Representing music as work in progress

Gerard Roma, Perfecto Herrera
Music Technology Group, Universitat Pompeu Fabra, Spain

ABSTRACT

In this chapter we discuss an approach to music representation that supports collaborative composition given current practices based on digital audio. A music work is represented as a directed graph that encodes sequences and layers of sound objects. We discuss graph grammars as a general framework for this representation. From a grammar perspective, we analyze the use of XML for storing production rules, music structure, and references to audio files. We describe an example implementation of this approach.

INTRODUCTION

The widespread adoption of internet access has raised great expectations with respect to music creation. On one hand, networks extend the possibilities for collaborative composition using computer-based tools by allowing intermediate objects to be shared. On the other, these tools can be accessed by a larger audience, and designed to be used by people with little or no musical training.

The recent focus on media sharing by internet users is reinforcing such expectations. The habit of sharing multimedia objects has facilitated an explosion in the culture of creative repurposement and recombination. Specifically in the case of sound recordings, there is a long tradition in sharing files for creative reutilization. Content in sites such as freesound.org (http://freesound.org), soundsnap.com (http://www.soundsnap.com) or sampleswap.org (http://sampleswap.org) is typically downloaded to be reused in music and multimedia products. This trend in the use of sound samples can be seen as an expression of an audio culture (Cox & Warner, 2004), influenced by a number of aesthetic traditions that have exploited the specific constraints of sound recordings, such as *Musique Concrète*, *Plunderphonics*, *Soundscapes* and acoustic ecology, or *Hip Hop*. The widespread of digital technologies has thus allowed using digital audio as matter for musical discourse, in a way that can no longer be represented using traditional music notation. Since understanding sound files is now part of the standard computer literacy, this kind of discourse can now be used as a means for expression by many computer users without the need of formal music training. As computers keep invading different areas of

music production, sound files have become prevalent as a way to represent musical events. Samplers and sample based synthesizers are among the most commonly used tools, offering simplicity and realism over other types of synthesis. On the other hand, most music is at some point recorded in some sort of audio sequencer or multi-track editor as an organized ensemble of sound files.

Some tools have appeared that attempt to relate the use of audio sequencers with the explosion of social networking and social media. Companies such as *SoundCloud* (http://www.soundcloud.com) are offering web hosting of audio tracks, and sites such as Indaba Music (http://www.indabamusic.com) are already offering online tools for basic audio mixing and sequencing. The makers of one of the most popular programs for audio sequencing, Ableton Live (http://www.ableton.com), currently offer a collaboration feature based on progressive uploading and downloading of audio clips.

While these movements toward the use of network servers for storing audio are allowing greater degrees of collaboration, current tools and their interfaces are still focused on single user operation, in many cases under the influence of classic western music notation. Currently popular programs do little to represent deep music structure, especially for practices based on digital audio manipulation. Moreover, most music is stored in proprietary formats and can't be moved from one program to another.

The difficulties of understanding music, and especially musical structure, when using sound recordings were largely explained in Schaeffer's *Traité des Objets Musicaux*. (Schaeffer, 1966). Given the impossibility to describe the practices that magnetic tapes made possible from the established music theory, Schaeffer frequently borrowed concepts from the linguistic theories of Saussure and Jackobson (an analysis of the relationships between music and language in the Traité can be found in Chion (1983)). In the 1970s, pioneers of computer music like Curtis Roads and Otto Laske proposed the adaptation of formal grammars to the practice of composing music with sound objects. While the use of grammars has been established in fields such as computational musicology, the early use of grammars for sample-base music composition provides a ground for current needs with respect to collaborative recombination of shared

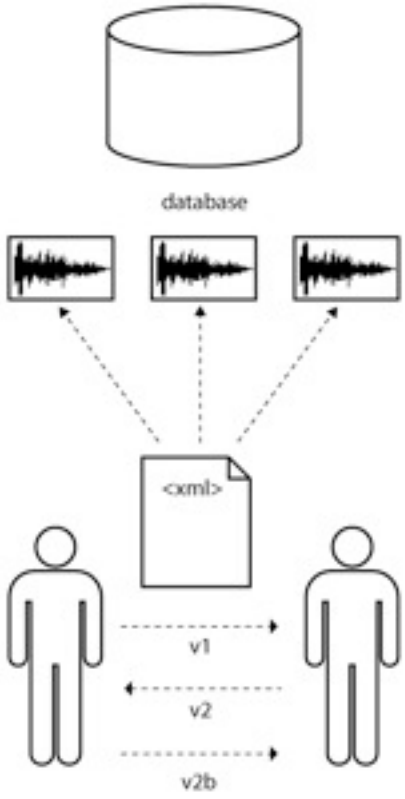

Figure 1. Separation of structure and content

media.

The separation of musical structure from audio signals allows the use of could-based services and shared databases for the audio, while musical structure can be represented and exchanged using text markup formats such as XML (figure 1). In composition activities, music structure can be typically stored in lightweight documents that may change frequently, and transmitted through established text communication channels such as email. Each music document may make reference to a number of bigger sized audio files. Transmission and local caching of these files can be dealt independently between each participant and the remote location through standard web technologies and services, which avoids the need of potentially complex specialized p2p tools for the synchronization of audio collections among different participants.

