A MUSICO-LINGUISTIC COMPOSITION ENVIRONMENT

Reginald Bain
Northwestern Computer Music
School of Music, Northwestern University
Evanston, IL 60201 USA
(312) 491-3895/5431
ihnp4!numusic!reg

ABSTRACT

This paper presents work being done at Northwestern University developing compositional tools that perform transformational processes on text which generate music having parallel syntactic, semantic, and eventually prosodic structures. The generation of the initial musical gesture and subsequent motivic cells is derived from computational linguistic analysis of the text. The resulting music is then subjected to the composer's aesthetic evaluation and integrated into the final composition.

1. ISSUES AND CONCERNS

The genesis of my interest in the logical structures of natural language, and its eventual mapping into a musical experience, originated from an attempt to set Wittgenstein's *Tractatus Logico-Philosophicus* for orchestra and chorus. The *Tractatus* is a philosophical text concerned with the principles of symbolism and the conditions that must be met by a logically perfect language. Confronted with an aphoristic structure and the relations of predicate calculus in the text, it seemed evident that a musico-dramatic presentation of the text should include algorithmic transformations of natural language and symbolic logic to music. Instead of pursuing universal representations of natural language, one can view the relationship of music to language as primarily a metaphorical one (Cope 1987) and thus use the computer simply as a means of creating an inexorable link between text and music. The primary objective of my work has been to use personal representations of linguistics and music which in turn create "interesting" logical structures.

2. A CONTEXT-FREE GRAMMAR FOR MUSIC COMPOSITION

Three things are needed in order to perform the syntactic transformation from linguistic structures into a music composition: 1) compositional tools for music analysis and transformation 2) a rules base or grammar to guide and select operations to be performed on the musical domain 3) an augmented transition network (ATN) for parsing and storing the linguistic structures of the text. Also needed is a description of propositional logic and predicate logic to provide a situational semantic context for the eventual musical mapping.

The Object Logo music environment for the Macintosh (Greenberg 1987, Krakowsky 1986) provides the basis for my compositional tools, and my linguistic analysis and transformation routines. The use of a symbolic language such as LOGO or LISP provides the modularity, recursive structure, and interactive environment needed to realize this grammar. Built-in MIDI drivers, MacWrite/Paint editing features, objects for creating windows and menus, and full support for input/output streams between windows are just a few of the resourses of this Object LOGO environment. This environment will eventually run on a Macintosh II with 8Mb.

2.1 Transformational Tools

The first thing needed is a set of interactive compositional tools for music analysis and transformation. All of the transformational composition tools have been developed according to the definitions and theorems set forth in John Rahn's, *Basic Atonal Theory* (1980). This model was chosen because the transformations and pitch organization of atonal and serial music are most congruent with my compositional style. The analytical music tools analyze intervalic content. For example, one could determine the possible common-tone structures, or examen set relationships such as z-relations, symmetry, or subset content. The transformational music tools are needed to map musical structure onto musical materials. These transformations include operations like transposition, inversion, retrograde, retrograde inversion, as well as multiplicative interval operations (e.g. the M5(x) circle of fourths transformation). Two examples of compositional tools are presented in **figure #1. Figure #1a** shows the predicate function **z_relatedp** which tests to see if two pitch class sets have the same **interval_vector. Figure**

#1b shows the procedure TICS_vector which outputs a 12-tuple vector representing the TnI-common-tone structure of some set A. It enumerates the number of pitch classes in common between A and TnI(A) for each of the 12 indices n = 0,11.

Figure #1a: Examples of Compositional Tools

```
to z_relatedp :pc.set1 :pc.set2
  if equalp (interval_vector :pc.set1) (interval_vector :pc.set2) [output "TRUE]
  output "FALSE
end
```

Figure #1b

```
to TICS_vector :pc.set
  output (add_vectors (multipy_vector_by_scalar (M+AN* :pc.set) 2)(Q+AN* :pc.set))
 end
```

*<M+AN denotes M+A(n), the multiplicity of the sum n of pairs of pitch classes from set A, and Q+AN denotes Q+A(n) the number of pitch classes in a set A which add to themselves to make the index n. For more information see Rahn 1980, *Basic Atonal Theory* p. 111>

2.2 Rules Base

In order to create music, the compositional tools need a rules base or grammar to guide and select operations to be performed on the musical material. The rules base is essentially a meta-grammar (Hemke 1988) composed of algorithms for generating musical material that parallels the syntactic and semantic nature of the text. The rules base also invokes computational linguistic analysis of the text to generate both the initial musical gesture and subsequent musical generations. The results of the computational analysis and generation of musical material can be archived by the composer for further reference by the rules base.

