Global Musical Tempo Transformations using Case Based Reasoning

by

Maarten Grachten

Submitted in partial fulfilment of the requeriments for the degree of Diploma of Advanced Studies Doctorate in Computer Science and Digital Communication Department of Technology

Tutor: Dr. Josep Lluís Arcos

Universitat Pompeu Fabra

Barcelona, September 2004

Acknowledgments

I would like to express my gratitude to Ramon López de Mántaras and Josep Lluís Arcos. In the first place for offering me a great place to work, to study, and to meet people at the Artificial Intelligence Research Institute (IIIA) in Barcelona. Secondly, for sharing their experience through invaluable advice, throughout the time I have worked with them.

And I also want to mention Lieveke van Heck. We have been together for a lot of years now and she showed patience, soothed doubts and raised spirits. Thanks!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this research work, in partial fulfillment of the doctoral programme 'Informàtica i communicació digital' of the Pompeu Fabra University, Barcelona, we present our current work and the goal of our thesis research. We will report our recent activities of applying a problem solving technique called Case Based Reasoning (CBR) in the area of expressive music processing. To provide a broader perspective for this work, we will give an overview of past and current research in the field of music performance research and CBR. Finally, based on the current state of our research work, we will propose some concrete lines of work to fulfill the research goals.

In the rest of this chapter, we will introduce the field of expressive music research, and musical tempo transformations as a special kind of this research. In chapter 2, we will have a more detailed look at some topics in music performance research. We will pay attention to more fundamental and cognitive/psychological research as well as review various approaches to expressive music generation by computational means. In chapter 3, we will introduce Case Based Reasoning (CBR), a problem solving approach that we believe is suitable for application in the domain of musical expression processing. The research-goals of this research will be presented in chapter 4. In chapter 5, we will describe a technique for realizing tempo transformations of musical phrases, that is, generating a performance for a given score at a certain tempo, given a performance of that score at a different tempo. Finally, in chapter 6, we will present conclusions and propose future research.

1.1 Expressiveness in Musical Performances

It has been long established that when humans perform music from score, the result is never a literal, mechanical rendering of the score (the so called

nominal performance). Even when (skilled) musicians intentionally play in a mechanical manner, noticeable differences from the nominal performance occur [107, 12]. Furthermore, different performances of the same piece, by the same performer, or even by different performers, have been observed to have a large number of commonalities [59, 107]. Repp [98] showed that graduate piano students were capable just as well as professional piano players, of repeatedly producing highly similar performances of the same piece.

Investigations into the performance of music dates from at least the end of the nineteenth century and early twentieth century. For example, around 1896, Binet and Courtier [14] prepared a grand piano to display the dynamics of the keys pressed on a paper. In 1913, Johnstone [66], observed that in piano performances, the notes that belong to the melody are often played slightly earlier than chord notes at the same metrical position. Around 1930, extensive research on music performance was carried out and reported by a group of researchers led by Seashore [107]. This research was mainly concerned with singing, violin, and piano performances.

The performance of a musical piece is determined by several factors. Firstly, physical conditions of the musician and her instrument are of influence. Obviously, the type of instrument determines to a large extent the character of the performance. Also, physiological conditions of the musician (such as fatigue, or state of health) can play a role. Secondly, the motor skills of the musician are of importance. This becomes clear when comparing the performances of a novice to those of a skilled musician. With practice, the musician trains his motor speed and accuracy, reducing the amount of unintentional deviation from performance to score. A third factor consists of the cognitive, and affective aspects of the musician. It has been shown by Sloboda [110] that performers deviate systematically from the score when they play variations of the same score that consist of exactly the same sequence of notes (only their placement within the meter was changed). This result rules out the possibility of deviations due to motor incapacities and shows the influence of meter on the performance of a score. Other studies have shown systematic deviations in performances that were played with different moods or expressive intentions [102, 43, 32].

1.1.1 Functions of Performance

As far as performance deviations are intentional (that is, they originate from cognitive and affective sources as opposed to e.g. motor sources), they are commonly thought of as conveying *musical expression*. But what is it that is being expressed? Two main functions of musical expression are generally recognized. We will address both functions.

Expression and Musical Structure Firstly, expression is used to clarify the musical structure (in the broad sense of the word: this includes metrical structure, but also the phrasing of a musical piece, harmonic structure etc.). The research by Sloboda mentioned above [110] showed the influence of metrical structure on performances by having pianists perform the identical sequences of notes, differing in the position of the bar lines. The pianists varied their performances such that e.g. the notes at the beginning of measures were played louder and more legato than other notes. Furthermore, he observed that the more advance the performers were, the more they utilized this kind of expression, and the better listener's were able to transcribe the performed music correctly. This is a clear indication that expression is used to clarify metrical structure.

Phrase structure also has a salient effect on performance. Phrases were found to start and end slow, and be faster in the middle [59] . Moreover, Todd [116, 117] invented a model that predicts the level of rubato of a given musical piece, given a hierarchical grouping structure of the piece. The predictions of this model are similar to the rubato patterns in professional performances of the piece. Gabrielsson [42] found that pianists performing Mozart's Piano Sonata K. 331, tended to lengthen note durations considerably at the end of phrases. Similarly, he found the tones to be relatively loud in the middle of the phrases and relatively soft at the beginning and end.

Another form of structure that influences performance is harmonic and melodic tension. Harmonic and melodic tension are commonly defined by reference to the circle of fifths (where the tension is low for notes or chords that are close together on the circle of fifths and high for those that are far apart) [77]. Palmer [90] calculated a positive correlation between note lengthening and tension. Contrastingly, no correlation was found between note intensity and tension. In the same article, Palmer showed that *melodic expectancy* (the extent to which an implied continuation of a melody is actually realized (see also section 3.5.1 on the Implication/Realization model) did correlate positively with note intensity (unexpected notes were played louder), but not with note lengthening. As an explanation of the fact that the expression of tension-relaxation and melodic expectancy are realized in unrelated ways, Palmer notes that the two phenomena manifest themselves on different time scales; tension-relaxation is a phenomenon at the level of phrases and sub phrases (that is, a large time scale), whereas melodic expectancy is manifest from note to note, i.e. on a smaller time scale.

In a general survey of the relation between expression and musical structure, Clarke [24] proposes the interesting view that expression is tied to structure by a limited set of rules (like the rules proposed by Sundberg and co-workers citeSundberg91a,Friberg91, see section 2.2.2). Hence, the diver-

sity of ways in which a piece can be played is not due to ambiguous rules for expression, but due to the diversity of ways in which the music can be structurally interpreted (i.e. ambiguous musical structure). In this context, he notes the practice of the live performances of jazz standards. In this practice, the music that is played belongs to a widely known and fixed repertoire. Therefore the audience is usually acquainted with the music, and the expressiveness of the performance is not constrained by the requirement that it should clarify the basic musical structure to the listeners. The musicians can thus freely vary their expressiveness to surprise the audience through an unexpected musical interpretation of the piece.

Another structural aspect of music that has been found to influence musical expression is the melody. In ensemble performance (but also polyphonic piano performances), the voice that plays the melody tends to be slightly (by around 20–30 ms.) ahead of the accompaniment [95] [89]. The purpose of *melody lead* is presumed to be the avoidance of masking of the melody by the accompaniment, and to facilitate voice separation in human perception.

Surveys of music performance research in relation to structural aspects are [41, 25, 91].

Expression and Emotional Content Secondly, expression is used as a way of communicating, or accentuating affective content. Langer [75] proposed the view that the structure of music and the structure of moods or feelings are isomorphic. Langer notes that similar properties are ascribed to music and emotional life (such as 'excitation' and 'relief'). Another influential theory about music and meaning (which subsumes emotion) is from Meyer [82]. He states that meaning (be it emotional, or aesthetic) arises in music when expectations raised by the music are not realized. Early work to investigate the relation between emotional character and musical structure is reviewed in Rigg [102]. Some typical regularities were found, e.g.: solemn music is often played slow, low pitched and avoids irregular rhythms and dissonant harmonies; happy music is fast, high pitched and in major mode and contains little dissonance; exciting music is fast and loud and apt to contain dissonance [43] (another mapping between emotional characters and musical cues can be found in [67]). Gabrielsson and Lindström [43, 81] have studied the relation between emotional intentions and micro structure in music (e.g. timing deviations, intensity changes and articulation). They compared versions of "Oh, my darling Clementine", played with emphasis on different emotional characters ('happy', 'sad', 'solemn', 'angry', 'soft', 'expressionless'). Their results with regard to the overall properties tempo and loudness were mainly in accord with previous results: angry and happy

versions were played faster than sad, soft, and solemn versions; angry versions were played loudest. With respect to the micro structure they also found clear differentiation between the different versions, Notably the variance of articulation and loudness. Duration ratios in the rhythm were also different for different emotional characters (the performers were given great freedom of performance, only the pitch sequence was to be kept intact).

Similar results have been found by Canazza et al. [19], who have studied how physical parameters of recorded performances (e.g. timbre, and temporal parameters like articulation and global tempo) affected by varying expressive intentions of the musician. The performer was told to communicate a particular mood through his performance, that was described by a sensorial term, e.g. 'heavy','light','bright', or 'dark'. The sonological analysis of the recordings made it possible to tie particular parametric values to particular moods (for instance a light performance turned out to be in fast tempo, with shortened note durations and soft attacks). The results were validated by performing listening tests on synthetic performances that were generated with the physical parameter values corresponding to particular moods. The subjects were able to recognize the intended mood in the synthesized performances.

In a related perceptual analysis of listener judgments of expressed moods in musical performances [20], Canazza et al. found that there is a large degree of consistency between the mood ratings of the listeners. They concluded that tempo and note attack time are two important factors by which listeners rank the sensorial terms (as those mentioned above). Moods like 'heavy' were opposed to 'light' and 'bright' on the scale of tempo, whereas 'soft', 'dark' and 'hard' were distinguished from each other on the scale of attack time.

From the above, it becomes clear that listeners are often very well able to identify the mood in which a piece is played, in a consistent way. The use of musical expressiveness to communicate a particular mood or feeling is one of the phenomena we wish to investigate further in this thesis. We will come back to this in chapter 4.

See [44] for a review on music performance research.

1.1.2 Methodology

Until now, we have defined expression simply as 'deviations from the nominal performance'. This is a common definition, but there are other ways to define expression [115]. Expression can also be defined as the micro structure of a performance that has been left unspecified by the score. Alternatively, expression could be defined relative to the performance itself. We will discuss each of these definitions in some more detail in chapter 2.

Another fundamental question is where the focus of expressive music research should be. Traditionally, research was mainly focused on the performance of music, being the 'carrier' of expressive information. But, as has been stressed by Desain and Honing [63], expression has intrinsically perceptual aspects. That is, deviations from the nominal performance should not be regarded as expression when they are too small to be perceived by humans. On the other hand, if the performance deviations from a particular musical structure are too large, the performance may become ambiguous (in the sense that the listener cannot uniquely identify the musical structure that has been performed). This suggests that expressive music research should not only focus on the production of music, but also on the perception of music. We will discuss some perception oriented music performance research in section 2.1.

As a final issue in the study of music performance, we mention another methodological issue on the analysis of music performance. When analyzing the timing deviations of the notes a musician plays, it is common to construct a *tempo curve* from this deviations. This curve denotes the instantaneous tempo, inferred from the played durations (and possibly onsets) of the notes, as a function of score-time. For example, when a note is played shorter than in the nominal performance, the instantaneous tempo will be higher and when the note is played longer, the tempo will be lower than the nominal tempo. Desain and Honing [36] argued that the interpretation of a sequence of duration-ratios as a tempo curve is a misconception. The idea of a tempo curve has several false implications, for example that the tempo curve of a piece is something independent of the piece itself, that could possibly transferred to another piece to obtain an appropriate performance. Another implication that does not hold according to Desain and Honing is that new notes could be inserted into the performance in a musically satisfying way, by assigning them the timing values based on their position in the tempo curve (in practice, this means that the timing of the note will be interpolated from the surrounding notes). According to Desain and Honing, this way of handling expressive tempo is fruitless because the tempo curve does not reflect a cognitively real concept in the mind of the musician or listener. That is, the timing of musical events is determined rather by hierarchical factors (musical structures, like meter and phrase) than by temporal order.

1.2 Musical Tempo Transformations

In this section, we will elaborate on a more specific aspect of music performance, that of musical tempo transformations. The central question here

is how the performance of a musical piece relates to the performance of the same piece at a different tempo. An important concept in this context is that of (*relational invariance*). In general, this is the idea that under changing conditions (e.g. the global tempo, or global loudness of the performance), performance aspects (like timing and dynamics) remain proportionally the same. For example, relational invariance of the timing of notes across tempo would imply that the ratios between successive *Inter onset Intervals* (*IOI*, the time interval between the onset of a note and the onset of a subsequent note) will remain constant independent of the tempo at which the piece is performed.

