DISCOVERING EXPRESSIVE TRANSFORMATION RULES FROM SAXOPHONE JAZZ PERFORMANCES

Rafael Ramirez, Amaury Hazan, Emilia Gómez, Esteban Maestre, Xavier Serra
Music Technology Group, Pompeu Fabra University
Ocata 1, 08003 Barcelona, Spain, Tel:+34 935422165, Fax:+34 935422202
{rafael,ahazan,egomez,emaestre,xserra}@iua.upf.es

ABSTRACT

If-then rules are one of the most expressive and intuitive knowledge representations and their application to represent musical knowledge raises particularly interesting questions. In this paper, we describe an approach to learning expressive performance rules from monophonic recordings of Jazz standards by a skilled saxophonist. We have first developed a melodic transcription system which extracts a set of acoustic features from the recordings producing a melodic representation of the expressive performance played by the musician. We apply machine learning techniques, namely inductive logic programming, to this representation in order to induce first order logic rules of expressive music performance.

1. INTRODUCTION

Expressive performance is an important issue in music which has been studied from different perspectives (e.g. [10]). The main approaches to empirically study expressive performance have been based on statistical analysis (e.g. [28]), mathematical modelling (e.g. [32]), and analysis-by-synthesis (e.g. [8]). In all these approaches, it is a person who is responsible for devising a theory or mathematical model which captures different aspects of musical expressive performance. The theory or model is later tested on real performance data in order to determine its accuracy.

In this paper we describe an approach to investigate musical expressive performance based on inductive machine learning. Instead of manually modelling expressive performance and testing the model on real musical data, we let a computer use machine learning techniques [19] to automatically discover regularities and performance principles from real performance data: monophonic recordings of Jazz standards. In particular, we apply inductive logic programming to obtain first order logic rules.

The rest of the paper is organized as follows: Section 2 describes how symbolic features are extracted from the monophonic recordings. Section 3 describes how we apply machine learning techniques to some of the extracted symbolic features in order to induce rules of expressive music performance. Some induced rules are presented. Section 4 reports on related work, and finally Section 5 presents some conclusions and indicates some areas of future research.

2. MELODIC DESCRIPTION

In this section, we summarize how we extract a symbolic description from the monophonic recordings of performances of Jazz standards. We need this symbolic representation in order to apply machine learning techniques, in particular inductive logic programming techniques, to the data. In this paper, we are interested in modeling note-level transformations such as onset deviations, duration transformations, energy variations, and melody alterations. Thus, descriptors providing note-level information are of particular interest.

2.1. Algorithms for feature extraction

Figure 1 represents the steps that are performed to obtain a melodic description from audio. First of all, we perform a spectral analysis of a portion of sound, called analysis frame, whose size is a parameter of the algorithm. This spectral analysis lies in multiplying the audio frame with an appropriate analysis window and performing a Discrete Fourier Transform (DFT) to obtain its spectrum. In this case, we use a frame width of 46 ms, an overlap factor of 50%, and a Keiser-Bessel 25dB window. Then, we perform a note segmentation using low-level descriptor values. Once the note boundaries are known, the note descriptors are computed from the low-level and the fundamental frequency values.

2.2. Low-level descriptors computation

The main low-level descriptors used to characterize expressive performance are instantaneous energy and fundamental frequency.

2.2.1. Energy computation

The energy descriptor is computed on the spectral domain, using the values of the amplitude spectrum at each analysis frame. In addition, energy is computed in different

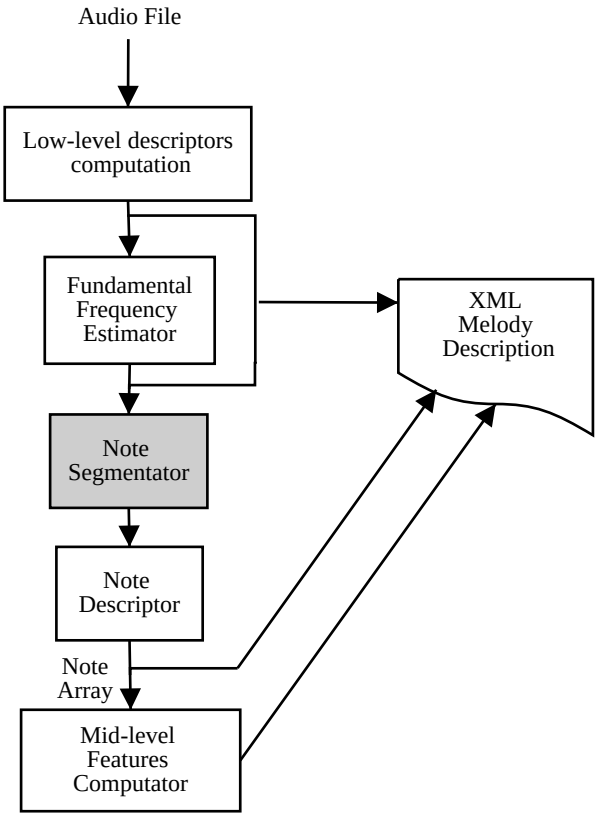

Figure 1. Block diagram of the melody descriptor

frequency bands as defined in [15], and these values are used by the algorithm for note segmentation.

2.2.2. Fundamental frequency estimation

For the estimation of the instantaneous fundamental frequency we use a harmonic matching model derived from the Two-Way Mismatch procedure (TWM) [17]. For each fundamental frequency candidate, mismatches between the harmonics generated and the measured partials frequencies are averaged over a fixed subset of the available partials. A weighting scheme is used to make the procedure robust to the presence of noise or absence of certain partials in the spectral data. The solution presented in [17] employs two mismatch error calculations The first one is based on the frequency difference between each partial in the measured sequence and its nearest neighbor in the predicted sequence. The second is based on the mismatch between each harmonic in the predicted sequence and its nearest partial neighbor in the measured sequence. This two-way mismatch helps to avoid octave errors by applying a penalty for partials that are present in the measured data but are not predicted, and also for partials whose presence is predicted but which do not actually appear in the measured sequence. The TWM mismatch procedure has also the benefit that the effect of any spurious components or partial missing from the measurement can be counteracted by the presence of uncorrupted partials in the same frame.

