CAMM - Automatic Composer of Musical Melodies

Eloi Fernando Fritsch *
Rosa Maria Vicari 

Abstract

The purpose of this paper is to present the implementation of a grammar-based software named CAMM - Automatic Composer of Musical Melodies (or in Composer Automático de Melodias Musicais, in portuguese) - which is capable of generating melodies. CAMM has a set of rules that represent the musical knowledge needed to generate simple melodies in a limited and well-defined musical universe, i.e. style.

Musical parameters, such as notes, durations and intensities, are put together by means of a grammar in order to generate simple melodies which can be used for the composition of pieces of music.

1 Introduction

In order produce a system that uses rules to compose melodies (Miranda, 1990), it is necessary to study how music can be composed and the ways to compose, to arrange in musical harmony and to improve in music (Cope, 1987). Based on these compositional principles, it is possible to extract certain cognitive aspects of music composition with a computational approach. In this manner, we believe that it is possible to abstract from the musical universe certain aspects related to the task of melody composition.

We believe that musical models and structures represented in the listener's mind are made using three basic components:

- Melody
- Harmony
- Rhythm

In order to represent these models and structures in the computer it is necessary to define the rules that govern the relationship between their components. The musical language, from the computational point of view, must be treated as an organized symbol system (Rounds, 1965). This enables the creation of the grammar syntax and rules. From now on we will refer to these rules as components of a grammar.

CAMM was developed with the following objectives:

• to create musical grammars using the Prolog programming language to represent the musical knowledge;

---


**PhD in Computer and Electronic Engineering (Goiânia/Portugal). Areas of interest: Artificial Intelligence, Logic Programming. Universidade Federal do Rio Grande do Sul, Instituto de Informática - UFRGS, Curso de Pós-Graduação em Ciências da Computação - CPGCO, Av. Bento Gonçalves, 9500 Bloco IV - Agronomia - Campus do Vale, CEP 91501-970 - Porto Alegre - RS - Brasil. Caixa Postal: 15064 FAX: +55 (0)11 333-5576, E-mail: rosa@inf.ufrgs.br
2 HARMONY AND MUSICAL IMPROVISATION

Most popular music styles which involves improvisation, such as contemporary jazz, originated from blues. Thus we selected blues as the musical style to be studied in this work. From the harmonic point of view, a blues scale may be used very efficiently within the twelve bars, typically found in blues (Pogliori, 1983).

CAMM uses only one scale at a time to build a complete melody. The amount of variations that CAMM is able to generate is limited, taking into account the amount of possible combinations of the notes and of rhythmic figures that would be possible to do. On the one hand this limitation is good because it constrains the system to produce, let us say, only a small set of consistent melodies according to a simple grammar. On the other hand the output can get quite repetitive and loose musical interest.

3 BASIC ASPECTS OF IMPLEMENTATION

CAMM's grammar deals with four basic parameters of a melody:

- notes
- durations
- pauses
- intensities

We represent these musical parameters in textual form. Each statement has its corresponding MIDI code (Gomes, 1988). The distribution and grouping of parameters are arranged according to their use in the construction of musical phrases. For example: all the scales and all rhythmic cells are grouped in the same way. Other parameters are already completely represented in the knowledge base. They do not need to be linked to other parameters to mean something. Pauses, for example, do not need to be mapped to intensities nor to notes.

The DCG formalism (Definite Clause Grammar) and Prolog language are used to build the musical grammar (Arity, 1986). With the use of the DCG formalism it has been possible to write the grammar in a simple way (Vicari, 1992). The grammar provides means for the system to select, from a finite set, the parameters which generates a musical event.

CAMM still misses a bold graphic interface. For this reason the program input is directly made in the interpreter's command line, typing the appropriate command followed by parameters. The first parameter is the name of the scale to be used. The second parameter is the melody's tempo. The melody's tempo will be slower if it has time figures with bigger values and it will be faster if it has time figures with smaller values. The third parameter is the melody's intensity, i.e. how loud its notes should sound.

For example: printtema(a, lento, forte) means that the grammar will compose a melody using the A scale, with a slow tempo (lento) and that most of the notes have to be played very loud (forte).

