The identification and modelling of a percussion 'language', and the emergence of musical concepts in a machine-learning experimental set-up

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)

THE IDENTIFICATION AND MODELLING OF A PERCUSSION 'LANGUAGE', AND THE EMERGENCE OF MUSICAL CONCEPTS IN A MACHINE-LEARNING EXPERIMENTAL SET-UP

Jim Kippen & Bernard Bel

Abstract

In experimental research into percussion 'languages', an interactive computer system, the *Bol Processor*, has been developed by the authors to analyse the performances of expert musicians and generate its own musical items that were assessed for quality and accuracy by the informants. The problem of transferring knowledge from a human expert to a machine in this context is the focus of this paper. A prototypical grammatical inferencer named *QAVAID* (Question Answer Validated Analytical Inference Device, an acronym also meaning 'grammar' in Arabic/Urdu) is described and its operation in a real experimental situation is demonstrated. The paper concludes on the nature of the knowledge acquired and the scope and limitations of a cognitive–computational approach to music.

Keywords

formal grammars, stochastic automata, language identification, inductive learning, drumming, cognition, dialectical anthropology, ethnomusicology

GRTC / 311 / Septembre 1988

Jim Kippen & Bernard Bel

Computers and the Humanities, 23.3, 1989, pp.199-214

The identification and modelling of a percussion 'language', and the emergence of musical concepts in a machine-learning experimental set-up

Jim Kippen
Dept Social Anthropology and Ethnomusicology
Queen's University
Belfast BT7 1NN
Northern Ireland
eihe4874@v1.qub.ac.uk (JANET)

Bernard Bel
Groupe Représentation et Traitement des Connaissances
Centre National de la Recherche Scientifique
31 ch Joseph Aiguier
13402 Marseille cedex 9
France
grtc@frmop11.BITNET

The *tabla* drumming of North India has become the focus of experimental research into percussion 'languages' and musical cognition. (The metaphorical usage of 'language' for a musical 'system' is paralleled by a literal usage that refers to the ways in which many drum musics may be represented with spoken syllables.) During the first major stage of the research (1983-88) a computer program called the *Bol Processor*, comprising a special editor and an inference engine, was designed for a microcomputer portable enough for fieldwork on location. Fragmentary, and sometimes contradictory, information gathered from analyses of musicians' performances and their statements about musical structure was formalised as a knowledge base of transformational-generative rules. This knowledge base then acted as an initial hypothetical model of musical structure which was used to analyse performances as well as generate its own musical items (represented in syllables) that were assessed for quality and accuracy by the musicians. Informants' analytical observations were incorporated into the model in order to correct the machine's inadequacies and to form increasingly valid hypotheses of musical structure. (For the fullest detailed account of the development and application of the Bol Processor system, see Kippen and Bel, 1988, 1989.)

A large part of the tabla repertoire is characterised by improvisation, here taken to mean the repetition, substitution, and permutation of key strings of drum strokes that can be represented, conveniently, by verbal symbols (quasi-onomatopoeic mnemonic representations) called *bol*s that carry no semantic meaning: *dha, ge, ti, na, tirakita* etc. Many thousands of new strings, or 'variations', may be derived from a single given pattern, or 'theme'; importantly, only a small proportion of these would be considered musically viable, and a far smaller number, sometimes as few as five or six and only rarely more than twelve, would be played during the course of a performance (see Kippen, 1988a, pp.161-68).

Our task has been to generalise a descriptive model that expresses the underlying procedures for all possible correct variations. However, tabla improvisation is not nearly so systematic as either the literature (see Gottlieb, 1977, pp.53-58, and Sharma, 1981, pp.89-103) or indeed some musicians have indicated. Apart from stating that a piece has particular temporal and vocabular confines, musicians do not tend to verbalise rules or procedures (Kippen, 1987, p.180); and even when prompted to do so by researchers, most musicians will at best give incomplete and often ambiguous information about inhibited sequences and chunks of bols which they perceive to form complete units, or phrases. A tabla player's intuitive knowledge of correct improvisatory procedures is, it appears, very gradually absorbed over a period of many years spent both listening to and imitating his teacher, and by being corrected little by little; very similar, it would appear, to the way in which a child learns a 'natural' language.

The methodological priority of this research has been to forge a dialectical human-computer interaction that is fast, efficient, and also sensitive to contexts which are familiar to Indian musicians. Thus during experimentation the machine assumed the role of a student, and its utterances were subject to the musician's correction and control. This process was articulated by the analyst who was required to interpret musical information as hypothetical production

rules that modified the behaviour of the machine. The grammars produced in this way were, however, unlikely to reflect the full scope of the composition, and so the process was reversed and a new set of the musician's variations was submitted to the machine that was in turn requested to assess their correctness (*membership test*). Any variations left unrecognised suggested further generalisations of the grammar. Following a series of work sessions during which both synthesis and analysis modes were tested in turn and a reasonable stability had been achieved, it was assumed that the grammar was correct.

The Bol Processor was relatively successful as a descriptive and generative model of musical structure. This was due to its grammar format, a combination of phrase structure and pattern languages (Bel, 1988b, pp.6-9, Kippen and Bel, 1989). A large number of complex compositions in which context sensitivity and complex patterns play an important role were implemented as Bol Processor grammars. Furthermore, the interactive process was generally satisfactory and mutually rewarding: analysts gained insights into music while musicians were stimulated to think and comment in different ways about their musical decisions (see Kippen, 1988b, p.29). On the other hand, some of the initial grammars that were based on incomplete information and our personal presuppositions could not easily be corrected for two reasons: firstly, we tended to formalise generalisations at preliminary stages instead of systematically formalising independent production rules; secondly, there was no straightforward method for correcting a given grammar from an example rejected by a musician. Although the structure of compositions could be implemented relatively easily, the major difficulty lay in defining a vocabulary (i.e. chunks of symbols or the segmentation of strings) that would lead to simple (and preferably parsimonious) descriptions (regular or context-free grammars).

The success of the Bol Processor as an anlytical tool was clearly dependent upon the skill and intuition of the analysts in making generalisations and inferring knowledge without strict empirical justification. Wherever analysts are themselves conversant with the musical system they are studying, the likelihood that they will formalise their own assumptions increases. This danger must be emphasized as the implications have yet to be fully realised in much research, as evidenced by Baroni, Dalmonte, and Jacoboni (1987) who claim to have 'formulated rules which are not supported by concrete examples in the sample...we have assumed that the lack of an explicit example was due to a fortuitous absence in our limited sample rather than to any structural reasons'.

We have now embarked upon a second major stage of this research that is partly an attempt to address the shortcomings of work with the Bol Processor and partly a response to the latest technical and theoretical developments in knowledge engineering. Our aim is to automate the process of analysis using a new computer system in a machine-learning experimental set-up. We believe that the automation of the descriptive and analytical processes would not only make them more empirical, but also would promote a more direct interaction between the knowledge models of the informant and the computer, so minimising the need for analysts to perform the role of 'interpreter'.