In collaborative composition, though, the exchange of music documents doesn't need to be reduced to a single document representing a complete work: much of this activity works through sharing and reusing lower level building blocks. Formal grammars provide a framework for the representation of different structural levels in music composition. Formal languages already allow high levels of cooperation in computer music. For example, Music-N style languages and environments are very often driven by lively communities that continuously exchange code and knowledge. We propose similar approach to support collaboration in sample-based music composition.

MUSIC REPRESENTATION WITH FORMAL GRAMMARS

Formal Grammars

Formal grammars were introduced by Noam Chomsky in the 1950's as a means for formal analysis of natural language (Chomsky, 1956). The view of grammars as mathematical devices has since then remained at the foundation of Computer Science. From the point of view of linguistics it was an important revolution that introduced a generative perspective: grammars served as language theories that could explain and predict linguistic phenomena by generating them from a formally defined set of laws.

In the 1970s and 1980s, the idea of modeling music with grammars became very popular. Formal grammars can be used to generate expressions of a language, and they can also be used to parse existing expressions, and give an interpretation of how they were generated. Thus, grammars were used, on one hand, for automatic or computer-aided music composition, and on the other for musicological analysis. Both directions can be thought as complementary, and in some cases grammars are learned from existing musical works and then used to generate new ones.

For the case of collaborative music composition, grammars provide a suitable framework for sharing, reusing and assembling parts of music compositions. In this sense, a grammar can

support the metaphor of a common language, as a means for the computational representation and manipulation of musical fragments. From the point of view of analysis, grammars can be used for computational modeling of the style of different participants or groups in collaborative applications. From the perspective of generation, they can serve to facilitate creativity by producing new combinations of sounds, or to assist the search of sounds for a given musical context.

A formal grammar is usually defined as a 4-tuple (V, Σ, S, P) , where:

- Σ is a terminal alphabet, a set of symbols that are used to form sentences in the language. In common music notation, terminals could be notes, durations or chords, while in sample-based music they can be sound objects.
- V is an alphabet of non-terminal symbols or variables. Variables represent strings of terminal or non-terminal symbols in intermediate stages of the generation. In a music composition process, variables can be used, for example, to represent groups of terminals that are often used together.
- S is the start symbol, a special variable that is used to begin the generation process.
- P is a set of production rules that allow a given string to be replaced by another string. For example they can specify how a part of a musical piece can be replaced by its subparts.

In summary, a formal grammar can be explained as a set of rules that rewrite a string of symbols with another one. For example the rule $A \rightarrow AB$ defines that a string composed of the symbol "A" can be replaced (rewritten) by the string "AB". This rule can be used to generate the string "AB" in presence of string "A". Thus, the rule could be applied again to "AB" to produce "AAB", and so on. Also, it can give one possible explanation of how the string "AB" (or "AAB" for that matter) was produced in a given language. This is the idea behind the generative linguistics methodology: a set of rules that is able to generate all observable sentences of a language constitutes a theory of that language that is able to predict (i.e. generate) unobserved sentences and provide a possible explanation of how the observed ones were produced.

Intuitively, grammars can be understood as formal way to specify structural groupings of a language. For example we can state that all sentences in a natural language are composed of a "noun"' sub-sentence and a "verb" sub-sentence, and then define a rule that defines this decomposition using abstract symbols (say $S \rightarrow NV$). Thus, they can be used to specify the structure of a musical piece. For example we could define a structure that is always composed by a sequence of themes "ABA' ".

Chomsky defined a hierarchy for formal languages and the associated grammars that has become standard in linguistics and computer science. The most general class, type 0, is the class of recursively enumerable languages. Grammars that parse or generate such languages are not subject to restrictions in the production rules. Type 1 is the class of context-sensitive languages. Production rules for grammars that generate these languages can only replace a string with another string of equal or greater length. Still, context-sensitive rules such as aAb -> aBb are allowed. Type 2 is the class of context-free languages. Grammars that produce these languages are subject to a second restriction, which forbids context-sensitive productions. The left side of

productions must consist only of a non-terminal variable. Also, the right side of productions cannot consist exclusively of the empty word. Intuitively, context-sensitive rules seem to have many applications in music. However, the most general types of grammars are also the most complex ones and in practice they are difficult to parse efficiently. Thus, it is common to use context-free grammars, sometimes augmented with transformation rules or control mechanisms. Context-free grammars are also widely used to describe and design programming languages. Finally, type 3 defines regular languages, the most limited class of formal languages. In addition to the above restrictions, the right side string of a production in a regular grammar can only consist of a terminal symbol and a non-terminal variable. Chomsky described this model as an equivalent to markovian processes and dismissed them for the analysis of natural language mainly because of their inability to describe nested structures. The stochastic nature of markovian processes does not affect this fundamental limitation. The same reason applies to their use for music analysis or generation. Despite this limitation, markov chains have been very popular as a simple mechanism for algorithmic music composition (Ames, 89). It can be argued, though, that the use of markov chains does not justify by itself the grammar paradigm: markov chains are represented simply using transition matrices.

Grammars in music composition

The application of grammars to collaborative composition can rely on a long tradition in the use of grammars for computer music. A brief review of some of the approaches can be helpful to understand their potential.