Basic computational linguistics tools can calculate the probability of a letter or word occurring, the average length of a word, the relative position of a letter or word, or the contextual significance of the reiteration of a word. Comparisons are made by the predicate **equal_phrasep** and other linguistic analysis predicates. Two simple examples follow. **Figure #2a** shows the procedure **start_with_phrase** which outputs the phrase argument and whatever follows it in the sentence. **Figure #2b** shows the procedure **get_descriptions** which outputs a list of all phrases in which the argument *:key.phrase* appears. The argument *:text* is a list of any complete section of the text. The procedure **after_phrase** outputs whatever follows the given *:key.phrase*.

Figure #2a: Examples of Linguistic Context Primitives

```
to start_with_phrase :phrase :sentence
    if lessp (count :sentence) (count :phrase) [output "FALSE]
      ifelse equalp (first :sentence) (first :phrase)
_             [if (start_phrasep :phrase :sentence) [output :sentence]
                                  [output start_with_phrase :phrase (bf :sentence)]
_
   output (start_with_phrase :phrase (bf :sentence))
end
```

Figure #2b

```
to get_descriptions :key.phrase :text
    if (emptyp :text) [output []]
    output fput (after_phrase :key.phrase (first :text))
_             (get_descriptions :key.phrase (bf :text))
end
```

<Propositional contexts can also be directly tested for their musical mapping potential without the aid of a computer interface by using concordances (Plochmann 1962), indexes, or glossaries which often accompany philosophical texts and mapping them to music directly. This has proven to be quite useful since parsing the text can fill the object list very quickly. Saving computation time in the early stages of developing linguistic-music transformations lets one concentrate more on the compositional issues at hand.>

2.21 Initial Linguistic-Music Mapping

One of the most interesting applications of this musico-linguistic environment is its ability to explore many different possible mappings from language to music. From the most basic transformation to one that is deeply nested, it is possible to create musical gestures for immediate auditory analysis by the composer. For the sake of brevity, the concerns here will be limited to pitch mappings, even though the program analagously extapoltates this process to the domains of duration and amplitude. At first, a one to one mapping from letter to note might seem to be a somewhat obvious choice. However, this approach to musical realization of a text can create very interesting possibilities. For example, if we used Messiaen's communicable language letter to note mapping from his *Meditationen über das Geheimnis der heiligen Dreieinigkeit* für Orgel (see **figure #3a**, middle C = C(0)) and apply it to Wittgenstein's *Tractatus* we can see the results. The procedure **map_pitch_to_integer** (see **figure #3b**) takes a given mapping of pitches to integers and performs the necessary binding of variables.

Figure #3a: Letter to Note Mapping using Messiaen's Communicable Language

A	B	C	D	E	F	G	H	I	J	K	L	M	N
A(0)	Bb(0)	C(1)	D(1)	E(1)	F(1)	G(1)	B(0)	F#(2)	F#(1)	C(-1)	Eb(1)	Ab(1)	Eb(0)
O	P	Q	R	S	T	U		V	W	X	Y	Z	
B(1)	G(-1)	C(0)	E(2)	F(2)	D(-2)	C#(-1)		D(0)	D(2)	G#(0)	F#(0)	F(-1)	

<The "musical alphabet" is derived as follows. 8 basic letters [a, b, c, d, e, f, g, h], palatals [i, j, y], 5 vowels [a, e, i, o, u], sibilants [s, z], dentals [d, t], K [c q k], labials [b, f, m, p, v], linguals [l,n], and [r, w, x]. All articles, pronouns, prepositions, and adverbs are eliminated in order to avoid verbal agglomerations. Messiaen employs the case system of Latin declensions providing each word with a musical formula or "case". The verbs *to be* and *to have* have been assigned a descending and ascending melodic formula respectively. The durations are Messiaen's.>

Musical Figure #1: A setting of 'The world is everything.'

