AFCET / INRIA*, Antibes, 1987a:353-66

'Les grammaires et le moteur d'inférences du Bol Processor', note 237, Groupe Représentation et Traitement des Connaissances, CNRS, Marseille, 1987b

'Designing tools for knowledge representation in the anthropological study of a musical system', unpublished, 1988

Booth, T.L. and R.A. Thompson

'Applying Probability Measures to Abstract Languages', *IEEE Transactions on Computers*, Vol.C-22, n°5, 1973:442-50

Fu, K.S. and T.L. Booth

'Grammatical Inference: Introduction and Survey - Part I', *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-5, n°1, 1975:95-111

Hart, Johnson M.

'Derivation Structures for Strictly Context-Sensitive Grammars', *Information and Control* 45, 1980:68-89

Kippen, Jim

'The dialectical approach: a methodology for the analysis of tabla music', *International Council for Traditional Music (UK Chapter) Bulletin* n°12, 1985:4-12

'An ethnomusicological approach to the analysis of musical cognition', *Music Perception* Vol.5 Part 2, 1987:173-95

The Tabla of Lucknow: a Cultural Analysis of a Musical Tradition, Cambridge University Press, 1988a

'On the uses of computers in anthropological research', *Current Anthropology* Vol.29 Part 2, 1988b:317-20

'Computers, fieldwork, and the problem of ethnomusicological analysis', *International Council for Traditional Music (UK Chapter)* N°20, 1988c:20-35

Kippen, Jim and Bernard Bel

'Can a computer help resolve the problem of ethnographic description?', *Anthropological Quarterly*, forthcoming, 1989

Maryanski, F.J., and T.L. Booth

'Inference of Finite-State Probabilistic Grammars', *IEEE Transactions on Computers*, Vol.C-26, n°6, 1977:521-36
[Some anomalies of this paper are corrected in B.R. Gaine's paper 'Maryanski's Grammatical Inferencer', *IEEE Transactions on Computers*, Vol.C-27, n°1, 1979:62-64]

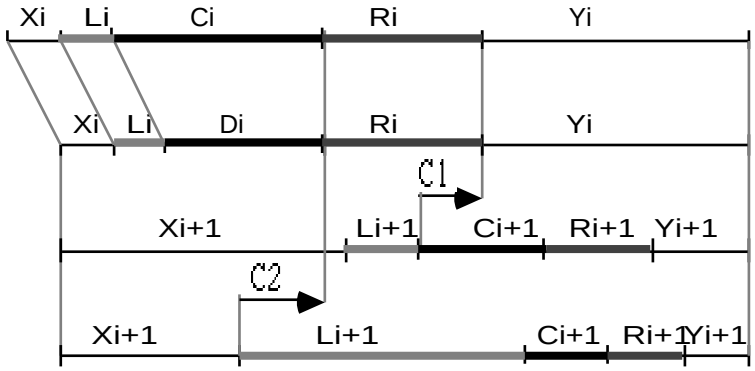
Appendix 1: context-sensitive canonic rightmost derivation

The formal definition of the context-sensitive rightmost definition adapted from Hart (1980) and used in the membership test of LIN grammars is given below. Vertical bars denote the lengths of strings:

Let G be a length-increasing grammar. The derivation in G:
$W_0 => W_1 => ... => W_n$
is *context-sensitive rightmost* iff:
$\forall i \in [0, n-1]$, $W_i = X_iL_iC_iR_iY_i$
$W_{i+1} = X_iL_iD_iR_iY_i$ when applying rule f_i: $L_iC_iR_i \rightarrow L_iD_iR_i$,
and at least one of the two following conditions is satisfied:
(C1) $|C_{i+1}R_{i+1}Y_{i+1}| > |Y_i|$
(C2) $|L_{i+1}C_{i+1}R_{i+1}Y_{i+1}| > |R_iY_i|$

In Hart's definition (1980:82), $|C_i| = |C_{i+1}| = 1$, so that C1 may be written $|R_{i+1}\ Y_{i+1}| \geq |Y_i|$.

The diagram and commentary below illustrate conditions **C1** and **C2**:



Suppose conditions **C1** and **C2** are not satisfied. Since **C2** is not true, rule f_{i+1} could have been applied before f_i as $L_{i+1}C_{i+1}R_{i+1}$ would be a substring of R_iY_i. Besides, since **C1** is not true, applying rule f_{i+1} would only modify Y_i without changing the context R_i. In such a case the order in which f_i and f_{i+1} are applied might have been inverted: a change that would have been justified since all symbols rewritten by f_{i+1} are to the right of those rewritten by f_i.

