CAMM - Automatic Composer of Musical Melodies

Eloi Fernando Fritsch [*]
Rosa Maria Viccari [†]

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 Compositor 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 improvise 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 strucutres in the computer it is necessary to define the rules that govern the relationship between their componentes. The musical language, from the computational point of view, must be treated as an organized symbol system (Roads, 1985). 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;

[*]MsC student in Computer Science (CPGCC/UFRGS), Graduated in Computer Science (UCS, 1991). Areas of interest: Artificial Intelligence & Music. Universidade Federal do Rio Grande do Sul, Instituto de Informática - UFRGS, Curso de Pós-Graduacao em Ciências da Computaçao - CPGCC, Av. Bento Gonçalves, 9500 Bloco IV - Agronomia - Campus do Vale, CEP 91501-970 - Porto Alegre - RS - Brazil, Caixa Postal: 15064 FAX: ++55 (051) 336-5576, E-mail: fritsch@inf.ufrgs.br

[†]PhD in Computers and Electronics Engeneering (Coimbra/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 - CPGCC, Av. Bento Gonçalves, 9500 Bloco IV - Agronomia - Campus do Vale, CEP 91501-970 - Porto Alegre - RS - Brazil, Caixa Postal: 15064 FAX: ++55 (051) 336-5576, E-mail: rosa@inf.ufrgs.br

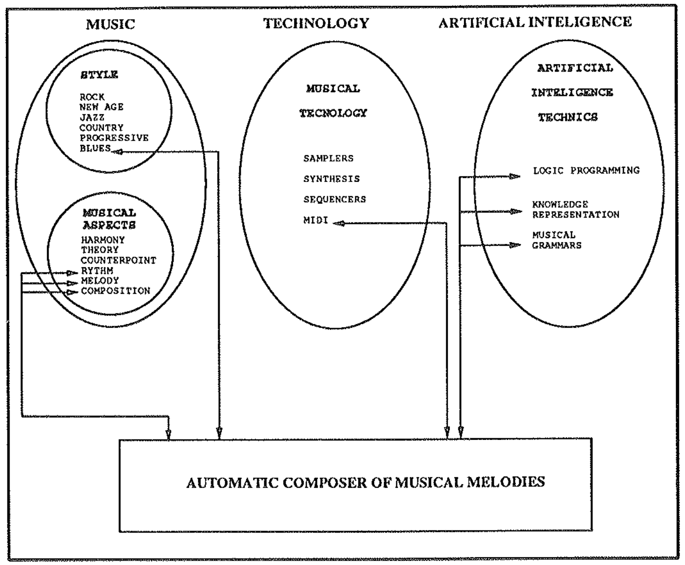

Figure 1: Representation of the universe of music, science and technology used to model CAMM

- to construct a musical program capable of using the MIDI system resources, such that the melody generated can be executed by an instrument and stored in a sequencer;
- to use the program in the composition of melodies where musical phrases generated by the program may be used by a human composer in a piece of music;
- to show that some melodies generated by a grammar written in Prolog may be artistically beautiful;

Although there are other softwares similar to CAMM, the system architecture, the heuristics and the grammar used in CAMM are slightly different. Among many musical systems used for automatic music composition we can mention here two systems which share similar characteristics with CAMM: EMI and the Computational Generation and Study of Jazz Music. The EMI system (Experiments in Music Intelligence), cretated by David Cope, also uses grammars. It is more complex though. It works with many other different musical aspects, such as the ability to manipulate intervals (Cope, 1987). In CAMM we limit the scope of the grammar to fewer parameters. The Computational Generation and Study of Jazz Music, by Francesco Giomi and Marco Ligabue, generates harmonic paths e improvises jazz melodies (Giomi, 1989). This system uses rhythmic cells similar to those of CAMM and also has more sofisticated functions. Nevertheless, CAMM approaches mainly melodic aspects, instead of harmonic ones and more, it generates blues melodies, and not jazz melodies.

