GENERATIVE GRAMMARS FOR INTERACTIVE COMPOSITION BASED ON SCHAEFFER'S TARTYP

Israel Neuman
University of Iowa
Department of Computer Science

ABSTRACT

Noam Chomsky's Phrase Structure (PS) grammars and the [∑, F] form of rewrite rules are an efficient analytical tool for complex musical structures as well as a generative tool for classification-based compositional processes. Pierre Schaeffer's summary table of sound typology, the TARTYP, is a milestone in the evolution of contemporary approaches to the organization of musical material. In this paper, we propose a compositional method that combines the TARTYP classification of sound objects with generative grammars derived from this table. These grammars enable the creation of musical structures that reflect the inter-relationships suggested by the table's structure. The tools presented in this paper are designed for real-time compositions in interactive environments. They are embedded in Max/MSP or Pure Data as extensions of the MaxObject class and directly engage the sound processing capabilities of these environments. The complex musical structures generated by these tools are brought to life at the surface of the composition in a versatile way that uses the spectral signatures of sound objects from Schaeffer's sound examples.

1. INTRODUCTION

In 1957 Noam Chomsky introduced Phrase Structure (PS) grammars, a form of generation systems denoted [∑, F], where ∑ is a set of initial symbols and F is a set of rewrite rules. Such rewrite rules have proven to be a suitable tool for musical analysis and composition. A phrase in music is often expanded or contracted to define the different hierarchical levels of musical structures. Hence, if a set of rewrite rules defines and produces a "legal" musical phrase it can also produce more complex musical structures. Holtzman [11] demonstrates the use of the GGDL grammars to generate descriptions of the micro components of musical objects as well as complete sections of a composition. Lerdahl and Jackendoff [17] compare their use of PS grammars for defining hierarchical structures of temporal organization with the process known in Schenkerian theory as prolongation and reduction.

Both Chomsky's theory and contemporary music have strong ties to set theory. Chomsky was interested in classes of derivations and sets of rules that would generate the same terminal language [7] [15]. Yet PS grammars are effective in describing the musical language because they can be used to create multiple legal variants of the same sentence. Consider Chomsky's example "the man hit the ball" [7]. In the grammar describing this sentence the terminals are the words (the, man, hit, the, ball): variants of this sentence can be formed, e.g., by replacing nouns with placeholder variables, yielding "the X hit the Y", where X = [man | woman | boy] and Y = [ball | table | wall]. By replacing each terminal by typed variables denoting classes of words, the same set of rewrite rules would produce legal variants of the original sentence.

The recent focus on classifications in musical research corresponds to the introduction of musical set theory where terms such as "pitch class" and "interval class" are commonly used. Serialism expands the use of classifications to compositional elements other then pitch, such as rhythm, dynamics, articulation, orchestration and timbre. A well-formed classification method of compositional material, however, is not always sufficient for defining ways for composing out this material. Pierre Boulez's pitch-class sets multiplication system produces domains or collections of pitch-class sets which are structurally tied with the twelve-tone series from which they originate. While it is known that Boulez used this classification in *Le marteau sans maître* (1955), the order by which the collections are being used in the composition is a matter of some debate among researchers [10, 12].

Pierre Schaeffer introduced the TARTYP (*Tableau Récapitulatif de la Typologie*) in [23] as part of his typology of sound objects. The table is a classification of sound objects based on their properties in the time and frequency domains, and it introduces an alphanumeric notation for sound objects. Its structure alludes to inter-relationships between sub-collections of sound objects. Nevertheless, Schaeffer provides limited direction for how to use the classification in a compositional process, focusing mainly on using the table's notation to construct sequences of symbols that describe more complex sounds [28].

In this paper, we propose a compositional method that combines the TARTYP classification with generative grammars. The tools presented in this paper are intended for use in interactive environments and are designed for real-time composition. The grammars are derived from the TARTYP and enable the creation of hierarchical musical structures that reflect the hierarchical structure of the table. These complex structures are brought to life at the surface of the composition in a versatile way enabled by the spectral signatures of sound objects from Schaeffer's sound examples.

2. RELATED WORK

Generative grammars have been used in musical research to both define the musical language and as a tool for analysis and re-composition. Laske's KEITH [13] is a rule-based system for comprehensive analysis of different musical perspectives (composer , performer and listener) . In [14], Laske writes:

> "The idea of generative grammar for music is the outcome of research geared toward the formulation of a system, or a set of rules, capable of rewriting the sequence of mental representations (of sound structures) that are assumed to underlie the execution of activities called musical."