That expressive performance timing is actually relational invariant with respect to tempo is not very probable. Although Repp [97] found evidence for this, counter-evidence is provided by Desain and Honing [36, 37]. Desain and Honing suggested that the contrasting results may be explained by the fact that the tempo range of the performances under consideration was smaller in Repp's experiment (plus and minus 15%) than in their experiment (plus and minus 30%). The relational invariance found by Repp might therefore hold only for small tempo changes, and be violated in larger tempo changes. Furthermore, the different results might be due to the different pieces that were performed (involving different rhythmic structures). Subsequent experiments by Repp [99] also produced weak counter evidence for relational invariance for expressive timing. More specifically, he found that for performances at fast and slow tempos the expressiveness was decreased with respect to the performances at preferred (medium) tempo.

In the same study, Repp showed that listeners preferred a higher degree of expressiveness for low tempos than was present in the performances. Repp suggested that the decreased degree of expressiveness could be due to the fact that the musicians were not comfortable with playing the piece at a relatively low tempo, and had to put too much effort in keeping a steady tempo to produce a appropriate expressiveness. As an explanation for the timing of grace notes (for which relational invariance is violated most strongly), Repp mentioned the possibility of motor capacity or perception as limiting factors for producing grace notes of proportional lengths at fast tempos. That is, playing a grace note of proportional duration at a faster tempo, would either lead to a succession of notes that would be too fast to play, or too fast to be perceived as distinct notes. Desain and Honing [36] proposed other, less concrete but interesting explanations for lack of relational invariance in expressive timing. According to them, expressiveness is closely related to musical structure. For example, when decreasing the tempo, the pulse of the music tends to shift to lower metrical levels (e.g. from quarter notes to eighth notes). This may imply that notes in the lower level metrical grid

gain importance, which could account for a difference in timing. Correspondingly, when performing at faster tempos, the musician probably omits such expressive details and uses the expressive timing to stress larger time scale phenomena (like phrasing). This explanation is in accordance with Clarke's idea mentioned earlier in this chapter, that expressiveness in the performance reflects the structural interpretation of the music by the performer [24].

With respect to rhythm production, as an alternative to the relational invariance hypothesis, the hypothesis of ratio simplification has been proposed [92]. This hypothesis states that when tempo increases, complex duration ratios (e.g. 5:8) tend to be played as simpler ratios (e.g. 1:1, or 1:2). This hypothesis is founded by the dynamical systems approach to human motor skills, in which simple duration ratios are stronger attractors than complex duration ratios. Repp et al. [100] have also investigated the effect of tempo on the performance of different types of simple rhythms. They found that for two-note rhythms, ratio simplification did not occur. The timings of the notes were found to be relational invariant instead. For three-note rhythms, the results were different. Whereas the duration of the shortest note was relational invariant, the ratios between the two longer note durations were simplified with increasing tempo.

Chapter 2

Music Performance Research

The vast majority of studies in music performance research available at this time, focus on Western classical music, dating from roughly speaking the seventeenth to the twentieth century (e.g. music from well-known composers like Mozart, Bach, Beethoven, Schumann, and Chopin). In addition, most of the studies study piano performance. Obvious reasons for this are not hard to conceive: the relative ease with which performances can be measured on piano, in terms of note onsets, and the lack of continuous control over tone production, causing the expressive space to be relatively tractable. Studies of non-classical music and other instruments than piano are less numerous. Nevertheless, in the last decades, an increasing number of studies were concerned with jazz music [27], [96]. Recently, a special issue of the journal *Music Perception* [86] has been devoted to music research in jazz music.

It is an important question to what extent expressive performance principles and regularities discovered in studies of classical music can be generalized to jazz music. Intuitively, we may expect that there is at least some degree of similarity among the performances of different kinds of music, to the extent that the performance is determined by motor and perceptual aspects of the human body. On the other hand, one would naturally expect differences in performance of classical and jazz music as well, simply by the fact that they are different musical styles, and a musical style is not just constituted by structural and syntactical properties of the music, but also by performance practice. As Iyer and others [64, 65] have pointed out, in non-classical music, notably African-American music like jazz, funk and hip-hop, expressivity can be identified just as well as in classical music, but the dimensions in which expressiveness is manifest are different. For example, expressiveness through variation of local tempo (*ritardando* and *accelerando*), which is a major expressive factor in classical music, is much less common in African-American music, because the music is based on a *groove*, a steady pulse that is usually

present throughout the whole piece. In its turn, rhythmic expressiveness, involving very subtle temporal deformations of duration ratios, is more dominant in such groove based music.

Apart from the kind of music under study, the area of music performance research comprises several quite distinct approaches, which can be characterized according to different dimensions: theory-driven vs. data-driven, oriented towards cognitive plausibility vs. computational simplicity, perception-oriented vs. production-oriented, etcetera. In this chapter, we will give an overview of some research work in different fields of music research. After this exposure, we will try to relate these approaches to each other and discuss some advantages and disadvantages of the various approaches. This discussion is partly based on a panel on music performance at the MOSART Workshop 2001 [49].

2.1 Fundamental and Methodological Issues in Music Performance Research

2.1.1 Definitions of Expression

In [115], Timmers and Honing elaborate on some fundamental issues in music performance research. They discuss several definitions of musical *expression*. Firstly, there is the definition of 'expression as micro structure'. This definition conceives of expression as everything that is left unspecified in the score. The score only specifies the macro structure of the music, leaving undecided how the low level attributes of the macro elements should be realized. This is a rather broad definition. The second definition is 'expression as deviation from a musical score'. In this definition, which is the most common definition, the expression is taken as the deviation from a mechanical rendition of the score, in terms of timing, dynamics et cetera. A third definition, proposed by Desain and Honing [35], defines 'expression as deviation within a performance'. More precisely, 'expression is the deviation of a lower order unit from the norm as set by a higher order unit' ([115], page 5). This definition thus assumes a hierarchical description of the musical piece. For example, the deviation of a beat duration can be calculated as from the duration of the enveloping bar.

After concretizing these definitions for expressive timing (in terms of IOI), Timmers and Honing argue that different definitions of expression emphasize different aspects of the performance. The 'expression as micro structure' definition for timing can be realized by normalizing performed IOI's by their corresponding score IOI's, or alternatively by computing local beat durations

from the performance IOI's. The first alternative shows mainly small time scale timing deviations, and obscures trends, or larger time scale changes in timing (such as rubato). The second alternative shows such global trends more clearly. The second definition, 'expression as deviation from the score', is realized as the deviations of IOI's from the mean note or beat duration. The results are equivalent to those of the first definition. Measurement of expressiveness according to the third definition, 'expression as deviation within a performance', is performed by relating the performed note IOI's to the performed bar IOI's. This approach can clarify timing regularities that relate to meter and rhythmical patterns.

2.1.2 Representation of Timing and Tempo

Related to the issue of interpreting different aspects of expressivity is the argument for differentiation of tempo changes and timing [62, 13]. Although both tempo changes and timing become manifest through temporal displacement of musical events, they refer to different musical phenomena. Tempo changes refer to the effect of speeding up or slowing down the performance of the melody in a continuous manner, whereas timing refers to local displacements of notes, chords or other musical events, with respect to the tempo (in a manner of speaking, timing is on top of tempo changes). Honing [62] notes that representations commonly used for describing musical performances (like tempo-curves that map score times to instantaneous tempo, or time-shift functions that map score times to deviations with respect to the score) fail to capture both aspects separately. As a result, they cannot well be used in systems for performance generation, since applying such functions in composition (that is, apply one function on the result of another function) yields undesired results. This is caused by the fact that the input of the second function is not score-time but performance time. Time maps (TM's) are a better solution for such applications. A TM maps performance time before transformation (pre-perturbed time) to performance time after transformation (perturbed time), which allows for composition. Another practical advantage of TM's over tempo-curves or time-shift functions is that the former provides performance time directly as output, whereas the latter two require extra calculation to obtain performance time. However, TM's also have some limitations:

- Score times are lost in composition
- Support for concatenation is limited
- Access to score and performance durations in composition

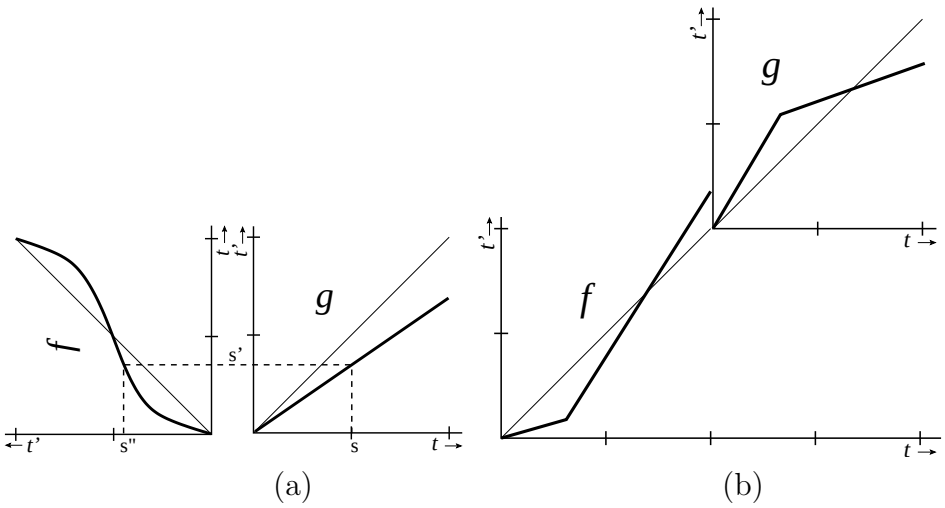


Figure 2.1: Time Maps; (a) Two TM's f and g composed ($s'' = f(s') = f(g(s))$); (b) Two TM's f and g concatenated by aligning score time.

The first limitation is similar to the limitation of tempo-curves and time-shift functions just mentioned. This is a problem for TM's that are defined in terms of score position (e.g. a TM for applying 'swing'-timing to a performance). The second limitation refers to the problem that when two TM's are concatenated in such a way that the pre-perturbed times are aligned, the resulting function is not necessarily monotonically non-decreasing. The third limitation is that the time intervals of events may be changed by applying a TM, so that a subsequently applied TM that is defined in terms of pre-perturbed times, will operate on the wrong time interval.

Figure 2.1 shows graphical representations of a composition of two TM's and a concatenation of two TM's, respectively. Note that discontinuities of performance time as a result of concatenation can affect the order in which score events are played.

Although the first and second limitation can be solved by adding score time as a parameter to the TM functions, the third problem cannot be solved in this way. Honing therefore proposes to construct a broader framework for the representation of timing: *timing functions* (TIF's). A TIF is a function that is composed of a TM representing tempo changes and a TM representing timing. the application:

$$\mathbf{f} \equiv \langle f^{+}, f^{\times} \rangle$$

where f^{+} is a function $f^{+}(s,t) \rightarrow t'$ denoting the timing TM and f^{+} is a function $f^{\times}(s,t) \rightarrow t'$ denoting the tempo change TM. The evaluation of $\mathbf{f}$,

given a score time s and a performance time t, through an evaluation function $\mathbf{E}$ is defined as follows:

$$\mathbf{E}(\mathbf{f}, s, t) = f^{+}(s, f^{\times}(s, t))$$

The composition of two TIF's $\mathbf{f}$ and $\mathbf{g}$ is defined as:

$$\mathbf{f} \otimes \mathbf{g} = \langle f^{+} \otimes g^{+}, f^{\times} \otimes g^{\times} \rangle$$

The composition of TM's $f(s,t)$ and $g(s,t)$ is defined equally for both tempo change and timing TM's:

$$(f \otimes g)(s, t) = f(s, g(s, t))$$

Concatenation of TIF's $\mathbf{f}$ and $\mathbf{g}$ at score time m is defined as:

$$\mathbf{f} \oplus_{\mathbf{m}} \mathbf{g} = \langle f^{+} \oplus_{m} g^{+}, f^{\times}_{m} \oplus g^{\times} \rangle$$

where concatenation of timing TM's is done by aligning score-times and concatenation of tempo change TM's by aligning performance-times.