As we are interested in the use of AI in music we selected a declarative programming style for implementation using Prolog. There are many other systems programmed in a declarative way, such as the Program for Music Segmentation, by John Roeder and ARTIST (for Artificial Intelligence-based Synthesis Tools) by Eduardo Miranda (Miranda, 1994). The former does not generate music though. Roeder's system was developed with pedagogic interest in the field of music analysis (Roeder, 1989). However it is very inspiring how Roeder uses a kind of grammar-orientated paradigm for music segmentation. The latter is a system that uses a kind of natural language (i.e. words in English) to communicate with a synthesiser aimed for producing sounds from qualitative, perceptually-orientated descriptions.

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 melodic point of view, a blues scale may be used very efficiently within the twelve bars, typically found in blues (Prediger, 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 ilimited, 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 contraints 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 this maner we can have musical knowledge bases that represent exactly the four parameters metioned above. [1] 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 DGC formalism (Definite Clause Grammar) and Prolog language are used to build the musical grammars (Arity, 1986). With the use of the DCG formalism it has been possible to write the grammar in a simple way (Viccari, 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. Since the melody is always built on a quaternary rhythm, what varies is the duration of notes. Therefore, 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 sequency 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 same note is mapped to a duration value and to a intensity value, than it becomes to be part of the melody.

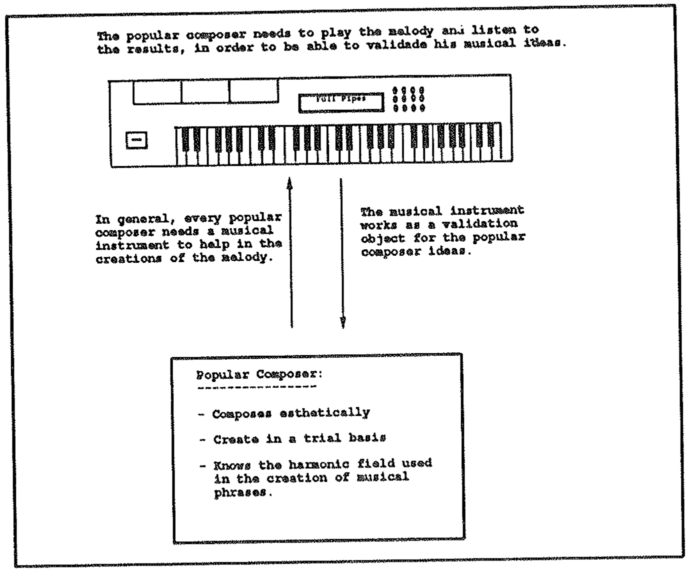
The popular composer needs to play the melody and listen to the results, in order to be able to validade his musical ideas.
Full Pipes
In general, every popular composer needs a musical instrument to help in the creations of the melody.
The musical instrument works as a validation object for the popular composer ideas.
Popular Composer:
- Composes esthetically
- Create in a trial basis
- Knows the harmonic field used in the creation of musical phrases.

Figure 2: An attempt to model the cognitive process of music creativity.

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 are realted 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 impredictable 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, F#, B, E, A, D, G.

Figure 4 shows the possible values for the first parameter of the grammar. Each of these scales is just a list of coded notes, treated as such by the program.

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

notas_de_blues (c, [nota (c1,60), nota (eb1,63), nota (f1,65), nota (gb1,66), nota (g1,67), nota (bb1,70), nota(c2,72)]).

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 for the octaved tonic was made accordig 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 rest sensation when

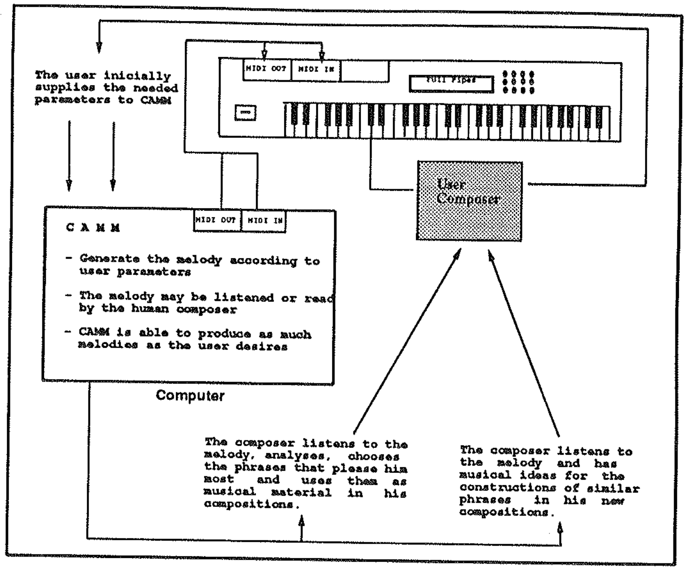
The user inicially supplies the needed parameters to CAMM
MIDI OUT
MIDI IN
Full Pipes
User Composer
C A M M
- Generate the melody according to user parameters
- The melody may be listened or read by the human composer
- CAMM is able to produce as much melodies as the user desires
Computer
The composer listens to the melody, analyses, chooses the phrases that please him most and uses them as musical material in his compositions.
The composer listens to the melody and has musical ideas for the constructions of similar phrases in his new compositions.