Figure 2 shows the block diagram for the fundamental frequency estimator following a harmonic-matching approach.

First, we perform a spectral analysis of all the windowed frames, as explained above. Secondly, the prominent spectral peaks of the spectrum are detected from the spectrum magnitude. These spectral peaks of the spectrum are defined as the local maxima of the spectrum which magnitude is greater than a threshold. The spectral peaks are compared to a harmonic series and a two-way mismatch (TWM) error is computed for each fundamental frequency candidates. The candidate with the minimum error is chosen to be the fundamental frequency estimate.

After a first test of this implementation, some improvements to the original algorithm where implemented to deal with some errors of the algorithm:

- Peak selection: a peak selection routine has been added in order to eliminate spectral peaks corresponding to noise. The peak selection is done according to a masking threshold around each of the maximum magnitude peaks. The form of the masking threshold depends on the peak amplitude, and uses three different slopes depending on the frequency distance to the peak frequency.

- Context awareness: we take into account previous values of the fundamental frequency estimation and instrument dependencies to obtain a more adapted

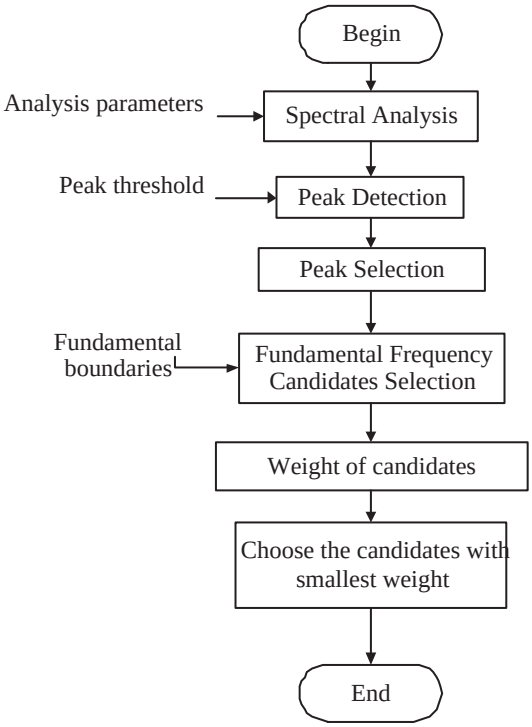

Figure 2. Flow diagram of the TWM algorithm

result.

- Noise gate: a noise gate based on some low-level signal descriptor is applied to detect silences, so that the estimation is only performed in non-silent segments of the sound.

2.3. Note segmentation

Note segmentation is performed using a set of frame descriptors, which are energy computation in different frequency bands and fundamental frequency. Energy onsets are first detected following a band-wise algorithm that uses some psycho-acoustical knowledge [15]. In a second step, fundamental frequency transitions are also detected. Finally, both results are merged to find the note boundaries (onset and offset information).

2.4. Note descriptor computation

We compute note descriptors using the note boundaries and the low-level descriptors values. The low-level descriptors associated to a note segment are computed by averaging the frame values within this note segment. Pitch histograms have been used to compute the pitch note and the fundamental frequency that represents each note segment, as found in [18]. This is done to avoid taking into account mistaken frames in the fundamental frequency mean computation.

First, frequency values are converted into cents, by the following formula:

$$c = 1200 \cdot \frac{\log\left(\frac{f}{f_{ref}}\right)}{log2} \quad (1)$$

where $f_{ref} = 8.176$. Then, we define histograms with bins of $100\ cents$ and hop size of $5\ cents$ and we compute the maximum of the histogram to identify the note pitch. Finally, we compute the frequency mean for all the points that belong to the histogram. The MIDI pitch is computed by quantization of this fundamental frequency mean over the frames within the note limits.

2.5. Implementation

All the algorithms for melodic description have been implemented within the CLAM framework [1]. They have been integrated within a tool for melodic description, *Melodia*. Figure 3 presents a screenshot of the melodic description tool.

3. LEARNING EXPRESSIVE PERFORMANCE RULES IN JAZZ

In this section, we describe our inductive approach for learning expressive performance rules from performances of Jazz standards by a skilled saxophone player. Our aim is to find rules which predict, for a significant number of cases, how a particular note in a particular context should be played (e.g. longer than its nominal duration) or how

[1] http://www.iua.upf.es/mtg/clam

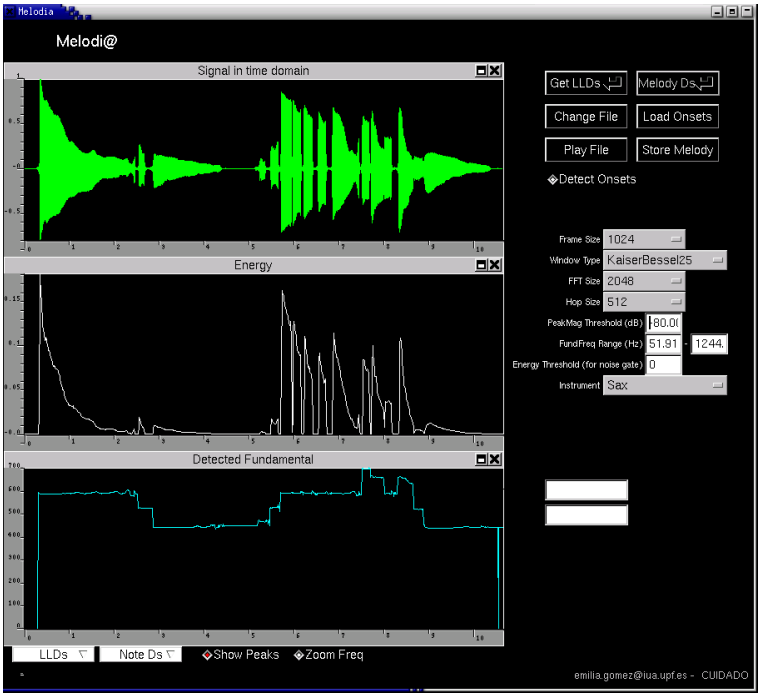

Figure 3. Tool for melody description

a melody should be altered by inserting or deleting notes. We are aware of the fact that not all the expressive transformations performed by a musician can be predicted at a local note level. Musicians perform music considering a number of abstract structures (e.g. musical phrases) which makes of expressive performance a multi-level phenomenon. In this context, our aim is to obtain an integrated model of expressive performance which combines note-level rules with structure-level rules. As a first step in this direction, we have based our musical analysis on the *implication/realization model*, proposed by Narmour [21, 22].

