

Can the computer help resolve the problem of ethnographic description?†

Jim Kippen*

Bernard Bel**

Introduction

Anthropology has for some time been experiencing a paradigmatic shift where arguably the problem of description has emerged as the main methodological issue, so supplanting the discussion of issues relating to generalisation and comparison (Holy 1987). Accordingly, anthropologists have become increasingly interested in the process of constructing reliable, accurate, and valid primary data in a systematic way rather than in what to do with information once it has been collected (Bernard et al 1986). This, in turn, has led to a degree of thoughtful introspection (as suggested, perhaps, by 'Ethnography as Autobiography' as the proposed topic for the 1989 ASA meetings) and to critical re-examinations (some quite vehement) of the ethnographic method and in particular of the status and authority of the ethnographic record (see, for example, Tedlock 1987, Watson 1987, and Sangren 1988). At the heart of the matter is the apparent paradox that, on the one hand, anthropologists claim to be concerned with the description of culture from the actors' perspective(s) while, on the other, they more or less acknowledge that ethnographic facts are 'joint constructions of the anthropologist and the people studied' (Holy 1987:7).

Recently, computational approaches have suggested possible solutions to the problem of ethnographic description: a problem that may be summed up as the search for ways to disentangle folk models from the joint constructions that are analytical models. In particular, *knowledge-based systems* have contributed to the development of formal structures for the representation and manipulation of *symbols* associated with particular physical and conceptual phenomena. A knowledge-based system comprises two components: a base of knowledge (facts and rules specific to the domain of application) and an inference engine allowing the solution of problems by transforming the knowledge base, using a mechanized model of reasoning (abstractly speaking, a *derivation schema*). Consequently, 'a problem may be formalised as an initial state, a goal state, and a collection of actions or of state transformation operators' (Chouraqui 1984:153). Expert systems, a kind of knowledge-based system in which inference is mainly of a hypothetico-deductive type, have played a major role in popularising Artificial Intelligence through practical applications, to the extent that there is a general tendency nowadays (outside the AI world) to call any computer program designed to imitate the behaviour of a human expert an 'expert system'. In this paper we will invoke this extended meaning to designate a knowledge-based system, the *Bol Processor* (BP), in which derivation schemata are borrowed from the theory of formal languages.

Although wary of being over-optimistic about the effectiveness of expert systems, Fischer (1986) has pointed out their potential for the manipulation of qualitative data and for interactive work with the human experts whose knowledge they are modelling. In support of this view, Kippen (1988a:318) has suggested that knowledge-based systems offer the prospect of more efficient analytical tools because their output is interpretable by experts in the domain described by the knowledge base, so that it can be evaluated in order to assess the accuracy

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

* Leverhulme 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: bel@frmop11.bitnet (EARN)

and relevance of the stored information. This suggests a new methodological approach: one in which the computer can be incorporated into the process of data collection, representation, and analysis in order to facilitate an 'apprentice-like' (cf. Emmet 1976:85) interaction between informants and analysts where priority may be given to the modelling of informants' and not analysts' views. Now, more and more anthropologists are adopting the view that, where appropriate, computer systems like expert systems should be integrated into the processes of data collection and analysis, at the very least as a supplement to the ethnographic and analytical enterprise.

This paper will test the claim that knowledge-based systems can be useful to anthropologists by describing one such system and its role in research recently conducted into the *tabla* drumming of North India. Musical knowledge in general, and the intuitive knowledge models of Indian musicians in particular, are appropriate fields of application for existing 'expert' systems because they represent (implicit) cultural knowledge that is quantified (through encoding), non-hierarchical, reasonably coherent, consistent, and above all bounded. The advantage of such domains is that they naturally lend themselves to reproducible experiments. Furthermore, experimental work produces important, yet often underestimated, feedback to computer science. As pointed out by Chouraqui (1984:154),

First, AI remains largely an experimental science; on the other hand, it involves the coming together of several disciplines. This state of affairs implies not only a deepening of theoretical concentration on the theories and methods borrowed from other disciplines for the computational modelling of reasoning, but also the development of a theoretical and experimental programme of research with the aim of mechanizing and validating models developed as part of applications relating to the real world. But a large number of pieces of research in AI have rested upon conventional and isolated examples. This is probably one of the causes of the difficulties which this field of research has faced in the last few years in solving the — often ambitious — problems which it has set itself. In fact the choice of application domains is one of the principal elements contributing to the way any experimental advance defines its epistemological framework.

In setting up an experimental programme involving computational techniques there are problems that must not be overlooked as they require experience both in the field of application as well as in knowledge engineering: (1) the choice of a description language for data; (2) appropriate rule formats; (3) the design of domain-specific heuristics in the manipulation of derivation schemata; and (4) an experimental methodology that makes it possible to transfer knowledge from human experts to machines.

The general methodology of the research

The *tabla* (two-piece, tuned drum set) is the most popular drum of North India, and is widely used both as an accompanying and solo instrument. A large part of its highly-developed solo 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 called *bols* that carry no semantic meaning: *dha*, *ge*, *ti*, *na*, *tirakita* etc. Many thousands of new strings (variations) may be derived from a single given pattern (or 'theme') of, say, sixteen *bols*, yet 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 at any one time (see Kippen 1988b:161-68).