Figure 3: Process where CAMM provides melodic phrases, unexpected by the composer, which originate new musical ideas that, later on, will be transformed in new compositions.

it is played, it was set to be selected with more probability in order to create a good musical output [2].

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 (Yavelow92). Hence, when the grammar generates the textual musical notes, it will also generate a set of MIDI messages for the synthesiser. The codification for flat notes and sharp notes are the same. This means that, for example, the code for a F# is the same 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 - wich 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 textualy along the program.

A bar is constituted by pauses and notes with their durations. Each bar generated by CAMM uses by default a 4/4 rhythm. CAMM selects and fill a bar duration values according to the space still available after the previous selection.

[2] according to the authors's musical taste.

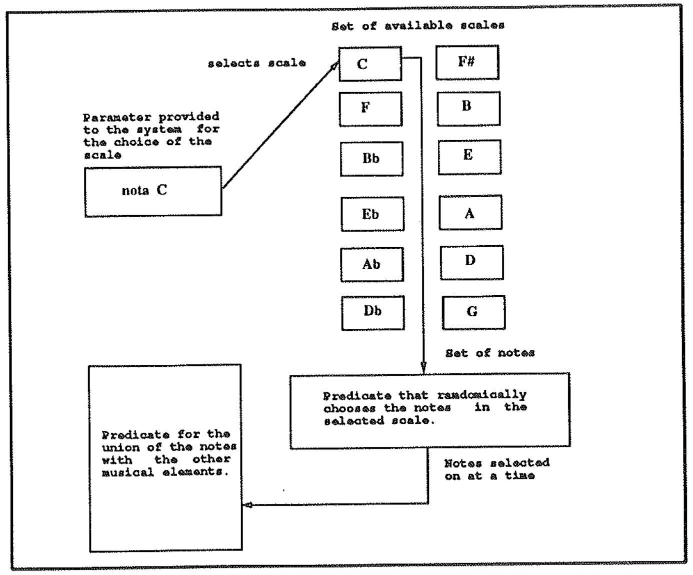
Set of available scales
selects scale
C
F#
F
B
Bb
E
Eb
A
Ab
D
Db
G
Parameter provided to the system for the choice of the scale
nota C
Set of notes
Predicate that randomically chooses the notes in the selected scale.
Predicate for the union of the notes with the other musical elements.
Notes selected on at a time

Figure 4: Selecting notes from a blue scale.

Depending on the tempo especified 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 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.

```
fig_de_tempo (
lento, [semibreve,semibreve,semibreve,semibreve,semibreve,
semibreve,semibreve,semibreve,semibreve,semibreve,
minima_pontuada,minima_pontuada,minima_pontuada,
minima_pontuada,minima_pontuada,minima_pontuada,
minima_pontuada,minima,minima,minima,minima,minima,
minima,minima,minima,minima,minima,minima,minima,minima,
minima,minima,minima,minima,minima,minima,minima,minima,
minima,minima,minima,seminima_pontuada,seminima_pontuada,
seminima_pontuada,seminima_pontuada,seminima,seminima,
colcheia_pontuada,colcheia,semicolcheia_pontuada,
semicolcheia,fusa_pontuada,semifusa]).
```