We now explain how the inference engine of the BP handles ambiguity in bottom-up parsing. Suppose that for a working string W_i there are two candidate rules:

f_i $L_iC_iR_1 \rightarrow L_iD_iR_i$
f'_i $L'_iC'_iR'_i \rightarrow L'_iD'_iR'_i$ where $X_iL_iC_iR_iY_i = X'_iL'_iC'_iR'_iY'_i = W_i$

The selection criterion is the following: f_i will have priority over f'_i if one of the following conditions (in this order) is satisfied:

(D1) $|X_i L_i C_i| > |X'_i L'_i C'_i|$
(D2) $|X_i L_i C_i R_i| > |X'_i L'_i C'_i R'_i|$
(D3) $|L_i C_i R_i| > |L'_i C'_i R'_i|$
(D4) $i > i'$

It can be proved (Bel 1987b:8) that once D1 has been considered, D2 is no longer relevant and ambiguity may therefore be handled by D3 and D4. D3 makes a decision on the basis of the length of the conclusions of the two rules, and D4 is a final arbitrary decision that takes into account the order in which the rules appear in the grammar. To augment efficiency, D3 is not considered by the inference engine of the BP, and therefore the test relies on the partial ordering of rules in the grammar (see 'chunk' rule).

Appendix 2: Delhi qa'ida (as performed by Ustad Afaq Husain Khan of Lucknow

```
GRAM#1 [1]  RND
GRAM#1 [2] <100>  S  <—>  (=+ S64 ;)
GRAM#1 [3] <100>  S  <—>  ...  Other patterns
-----------------------------------------------------------------------------------------------------------------------
GRAM#2 [1]  LIN
GRAM#2 [2] <100>  S64  <—>  L16 + S48
GRAM#2 [3] <100>  S64  <—>  L14  S50
GRAM#2 [4] <100>  S64  <—>  L12  S52
GRAM#2 [5] <100>  S64  <—>  L24  S40
GRAM#2 [6] <100>  S48  <—>  M16 + S32
GRAM#2 [7] <100>  S48  <—>  M14  S34
GRAM#2 [8] <100>  S48  <—>  M40  O8
GRAM#2 [9] <100>  S50  <—>  M18 + S32
GRAM#2 [10] <100>  S50  <—>  M34 + S16
GRAM#2 [11] <100>  S50  <—>  M18 + * (=+ N14 )  O18
GRAM#2 [12] <100>  S16  <—>  O16
GRAM#2 [13] <100>  S50  <—>  V10  A'8-2  * (= N18 +) + O16
GRAM#2 [14] <100>  S50  <—>  M20  * (= N14 +) + O16
GRAM#2 [15] <100>  S52  <—>  V28  A8-2  O18
GRAM#2 [16] <100>  S52  <—>  M20 + S32
GRAM#2 [17] <100>  S40  <—>  M8 + S32
GRAM#2 [18] <100>  S32  <—>  * (=+ N16 +) + S16
GRAM#2 [19] <100>  S32  <—>  * (= /6+ V12 /4 A8 +) + O16
GRAM#2 [20] <100>  S32  <—>  * (= /6+ A16  V8 + /4) + O16
GRAM#2 [21] <100>  S32  <—>  * (= /6+ V24 + /4) + O16
GRAM#2 [22] <100>  S32  <—>  * (= /8+ N'16 + A16 + /4) + O16
GRAM#2 [23] <100>  S32  <—>  * (=+ N14 )  O18
GRAM#2 [24] <100>  S34  <—>  O34
GRAM#2 [25] <100>  S34  <—>  * (= N18 +) + S16
GRAM#2 [26] <5>  S16  <—> /8+ A16  O16 ; /4
-----------------------------------------------------------------------------------------------------------------------
GRAM#3 [1]  RND
GRAM#3 [2] <100>  LEFT  M16  <—>  V16
GRAM#3 [3] <100>  LEFT  N18  <—>  V18
GRAM#3 [4] <100>  LEFT  (=+ L16  <—>  (=+ A16
GRAM#3 [5] <100>  LEFT  (=+ L16  <—>  (=+ V16
GRAM#3 [6] <100>  LEFT  (=+ L14  <—>  (=+ V10  A'6-2
GRAM#3 [7] <100>  LEFT  M14  <—>  A'16-2
GRAM#3 [8] <100>  LEFT  (=+ L14  <—>  (=+ A16-2
GRAM#3 [9] <100>  LEFT  (=+ L14  <—>  (=+ A'16-2
GRAM#3 [10] <100>  LEFT  (=+ L12  <—>  (=+ A16-4
GRAM#3 [11] <100>  LEFT  (=+ L24  <—>  (=+ V24
GRAM#3 [12] <100>  LEFT + M16 + * <—> + V10  A'6 + *
GRAM#3 [13] <100>  LEFT + M16  <—> + A'16
GRAM#3 [14] <100>  LEFT  M8 + * <—>  A'8 + *
GRAM#3 [15] <100>  LEFT  M16 + * <—>  V8  A'8 + *
GRAM#3 [16] <100>  LEFT  M14  <—>  V8  A'8-2
GRAM#3 [17] <100>  LEFT  M18 + * <—>  V10  A'8 + *
GRAM#3 [18] <100>  LEFT  M18 + * <—>  V18 + *
GRAM#3 [19] <100>  LEFT  M34 +  <—>  V28  A'6 +
GRAM#3 [20] <100>  LEFT  M34 +  <—>  V26  A'8 +
GRAM#3 [21] <100>  LEFT  M20 + * <—>  V12  A'8 + *
GRAM#3 [22] <100>  LEFT  M20  <—>  V20
GRAM#3 [23] <100>  LEFT  M40  <—>  V8  A'8-2  V26
GRAM#3 [24] <100>  LEFT  * (= /8+ N'16  <—>  * (= /8+ V16
GRAM#3 [25] <100>  LEFT  * (= /8+ N'16  <—>  * (= /8+ N16
GRAM#3 [26] <100>  LEFT  N16 + <—>  V12  A4 +
GRAM#3 [27] <100>  LEFT  N16 + <—>  V8  A8 +
GRAM#3 [28] <100>  LEFT  N16 + <—>  V10  C6 +
GRAM#3 [29] <100>  LEFT  * (=+ N16  <—>  * (=+ A16
```