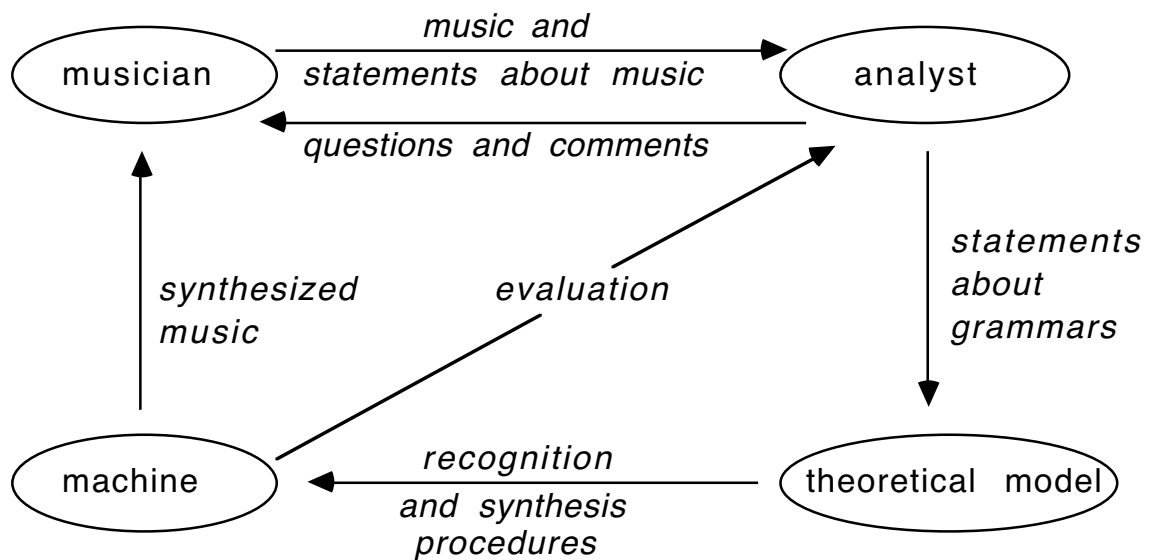
Initial attempts to involve a computer in the research resulted in the redesign of a word processor for *bol* notation (hence the name *Bol Processor*). A key-correlation system was created whereby a key signified a *bol* and not a single character, facilitating error-free and fast typing: thus only one byte was needed to encode *dha* or *tira*. This led to the idea of using search-and-replace procedures arranged in pre-programmed sequences (e.g. Applewriter's *Word Processing Language*) where each sequence emulates a rewriting process by which a variation is presumably derived from a theme. An elementary rewriting process may be represented as a *production rule*, and sets of such rules are arranged in formal grammars in

which all variations plus the theme itself are derivations of a unique starting symbol. In order to generate a whole set of variations, the choice of derivation sequences needs to be non-deterministic (enumerative or stochastic). The actual process is therefore accomplished under the supervision of a module called an *inference engine*. The inference engine (and the editor) of the BP have been implemented in assembly language on the only microcomputer available that was portable enough for fieldwork on location at the time we started the project (in 1983, the Apple IIc).

A detailed description of the Bol Processor and its operation may be found in Kippen & Bel (1988b). We will restrict here to the main features of BP grammars, pointing out at similarities and differences between derivation schemata based on grammars and those based on logic.

Music synthesis with the BP is similar to *modus ponens* in expert systems: from an initial state represented as the starting symbol (the *axiom*) the machine is requested to derive a sentence — a set of terminal symbols (a *theorem*). The sentence displayed on the screen is recited by the analyst so allowing expert musicians to assess its quality and accuracy. However, to prove that the current model is representative of the full scope of the piece under investigation, it is necessary to invoke a proof procedure (*modus tollens*, or *membership test*) for any sentence proposed by the informant as a correct variation.

The interaction between the expert, the machine, and the analyst in Bol Processor experiments may be summarised in the following flowchart:



Informants' analytical observations are thus incorporated into the model in order to correct the machine's inadequacies and to help in the formulation of increasingly valid hypotheses of musical structure.

Even though calling the BP an 'expert system' may be argued if one compares its architecture and derivation schemata with those of current expert systems, this flowchart highlights similarity in terms of *knowledge acquisition*, a typical experimental process. Knowledge acquisition in expert systems is primarily based on the interpretation of interviews between expert(s) and a knowledge engineer. A closer interaction is possible in BP experiments as the expert gets an immediate feed-back of his attempts to improve the model. This methodology has been suggested by dialectical anthropologists: models should be elaborated and evaluated by informants themselves.

Readers who wish to be introduced to (non-technical aspects of) knowledge acquisition in expert systems may consult Anna Hart's *Knowledge acquisition for expert systems* (1986). Besides, an excellent statement on how much AI may benefit from ethno science(?s) in the

field of knowledge acquisition has been proposed by anthropologist/computer scientist Claude Vogel in his book *Génie cognitif* (1988).

The BP is not an attempt to model human behaviour on the basis of rules explicitly formulated by human experts and rewritten to a suitable machine code by an analyst. Music improvisation is mainly the result of ‘subcognitive’ processes that cannot be reduced to (a finite set of) verbal statements. The aim of a computational model is to analyse as well as generate acceptable music, its performance in both tasks being improved during interactive experiments highlighting those aspects of music that human experts consider most relevant. If the BP were merely a machine producing random music, it would barely be more refined than any piece of software claiming to generate ‘poetry’ by shuffling a small set of words in which all semantic relations are predefined (Roszak 1988:97ff). The outcome of the analytical process — the major part of our experimental method — is that acceptable sequences of bols are assigned deep structures. Evaluating a sequence, therefore, is a pattern recognition process that may be paralleled to a semantic operation in computational linguistics. This is supported by our informants’ views that incorrect drumming sequences have no ‘meaning’. Indeed, this relates to an aesthetical evaluation rather than to the information that is supposedly conveyed by the sequence.