Grammars as characteristic descriptors of rhythmic compositions

This research centres around the characterisation of sets of sentences representing all possible variations derived from a finite number of *themes* (i.e. certain kinds of tabla compositions, such as *qa'ida*, that lend themselves to improvisation: see below under *modus operandi* for further details). A *sentence* in a (monodic) drumming language is a string of symbols (bols) belonging to an alphabet Vt, including a special symbol '-' that expresses a silence of one time unit, and, where necessary, symbols that indicate changes of rhythmic density (duple to triple time etc.). Each set of variations is a formal language, i.e. a subset of Vt*, the set of all finite strings formed with Vt. We prefer to use the term 'sentence' for strings that, it may be presumed, belong to a formal language, even though those strings may not be attributed any 'meaning'. In this application we deal with *finite* languages because (1) the alphabet is finite, and (2) all strings have restricted lengths, usually one or two of the metric cycles in which they are set.

Given a finite set of sentences that appear in a tabla performance, it is always possible to assign to it a category so that finite fragments of the discourse form mutually exclusive classes. If a particular class is assigned to non-grammatical strings ('garbage' sentences), then the set of classes is a partition of Vt*. In this context, learning may be viewed as a heuristic search through a space of *Boolean descriptors* with the goal of finding *discriminant descriptors* for the classes. A Boolean descriptor is a function that returns 'yes' or 'no' when applied to any object. A descriptor may be termed discriminant of a class if it returns 'yes' for all objects belonging to the class and 'no' for those belonging to other classes. When all strings are attributed a class, discriminant descriptors may also be called *characteristic*. A *decidable* grammar G is a typical characteristic Boolean descriptor (also called an *acceptor*) of a language: it returns 'yes' for strings in the language and 'no' for other strings. This process is called a *membership test*. Decidability implies that the answer may be found after a finite number of computational steps. Languages generated by such grammars are called *recursive*. In the Chomskian hierarchy of formal languages, every *context-sensitive* (*type 1* or *length-increasing*) language is recursive (Révész, 1985, pp.94ff). Readers not familiar with formal language theory may wish to consult Révész's *Introduction to Formal Languages* (1985).

Language identification in the limit: a paradigm for inductive inference

The problem of finding discriminant descriptors for a finite set of strings partitioned in N classes can be reduced to N problems of inferring a grammar from positive (class-belonging) and negative (non-class-belonging) examples. This *presentation protocol* is typically that of an *informant*, the equivalent of an *oracle* in AI literature (Valiant, 1984, pp.1134-36). It may be imagined that the informant will spontaneously give negative as well as positive examples (*methodical* informant), or that he/she will reject some of the strings uttered by the (human or mechanical) learner on the basis of the currently guessed grammar (*request* informant). Gold (1967, p.467) has established that, given an ordered set of positive and negative examples, any grammar G belonging to an *enumerable* set of decidable grammars can be *identified in the limit*: for any information sequence the machine will make only a *finite* number of wrong guesses; success consists of eventually offering a correct guess from which there is no subsequent deviation. A set is enumerable if its elements can be arranged in some sequence where each element occurs at least once.

In many learning situations, only positive examples are supplied. This presentation protocol is referred to by Gold (1967, p.450) as *information by arbitrary text*. Under such conditions, identification in the limit can only be achieved for finite languages. Finite languages can be represented with rules in the format of *right-linear* (*type 3* in the Chomskian hierarchy of formal languages) grammars. One way to formalise regular grammars is to use exclusively rules in either

X—> a Y or Z—> b

format, in which *X*, *Y*, and *Z* are variables and *a* and *b* are terminal symbols. The set of terminal symbols is the alphabet of the language Vt. Variables, or 'non-terminal symbols', are represented with arbitrary symbols taken from a set Vn. A regular grammar can also be represented as a finite-state automaton, i.e. a directed graph in which *X*, *Y*, and *Z* would be state labels, and *a* and *b* transition labels. In other words, to rewrite *X* as *aY* is equivalent to jumping from state *X* to state *Y* following the transition labelled *a*. The second rule is represented as a transition from state *Z* to a final state that is also called an *empty* or *accepting* state of the automaton:

Fig.1

The state from which all paths originate is labelled *S*, the initial or *sentence* symbol in the grammar. To analyse a string, each of its component symbols (from left to right) is used as a

'road sign'. The string is grammatically correct if it is possible to move from *S* to an accepting state following all the road signs.

A grammar may be viewed as a *theory* characterising the set of acceptable sentences; therefore the learning process may be called *descriptive generalisation* (learning from observation) rather than *concept acquisition* (learning from examples) (Michalski, 1983, pp.90-91).

Part of the research into inductive learning is aimed at defining classes of representations that can be learned in a realistic sense (Valiant, 1984, Boucheron and Sallantin, 1988). All representations can be mapped to positive integers, and most results may therefore be derived from number theory and the *theory of recursive functions* (Blum and Blum, 1975, p.126, Andler, 1987, p.221, Osherson, Stob and Weinstein, 1986, pp.8-33). Results in the field of formal languages have been briefly presented in this section. The other aspect of research, namely machine-learning, deals with methodologies that are applicable to engineering: problems that are theoretically solvable must also be solved in reasonable time on existing (sequential) computers. This aspect will be dealt with in the next sections.

Vocabulary and meaning

A striking difference between music and natural languages is that music does not use words taken in a predefined lexicon, and no semantic 'meaning' as such may be attributed to a musical sequence. It is therefore delusory to look for a descriptive model of music based on intuitive reductions (see for instance Jackendoff and Lerdahl, 1982, 1983) if it is to be applied to contexts in which the concepts of word and word sequence are not formally defined.

Although musicians and analysts have intuitive ideas about 'meaningful' word sequences in a given set of variations, our experience has proved, at least in tabla playing, that general definitions of words fail to produce a significant segmentation in the absence of information specific to the grammar (Bel, 1987a). Segmentation can be approached as a multistage decision process, using positional characteristics of symbols in 'sentences' (Siromoney and Huq, 1988). This method works well for the analysis of a large number of short strings, given the fact that many of these strings are potential single words. If data consists of a small number of long strings then positional characteristics are less relevant (Bel, 1987a) and a meaningful segmentation must be inferred as a parallel to deep structure.

Learning strategy

A major aim of this research has been to design an *incremental* learning strategy under human control (questions and answers) by which words and word sequences can be identified in each specific grammar. Backtracking takes place whenever the informant disagrees with statements inferred by the machine on the basis of his/her previous decisions. In incremental learning, the system does not recompute the entire data set each time a new (positive or negative) example is supplied. Our system has been named 'QAVAID' (Question Answer Validated Analytical Inference Device), an acronym which is derived from the Arabic word *qava'id* used in Urdu by many Indian musicians to denote a 'grammar' for, or 'rules' of, composition and improvisation (Kippen, 1987, p.180) with the added notion of 'vocabulary'.

Presupposed knowledge

In QAVAID a small amount of generic knowledge is contained in the semantic of the description language (i.e. the format of grammars), and the knowledge specific to the grammar is acquired by the system both from the input data and through questioning. Before the first example is entered the machine possesses neither syntactic nor lexical knowledge. This approach may be called *strictly incremental*: the machine attempts to mimic a learning process using information sequences in exactly the same format in which they are transmitted by experts to students.

A very important aspect of this research is our attempt to unveil the process by which a student of tabla can acquire sufficient knowledge about the implicit grammar of a piece from a *very small* set of positive examples. To this effect, when the learning machine fails to infer satisfactory descriptions, it is essential to determine empirically how much additional knowledge would be needed to perform the same task successfully. In the example shown under *modus operandi*, we end by specifying the exact nature and amount of presupposed (musical) knowledge that helps to define the vocabulary of a set of variations in a typical learning situation.