GRAM#3 [30] <100> LEFT M14 <—> V8 A8-2
GRAM#3 [31] <100> LEFT * (=+ N14 <—> * (=+ V8 A8-2
GRAM#3 [32] <100> LEFT * (=+ N14 <—> * (=+ A16-2
GRAM#3 [33] <100> LEFT N14 + <—> V6 A8 +
GRAM#3 [34] <100> LEFT O8 ; <—> A8 ;
GRAM#3 [35] <100> LEFT O34 ; <—> V26 A8 ;
GRAM#3 [36] <100> LEFT O18 ; <—> V10 A8 ;
GRAM#3 [37] <100> LEFT O16 ; <—> V8 A8 ;
GRAM#3 [38] <100> LEFT + O16 ; <—> + A16 ;

GRAM#4 [1] LIN
GRAM#4 [2] <100> V30 <—> V V V28
GRAM#4 [3] <100> V28 <—> V V V26
GRAM#4 [4] <100> V26 <—> V V V24
GRAM#4 [5] <100> V24 <—> V V V V V20
GRAM#4 [6] <100> V20 <—> V V V18
GRAM#4 [7] <100> V18 <—> V V V16
GRAM#4 [8] <100> V16 <—> V V V V V12
GRAM#4 [9] <100> V12 <—> V V V10
GRAM#4 [10] <100> V10 <—> V V V8
GRAM#4 [11] <100> V8 <—> V V V6
GRAM#4 [12] <100> V6 <—> V V V V V V