4 THE NOTE PARAMETER

The way CAMM treats musical notes is based upon the way we believe pop music composers do (Fritsch, 1992) (see fig. 2). CAMM has a knowledge base that contains scales of blues. The user selects the scale that will be used to produce the melody. Each scale has a set of notes whose sequence is determined by a distribution function.

1 In a later step, these elements are gathered. For example: a musical note without duration and intensity can not be part of the context of the melody, but when this note is mapped to a duration value and to an intensity value, than it becomes to be part of the melody.
Once the user has selected a scale CAMM will manipulate the relation among its notes based on the harmonic field of the scale. We say that this harmonic field defines how notes arerealised to each other. The interaction between the user and CAMM is illustrated in Fig. 3. There is no deterministic mechanism which defines the order or how many times the same notes will occur in a melody because, as shown in Fig. 3, the purpose of CAMM is to present new options to the composer. Sometimes these will be very unpredictable indeed.

The repetition of the function that selects notes is determined by the sum of the durations which they are mapped to. When the twelve bars are filled with notes concatenated with its respective duration, then the function that selects notes will not be invoked anymore and the composition will be sent to system output.

The blues scales that compose the knowledge base are: C, F, Bb, Eb, Ab,Db, G♯, B, E, A, D, G.

As an example, we present an scale extracted from the CAMM knowledge base:

\[
\begin{align*}
\text{notes}&: \text{blues (c, [ nota (c1,60), nota (eb1,63), nota (f1,65), nota (gb1,66), nota (g1,67), nota (bb1,70), nota (c1,72)] })
\end{align*}
\]

The c, outside the square brackets, is a constant which indicates that the notes set is a C scale. All notes have the same chance of being chosen. Since they are seven notes, each one has a 14.3% chance of being selected. The decision of the octaved tone was made according to an aesthetic experiment made by the author. As shown in Fig. 5, the Prolog program can be altered, in order to increase or decrease the probability to generate specific notes (Roeder, 1989). Since the tonic represents a root sensation when it is played, it was set to be selected with more probability in order to create a good musical output.

If the knowledge base has to be altered, it is necessary that the music expert and the knowledge engineer interact. Ideally both should be the same person. The knowledge engineer's function is to represent the experience of the music expert, altering facts and rules already in CAMM.

As mentioned earlier, CAMM generates notes in textual and coded representation. For this reason, each note in the scale is associated to its decimal representation of the standard MIDI code (Yarrow, 1985). Hence, when the grammar generates the textual musical notes, it will also generate a set of MIDI messages for the synthesizer. The codification for flat notes and sharp notes are the same. This means that, for example, the code for a G♯ is the same that than that for a Gb, if they are in the same octave. The current octave is indicated by the number that follows the note, e.g. C1 corresponds to the central C of the piano, C2 one octave above the central C of the piano, and so on.

CAMM uses a distribution function to determine whether the events of the melody will be a note or a pause. However the probability for selecting a note is much superior than for selecting a pause. Otherwise the output would probably have only dispersed notes and a few individual short sequences which is not the characteristic of blues melodies.

5 THE DURATION PARAMETER

In CAMM, before the melody is outputted, its notes are mapped to duration values. Durations values are written textually along the program. A bar is constituted by paces and notes with their durations. Each bar generated by CAMM uses a default 4/4 rhythm. CAMM selects and fill a bar duration values according to the space still available after the previous selection.

\[\text{according to the author's musical taste.}\]
Depending on the tempo specified by the user, the duration values will be longer or shorter. If they are longer the melody will have a slower tempo and if they are shorter the melody will have a faster tempo.

There are two ways to provide duration values to notes: by selecting a value for each individual note or by selecting a rhythmic cell, i.e. a short rhythm pattern. In the current version of the program there is a probability of 60% that the system uses the latter method and 40% the the it uses the former method.