The description language in the BP

In many knowledge-based systems, first-order logic is used for representing facts and knowledge about facts. This logic is closest to natural languages (e.g. English) and derived from set theory, making use of the elementary Boolean operations (‘and’, ‘or’, and ‘not’) and quantifiers (*universal* = ‘for all...’ and *existential* = ‘there exists...’). For instance, a predicate like ‘*Socrates is a man*’ may be represented as:

man(Socrates)

and a production rule like ‘*Every man is mortal*’ is translated

for all X, mortal(X) or not(man(X))

or equivalently

not (there exists X, man(X) and not(mortal(X)))

which may be written in *natural logic* :

man(X) \rightarrow mortal(X)

Using this rule, *natural deduction* consists of inferring *mortal(Socrates)* if *man(Socrates)* is present in the base of knowledge.

In the BP, production rules are elementary *rewritings* (or *derivations*). A derivation is similar to inference in natural logic except that the expression unified with the premise of the rule is erased after the inference. Applied to logic, a derivation would erase the fact *man(Socrates)* as soon as *mortal(Socrates)* has been inferred. In addition, derivations are applied to *strings*, which may be viewed as ordered sets of facts (symbols). Each rewriting is therefore one step of a *derivation sequence* of the *work string* characterising the current *state* of the problem. For instance, if the current work string is *T2 T1 T1 T2*, applying rules

(1) T1 \rightarrow dha

(2) T1 \rightarrow -

(3) T2 \rightarrow tite

(4) T2 \rightarrow dheena

in order 3, 1, 2, 4 will result in any of the following derivation sequences

T2 T1 T1 T2 \Rightarrow tite T1 T1 T2 \Rightarrow titedha T1 T2 \Rightarrow titedha- T2 \Rightarrow titedha-dheena

T2 T1 T1 T2 \Rightarrow tite T1 T1 T2 \Rightarrow tite T1 dha T2 \Rightarrow tite-dha T2 \Rightarrow tite-dhadheena

T2 T1 T1 T2 \Rightarrow T2 T1 T1 tite \Rightarrow T2 dha T1 tite \Rightarrow T2 dha-tite \Rightarrow dheenadha-tite

T2 T1 T1 T2 => T2 T1 T1 tite => T2 T1 dhatite => T2 -dhatite => dheena-dhatite

depending on the position of the derivation. In this case, the same initial states and rule ordering result in different final states. Among these derivation sequences, the first is called a *leftmost* derivation. It is often realistic to assume that sentences are built from left to right, in which case a grammar may be instructed to perform leftmost derivations, using the instruction 'LIN' (see *GRAM#3* and *GRAM#4* in the appendix).

Multi-stage inference and transformational grammars

In logic inference original facts are not erased; repeated application of production rules on original and deduced facts may therefore lead to combinatory explosion. Consequently, heuristics are used to explore *deduction paths* in such a way that the first facts deduced are those that an expert would consider most 'meaningful'. Heuristic deduction strategies which are not domain-dependent may be expressed in second-order logic (Laurière 1984:15), i.e. rules that modify rules.

One of the methods for controlling inference in modus ponens consists of splitting the base of rules into subsets that refer to several deduction stages (e.g. *FIN ETAPE* in expert system SNARK, Laurière 1984:11). This amounts to dividing a problem into sub-problems with intermediate goals. In the same way, derivation sequences in a formal grammar may be seen as distinct stages defined with the aid of several layers of *transformational grammars* (*subgrammars*). We use the term 'transformational' as defined in formal language theory (Bel 1987b:356) and not linguistics. In a subgrammar there is more than one starting symbol, and a derivation sequence in that subgrammar consists of transforming a string of starting symbols to a string of terminal symbols. Symbols that are terminal to a subgrammar may become the starting symbols of the next subgrammar(s) to be applied, and so on.

The grammar in appendix 2 is layered in five subgrammars. For instance, *T1* and *T2* are the terminal symbols of subgrammar #3 and the starting symbols of subgrammar #4. *S1F* is a terminal symbol of subgrammar #1 and a starting symbol of both subgrammars #2 and #5. The inference engine takes one subgrammar at a time, attempting to derive all possible transformations from its rules. Only then it considers the set of rules in the next grammar. The splitting of grammars to subgrammars is mainly conceptual, but there are also restrictions on formal languages that render it necessary.

In modus ponens, subgrammars are needed if some variables are used as contexts. For example, rules in subgrammar #2 are context-sensitive. Should all subgrammars be merged, a variable like *S1F* might be derived too early if rule [7] of subgrammar #5 (see appendix) was used. This would lead to the following derivation sequence:

S → S64 → S1F S2F S1V S2F E32 → dhadhatitedhadhadheena S2F S1V S2F E32

ending in a *cul-de-sac* as no further derivation is possible. In theory, this does not disqualify the grammar, because backtracking would take care of finding successful derivation paths, but in practice much computing time may be wasted in this process.

In modus-tollens (see below under 'membership test'), subsets of rules are determined in such a way that rules recognising large patterns are considered first. See for instance subgrammar #4 in the appendix: small patterns like *dhadha*, *tite*, *teena*, etc. may be identified individually only if they are not part of larger patterns such as *+dhadhateena*, etc. Since these larger patterns belong to subgrammar #5, which is considered first by the inference engine, they can be recognised correctly.

Rule format in the BP

In computational linguistics, the most common grammar format is *context-free* (type 2 in Chomsky's classification), i.e. the premise of a rule must be a single variable. The main

reason for adopting this format is the existence of efficient parsing techniques for context-free grammars. Indeed, all languages representing tabla compositions are finite (bound by the metre) and therefore can be represented with context-free grammars. Yet context-sensitive rules describe more clearly the dependence of derivations on information defining certain positions within the musical structure. For instance in subgrammar #5, symbols ‘++’ and ‘+’ are used as contexts to mark, respectively, the beginning and middle of a phrase and the position of a cadence.