One of the first documented efforts to use formal grammars in music composition is due to Curtis Roads. In *Composing Grammars* (Roads, 1978) he described a system for music composition based on context-free grammars augmented with control procedures. The system provided the composer with a workflow for experimenting with structural and semantic aspects of composition. First, the composer would specify a grammar using a specialized language (Tree) and an associated compiler (Gram). The program would generate a compiler for the specified grammar. The composer would then work on valid derivations according to the grammar to create the syntactic surface. A second language (CoTree) and its corresponding compiler (GNGRAM) would aid in the generation of the score. A final task, the lexical mapping, consisted in pairing the terminals of the grammar with sound objects previously created by the composer. Such amount of relatively low-level tasks reflects the kind of interaction that computers supported at that time. Still, the emphasis on the lexical mapping and the use of sound objects makes this pioneering work interesting in the context of social media.

In *Grammars as representations for music* (Roads, 1979), Roads presented a synthesis of formal grammar theory and surveyed the use of grammars for music analysis. Perhaps more importantly, he summarized the powers and limitations of the grammar approach. Considering iconic (i.e. based on analogies) and symbolic (based on convention) representations, it is quite obvious that, as symbolic representations, grammars rely on a discretization of the sound material. This limitation is however less restrictive for compositional purposes than for analysis. An example of discrete treatment of sound is the *schaefferian* concept of sound object (Schaeffer, 1960). A second limitation is the compromise in complexity. As we mentioned, the most complex

types of grammars are often too complex to parse, while simple grammars can be too trivial and less effective than other models. A third limitation is that grammars are purely structural and hence they don't deal with the semantic and social implications of music. Despite these limitations, the scope of grammars for modeling different kinds of music is huge. In computer music, where most applications rely on some sort of storage, grammars can be used to represent structure in a very broad sense.

Holtzman's Generative Grammar Definition Language (GGDL) was developed as a tool for investigation of structural aspects of music using a computer (Holtzman, 1980). The language could be used both by composers and musicologists. GGDL allowed to specify type 0 (i.e. free) grammars and provided support for phrase structure rules and transformational rules. Phrase structure rules are standard formal grammar string rewriting rules. Since many rules can be applied at a given point, the system provided a mechanism for defining functions to control this choice. For example "blocked" generation allows random choice of rules but restricts the use of a rule until all the possible selections have been made. Metaproductions are a special type of rewrite rule that allow the generation of rewrite rules at initialization time, before actual variables are initialized. Transformation rules modify the strings generated by phrase structure rules in different ways, such as transposing them or inverting them. Finally, GGDL provided a means for mapping abstract symbols to actual sounds synthesized with the possibilities offered by computers of the time. Holtzman provided a complete example of the generation of a piece. As a given grammar can generate a large number of pieces, the composer is encouraged to experiment with the program until an acceptable result is obtained. This resort to manual experimentation can be seen as an effect of the lack of restrictions that the language imposes to grammars. The author also explained how Schonberg's *Trio* could have been generated by the example grammar. Still , the system doesn't provide any automatic support for such musicological analysis.

Kippen and Bel's development of the BOL processor system (Kippen and Bel, 1988) has been extensively documented along different phases. The system was originally conceived for linguistic analysis of North-Indian tabla music, a very formalized system that uses an oral notation system of mnemonic symbols called Bols. Tabla music is usually improvised, typically involving permutations of a reference pattern. Expert musicians can assess whether a given permutation is correct or not. On this basis, the authors tailored several formal grammars that reflected correct variations. The particularities of tabla music led to the introduction of different context-sensitive rules, such as negative contexts (a string is replaced except if found in a given context) in order to reflect the description of the rules by musicians. The main components of the systems are the inference engine, which generates sentences from the grammar, and the membership test that determines whether a given sentence belongs to the grammar. A graphical interface allowed users to perform both analysis and synthesis. In order to parse strings introduced in the graphical editor without the full formal structure specification, a system of templates was introduced. A second iteration of the Bol processor, named BP2 targeted grammar-based music composition from a more general perspective, allowing composers to specify their grammars to generate sound object compositions. Because of this focus on composition, BP2 omitted the parser mechanism and allowed a more free approach to grammar specification, subordinating the issue of correctness to aesthetic considerations.

Finally, one of the most well-known uses of grammars for music composition is David Cope's Experiments in Music Intelligence (EMI) (Cope, 2001). Over the years, Cope has refined a database-driven system that imitates the compositional style of classic composers. The works of the target composers are segmented and described in a database, and each fragment is assigned to a category according to a system called SPEAC: Statement, Preparation, Extension, Antecedent and Consequent. Such categories attempt to define a basic formalization of the dynamics of tension and relaxation in western tonal music. Thus, the system defines a set of rules that make a sequence of patterns of different categories correct. For example, an Antecedent fragment can only be followed by an extension or a consequent fragment. The music generation engine is based on an Augmented Transition Network, which allows for faster parsing and generation of context-sensitive rules.

One issue of music grammars that is not covered by linguistics or formal languages literature is parallelism (Roads, 1982). Both Roads and Holtzman made use of parallel rules, where two parallel tokens are meant to start at the same time. However, parallel rules introduce some ambiguity. For example if we have a musical sequence "AB" and a parallel rewriting rule "A → D/E" (meaning that D and E start at the same time), it is not clear, upon replacement of A, if B will follow after D or after E. Graph grammars provide a general framework that allow us to deal explicitly with sequential and parallel structures.