A similar attempt to formalize the music language with the use of generative grammars can be found in Lerdahl and Jackendoff [16], as well as work by others [22,25]. Analysis studies often present examples of the use of generative grammar and rewrite rules in the re-composition of a particular repertoire for the purpose of analysis (see, e.g., [1,5,9,19,20,21,25,26,27]).

This paper also draws from Schaefferian theory and, in particular, the TARTYP. Most studies of the TARTYP offer a translation, adaptation or revision of this sound-object classification. Thoresen [28], in his adaptation, removes some of the elements defining the time domain axis of the table. Normandeau [18] in his revision to the table interprets the great dichotomies of the table: Mass-Facture, Duration-Variation and Balance-Originality. Dack [8] discusses the "excentric" sound-objects, a group of sound objects that were defined by Schaeffer as unsuitable for music yet according to Dack these object are commonly used in electroacoustic compositions. A departure point for many of these studies is Michel Chion's *Guide To Sound Objects. Pierre Schaeffer and Musical Research* [6]. This is a lexical collection of many terms in Schaefferian theory including the terms defining the TARTYP.

The work of Bernard Bel [2, 3, 4] combines the Schaefferian approach to sound objects and generative grammars in a creative environment that he calls the Bol Processor. This environment supports composition and improvisation using a system of rewrite rules. The grammars of the Bol Processor are derived from the metaphoric language for drumming in Asia and Africa called Qa'idas. The focus here is on the mapping of the objects to a structural organization in physical time, hence, the consideration is of the time domain properties of sound objects (i.e., it does not take into account classifications and defining features). The output of the Bol processor is a list of MIDI messages or a CSound score. While having some real-time capabilities, the Bol processor requires a communication with an external sound processor.

The tools presented in this paper utilize existing interactive environments: as extensions of the MaxObject class, these tools are embedded in Max/MSP or Pure Data and directly engage the sound processing capabilities of these environments. The grammars of these tools are derived directly from Schaeffer's classification of sound objects as presented in the TARTYP and as interpreted by Michel Chion [6] and Robert Normandeau [18]. These grammars as well as the compositional process suggested in this paper take into account the characteristics of sound objects both in the time and frequency domain as presented in the TARTYP table and demonstrated by Schaeffer's sound examples [24].

3. TARTYP SOUND-OBJECT CLASSIFICATION

Sound-object characteristics are specified in the TARTYP at the margins of the table, with time domain characteristics along the upper row of the table and the frequency domain characteristics along the leftmost column (see Figure 1). The frequency domain terms that correspond to rows in the table describe the frequency content of sound objects by their variability on a scale between fixed sound (mass) and unpredictable noise. Hence, *fixed mass* can be of *definite pitch* (harmonic sound) or *complex pitch* with some inharmonicity and/or noise. In addition, the *not very variable mass* is glissando like and the *unpredictable variation of mass* is non-periodic noise sound. Variation in this table refers to internal variation in the frequency domain, i.e., sounds such that their ending differs from their beginning [6].

The seven terms that describe the time domain characteristics correspond to columns in the table. At the center of this row appears the term *impulse,* referring to a very short sound. To the right of *impulse* appear terms describing the characteristics or gain envelopes of *iterative sounds* including *formed iteration*, (iterative) *non-existing facture* and (iterative) *unpredictable facture*. To the left of *impulse* appear terms describing the

characteristics of *held sounds* including *formed sustainment*, (held) *non-existing facture* and (held) *unpredictable facture*. The term *facture* refers to the way a sound evolves over time [18]. *Non-existing factures* are sounds that are too redundant to exhibit change over time. Similarly, *unpredictable factures* exhibits too much instability to be formed.

Combinations of characteristics at the horizontal and vertical planes of the table point to the sound objects notated in the body of the table. Hence the combination of an *impulse* type envelope and a *definite pitch* points to the sound object notated as N'. Similarly, sound object X" is defined by a *formed iteration* type envelope and a *complex pitch*. This notation provides an abstraction of the different types of sound objects. Clearly, there are multiple sound objects that fit the defining characteristics of N' or X": thus, in Schaeffer's sound examples, most of the notation symbols are demonstrated by more then one sample [24]. Each symbol therefore represents a *sound-object class*, a collection of sound objects that share the same TARTYP defining characteristics.