5.1 SELECTING DURATION VALUES

CAMM selects duration values according the tempo of the melody. Tempo is provided by the user. There are three options for tempo: slow tempo, medium tempo and fast tempo. See below an example of a set of duration values for the slow tempo.

\[
\text{fig.de tempo (}
\text{lento, } \text{semibreve, semibreve, semibreve, semibreve, semibreve,}
\text{semibreve, semibreve, semibreve, semibreve, minima, pontuada, minima, pontuada,}
\text{minima, pontuada, minima, pontuada, minima, pontuada, minima, pontuada,}
\text{minima, pontuada, minima, minima, minima, minima, minima, minima, minima,}
\text{minima, minima, minima, minima, minima, minima, minima, pontuada,}
\text{pontuada, semiminima, pontuada, semiminima, pontuada,}
\text{semiminima, semiminima, pontuada, semiminima, pontuada, semiminima, pontuada,}
\text{pontuada, colcheia, pontuada, colcheia, semicolcheia, pontuada,}
\text{semicolcheia, fusaz, pontuada, semifusaz)).}
\]

In the above example, extracted from CAMM’s knowledge base, we can observe that some duration values, or duration figures if you like, are repeated and that, for this reason, they will have a bigger probability to be selected. For this reason it is more likely that longer duration values will be selected more often, originating a slower tempo rather than a fast one.

CAMM’s grammar also features a mechanism for providing bars with duration figures. For each bar it considers the time space still available, the tempo specified by the user and what is available in the knowledge base in the terms of valid duration figures and rhythmic cells. As the available time space in a bar decreases CAMM selects those time figures which still fit. CAMM’s grammar was designed such that it avoids to attach very short durations to notes too often. This is not desired here specially when the melody’s tempo is to be slow. The same rule is also considered when the melody’s tempo is to be medium or fast. Notes with short duration (relative to the tempo of the melody) are avoided and notes with long duration are encouraged. This is so because we want to avoid beginning a bar with, let us say, long duration notes and drastically ending it with short notes.

Speaking in terms of numbers, the algorithm works as follows: If the total duration of a bar is other than 64 (which corresponds, let us say, to four quarter-notes) then it will try to select a suitable figure. In the case of a slow tempo, for example, in order to select figures that are not too short it tests if the difference between the total size of the bar and the space still available is bigger than or equal to 8 (eight-note). Eight is the value of an eight-note which is the smallest allowed duration figure for the rest of the bar. Following this rule, CAMM will only choose figures whose values are bigger than or equal to the eight-note to fill the bar. The same is valid for the medium tempo and for the fast tempo. The difference is that for a medium tempo the shortest value is 4 i.e. (a sixteenth note) and for the fastest tempo the shortest value is 2 (thirty-second note) (see Fig. 6 ).

5.2 RHYTHMIC CELLS

The rhythmic cells, or rhythmic patterns, are sets of pre-defined duration figures stored in CAMM’s knowledge base. In the current version the knowledge base has stored 75 different rhythmic
cells. These cells are different of each other and may be used for all. Each cell is also characterized for its own duration, which is the sum of the durations of its duration figures.

The statement below (one of the 75 rhythmic cells) obtains the set of rhythmic cells through the predicate andamento célula, according to the parameter supplied by the user. In this manner, the Con.celula variable receives the group of cells related to the referred tempo. The random_picks predicate randomly selects just one element from the list and passes this element to the dur.celula predicate, which computes the total size of the rhythmic cell. The Duracao.celula variable is instantiated with the total duration of the chosen cell. If the size of the chosen rhythmic cell is bigger than the available space left in a bar, then the system selects another cell that fits.

celulas.ritmicas(P1,P2,P3,Cont.) \[ \rightarrow \]
{ andamento.celula(P2,Conj.celula),
random_picks(Conj.celula,X), dur.celula(X,Duracao.celula),
Dur.Tot is Duracao.celula + Cont, Dur.Tot <= 94, }
celula.celula(X,P1,P3), continua.compas(P1,P2,P3,Dur.Tot).

6 THE PAUSE PARAMETER

Pauses are manipulating by the grammar in a similar way to notes (see Fig. 8). The difference is that notes are entities outside time at the moment of selection. Only afterwards a note is mapped to a duration value (which is also given by the grammar). On the other hand, pauses don’t need to be mapped to a duration value because their representation already includes an inherent duration value.

Pauses should not occur too frequently in a melody. This could produce to many gaps in the melody. Nevertheless, pauses should exist, in order to strengthen the rhythmic sensation of musical phrases.

7 THE INTENSITY PARAMETER

Another important aspect of a melody is the expressiveness given to its performance. This aspect has to do with the music dynamics, i.e., whether it is played in a loud, medium or quiet way. So CAMM also maps an intensity value to each note of the melody. The user also informs the system the intensity he or she wants. For example if the user wants a loud melody, then most of the intensity values will be within a loud bandwidth of values.