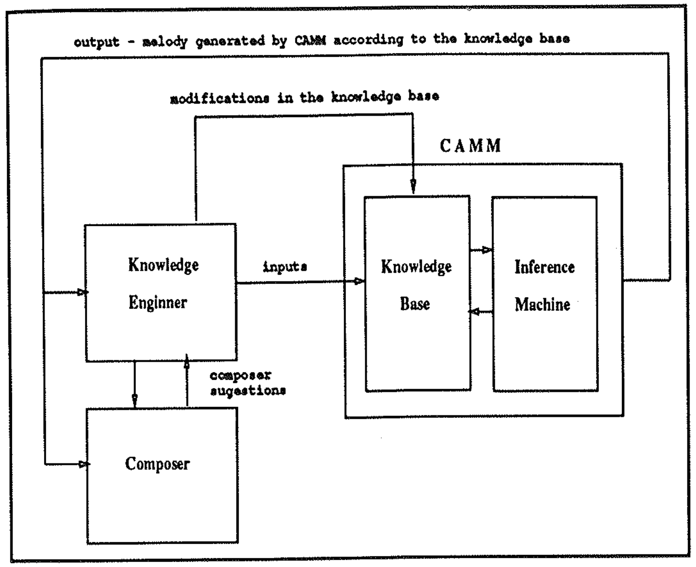
output - melody generated by CAMM according to the knowledge base
modifications in the knowledge base
CAMM
Knowledge Enginner
inputs
Knowledge Base
Inference Machine
composer sugestions
Composer

Figure 5: Knowledge Base modificiation schema, according to the needs of the composer

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 especified by the user and what is available in the knowledge base in the terms of valid duration figures and rhythmics 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 begining a bar with, let us say, long duration notes and dratically 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 fast 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

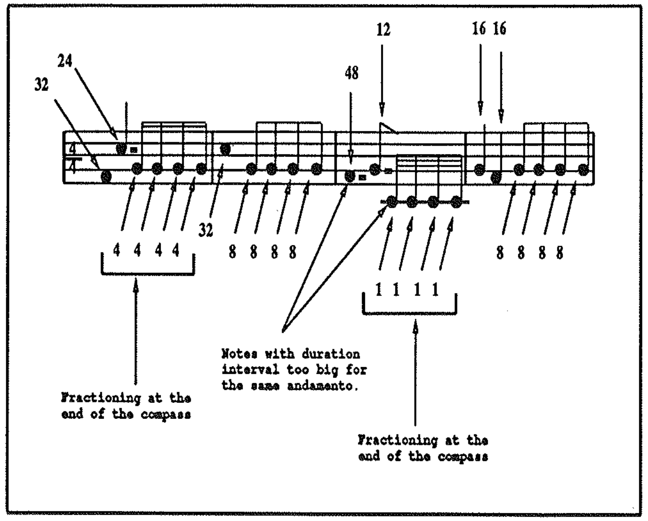
24
32
48
12
16 16
4 4 4 4
32
8 8 8 8
1 1 1 1
8 8 8 8
Notes with duration interval too big for the same andamento.
Fractioning at the end of the compass
Fractioning at the end of the compass

Figure 6: The problem of finding right the duration value for the end of the compass.

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 bellow (one of the 75 rhythmic cells) obtains the set of rhythmic cells through the predicate andamento_cel_rit, according to the parameter supplied by the user. In this manner, the Conj_cel_rit variable receives the group of cells related to the referred tempo. The random_pick predicate randomically chooses just one element from the list and passes this element to the dur_celular predicate, which computes the total size of the rhythmic cell. The Duracao_celular variable is instatiated with the total duration of the chosen cell. If the size of the chosen rhythmic cell is bigger then the available space left in a bar, then the system selects another cell that fits.

```
células_rítmicas(P1,P2,P3,Cont)->
{ andamento_cel_rit(P2,Conj_cel_rit),
random_pick(Conj_cel_rit,X), dur_celular(X,Duração_celular),
Dur_Tot is Duração_celular + Cont, Dur_Tot =< 64 },
célula_rit(X,P1,P3), continua_compas(P1,P2,P3,Dur_Tot).
```

6 THE PAUSE PARAMETER

Pauses are manipulated by the grammar in a similar way to notes (seeFig. 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 gammar). On the other hand, pauses don't need to be mapped