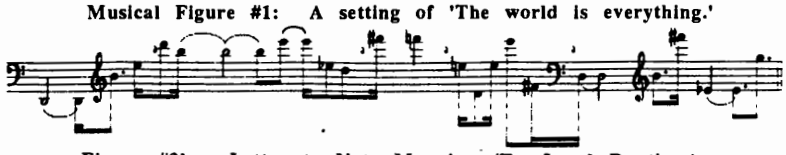


Figure #3b: Letter to Note Mapping (Top-Level Routines)

```
map_pitch_to_interval :mapping  ; assign pitch names integer values
localmake "text (get_proposition "i "i) ; temporary variable text for section i, proposition i

play [(eval_list_characters (explode :text)) ; plays a list of pitch integers with a
_                                            [:durations] ; list of durations and a
_                                            [:amplitudes]] ; list of amplitudes
```

2.3 Augmented Transition Network (ATN)

A recursive transition network with LOGO procedures to enforce grammatical constraints and generate a deep structure is called an augmented transition network. A transition network can be described as a representation of a finite grammar. The LOGO procedures that enforce the grammatical constraints are the predicate compositional tools and the propositional and predicate logic functions described in section 3.0. Conditions are written as LOGO predicates which perform tests for certain attributes in the definition of the current word or object. A top-level example of the method of storing a transformation can be seen in **Figure #4** using 'Verlaine shot Rimbaud.' as the sample text and a very basic ATN parser.

Figure #4: Example of Transformation Data Storage

```
[S DCL [NP Verlaine]                    S- sentence, DCL- type=declaritive
       [TNS past]                        NP- noun phrase, TNS- tense, VP- verb phrase
       [VP [V shoot] [NP Rimbaud]]       V -verb
```

<For a more comprehensive definition and graphic analysis of ATN's see Charniak 1984, pp. 195-223.>

One could envision a transformation not unlike Messiaen's communicable language that could convert the ATN's resulting data to a musical representation. For example, one could use a mapping of lexical terminals (e.g. verb,

noun, etc.) directly to music using intuitive musical resultants such as verb -> motion or noun -> stasis. But substituting lexical types or even syntactic transformations with intuitively analagous transformations of "musical strings" (Bernstein 1976) is not the main issue here. What is of immediate interest to our linguistic-music transformation is the meta-grammar's specific implementation. The meta-grammer is used to generate logical musical structures from the situational semantics of a given aphorism in relation to its main proposition. Wittgenstein's *Tractatus* was conceived as a set of propositions with a logical importance explicitly indicated by the author's numbering system. From the context provided, we can extract semantic relations using propositional logic and predicate logic to serve as conditions for our rules base.

2.4 Semantic Analysis

A brief introduction to the aspects of propositional logic and predicate logic which hold meaning in this program's implementation follows.

2.41 Propositional Logic

The simplest type of symbolic logic, propositional logic, can categorize arguments such as:

given: If it is Sunday Paul will play the piano. & It is Sunday.
conclusion: Paul is playing the piano.

It is apparent that as long as we preserve the if ... then ... connective in what is given and retain the general structure of the argument one will arrive at a valid argument. A proposition then is a sentence which makes a declaration and evaluates to either TRUE or FALSE. Thus, in symbolic notation the following (*modus ponens* rule) is a valid inference:

p and q are propositions:			**other operators can be defined:**	
given:	p -> q	; if p then q	p v q	; p or q
	p	; p occurs	p ^ q	; p and q
conclusion:	q	; q results	~p	; not p

The following rules of propositional logic can generate all possible formulas:
1. A proposition is a formula.
2. If X is a formula, [~X] is a formula.
3. If X and Y are formulas, [X ^ Y], [X v Y], and [X -> Y] are formulas.
4. All formulas are generated by applying these rules.

2.42 Predicate Logic

Some of the meanings of English connectives have yet to be accounted for by the rules of propositional logic. For example, propositional logic doesn't delineate any differentiation between strict implication and material implication, where there is a causal relationship between X and Y [if X then Y]. Since each line is a separate proposition we cannot represent it with propositional logic. Let's call Beethoven the object and say that the proposition 'Beethoven is a man' asserts the predicate 'is a man' of the object Beethoven and that 'Beethoven is mortal' asserts the predicate 'is mortal' about the exact same object. Let us call Beethoven a **term**, which is to say that its value is some object in the universe. **Predicates** can then be described as functions which operate on terms and evaluate to the values TRUE and FALSE. We can then represent 'Beethoven is mortal' with a function $f(x)$ where 'is mortal' is the function f and x is the object.

given:	All men are mortal.	Beethoven is a man.
	M(Beethoven) ; 'is mortal' (Beethoven)	
conclusion:	Beethoven is mortal.	

In the above example M is the predicate, Beethoven is the argument, and M(Beethoven) is called an atom. Retention of the logical connectives of propositional logic and the introduction of a universal quantifier to account for propositions such as 'All men are mortal' will complete our definition of predicate logic. Assign x to represent the object of one or more predicates [Beethoven], then e(x) represents a predicate logic expression. Thus we can rewrite the original propositions as:

given: (Vx) P(x) -> M(x) ; for all x, if x is a man then x is mortal,

P(Beethoven) ; where 'P' represents 'is a man'
conclusion: M(Beethoven)

(A definition of a formula for predicate logic can be found in any textbook on symbolic logic. The inclusion of restricted quantification and semantic nets, etc. are left out for the sake of brevity.)