For a better representation of the intensity parameter with which the melody notes must sound, the intensity values are divided into groups that correspond to a particular kind of expression. For example, for loud melodies values within a bandwidth of loud intensity values will have a higher probability to be selected than any other values.

8 PRESENTATION OF THE GENERATED MELODY

Once CAMM creates a melody according to the parameters provided by the user, it can present it in two ways: textually and coded. Both outputs are displayed in the screen. Alternatively, melodies can also be saved in a MIDI file (Rattie, 1992).

The textual presentation was devised to ease the visualisation of the MIDI file and the notes played by the instrument. Through this textual presentation, any user, with a minimum understanding of music, can transcribe the output to traditional music notation when writing larger pieces of work by hand. This task could also be automatically accomplished by sequencer with music notation facilities.
Figure 8: Hence, the pauses are also chose by a random predicate.

We illustrate below a melody in its textual form. Here the user inputted: note A, slow tempo and loud intensity.

d1_minima_nf gl_minima_f / cl_minima_f ebl_minima_f / gl_seminima_pontuada_nf p_minima ebl_colcheia_f / cl_minima_f al_minima_f //
e1_minima_f ebl_colcheia_f di_seminima_f al_colcheia_f /
cl_minima_f di_minima_f //
ci_minima_f p_seminima ci_colcheia_f al_colcheia_f /
cl_minima_f ci_minima_f //
p_minima di_minima_nf //
al_minima_f gl_colcheia_f gl_seminima_f al_colcheia_f //
ci_seminima_pontuada_nf di_seminima_pontuada_f ci_seminima_pontuada_f
gl_seminima_nf al_seminusa_f ci_colcheia_f /
ebl_minima_f di_colcheia_f di_seminima_f al_colcheia_f //

Figure 9: Mapping between musical elements.

The file produced by the grammar has the following format:

< note > < duration > < intensity >

where each of the three elements are formed by three decimal numbers.

The Amount of time between a message NOTE ON and a message NOTE OFF is specified by a loop in the program, which executes a wait function, CAMM performs this loop as many times as required to delay the program's flow.

The following example illustrates how the program send data to the MIDI interface using the OUT command (of ATARI's GFA Basic):

OUT 3,144
OUT 3,060
OUT 3,127

These three commands send through the MIDI channel 1 (represented by the MIDI code 144) a note C (represented by the MIDI code 60) with maximum intensity i.e. MIDI velocity (represented by the MIDI code 127). The number three, in each command, is used because it addresses the computer's output port 3. Any MIDI synthesizer may be used to play the generated melodies. Any MIDI instrument that owns the MIDI IN and MIDI OUT ports may be used to reproduce the generated sounds.

9 MIDI IMPLEMENTATION

In the MIDI file, only the codification for note duration is not implemented according to the standard MIDI specification (Ratzen, 1992) (Yavelow, 1992). Our program that sends MIDI codes from files to the synthesizer uses a different technique for MIDI control, which is not aimed to be fully described in this paper (Korg, 1992). This program has been implemented in GPA Basic, running in a ATARI 1040 ST.
10 Conclusion

In this paper we introduced CAMM, a computer implementation of a grammar for automatic melody composition. In this work we wanted to show that the computer can compose interesting melodies using a simple grammar which defines their style. What is different in this work is that we use declarative programming for defining the grammar and for implementing the engine for melody composition. We believe that declarative programming is a very good way for communicating ideas to the computer. Rather than describing "how" the computer has to compose melodies, i.e., procedural programming, one needs only to describe "what" the machine has to do.

Although it is in its infancy, CAMM proved to be a good starting point for future developments. Perhaps the next step is to provide a machine learning mechanism for automatic construction of grammars either from a set of examples or from user interaction. Also we plan to devise an interface aimed for enabling easy communication between the musician and the computer. At the moment the user still has to master Prolog in order to edit the grammar.

We have been effectively using CAMM. A Porto Alegre pop band has stored several CAMM generated melodies in a MIDI sequencer which are triggered during the show. Here the computer acts as another musician in the stage.

REFERENCES