3.1. Musical analysis

The Implication/Realization model is a theory of perception and cognition of melodies. The theory states that a melodic musical line continuously causes listeners to generate expectations of how the melody should continue. The nature of these expectations in an individual are motivated by two types of sources: innate and learned. According to Narmour, on the one hand we are all born with innate information which suggests to us how a particular melody should continue. On the other hand, learned factors are due to exposure to music throughout our lives and familiarity with musical styles and particular melodies. According to Narmour, any two consecutively perceived notes constitute a melodic interval, and if this interval is not conceived as complete, it is an *implicative interval*, i.e. an interval that implies a subsequent interval with certain characteristics. That is to say, some notes are more likely than others to follow the implicative interval. Two main principles recognized by Narmour concern *registral direction* and *intervallic difference*. The principle of registral direction states that small intervals imply an interval in the same registral direction (a small upward interval implies another upward interval and analogously for downward intervals), and large intervals imply a change in registral direction (a large upward interval implies a downward interval and analogously for downward intervals). The principle of intervallic difference states that a small (five semitones or less) interval implies a similarly-sized interval (plus or minus 2 semitones), and a large interval (seven semitones or more) implies a smaller interval. Based on these two principles, melodic patterns or groups can be identified that either satisfy or violate the implication as predicted by the principles. Such patterns are called structures and are labeled to denote characteristics in terms of registral direction and intervallic difference. Figure 5 shows prototypical Narmour structures. A note in a melody often belongs to more than one structure. Thus, a description of a melody as a sequence of Narmour structures consists of a list of overlapping structures. Grachten [12] developed a parser for melodies that automatically generates an implication/realization analysis. Figure 6 shows the analysis for a fragment of *Body and Soul*.

3.2. Training data

The training data used in our experimental investigations are monophonic recordings of four Jazz standards (*Body*

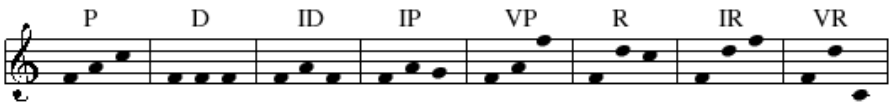

Figure 4. Basic Narmour I/R melodic units

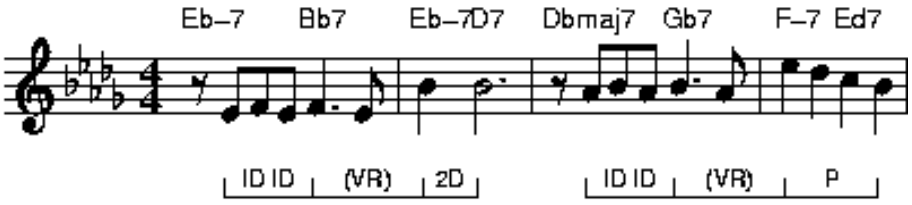

Figure 5. Narmour structure parsing of the first phrase of *Body And Soul*

and Soul, *Once I Loved*, *Like Someone in Love* and *Up Jumped Spring*) performed by a professional musician at 11 different tempos around the nominal tempo. For each piece, the nominal tempo was determined by the musician as the most natural and comfortable tempo to interpret the piece. Also for each piece, the musician identified the fastest and slowest tempos at which a piece could be reasonably interpreted. Interpretations were recorded at regular intervals around the nominal tempo (5 faster and 5 slower) within the fastest-slowest tempo limits. The data set is composed of 4360 performed notes. Each note in the training data is annotated with its corresponding class and a number of attributes representing both properties of the note itself and some aspects of the context in which the note appears. Information about the note include note duration and the note metrical position within a bar, while information about its melodic context include information on neighboring notes as well as the Narmour group(s) to which the note belongs.

3.3. Learning task

In this paper, we are concerned with note-level expressive transformations, in particular transformations of note duration, onset and energy, as well as melody alterations. The set of the attributes we consider for each note includes the note's nominal duration, the duration of previous and following notes, the extension of the pitch intervals between the note and the previous and following notes, the Narmour groups the note belongs to, and the tempo at which the note is played.

The performance classes that interest us are *lengthen*, *shorten* and *same* for duration transformation, *advance*, *delay*, and *same* for onset deviation, *soft*, *loud* and *same* for energy, and *consolidation*, *ornamentation* and *none* for note alteration. A note is considered to belong to class *lengthen*, if its performed duration is 20% longer (or more) that its nominal duration, e.g. its duration according to the score. Class *shorten* is defined analogously. A note is considered to be in class *advance* if its performed onset is 5% of a bar earlier (or more) than its nominal onset. Class *delay* is defined analogously. A note is considered to be in class *loud* if it is played louder than its predecessor and louder then the average level of the piece. Class *soft* is defined analogously. We decided to set these boundaries after experimenting with different ratios. The main idea was to guarantee that a note classified, for instance, as lengthen was purposely lengthened by the performer and not the result of a performance inexactitude.

We also consider transformations consisting of alterations to the melody (as specified in the score) by introducing or suppressing notes. These transformations play a fundamental role in Jazz expressive interpretations and, in contrast to classical music, are not considered as interpretation errors. We have categorized these transformations as *consolidations*, *fragmentations* and *ornamentations*. A *consolidation* represents the agglomeration of multiple score notes into a single performed note, a *fragmentation* represents the performance of a single score note as multiple notes, and an *ornamentation* represents the insertion of one or several short notes to anticipate another performed note. However, in our data set the number of fragmentation examples is insufficient to be able to obtain reliable general rules. Hence, we restrict the classes we consider to *consolidation* and *ornamentation* (and *none* to denote the fact there is no alteration at all).