2.1.3 Perception of Time and Rhythm

Desain and Honing [38] studied the perception of rhythm by listeners who listened to three-note rhythms that were continuously varied, in order to explore the space of all temporal patterns. Listeners were asked to categorize the temporal patterns they heard, that is, to transcribe the perceived rhythms into musical notation. An interesting result Desain and Honing found was that the space of perceived temporal patterns did not correspond linearly to the space of performed patterns. In other words: in the perception of music, time does not flow homogenically, but it gathers in clumps. For the three-note pattern, this phenomenon can be displayed on so-called *time clumping maps*, triangular maps where each of the edges represents the duration of one of the three notes. Each three-note pattern can be represented as a point on this map. By coloring areas on the map where the temporal patterns were categorized identically, clumps emerge that correspond to perceived rhythms.

2.2 Expressive Music Generation

In addition to analysis, an important part of music performance research is the design of computer systems that will synthesize, rather than analyze, expressive music. Apart from the practical use of such systems to render

music expressively, they provide a means to test the validity of hypotheses and models of expressivity. Those hypotheses and models might either be designed 'by hand', or derived automatically from performance data. In this section, we will discuss several approaches to expressive music generation, or performance rendering.

2.2.1 Machine Learning, Musical Data-mining and Performance Prediction

In recent research, Widmer and his research group at ÖFAI, Vienna [119, 121, 120] have used machine learning techniques to derive music performance knowledge from recorded music performances. In [121], they report the automatic derivation of rules, or principles for playing expressively. The data used for this experiment were recordings of 13 complete Mozart piano sonatas, played by a concert pianist. The music was played on a computer monitored grand piano, so the recording was stored in MIDI format. In addition to the performance, the musical scores were coded, including annotations as to which notes constitute the melody, or the way the notes should be performed (e.g. 'staccato'). For the derivation of the rules, only the melody notes were taken into account.

Since the representation of the data did not capture higher level structure in the melodies, the information in the data could only be interpreted in terms of the lowest musical level, the note level. This has implications for the learnability of the performances, since it is plausible that regularities in music performance do not only refer to single note events, but also to larger scale phenomena such as musical phrases. A previous attempt [119] to predict performance by the derivation of rules without taking into account higher musical levels lead to very large number of rules (over 3000) that were mostly difficult to interpret musically. This lead to the idea that a model (set of rules) derived from the data should only explain what it can plausibly explain based on the data, rather than trying to explain all data at the cost of simplicity and interpetability. That is, a model that predicts performance expression from the note level perspective should only be a *partial* model, rather than a *complete* model.

To achieve this, a new machine learning algorithm, *PLCG*, was designed [122]. This algorithm is oriented towards finding simple and robust classification rules in complex and noisy data. This is achieved by first learning many rules that cover a subset of the data, clustering the learned rules so that similar rules (syntactically and semantically) are in the same cluster; then for each cluster of rules a rule is formed that generalizes the rules in the cluster.

From the resulting set of rules, the rules that have the highest percentage of correct predictions are selected. In this way the obtained rule set consists of a modest number of rules that all have a reasonable degree of generality.

The rules that are learned are conditional rules by PLCG, consisting of a set of conditions (with *and*/*or* relations), that refer to the score, and an action that is to be performed if the conditions are met. The actions refer to the timing, dynamics and articulation of notes. They are not quantitative but qualitative (actions can for example be 'lengthen note', or 'play louder'). An example of a rule that was discovered is:

```
lengthen IF
    next_dur_ration < 1 &
    metr_strength ≤ 2
```

which should be read as *"Lengthen a note if it is followed by a longer note and if it is in a metrically weak position."*.

The models predicted in this way are partial models based on the note level, so they leave opportunity for predicting regularities at higher structural levels by other means. Widmer [120] used this opportunity by predicting higher level regularities such as *crescendo*/*decrescendo* and *accelerando*/*ritardando* using another machine learning approach, called *Nearest Neighbor Learning* (NN). To achieve this, the performances that serve as experience, are analyzed in terms of tempo and dynamics. Both tempo and dynamics curves of the performances are fitted by quadratic functions iteratively (in each round a function is fitted to the residual of the previous round). These functions are stored for each performance. To predict the tempo and dynamics of a new score, the tempo and dynamics functions of the most similar 'learned' scores are taken. The similarity between scores is taken as the Euclidean distance between a vector of musical attributes.

When the NN algorithm for predicting phrase level performance phenomena is combined with the PLCG algorithm for predicting note level phenomena, a composite and complete model emerges that predicts better than either model alone.

ÖFAI's machine learning approach to machine performance of music as just described is currently one of the most advanced approaches in the field. This is confirmed by the fact that there performance rendering system has won the second prize at the 2002 *RenCon* Contest for machine rendering of piano music. Nevertheless, Widmer notes some limitations and room for improvement. The polynomial curve fitting seems to be more promising for predicting dynamics than for tempo. The performance of the system could also benefit from a richer representation language. Furthermore, possible

dependencies between curves at different hierarchic phrase levels are not detected by the current algorithm. Also, it could be interesting to look for inter-dependencies of different dimensions (e.g. tempo and dynamics).

2.2.2 A Grammar for Musical Expression

Another approach to automatic performance of music is to construct a set of performance principles that allow for reconstruction of real expressive performances, as it were, a grammar that describes the structure of musical expression in terms of the musical score.

This approach has been taken by Sundberg, Friberg, and Frydén and co-workers [40, 111], at KTH, Stockholm, Sweden. They have defined a set of context-dependent rules. These rules prescribe small deviations for timing and dynamics of notes, based on their musical context. They can act in ensemble to sequentially process a sequence of notes, to synthesize an expressive performance. In this way, the validity of the rules can be checked, either by judging the musical acceptability of the synthesized performance, or by rating the quantitative difference of expressive deviations from a human performance of the the same piece of music. This approach is classified as *analysis-by-synthesis*, referring to the procedure of analyzing musical expressivity by synthetically mimicking it. In this approach, the factors contributing to musical expressivity are validated by judging their effects on performances. The rules have been incorporated into a software application called *Director Musices* (*DM*), which processes MIDI files to generate expressive performances.

The rules have been developed with the help of a musical expert (L. Frydén, a music performer and music teacher). The method of rule development is as follows: An intuition by the expert about the effect of a particular part in a musical score on the performance of that part of the music, is translated into an initial rule. This rule is used to synthesize an 'expressive' MIDI performance of the score. The expert judges the result and adjusts his instructions if the result is unsatisfactory. Thus, performance rules evolve in an iterative process of synthesis, judgment, and adjustment. The rules affect the duration, overall sound level (dynamics), time envelope of the sound level (articulation), vibrato speed and depth, and fine tuning of the pitch. The rules have a conditional part that specifies the target notes to which the rule applies. Furthermore, each rule has a corresponding quantity parameter, specifying how large the effect of the rule should be (i.e. a scaling factor for the deviations prescribed by the rule).

When the rules are identified by their purpose, three major groups can be distinguished:

Differentiation Rules that help the listener to identify pitch and duration categories.

Grouping Rules that help the listener to perceive contiguous sets of tones as a group, or melodic *Gestalt*.

Ensemble Rules that specify the synchronization and tuning between different voices.

The effects of the differentiation and grouping rules are additive and each rule processes the score in a sequential order, from the start to the end. The status of the ensemble rules is somewhat different: they only apply in the case of polyphonic music, and they do not process the score sequentially, but as a whole. When the ensemble rules are to be applied, the differentiation and grouping rules that affect the timing and duration of the notes are applied to a voice that is extracted from the score by certain guidelines, and the other voices are synchronized to this voice, to avoid contradicting expressive effects within different voices.

Examples of differentiation rules are:

Duration contrast Shorten the short notes and lengthen the long notes.

High sharp Increase the pitch of the notes; the higher the note the sharper it is played.

Melodic charge Emphasize melodically charged notes (relatively dissonant notes given a particular harmonic context). Emphasis is given by increasing amplitude, duration and vibrato depth.

Examples of grouping rules are:

Leap articulation Insert micro pauses between two tones forming a *leap* (a large interval) and create overlap between two tones forming a *step* (a small interval).

Phrase marking Insert micro pauses between tones that separate two phrases; in addition, increase the duration of the phrase-final tone.

Faster uphill Shorten the durations (i.e. increase tempo) of tones initiating an ascending interval.

Examples of ensemble rules are:

Melodic synchronization extract an appropriate voice from the piece and after applying the timing and duration varying rules to that voice, synchronize the other voices to the notes of that voice.

Harmonic intonation adjust pitches of chords in order to minimize frequency beating.

After establishing the rules as described above, experiments were done to tune the quantity parameters that control the level of effect of the rules. This was done in a setup where musical experts were asked to adjust the quantity parameters by means of sliders, so as to produce the performance they liked best [112]. The preferred quantities were generally above zero, affirming the musical relevance of the rules.

An interesting idea is that of having different 'rule cocktails' or 'rule palettes', mixes of (subsets of) rules with varying quantities parameters. These cocktails may be used to perform a piece of music with different moods, like anger, fear, happiness, tenderness, sadness, solemnity, or neutral. This idea was employed by Bresin and Friberg [17]. Their starting point were the findings of studies like those of Gabrielsson [43] (mentioned in section 1.1.1, page 12), which had produced a table of acoustical features (e.g. articulation, tempo, sound level) correlated to music that was performed in particular moods. Bresin and Friberg adjusted the quantity parameters of selected rules to model each of the seven moods mentioned above. The resulting rule palettes were used to generate performances of a piece, each with a different mood. The results were submitted to a listening test, where listeners were asked to assign one of the seven moods to each of the performances. This test showed that the intended moods were mostly recognized correctly. Recognition was best for the happiness and anger moods. Confusion was highest between tenderness/fear, and tenderness/sadness respectively. This may be due to the fact that the moods involved (especially tenderness), are somewhat 'secondary' moods, i.e. they are not as universal and archetypic as anger and happiness.

2.2.3 A Partial Model of Performance: Predicting Rubato

Todd [116, 117] argued that rubato, the pattern of speeding up and slowing down during a performance, is largely determined by the phrase structure of the music. To demonstrate this, he designed a system that predicts a rubato curve based on the time-span reduction of a musical piece ([78]). The system mapped parabolic functions to the different subtrees of the global time-span reduction tree, and added these functions to obtain a final rubato curve.

2.2.4 SaxEx: A Case Based Reasoning System for Expressive Music Generation

A different approach to the generation of expressive performances of music has been explored by Arcos, Mantaras, and Serra [5]. They developed a system called SaxEx, that generates expressive music performances by using previously performed melodies as examples. As mentioned in the introduction, the work presented in this document springs from and extends this approach. Hence, we will address some of the topics involved (especially Case Based Reasoning and musical models) in more detail in subsequent chapters. In the following overview of SaxEx, we will mention those topics briefly, referring to the relevant sections for further details.

The task of SaxEx is to transform inexpressive performances into expressive performances, allowing user control over the nature of the expressivity, in terms of expressive labels like 'tender', 'aggressive', 'sad', and 'joyful'. It is implemented in *Noos* [6, 10], a reflective object-centered representation language designed to support knowledge modeling in problem solving and learning tasks. The input to the system is a musical phrase in MIDI format, together with a sound file of an inexpressive performance of that phrase. This sound file is analyzed using SMS techniques [108], which yields a description of the expressive features of the performance, like note attack, vibrato, etc. To obtain appropriate values for these features, Case Based Reasoning (see section 3.1) is used. Thus, a case base is employed, containing examples of expressively played phrases. For each note of the input phrase, a set of similar notes is retrieved from the case base. This retrieval step employs different *perspectives* on the phrases. One perspective used in SaxEx refers to the musical analysis of the phrase, using music theories such as the Implication/Realization model and GTTM (see section 3.5.1). Such analyses are useful to determine the role of notes in their musical contexts and can be used to assess similarities between notes.

Another perspective is constructed using the desired expressive character of the output performance. This requirement is specified in advance by the user, on three bipolar scales (tender-aggressive, sad-joyful, and calm-restless, respectively). Each pole on these scales is associated with particular expressive values (e.g. tender corresponds to slow attacks and legato note transitions). In this way, the affective requirement perspective is used to bias the performances in the case base that meet those requirements.

Chapter 3

Case Based Reasoning

In this chapter we will have a more detailed look at Case Based Reasoning. Sections 3.1, 3.2, and 3.3 will be about Cased Based Reasoning as problem-solving/learning approach in general, it's advantages, and the tasks involved. In section 3.4, we will focus on the specific kind of Case Based Reasoning where general domain knowledge plays an important role (KI-CBR) and in section 3.5 we will try to establish what are the kinds of knowledge required to construct a Case Based Reasoning system that performs expressive transformations on musical phrases.