'Tight-fit' generalisation and specialisation

An incremental learning machine performs a *generalisation* to accept a new positive example, or a *specialisation* to reject a negative example. Since the information sequence from the musician contains only positive examples (*text* presentation protocol), negative examples are only those produced by the system at the point where the informant is asked to assess the correctness of the machine's own productions (*request informant* presentation protocol).

Performing a generalisation on a finite-state acceptor amounts to merging some of its states (see *derived* grammars in Fu and Booth , 1975a, pp.98-99). On the other hand, specialisations have no simple computational representation. Therefore in a text presentation protocol it is safe to restrict generalisation to *tight fit* models that generate/recognise exactly the input sequence. The corresponding algorithm is the following:

> Let A_i be the finite-state acceptor recognising exactly L_i, the language formed with the *i* positive examples already analysed by the system. When the (i+1)th example e_{i+1} is proposed, let B be the acceptor recognising exactly e_{i+1}. Generalise A_i to A_{i+1} by merging the largest possible number of states between A_i and B so that A_{i+1} recognises exactly $L_i \cup \{e_{i+1}\}$. (See Fig.2 below under *modus operandi*)

If the largest possible number of states are merged then the minimum complexity of the description will be obtained.

The specialisation process (Bel, 1988a, 1988b) is informally described as follows:

> Given a finite-state acceptor A_i recognising exactly L_i, and a negative example e_n, delete and reconstruct paths in A_i until the language recognised is exactly $L_i - \{e_n\}$.

Inductive inference

Inductive inference is performed by QAVAID in the *informant presentation protocol*: the machine is required to generate variations not belonging to the input examples, and these are then submitted to the experts for an assessment of their correctness. This process may be summarised as follows:

> Given a finite-state acceptor A_i, merge states or construct new paths generating 'interesting' strings not recognised by A_i, where 'interesting' denotes domain-dependent evaluation procedures.

The actual inference takes place only when a tight-fit generalisation has been performed on a reasonably large set of examples provided by the expert: through this initial process, not only has the machine remembered all examples but it has also made assumptions about the segmentation of sentences and has inferred a vocabulary. A formal definition of inference in QAVAID may be found in Bel (1988a, 1988b). The methodology is based on Michalski's general paradigm for inductive inference (1983, pp.88-89) and will be explained briefly under *modus operandi* below.

The modus operandi of QAVAID with reference to a specific experiment

(a) Entering data and initiating the analysis

What follows is a demonstration of the first steps of the analysis by QAVAID of a set of variations (improvisations) gathered from the expert tabla player Ustad Afaq Husain Khan of Lucknow in a teaching/demonstration situation (May 1988). This sample sequence is based on a type of composition generically called *qa'ida*, in this case a very well-known piece (read left to right):

dha ti dha ge na dha tira kita dha ti dha ge dhee na ge na

dha ti dha ge na dha tira kita dha ti dha ge tee na ke na

ta ti ta ke na ta tira kita ta ti ta ke tee na ke na

dha ti dha ge na dha tira kita dha ti dha ge dhee na ge na

This piece is considered by many musicians to offer the widest scope for improvisation, and its variations are certainly complex to formalise despite the brevity of the parent composition: an acceptable grammar developed recently comprised no less than 138 rules (see Kippen and Bel, 1989). Whereas it was possible to represent formally any structure, we failed to find a systematic way of expressing the set of acceptable permutations, and any identification of the piece's vocabulary proved almost impossible despite the fact that relatively few bols were in use. This led, therefore, to the idea of developing methods for determining regularities in bol sequences.

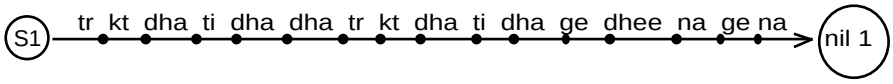
In North Indian drumming, there are structural constraints that indicate repetitions and 'open-closed' (*khula-band*) patterns where statements of voiced bols are partly or wholly transformed into unvoiced statements, e.g. *dhagedheenagena* to *taketeenakena* etc. (See Kippen ,1987, pp.180-81, and Bel, 1987c, p.355 for computer representations). It will be noticed that the second, third, and fourth lines of this qa'ida are repetitions of the first, though in part of the second and all of the third voiced phonemes are replaced by unvoiced ones. (In this way, the composition conforms to a voiced/unvoiced structure that is implicit in the metric cycle that governs it.) All that need be known for an appreciation of the current discussion is that while certain lines are left unaltered, others are transformed by repetition, substitution, and permutation. Thus, it will only be necessary to concentrate on altered lines. In the examples below, *tr* and *kt* are shorthand notations for *tira* and *kita*.

The set of input sentences (altered lines in variations) is the following:

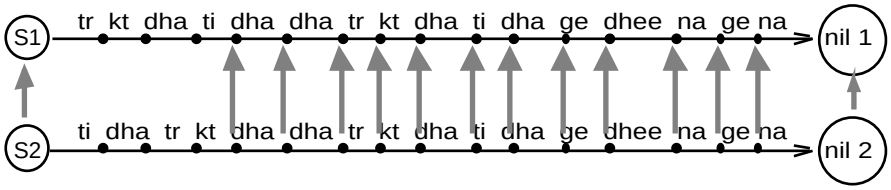
1. tr kt dha ti dha dha tr kt dha ti dha ge dhee na ge na
2. ti dha tr kt dha dha tr kt dha ti dha ge dhee na ge na
3. dha tr kt dha ti dha tr kt dha ti dha ge dhee na ge na
4. dha tr kt dha tr kt dha ge dha ti dha ge dhee na ge na
5. dha tr kt dha tr kt dha dha dha ti dha ge dhee na ge na
6. dha tr kt dha ti - dha ti dha ti dha ge dhee na ge na
7. ti - dha ti dha dha tr kt dha ti dha ge dhee na ge na
8. dha ti dha tr kt dha tr kt dha ti dha ge dhee na ge na
9. tr kt tr kt dha dha tr kt dha ti dha ge dhee na ge na
10. tr kt dha tr kt dha ge na dha ti dha ge dhee na ge na

The alphabet of symbols recognised by the system is {*tr, kt, dha, ti,* etc.}. The first stages of grammatical inference are summarised in Fig.2. *S* is the starting state and *nil1, nil2,* etc. the accepting states of the finite-state acceptor.

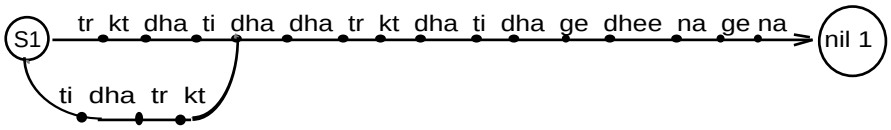
Take first example:



Take second example:



After merging states:



Take third example, etc... **Fig.2**

Heuristics are used to determine the minimum information that need be added to the finite-state acceptor in order for the new example to be accepted (Bel, 1988a). The final automaton is:

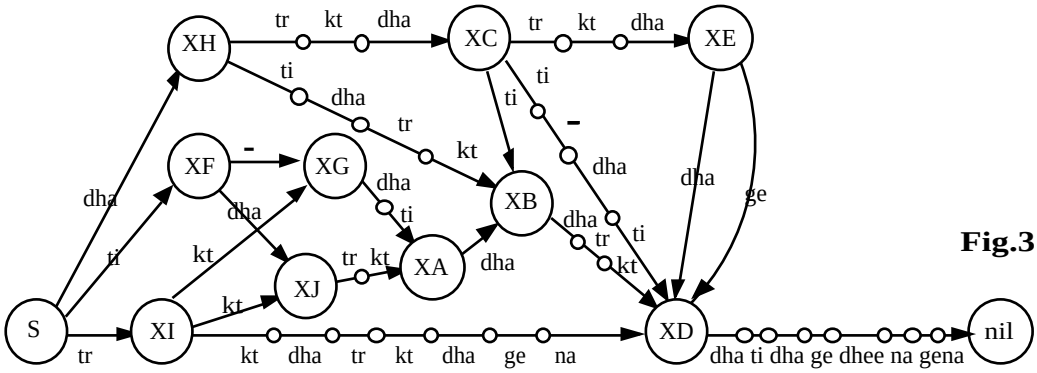
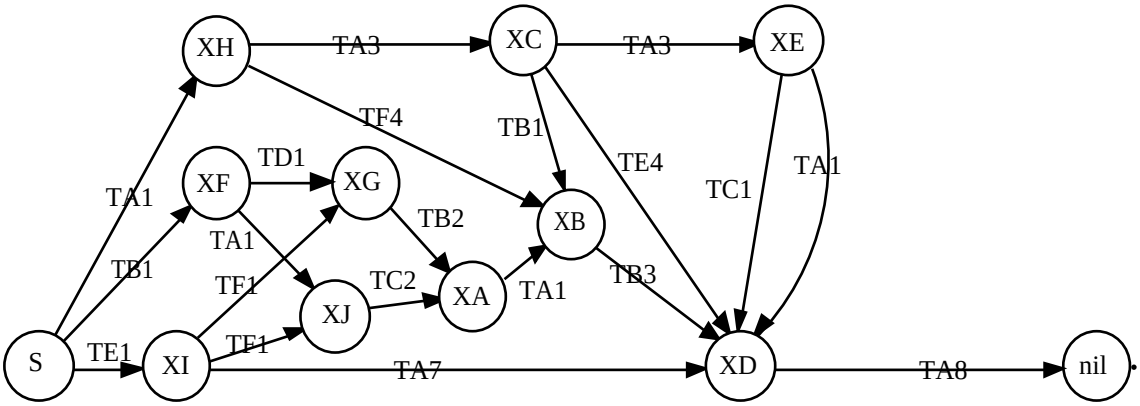


Fig.3

This representation is simplified in the sense that only those states that are (diverging or converging) nodes of the graph have been labelled. This suggests an alternative representation using a 'two-layer' acceptor, given in Fig.4:



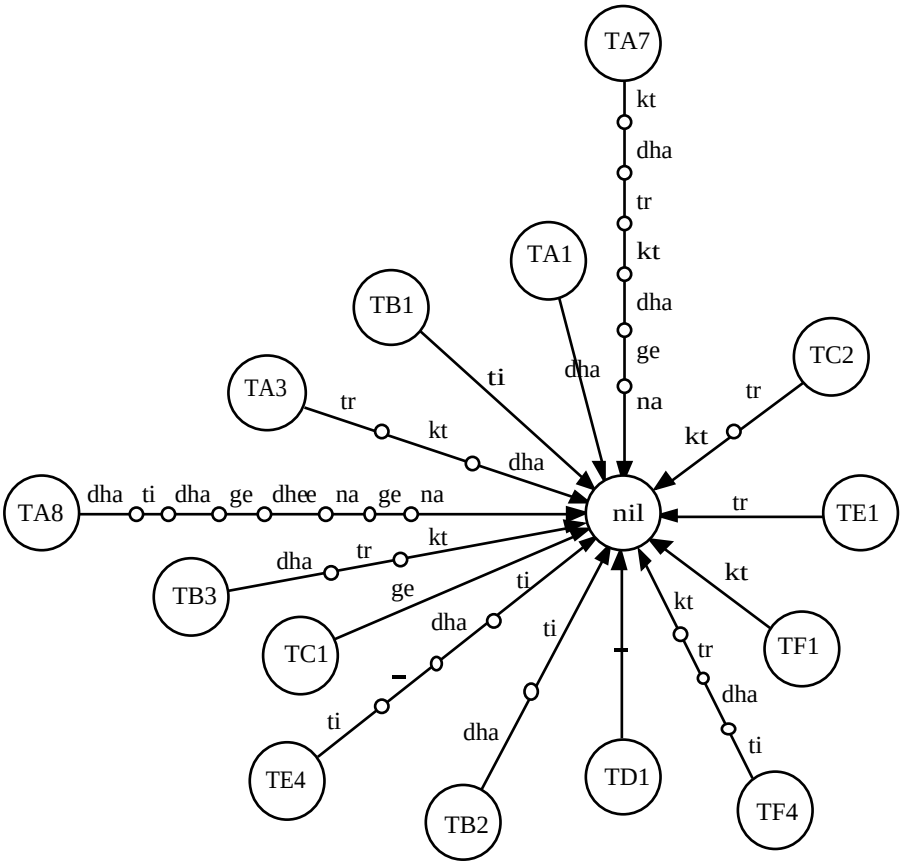


Fig.4

(b) The grammar generated

The grammar for the two-layer acceptor shown in Fig.4 is the following:

GRAM#1

S —> TE1 XI
XI —> TA7 XD
XD —> TA8
XI —> TF1 XJ
XJ —> TC2 XA
XA —> TA1 XB
XB —> TB3 XD
XI —> TF1 XG
XG —> TB2 XA
S —> TA1 XH
XH —> TF4 XB
XH —> TA3 XC
XC —> TE4 XD
XC —> TA3 XE
XE —> TA1 XD
XE —> TC1 XD
XC —> TB1 XB
S —> TB1 XF
XF —> TA1 XJ
XF —> TD1 XG

GRAM#2

TA7 —> ktdhatrktdhagena
TC2 —> trkt
TE1 —> tr
TF1 —> kt
TF4 —> tidhatrkt
TD1 —> -
TB2 —> dhati
TE4 —> ti-dhati
TC1 —> ge
TB3 —> dhatrkt
TA8 —> dhatidhagedheenagena
TA3 —> trktdha
TB1 —> ti
TA1 —> dha

This grammar is context-free since the right members of most rules in GRAM#1 contain two variables. Here it is split into two regular 'transformational' sub-grammars corresponding to

the two automata shown in Fig.4. The term ‘transformational’ is borrowed from formal language theory (Kain, 1981, p.24, Bel ,1987c, p.356), not linguistics.

Inferring a two-layer context-free grammar amounts to inferring its equivalent regular grammar. The difference between the two representations lies more in their explanatory power and formatting than in their properties: two-layer context-free grammars are compact and human-oriented representations of finite-state automata. Besides, the inference algorithm in QAVAID writes the two-layer context-free grammar directly.