In some cases, non-restricted (type 0) grammar formats may even be used as they provide a more compact representation. For instance, subgrammar #3 could have been written

```
V15 → A A A A A A A A A A A A A A A A A
V14 → A A A A A A A A A A A A A A A
V13 → ...
...
V6 → A A A A A A
A → T1
A A → T2
```

in which the last rule is type 0 (length-decreasing). Yet using a context-free grammar here was the only way to suppress selectively subsets of sentences that do not fulfil certain musical constraints (e.g. accentuation, see Bel 1987b:361). In addition, context-free grammars are best suited to probabilistic inference (see below).

The BP grammar format is also designed for representing *patterns* (Kippen & Bel 1988b): a pattern is a repeated sequence, in which the transformation voiced/unvoiced may occur (see Kippen 1988b:162-64). For example, the repetition pattern in *dhadhatitedhadhatite* may be explicitly represented as $(=dhadhatite)(:dhadhatite)$ and the voiced/unvoiced transformation in *dhadhadheenatateena* as $(=dhadhadheena)*(:tateena)$. Rules with explicit pattern structures are found in subgrammar #2 (see appendix 2). If a rule is written:

$$A \rightarrow (= B) (: B)$$

these two occurrences of variable B will be forced identical derivations.

Brackets and other structural symbols (‘=’, ‘:’, ‘*’, ‘+’, etc.) are removed automatically from final sentences: sentences are displayed with a lay-out suitable to their time structure, so that they can be quickly read by the analyst (see appendix 1). If, on the other hand, structural symbols are kept in a final sentence, and all terminal symbols replaced with dots, a description of the pattern structure is obtained which we call a *template*. For instance, the template of $(=dha dha dhe e na)*(:ta ta tee na)$ is:

$$(= \dots) * (: \dots)$$

The six templates, i.e. pattern structures, of all sentences generated by the grammar are listed in appendix 3. These have been generated by the inference engine, and will be needed for membership tests (see below).

A probabilistic model

In modus ponens, the BP may be instructed to produce the set of all possible sentences that the current grammar generates. In practice this non-deterministic enumerative process is rarely used as the set is very large. The only realistic method for testing a grammar with an expert musician, therefore, is to instruct the machine to produce one randomly chosen sentence at a time. If the sentence is assessed correct, the procedure is called again and another sentence is generated. Generally, the grammar is considered to be satisfactory if all sentences generated within a few sessions have been accepted by the expert.

Since the correctness of a grammar can never be fully assessed — indeed, like musicians

themselves, machines may be allowed casual mistakes — it is important to enable the stochastic production process to generate sentences from a wide and representative subset of the language. This can be achieved by weighting the decisions of the inference engine.

Unlike representations in logic, formal grammars offer the possibility of defining consistent probabilistic models that apply to a very wide class of languages. We developed such a model as a response to the need to inhibit rules in context-free grammars. This model 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 context-free rule. The rule probability is computed from weights 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, the set of rules

[4]	<100>	T2	—>	dhadha
[5]	<100>	T2	—>	tite
[6]	<50>	T2	—>	dheena
[7]	<50>	T2	—>	teena

in which the sum of the weights is $100+100+50+50 = 300$. The probability of choosing rule (6) in the derivation of a string containing *T2* is therefore $50/300 = 0.166$. Using weights instead of probabilities has the advantage that it does not presuppose the sum of coefficients of all candidate rules to be 1.

Weights (and their associated probabilities) are used in *modus ponens* to direct the BP's production along paths more or less likely to be followed by musicians. In some context-free grammars — those that fulfil the *consistency* condition expressed by Booth & Thompson (1973:442) — they may be used for computing a *probabilistic sentence function*, i.e. a coefficient representing the likelihood of occurrence of each sentence in the language. Grammars that are constructed in a systematic way (like the one defining all sequences of *T1* and *T2* in the above example) are good examples of consistent probabilistic grammars.

Another remarkable feature of consistent grammars is that rule probabilities can be inferred from a set of sentences (Maryanski & Booth 1970:525). Given a correct 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 are 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 algorithm described here is more general than the one devised by Maryanski and Booth, since the latter requires the choice of a sample set in which *all* rules have been used.)

Evidently, rules that are never used in the analysis of the sample set remain with weight (and therefore probability) zero, which inhibits their use in *modus ponens*. Those rules may be scrutinised to see whether they are incorrect or whether they point to fragments of the language that have not yet been explored. To test this, their weights are set to a high value so that the BP is forced to generate sentences that either have never been assessed or at least are not being 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.

The problem of the membership test

Modus-tollens is much more demanding on rule format, as reflected dramatically in first-

order logic: if a fact is provable then it can be proved in finite time (the logic is *complete*), but if it is not provable then the resolution algorithm may loop for ever (the logic is not *decidable*). Therefore, expert systems are generally based on a decidable subset of first-order logic for which there exist efficient proof procedures. One of the most common rule formats is that of *Horn clauses*, for which a proof algorithm is based on Robinson's *resolution principle* — the actual foundation of the Prolog programming language (Robinson 1968, Colmerauer 1983). In a Horn clause, the conclusion of a rule must not contain any disjunctive form. For example, a rule like 'every child is a boy OR a girl' cannot be inserted in the knowledge base of most expert systems.

If for some application domain a different logic is required, for which there is no efficient proof procedure, *heuristics* are used to make sure that in most cases the proof will be computed in realistic time.

Formal grammars used in the Bol Processor must be decidable (or *recursive*): given a grammar G and a sentence P, there must be a procedure to assess (in finite time) whether or not P belongs to the language generated by G. The theory indicates that context-sensitive (or *length-increasing*) grammars are decidable (Salomaa 1973:82-93), although there are also length-decreasing grammars that share this property (Bel 1987a:4). A grammar is length-increasing if the conclusion of each rule contains a number of symbols larger than or equal to the premise.