Because the terms 'problem', 'solution', and 'case' will be used very frequently in the course of this chapter and their usage may be confusing, we will give some hints about their meanings in the context of Case Based Reasoning. The term 'problem' is commonly used to refer to the general description of the task that a Case Based Reasoning system is to perform (e.g. 'Propose a diagnosis of patient X'). But the term is also used in a more specific sense, to denote that part of a case that specifies the situation that requires a solution (e.g. 'Patient X displays such and such symptoms'). In this sense, a problem is part of the 'problem description', which describes the problem, and possibly also requirements that a solution to the problem should satisfy. 'Solution' again is a general term that, depending on the type of CBR system, could take different forms such as a diagnosis, a classification, or an answer to a question. Finally, a 'case' is a representation of a solved problem, incorporating both a problem description and a solution.

3.1 What is CBR?

In this section, we will survey Case Based Reasoning (CBR), a technique for automatically solving problems developed in the field of Artificial Intelli-

gence. Problem solving by CBR is achieved by retrieving a problem similar to the problem that is to be solved from a case base of previously solved problems (also called cases), and using the corresponding solution to obtain the solution for the current problem. CBR is often regarded as a special case of Instance Based Learning (IBL, see [84] for an overview of IBL), a machine learning technique mostly used for classification tasks, where the class of an unclassified instance is predicted to be identical to the class of the most similar instance within the set of previously seen instances (this approach is also called *Nearest Neighbor Learning*). In this kind of learning, instances are usually represented as a vector of numerical features, where similarity between the instances is taken to be inverse of the euclidean distance between the vectors. However, CBR is characterized by the use of a richer representation of the instances (therefore called 'cases'): values of features can be symbolic, cases can contain more elaborate information than just superficial attributes (e.g. its relation to other cases) and hence similarity computation between cases can involve rather complex comparisons. In this comparison, general domain-knowledge may also be used (for example in the form of rules). Due to it's representational richness, CBR is also suitable for other tasks than classification, e.g. query/answer systems ([70], see below), structural design [60], diagnosis [72], and legal reasoning [11]. The concept of problem solving using CBR is illustrated in figure 3.1, where problem solving is conceived of as mapping points on a problem plane to points on a solution plane. The transition from problem to solution plane is made via a (possibly multiple) neighboring problem, for which the solution is known. The processes of finding a neighboring problem (Retrieval) and adapting the solution of the neighboring problem (Reuse) will be explained more thoroughly in subsections 3.3.2, and 3.3.3

A conceptual distinction can be made between to types of tasks in CBR: *analytical*, and *synthetical* tasks [93]. In analytical tasks there is a limited number of solutions and solutions are simple, non-aggregate entities (classification/diagnosis is a typical analytical task). In synthetic tasks, the solutions have a composite structure, and as a result the number of possible solutions is usually very large. A typical example of a synthetic task is structural design.

Some early work that lead to CBR is from Schank [104], who proposed a model for memory, where experiences are stored in a structure that can be dynamically changed. In this structure, experiences are generalized and similar experiences are grouped together under the same generalization. This model was employed in a computer system called CYRUS by Kolodner [70], a query-answer system that is able to answer queries about diplomatic meetings and incorporate new knowledge in this context. Other work that lead to

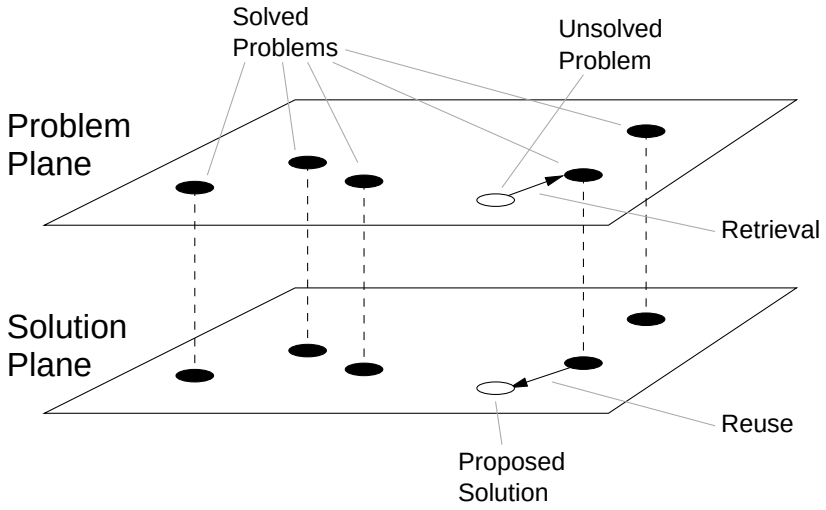


Figure 3.1: Problem solving in CBR.

the emergence of CBR in the field of problem solving was Gentner's theory of *structure mapping* [45]. His theory states that analogy is a structure-preserving mapping between different domains. This is particularly relevant for CBR, since the inference of a solution for a problem in CBR typically depends on analogies of the problem with previously solved problems.

3.2 Why CBR?

What are the advantages of CBR over other problem solving approaches? Kolodner [71] mentions (among others) the following advantages:

Firstly, CBR allows generation of solutions quickly, because adapting old solutions to fit a new case is often less time consuming than constructing a solution from scratch.

Secondly, with CBR it is possible to generate good solutions in domains that are hard to fully comprehend. The lack of comprehension is not a problem as long as the CBR system contains a case with a problem that is sufficiently similar to the current problem, so that the uncomprehended parts of the domain are in the parts of the solution that can be reused without adaptation.

Riesbeck and Schank [101] related the use of cases in problem solving to the use of rules. They noted that the actual process of problem solving as human experts experience it, is hardly driven by rules. This is conveyed by the fact that the rules experts can formulate often have many exceptions. Rather, experts motivate their choices in the problem solving process by

referring to solutions to earlier problems. Riesbeck and Schank refined the relation between cases and rules by proposing that rules are formed when many cases the expert has seen are consistent in some respect. As long as there is not enough consistency between cases to form a rule, the cases will be retained in memory as individual cases.

Related to the above is the issue of psychological plausibility. Ross [103] demonstrated by experiments that when novices learn in a formal domain like algebra, examples that illustrate general principles play a crucial role, and that the capability to solve a new problem is largely dependent on the subjects' capability to recall an example that is similar to the new problem.

Aamodt and Plaza [3] mention the fact that in CBR, as opposed to most other problem solving approaches, it is possible to employ (or gather) both specific knowledge (about individual problem cases) and general knowledge. Branting [15] studied this combination of general knowledge and specific cases in a CBR system. Another example of the combined use of general and specific knowledge is CASEY [73], a medical diagnosis system where cases were used in combination with a causal model of the domain (heart failure).

A final advantage of CBR, previously mentioned in [31], is that the (generalized) cases from the case base that are matched to the problem may serve as an explanation for the solution that were proposed for the problem. Such explanations will be probably more intuitive than a problem solving trace from e.g. a rule based expert system.

3.2.1 Lazy versus Eager Learners

To determine the position of CBR in the spectrum of problem solving/learning techniques, it may be useful to adopt a more technical, machine learning perspective. In the field of machine learning, learning algorithms are divided into the categories of *eager learners* and *lazy learners*. In the first category of algorithms, learning (or generalizing) takes place at the stage of training (when the system incorporates the data for which the solution is known). The lazy learners, on the other side, only learn or generalize from the training instances when a new problem must be solved. As a result of this, eager learners construct one global approximation of the *target function* (the function that maps problems to corresponding solutions, and which the learner is supposed to implement), whereas lazy learners create many local approximations of this function. This makes lazy learners more apt for approximating complex target functions.

The difference between eager and lazy learners is also reflected in the computational effort of the different stages: eager learners are usually slow when training because the target function approximation is constructed then, but

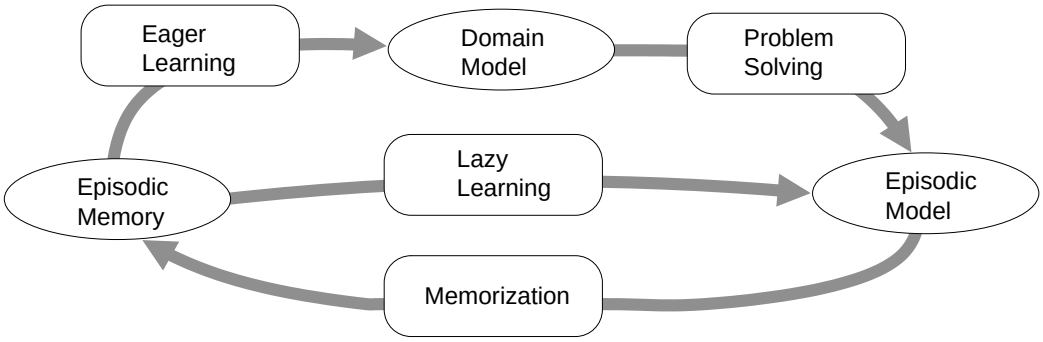


Figure 3.2: Lazy and eager learning and their roles in the use and construction of memory.

once the approximation has been made solutions for problems can be found relatively fast. Contrarily, lazy learners are relatively fast when training since the training instances are just stored without further action (although a sophisticated CBR system might require some effort to fit the data into a memory structure in the appropriate way), but slow when solving problems (since an approximation of the target function must be constructed *ad hoc*).

Figure 3.2 shows how lazy and eager learners use the episodic memory, or training instances (the solved problems to learn from), to solve a new problem. Eager learners (for example *Decision Tree Learners* such as *ID3* and *C4.5*) generate a domain-model based on the entire set of training instances. This is a general model that will be used for every new problem to be solved. The problem solving process, that is the construction of an episodic model for the new problem (the model that defines a solution for a particular problem) is usually a rather trivial task once a domain model has been generated. Lazy learners instead postpone the construction of an episodic model to the point at which the new problem is actually known. The lazy learner uses the problem specification to select the relevant portion of episodic memory, which is then used to construct the episodic model for the new problem. Once the problem has been solved, the problem and solution can be memorized by incorporating it in the episodic memory.

3.3 The CBR-cycle

After these general remarks about CBR, in this section we will give an overview of the main tasks in the process of problem solving with CBR, following the framework described in [3]. According to this framework, solving a new problem in a CBR system involves four major tasks:

Retrieve Pick earlier solved problems from the case base, that are similar to the problem to be solved currently.

Reuse Adapt (or construct) an initial solution for the current problem, based on the solutions of the retrieved problems.

Revise Test the proposed solution for adequacy (typically by user feedback) and if necessary revise the solution to meet the requirements.

Retain Store the problem, together with its revised solution, in the case base for future use.

Figure 3.3 shows how the four CBR tasks are organized within the problem solving process. For generating a solution, a CBR system needs must at least provide methods for the Retrieve and Reuse tasks. The Revise and Retain tasks are necessary for storing proposed and corrected solutions as cases in the case base, thus enabling learning.

After a subsection about the representation of cases in CBR systems, which is an implicit but very important aspect of CBR, each of the four tasks will be treated more in detail.

3.3.1 Case Representation

An important aspect of CBR system design is the representation of the cases. In the first place the representation of information in the cases constrains the way in which cases in the case base can be used in the various CBR tasks. A case could be as simple as a vector of attribute-value pairs comprising information about the problem and the solution (which is a common representation in Nearest Neighbor Learning, see page 30). This kind of representation is called a *flat* representation. In most CBR systems however, a case contains extra information in addition to the problem description and the solution, that specifies for example the dependencies of certain aspects of the solution on certain features of the problem description, or how the solution was obtained. The latter touches on another aspect of case representation, that is closely related to the methods used for the Reuse task (see section 3.3.3). That is the question whether a case should contain the outcome of the reasoning process (the solution), or rather a *problem solving trace*, (that is, a description of how the solution was obtained), or a combination of solution and trace. Such traces may offer hints on how the solution can be constructed for the input problem.