Graph Grammars

Graph grammars were introduced by Pfaltz and Rosenfeld in the late 1960s (Pfatz and Rosenfeld, 1969) as an extension of traditional grammars to languages of directed graphs. A directed graph is defined as a tuple (N,E) where N is a set of nodes and E a set of edges that connect nodes in a certain direction. Clearly, strings are a class of directed graphs where symbols are nodes and edges define the sequence of symbols. In this sense, edges of a string define a total order relation. If cycles are forbidden in a directed graph (where cycles are defined as loops involving more than one node), the set of edges defines a partial order relation on the nodes, which allows the generalization of string grammars to acyclic directed graphs.

A graph grammar can be defined in similar terms to string grammars. However, graph rewriting productions are more complex than string rewriting productions as they have to define how to connect the result of the production to the enclosing graph. Thus, productions are defined as triples (α, β, E) where α is the (sub)graph to be replaced and β is the replacement, while E defines the embedding of β in the host graph. Graph grammars can be categorized in the same way as string grammars. For example, node replacement grammars (Engelfriet and Rozenberg, 97) are context-free graph grammars where the left hand of each production is restricted to a single node.

Development of graph grammars has continued over the years both at a theoretical and at a practical level fostered by applications very diverse fields such as image recognition or graphical languages for engineering (Ehrig, H., Engels, G., Rozenberg, G. and Kreowski, H., 1999). The extension of strings to graphs seems naturally suited for music representation by allowing parallelism. However, explicit mention of graph grammars for music is rare in the literature.

Some works (Cook and Holder, 2000, Madsen, 2003) have showed their use for mining classical music scores represented as graphs with multiple possible connections between consecutive notes. Since these connections are not specified in the score, this approach bears a high level of complexity that is not needed in the context of music composition. In the tradition of composing grammars, music surface may be represented by a rooted tree, where the direction of links indicates temporal sequences (note that this is different from parse tree that could represent a musical piece as a vertical hierarchy). A rooted tree can be defined as a directed acyclic graph with a root node where there is a unique path from the root node to any node. Intuitively, this structure forbids two things. On one hand, no cycles can exist in the graph. This means that edges define a partial order relation, which allows them to represent time sequences. Also as shown by Pfatz and Rosenfeld (1969), acyclic graphs have the property of being contractable, which allows the definition of grammar expansion rules. One problem with acyclic graphs is that it's quite common in music to use cyclic structures. Since loops (i.e, edges from one node to itself) are still possible with respect to the partial order relation, musical cycles can be understood as a finite number of repetitions of a single node, to which a graph has been contracted (figure 2). A second restriction of this representation is that a given node can only be the target of one edge. Two edges arriving at the same target would imply the scheduling of the same node (and all the following structure) at two different moments in time, which has the effect of creating multiple auditory tracks from the same graph specification, and breaking the intuition of the temporal sequence of the representation.

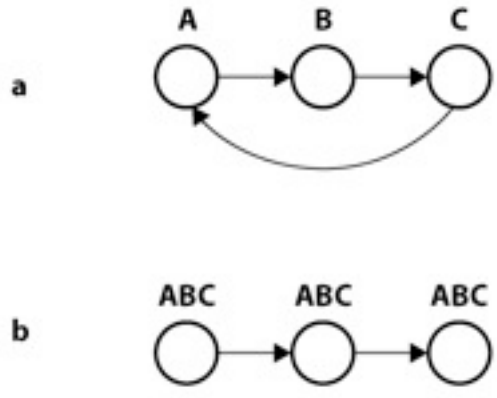

Figure 2.
A cycle can be avoided by contracting and repeating the graph

An example of such approach is shown in figure 3. Figure 3a shows a rhythmic pattern in a visual time grid as is common in many music programs. Each row of this grid is mapped to a different percussion instrument. Figure 3b represents the same pattern in a graph. An initial node has been added to represent the common starting point. From then on, time is determined by the duration of each sound object, including silences. Assuming all nodes have a fixed duration, this representation is obviously equivalent to the previous one, but is based on the sound objects instead of the time grid. As this pattern could appear a number of times, one might wish to

identify groupings. Figure 3c shows a layout that better represents the relationship between the bass drum and the snare drum pattern. This sub-pattern can now be collapsed and reused elsewhere.