A membership test must be performed at high speed and use a minimum of memory space. In experimental situations, the expert musician proposes a new variation which he expects the machine to evaluate. The variation can be typed almost at performance speed using the keyboard encoding (one key to one syllable). Then the machine is requested to give its opinion within a few seconds, failing to which experts would quickly lose interest in the game... The BP membership test, therefore, needed to be computed at high speed within the very small amount of memory available in the Apple IIc.

The heuristic on which the test has been based is that, in most cases, large 'chunks', i.e. substrings of the input sentence, must be recognised first. We call this the 'chunk' rule. Following this principle it has been possible to implement a deterministic bottom-up (data-driven) parsing, given an algorithm for selecting the proper rule and position of derivation at each step of the derivation. This algorithm is based on *context-sensitive rightmost derivation* as defined by J.M. Hart (1980:82) for *strictly* context-sensitive grammars, i.e. rules in which the premise contains no more than one variable, which we extended to all length-increasing grammars (Bel 1987b:357). The derivation sequence used in the membership test rewrites symbols from right to left in the work string. For any correct sentence of length N, the analysis is completed in less than N steps. See appendix 3 for examples of derivation sequences.

One major advantage of the parsing technique in the BP is that it is *deterministic*: if at some point of the derivation a dead end is reached (no further derivation possible) then the test is negative. It is therefore very easy to follow this process step by step and to understand why it was unsuccessful. This fulfills a requirement of expert systems that an *explanation* of failures pointing to a deficiency of the base of rules is needed.

The 'chunk' rule imposes a few restrictions on grammar format, the main one being a partial ordering of rules. For instance, in subgrammar #4, *dha* is a substring of *dhadha*. Should rule 2 be considered before rule 4, then *dhadha* would never be recognised. To facilitate this, the inference engine gives priority to rules at the bottom of each subgrammar whenever the derivation is ambiguous; consequently, rule 4 will be considered first.

There is also a non-deterministic feature in the membership test algorithm, and this relates to patterns and templates (see above under 'rule format'). Before a sentence is parsed it must be written with the symbols that indicate its pattern structure: brackets, '*', etc. This is

performed automatically by the inference engine: taking one template after the other, it attempts to write the terminal symbols of the sentence over the dots, checking that strings are structurally correct. For each acceptable template the membership test is invoked (see appendix 3). This template matching may result in assigning several hypothetical structures if the sentence is structurally ambiguous, each of which may also be assigned a probability.

Language modelling: putting theory into practice

Features imbedded in BP grammars, especially pattern representation, point at deficiencies of simple models of formal languages when essential aspects of music need to be represented in a comprehensive way. Simple models are attractive because they have known mathematical properties, e.g. context-free grammars are easy to parse. Still, since no natural language — and indeed no musical language — is context-free, a more general model is required. In computational linguistics, using Prolog's tree structures instead of simple variables has led to *metamorphosis grammars* (Colmerauer 1978) which have a wider generative power than context-free grammars. For technical reasons we developed the BP grammar model on an easy-to-parse subset of unrestricted languages in which patterns are represented in a simple manner. (In fact, implementing just the kernel of Prolog II requires more memory space than the whole BP system itself.)

At present we are developing the BP theoretical model to make it fit with more general models of music, e.g. polyphonic structures, for which no computer representation is available that would describe the management of time *as perceived by musicians*. This points at the difference between prescriptive human-oriented models of music, that provide insights to musical structures, and machine-oriented models that are mainly aimed at generating musical sequences.

Strictly speaking, the Bol Processor is a *programming language* and its inference engine is the *generic* part of knowledge about a certain type of music representation. A grammar is *specific* knowledge, yet not just 'frozen' data. It may be seen as the *dynamic* description of a potentially very large set of data. Therefore a grammar is a *descriptive generalisation* inferred from a very incomplete, although certain, set of facts (the examples provided by experts).

Declarative and procedural knowledge in the Bol Processor

There has been a controversy in cognitive science as to whether knowledge should be viewed primarily as declarative or procedural. Declarative knowledge is formatted as unordered 'knowing that' assertions like "*all children like candies; Jim is a child; candies are cheap...*", whereas in a procedural representation emphasis is put on 'knowing how': "*to make a phone call, first insert a coin, then...*". Declarative knowledge is the essence of logic representations; this has led to the concepts of logic programming (the Prolog language) and production-rule based systems (e.g. expert systems or the Bol Processor). In a pure declarative programming environment, the order in which bits of data have been encoded has no bearing on the result of the computation. In other words, the specification of the problem, e.g. 'generate a sentence', is clearly separated from the method of solution (Kowalski 1985:82). Procedural programming, on the other hand, amounts to defining methods by which objects are created, deleted, or interact with each other. It is found typically in procedural languages (e.g. Lisp or Logo) and as part frame-oriented knowledge bases. Indeed, in most recent AI applications both declarative and procedural representations are used in some non-conflicting way.

Part of the success of the Bol Processor as a model of music improvisation and evaluation lies in the fact that its grammars may be viewed as both declarative and procedural. In modus

ponens, the order in which rules are arranged in the grammar is irrelevant: knowledge is therefore declarative. This suits experimental situations in modus ponens where production rules are inferred from unrelated bits of information — sets of positive instances of the language. In modus tollens, however, the inference engine requires that rules be partially ordered, as this order will be reflected in the derivation sequence that yields the answer to the membership test. The step by step parsing of a sentence may therefore be viewed as a *procedural* description of its inner structure, in which each rule is used as a program instruction. In fact, the inference engine could take care of rule ordering since the information that is used in this process is the same for any grammar. But in implementing such a *resolution technique* one would lose the explanatory power of step by step trackings of membership tests. Experience has shown that these trackings are the most effective way of highlighting the inadequacies of the grammars, where grammars may be viewed as *pattern recognition procedures*.