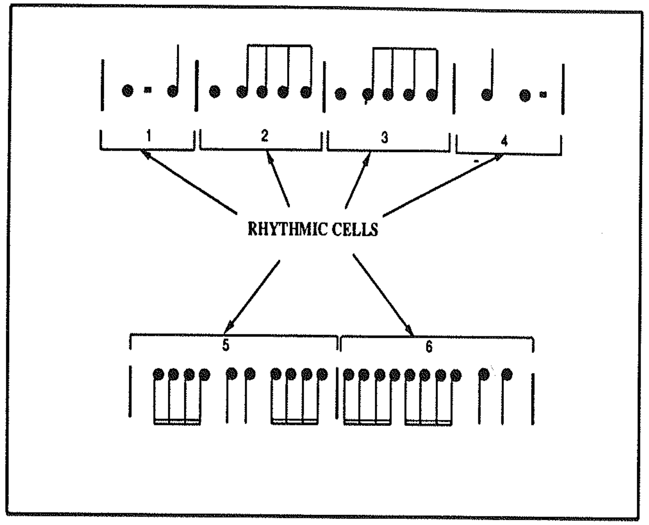

Figure 7: Rhythmic Cells

to a duration value because their representation already includes an inherent duration value.

Pauses should not accur too frequently in a melody. This could produce to many gaps in the melody. Nevertheless, pauses should exist, in order to strength 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 bandwidht 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 (Ratton, 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.

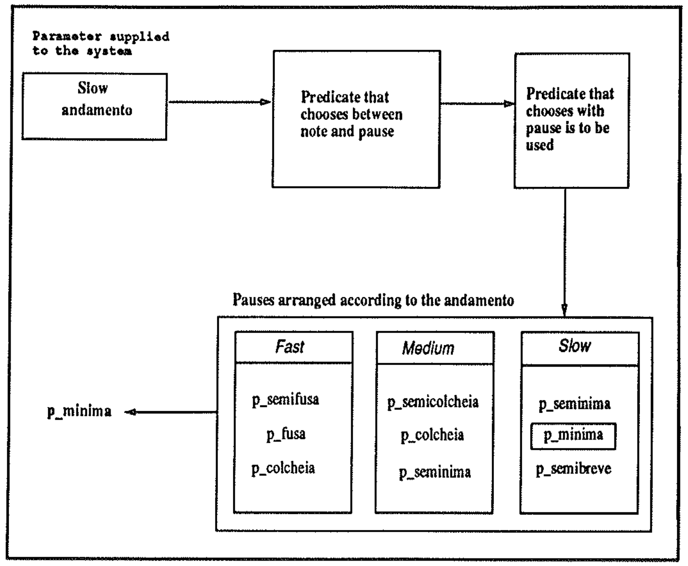
Parameter supplied to the system
Slow andamento
Predicate that chooses between note and pause
Predicate that chooses with pause is to be used
Pauses arranged according to the andamento
Fast
Medium
Slow
p_semifusa
p_fusa
p_colcheia
p_semicolcheia
p_colcheia
p_seminima
p_seminima
p_minima
p_semibreve
p_minima

Figure 8: Hence, the pauses are also chose by a randomic predicate.

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

```
d1_minima_mf g1_minima_f / c1_minima_f eb1_minima_f /
g1_seminima_pontuada_mf p_minima eb1_colcheia_f / c1_minima_f a1_minima_f //
e1_minima_f eb1_colcheia_f d1_seminima_f a1_colcheia_f /
c1_minima_f d1_minima_f //
c1_minima_f p_seminima c1_colcheia_f a1_colcheia_f /
c1_minima_f c1_minima_f //
p_minima d1_minima_mf //
a1_minima_f g1_colcheia_f g1_seminima_f a1_colcheia_f //
c1_seminima_pontuada_mf d1_seminima_pontuada_f c1_semicolcheia_pontuada_f
g1_semifusa_mf e1_semifusa_f c1_colcheia_f //
eb1_minima_f d1_colcheia_f d1_seminima_f a1_colcheia_f //
```

9 MIDI IMPLEMENTATION

In the MIDI file, only the codification for note duration is not implemented according to the standard MIDI specification (Ratton, 1992) (Yavelow, 1992). Our program that sends MIDI codes from files to the synthesiser 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 GFA Basic, running in a ATARI 1040 ST.