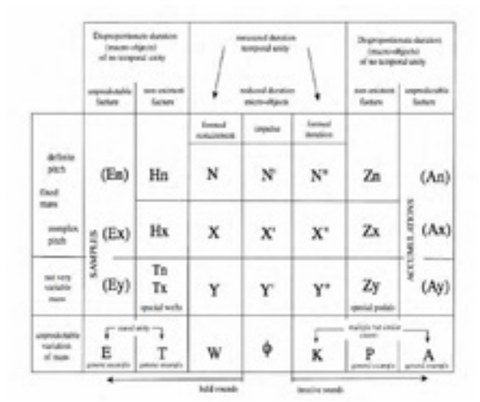

Figure 1. Pierre Schaeffer's TARTYP: time domain terms correspond to columns and frequency domain terms correspond to rows [6].

Schaeffer also subdivides this table into sub-collections of sound-object classes that are not explicitly notated in the table. The center of the table is a collection of nine sound-object classes (N, N', N", X, X', X", Y, Y', Y"): these sound-object classes are called *Balanced* sound objects. The columns to the right and left of the *Balanced* sound objects (three rows from the top) define the *Redundant* and *Homogenous* (*RH*) sub-collection, with the latter further subdivided into *RH_Held* (Hn, Hx, Tx, Tn) and *RH_Iter* (Zn, Zx, Zy). The entire bottom row of the table (W, Φ, K, P, A, T, E) and the far-right and far-left columns constitute the sub-collection of *Excentric* objects, further subdivided into *Sample* objects on the far-left column (En, Ex, Ey, E) and *Accumulation* objects on the far-right column (An, Ax, Ay, A) [6].

4. TARTYP DERIVED GRAMMARS

In this section we present generative grammars derived from the TARTYP classifications of sound objects and its defining elements as well as from the structure of the table and its sub-collection. The rewrite rules of these grammars use the time and frequency properties specified at the margins of the table as terminals. For each of the sub-collections of the table, we define a grammar in which each terminal equals a subset of the notated sound-object classes. A set of rewrite rules in such a grammar yield a large number of paths where each of these paths can be composed out as a sequence of sound objects. In addition, we will present a table grammar that unifies all the sub-collection-based grammars and initiates a hierarchical structure of sound object sequences.

To explain and exemplify a sub-collection-based grammar, we now discuss the grammar of *Balanced Object* sub-collection. This collection includes the nine sound-object classes at the center of the table (N, N', N", X, X', X", Y, Y', Y"). The grammar for this sub-collection uses the set terminals specified in formula (1). The latter also specifies the subsets of the *Balanced object* sub-collection equivalent to each one of these terminals.

```
DEFINITE = {{[N | N' | N" |]+}}                    (1)
COMPLEX = {{[X | X' | X" |]+}}
VARIABLE = {{[Y | Y' | Y" |]+}}
IMPULSE = {{[N' | X' | Y' |]+}}
FORMED_ITER = {{[N" | X" | Y" |]+}}
FORMED_SUS = {{[N | X | Y |]+}}
```

The general structure of a rewrite rule is:

```
head -> body                                       (2)
```

denoting that the head of the rule can be rewritten as the body. More specifically, the structure of our rules is always:

```
head -> [terminal] terminal [non-terminal]         (3)
```

where the items in [] are optional. Thus legal rules may look like, e.g.,

```
head -> terminal                                   (4)
head -> terminal non-terminal
```

A "terminal" corresponds to subsets of sound-object classes, while "non-terminal" may be any other symbol. The head is always selected from the set of "non-terminal" symbols plus a special start symbol.