Results achieved

About 20 grammars have been evolved for the successful description of a significant core of traditional tabla music, and many more are still undergoing modifications. We have shown that an analyst can construct comprehensive models of musical structure. Indeed, formal representations (and methods to improve them) have been refined to a point where, in just a few interactive sessions, experts may approve music generated by the BP, and the BP is able to distinguish a correct from an incorrect variation. Often, machine-generated variations are judged to be excellent.

As a result of these experiments, we have often been asked if a machine (indeed, our machine) might consistently reproduce the quality of composition or improvisation expected of human experts. We have observed that a number of musical effects that could be thought of as representative of high-quality composition can be analysed a posteriori in terms of, for instance, meta-patterns or polyrhythmic effects that can be made explicit with the aid of production rules. However, the grammars in which they are imbedded tend to grow beyond the limits of comprehensible statements that a human analyst may be able to cope with. This problem was encountered when trying to model music created in concerts, but not during lessons. In the latter, the models transmitted tended instead to be simple, uncontroversial, and idealised. This is for two reasons: first, traditional masters attempt to pass on sets of ‘fixed improvisations’ that function as implicit structural descriptions of the system; second, they seem to be aware that, crystallised in this form, their music will resist detailed scrutiny by competitors (see Kippen 1988c:30). The problem of analysing music that is taught rather than performed relates to a similar issue concerning the relevance of information gathered in interviews in social research, as these produce data that are too context-laden (see Phillips 1971). More recently, however, Briggs (1986) and Mishler (1986) have attempted to sort out the relationship between what is said during interviews and what happens outside this context.

Another important aspect of the human-machine interaction has been a gradual shift of the rôle played by some of the experts who, rather unconsciously, took advantage of the ability of a theoretical model to clarify their own conceptions about music. This may even become problematic as these experts tend to distort their own intuitive models in view of ‘trying something new’. This situation is not uncommon in knowledge acquisition for expert systems:

Un autre problème susceptible de se poser est le suivant: l'expert, consciemment ou non, peut se servir du cogniticien — s'il s'intéresse au processus d'extraction du savoir ou s'il a très envie de comprendre son propre savoir — pour expérimenter différents modèles de son domaine de savoir qu'il a mis au point. Cela prendra toujours beaucoup de temps, bien que cela puisse être utile; mais, en général, une expérimentation faite sans que le cogniticien ne soit au courant provoque de la confusion et gêne habituellement les progrès du travail.

(A. Hart 1986: ??? p.29 in French text, under 'Chap.4: Le rôle du cogniticien: Acquisition du savoir')

Conclusion

Is it realistic to try to determine where folk models end and where analytical reconstructions begin? And if so, then how far can experimentation with expert systems like the BP help resolve the problem? In order to answer these questions we need to examine critically two essential shortcomings of the analytical method employed in this research. First, in some cases we have been unable to investigate in detail the assumptions inherent in the hypotheses of musical structure that were used as initial representational models to trigger the interaction with informants. Consequently it became impossible to determine whether the operational model (the grammar) had reached a point of no return because the information gathered was incomplete or because we had formalised our own intuitions that were several steps ahead of what might have been expected during preliminary stages. Indeed, whatever the source, information is represented at the same (low) theoretical level, and therefore it is not possible to separate general analytical statements from specific instances of facts. We conclude that a BP grammar can be nothing other than a joint construction of the informant and the analyst. Second, whenever a computer-generated variation was rejected by an informant, no straightforward procedure for correcting the grammar was apparent. This reflects the general problem of knowledge acquisition in expert systems, and the need for automated learning procedures. The inference of probabilistic grammars has been our first attempt to solve this problem. We are now developing a learning module attached to the BP by means of which grammars are constructed in a systematic manner. Initial findings in this new field of research have been summarised in Kippen & Bel (1988a).

References cited

Bel, Bernard

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

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

'Designing tools for knowledge representation in the anthropological study of a musical system', lecture at the AI Dept., University of Edinburgh, U.K., 1988

Bernard, H. Russell et al

'The construction of primary data in cultural anthropology', *Current Anthropology* Vol.27 Part 4, 1986:382-96

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

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

Briggs, Charles L.

Learning How to Ask, Cambridge University Press, 1986

Chouraqui, Eugène

'Computational models of reasoning', in S. Torrance (ed.) *The Mind and the Machine: philosophical aspects of artificial intelligence*, Ellis Horwood, 1984:145-55

Colmerauer, Alain

'Metamorphosis Grammars', *Lecture Notes in Computer Science*, Vol.63, 1978:133-89

'Prolog in 10 figures', *5th International Joint Conference on Artificial Intelligence*, 1983:487-99

Emmet, Dorothy

'"Motivation" in Sociology and Social Anthropology', *Journal of Social Behaviour*, Vol.6 Part 1, 1976:85-104

Kippen-Bel: Can the computer help resolve the problem of ethnographic description?

Fischer, Michael D.

'Expert systems in anthropological analysis', *Bulletin of Information on Computing and Anthropology* 4, University of Kent, 1986:6-14

Hart, Anna

Knowledge acquisition for expert systems, London, Kogan Page, 1986

Hart, Johnson M.

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

Holy, Ladislav

'Description, generalization and comparison: two paradigms', in Ladislav Holy (ed.) *Comparative Anthropology*, Oxford, Blackwell, 1987

Kippen, Jim

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

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

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

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

Kippen, Jim & Bernard Bel

'The identification and modelling of a "percussion" language, and the emergence of musical concepts in a machine-learning experimental set-up', *Computers and the Humanities* 23,3, forthcoming, 1988a

'Modelling music with grammars: formal language representation in the Bol Processor', in A. Marsden & A. Pople (eds.) *Computer Representations and Models in Music*, London, Academic Press, forthcoming, 1988b

Kowalski, Robert

'Logic as a computer language in education', in L. Steels & J.A. Campbell (eds.) *Progress in Artificial Intelligence*, Ellis Horwood, 1985:71-92

Laurière, Jean Louis

Un langage déclaratif: SNARK — Symbolic Normalized Acquisition and Representation of Knowledge, Institut de Programmation, Université Paris VI, 1984

Maryanski, F.J., & 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]

Mishler, Elliot G.