Besides the question how the cases should be represented, an important issue is the representation/organization of the case base as a whole. Especially

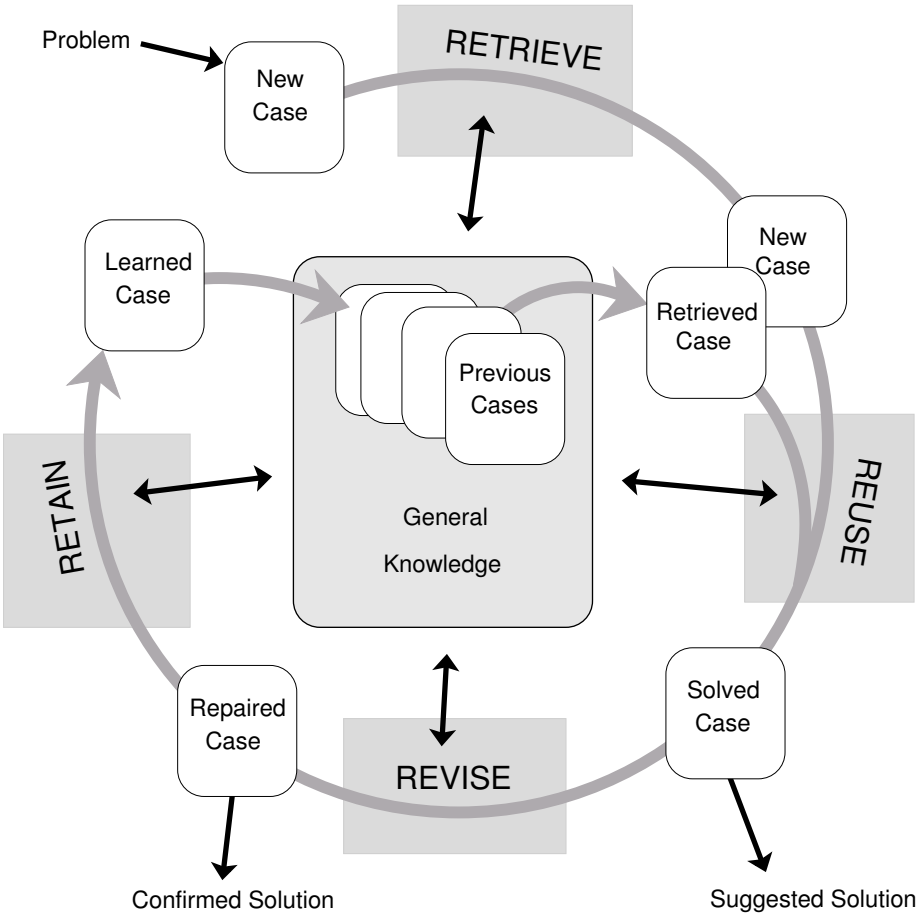


Figure 3.3: The Case Based Reasoning cycle, which shows the relations among the four different stages in problem solving: Retrieve, Reuse, Revise, Retain.

the Retrieve task relies heavily on the case base organization for effectivity and efficiency of the task (although all tasks can potentially involve interaction with the the case base). A common way of structuring the case base is based on Schank's dynamic memory model mentioned above [104, 70]). In this approach the case base is represented as an indexical tree-structure where the retrieval of cases takes place by traversing the tree from the root to a leaf, at each node taking the branch that has input features corresponding to those of the current problem. This approach has been rejected by Thagard and Holyoak [114], mainly because of its lack of psychological plausibility. In turn, they propose a theory for analogy that allows for making inter-domain mappings (rather than intra-domain mappings, as is usual in CBR). In their theory mapping is a result of semantic, structural and pragmatic constraints [61].

3.3.2 The Retrieve Task

The Retrieve task consists in finding a matching case, given a problem (or rather, an input description, specifying the problem situation and the requirements that a solution should fulfill). Retrieval methods can range from matching cases with only superficial similarities to the problem description, to matching based on a semantic analysis of the input description. The latter case typically requires a large amount of domain knowledge and is therefore called Knowledge-Intensive CBR (KI-CBR, see section 3.4). The Retrieve task in general can be divided into three subtasks: identify, match and select, which will be described in turn.

Identify

The aim of this subtask is to identify relevant features of the input description. In a "knowledge-poor" approach, the features of the input description might be just the input description itself (or some part of it). A "knowledge-intensive" approach might involve deriving abstract, or higher level features from the input description, discarding noisy parts of the description, and predicting unknown features of the input description, by using domain-knowledge.

As an example of the Identify subtask we can see how this task is performed in Tempo-Express [52], a CBR system for tempo-transformations preserving musical expression that we have developed (see chapter 5). In this system, the input problem description is formed by a representation of a melody, a performance of that melody at a particular tempo (T_i), and a desired tempo at which the melody is to be performed (T_o). The prob-

lem descriptions of the cases in the case base also contain a melody and performances at various tempos. To generate a performance of the input melody at the desired tempo, similar melodies are retrieved from the case base, whose case contains a performance at tempo T_o. So in the Identify subtask the availability of a performance at the desired tempo is used as a first (knowledge-poor) criterion for matching. Secondly, the melody is analyzed using a musical model [87] (see sections 3.5.1 and 5.3.2). The resulting analysis can be regarded as an abstract representation of the melody. Thirdly, an annotation of the input performance (an elaborate mapping of the performance to the input score), is constructed to serve in the matching task described below.

Match

Once the relevant features have been derived from the input description, an initial match can be made from the input description features to cases in the case base. Of course, the way this match is made depends on the way the case base is organized. If the case base is structured, as for example a dynamic memory (see subsection 3.3.1), the structure can be used to retrieve matching cases. If the case base does not have such a structure, an alternative is to define a similarity metric between problem descriptions, so that the similarity can be computed between the input description and the problem descriptions of the cases in the case base. The most similar cases can then be selected.

In case the match between input description and cases is made based on a subset of the description features, an additional test is needed to check whether the matched cases are really relevant, since the features that were disregarded in the matching may be different and make the solutions of the matched cases irrelevant for the current problem.

For an example of matching, we consider Tempo-Express again. In Tempo-Express, the cases (melodies with a number of performances at different tempos) in the case-base are not structured according to an indexical structure. Instead, similarity measures are used to retrieve relevant cases. After a first filtering of the case base, to exclude cases that do not have a performance at the desired output tempo. Similarity between the melodic analysis of the input melody and the analyses of the melodies in the case base serves as the main criterion for matching.

Select

When the matching of cases as described in the previous subsection resulted in more than one matching case from the case base, a method is needed

to select the best matching case or to rank the cases according to their relevancy. This method can be integrated with the initial match method, but it can also be treated as a separate method. The Select task usually involves a more detailed matching of the problem description of the cases to the input description. Ideally, the method should prefer those cases whose solutions offer the best starting point for generating the solution for the input problem. This can be done by assessing how relevant the differences between input description and problem description of the matched case are, for example by weighting the description features or the discriminatory power of the features.

To continue the example from the previous subsection, we consider the selection, or rather ranking task as performed in Tempo-Express. The ranking is performed upon a number of matching cases returned by the match task. For each case the performance available at the input tempo T_i is compared to the input performance by using a similarity measure for performances. The cases are ranked according to these similarities. The performance at T_o of the best matching cases is taken as the input for the Reuse task.

3.3.3 The Reuse Task

The generation of the solution for the input problem based on the retrieved case(s) is defined as the Reuse task (also called *adaptation*). Since Reuse is generally considered a rather difficult task, that involves thorough knowledge of the problem domain, it is sometimes circumvented by increasing the size of the case base until most problems are represented in the case base. In that case, a solution for a new problem can be transferred literally, that is without adaptation, from the most similar (or preferably, identical) case in the case base to the new problem. Obviously, this approach is not feasible in most real life situations, since the number and variety of cases is too large. For these situations Reuse is a way of making problem solving tractable in two opposite ways. Firstly, it reduces the need for a huge case base that covers all conceivable problems and their solutions, since parts of the problem space can be covered by interpolating (through Reuse) from solutions of nearby problems. Secondly, it serves to reduce the computational effort needed to construct a solution (when compared to non-CBR approaches), by offering ways to use existing solutions to similar problems as a starting point, instead of starting from scratch.

In the Reuse task, the central question is whether (parts of) the retrieved solution(s) can be reused in the new solution, and if so, whether those solutions need to be adapted to fit the requirements of the input description. There are several frameworks for building a new solution based on retrieved

solutions (see [22, 3, 123]). We will briefly describe them below. (Note that these methods are primarily used for constructing solutions in synthetic tasks (see page 30), since the solutions in analytic tasks are not composite structures)

Transformational Reuse This term is used for approaches in which the new solution is an adapted version of (one or more) old solutions [21]. In this approach, the retrieved solution is increasingly modified until a solution is obtained that satisfies the input requirements. Wilke and Bergman [123] distinguish between Substitutional Reuse and Structural Reuse. The former involves only changing values of attributes of the solution, leaving any structure intact. The latter is more a more rigorous kind of Reuse, that modifies the structure of the solution. To illustrate this, an example is given in [123], of a CBR system that proposes hardware configurations for personal computers based on the intended use (e.g. text-processing, playing games, audio-editing). An example of Substitutional Reuse would be the case where a the hard disk of a previous solution would be replaced by a hard disk with larger capacity, because the current problem description indicates a kind of use (e.g. audio- or video-editing) that requires much disk space. Here the PC configuration (viz. the structure of the solution) would be left intact. On the other hand, when the intended use is more different from the intended use in the retrieved cases, it is possible that a component must be added or removed from the retrieved configurations. This would be an example of Structural Reuse. This example shows that the two kinds of Transformational Reuse can be used as complementary methods, where Substitutional Reuse is used when the case base contains cases that match the new problem well, and Structural Reuse is used when a closely matching case is not available.

Generative Reuse In this kind of Reuse, the solution to the input problem is not a modified version of solutions from retrieved cases, but rather a newly constructed solution based on information from the retrieved cases. The most widely used type of Generative Reuse is known as *Derivational Analogy* [22]. In this approach, the solutions of retrieved cases are not reused directly for the construction of the new solution. Instead, the cases contain a *trace* of the process that lead to the construction of the solution. This trace is used as a guide for the construction of the new solution. Figuratively speaking, the problem solving process of the retrieved case is 'replayed' in the context of the new problem (therefore, the term *Derivational Replay* is also used to refer to this type of Reuse). The trace may contain information such as subgoals of the problem, alternatives that were considered, failed search paths, etc.

Another type of Generative Reuse, called *Constructive Adaptation*, was proposed by Plaza and Arcos [93]. This approach restates the Reuse task as a search problem: the construction of a solution is realized by searching through a space of partial solutions until a satisfying solution is found. To execute the Reuse task in this way, several components must be defined. Firstly, it must be possible to construct a *configuration*, a state that can be translated to a (partial) solution. Secondly, there must be an operator component that transforms configurations in one or more new (and typically more complete) configurations. Thirdly, an evaluation component must be defined that ranks a set of configurations. This ranking is done by evaluating the quality of the partial solutions corresponding to those configurations. Lastly, it must be possible to generate an initial (possibly empty) configuration, and a stop criterion must be defined, that decides whether a complete solution satisfies the input requirements of the problem. With these elements it is possible to find a solution by best-first search: The search is initiated by generating an initial configuration, as an initial search state. This state is then expanded by using the operator component to generate successive configurations. The set of new states is ranked according to solution quality. The state with the best partial solution is again expanded, and the new states are incorporated into the set of states that are to be expanded. This process is repeated until a configuration is found that satisfies the stop criterion.

We will explain CA in more detail in subsection 5.5.2

3.3.4 The Revise Task

The aim of this task is to assure the correctness of solutions that were generated during the Reuse task. The first part of this task is to evaluate generated solutions. Typically, this is done by the user of the CBR system, by trying out the solution in the real world. Based on this experience, the user may provide feedback on the adequacy of the solution. When the solution is not found to be satisfactory, this feedback is used to repair the solution, which is the second part of the Revise task. To be able to repair the solution, the shortcomings that were reported by the evaluator must be accounted for. When explanations are found for the shortcomings (this requires domain knowledge in the form of a causal model), the points of failure in the generation of the solution can be determined, and an alternative solution can be constructed. Note that this amounts to increased Reuse task knowledge. Thus, when the repaired solution is satisfactory (which is checked by re-performing the Revise task for the repaired solution), the general knowledge that is used when constructing new solutions can be adapted to avoid the generation of similar unsatisfactory solutions in the future.

3.3.5 The Retain Task

When a good solution has been obtained in the previous steps, a case may be constructed from that solution, to be stored in the case base, for re-use in future problem solving tasks. The first subtask is to *extract* relevant information for storing in the case. Obviously, the problem description should in some way be represented in the case, and also the solution. When derivational analogy is used as a Reuse technique, a trace of the problem solving process must be constructed to store in the case. The second task is to *index* the case, that is, to determine which are the features of the case to be used for retrieving it from the case base again. A straight-forward approach is to use all features of the case as indices. More sophisticated approaches would use some kind of filtering against the cases in the case base (for example by finding cases that have many features in common with the new case, and using the same features as indices that were used to index the similar cases). The last step is to *integrate* the case within the case base. When the case base is structured as a dynamic memory model (see page 34) the indices of the case serve as a guide to position the case within the tree structure. At the final node, a decision must be made to maintain the case separately, or, if the case is sufficiently similar to a number of other cases, to construct a *generalized case*, that represents what the subsuming cases have in common.