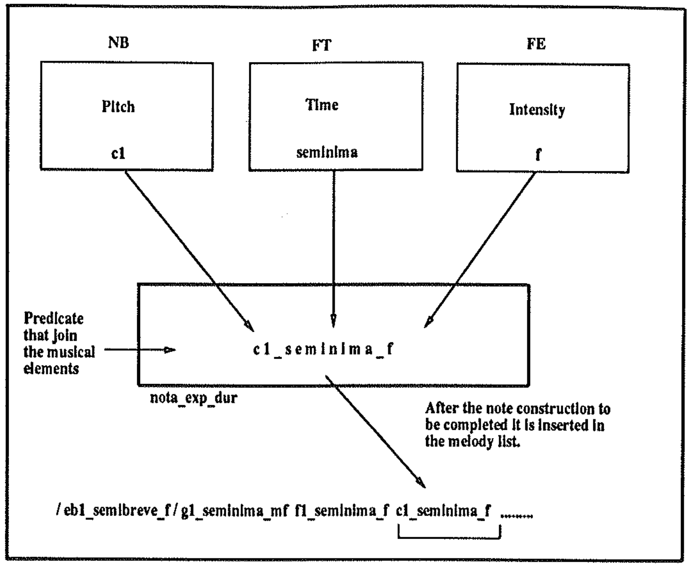
NB
FT
FE
Pitch
c1
Time
seminima
Intensity
f
Predicate that join the musical elements
c1_seminima_f
nota_exp_dur
After the note construction to be completed it is inserted in the melody list.
/eb1_semibreve_f/g1_seminima_mf f1_seminima_f c1_seminima_f ....

Figure 9: Mapping between musical elements.

The file produced by the grammar has the following format:

$$< \text{note} > < \text{duration} > < \text{intensity} >$$

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 synthesiser 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.

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 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 contruction 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 have 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

ADOLFO, Antônio. (1983) *O Livro do Músico:* Harmonia e Improvisação. Rio de Janeiro: Lumiar Editora.

ARITY Corporation. (1986) *The Arity/Prolog Programming Language.* ARITY Corporation.

COPE, David. (1987). An Expert System for Computer-Assisted Composition. Computer Music Journal, 11/4, 30-46.

FARIAS, Nelson (1963) *A Arte da Improvisação Para Todos os Instrumentos.* Rio de Janeiro,Lumiar Editora.

FRITSCH, Eloi Fernando. (1993) *Um Estudo Sobre Música & IA e a Implementação de um Sistema Especialista Teórico Musical.* (Trabalho individual 301). Porto Alegre: CPGCC da UFRGS.

GIOMI, Francesco. (1989) LIGABUE, Marco. *Computacional Generation and Study of Jazz Music..*

GOMES, Luis Carlos Elias (1988) *Som Três - Pequeno Dicionário MIDI.* São Paulo: Editora 8 Três.

KORG Incorporation. (1992) *Music Workstation: 01/W FD Owner's Manual* Tokyo, Japan: KORG Incorporation.

MIRANDA, Eduardo Reck. (1990) *Música e Inteligência Artificial Paradigmas e Aplicações.* Porto Alegre: CPGCC-UFRGS. (Trabalho Individual, 200)

MIRANDA, Eduardo Reck. (1994) *From Symbols to Sound:* AI Investigation of Sound Synthesis. in Contemporary Music Review, in press.

MOORER, J.A. (1975) *On the segmentation and analysis of continuous musical sound.* Stanford California: RepStan-m3, Dpto of Music, Stanford Univ.

PREDIGER, José Aluísio. (1993) *Blues, Harmonia e Improvisação Musical.* Porto Alegre: Prediger Academia, Notas de Aula.

RATTON, Miguel Balloussier. (1992) *MIDI: Guia Básico de Referência.* Rio de Janeiro: Editora Cam-

pus.

ROADS, Curtis. (1985), Research in music and artificial inteligence. *Computing Surveys*, Cambridge, v.17, n.2.

Roeder, John. (1989) *A Prolog Program for Music Segmentation* , School of Music, University of British Columbia, Musicus 1/ii, Dezembro.

YAVELOW, Christopher. (1992) *Music & Sound Biblle.* San Mateo,California,: IDG Books WorldWide, Inc.

VICCARI, Rosa. (1992) *Ferramentas para Inteligência Artificial.* Porto Alegre: CPGCC da UFRGS, Notas de Aula.