An example of a legal set of rewrite rules for the BALANCED grammar having special start symbol "balanced" is:

```
R0: bal_expre_v ->   VARIABLE                      (5)
R1: bal_expre_fs ->   FORMED_SUS
R2: bal_expre_fi -> FORMED_ITER COMPLEX
R3: bal_expre_fs -> FORMED_SUS IMPULSE
R4: bal_expre_i -> IMPULSE FORMED_SUS bal_expre_fs
R5: bal_expre_v -> VARIABLE FORMED_ITER bal_expre_fi
R6: bal_expre_fi -> FORMED_ITER IMPULSE bal_expre_i
R7: bal_expre_fi -> FORMED_ITER VARIABLE bal_expre_v
R8: bal_expre_v -> VARIABLE FORMED_SUS bal_expre_fs
R9: bal_expre_fs -> FORM_SUS FORM_SUS bal_expre_fs
R10: bal_expre_fs -> FORMED_SUS IMPULSE bal_expre_i
R11: bal_expre_i -> IMPULSE IMPULSE bal_expre_i
R12: bal_expre_v -> VARIABLE IMPULSE bal_expre_i
R13: bal_expre_c -> COMPLEX FORMED_SUS bal_expre_fs
R14: bal_expre_fs -> FORMED_SUS COMPLEX bal_expre_c
R15: bal_expre_d -> DEFINITE FORMED_SUS bal_expre_fs
R16: bal_expre_c -> COMPLEX FORMED_ITER bal_expre_fi
R17: bal_expre_v -> VARIABLE VARIABLE bal_expre_v
R18: balanced ->   FORMED_SUS bal_expre_fs
R19: balanced ->   IMPULSE bal_expre_i
```

Here, we use all caps to indicate terminals, and lower case to indicate non-terminals, although the choice of non-terminal symbols is entirely arbitrary, and here are selected to reflect the structure of the TARTYP collections. The rule set is legal, because it contains at least one rule with a start head, contains no duplicate rules, and all non-terminal symbols that appear in the body of a rule have at least one other corresponding rule where they appear only in the head.

The rewrite rules are used to construct a path, that is, a sequence of terminal symbols selected according to the grammar expressed in the rewrite rules. To construct a path, we set it initially to the special start symbol (e.g., "balanced" in (5)). While the path contains non-terminals symbols, we select at random a rule having that non-terminal as the head, and replace the non-terminal with the body of the rule. Figure 2 shows an example of a path constructed by the set of rules in (5), consisting of a sequence of terminals derived from Rules 19, 11, 4, and 1.

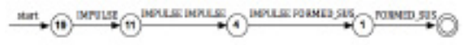

Figure 2. A path generated from the sample rule-set of equation (5). This is one of many alternative legal paths through the space defined by these rules.

The structure of a rewrite rule and a rule-set in the other grammars - *RH_Held*, *RH_Iter, Excentric, Sample* and *Accumulation* – is similar to that of *Balanced* grammar. The differences between these grammars lie in the set of terminals of each grammar and in the equivalent subsets of sound-object classes. The following example specifies the set of terminals and the equivalent subsets of sound-object classes for the grammar *Sample*:

```
DEFINITE = {{[ En ]+}}                             (6)
COMPLEX = {{[ Ex ]+}}
VARIABLE = {{[ Ey]+}}
UNPREDICTABLE = {{[ E ]+}}
HELD_UF = {{[En | Ex | Ey | E ]+}}
```

To combine these six grammars into a single process, we add collections of rules whose terminals correspond to the six different special start symbols of the six different grammars. Thus generating a path in this "table grammar" consists of sequencing invocations of the other six grammars accordingly, starting from the new global special start symbol "start". Note that the new grammar reflects the structure of the TARTYP sub-collections; thus, we can implement the sub-table structure using appropriate rules (see Figure 3).

Figure 3. Grammar-based hierarchical structure enabled by table and the sub-table grammars. This tree-like hierarchy unifies all the types of grammar.

5. RULE GENERATION

We implement a generate-and-test algorithm designed to produce a set of rewrite rules that conform to the three legality criteria discussed in the previous section. The input to our generation algorithm consists of a set of parameters indicating how many rules are in the rule set, how many of these rules are required for each special start symbol, and how many rules have bodies consisting solely of terminal nodes. The algorithm selects the next rule to generate in accordance with the rule set parameters (e.g., having a special start symbol as head, or having only terminals in the body). First, we randomly select the terminals in the rule body from the terminal symbols and then optionally include a non-terminal symbol in accordance with the terminal symbols used. The head of the rule is either a special start symbol or one of the other non-terminal symbols. Duplicate rules are rejected, and a new rule is generated to replace it. Once the rule set is complete, we check the rule set for consistency, ensuring that every non-terminal in a rule body has a matching head in at least one other rule. If a rule with no such match exists, it is eliminated, and a new rule is generated in its place. (See Figure 4).