(c) The vocabulary

TA7, TC2, etc. denote the vocabulary of the piece, and rules in GRAM#2 may be called *lexical* rules. It is interesting to compare this mathematical interpretation of vague concepts like ‘word’ and ‘segmentation’ with musical evidence: chunks like *trkt*, *dhatrkt*, and *dhatidhagedheenagena* may be called ‘words’ or ‘sequences of words’ in the sense that they represent blocks that can be substituted or permuted. However, a unit like ***kt*** is never used as a separate block but is always preceded by *tr*. Consequently, a block like *ktdhatrktdhagena* should not be thought of as a ‘word sequence’. Furthermore, within the vocabular confines of this piece, the same principle applies to *ge* which cannot be conceived of as a separate unit because it is always part of *dhage.* Notwithstanding this, *ge* was isolated during processing because the system looked for the longest string common to *trktdhadha* and *trktdhage*, resulting in *trktdha/dha* and *trktdha/ge*. A more meaningful split would have been *trkt/dhadha* and *trkt/dhage*. This confirms that the search for a meaningful segmentation cannot be reduced to a search for the set of the longest common strings in sentences. On the other hand, it is not easy to make a decision regarding possible segmentations of *ti-dhati* or *trktdha* without the wider context of adjacent parts of the string. Generally speaking, musicians are not able to formalise unambiguously the entire lexicon of most compositions.

It may be argued here that all one-symbol units, except *dha* and ‘-’, are not words because they are not interchangeable (i.e. they cannot be mutually permuted or substituted). A constraint designed to reject all one-unit words has therefore been implemented in the system. Thus, in this operational mode (‘chunk’ mode), *dha* and ‘-’ will also be attached to neighbouring blocks. Applied to the set of ten examples presented above, the ‘chunk’ mode led to the following vocabulary

trktdhatrktdhagena	
dhatidhatrktdha	trkt
ti-	tidha
dhati	dhatrktdha
ti-dhati	dhadha
dhage	dheena
dhatidhagedheenagena	gena

and the following finite-state acceptor (Fig.5) which may be compared with that shown in Fig.3.

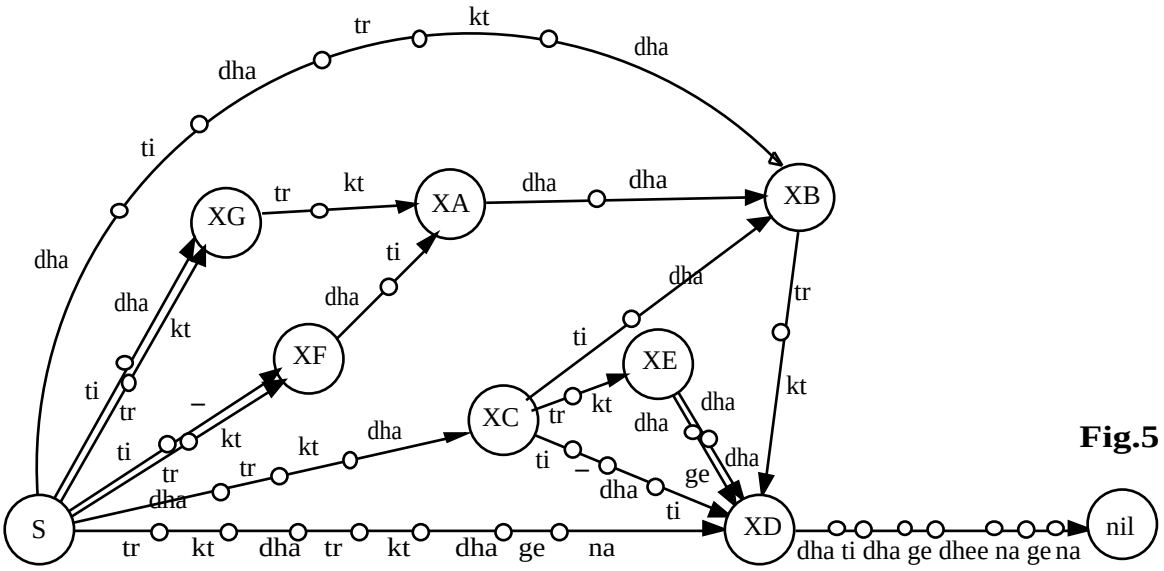


Fig.5

In order to assess the validity of this vocabulary in context we may look at the resulting segmentation of the input data:

1. trkt / dhati / dhadha / trkt / dhatidhagedheenagena
2. tidha / trkt / dhadha / trkt / dhatidhagedheenagena
3. dhatrktdha / tidha / trkt / dhatidhagedheenagena
4. dhatrktdha / trkt / dhage / dhatidhagedheenagena
5. dhatrktdha / trkt / dhadha / dhatidhagedheenagena
6. dhatrktdha / ti-dhati / dhatidhagedheenagena
7. ti- / dhati / dhadha / trkt / dhatidhagedheenagena
8. dhatidhatrktdha / trkt / dhatidhagedheenagena
9. trkt / trkt / dhadha / trkt / dhatidhagedheenagena
10. trktdhatrktdhagena / dhatidhagedheenagena

The segmentation of nearly all sentences is still unsatisfactory. Yet the results are not uninteresting for the very fact that they highlight new problems. For instance, take sentences 4 and 5: the machine has failed to recognise *dhatrktdhatrkt* as a repetition of *dhatrkt*. If it could be established that *dhatrkt* is a word beginning both sentences, then this would have important consequences for the segmentation of sentences 3 and 6 which begin similarly, and we would expect the machine to be able to achieve this revised segmentation automatically. If, on the other hand, the word has not been established, then the machine should ask questions to determine the best possible split of *dhatrktdhatidhatrkt* (example 3) and the other sentences.

(d) The dialogue with QAVAID

Musicians sometimes offer different solutions to the segmentation of the same string, solutions arrived at when they shift accents to create variety in the musical string of bols. The segmentation is apparent because the musicians tend to tap their fingers to reinforce the accentuation of the patterns. The acceptance of one split in preference to another may potentially lead to different structural descriptions later on, although these interpretations are

bound to converge in the process of inductive generalisation (see (f) below). For example, the segmentation of sentence 9 above is acceptable, but it might also be interpreted as

trkt / trktdha / dhatrkt / dhatidhagedheenagena or trkt / trkt / dhadhatrkt / dhatidhagedheenagena

with equal justification. Should the system be designed to handle such ambiguity? So far this has not been considered necessary in view of the adoption of the strategy that if the segmentation of a block is ambiguous in all possible contexts, then this block should remain an indivisable unit (a 'word').

The machine is also required to make decisions regarding 'specific paths', and this it does by questioning. As each new example is introduced, the part of the sentence which represents the 'new information' is displayed. Owing to the systematic nature of the information sequence, experts are intuitively aware of that information.

In the dialogue with QAVAID, an alternative answer to 'yes' exists in the 'optional yes' which requests the system to try to proceed. However, the system may come up with unacceptable proposals when exploring the consequences of this option. Answering 'no' to a question forces the system to backtrack. Backtracking to a 'try' decision results in retrospectively replacing the 'try' with the 'no' option. If all questions are answered with 'no', the system continues to backtrack until only the starting and final states can be merged (see Fig.2 above). In this case, the whole sentence is taken as 'new' information.