3.3.6 Case Base Maintenance

A task related to the retaining of new cases is the task of *case base maintenance*. This task concerns the state of the case base as a whole. One of the central issues in case base maintenance is the distribution of cases over the problem domain. Ideally, the case base should contain cases for that deal with all possible problem descriptions. Of course this is normally impossible, due to the virtually infinite number of problems in many domains. Also the size of the case base would be too large to deal with. The number of cases actually required to deal adequately with problems throughout the whole problem domain, depends partly on the approach taken in the Reuse task. When the Reuse step makes it possible to construct a solution for a problem by using solutions from rather dissimilar cases, this reduces the amount of cases needed for covering the whole problem domain. Of course this also depends on the problem domain itself. In domains where it is hard to gather knowledge needed for Reuse can only be solved if the case base contains a solution for a nearly identical problem. That is, in terms of the illustration in figure 3.1 (page 31), if the new problem is located close to an already solved problem in the problem plane.

Another issue is assuring the consistency of the case base. A case base inconsistency occurs when the case base contains two cases with the same problem description, but contradictory solutions. Obviously, different solutions may not always be contradictory, since in many domains (e.g. expressive music performance) there may be multiple valid solutions. But for analytical CBR tasks such as classification, assigning contradictory labels to the same problem is a realistic possibility. Assuming the correctness of the solution in both cases, an inconsistency may indicate an incorrect problem description (causing different problems to be represented in identical ways).

In a special issue of Computational Intelligence on maintaining case based reasoning systems [28], Wilson and Leake [124] propose a general framework for maintenance of CBR systems. To categorize maintenance policies, they divide the maintenance process into three phases: 1) data collection, 2) triggering, and 3) execution. During data collection, the maintainer gathers information about the state and performance. This information is used in a triggering phase to decide whether an actual maintenance operation should be carried out, and selects an appropriate operation from a set of possible operator types. In the execution phase, this operation is interpreted to the actual situation of the CBR system, to describe when and how the operation should be executed. The choices that are made in the general process of maintenance depend on the goals and constraints of the CBR system that is being maintained, such as problem solving efficiency goals, solution quality goals, size limits of the case base, or the expected distribution of future problems. Different combinations of goals and constraints lead to different maintenance policies. It should be noted that the maintenance does not apply solely to the case base itself, it may also operate on other knowledge containers, such as retrieval and reuse components of the CBR system.

3.4 KI-CBR

As mentioned in the beginning of this chapter, the spectrum of Instance Based Learning techniques contains rather simple techniques like Nearest-Neighbor Learning at one end of the spectrum. At the other end, we find the types of CBR that make thorough use of knowledge about the problem domain, the so-called Knowledge-Intensive CBR (KI-CBR). In this section we will focus on this kind of CBR.

In real-world problems (like medical diagnosis, investment planning, and geological planning, but also expressive music transformation), approaches to problem solving that select and adapt cases only using syntactical criteria are often not adequate. Rather, a semantical approach may be needed. In

order to treat case information in a semantical way, domain knowledge will be needed. This knowledge can take different forms, such as a set of rules, or a model describing the (causal) relationships between concepts in the domain.

Aamodt [2] noted that problem solving in real-world situations usually involves *open problems*, that is, problems for which the domain theory is weak or intractable. Relationships between concepts are often not fully specified, and involve uncertainty rather than clear truth or falsity. He states that knowledge based problem solving systems for solving this class of problems, must meet the following requirements:

1. It should have an *expressive* and *extendible* way of *representing knowledge*, that allows for explicit and thorough modeling of all relevant knowledge types.

2. It should have a *problem solving and reasoning method* that can combine reasoning from previous cases with the use of general domain knowledge.

3. It should have a *learning method* that can retain concrete problem solving experiences, so that the quality of knowledge and the problem solving performance will increase.

In order to realize systems that meet these requirements, Aamodt [1, 2] proposes a system architecture for KI-CBR systems (*Creek*). Actually, this architecture allows an integration of CBR and other problem solving methods like problem solving based on conditional rules (*Rule Based Reasoning*, or *RBR*), or based on a (causal) model of the problem domain (*Model Based Reasoning*, or *MBR*). The Creek architecture was designed with the class of diagnosis problems (which is an analytical task) in mind. Furthermore, diagnosis is a *declarative* problem solving task, meaning that the problem solving process takes place deliberately in the head of the human problem solver. Generating expressive musical transformations, the kind of problem solving on which we focus in this thesis, differs from diagnosis in both respects: it is a synthetic task (involving the construction of a solution as a composite structure), and it is of a *procedural* kind rather than declarative. That is, the knowledge used by the musician to perform an expressive transformation, is unlikely to be fully available for explicit reasoning. Rather, the knowledge relevant for such problem solving takes the form of theories that hypothesize about the cognitive process that underly music performance and listening (like the Implication/Realization model; we will return to this issue shortly). Although these differences are not trivial, some aspects of the Creek architecture may still be useful in our context. We will therefore mention some key-aspects of Creek.

Creek's knowledge representation is implemented using an knowledge base. This object knowledge base is a structure of *frames* (a notion proposed by Minsky [83]) that contains domain concepts (e.g. 'car', 'battery', and 'has-color') as well as problem solving and learning task concepts (such as 'symptom', 'case-index'). In addition to the concept definitions, the knowledge base contains a set of solved cases, and additional domain knowledge in the form of heuristic rules. Furthermore there is a problem solving component. This component handles new problems by trying to find frames in the knowledge base that match features in the problem description. In this way, the system tries to 'understand' the problem, by integrating it into the knowledge base. If the features give a reminding of a solved case in the knowledge base, and this reminding has a strength above a certain threshold, then a CBR approach will be tried to solve the problem. If no case is retrieved, or the strength value is too low, then a RBR approach is taken. The strength threshold parameter should be dependent on the ratio of the performance of the RBR and CBR subsystems respectively (e.g. if the system has a well-performing rule set at its disposal and a relatively small number of solved cases, the strength parameter is increased to favor RBR). If RBR fails as well, MBR could be tried as a resort.

The reasoning phase is preceded by a spreading of activation through the frame network: The problem feature frames and the concepts representing the goal are given an initial activation, and this activation is spread out along related frames. The 'paths' of activated concepts that connect the problem and goal concepts, form the set of concepts that is used in either CBR or MBR (whichever approach is chosen).

After a problem has successfully been solved, learning in Creek takes place by strengthening the remindings of the problem features to the case that was used to solve the problem. The newly solved problem is stored in the knowledge base by trying to generalize the case that was used for problem solving to subsume the new case. If the problem was solved using RBR or MBR, the problem and its solution are stored as a new case.

3.4.1 NOOS: a Knowledge Representation Language

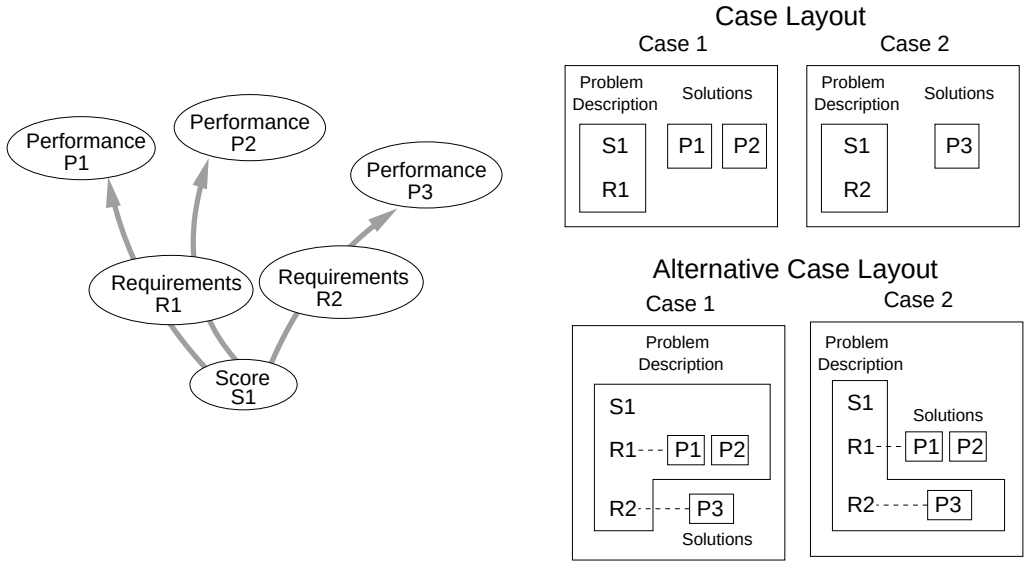
Arcos and Plaza [9, 7] have designed *NOOS* a language to support the development of knowledge intensive problem solving systems. The language allows the user firstly to specify his problem by defining an *ontology*, containing all the types of objects that figure in the problem. Secondly, a *domain model* is constructed to capture the facts about the problem domain. In a system that should diagnose car defects, the domain model would for example contain facts like ''. Thirdly, the user provides *problem solving methods*,

which use the knowledge of the domain model to find a solution to specific problems.

3.5 Application of KI-CBR for Expressive Music Performance

In this section, we will mention some aspects of the application of a KI-CBR approach in the context of expressive musical performances, and the role of musical modeling in this context.

An attractive aspect of CBR as a problem solving approach in general, is that there is no need to make explicit all knowledge of the problem domain in order to make use of it. In a CBR system, the knowledge is implicit in the solved cases, and by retrieving similar cases and reusing the solutions of those cases, this knowledge is 'transferred' to the solutions of new problems. In the context of a music performance generation system, an intuitive manner of applying CBR would be to view unperformed music (e.g. a score) as a problem description (possibly together with requirements about how the music should be performed) and to regard a performance of the music as a solution to that problem. It may be instructive to point out the relation between score (part of the problem description), performance requirements (part of the problem description), and performances (the solutions), as illustrated in figure 3.4 (a). Firstly, a score can be performed in many different ways. In a CBR system, such different performances would be specified by the performance requirements. The requirements could contain specifications such as the tempo, or the mood of the performance. Secondly, there will typically be multiple performances that satisfy a single set of requirements, where less specific requirements allow for more widely varying performances. This relation can lead to different layouts of cases in the CBR system, depending on the task of the system. If the task is to generate a performance from scratch, based on a score and performance requirements, the score and performance requirements form the problem description and the performance are taken to be a solution to this description. This layout is shown in figure 3.4 (b) top. Notice that the constellation shown in figure 3.4 (a) leads to two separate cases, since two different problem descriptions can be formed ($\langle S1, R1\rangle$ and $\langle S1, R2\rangle$, respectively, the first having two solutions, $P1$ and $P2$, and the second having $P3$ as a solution). If, on the other hand, the task of the CBR system is performance transformation rather than generation from scratch, the problem description will not only contain the score and performance requirements, but also the performances corresponding to all but one of the



(a) (b)

Figure 3.4: (a) The relation between score, performance requirements, and performances. (b) Two types of case layout, both representing the constellation shown in (a).

available performance requirements. In this way, the constellation shown in figure 3.4 (a) again leads to two separate cases, as shown in figure 3.4 (b) bottom.

Solving a problem then implies using the performance of a similar previously played melody to perform a new melody. An interesting question is which part of the musician's knowledge of how to perform a melody can be transferred from one problem to another, and what kind of knowledge is needed explicitly to actually realize such a transfer. The actual problem solving steps where explicit domain knowledge is needed, are the Retrieve and the Reuse steps (we leave the learning oriented steps Revise and Retain aside here). For the retrieval of similar problems, the distance between problem descriptions must be assessed. That is, we must be able to specify how similar two scores are, and how similar two sets of performance requirements are. For this distance assessment to be useful, it must return small distance values for problems where the solution of one problem can be easily used to construct a good solution for the other problem. This may imply that the distance assessments should not be done directly on the problem descriptions itself, but on a derived, more abstract representation of it (e.g. a musical analysis of the score). For effective reuse of a performance that was retrieved

based on similarity between problem descriptions, we need to know how the (expressivity of a) performance relates to the score and performance requirements. As we have seen in chapter 2, the nature of this relation is the topic of ongoing research and is far from being completely known. Nevertheless, some relevant aspects of musical expressivity are commonly agreed upon, e.g.:

- expressivity serves to clarify musical structure
- expressivity serves to express affective intentions

In the next subsections, we will review some types of musical domain knowledge that we believe will be useful in a CBR system for transformation and generation of music performances.