Recall that once a legal rule set is produced, a second algorithm is used to extract a path or a sequence of terminals. First, find all the rules with head corresponding to the start symbol; randomly select one of these rules. Next, Find all the rules whose head matches the non-terminal

ending the selected rule; randomly select one of these rules, and replace the matching non-terminal with the body of the selected rule. Finally, if the rule selected in the second step has a non-terminal, repeat the second step.

```
define (n_start: int, n_continue: int, n_end int)
  while (n_start>0 or n_continue>0 or n_end>0)
    # Generate complete set of rules
    while (n_start>0 or n_continue>0 or n_end>0)
      generate rule i
    # Check for duplicates
      if (rule i is not a duplicate)
        add rule to rule-set
            decrement n_start, n_continue or n_end as
            appropriate
    endwhile

    # Ensure rule set is legal
    while (n_start=0 and n_continue=0 and end=0)
      for rule in ruleset
        for each non-terminal in body of rule
          if no other rule has head that matches
                  non-terminal
              delete rule
              increment n_start, n_continue or n_end as
                    appropriate
    endwhile
  endwhile
```

Figure 4. Formal specification of legal rule-set generation algorithm described in text.

6. INTERACTIVE INTERFACE AND COMPOSITION

The algorithms above were implemented as four Java classes. These four classes are embedded in the Max/MSP or the Pure Data environments as extensions of the MaxObject class. They are combined to create an **mxj** type object (or **pdj** object in PD), which creates (in real time) a legal set of rules in one of the grammars and then constructs (in real time) multiple legal paths from the same set of rules. The process repeats, generating new sets of rules and extracting additional paths. **mxj** objects are available for all grammars – *Balanced, RH_Held, RH_Iter*, *Excentric, Sample* and *Accumulation*, as well as for the structural grammars *table* and *sub-table*. These objects function in a simple way. They receive a list of integers in the left inlet. This list is interpreted by the object as the number of rules of each type to be generated. The set of rules is posted in the max window. Following the generation of the rule-set, with each bang received in the left inlet the object outputs a path or a sequence of terminals as symbol list from its outlet.

We now present a way to create an interactive interface with the above **mxj** grammar objects as well as a method to compose with this interactive interface. Since an mxj grammar object output a path as a symbol list, this list can be stored in a **coll** object to allow access to items on this list. The list is entered in the **coll** object and read from it using the patch shown in Figure 5. The list is read in this case in an arbitrary timing based on the bangs generated by the **metro** object, however, a similar patch can be used to reflect a structural time organization.

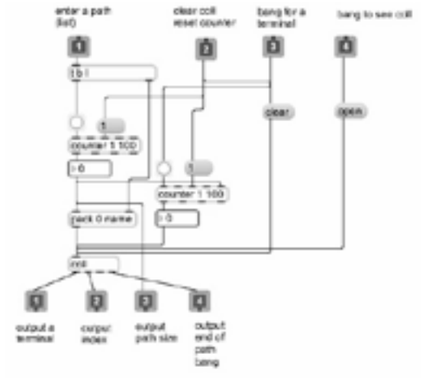

Figure 5. A Max/MSP patch using a **coll** object for storing and accessing a path as a list.

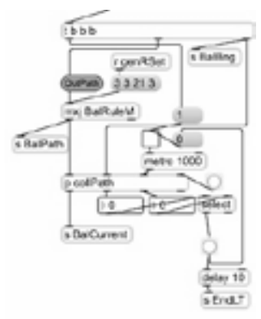

Figure 6. A Max/MSP patch using a grammar object to generate rule-sets and paths.

The patch in Figure 5 is embedded in the patch in Figure 6. The latter includes the object **<mxj_BalRuleM>**. As shown in the figure, a message box sends a list to the inlet of this object that specifies the number of rules to be generated. The **live.text** object bang the **<mxj_BalRuleM>** object to output a path. The patch in Figure 6 is part of a larger patch that simulates the tree-like hierarchy represented in Figure 3. The upper part of this larger patch includes the grammar object **<mxj_TabRuleM>** that generates a rule-set in the *table* grammar. When a path is output by this object the list of terminals is compared in the **select** object that activates the grammars *Balanced*, *RH_Held*, *RH_Iter* or the *sub_table* grammar that in turn would activate the grammars *Excentric, Sample* and *Accumulation*. The activity in this tree like patch is monitored in an interface such as the one shown in Figure 7.