Research Interviewing: Context and Narration, London, Harvard Univ. Press, 1986

Phillips, Derek

Knowledge From What?, Rand McNally, 1971

Robinson, J.A.

'The Generalised Resolution Principle', in Dale & Michie (eds.) *Machine Intelligence* 3, Edinburgh, Oliver & Boyd, 1968:77-93

Roszak, Theodore

The Cult of Information: the Folklore of Computers and the True Art of Thinking, Paladin, 1988

Salomaa, A.

Formal Languages, Academic Press, 1973

Sangren, P. Steven

'Rhetoric and the authority of ethnography', *Current Anthropology* Vol.29 Part 3, 1988:405-35

Tedlock, Dennis

'Questions concerning dialogical anthropology', *Journal of Anthropological Research* Vol.43 Part 4, 1987:325-44

Vogel, Claude

Génie cognitif, Paris, Masson, 1988

Watson, Graham

'Make me reflexive — but not yet: strategies for managing essential reflexivity in ethnographic discourse', *Journal of Anthropological Research*, Vol.43 Part 1, 1987:29-41

Appendix 1: example of qa'ida

The following sentences have been generated by the BP and represent a few variations that our informants consider 'correct', although some may be aesthetically more pleasing than others, ranging from 'very good' (sentence 5) to 'very ugly' (sentence 6). Sentence 1 may be termed the *theme* of the qa'ida as it is conventionally used to introduce the improvised sequence. Sentence [3] may be considered ambiguous as it fits two hypothetical structures (*templates*, see appendix 2).

[1]	dhadhatite dhadhatite tatatite dhadhatite	dhadhadheena dhadhadheena tatateena dhadhadheena	dhadhatite dhadhatite tatatite dhadhatite	dhadhadheena dhadhateena tatateena dhadhadheena
[2]	dhadhadhadha dhadhatite tatatata dhadhatite	teenatite dhadhadheena teenatite dhadhadheena	dhadhatite dhadhatite tatatite dhadhatite	dhadhateena dhadhateena tatateena dhadhadheena
[3]	dhadhateena dhadhatite tatateena dhadhatite	dheenadhadha dhadhadheena teenatata dhadhadheena	titetite dhadhatite titetite dhadhatite	teenateena dhadhateena teenateena dhadhadheena
[4]	dhadhatite titetite tatatite titetite	dhadhadheena titetite tatateena titetite	dhadhatite dhadhatite tatatite dhadhatite	dhadhateena dhadhateena tatateena dhadhadheena
[5]	dhadhatite dhadhatite tatatite dhadhatite	dhadhadheena dhadhadheena tatateena dhadhadheena	dheena-dhee dhadhatite teena-tee dhadhatite	nadhateena dhadhateena natateena dhadhadheena
[6]	dhadhadhadha dhadhatite tatatata dhadhatite	dhatitedha dhadhadheena tatiteta dhadhadheena	dhateenadha dhadhatite tateenata dhadhatite	dhadhadha- dhadhateena tatata- dhadhadheena