3. THE LINGUISTIC-MUSIC TRANSFORMATION

3.1 Tractatus Logico-Philosophicus

The opening of *Tractatus Logico-Philosophicus* appears in **figure #5.** The English translation by C. K. Ogden follows the original German text. Decimal figures appear to the left of each proposition. According to Wittgenstein, they indicate the logical importance of the propositions in his exposition. Thus, proposition x.1, x.2, x.3 , etc. comment upon proposition No. x and x.y1, x.y2, etc., are comments on proposition No. x.y. Here we are given a macro-level semantic context that can be used as a musical form generating element. Given the constructs of propositional logic and the logical importance assigned to each proposition by the Wittgenstein, one can construct musical transformations that generate elements paralleling the situational semantics of a particular set of related propositions. **Figure #6** shows the similaritiy between the context-free representation and a grammatical representation of the logical importance of propositions.

Figure #5

1 Die Welt ist alles, was der Fall ist.
1.1 Die Welt ist die Gesamtheit der Tatsachen, nicht der Dinge.
1.11 Die Welt ist durch die Tatsachen bestimmt und dadurch, dass es alle Tatsachen sind.
1.12 Denn, die Gesamtheit der Tatsachen bestimmt, was der Fall ist und auch, was alles nicht der Fall ist.
1.13 Die Tatsachen im logischen Raum sind die Welt.
1.2 Die Welt zerfällt in Tatsachen.

1 The world is everything that is the case.
1.1 The world is the totality of facts, not of things.
1.11 The word is determined by the facts, and by these being *all* the facts.
1.12 For the totality of facts determines both what is the case, and also all that is not the case.
1.13 The facts in logical space are the world.
1.2 The world divides into facts.

Figure #6: Grammar Representation Comparison

a natural language representation (ATN)		the logical importance of propositions
so -> $ sent z		1 -> 1.1 | 1.2
sent -> sub vp		1.1 -> 1.11 | 1.12 | 1.13
vp -> verb obj		1.2 -> 1.21
z -> $	($=blank, z=terminal state)	
obj -> det noun | det adj noun		

terminal lexical categories: det, noun, adj, verb 1.11, 1.12, 1.13, 1.21

<This hierarchical structure is used to determine the inheritance relationships of gesture objects. Thus, for the current environment a gesture object is created for each proposition which holds the text, its inital mapping to music, and the procedure names and results of any transformations performed. The gesture objects can thus communicate via window objects (where data is) to logically related propositions comparing information about the transformations of musical ideas that have been performed in the piece.>

3.2 Interaction of the Compositional Tools and Rules Base

Music generation is controlled by the rules base construction in **figure #7.** The LOGO construction for dynamic execution of predicate functions Pkn is **invoke :state :inputs** where *:state* (a node in the transformation network) is one of the transformational composition tools and *:inputs* is the integer/pitch data to be transformed. The predicate functions are taken from either truth functions in the *Tractatus* text or the propositional/predicate logic equivalent of the propositions. If some Pkn evaluates to TRUE then it is stored in the most recent gesture object along with Tkn so that any part of the musical material generated can be accessed.

Figure #7: Rules Base Construction

```
[PT1 [key.name [T1 T2 ... Tn]
       T1 [T11 P11 T12  P12 ... T1n  P1n]
       T2 [T21 P21 T22  P22 ... T2n  P2n]
       Tk [Tk1 Pk1  Tk2   Pk2 ... Tkn  Pkn]]
[PTn [etc...
```

<PTn - a compound musical operation (e.g. a sequence of transpositions, inversions, etc.)
key.name - name of gesture object to be transformed, Tn - a transformational composition tool, Pkn - an analytical composition predicate.>

4.0 MUSICAL REALIZATION

To summarize, taking proposition 1.2 'The world divides into facts' and running it through the linguistic-music transform yields the following results.

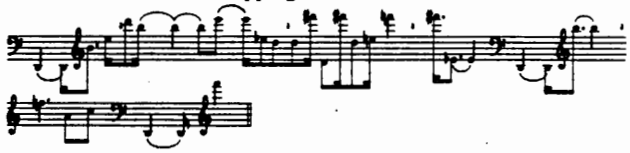
4.1 Flow of Control'

1) Initial Mapping [2.21].
2) ATN parser - stores syntactic representation and propositional/predicate logic representations [2.3].
3) Meta-Grammar - Invokes rules base and performs comparisons between representations stored by the ATN.
4) Rules Base - creates musical gestures which are stored as gesture objects by invoking the compositional tool operations according to the conditional scheme set forth in each compound compositional operation [3.2].
5) Score production and Realization using MIDI (Greenberg 1988).