3.4. Learning algorithm

Using the training data we are interested in inducing rules of expressive performance. Typical examples of *if-then* rules used in machine learning are classification rules and association rules. Classification rules are a popular approach to classification learning in which the *antecedent* of a rule is generally a conjunction of tests (e.g. $dur = eight \wedge pitch = 65$) and the *consequent* is the predicted class (e.g. $performed = lengthen$) or classes (possibly with a probability associated to each class). Association rules [4] are similar to classification rules except that they can predict any attribute or combinations of at-

tributes, not just the class. It is often assumed implicitly that the conditions in (classification and association) rules involve testing an attribute value against a constant. Such rules are called *propositional* because they have the same expressive power as *propositional logic*. In many cases, propositional rules are sufficiently expressive to describe a concept accurately. However, there are cases, as it is ours, where more expressive rules would provide a more intuitive concept description. These are cases where the knowledge to be learned is best expressed by allowing variables in attributes (e.g. $duration(X, lengthen)$) and thus in rules (e.g. *if succ(X,Y) then duration(Y,lengthen)*). One important special case involving learning sets of rules containing variables is called *inductive logic programming* [29, 23]. A type of rules in inductive logic programming, called Horn clauses, are of particular interest because rules in this form can be interpreted and directly executed as a program in the logic programming language Prolog [5]. Thus, an algorithm capable of learning such rule sets may be viewed as an algorithm for automatically inferring Prolog programs from examples. A variety of algorithms that directly learn Horn clauses have been proposed. In this paper we apply a standard inductive logic programming sequential covering algorithm that incrementally constructs a theory as a set of first-order rules (Horn clauses) by learning new rules one at a time, removing the positive examples covered by the latest rule before attempting to learn the next rule. Before describing the algorithm in more detail, let us introduce some terminology from formal logic.

Logic expressions are composed of *constants* (e.g. 5, peter), *variables* (e.g. X, Y), *predicate symbols* (e.g. p, duration), and *function symbols* (e.g. sum). The difference between predicates and functions is that predicates take on values of *true* or *false*, whereas functions may take on any constant as their value. Following Prolog syntax, we will use lowercase for predicates, functions and constants, and uppercase for variables. A *term* is any constant, any variable, or any function applied to any term (e.g. 5, X, f(peter)). A *literal* is any predicate (called *positive literal*)or its negation (called *negative literal*) applied to any term. A Horn clause is a logic formula of the form:

$$H \leftarrow (L_1 \wedge \cdots \wedge L_n) \qquad (2)$$

where $H,\ L_1, \ldots, L_n$ are positive literals. (2) is equivalent to $if L_1, \wedge \cdots \wedge L_n, then H$. The Horn clause precondition $L_1, \wedge \cdots \wedge L_n$ is called the clause *body* and the literal H is called the clause *head*.

The sequential covering algorithm we applied in the research reported in this paper is specified as follows:

```
SeqCovAlgo(Target_predicate, Predicates, Examples)
Pos = Examples for which the Target_predicate is true
Neg = Examples for which the Target_predicate is false
Learned_rules = {}
While Pos do
     NewRule = rule that predicts Target_predicate
                                with no precondition
     NewRuleNeg = Neg
     While NewRuleNeg do
          Candidate_literals = Candidate new literals
                       for NewRule, based on Predicates
          Best_literal = argmax Gain(L,NewRule) where
                         L is in Candidate_literals
          add Best_literal to the preconditions of NewRule
          NewRuleNeg = Examples in NewRuleNeg which
                       satisfy NewRule preconditions
      Learned_rules = Learned_rules + NewRule
      Pos = Pos - {members of Pos covered by NewRule}
Return Learned_rules
```

The outer loop learns new rules one at a time, removing the positive examples covered by the latest rule before attempting to learn the next rule. The inner loop performs a general-to-specific search through the space of possible rules in search of a rule with high accuracy. At each iteration, the outer loop adds a new rule to its disjunctive hypothesis, Learned_rules. The effect of each new rule is to generalize the current disjunctive hypothesis (i.e. increasing the number of instances it classifies as positive) by adding a new disjunct. At this level, the search is a specific-to-general search starting with the most specific hypothesis (i.e. the empty disjunction) and terminating when the hypothesis is sufficiently general to cover all training examples. The inner loop performs a finer-grained search to determine the exact definition of each new rule. At this level, the search is a general-to-specific search beginning with the most general rule, i.e. the rule with empty body, then adding literals one at a time to specialize the rule until it avoids all negative examples.

A critical decision in the algorithm is how to select the best literal (Best_literal) from the set of candidate literals (Candidate_literals). The evaluation function to estimate the utility of adding a new literal is based on the numbers of positive and negative bindings covered before and after adding the new literal.

$$Gain(L, R) = p(log_2(p_1/p_1{+}n_1){-}log_2(p_0/p_0{+}n_0))$$

where p_0 and n_0 are the number of positive and negative bindings of rule R, respectively, and p_1 and n_1 are the number of positive and negative bindings of the rule R' resulting by adding literal L to R, respectively. p is the number of positive bindings of rule R that are still covered after adding L to R.

Note that the algorithm considers two-class classification problems whereas our classification task involves three classes, e.g. in the case of duration the classes are *shorten*, *lengthen* and *same*. We have reduced our problem to a two-class classification problem by taking the examples of one class as positive examples and the examples of the other two classes as negative examples.

We have applied the learning algorithm with the following target predicates: `duration/3`, `onset/3`, `energy/3`, and `alteration/3` (where `/n` at the end

of the predicate name referes to the predicate arity, i.e. the number of arguments the predicate takes). Each target predicate corresponds to a particular type of transformation: `duration/3` refers to duration transformation, `onset/3` to onset deviation, `energy/3` to energy transformation, and `alteration/3` refers to note alteration.

For each target predicate we use as example set the complete training data specialized for the particular type of transformation, e.g. for `duration/3` we used the complete data set information on duration transformation (i.e. the performed duration transformation for each note in the data set). The arguments are the musical piece, the note in the piece and performed transformation.