In the tree-like hierarchy shown in Figure 3, the *table* and the *sub-table* grammars provides the background level of the structural organization while the other grammars are the middle ground of this organization. A path extracted in the *table* or the *sub-table* grammars would be advanced in relation to the paths extracted in the middle ground grammars. If for example the path BALANCED RH_HELD BALANCED RH_ITER is extracted in the table grammar, when the first terminal is read it activates a path in the *Balanced* grammar. The second terminal will be read only when the path in the *Balanced* grammar ended. Therefore, the middle ground grammars are a defining element in the time organization of the hierarchical structure generated by this patch.

There are many ways to connect such a structure to a foreground of sound generation and processing. One possible way is to map the terminals through **select** objects to bang messages that activate the playback of sound files. Another way is described in the following lines. This method is using spectral signatures derived from Pierre Schaeffer's sound recordings that exemplified the TARTYP and its defining characteristics [24]. From each one of the sound objects in these examples we extracted 64-bin FFT frames to create a spectral signature. These frames were saved as lists of values in simple text files. In a PD patch like the one shown in Figure 8, the FFT frames are used to filter a live sound and hence apply the a spectral signature of a Schaefferian sound object on the live sound. Similarly the waveform of a Schaefferian sound object, stored in an array object, is used to generate the amplitude envelope of the processed signal.

The sub-patch <**pd_Paths**> in the upper-right corner of Figure 8 use the grammar objects to generate rule-sets and sequences of terminals in the way exemplified in Figure 5 and 6. The patch then selects FFT frames and waveforms originated from sound objects that are members of the sub-sets equivalent to terminals in the sequence. The data of these FFT frames and waveforms is entered to the arrays FFTframe and AmpEnv. For example, if the terminal DEFINITE is part of a path extracted from the *Balanced* grammar it will cause the selection of FFT frames and waveforms extracted from the sound objects exemplifying the sound-object classes N or N' or N".

This sound processing system is combined with the hierarchical structure generated by the grammars discussed in previous sections. As discussed before, the sub-collection grammars define the time organization at the middle ground of this compositional process. At the foreground, the temporal organization is derived from the amplitude envelopes generated by the waveforms selected from Schaefferian sound objects. For example, if the DEFINITE terminal of the previous paragraph is followed by an IMPULSE terminal, the latter will be read once the envelope generated for the DEFINITE terminal ends.

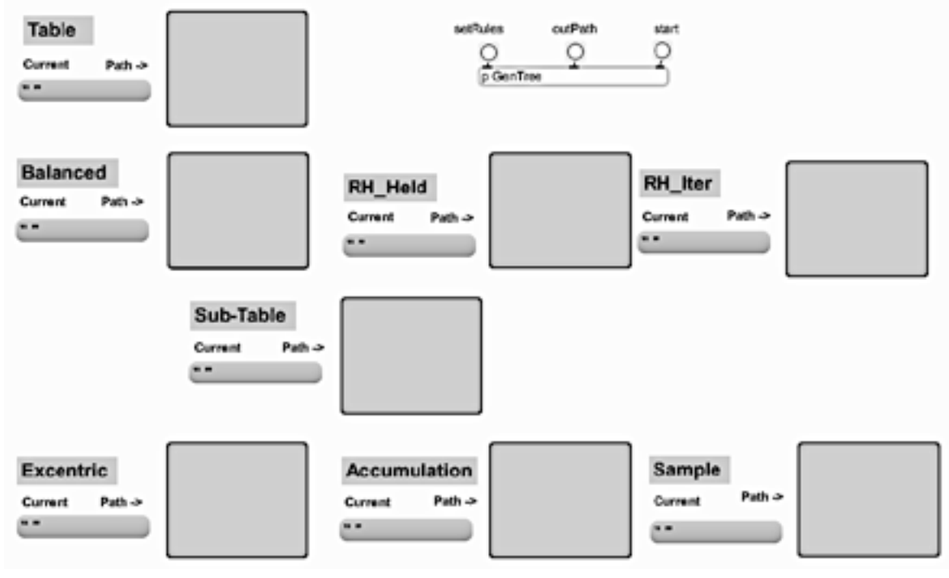

Figure 7. A sample interface used to monitor the activity in the tree-like Max/MSP patch described in the text. In this interface the extracted path are shown in the larger boxes and the current terminal playing in the message box under the title of the grammar.