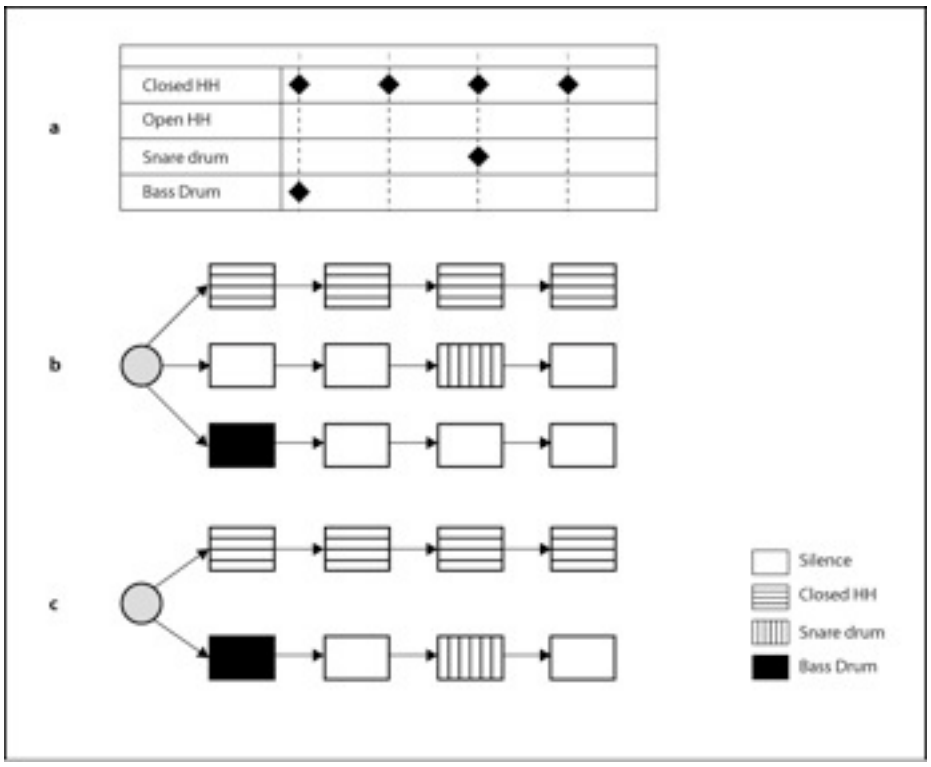

Figure 3. Representation of a drum pattern as a graph

The rest of this chapter presents and discusses XML representation of music as a network of sound samples using graph grammars. This implies the representation of music at three distinct levels: grammar rules, musical surface and lexical mapping. Grammar rules represent patterns that can be extracted from musical compositions. If a normative approach is pursued, compositions must validate against these rules. In the field of computer music composition, we have seen that grammars have been used by composers to define the rules that govern their pieces. Other examples of normative forms are conventional pop songs or sonatas. The process of composition then consists in the application and refinement of the rules. Representation of music grammars can be used by music editing programs both to parse and validate compositions created by users. Rules can also be understood as frequent patterns that are mined from several musical pieces and used to model the style of a user, group or community, and generate new pieces that follow these patterns. At the surface level, it is useful to represent music structure in an analog way to what is presented to the listener. This is the most common kind of document that is used to store and exchange music. The grammar approach provides the appropriate framework for the representation and computational manipulation of different fragments of the surface representation. Finally, we need to represent the mapping of the symbols in the surface representation to actual sounds and their physical location. This mapping can be defined as a direct link to sound resources, or it can be specified as an abstract description of a sound segment.

GRAMMAR REPRESENTATION

We have seen how grammars can be used both to parse or analyze the structure of musical compositions and to generate new compositions. This implies that a grammar can be used as a compositional aid to analyze and model the style of different users or groups, and suggest potential interesting combinations of sounds. A grammar can also be seen as an agreement between remote collaborators working in different parts of a musical work. A central issue is how can we encode grammars in XML documents so that they can be shared among different programs and still remain readable.

The most well known representation of (context free) grammars is the Backus-Naur Form (BNF). Essentially, it defines a grammar as a set of production rules, each one composed of a non-terminal symbol (the left hand side of the production) and a set of alternative possible replacement strings, composed of both terminal and non-terminal strings. Several variations of this form exist and are widely used in the definition of programing languages and protocols. An example of XML representation of grammars that is equivalent to a BNF variant can be found in the speech recognition grammar specification (http://www.w3.org/TR/speech-grammar), which provides the option of ABNF and XML representations. Rule definitions have an attribute that uniquely identifies the rule within a document. Thus the Universal Resource Identifier (URI) of a rule is absolutely defined by the URI of the document and the identifier attribute of the rule. A rule definition contains rule expansions, which may be tokens, rule references, or sets of any of both (sequences, alternatives and repeats). This limits the scope of these grammars to context-free grammars, where each rule represents a variable in the grammar and can be expanded to sequences or alternatives of variables and terminals. Tokens are terminal symbols (in the case of speech recognition grammars, tokens are usually text). Rule references are simply references to URIs of other rules. Sequences are defined by the order of XML elements, while alternatives are defined by a specific tag. Repeats are marked by an attribute in the repeated element.

This example illustrates some straightforward aspects of the design of XML representations of grammars, such as the nesting of rules and the use of references. For music representation it is common to try to preserve context sensitivity, depending on the level of automation that is needed. Context-sensitive grammars are usually quite complex for computational manipulation, but some composers chose to manually deal with such grammars in the 80s. Context sensitivity implies that both the left hand and the right hand of a production rule can be represented as a compositions of tokens and references to other rules. On the other hand, musical grammars will need to accommodate parallelism, so these compositions can be represented as graphs. The representation of graphs is discussed in the next section. However, graph grammars introduce the issue of embedding, which must be taken into account for the representation of rules.