We use (background) predicates to specify both note musical context and background information. The predicates we consider include `context/6`, `narmour/2`, `succ/2` and `member/3`. Predicate `context/8` specifies the local context of a note. Its arguments are: note identifier, note's nominal duration, duration of previous and following notes, extension of the pitch intervals between the note and the previous and following notes, and tempo at which the note is played. Predicate `narmour/2` specifies the Narmour groups to which the note belongs. Its arguments are the note identifier and a list of Narmour groups. Predicate `succ(X,Y)` means `Y` is the successor of `Y`, and Predicate `member(X,L)` means `X` is a member of list `L`. Note that `succ(X,Y)` also mean `X` is the predecessor of `Y`. The `succ(X,Y)` predicate allows the specification of arbitrary-size note-context by chaining a number of successive notes:

$$succ(X_1, X_2), succ(X_2, X_3), \ldots, succ(X_{n-1}, X_n)$$

where X_i ($1 \leq i \leq n$) is the note of interest.

Inductive logic programming has proved to be an extremely well suited technique for learning expressive performance rules. This is mainly due to three reasons: Firstly, inductive logic programming allows the induction of first order logic rules. First order logic rules are substantially more expressive than the traditional propositional rules used in most rule learning algorithms (e.g. the widely used C4.5 algorithm [24]). Secondly, Inductive logic programming allows considering an arbitrary-size note context without explicitly defining extra attributes. Finally, the possibility of introducing background knowledge into the learning task provides great advantages in learning musical concepts.

One way to see the expressive power of the rules obtained by applying inductive logic programming is to consider the programming language Prolog [5]. In Prolog, programs are sets of first order logic rules like the induced expressive performance rules we describe in the next section. In fact, the induced rules form a valid Prolog program. Thus, an algorithm capable of learning such rule sets may be viewed as an algorithm for automatically inferring Prolog programs from examples.

3.5. Induced rules

The induced rules are of different types. Some focus on features of the note itself and depend on the performance tempo while others focus on the Narmour analysis and are independent of the performance tempo. Rules referring to the local context of a note, i.e. rules classifying a note solely in terms of the timing, pitch and metrical strength of the note and its neighbors, as well as compound rules that refer to both the local context and the Narmour structure were discovered. In order to exemplify the discovered rules we present some of them below.

STRETCH RULES

S-1: [Pos cover = 12 Neg cover = 1]

```
duration(A, C,lengthen) :-
     succ(C,D),
     context(A,B,D,4,-1,-1,0,-1,1,nominal).
```

"Lengthen a note at an offbeat position if its successor is a quarter between two shorter notes, the former one being at the same pitch, the next one being lower, at a nominal tempo"

S-2: [Pos cover = 21 Neg cover = 1]

```
duration(A, C, shorten) :-
     succ(C, D), succ(D, E),
     narmour(A, E, [nargroup(p, 1)|F]),
     narmour(A, C, [nargroup(p, 2)|F]).
```

"Shorten a note n if it belongs to a P Narmour group in second position and if note n+2 belongs to a P Narmour group in first position"

S-3: [Pos cover = 41 Neg cover = 1]

```
duration(A, C, same) :-
     succ(C, D), succ(D, E),
     narmour(A, E, [nargroup(vr, 3)|F]),
     member(nargroup(p, 1), F).
```

"Do not stretch a note n if note n+2 belongs to both VR Narmour group in third position and P Narmour group in first position "

ONSET DEVIATION RULES

O-1: [Pos cover = 41 Neg cover = 2]

```
onset(A, C, same) :-
     succ(C, D),
     narmour(A, D, [nargroup(vr, 3)|E]),
     member(nargroup(d, 1), E).
```

"Play a note at the right time if its successor belongs to a VR Narmour group in third position and to a D Narmour group in first position"

O-2: [Pos cover = 10 Neg cover = 1]

```
onset(A, C, delay) :-
     succ(D, C),
     narmour(A, D, [nargroup(id, 3)|E]),
     member(nargroup(ip, 2), E).
```

"Play a note n with delay if its predecessor belongs to a ID Narmour group in third position and to a IP Narmour group in second position"

O-3: [Pos cover = 17 Neg cover = 1]
```
onset(A, C, advance) :-
     succ(C,D), succ(D,E),
     narmour(A,E,[nargroup(ip,1)|F]),
     narmour(A,D,[nargroup(p,3)|F]).
```
"Play a note n in advance if n+1 belongs to a P Narmour group in third position and if n+2 belongs to an IP Narmour group in first position"

O-4: [Pos cover = 3 Neg cover = 0]
```
onset(A, C, advance) :-
     context(A,C,6,0,0,1,1,0,slow),
     narmour(A, C, [nargroup(p, 3)|D]).
```
"In slow interpretations, play a triplet in advance if it is between two higher triplets, if it is neither in a beat position nor an offbeat position, and if it belongs to a P Narmour group in third position"

ENERGY RULES

E-1: [Pos cover = 26 Neg cover = 0]
```
energy(A, C, loud) :-
     succ(D, C),
     narmour(A, D, [nargroup(d, 2)|E]),
     narmour(A, C, [nargroup(id, 1)|E]).
```
"Play loudly a note if it belongs to an ID Narmour group in first position and if its predecessor belongs to a D Narmour group in second position"

The following two rules go together. The examples covered by the second one are exactly the successors of the ones covered by the first one.

E-2a: [Pos cover = 34 Neg cover = 1]
```
energy(A, C, soft) :-
     succ(C, D),
     narmour(A, D, [nargroup(p, 4)|E]),
     context(A, C, [nargroup(p, 3)|E]).
```
E-2b: [Pos cover = 34 Neg cover = 1]
```
energy(A, C, soft) :-
     succ(D, C),
     narmour(A, D, [nargroup(p, 3)|E]),
     narmour(A, C, [nargroup(p, 4)|E]).
```
"Play softly two successive notes if they belong to a P Narmour group respectively in third and fourth position"

E-3: [Pos cover = 19 Neg cover = 0]
```
energy(A, C, loud) :-
     succ(D, C),
     context(A,D,8,0,0,-1,1,2,nominal).
```
"At nominal tempo, play loudly an eight between two eights if it is on second or forth bar beat and if the 3 notes form a regular ascending scale"