Below is an illustration of the dialogue established with the machine during the analysis of the ten examples:

[1] >> trktdhatidhadhatrktdhatidhagedheenagena

Specific path: /trktdhatidhadhatrktdhatidhagedheenagena/

Accept? Y)es N)o....Y

[2] >> tidhatrktdhadhatrktdhatidhagedheenagena

? trktdhati/dhadhatrktdhatidhagedheenagena

Y)es N)o T)ry R)eturn to preceding choice....Y

Specific path: /tidhatrkt/dhadhatrktdhatidhagedheenagena

Accept? Y)es N)o....Y

[3] >> dhatrktdhatidhatrktdhatidhagedheenagena

? dhadha/trktdhatidhagedheenagena

Y)es N)o T)ry R)eturn to preceding choice....Y

Specific path: /dhatrktdhatidha/trktdhatidhagedheenagena

Accept? Y)es N)o....N

? dhadhatr/ktdhatidhagedheenagena

Y)es N)o T)ry R)eturn to preceding choice....N

? trkt/dhatidhagedheenagena

Y)es N)o T)ry R)eturn to preceding choice....Y

Specific path: /dhatrktdhatidhatrkt/dhatidhagedheenagena

Accept? Y)es N)o....Y

[4] >> dhatrktdhatrktdhagedhatidhagedheenagena

? dhatrktdha/tidha

Y)es N)o T)ry R)eturn to preceding choice....N

? dhatrktdha/tidhatrkt

Y)es N)o T)ry R)eturn to preceding choice....N

? dhatrkt/dhatidha

Y)es N)o T)ry R)eturn to preceding choice....N

? dhatrkt/dhatidhatrkt

Y)es N)o T)ry R)eturn to preceding choice....Y
Specific path: dhatrkt/dhatrktdhage/dhatidhagedheenagena
Accept? Y)es N)o....Y
[5] >> dhatrktdhatrktdhadhadhatidhagedheenagena
? dhatrkt/dhage
Y)es N)o T)ry R)eturn to preceding choice....Y
Specific path: dhatrktdhatrkt/dhadha/dhatidhagedheenagena
[6] >> dhatrktdhati-dhatidhatidhagedheenagena
? dhati/dhatrkt
Y)es N)o T)ry R)eturn to preceding choice....Y
Specific path: dhatrktdhati/-dhati/dhatidhagedheenagena
Accept? Y)es N)o....N
? dhati/dhagedheenagena
Y)es N)o T)ry R)eturn to preceding choice....Y
Specific path: dhatrktdhati/-dhatidhati/dhagedheenagena
Accept? Y)es N)o....N
Specific path: dhatrkt/dhati-dhati/dhatidhagedheenagena
Accept? Y)es N)o....Y
[7] >> ti-dhatidhadhatrktdhatidhagedheenagena
Specific path: /ti-/dhatidhadhatrktdhatidhagedheenagena
Accept? Y)es N)o....Y
[8] >> dhatidhatrktdhatrktdhatidhagedheenagena
Specific path: /dhatidhatrkt/dhatrktdhatidhagedheenagena
[9] >> trkttrktdhadhatrktdhatidhagedheenagena
? tidha/trkt
Y)es N)o T)ry R)eturn to preceding choice....Y
Specific path: /trkt/trktdhadhatrktdhatidhagedheenagena
[10] >> trktdhatrktdhagenadhatidhagedheenagena
Specific path: /trktdhatrktdhagena/dhatidhagedheenagena
Accept? Y)es N)o....Y

One of the most interesting features of the dialogue is the backtracking relating to example 3. The machine's first attempt to segment the string results in a specific path that isolates a *trkt* from a *dha*. In view of the string's inherent symmetry of two *dhatrkt* separated by a *dhati* this appears unsatisfactory. In consequence, 'no' is returned. Further questioning establishes nothing more than a correct specific path recognising the variable part of the string and splitting it from the cadence common to all examples. The machine seeks further clarification during the analysis of examples 4 and 6, and several 'no' decisions are returned before a correct segmentation is reached.

Following the analysis, the resulting segmentation of the input data is:

1. trkt / dhati / dhadhatrkt / dhatidhagedheenagena
2. tidha / trkt / dhadhatrkt / dhatidhagedheenagena
3. dhatrkt / dhati / dhatrkt / dhatidhagedheenagena
4. dhatrkt / dhatrkt / dhage / dhatidhagedheenagena
5. dhatrkt / dhatrkt / dhadha / dhatidhagedheenagena
6. dhatrkt / dhati-dhati / dhatidhagedheenagena
7. ti- / dhati / dhadhatrkt / dhatidhagedheenagena

8. dhatidhatrkt / dhatrkt / dhatidhagedheenagena
9. trkt / trkt / dhadhatrkt / dhatidhagedheenagena
10. trktdhatrktdhagena / dhatidhagedheenagena

Knowledge on vocabulary is stored in the following format:

Recognised words:

trktdhatrktdhagena
tidha
dhatidhatrkt
ti-
dhati-dhati
dhagedheenagena
dhati
dhage
dhatrkt
dhadhatrkt
dhatidhagedheenagena
trkt
trktdhatidhagedheenagena
dhadha

Correct splits:

tidha / trkt
dhati / dhagedheenagena
dhati / dhatrkt
dhatrkt / dhage
dhatrkt / dhatidhatrkt
trkt / dhatidhagedheenagena
dhadha / trktdhatidhagedheenagena
trktdhati / dhadhatrktdhatidhagedheenagena

Incorrect splits:

dhatrkt / dhatidha
dhatrktdha / tidhatrkt
dhatrktdha / tidha
dhadhatr / ktdhatidhagedheenagena

(e) Inferring complete segmentation

The machine can be requested to check whether or not smaller significant chunks may be found in the input data, thereby completing the structural description of the input sequence. This leads to the following dialogue:

split dhatrkt / dhagena ? Yes
split dhati / -dhati ? No
split dhati- / dhati ? No
split dhage / dheenagena ? No

and to the segmentation:

1. trkt / dhati / dhadha / trkt / dhati / dhagedheenagena
2. tidha / trkt / dhadha / trkt / dhati / dhagedheenagena
3. dhatrkt / dhati / dhatrkt / dhati / dhagedheenagena
4. dhatrkt / dhatrkt / dhage / dhati / dhagedheenagena
5. dhatrkt / dhatrkt / dhadha / dhati / dhagedheenagena
6. dhatrkt / dhati-dhati / dhati / dhagedheenagena
7. ti- / dhati / dhadha / trkt / dhati / dhagedheenagena
8. dhatidhatrkt / dhatrkt / dhati / dhagedheenagena
9. trkt / trkt / dhadha / trkt / dhati / dhagedheenagena
10. trkt / dhatrkt / dhagena / dhati / dhagedheenagena

Here the system has split *dhati / dhagedheenagena* on the basis of a decision taken during the analysis of item 6 above. Nevertheless this sequence is known to be a common suffix to each sentence in this qa'ida. Therefore, any generalisation tending to modify it is likely to be irrelevant. To cut the search space of generalisations, it is possible to instruct the machine to relink *dhati* and *dhagedheenagena.* Segmentation is reversible since it does not alter the original finite-state acceptor. Once the segmentation process has been invoked, it is automatically re-activated each time a new positive example is entered.

Segmentation may be viewed as inferring syntactic knowledge: labelling particular states in the finite-state acceptor. Labelled states are eligible to have new paths branching from them, or they may be merged in a generalisation process. In segmentation, lexical knowledge is modified. For instance a new word, *dhagena*, has been inferred, and expressions like *trktdhatrktdhagena* that appear as sequences of words have been deleted from the lexicon.

(f) Generalising the grammar

The grammar given below is the outcome of the processes described above. For the sake of clarity, the system has relabelled the states (variables) appending numbers that represent their durations. For example, *SA10* is always derived as a string (a *suffix* sequence) of ten terminal symbols.