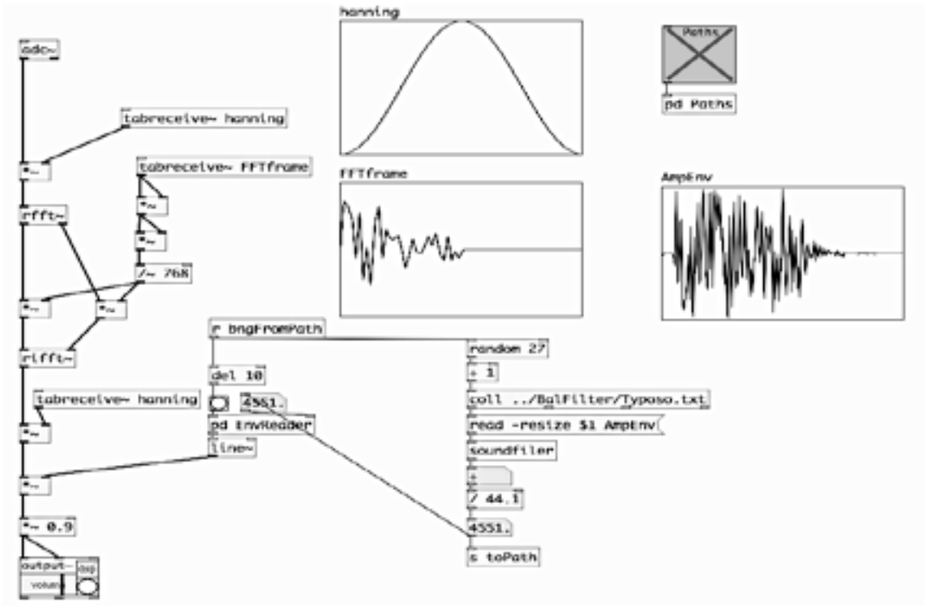

Figure 8. A sample sound processing PD patch applying the a spectral signature of a Schaefferian sound object on live sound.

7. CONCLUSION

The tools presented in this paper are aimed toward advancement of the capability to generate and re-compose in real time unified musical structures. The generative grammars created for these tools were constructed to reflect the structure of Schaeffer's classification of sound objects as presented in the TARTYP. Furthermore, the grammars preserve the terminology and the meaningful inter-class relationships which are an essential part of this table. Finally this paper presents a compositional method that utilize these tools while maintaining contextual ties to the same resource. Currently, this level of structural relationships is achieved partly by predetermined programing elements as well as predetermined choice of context and resources. In future work, we hope to develop the tools that will allow the user to define grammars according to the user's choices of the context and resources, while maintaining the same level structural relationships.

8. REFERENCES

[1] Baroni M, Brunetti R, Callegari L, Jacoboni C. "A grammar for melody. Relationships between melody and harmony." In *Musical grammars and computer analysis*. Edited by M. Baroni, L. Callegari. Casa Editrice Leo S. Olschki, Florence, 1984.

[2] Bel, Bernard. "Symbolic and sonic representations of sound-object structures." In *Understanding music with AI*. Edited by M. Balaban, K. Ebcioglu, O. Laske. The Press MIT, Cambridge, 1992.

[3] Bel, Bernard. "Migrating Musical Concepts: An Overview of the Bol Processor" *Computer Music Journal* 22, no. 2 (1998): 56-64.

[4] Bel, Bernard and Jim Kippen. "Bol Processor Grammars." In *Understanding music with AI*. Edited by M. Balaban, K. Ebcioglu, O. Laske. The Press MIT, Cambridge, 1992.

[5] Camilleri, Lelio. "A Grammar of the Melodies of Schubert's Lieder." In *Musical grammars and computer analysis*. Edited by M. Baroni and L. Callegri. Casa Editrice Leo S. Olschki, Florence, 1984.

[6] Chion, Michel. *Guide des objets sonores*: *Pierre Schaeffer et la recherché musicale*. Editions Buchet/Chastel, Paris, 1983. English translation by John Dack and Christine North. London, 2009. http://www.ears.dmu.ac.uk /IMG/pdf/Chion-guide (accessed January 9, 2013).

[7] Chomsky, Noam. *Syntactic Structures*. Mouton & Co. The Hague, 1966.