E-4a: [Pos cover = 30 Neg cover = 2]
```
energy(A, C, same) :-
     narmour(A, C, [nargroup(ip, 1)|D]).
```
E-4b: [Pos cover = 34 Neg cover = 2]
```
energy(A, C, same) :-
     succ(D, C),
     narmour(A, D, [nargroup(ip, 1)|E]).
```
"Play two notes at a normal level if the first one belongs to an IP Narmour group in first position"

ALTERATION RULES

A-1: [Pos cover = 232 Neg cover = 0]
```
alteration(A, C, none) :-
     narmour(A, C, [nargroup(p, 2)|D]).
```
"Do not perform alteration of a note if it belongs to a P Narmour group in second position"

A-2: [Pos cover = 8 Neg cover = 0]
```
alteration(A, C, ornamentation) :-
     succ(C, D),
     narmour(A, D, [nargroup(d, 2)|E]),
     member(nargroup(ip, 1), E),
     narmour(A, C, [nargroup(vr, 3)|F]).
```
"Ornamentate a note if it belongs to a VR Narmour group in third position and if its successor belongs to D Narmour group in second position and to IP Narmour group in first position"

A-3: [Pos cover = 14 Neg cover = 1]
```
alteration(A, C, consolidation) :-
     succ(C, D),
     narmour(A, D, [nargroup(D, 2)|E]),
     narmour(A, C, [nargroup(D, 2)|E]).
```
"Consolidate a note n with note n+1 if note n belongs to a D Narmour group in second position and note n+1 belongs to an D Narmour group in second position"

3.6. Rule interpretation

Some of the induced rules seem to capture intuitive performance principles while others seem to be of less musical interest or even counter-intuitive or surprising. An example of the former rules is Rule E-1 which states that a note (marked with a dot in Figure 6) is to be played louder if it appears in the musical context in Figure 6 (Narmour groups D and ID are shown in Figure 4):

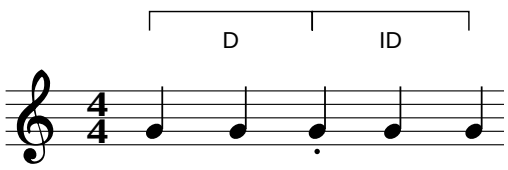

Figure 6. Musical context of note described by Rule E-1

This is, a note is to be played louder if the two preceding notes have the same pitch and its intervals with the next two notes are X and Y, respectively, where X is a small interval (i.e. less or equal than five semitones) and $X > Y$. Figure 7 shows two instances of this situation.

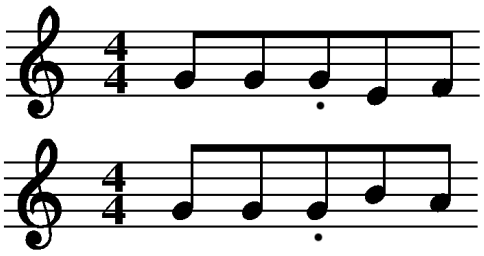

Figure 7. Two instances of the context of note described by Rule E-1

The idea of stressing the last note (e.g. by playing it louder) of a unison interval sequence seems to be intuitive specially if the note is in a strong metrical position. In our training data, the 26 positive examples covered by the rule satisfy that the stressed note appears in a strong metrical position. We plan to extend our training data with examples in which the note appears in different metrical positions in order to induce rules which take into account the note metrical strength.

Figure 8 shows the musical context of the note captured by the rules S-2, S-3, O-1, O-2, O-3, E-2a, A-1, A-2 and A-3, in which the note was performed shorter, same duration, same onset, delayed, advanced, softer, not ornamented, ornamented, and consolidated, respectively.

4. RELATED WORK

Previous research in learning sets of rules in a musical context has included a broad spectrum of music domains. The most related work to the research presented in this paper is the work by Widmer [33, 34]. Widmer has focused on the task of discovering general rules of expressive classical piano performance from real performance data via inductive machine learning. The performance data used for the study are MIDI recordings of 13 piano sonatas by W.A. Mozart performed by a skilled pianist. In addition to these data, the music score was also coded. The resulting substantial data consists of information about the nominal note onsets, duration, metrical information and annotations. When trained on the data an inductive rule learning algorithm discovered a small set of 17 quite simple classification rules that predict a large number of the

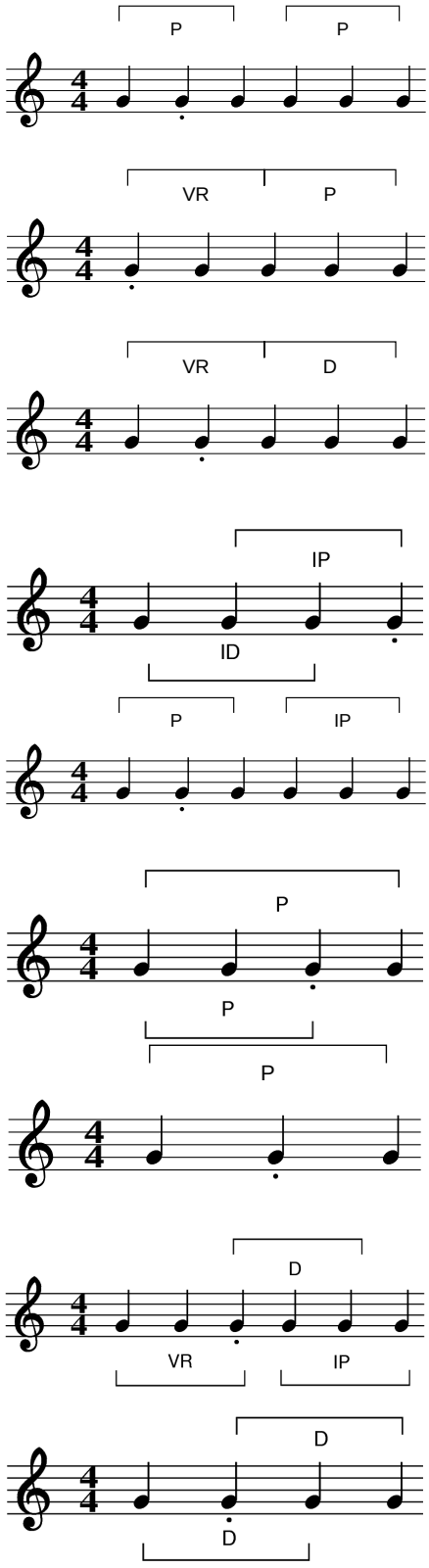

Figure 8. Musical context of a note (marked with a dot) described by Rules S-2, S-3, O-1, O-2, O-3, E-2a, A-1, A-2 and A-3, respectively from top to bottom