GRAM#5 [1] LIN
GRAM#5 [2] <100> ? V <—> ? -
GRAM#5 [3] <100> V <—> dha
GRAM#5 [4] <100> V V <—> trkt
GRAM#5 [5] <100> V V <—> dheena
GRAM#5 [6] <100> V V <—> teena
GRAM#5 [7] <100> V V <—> dhati
GRAM#5 [8] <100> V V <—> gena
GRAM#5 [9] <100> V V <—> dhage
GRAM#5 [10] <100> + V V <—> +tidha
GRAM#5 [11] <100> - V V <—> -tidha
GRAM#5 [12] <100> kt V V <—> kttidha
GRAM#5 [13] <100> na V V <—> natidha
GRAM#5 [14] <100> ge V V <—> getidha
GRAM#5 [15] <100> - V V <—> -ti-
GRAM#5 [16] <100> kt V V <—> ktti-
GRAM#5 [17] <100> ge V V <—> geti-
GRAM#5 [18] <100> na V V <—> nati-
GRAM#5 [19] <100> V V V <—> dhagena
GRAM#5 [20] <100> V V V <—> teenake
GRAM#5 [21] <100> V V V <—> dheenage
GRAM#5 [22] <100> V V V <—> dhatrkt
GRAM#5 [23] <100> V V V <—> trktdha
GRAM#5 [24] <100> V V V V <—> tidhagena
GRAM#5 [25] <100> V V V V <—> dhagedheena
GRAM#5 [26] <100> V V V V <—> teena-ta
GRAM#5 [27] <100> #ti V V V V <—> #ti tidhatrkt
GRAM#5 [28] <100> V V V V <—> dheenagena
GRAM#5 [29] <100> V V V V <—> teenakena
GRAM#5 [30] <100> V V V V V <—> dhagenadheena
GRAM#5 [31] <100> V V V V V <—> dhagenateena
GRAM#5 [32] <100> V V V V V <—> dhagenadhati
GRAM#5 [33] <100> V V V V V <—> dhatrktdhati
GRAM#5 [34] <100> V V V V V <—> dhatidhatrkt
GRAM#5 [35] <100> V V V V V <—> dhatidhagena
GRAM#5 [36] <100> V V V V V V <—> dheenagedhatrkt
GRAM#5 [37] <100> V V V V V V <—> genagedhatrkt
GRAM#5 [38] <100> V V V V V V <—> dhagedheenagena
GRAM#5 [39] <100> V V V V V V <—> dhageteenakena
GRAM#5 [40] <100> #ti V V V V V V <—> #ti tidhagedheenage
GRAM#5 [41] <100> V V V V V V <—> teenakegenage
GRAM#5 [42] <100> V V V V V V V V <—> dhagenagenanagena

GRAM#5 [43] <100> V V V V V V V V <—> dhatidhagedheenagena
GRAM#5 [44] <100> V V V V V V V V <—> dhatidhageteenakena
GRAM#5 [45] <100> V V V V V V V V A8 <—> dhatrktdhatidhatrkt A8
GRAM#5 [46] <100> V V V V V V V V A'8 <—> dhatrktdhatidhatrkt A'8

GRAM#6 [1] ORD
GRAM#6 [2] LEFT + ? ? ? ? ? ? ? ? ? ? ? A'6-2 <—> + ? ? ? ? ? ? ? ? ? ? ? dhageteena
GRAM#6 [3] LEFT + ? ? ? ? ? ? ? ? ? ? ? C6-2 <—> + ? ? ? ? ? ? ? ? ? ? ? dhagedheena
GRAM#6 [4] LEFT + ? ? ? ? ? ? ? ? ? ? ? A'6-2 <—> + ? ? ? ? ? ? ? ? ? ? ? dhageteena
GRAM#6 [5] LEFT + ? ? ? ? ? ? ? ? ? A8-2 <—> + ? ? ? ? ? ? ? ? ? dhatidhagedheena
GRAM#6 [6] LEFT A8-2 ? ; <—> dhatidhagedheena ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ;
GRAM#6 [7] LEFT + ? ? ? ? ? ? ? ? ? A'8-2 <—> + ? ? ? ? ? ? ? ? ? dhatidhageteena
GRAM#6 [8] LEFT + ? A'8-2 <—> + ? dhatidhageteena
GRAM#6 [9] LEFT A'6 + <—> dhageteenakena+
GRAM#6 [10] LEFT A'8 + <—> dhatidhageteenakena+
GRAM#6 [11] LEFT A4 + <—> dheenagena+
GRAM#6 [12] LEFT C6 + <—> dhagedheenagena+
GRAM#6 [13] LEFT A8 + <—> dhatidhagedheenagena+
GRAM#6 [14] LEFT A8 ; <—> dhatidhagedheenagena;
GRAM#6 [15] LEFT + A16-4 <—> +dhatidhagenadhatrktdhatidhage
GRAM#6 [16] LEFT + A16-2 #ge <—>+dhatidhagenadhatrktdhatidhagedheena #ge
GRAM#6 [17] LEFT + A'16-2 #ke <—>+dhatidhagenadhatrktdhatidhageteena #ke
GRAM#6 [18] LEFT + A16 <—> +dhatidhagenadhatrktdhatidhagedheenagena
GRAM#6 [19] LEFT + A'16 <—> +dhatidhagenadhatrktdhatidhageteenakena
----