3.5.1 Music Analysis

The Implication/Realization Model

Narmour [87, 88] has proposed a theory of perception and cognition of melodies, the *Implication/Realization model*, or *I/R model*. According to this theory, the perception of a melody continuously causes listeners to generate expectations of how the melody will continue. The sources of those expectations are two-fold: both innate and learned. The innate sources are 'hard-wired' into our brain and peripheral nervous system, according to Narmour [87], whereas learned factors are due to exposure to music as a cultural phenomenon, and familiarity with musical styles and pieces in particular. The innate expectation mechanism is closely related to the *gestalt theory* for visual perception as proposed by Koffka [68] and Köhler [69] and for audition by Bregman [16]. Gestalt theory states that perceptual elements are (in the process of perception) grouped together to form a single perceived whole (a 'gestalt'). This grouping follows certain principles (*gestalt principles*). The most important principles are *proximity* (two elements are perceived as a whole when they are perceptually close), *similarity* (two elements are perceived as a whole when they have similar perceptual features, e.g. color or form, in visual perception), and *good continuation* (two elements are perceived as a whole if one is a 'good' or 'natural' continuation of the other). Narmour claims that similar principles hold for the perception of melodic sequences. In his theory, these principles take the form of *implications*. Any two consecutively perceived notes constitute a melodic interval, and if this interval is not conceived as complete, or closed, it is an *implicative interval*, an interval that implies a subsequent interval with certain characteristics. In

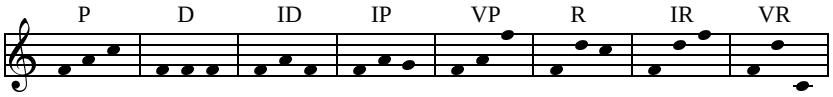


Figure 3.5: Eight of the basic structures of the I/R model.

other words, some notes are more likely to follow the two heard notes than others. 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 analogous for downward intervals), and large intervals imply a change in registral direction (a large upward interval implies another upward interval and analogous 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 intervals (seven semitones or more) implies a smaller interval.

Based on these two principles, melodic patterns can be identified that either satisfy or violate the implication as predicted by the principles. Such patterns are called *structures* and labeled to denote characteristics in terms of registral direction and intervallic difference. Eight such structures are shown in figure 3.5. The P structure ('Process') is a small interval followed by another small interval (of similar size), thus satisfying both the registral direction principle and the intervallic difference principle. The D structure also continues in the same direction (lateral) with a similar interval (the unison). The IP ('Intervallic Process') is a structure that satisfies the intervallic difference principle (since the small interval implied by the initial small interval is realized), but violates the registral direction principle (since the small interval implies a continuation in the same direction). Conversely, VP ('Vector Process') satisfies registral direction and violates intervallic difference. In the same way, the R ('Reversal') structure satisfies both principles, and has two counter-parts IR and VR where one of the principles is violated. The ID structure ('Identity'), appeals to another principle, *registral return*, that we will not discuss here.

Additional principles are assumed to hold, one of which concerns *closure*, which states that the implication of an interval is inhibited when a melody changes in direction, or when a small interval is followed by a large interval. Other factors also determine closure, like metrical position (strong metrical positions contribute to closure, rhythm (notes with a long duration contribute to closure), and harmony (resolution of dissonance into consonance contributes to closure).

According to the I/R model, the above principles are innate, and govern human perception of melodies in a data-driven, *bottom-up* way, as they are claimed to emerge from the anatomical organization of our auditory system. As a result, the validity of the principles is assumed to hold throughout all human cultures. In addition to these innate principles, Narmour assumes principles of another nature, these principles are learned, and arise through our experience with previously heard music. This experience is bound to personal history and therefore culturally dependent. For instance, Western music is governed by the principle that the melody normally adheres to a single key (save exceptional circumstances such as modulation). But also more specific contexts such as a musical style may create expectations about continuation in the listener, when he/she is familiar with the musical genre to which the heard piece is known to belong. At an even more specific level, a musical piece itself may have it's idiosyncrasies that become apparent to the listener during listening. In this case, the implied continuations may be influenced by the particularities of the musical piece as well. This factor is called *intra-opus style* by Narmour, whereas genre and cultural typicalities are called *extra-opus style*. These style specific factors are assumed to interfere with perception in a *top-down* manner, as the style implications stem from the memory of the listener.

The I/R model states that both bottom-up and top-down processes act at the same time when music is heard. The final perceived implication (or inhibition of implication) at a given point in the melody, is determined by the interaction of all principles governing the bottom-up process, plus the interfering influence of the top-down process. Bottom-up principles can reinforce each other, but also cancel each other out. For example, when a melody is accompanied by a dissonant harmony on a strong metrical position (e.g. the first beat of a measure), there is closure in meter, and non-closure in harmony. This creates an ambiguity in musical structure. On the other hand when a dissonant chord on the last beat of a measure is followed by resolving consonant chord on the first beat of the next measure, a strong closure occurs, since both meter and harmony imply closure. This inference may be inhibited when within the piece or a musical genre it is a common pattern that e.g. a musical motif ends on a strong metrical position, or alternatively, ends with a dissonance. In this case, the closural effect of these phenomena does not occur (given that the listener is familiar with such idiosyncratic patterns). From this example, Narmour's conception of bottom-up processes as the application of rules, and top-down processes as imposing exceptions to those rules, becomes apparent.

Based on this model of implication and realization, it is possible to 'parse' or 'analyze' a melody, generating a sequence of structures. Figure 3.6 shows

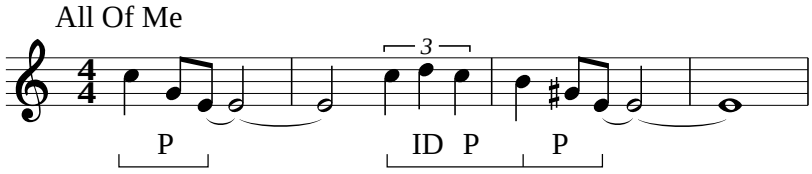


Figure 3.6: First measures of 'All of me', with corresponding I/R structures.

such an analysis for the first measures of the jazz song 'All of me'. Assuming that the violation or realization of implications is an essential aspect of the way humans listen to music, this sequence of structures is would reflect the listener's representation of the melody.

Interesting experiments have been carried out by Schellenberg [105, 106] and Cuddy and Lunney [29], in which they presented subjects with either a melody [105, 106] or a single melodic interval [29] and asked them to rate different continuations. These ratings were compared with predictions of the I/R model (where the effect of the different I/R principles was quantized by the authors in accordance with the model). It turned out that the I/R model was able to predict listeners' ratings well above chance level. A principal-component analysis of the five I/R principles under consideration (registral direction, intervallic difference, proximity, closure and registral return) showed that the model was redundant, in the sense that three of the principles were highly correlated. Additional experiments were done with revised versions of the principles. Eventually Schellenberg [106] proposed a simplified version of the I/R model containing two principles, of which the first corresponds to revised versions of registral direction/registral return and the second to proximity. The *proximity principle* states that listeners prefer smaller realizing intervals to larger realizing intervals, i.e. small intervals are more implied than large intervals, regardless of the size of the implicative interval. The *pitch-reversal principle* (Schellenberg's term for the linear combination of the revised registral direction and registral return principles) states that listeners prefer the second tone of the realizing interval to be close to the first tone of the implicative interval. As Schellenberg points out, this principle extends the proximity principle to tones that are non-contiguous. Schellenberg found this simplified I/R model to predict listeners responses better than the original I/R model proposed by Narmour.

3.5.2 Melodic Similarity

In a CBR system where problem descriptions contain melodies, the concept of melodic similarity will in some form or other have to play a role in the

Retrieve task of the system. The retrieve task of such a system will basically be stated as: 'given a particular melody in the problem description, find a (set of) cases with a similar melody' (although besides the melody, other parts of the problem description may also be involved in the case retrieval). Thus, we will in this section focus on melodic similarity. We will mention some conceptual issues, and review the main formal/computational approaches to melodic similarity that have been proposed.

A Cognitive Theory of Melodic Similarity

Deliège [33] proposed a hypothesis about the cognitive process underlying human similarity judgments of music. In brief, Deliège states that when humans hear music, they abstract from what they hear by means of *cues*. She defines a clue as 'a salient element that is prominent at the musical surface' ([34], page 237). This concept is akin to Gestalt psychology, since it conceives of the perception of music as having a background (the normal flow of music) and a foreground (salient elements in this flow). Memorization is realized through the cues, which are the parts of the music that are stored in the listener's long-term memory. The cues serve as a 'summarization' of the music.

The similarity judgment between two fragments of music, according to Deliège, is based on the similarity (or identity) of clues. This is implied by the fact that music is memorized as clues, since similarity judgments involve the listeners memory of the music (the fragments of music compared are not heard at the same time, and may not be heard at the moment of judgment at all). In particular, the perception of different musical fragments as *variations* of each other, is thought to be caused by the fact that the fragments give rise to identical clues. In this way, the concept of clues can form the basis of a theory of categorization of music, where musical motifs can belong to the same category based on the perceptual cues they provide, with varying degrees of prototypicality.

Empirical Studies of Melodic Similarity Perception

Lamont and Dibben [74] have performed listening experiments to investigate melodic similarity as perceived by humans. They were interested in questions such as: Do listeners perceive similarity while listening? Is perceived similarity based on deep, thematic relations between the musical fragments, or rather on shared surface attributes of the music? Do listeners employ the same similarity criteria across different styles of music? Are the employed criteria the same for listeners with different levels of musical experience?

The listening experiments involved both trained musicians and non-musicians as subjects. They were asked to rate the similarity of pairs of extracts from two musical pieces (one from Beethoven and one from Schoenberg, a dodecaphonic piece). In addition, they were asked to rate for each extract the appropriateness of a set of adjectives, along a bi-polar scale.

The main conclusions from the experiment were the following:

- Similarity perceptions in general (i.e. across musical styles) are based on surface features: e.g. contour, rhythmic organization, texture, and orchestration.

- Both tested musical styles seem to have their own intrinsic subset of criteria, by which listeners rate similarity. For Beethoven, these were mainly dynamics, articulation and texture. For Schoenberg they were mainly tempo and dynamics.

- There are slight differences in the similarity ratings of musicians and non-musicians. Non-musicians seem to base their similarity judgments mainly on the final parts of extracts.

The most remarkable conclusion is that 'deep' structural relationships, that are often regarded as determining for the similarity of musical motifs, do not seem to have a significant impact on listeners judgments on similarity of musical motifs.

Melodic Similarity as Earth Mover's Distance

The earth mover's distance approach [26] was applied to melodic similarity by Typke et al. [118]. It treats melodies as a set of weighted points in a two dimensional space, where the pitch and onset time are taken as the coordinates and the duration of notes corresponds to the weight, or mass of the point. When two melodies are compared, one melody (the lighter, in terms of summed weight) is assigned the role of supplier, and the other (the heavier) the role of receiver. In the earth mover metaphor the supplier melody (in the form of weighted points) is construed has heaps of mass at particular places (where the the quantity is the weights of the points), and the receiver melody is regarded as a number of holes in the ground, with volumes corresponding to the weights. The transportation distance is defined as the minimum amount of work needed to move all the the mass into the holes. The amount of work for a given transportation of mass from the heaps to the holes is defined as the average distance traveled per unit of mass. The earth mover's distance can is formalized as follows: Let $\mathbf{x} =$

$\{\langle x_1, w_1\rangle, \cdots \langle x_m, w_m\rangle\}$, and $\mathbf{y} = \{\langle y_1, u_1\rangle, \cdots \langle y_n, u_n\rangle\}$ be the weighted point sets corresponding to two melodies, where $x_1, \cdots, x_m$, and $y_1, \cdots, y_n$ are the coordinates, and $w_1, \cdots, w_m$ and $u_1, \cdots, u_n$ are the corresponding weights. w^S is the total weight of $\mathbf{x}$, and u^S is the total weight of $\mathbf{y}$. f_{ij} denotes the flow of mass from element x_i to y_j. $F(\mathbf{x}, \mathbf{y}) = (f_{ij})$ is the set of all feasible flows, that satisfy the following conditions:

$$f_{ij} \geq 0, \text{ for } i = 1, \cdots, m, \ j = 1, \cdots, n \tag{3.1}$$

$$\sum_{j=0}^{n} f_{ij} \leq w_i, \text{ for } i = 1, \cdots, m \tag{3.2}$$

$$\sum_{i=0}^{m} f_{ij} \leq w_j, \text{ for } j = 1, \cdots, n \tag{3.3}$$

$$\sum_{j=0}^{n} \sum_{i=0}^{m} f_{ij} = \min(w^s, u^s) \tag{3.4}$$

Then the earth mover's distance $\text{EMD}(\mathbf{x}, \mathbf{y})$ is defined as:

$$\text{EMD}(\mathbf{x}, \mathbf{y}) = \frac{\min_{\mathcal{F} \in F(\mathbf{x}, \mathbf{y})} \sum_{j=0}^{n} \sum_{i=0}^{m} f_{ij}}{\min(w^s, u^s)}$$

Melodic Similarity as edit distance

The edit distance is a commonly used method for approximate matching in various domains, such as genetics (used for comparing DNA sequences), and text-analysis (used for comparing text strings). The method was proposed by Levenshtein [80] (hence, the distance measure is also called *Levenshtein distance*).