4.11 Initial Mapping

Using the same letter to note mapping described in [2.21], integer values are assigned to each letter and all pitch characters are declared globally fluid in the environment.

Musical Figure #2: A resultant mapping of 'The world divides into facts.'



4.12 ATN parser

The ATN parses the proposition accordingly:

syntactic description	propositional relation
[S [DCL [NP The World]	[The world -> divides into facts]
[TNS Pres]	
[VP [V divides] [OBJ into facts]	

4.13 Meta-grammar

When we are dealing with an initial gesture, the meta-grammar creates the necessary parent objects according to the logical importance of the propositions. For 1.2 the only parent object is proposition 1 (see figure #8a). The meta-grammar looks for the values of any variables first on the object list, then in all parent objects where the linguistic-music transformations have been performed as stored (see 2.21). Flow control is then sent to the rules base.

4.14 Rules Base

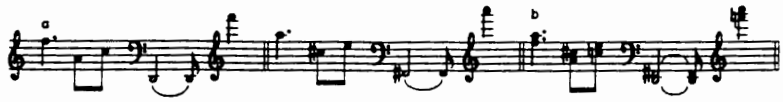
The rules base is then invoked with key.name 1.2 (see 3.2). Many different invocations are possible. As an simple example, say that [V divides] (see 4.13) evaluates to TRUE for some predicate Pnk in the rules base (see 3.2)

and this invokes the compositional tool **transpose** with the argument [OBJ into facts]. As a result, the pitch-integer representation of [into facts] is transposed by the pitch-class-integer representation for [divide]. If **t** is the composition tool for pitch class transposition the result would be:

(t(t(t(t(t(t(t (eval_list (explode "facts)) -3) 1) -3) 1) -3) -1) 12)

The result (transposition by 12 - 10 + 2 = 4 semitones) can be seen below in musical example #3a. Finally the musical result of this transformation is combined with the previous musical generations using "contrapuntal syntax" (Bernstein 1976) to determine how the new material should relate to the former material harmonically. For example, if computational linguistic analysis results indicate that it should sound in homophony with the previous generation, the result can be seen in ex. #3b. This dialogue between the rules base and meta-grammar continues until all of the text has been set to music.

Musical Figure #3a & b
A pitch representation of [into facts] transformed by [divide] resulting in 3b.



5.0 FUTURE DIRECTIONS

Though full automation is probably neither feasible nor desirable, an implementation that utilizes a set of exclusive restrictions as a dictate for style would be useful in the task of automation. Using predicate calculus and logically correct inference (Charniak 1984), it is intended that certain heuristics can be set up to mimic one's compositional process that are more sophisticated than those in the present implementation.

REFERENCES

Bernstein, Leonard (1976). *The Unanswered Question: Six Talks at Harvard.* Cambridge, MA: Harvard University Press.

Charniak, Eugene, and Drew McDermott (1984). *Introduction to Artificial Intelligence.* Reading, MA: Addison-Wesley Publishing Company.

Chomsky, Noam (1957). *Syntactic Structures.* The Hague: Mouton.

Cope, David (1987). "Experiments in Music Intelligence". In J. Beauchamp ed., *Proceedings of the 1987 International Computer Music Conference.* San Francisco: Computer Music Association.

Greenberg, Gary (1987). "Procedural Composition". In J. Beauchamp ed., *Proceedings of the 1987 International Computer Music Conference.* San Francisco: Computer Music Association.

Greenberg, Gary (1988). "Composing with Performer Objects". In C. Lischka ed., *Proceedings of the 1988 International Computer Music Conference.* San Francisco: Computer Music Association.

Hemke, Frederick L., and Gilbert K. Krulee (1987). *Patterns in Music: A Formal Representation..* Northwestern University, Evanston, IL.

Krakowsky, Philippe, ed. (1986). *Object LOGO Reference Manual.* Cambridge, MA: Coral Software Corporation.

Plochmann, George Kimball, and Jack B. Lawson (1962). *Terms in Their Propositional Contexts in Wittgenstein's Tractatus.* Southern Illinois University Press.

Rahn, John (1980). *Basic Atonal Theory.* New York, NY: Longman Inc.

Wittgenstein, Ludwig (1947). *Tractatus Logico-Philosophicus.* C.K. Ogden, ed. New York, NY: Harcourt, Brace & Company.