A graph grammar rewriting rule defines that a given subgraph (embedded in a host graph) can be replaced by another subgraph. The problem is: how will this new subgraph connected to the host graph? In string grammars, embedding is trivial. A string can be seen as a simple graph where each token is a node connected to the following token with a directed edge. A string replacement rule obviously implies connecting the first token of the new string to the token that preceded the replaced string, and the last token to the token that followed the replaced string. In graph grammars, the replaced subgraph can have an arbitrary number of connections to the host

graph depending on the problem. Again, this can be simplified in the case of context-free grammars. Several strategies, such as Node Label Controlled (NLC) embedding have been defined for node replacement grammars (Engelfriet and Rozenberg, 1990). In NLC grammars, embedding rules are defined globally for all rules as relations among nodes with some specific labels. It has been shown that this kind of grammars is less general for the purpose of graph rewriting than other grammars where embedding rules are defined locally. However, this mechanism can be used as a simple approach to define musical grammars. For example, one may concede that any music fragment has a starting point and an ending point. A node replacement grammar can then be defined so that when a node is replaced by a graph, the starting point inherits incoming edges of the replaced node, and the ending point inherits its outgoing edges. The ending node does not need be the one that ends last. Continuing with the example of a drum loop, the last cymbal sound in a drum pattern can be less important than the last bass drum or snare sound to define the end of the pattern. This strategy does not allow maintaining parallel connections among "tracks" in a single expansion. Still, it can be argued that the need of such parallel connections implies the need of separate rules. On the other hand, synchronous start of several sounds (parallelism) will require the addition of a virtual initial node with no sound. An example implementation of this idea is described later.

SURFACE REPRESENTATION

The concept of surface structure was introduced in the context of linguistics by Chomsky to emphasize the power of grammars to describe the "deep structure" of a language. From the point of view of a generative grammar, the observable structure of sentences is able to convey meaning because it has been generated by the rules of a grammar. Jackendoff and Lehrdal (Jackendoff and Lehrdal, 1981) popularized the identification of the score representation of music in the western tradition with the "musical surface". While the score representation is limited with respect to other musical cultures, the use of structured documents to represent musical surface can be used to share musical ideas of a wide range of styles and cultures. The point of the grammar approach is that the surface representation should reflect sequences and parallelism as relationships among musical events, and not merely represent their position in a time line. In this sense, the general use of timed events for storing and reproducing music implies that the actual musical structure is not taken into account by programs, which hinders their use for collaborative composition.

Many XML representations of music surface have been proposed for traditional western notation. Since they are subdivided in parts and measures, music scores can be represented naturally in XML as hierarchies. This is the case for example of MusicXML and MDL (Good, 2001). Still, there is a wide variety of musical practices that cannot be represented with western notation. A representation of music as an ensemble of audio segments can be used for some of them. This approach was enabled by the MPEG-7 standard with the definition of the Segment Descriptor Scheme (DS), a description scheme for multimedia objects that allows defining them as compositions of segments. The MPEG-7 Segment DS allows the definition of hierarchies and arbitrary relationships between media segments such as audio. Still, the standard is more oriented to the description of existing content (e.g. for indexing and navigation) than to the creation of the

new content. On the other hand, the standard addresses all kinds of multimedia content, including video, images and 3D. This generality adds a lot of overhead by introducing many complexity layers that are not needed for music composition.

As described earlier, a relatively general approach to music surface representation is to use a graph where nodes represent sound objects that can be linked to audio files. By restricting this graph to a rooted tree we can then easily cut and collapse fragments of a music document into grammar variables, or meaningful units that can be shared and reused. From a conceptual standpoint, the main issue with this approach is that silence is no longer considered an empty space, but a part of the composition. This contrasts with the "ruler-style" representation of time that has become general in music sequencers, although not with the representation of silence in traditional music notation. In this sense, the use of graph notation assumes that the sound objects associated to terminal tokens of the grammar already have an appropriate length with respect to the rhythm of the piece. The lexical map should allow to specify start and end points if the represented object is a fragment of the referenced sound file.

We can now simply focus on the representation of music fragments as graphs. A graph is usually described as a set of nodes and a set of edges. A general approach to representing graphs in XML is GraphML (http://graphml.graphdrawing.org). GraphML documents basically contain lists of node and edge elements. Node elements have a mandatory identifier attribute, and edges refer to node identifiers in their source and target attributes. A node identifier is necessarily unique only within a graph document, and is used to represent the structure of the graph. Thus, it cannot be used to represent any other information. Additional data can be defined for nodes and edges but is kept separate from the structure. GraphML aims to gather some consensus as a general way to exchange graph data. In this sense, using GraphML for music applications could enable the use of general purpose graph drawing programs and libraries for music editing. However, GraphML is not targeted at the definition of grammars. On the other hand, music applications can benefit from a more specific format that defines music oriented restrictions directly in the XML schema definition. For the case of music surface representation, nodes represent terminal tokens that at some point are associated to sounds. While in GraphML nodes can be absolutely identified by the document URI and the node identifier, it may be convenient that node elements can directly point to their lexical mapping (in the case of the musical surface) or to a rule definition (when graphs are used in the grammar specification). With respect to edges, additional information may typically not be necessary, but some uses can be described. For example for the representation of algorithmic compositions, edges can be labeled with probability values.

LEXICAL MAP

The interest in formal grammars for music composition was mainly influenced by their success with natural language. In linguistics, formal grammars have made possible an understanding of the importance of structure in the transmission of information with independence of the meaning of words. In natural languages, the meaning of words has generally nothing to do with their

written or phonetic representation but is conventionally defined. Musical objects, such as notes, generally do not refer to real objects or concepts, but they are articulated into higher level structures like in other languages. This parallelism in the use and articulation of discrete symbol systems has been related to a more general principle of self -diverifying systems (Merker, 2006).