[8] Dack, John. "At the limits of Schaeffer's TARTYP." *Music Without Walls? Music Without Instruments? Conference Proceedings*, 2001. http://www.dmu.ac.uk/research/research-faculties-and-institutes/art-design-humanities /mtirc/archive/musicwithoutwalls.aspx (accessed July 10, 2012).

[9] Granroth-Wilding, Mark and Mark Steedman. "Statistical Parsing for Harmonic Analysis of Jazz Chord Sequences." *Proceedings of the International Computer Music Conference* 2011.

[10] Heinemann, Stephen. "Pitch-Class Set Multiplication in Theory and Practice" *Music Theory Spectrum* 20, no. 1 (1998): 72-96.

[11] Holtzman, S. R. "Using Generative Grammars for Music Composition." *Computer Music Journal* 5, no. 1 (1981): 51-64.

[12] Koblyakov, Lev. *Pierre Boulez: a World of Harmony*. Harwood Academic Publishers, New York 1990.

[13] Laske, Otto. "KEITH: A Rule-system for Making Music-analytical Discoveries." In *Musical Grammars and Computer Analysis*. Edited by M. Baroni and L. Callegri. Casa Editrice Leo S. Olschki, Florence, 1984.

[14] Laske, Otto E. "In Search of a Generative Grammar for Music." In *Machine Models for Music*. Edited by Stephan M. Schwanauer and David A. Levitt. The MIT Press, Cambridge, 1993.

[15] Lasnik, Howard, Marcela Depiante and Arthur Stepanov. *Syntactic Structures Revisited: Contemporary Lectures on Classic Transformational Theory*. The MIT Press, London, 2000.

[16] Lerdahl, Fred and Ray Jackendoff, *A Generative Theory of Tonal Music*, The MIT Press, Cambridge, 1983.

[17] Lerdahl, Fred and Ray Jackendoff. "An Overview of Hierarchical Structure in Music." In *Machine Models for Music*. Edited by Stephan M. Schwanauer and David A. Levitt. The MIT Press, Cambridge, 1993.

[18] Normandeau, Robert. "A revision of the TARTYP published by Pierre Schaeffer." *Proceedings of the Seventh Electroacoustic Music Studies Network Conference,* 2010. http://www.ems-network.Org/IMG/pdf_ EMS10_ Normandeau.pdf (accessed July 10, 2012).

[19] Parson, Dale E. and Ryan R. Panuski. "Real-time Grammar-Based and Restructuring of Musical Streams." *Proceedings of the International Computer Music Conference*, 2011.

[20] Pelinski R. A generative grammar of personal Eskimo songs. In *Musical grammars and computer analysis*. Edited by M. Baroni and L. Callegri. Casa Editrice Leo S. Olschki, Florence, 1984.

[21] Rader, Gary M. "A Method for Composing Simple Traditional Music by Computer." In *Machine Models for Music*. Edited by Stephan M. Schwanauer and David A. Levitt. The MIT Press, Cambridge, 1993.

[22] Roads, Curtis. "An Overview of Music Representations." In *Musical Grammars and Computer Analysis*. Edited by M. Baroni and L. Callegri. Casa Editrice Leo S. Olschki, Florence 1984.

[23] Schaeffer, Pierre. *Traité des objets musicaux*. Paris, *Éditions du Seuil*, 1966.

[24] Schaeffer, Pierre. "Schaeffer's Typology of Sound Objects." In *Cinema for the Ear: A History and Aesthetics of Electroacoustic Music*. http://www.dmu.uem.br/aulas/tecnologia/SolObjSon/HTMLs/Schaeffer.html(accessed July 10, 2012).

[25] Steedman, Mark J. "A Generative Grammar for Jazz Chord Sequences." *Music Perception: An Interdisciplinary Journal* 2, no. 1 (1984): 52-77.

[26] Stefani, Gino. "Musical Competence, Analysis and Grammar." In *Musical Grammars and Computer Analysis*. Edited by M. Baroni and L. Callegri. Casa Editrice Leo S. Olschki, Florence, 1984.

[27] Sundberg, Johan and Bjorn Lindblom. "Generative Theories in Language and Music Descriptions." In *Machine Models for Music*. Edited by Stephan M. Schwanauer and David A. Levitt. The MIT Press, Cambridge, 1993.

[28] Thoresen, Lasse. "Spectromorphological analysis of sound objects: an adaptation of Pierre Schaeffer's typomorphology." *Organised Sound* 12, no. 2 (2007): 129–141.