GRAM#1

S —> TA3 SA13
SA13 —> TA3 SA10
SA10 —> TC2 SA8
SA8 —> TD2 SA6
SA6 —> TA6
SA10 —> TA2 SA8
SA13 —> TC5 SA8
SA13 —> TD2 SB11
SB11 —> TA3 SA8
S —> TB2 SA14
SA14 —> TD2 SA12
SA12 —> TA2 SB10
SB10 —> TB2 SA8
S —> TE2 SA14
S —> TF2 SB14
SB14 —> TB2 SA12
S —> TB2 SB14

...

S —> TB2 SC14
SC14 —> TA3 SA11
SA11 —> TB3 SA8
S —> TD2 SD14
SD14 —> TA3 SB11

GRAM#2

TB3 —> dhagena
TF2 —> tidha
TE2 —> ti-
TC5 —> dhati-dhati
TA6 —> dhagedheenagena
TD2 —> dhati
TC2 —> dhage
TA3 —> dhatrkt
TB2 —> trkt
TA2 —> dhadha

Space limitations prevent us from showing in full detail the interaction between the expert and the machine. In brief, QAVAID tries to enlist states that may be merged or connected with new paths. Domain-dependent knowledge limits the number of states to be considered: since all sentences must be of the same duration, two states are eligible only if they denote suffix sequences of equal durations. In the grammar above, the state mergings to be considered are: *SA11 = SB11*, *SD14 = SA14*, *SD14 = SB14*, *SD14 = SC14*, *SB10 = SA10*, and *SA14 = SB14 = SC14*. The last merging will be considered first due to the three production rules

S —> TB2 SA14
S —> TB2 SB14
S —> TB2 SC14

that make it reasonable to suppose that suffix sequences derived from *SA14*, *SB14*, and *SC14* may be similar as their prefix *TB2* is identical. QAVAID also determines pairs of states that may be connected via new paths using the known vocabulary. This amounts to substituting or permuting words of equal lengths such as : *trkt / dhati / ti- / tidha / dhage / dhadha* and *dhatrkt / dhagena*. Thus, using background knowledge, the search space for possible generalisations is considerably reduced: only 8 out of 132 pairs of states will be examined, among which three are considered more significant.

The most practical methodology for generalising a grammar is to allow the machine to choose a possible generalisation at random and generate the next possible variation. The variation is then assessed to be either correct or incorrect by the informant, and the process begins again.

If, after some time, no further variation may be produced by a selected generalisation, the machine will suggest that the generalisation be validated. It is important to use a stochastic process in synthesis: if variations were to be enumerated in a strictly systematic order musicians would quickly lose interest in the experiment.

(g) Random music synthesis and the probabilistic model

Music synthesis in QAVAID may be enumerative (producing all variations in a predetermined order) or stochastic (producing at random one variation at a time). Random synthesis can reflect the probability of occurrence of sentences in the sample sequence from which the grammar was inferred. To achieve this, *weights* are assigned to production rules. These can be used by the system to compute *rule probabilities*. The probabilistic model used in QAVAID is the same as in the Bol Processor (Kippen and Bel, 1988a) and in probabilistic grammars (Booth and Thompson, 1973), and the algorithm for inferring weights is similar to that proposed by Maryanski and Booth (1977, pp.525-26). Rule weights may be viewed as flows on the directed graph representing the corresponding finite-state acceptor. Therefore, inferring weights amounts to reinforcing paths that are used more frequently.

In synthesis filters can be set to select sentences satisfying properties expressed in statements such as, for example, 'a sentence that has a repeated pattern longer than four symbols and which does not contain *dhagena*'.

Limitations of the present implementation and scope for research

During the course of this research, we have often been asked whether or not a machine (indeed, our machine) might ever reproduce the quality of composition or improvisation expected of human experts. The assumption behind this question is that there is an indefinable 'something else', an extra 'je ne sais quoi' that turns 'correct' or'acceptable' music into 'great' music or an ordinary composer into a Bach, Mozart, or Beethoven. The underlying implication is that a machine is not capable of 'genius'. It is not our intention to desanctify genius, but with regard to the musical system under investigation we hold that a number of musical effects that could be thought of as representative of high-quality composition may be analysed a posteriori in terms of, for instance, meta-patterns or polyrhythmic effects. Indeed these effects can be programmed in synthesis, but they cannot be learned by our system.

We believe, therefore, that it is crucial to devote time and energy to refining models of non-rigorous reasoning (principally induction and analogy) that serve as heuristics in straightforward language identification. At the moment, the grammars inferred from positive examples are tight fits to the sequence, and the inductive process itself is activated on the basis of a complete structural description of the input sequence. In forthcoming versions, QAVAID will search for 'relevant' regularities in the input sequence, and start hypothesizing generalisations while positive examples are still being analysed. This is similar to the behaviour of an intelligent pupil who tries to predict a theory even before all facts are known to him/her.

At a higher level, the machine should be able to infer knowledge from the order in which musical examples are supplied, especially in demonstration/concert situations; this would amount to describing *sequences* rather than unordered sets of sentences. The machine would then be able to construct meaningful sequences of variations in the same way sentences are ordered and linked in a discourse. This problem may be viewed as a variant of sequence extrapolation (Blum and Blum, 1975, p.126) in which the prediction is non-deterministic (Dietterich and Michalski, 1986, p.65).

Increasing the prediction power of QAVAID will inevitably mean considering larger sets of hypotheses, thus requiring more computation time and memory space. The prototype of QAVAID has been written in Prolog II, and is currently implemented on a Macintosh computer. Running the analysis of the ten examples shown above takes about 45 seconds on a Mac II. In developing more advanced versions we may use Prolog II+, a much faster version of Prolog II, or ICON, both of which can be operated under MPW on Macintosh with hard disk. It should be kept in mind that the main feature of QAVAID is its dialogue with informants which enables

the system to gather information about the domain and validate its hypotheses in an incremental learning situation. Once validated, hypotheses become part of the representational model (the grammar or finite-state acceptor) and therefore no backtracking is needed. In a similar way, QAVAID deletes each input sentence as soon as it has been used to update the grammar.

Limitations of a cognitive model of music

There are a number of very important aspects of musical 'behaviour' that are not taken into account in this study. One is the perception of time within a set metric pattern. Improvisation is, of course, performed in real time, and therefore a tabla musician must be aware, at any given moment, of the amount of time left to him and of the different methods at his disposal to chunk words together to fill precisely that amount of time in some meaningful way (Bel and Kippen, 1988). Complex calculations are often made with a rapidity that suggests musicians do not consciously calculate but rather rely on a kind of proleptic leap to a solution. Nevertheless, the decision a musician makes can be simulated arithmetically, a process that is normally not captured in a model based on symbolic representations.

This example points to the question of the psychological validity of cognitive models of music. Many studies in 'classical' Artificial Intelligence, and indeed this one, are based on the paradigm that *cognition* is *computational*, thereby placing emphasis on *reasoning* prior to perception, the latter being seen as a particular case of the former (Barr, 1983). In such a view, perception is considered formalisable and symbolic, whereas the learning process is reducible to serial computation. The approach that has emerged from the neurosciences, or more specifically connectionism, takes precisely the opposite view: emphasis is placed on *perception*, i.e. recognition, discrimination, classification, and association, whereas reasoning and symbolic representations are seen as emerging from a mental *state* which is itself the result of stochastic interactions in a homogenous (unstructured) network of units of memory (*formal neurones*). This process has been termed 'subcognition' as opposed to 'cognition' by Hofstadter (1985, p.659). Connectionist networks have proved efficient in achieving complex tasks, some of which (e.g. fundamental pitch extraction of complex tones) are associated with musical concepts. It is expected that there will be a formidable expansion of this field of research and applications while new machines based on asynchronous parallelism are being developed. Indeed, we are far from being able to model, let alone understand, 'subcognitive' aspects of music improvisation. But we also feel that it would be far-fetched to believe that new machines will solve the problem on the basis of their structural resemblance to parts of real brains. Our feeling, therefore, is that attempts to model musical 'behaviour' (improvisation, perception, and evaluation) will for some time to come continue to rely on the patient identification of concepts that are not pre-framed in a verbal description.