The concept of a lexical map between terminal tokens of a grammar and actual sound objects was investigated by Roads (Roads, 1979). Roads defined three general forms of lexical mapping. The first form is arbitrary mapping. The other two forms require a lexicon of sound objects that is grouped according to some acoustic features. This grouping is understood as relevant to a grammatical function, so that the mapping is not arbitrary. In our drum loop example, this could be illustrated by a classification of percussion instrument sounds. Roads distinguished between injective (each terminal maps to one sound from the ordered lexicon) and polymorphic (one-to-many, many-to-one) mappings. Polymorphic mappings were regarded as a complex situation equivalent to context-sensitive rules.

Current audio description technologies based on feature extraction and machine learning techniques allow us to establish functional groupings. This means that grammars that use discrete descriptors as their terminal alphabet can be used for analysis and generation of music based on audio segments. Generally speaking, a lexical map will describe how the alphabet of terminals used in the production rules is assigned an actual sound object. In this sense we will consider any alphabet to be a partition of a collection of sound objects, so that all elements of the database are mapped to some symbol of the alphabet. If the partition is hard (i.e., each sound belongs to only one group), the mapping is equivalent to “one-to-many” (“injective” mapping being a particular case when there is one sound per group). Soft partitions, such as fuzzy or overlapping partitions will pose additional problems. For example mining patterns in musical graphs where each node can have more than one label will result in a combinatorial explosion. A perhaps preferable approach is to consider different musical facets (e.g. pitch, timbre, amplitude ...) where hard partitions can be used to obtain a discrete symbol system, and use different grammars for each musical facet.

One question with regard to a partition of a database is how many symbols are desirable. While the answer to this question will depend on the application, it may be desirable to consider symbol hierarchies. One example of hierarchy could be the a general classification of musical instruments into families and subfamilies. This approach allows the extraction of patterns with different levels of detail. If patterns have to be found in a small amount of audio graphs, it may be easier to work at a higher level in the hierarchy. This generality can be called the “lexical level” of an alphabet. Many hierarchical clustering algorithms allow the construction of such hierarchies. Supervised approaches based on hierarchies of classifiers are also common in audio data mining.

THE *GRAPHEME* REPRESENTATION

As part of our ongoing research in collaborative composition systems, we have implemented a music representation based on the ideas exposed in this chapter. The system evolved from the graph representation used in freesound radio (http://radio.freesound.org), an online radio station based on sounds freesound.org where a genetic algorithm generates recombinations of sounds from the compositions created by users (Roma, 2008; Roma, Herrera and Serra, 2008). The composition program in freesound radio and the associated graph representation were developed with the perspective that the sound file sharing community could benefit from greater possibilites for collaborative composition based on the sounds from the database. While the graph representation had a general good reception, it lacked the modularity of the grammar approach. The current framework based on graph grammars makes it possible for users to share intermediate representations, and to exploit the generative aspect of grammars as a compositional aid.

Our current prototype consists in a music creation program based on sounds from the freesound database. This database currently holds more than 100.000 sounds. The interface allows a user to search for sounds in the database, with the help of a partition performed by a clusteirng algorithm. Selected sounds are added to a palette, from which they can be dragged to a composition canvas. The conposition interface allows the piece to be specified as a set of relationships between objects, rather than placing them in a predefined temporal grid. It is possible, though, to arbitrarily add silence nodes and to clip existing sounds as well to edit the start and ending points of an object. However, an important restriction with respect to the previous version is that cycles are not allowed, which, as discussed, makes subgraphs contractable. The user is presented an initial and final special nodes that facilitate the embedding of any graph as a node of another graph. Since cycles involving several nodes are not possible, the only way to create a loop is to effectively contract a subgraph. This, along with the space limitation of the canvas, forces the user to continuously define the groupings that are meaningful in the composition, and hence the compositional process is split in several structural levels.

Each of the collapsed graphs are stored in an XML file that describes structure of the composition, as well as an automatically rendered audio file and a bitmap file for representing the node in higher level compositions. The structure is divided betwen a “mappings” section and a “graph” section. The mappings section contains the lowest level lexical mappings, which refer to actual sound files (terminals) or other graphs (variables). In the first case th mapping may include segment boundaries. The graph section consists of a sequence of nodes and a sequence of edges. Each node refers to one of the mappings, and many nodes can refer to the same mapping. The surface of a complete piece can be obtained by recursively expanding all of the referenced subgraph using the embedding rules based on virtual start and end nodes, while collecting all of the terminal mappings.

Higher level lexical mappings are generated in the same format using information stored in a generic database used by the application. These mappings depend on partitions of the database computed by a clusteirng algorithm. Thus, for the same graph, we can derive multiple patterns at different levels of lexical generality, which allows us to detect common patterns among users. Some initial experiments with this system were described in (Roma and Herrera, 2010).

```
<xml>
    <audiograph name="4keys" author="simple machines">
        <maps>
            <audionodemap
                key="9185__melack__claus_2#66267_141725"
                url = "9185__melack__claus_2"
                start="66267"
                end="141725"
                icon = "9185__melack__claus_2.png">
        </maps>
        <graph start="0" end="1">
            <nodes>
                <node id="0" x="-9" y="250"/>
                <node id="1" x="970" y="250"/>
                <node id="2" x="180" y="252" map="9185__melack__claus_2#66267_141725"/>
                <node id="3" x="401" y="268" map="9185__melack__claus_2#66267_141725"/>
                <node id="4" x="604" y="277" map="9185__melack__claus_2#66267_141725"/>
                <node id="4" x="795" y="269" map="9185__melack__claus_2#66267_141725"/>
            </nodes>
            <edges>
                <edge source="2" target="3"/>
                <edge source="3" target="4"/>
                <edge source="4" target="5"/>
                <edge source="5" target="1"/>
                <edge source="0" target="2"/>
            </edges>
    </audiograph>
</xml>
```