Templates:
[1]   TEM
[2]   (=+...............+...............+ * (=+...............+) +...............;)
[3]   (=+...............+...............+ * (=+...............+) +/8+...............................;/4;)
[4]   (=+...............+...............+ * (=/6+.........../4.......+) +...............;)
[5]   (=+...............+...............+ * (=/6+......................+/4) +...............;)
[6]   (=+...............+...............+ * (=/8+...............+...............+/4) +...............;)
[7]   (=+...............+...............+ * (=+.............) ....;)
[8]   (=+...............+...;)
[9]   (=+...............+............. * (=.................+) +...............;)
[10]   (=+...............+............. * (=.................+) +/8+...............................;/4;)
[11]   (=+...............................+ * (=+...............+) +...............;)
[12]   (=+...............................+ * (=+...............+) +/8+...............................;/4;)
[13]   (=+...............................+ * (=/6+.........../4.......+) +...............;)
[14]   (=+...............................+ * (=/6+......................+/4) +...............;)
[15]   (=+...............................+ * (=/8+...............+...............+/4) +...............;)
[16]   (=+...............................+ * (=+.............) ....;)
[17]   (=+...+...............;)
[18]   (=+...+/8+...............................;/4;)
[19]   (=+............................. * (=.................+) +...............;)
[20]   (=+................................ * (=.............+) +...............;)
[21]   (=+...;)

Appendix 3: Ajrara qa'ida (as performed by Ustad Afaq Husain Khan of Lucknow

```
GRAM#1 [1]          RND
GRAM#1 [2] <0>      S <—> S48
GRAM#1 [3] <13>     S <—> S96
-------------------------------------------------------------------------------------------------------------
GRAM#2 [1]    RND
GRAM#2 [2] <13>     S96 <—> (= F48 ) (= V24 ) F'24 *(: F48 ) (: V24 ) F24
GRAM#2 [3] <0>      S48 <—> (= V24 ) F'24 *(: V24 ) F24
GRAM#2 [4] <1>      V24 <—> (= V12 ) (: V12 )
GRAM#2 [5] <3>      V24 <—> (= V12 ) *(: V12 )
GRAM#2 [6] <0>      V24 <—> Q24
GRAM#2 [7] <2>      V24 <—> V12 V12
GRAM#2 [8] <7>      V24 <—> B24
GRAM#2 [9] <3>      V12 <—> (= B6 ) *(: B6 )
GRAM#2 [10] <5>     V12 <—> B12
-------------------------------------------------------------------------------------------------------------
GRAM#3 [1]          LIN
GRAM#3 [2] <3>      B3 <—> A3
GRAM#3 [3] <1>      B4 <—> A4
GRAM#3 [4] <6>      B6 <—> A6
GRAM#3 [5] <1>      B6 <—> C6
GRAM#3 [6] <3>      B24 <—> A3 B21
GRAM#3 [7] <0>      B24 <—> A4 B20
GRAM#3 [8] <4>      B24 <—> A6 B18
GRAM#3 [9] <0>      B24 <—> C6 B18
GRAM#3 [10] <3>     B21 <—> A3 B18
GRAM#3 [11] <0>     B21 <—> A6 B15
GRAM#3 [12] <0>     B21 <—> C6 B15
GRAM#3 [13] <0>     B20 <—> A4 B16
GRAM#3 [14] <0>     B20 <—> C6 B14
GRAM#3 [15] <2>     B18 <—> A3 B15
GRAM#3 [16] <0>     B18 <—> A4 B14
GRAM#3 [17] <4>     B18 <—> A6 B12
GRAM#3 [18] <1>     B18 <—> C6 B12
GRAM#3 [19] <0>     B16 <—> A4 B12
GRAM#3 [20] <0>     B16 <—> C6 B10
GRAM#3 [21] <2>     B15 <—> A3 B12
GRAM#3 [22] <0>     B15 <—> A6 B9
GRAM#3 [23] <0>     B15 <—> C6 B9
GRAM#3 [24] <0>     B14 <—> A4 B10
GRAM#3 [25] <0>     B14 <—> C6 B8
GRAM#3 [26] <4>     B12 <—> F'12
GRAM#3 [27] <3>     B12 <—> A3 B9
GRAM#3 [28] <0>     A3 B12 <—> A3 A4 B8
GRAM#3 [29] <1>     A6 B12 <—> A6 A4 B8
GRAM#3 [30] <0>     C6 B12 <—> C6 A4 B8
GRAM#3 [31] <0>     ) B12 <—> ) A4 B8
GRAM#3 [32] <4>     B12 <—> A6 B6
GRAM#3 [33] <0>     B12 <—> C6 B6
GRAM#3 [34] <0>     B10 <—> A4 B6
GRAM#3 [35] <0>     B10 <—> C6 B4
GRAM#3 [36] <3>     B9 <—> A3 B6
GRAM#3 [37] <0>     B9 <—> A6 B3
GRAM#3 [38] <0>     B9 <—> C6 B3
GRAM#3 [39] <1>     B8 <—> A4 B4
GRAM#3 [40] <3>     B6 <—> A3 B3
-------------------------------------------------------------------------------------------------------------
```