Conclusion

Experimental ethnomusicology has the potential to play a significant role in the development of new methods for the transfer of knowledge to computers because it deals with implicit (unformulated) models based on data which is quantified, bounded, and to some extent related and consistent. In North Indian drumming, the data is quantified with limited sets of symbols (the alphabet), languages are finite (bounded by the metre), language identification is related (each sentence is unambiguously assigned a parent theme from which it is derived), and musicians decisions in given contexts retain some kind of reliability (consistence). This paper has attempted to show that these properties make it possible to implement methods derived from the formal theory of inductive learning with the aid of algorithms that are computable in realistic time on existing machines.

At this stage, our project has started focussing more on the behaviour of informants and human/machine learners than on the content of the knowledge that is transmitted in teaching situations. In the same way that the Bol Processor was demonstrated to be useful in formalising some aspects of music production and evaluation, we expect that QAVAID will be helpful in describing some of the stages of the transfer of knowledge in the same context.

References

Andler, D. "L'apprentissage dans les sciences cognitives: approches théoriques". *Intellectica: Apprentissage et Machine*, 1 (1987), 213-34.

Baroni, M., R. Dalmonte, and C. Jacoboni "Relationships between music and poetry in the arias of Giovanni Legrenzi". (Unpublished paper) 1987.

Barr, A. "Artificial intelligence: cognition as computation". In *The Study of Information: Interdisciplinary Messages*. Ed. F. Machlup and V. Mansfield. New York: J. Wiley, 1983, pp.237-62.

Bel, B. *Grammaires de langages rythmiques*. Mémoire de D.E.A de Mathématiques et Informatique, G.I.A, Faculté des Sciences de Luminy, Université Marseille II, 1987a.

———*Les grammaires et le moteur d'inférences du Bol Processor*. Marseille: Note GRTC n°237, Centre National de la Recherche Scientifique, 1987b.

———"Grammaires de génération et de reconnaissance de phrases rythmiques". *Reconnaissance des Formes et Intelligence Artificielle*. Antibes: 6ème Congrès A.F.C.E.T, 1987c pp.353-66.

———*Mise en oeuvre d'un algorithme d'inférence d'automate probabiliste*. Marseille: Note GRTC n°303, Centre National de la Recherche Scientifique, 1988a.

———*Designing tools for knowledge representation in the anthropological study of a musical system*. Marseille: Note GRTC n°312, Centre National de la Recherche Scientifique, 1988b.

Bel, B., and J. Kippen. **"**Un modèle d'inférence grammaticale appliquée à l'apprentissage à partir d'exemples musicaux". *Neurosciences et Sciences de l'Ingénieur*. Marseille: Centre International de Rencontres Mathématiques, Université de Luminy, 1988.

Blum, L., and M. Blum**.** "Toward a Mathematical Theory of Inductive Inference". *Information and Control*, 28, (1975) pp.125-55.

Booth, T.L., and R.A. Thompson**.** **"**Applying Probability Measures to Abstract Languages". *IEEE Transactions on Computers*, C-22, n°5 (1973), pp.442-50.

Boucheron, S., and J. Sallantin**.** **"**Apprenabilité et Complexité". *Journées Françaises de l'Apprentissage*. Cassis: 1988, pp.21-37.

Dietterich, T.G., and R.S. Michalski**.** "Learning to Predict Sequences". In *Machine Learning: An Artificial Intelligence Approach*. Vol.2. Ed. R.S. Michalski, J.G. Carbonell and T.M. Mitchell. Los Altos: Morgan Kaufmann, 1986, pp.63-106.

Fu, K.S., and T.L. Booth**.** **"**Grammatical Inference: Introduction and Survey — Part I". *IEEE Transactions on Systems, Man and Cybernetics*, SMC-5, n°1 (1975a), pp.95-111.

———"Grammatical Inference: Introduction and Survey - Part II". *IEEE Transactions on Systems, Man and Cybernetics*, SMC-5, n°4 (1975b), pp.409-23.

Gold, E.M. **"**Language Identification in the Limit". *Information and Control*, 10 (1967), pp.447-74.

Gottlieb, R.S. *The major traditions of North Indian tabla drumming*. Vol.1. Munich: Katzbichler, 1977.

Hofstadter, D.R. *Metamagical Themas: Questing for the Essence of Mind and Pattern*. New York: Basic Books, 1985.

Jackendoff, R., and F. Lerdahl**.** **"**A Grammatical Parallel between Music and Language". In *Music, Mind and Brain: the Neuropsychology of Music*. Ed. M. Clynes. New York: Plenum Press, 1982, pp.83-117.

——*A Generative Theory of Tonal Music*. Cambridge, MA: MIT Press, 1983.

Kain, R. *Automata Theory: Machines and Languages*. Malabar: Krieger Publishing Company, 1981.

Kippen, J. **"**An ethnomusicological approach to the analysis of musical cognition". *Music Perception*, 5 (1987), pp.173-95.

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

———"Computers, fieldwork, and the problem of ethnomusicological analysis". *International Council for Traditional Music (UK Chapter) Bulletin*, 20 (1988b), pp.20-35.

Kippen, J., and B. Bel**.** **"**Can a computer help resolve the problem of ethnographic description?". Marseille: Note GRTC N°319, Centre National de la Recherche Scientifique, 1988.

———"Modelling music with grammars: formal language representation in the Bol Processor". Forthcoming in *Computer Representations and Models in Music*. Ed. A. Marsden and A. Pople. London: Academic Press, 1989.

Maryanski, F.J., and T.L. Booth**.** **"**Inference of Finite-State Probabilistic Grammars". *IEEE Transactions on Computers*, C-26, n°6 (1977), pp.521-36. [Some anomalies of this paper are corrected in Gaine, B.R. "Maryanski's Grammatical Inferencer". *IEEE Transactions on Computers*, C-27, n°1 (1979), pp.62-64.]

Michalski, R.S. **"**A Theory and Methodology of Inductive Learning". In *Machine Learning: An Artificial Intelligence Approach*. Vol.1. Ed. R.S. Michalski, J.G. Carbonell and T.M. Mitchell. Los Altos: Morgan Kaufmann, 1983, pp.83-134.

Osherson, D.N., M. Stob and S. Weinstein**.** *Systems That Learn*. Cambridge, MA: MIT Press, 1986.

Révész, G.E. *Introduction to Formal Languages*. New York: McGraw-Hill computer science series, 1985.

Sharma, B. *Taal prakaash*. (In Hindi.). 7th ed. Hathras: Sangeet Karyalaya, 1981.

Siromoney, G., and A. Huq**.** **"**Segmentation of Indus Texts: A Dynamic Programming Approach". *Computers and the Humanities*, 22 (1988), pp.11-21.

Valiant, L.G. **"** A Theory of the Learnable". *Communications of the Association for Computing Machinery*, 27, n°11 (1984), pp.1134-42.