note-level choices of the pianist [35]. In the recordings the tempo of a performed piece is not constant (as it is in our case). In fact, of special interest to them are the tempo variations throughout a musical piece. Our work, although similar in objectives, attack a different problem. The work by Widmer focus on classical piano data where (1) the tempo of the performed pieces is not constant and (2) there are no alterations to the melody (in classical piano these alterations are considered performance errors). Thus, while Widmer's work focus on global tempo transformations, we are interested in note-level tempo transformations (i.e. note onset and duration). Also, we induce rules about melody alterations (e.g. ornamentations) which are absent in Widmer's work.

Tobudic et al. [31] describe a relational instance-based approach to the problem of learning to apply expressive tempo and dynamics variations to a piece of classical piano music, at different levels of the phrase hierarchy. The different phrases of a piece and the relations among them are represented in first-order logic. The description of the musical scores through predicates provides the background knowledge. The training examples are encoded by another predicate whose arguments encode information about the way the phrase was played by the musician. Their learning algorithm recognizes similar phrases from the training set and applies their expressive patterns to a new piece. Our work differs from Tobudic's work in the same way as it differs from that of Widmer since both study classical piano expressive performance.

Lopez de Mantaras et al. [16] report on a performance system capable of generating expressive solo performances in jazz within a case-based reasoning system. Their system focuses on note onset, duration and energy. However, their system is incapable of explaining the predictions it makes and does not take into account melody alterations.

Other inductive logic programming approaches to rule learning in music and musical analysis include [7], [1], [20] and [13]. In [7], Dovey analyzes piano performances of Rachmaniloff pieces using inductive logic programming and extracts rules underlying them. In [1], Van Baelen extended Dovey's work and attempted to discover regularities that could be used to generate MIDI information derived from the musical analysis of the piece. In [20], Morales reports research on learning counterpoint rules using inductive logic programming. The goal of the reported system is to obtain standard counterpoint rules from examples of counterpoint music pieces and basic musical knowledge from traditional music. In [13], Igarashi et al. describe the analysis of respiration during musical performance by inductive logic programming. Using a respiration sensor, respiration during cello performance was measured and rules were extracted from the data together with musical/performance knowledge such as harmonic progression and bowing direction.

There are a number of approaches which address expressive performance without resourcing to machine learning techniques. One of the first attempts to provide a computer system with musical expressiveness is that of Johnson [14]. Johnson developed a rule-based expert system to determine expressive tempo and articulation for Bach's fugues from the *Well-Tempered Clavier*. The rules were obtained from two expert performers. In contrast, in our approach we apply machine learning techniques in order to automatically induce rules from examples.

A long-term effort in expressive performance modeling is the work of the KTH group [2, 8, 9]. Their *Director Musices* system incorporates rules for tempo, dynamic and articulation transformations. The rules are obtained from both theoretical musical knowledge, and experimentally from training using an analysis-by-synthesis approach. The rules are divided into *differentiation rules* which enhance the differences between scale tones, *grouping rules* which specify what tones belong together, and *ensemble rules* which synchronize the voices in an ensemble. Here again, the difference is that we apply machine learning techniques to our data.

Canazza et al. [3] developed a system to analyze the relationship between the musician's expressive intentions and her performance. The analysis reveals two expressive dimensions, one related to energy (dynamics), and another one related to kinetics (rubato).

Dannenberg et al. [6] investigated the trumpet articulation transformations using (manually generated) rules. They developed a trumpet synthesizer which combines a physical model with an expressive performance model. The performance model generates control information for the physical model using a set of rules manually extracted from the analysis of a collection of performance recordings. Dannenberg's work is not directly related to the work reported in this paper in the sense that here we do not induce any rules about intra-note expressive transformations. However, we have studied this in [27].

5. CONCLUSION

This paper describes an inductive logic programming approach for learning expressive performance rules from recordings of Jazz standards by a skilled saxophone player. Our objective has been to find local and structure-level rules which predict, for a significant number of cases, how a particular note in a particular context should be played or how a melody should be altered. In order to induce these rules, we have extracted a set of acoustic features from the recordings resulting in a symbolic representation of the performed pieces and then applied an inductive logic programming algorithm to the symbolic data and information about the context in which the data appeared. Inductive logic programming has proved to be an extremely well suited technique for learning expressive performance rules. This is mainly due to the expressiveness of the induced first order logic rules, and the fact that inductive logic programming allows background knowledge to play an important role in the learning process. In addition, inductive logic programming allows considering an arbitrary-size note context without explicitly defining extra attributes for each context extension.

Future work: We plan to increase the amount of training data as well as experiment with different information encoded in it. Increasing the training data, extending the information in it and combining it with background musical knowledge will certainly generate a more complete set of rules. We plan to induce rules about intra-note expressive transformations which certainly play a very important role in expressive Jazz saxophone. In the past, we have also used our research not only for obtaining interpretable rules about expressive transformations in musical performances, but also to generate expressive performances. In this direction, we are exploring how to best implement the induced rules in order to obtain expressive performances of a piece and at the same time be able to explain the transformations undertaken by the system [26]. We also intend to incorporate higher-level structure information (e.g. phrase structure information) to obtain a more complete integrated model of expressive performance. Another short-term research objective is to compare expressive performance rules induced from recordings at substantially different tempos. This would give us an indication of how the musician's note-level choices vary according to the tempo.

Acknowledgments: This work is supported by the Spanish TIC project ProMusic (TIC 2003-07776-C02-01). We would like to thank Maarten Grachten for providing his Narmour I/R structure parser implementation, as well as the anonymous reviewers for their insightful comments and pointers to related work.

6. REFERENCES

[1] Van Baelen, E. and De Raedt, L. (1996). Analysis and Prediction of Piano Performances Using Inductive Logic Programming. International Conference in Inductive Logic Programming, 55-71.