Figure 4. Main elements of an editing session in audiograph: surface element (audiograph), lexical map (audionodemap), and extracted rules (audiographrule)

CONCLUSIONS AND FUTURE DEVELOPMENTS

Current practices in computer-aided music composition often imply the manipulation and organization of sound objects represented as digital audio files. In this context, representations based on traditional notation are limited for music created with digital technologies. Still, meaningful structural representations are needed for collaborative use of these technologies enabled by networks. We have shown that graph grammars provide a comprehensive framework for the representation of music structure independently from the actual sounds that are used. Grammars provide mechanisms for analysis and generation of music that can be used in networked applications with simple interfaces. As an extension of string grammars, graph grammars allow the manipulation of both sequential and parallel structures. Representation of musical graph grammars can be encoded in XML.

We have presented an example implementation of this approach. Still, many possibilities remain to be explored. The described language has originated from a specific application but aims to be generalizable. We plan to evaluate it in several applications related with popular practices such as rhythm programming or soundscape composition. Specific applications can benefit from

the definition of appropriate sound object ontologies. Finally, probabilistic grammars can be used when many example compositions are available.

REFERENCES

Ames, C. (1989) The markov process as a compositional model, a survey and tutorial. Leonardo Music Journal, 22 (2).

Casey, M. (2005) Acoustic lexemes for organizing internet audio. Contemporary music review 24 (6).

Chomsky,N. (1957), Syntactic structures. The Hague: Mouton.

Chion, M. (1983) Guide des objets sonores. Paris: Buchet/Chastel.

Cook, D. and Holder, L. B., (2000) Graph Data mining. IEEE Intelligent Systems, 15(2)

Cox, C., & Warner, D. (Ed). (2004), Audio Culture: Readings in Modern Music. New York: Continuum.

Engelfriet, J. and Rozenberg, G. (1997), Node replacement graph grammars. In Rozenberg, G. (Ed). Handbook of graph grammars and computing by graph transformation (pp 1-94). World Scientific Publishing Co. Singapore.

Engelfriet, J., and Rozenberg, G. (1990). Graph grammars based on node rewriting: An introduction to nlc graph grammars. In Graph-Grammars and Their Application to Computer Science (pp. 12–23). Bremen: Springer-Verlag.

Good (2001) MusicXML for notation and analysis. Helwet, W.B and Selfridge-Field, E. The Virtual Score. Cambridge: MIT Press.

Jackendoff, R and Lehrdal, F. (1981) Generative music theory and its relationship to psychology, Journal of Music Theory 25(1).

Kim, H., Moreau, N., and Sikora, T. (2006) MPEG-7 audio and beyond: audio content indexing and retrieval. The Atrium: Wiley and Sons

Lerdahl, F. & R. Jackendoff (1983). A Generative Theory of Tonal Music. Cambridge, Mass: MIT Press.

Madsen, T.(2003) Automatic discovery of Parallelism and Hierarchy in Music. Master thesis. University of Arhus.

Manjunath, B.S, Salembier, P., Sikora, T. (2002) Introduction to MPEG-7. The Atrium: Wiley and Sons.

Merker, B. (2006), Layered constraints on the multiple creativities of music In *Musical creativity, Multidisciplinary research in theory and practice.* Psychology Press, New York.

Pfaltz, J. and Rosenfeld, A. (1967) Web Grammars. Joint International Conference on Artificial Intelligence. Whashington D.C.

Roads, C. (1978). Composing Grammars. In Proceedings of the 1977 International Computer Music Conference. San Francisco.

Roads, C. (1979). Grammars as a Representation for Music. Computer Music Journal, 3(1)

Roads, C. (1982). An overview of music representations. Baroni,M. and Callegari, L. (Ed). Musical Grammars and Computer Analysis. Firenze: Leo S. Olschi.

Roma, G. (2008) Freesound Radio: supporting collective organization of sounds. Master Thesis. Universitat Pompeu Fabra. Barcelona.

Roma, G Herrera, P. and Serra, X. (2009) Freesound Radio: supporting music creation by exploration of a sound database. Presented at the Computational Creativity Support Workshop. CHI 2009. Boston.

Roma, G. and Herrera, P. (2010) Graph grammar representation for collaborative sample-based music creation. 5th Audio Mostly Conference

Rozenberg, G. (Ed) (1997) Handbook of graph grammars and computing by graph transformation. World Scientific Publishing Co.: Singapore

Schaeffer, P. (1966). Traité des Objets Musicaux. Paris: Seuil.

Schwartz, D. (2007). Corpus based concatenative synthesis. Signal Processing Magazine, IEEE, 24(2)

Zils, A. and Pachet, F. (2001) Musical mosaicing. Proceedings of the COST G-6 Conf. on Digital Audio Effects (DAFX-01), Limerick, Ireland