Appendix 2: a BP grammar for this qa'ida

GRAM#1 [1]	RND				
GRAM#1 [2]	<100>	S → S64			
GRAM#1 [3]	<100>	S64 → S1F S2F S1V S2F E32			
GRAM#1 [4]	<100>	S64 → S1V S2F S1F S2F E32			
GRAM#1 [5]	<100>	S64 → S1F S2V S1F S2F E32			
GRAM#1 [6]	<100>	S64 → S1V S2V S1F S2F E32			
GRAM#2 [1]	RND				
GRAM#2 [2]	<100>	S1V S2F E32 → (= V8) S2F * (= S1F S2F) (: V8) S1F			
GRAM#2 [3]	<100>	S1V S2F S1F S2F E32 → (==+ A1 V7) S2F S1F S2F * (:++ A1 V7) * (=S2F) S1F S1F			
GRAM#2 [4]	<100>	S1V S2F S1F S2F E32 → (==+ A2 V6) S2F S1F S2F * (:++ A2 V6) * (= S2F) S1F S1F			
GRAM#2 [5]	<100>	S1F S2V S1F S2F E32 → S1F (= V6 + B2) S1F S2F * (= S1F) * (: V6 + B2) S1F S1F			
GRAM#2 [6]	<100>	S1V S2V S1F S2F E32 → (==+ A1 V11 + B4) S1F S2F * (:++ A1 V11 + B4) S1F S1F			
GRAM#2 [7]	<100>	S1V S2V S1F S2F E32 → (==+ A2 V10 + B4) S1F S2F * (:++ A2 V10 + B4) S1F S1F			
GRAM#2 [8]	<100>	S1V S2V S1F S2F E32 → (==+ A1 V13 + B2) S1F S2F * (:++ A1 V13 + B2) S1F S1F			
GRAM#2 [9]	<100>	S1V S2V S1F S2F E32 → (==+ A2 V12 + B2) S1F S2F * (:++ A2 V12 + B2) S1F S1F			
GRAM#2 [10]	<100>	S1V S2V S1F S2F E32 → (==+ A1 V15) S1F S2F * (:++ A1 V15) S1F S1F			
GRAM#2 [11]	<100>	S1V S2V S1F S2F E32 → (==+ A2 V14) S1F S2F * (:++ A2 V14) S1F S1F			
GRAM#3 [1]	LIN [Right-linear grammar]				
GRAM#3 [2]	<10>	V15 → T1 V14	GRAM#3 [16]	<10>	V8 → T1 V7
GRAM#3 [3]	<100>	V15 → T2 V13	GRAM#3 [17]	<100>	V8 → T2 V6
GRAM#3 [4]	<10>	V14 → T1 V13	GRAM#3 [18]	<10>	V7 → T1 V6
GRAM#3 [5]	<100>	V14 → T2 V12	GRAM#3 [19]	<100>	V7 → T2 V5
GRAM#3 [6]	<10>	V13 → T1 V12	GRAM#3 [20]	<10>	V6 → T1 V5
GRAM#3 [7]	<100>	V13 → T2 V11	GRAM#3 [21]	<100>	V6 → T2 V4
GRAM#3 [8]	<10>	V12 → T1 V11	GRAM#3 [22]	<10>	V5 → T1 V4
GRAM#3 [9]	<100>	V12 → T2 V10	GRAM#3 [23]	<100>	V5 → T2 V3
GRAM#3 [10]	<10>	V11 → T1 V10	GRAM#3 [24]	<10>	V4 → T1 V3
GRAM#3 [11]	<100>	V11 → T2 V9	GRAM#3 [25]	<100>	V4 → T2 V2
GRAM#3 [12]	<10>	V10 → T1 V9	GRAM#3 [26]	<10>	V3 → T1 V2
GRAM#3 [13]	<100>	V10 → T2 V8	GRAM#3 [27]	<100>	V3 → T2 T1
GRAM#3 [14]	<10>	V9 → T1 V8	GRAM#3 [28]	<10>	V2 → T1 T1
GRAM#3 [15]	<100>	V9 → T2 V7	GRAM#3 [29]	<100>	V2 → T2
GRAM#4 [1]	LIN				
GRAM#4 [2]	<100>	T1 → dha			
GRAM#4 [3]	<100>	T1 → -			
GRAM#4 [4]	<100>	T2 → dhadha			
GRAM#4 [5]	<100>	T2 → tite			
GRAM#4 [6]	<50>	T2 → dheena			
GRAM#4 [7]	<50>	T2 → teena			
GRAM#5 [1]	RND				
GRAM#5 [2]	<100>	+ B2 → +teena			
GRAM#5 [3]	<100>	+ B4 → +dhadhateena			
GRAM#5 [4]	<100>	++ A1 → ++dha			
GRAM#5 [5]	<100>	++ A2 → ++dhadha			
GRAM#5 [6]	<50>	++ A2 → ++dheena			
GRAM#5 [7]	<100>	S1F → dhadhatitedhadhadheena			
GRAM#5 [8]	<100>	S2F → dhadhatitedhadhateena			

The following are the *templates* generated by this grammar, i.e. a list of possible structures where each dot is a 'slot' into which a terminal symbol may fit.

- [1] (=.....) * (=.....) (:.....)
- [2] (==+.....) * (:++.....) * (=.....)
- [3] (=.....+..) * (=.....) * (:.....+..)
- [4] (==+.....+...) * (:++.....+...)
- [5] (==+.....+..) * (:++.....+..)
- [6] (==+.....) * (:++.....)

Template #3:

dhadhatitedhadhadheena (=dheena-dheenadha+teena) dhadhatitedhadhadheenadhadhatitedhadhateena *
(=tatatitotateena) * (:teena-teenata+teena) dhadhatitedhadhadheenadhadhatitedhadhadheena
dhadhatitedhadhadheena (=dheena-dheenadha+teena) dhadhatitedhadhadheena S2F * (=tatatitotateena) *
(:teena-teenata+teena) dhadhatitedhadhadheenadhadhatitedhadhadheena
dhadhatitedhadhadheena (=dheena-dheenadha+teena) dhadhatitedhadhadheena S2F * (=tatatitotateena) *
(:teena-teenata+teena) dhadhatitedhadhadheena S1F
dhadhatitedhadhadheena (=dheena-dheenadha+teena) dhadhatitedhadhadheena S2F * (=tatatitotateena) *
(:teena-teenata+teena) S1F S1F
dhadhatitedhadhadheena (=dheena-dheenadha+teena) dhadhatitedhadhadheena S2F * (= S1F) * (:teena-
teenata+teena) S1F S1F
dhadhatitedhadhadheena (=dheena-dheenadha+teena) S1F S2F * (= S1F) * (:teena-teenata+teena) S1F S1F
S1F (=dheena-dheenadha+teena) S1F S2F * (= S1F) * (:teena-teenata+teena) S1F S1F
S1F (=dheena-dheenadha+ B2) S1F S2F * (= S1F) * (:teena-teenata+ B2) S1F S1F
S1F (=dheena-dheena T1 + B2) S1F S2F * (= S1F) * (:teena-teena T1 + B2) S1F S1F
S1F (=dheena- T2 T1 + B2) S1F S2F * (= S1F) * (:teena- T2 T1 + B2) S1F S1F
S1F (=dheena T1 T2 T1 + B2) S1F S2F * (= S1F) * (:teena T1 T2 T1 + B2) S1F S1F
S1F (= T2 T1 T2 T1 + B2) S1F S2F * (= S1F) * (: T2 T1 T2 T1 + B2) S1F S1F
S1F (= T2 T1 V3 + B2) S1F S2F * (= S1F) * (: T2 T1 V3 + B2) S1F S1F
S1F (= T2 V4 + B2) S1F S2F * (= S1F) * (: T2 V4 + B2) S1F S1F
S1F (= V6 + B2) S1F S2F * (= S1F) * (: V6 + B2) S1F S1F
S1F S2V S1F S2F E32
S64
S

Successful. Other templates are then tried but none produces a structure that results in a successful test.