[2] Bresin, R. 2002. Articulation Rules for Automatic Music Performance. In Proceedings of the 2001 International Computer Music Conference. San Francisco, International Computer Music Association.

[3] Canazza, S.; De Poli, G.; Roda, A.;and Vidolin, A. 1997 Analysis and Synthesis of Expressive Intention in a Clarinet Performance. In Proceedings of the 1997 International Computer Music Conference, 113-120. San Francisco, International Computer Music Association.

[4] Chen, M., Jan, J. and Yu, P.S. (1996). Data Mining: An Overview from a Database Perpective. IEEE Trans. Knowledge and Data Engineering 8(6), 866-883.

[5] Colmenauer A. (1990). An Introduction to PROLOG-III. Communications of the ACM, 33(7).

[6] Dannenberg, R. B., and Derenyi, I. 1998. Combining Instrument and Performance Models for High-Quality Music Synthesis. Journal of New Music Research 27(3): 211-238.

[7] Dovey, M.J. (1995). Analysis of Rachmaninoff's Piano Performances Using Inductive Logic Programming. European Conference on Machine Learning, Springer-Verlag.

[8] Friberg, A.; Bresin, R.; Fryden, L.; and Sunberg, J. 1998. Musical Punctuation on the Microlevel: Automatic Identification and Performance of Small Melodic Units. Journal of New Music Research 27(3): 217-292.

[9] Friberg, A.; Bresin, R.; Fryden, L.; 2000. Music from Motion: Sound Level Envelopes of Tones Expressing Human Locomotion. Journal of New Music Research 29(3): 199-210.

[10] Gabrielsson, A. (1999). The performance of Music. In D.Deutsch (Ed.), The Psychology of Music (2nd ed.) Academic Press.

[11] Gómez, E., Gouyon, F., Herrera, P. and Amatriain, X. (2003). Using and enhancing the current MPEG-7 standard for a music content processing tool, Proceedings of the 114th Audio Engineering Society Convention.

[12] Gómez, E. Grachten, M. Amatriain, X. Arcos, J. (2003). Melodic characterization of monophonic recordings for expressive tempo transformations. Stockholm Music Acoustics Conference.

[13] Igarashi, S., Ozaki, T. and Furukawa, K. (2002). Respiration Reflecting Musical Expression: Analysis of Respiration during Musical Performance by Inductive Logic Programming. Proceedings of Second International Conference on Music and Artificial Intelligence, Springer-Verlag.

[14] Johnson, M.L. (1992). An expert system for the articulation of Bach fugue melodies. In Readings in Computer-Generated Music, ed. D.L. Baggi, 41-51, IEEE Computer Society.

[15] Klapuri, A. (1999). Sound Onset Detection by Applying Psychoacoustic Knowledge, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP.

[16] Lopez de Mantaras, R. and Arcos, J.L. (2002). AI and music, from composition to expressive performance, AI Magazine, 23-3.

[17] Maher, R.C. and Beauchamp, J.W. (1994). Fundamental frequency estimation of musical signals using a two-way mismatch procedure, Journal of the Acoustic Society of America, vol. 95 pp. 2254-2263.

[18] McNab, R.J., Smith Ll. A. and Witten I.H., (1996). Signal Processing for Melody Transcription,SIG working paper, vol. 95-22.

[19] Mitchell, T.M. (1997). Machine Learning. McGraw-Hill.

[20] Morales, E. (1997). PAL: A Pattern-Based First-Order Inductive System. Machine Learning, 26, 227-252.

[21] Narmour, E. (1990). The Analysis and Cognition of Basic Melodic Structures: The Implication Realization Model. University of Chicago Press.

[22] Narmour, E. (1991). The Analysis and Cognition of Melodic Complexity: The Implication Realization Model. University of Chicago Press.

[23] Quinlan, J.R. (1990). Learning logical definitions from relations. Machine Learning, 5, 239-266.

[24] Quinlan, J.R. (1993). C4.5: Programs for Machine Learning, San Francisco, Morgan Kaufmann.

[25] Ramirez, R. and Hazan, A. and Gmez, E. and Maestre, E. (2004). A Machine Learning Approach to Expressive Performance in Jazz Standards. Proceedings of 10th International Conference on Knowledge Discovery and Data Mining, Seattle, WA, USA.

[26] Ramirez, R. and Hazan, A. (2005). Modeling Expressive Music Performance in Jazz. Proceedings of the 18th Florida Artificial Intelligence Research Society Conference (FLAIRS 2005), Clearwater Beach, FL. (to appear).

[27] Ramirez, R. Hazan, A. Maestre, E. (2005). Intra-note Features Prediction Model for Jazz Saxophone Performance. Proceedings of International Computer Music Conference 2005; Barcelona

[28] Repp, B.H. (1992). Diversity and Commonality in Music Performance: an Analysis of Timing Microstructure in Schumann's 'Traumerei'. Journal of the Acoustical Society of America 104.

[29] Shapiro, E. (1983). Algorithmic Program Debugging. Cambridge MA, MIT Press.

[30] Srinivasan, A. (2001). The Aleph Manual.

[31] Tobudic A., Widmer G. (2003). Relational IBL in Music with a New Structural Similarity Measure, Proceedings of the International Conference on Inductive Logic Programming, Springer Verlag.

[32] Todd, N. (1992). The Dynamics of Dynamics: a Model of Musical Expression. Journal of the Acoustical Society of America 91.

[33] Widmer, G. (2002). Machine Discoveries: A Few Simple, Robust Local Expression Principles. Journal of New Music Research 31(1), 37-50.

[34] Widmer, G. (2002). In Search of the Horowitz Factor: Interim Report on a Musical Discovery Project. Invited paper. In Proceedings of the 5th International Conference on Discovery Science (DS'02), Lbeck, Germany. Berlin: Springer-Verlag.

[35] Widmer, G. (2001). Discovering Strong Principles of Expressive Music Performance with the PLCG Rule Learning Strategy. Proceedings of the 12th European Conference on Machine Learning (ECML'01), Freiburg, Germany. Berlin: Springer Verlag.