GRAM#4 [1] **RND**
GRAM#4 [2] <4> A3 <—> dhin--
GRAM#4 [3] <0> A3 <—> dha--
GRAM#4 [4] <2> A3 <—> dhagena
GRAM#4 [5] <1> A3 A3 <—> dhagenadhin--
GRAM#4 [6] <7> A3 A3 <—> dhagenadha--
GRAM#4 [7] <3> A4 <—> tirakita
GRAM#4 [8] <2> A6 <—> dha-dha-dha-
GRAM#4 [9] <1> A6 <—> dha-ta-dha-
GRAM#4 [10] <2> A6 <—> dheenedheenedheene
GRAM#4 [11] <1> A6 <—> dheenedha-dheene
GRAM#4 [12] <8> A6 <—> dheenedheenagena
GRAM#4 [13] <3> A6 <—> teeneteenakena
GRAM#4 [14] <1> A6 <—> dhatigegenaka
GRAM#4 [15] <2> C6 <—> tagetirakita

GRAM#5 [1] **RND**
GRAM#5 [2] <4> F'12 <—> dhatigegenakateeneteenakena
GRAM#5 [7] <0> Q24 <—> dhin--dhagenadha--
dhagenadhatigegenakadheenedheenagena
GRAM#5 [3] <12> F24 <—> tagetirakitadhin--
dhagenadhatigegenakadheenedheenagena
GRAM#5 [4] <1> F24 <—> tagetirakitagena-
dhagenadhatigegenakadheenedheenagena
GRAM#5 [5] <13> F'24 <—> tagetirakitadhin--
dhagenadhatigegenakateeneteenakena
GRAM#5 [6] <0> F'24 <—> tagetirakitagena-
dhagenadhatigegenakateeneteenakena
GRAM#5 [7] <13> F48 <—> dhin--dhagenadha--
dhagenadhatigegenakadheenedheenagenatagetirakitadhin--dhagenadhatigegenakateeneteenakena
GRAM#5 [8] <0> F48 <—> dhin--dhagenadha--
dhagenadhatigegenakadheenedheenagenatagetirakitagena-dhagenadhatigegenakateeneteenakena

---

Modelling music with grammars: formal language representation in the Bol Processor

Jim Kippen & Bernard Bel

Abstract

Improvisation in North Indian *tabla* drumming is similar to speech insofar as it is bound to an underlying system of rules determining correct sequences. The parallel is further reinforced by the fact that tabla music may be represented with an oral notation system used for its transmission and, occasionally, performance. Yet the rules are implicit and available only through the musicians' ability to play correct sequences and recognise incorrect ones. A linguistic model of tabla improvisation and evaluation derived from pattern languages and formal grammars has been implemented in the *Bol Processor*, a software system used in interactive fieldwork with expert musicians. The paper demonstrates the ability of the model to handle complex structures by taking real examples from the repertoire. It also questions the relevance of attempting to model irregularities encountered in actual performance.

Keywords

1. Formal grammars
2. Patterns
3. Membership test
4. Tabla drumming
5. Ethnomusicology