In general, the edit distance between two sequences can be defined as the minimum total cost of transforming one sequence (the source sequence) into the other (the target sequence), given a set of allowed edit operations and a cost function that defines the cost of each edit operation. The most common set of edit operations contains insertion, deletion, and replacement. Insertion is the operation of adding an element at some point in the target sequence; deletion refers to the removal of an element from the source sequence; replacement is the substitution of an element from the target sequence for an element of the source sequence. Mongeau and Sankoff [85] have described a way of applying this measure of distance to (monophonic) melodies (represented as sequences of notes). They propose to extend the set of basic operations (insertion, deletion, replacement) by two other operations

that are more domain specific: *fragmentation* and *consolidation*. Fragmentation is the substitution of a number of (contiguous) elements from the target sequence for one element of the source sequence; conversely, consolidation is the substitution of one element from the target-sequence for a number of (contiguous) elements of the source sequence. In musical variations of a melody for example, it is not uncommon for a long note to be fragmented into several shorter ones, whose durations add up to the length of the original long note.

The minimum cost of transforming a source sequence into a target sequence, can be calculated relatively fast, using a dynamic programming approach. Using this approach, the distance between two sequences a and b is calculated by computing the distance between all prefixes of a and b respectively. The distance between a particular prefix of a and a prefix of b can be efficiently calculated from smaller (previously calculated) prefixes. This can be seen from the recurrence equation that defines the distance d_{mn} between the sequences $a = \langle a_1, a_2, \cdots, a_m \rangle$ and $b = \langle b_1, b_2, \cdots, b_n \rangle$:

$$d_{ij} = min \begin{cases} d_{i-1,j} + w(a_i, \emptyset) & \text{deletion} \\ d_{i,j-1} + w(\emptyset, b_j) & \text{insertion} \\ d_{i-1,j-k} + w(a_i, b_{j-k+1}, ..., b_j), \ 2 \leq k \leq j & \text{fragmentation} \\ d_{i-k,j-1} + w(a_{i-k+1}, ..., a_i, b_j), \ 2 \leq k \leq i & \text{consolidation} \\ d_{i-1,j-1} + w(a_i, b_j) & \text{replacement} \end{cases}$$

for all $1 \leq i \leq m$ and $1 \leq j \leq n$, where m is the length of the source sequence a and n is the length of the target sequence b. Additionally, the initial conditions for the recurrence equation are are:

$$\begin{aligned} d_{i0} &= d_{i-1,j} + w(a_i, \emptyset) \quad \text{deletion} \\ d_{0j} &= d_{i,j-1} + w(\emptyset, b_j) \quad \text{insertion} \\ d_{00} &= 0 \end{aligned}$$

For two sequences a and b, consisting of m and n elements respectively, we take d_{mn} as the distance between a and b.

The weight function w, defines the cost of operations, such that e.g. $w(a_4, \emptyset)$ returns the cost of deleting element a_4 from the source sequence, and $w(a_3, b_5, b_6, b_7)$ returns the cost of fragmenting element a_3 from the source sequence into the subsequence $\langle b_5, b_6, b_7 \rangle$ of the target sequence. For computing the distances between the contour and I/R sequences respectively, the clauses corresponding to the cost of fragmentation and consolidation are simply left out of the recurrence equation.

Computing Similarity by compression

Cilibrasi et al. [23] have introduced a distance measure to the area of musical similarity that is based on compression. The core idea is that two musical fragments are similar to the extent that one fragment can be efficiently compressed by using the other fragment. Ideally, the efficiency of compression would be measured by the *Kolmogorov complexity* of the fragments. This quantity, denoted as $K(x)$ for a fragment x, is equal to the length of the smallest string of bits that allows exact reconstruction of the original fragment. Unsurprisingly, the Kolmogorov complexity of a piece of information is not effectively computable. Hence, Cilibrasi et al. take the length of compressed fragments using existing compression algorithms (such as 'gzip' and 'bzip2') as an approximation of the Kolmogorov complexity.

To compute the distance distance between two fragments, say x and y, the quantities $K(x|y)$ and $K(y|x)$ are needed, in addition to $K(x)$ and $K(y)$. $K(x|y)$ denotes the the length of the smallest string of bits to reconstruct x from y. The distance between x and y is defined as:

$$d(x,y) = \frac{max\{K(x|y), K(y|x)\}}{max\{K(x), K(y)\}}$$

Cilibrasi et al. have used this metric to compute the distance between a number of musical fragments, in the form of MIDI files. These files are preprocessed to generate internal representations of the fragments comprising only note-on and note-off information, and using a pitch-invariant representation of the melody. Pitch-invariance is obtained by representing pitches within a fragment as offsets from the modal pitch of that fragment. With this setting, distance matrices were computed for various sets of musical fragments. From these matrices, *phylogeny trees* were created (using a clustering technique called the *quartet method*), to visualize the distances. In the trees, fragments rated similar are spatially close. In this way, Cilibrasi et al. were able to distinguish both between different musical styles (e.g. jazz, classical, rock) and between different composers within one style. This is surprising, since the distance algorithm itself does not contain any music specific knowledge; it is said to be a 'universal' distance. The authors argue this approach is preferable over more 'ad hoc' approaches to musical similarity, since it springs from a well-founded mathematical idea. Despite the appeal of this argument, a theoretical draw-back can also be conceived. The total lack of context specificity of the distance makes it possible to compare anything to anything (provided the objects under comparison have a digital representation). Thus, the algorithm will unproblematically compute the distance between a musical piece and, say, a graphical image. However, such

comparisons would be commonly thought of as meaningless, and hence it is questionable whether this distance measure really captures what humans would consider crucial to comparison. This is related to the fact that the universality of the algorithm is achieved by comparing *representations* of objects, rather than the *content* of the objects. Whereas a music-specific distance algorithm could rate a melody represented as a raw wave file as very close to a melody represented as a MIDI file (provided the data is interpreted in terms of musical concepts, like pitches and duration), the compression based distance will most probably rate the objects as very different, since in terms of representation (viz. the file format) they *are* very different. From this point of view, universality is not necessarily a desirable attribute for a distance measure.

Chapter 4

Research Goals

4.1 The Broad Perspective

In a very global manner of speaking, this research could be characterized as an attempt 'toward machines that deal with musical expression'. We envision machine processing of musical expression primarily as having a conceptual framework in which common (high-level) concepts of expressive musical performance are related in a coherent way to each other, to lower level concepts, and eventually to actual music performances. By 'high-level' concepts we refer to for example 'emphasizing musical structure', or 'expressing an affection'. Such concepts are very general and can be realized by more concrete concepts, such as 'emphasizing metrical structure' or 'emphasizing phrasal structure' (which are kinds of emphasizing musical structure), and 'playing tenderly' (which is a kind of expressing an affection). Conceivably, these concepts can in their turn be broken down into even more concrete concepts, thus gradually approaching a level of performance description concepts that is tangible enough to work with in the context of machine learning techniques and automated reasoning. Figure 4.1 shows a small set of concepts of music performance, and their relations (where the arrows denote 'kind-of' relations). Obviously, this diagram shows only part of the inter-relations between concepts. For instance, it is quite possible that when a piece should be played to express sadness, the key of the piece is changed from major to minor in order to achieve that. Or when a piece should be played fast, the emphasis on rhythmical details is shifted toward an emphasis on overall phrase structure. When the inter-relations between concepts are captured to a sufficiently accurate extent, it may be possible to perform higher level transformations as compositions of several lower level transformations. However, with the current level of understanding of how the various concepts of

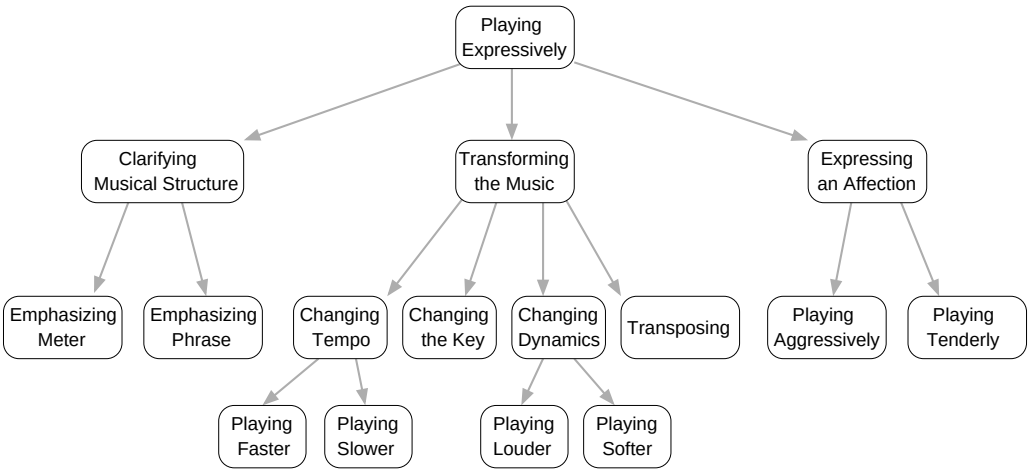


Figure 4.1: Some basic music performance concepts and their relations.

musical expressivity hinge together, it seems not yet possible to do this.

Ultimately, to realize expressive transformations, the machine must manipulate basic note attributes, like timing, duration, articulation, dynamics, timbre etcetera. It must do so driven by the higher level expressive concept that is to be realized, together with the characteristics of the music that is being processed, like its rhythm, meter, and phrase structure. For example, for a particular rhythm, the machine must know how to alter note attributes in order to emphasize that rhythm.

In this perspective, it becomes clear how the different branches of music performance research that were discussed in chapter 2, can contribute in complementary ways to the goal of machine understanding of musical expressivity. The approach of the research group at NICI aims at computational models of music cognition. Such models might clarify how different concepts of musical expressivity are related. The analysis-by-synthesis approach taken by the KTH group yields sets of rules that prescribe how to play particular music in an expressive manner. Using this approach, an interesting possibility is to formulate meta-rules, that prescribe which rules to use (and to which degree) for obtaining specific forms of expressivity. ÖEFAI employs a purely data-driven machine learning approach to discover similar kinds of rules – rules that prescribe how to adapt the nominal values note attributes in order to play music expressively. The knowledge obtained through the latter two approaches can be useful to translate higher level expressive transformations to transformation of the actual score.

4.2 Focus of Attention in This Research

It is in the plan outlined above that we hope to contribute with this research. To bring the research goals down to a proportion that is appropriate for Ph.D. research, we have chosen to focus on a specific kind of expressive music processing, namely global tempo transformations of music performances. This refers to the problem of how a performance played at a particular tempo can be rendered at another tempo, without the result sounding unnatural. Or, differently stated, how does expressiveness change with global tempo? Some research on this topic was reviewed in section 1.2, and it was concluded that tempo-transformations are not merely a uniform scaling of the time axis. Furthermore, we have restricted the scope of application to monophonic jazz recordings. The main reason to work with monophonic recordings is to facilitate the process of analyzing and representing the performances. While being far from trivial, the process of describing the content of monophonic audio is less complicated than describing the content of, and extracting melodic material from polyphonic audio. The reason to focus on jazz music, is twofold. Firstly, in jazz music, there is a large corpus of monophonic melodies, that are widely known and performed. This music is not strictly monophonic in the sense that it usually includes a harmonic and rhythmic accompaniment. But the melody is almost always distinguished clearly from the accompaniment, as opposed to many classical music, where melody and harmony are often interleaved (this difference is reflected in the fact that in jazz music, the harmonic accompaniment is usually notated by annotating the staff with chord symbols, rather than spelling the chords together with the melody on the staff). Secondly, we feel that jazz music has gained very little attention in the area of music performance research, since most studies focus on the performance of classical music. There may be explanations for this (see chapter 2), but there are also some strong reasons not to neglect jazz music. For example, jazz music appears to have a strongly idiosyncratic paradigm of expressiveness, that can be characterized by a liberal interpretation of notated music, leaving much room for creativity and improvisation. Also, the dimensions in which the expressivity is manifest are different from those of classical music (e.g. the classical musician strongly uses local tempo as a means of expressivity, whereas the jazz musician is bound to a steady rhythmical accompaniment, and consequently, his means of expressivity is rather the timing of the melody with respect to the rhythm).

Respecting the above restrictions, we aim at the development of a Case Based Reasoning system that is able to apply global tempo transformations to melodic descriptions of monophonic jazz performances (our choice for a CBR approach was motivated in section 3.5). Together with audio analysis

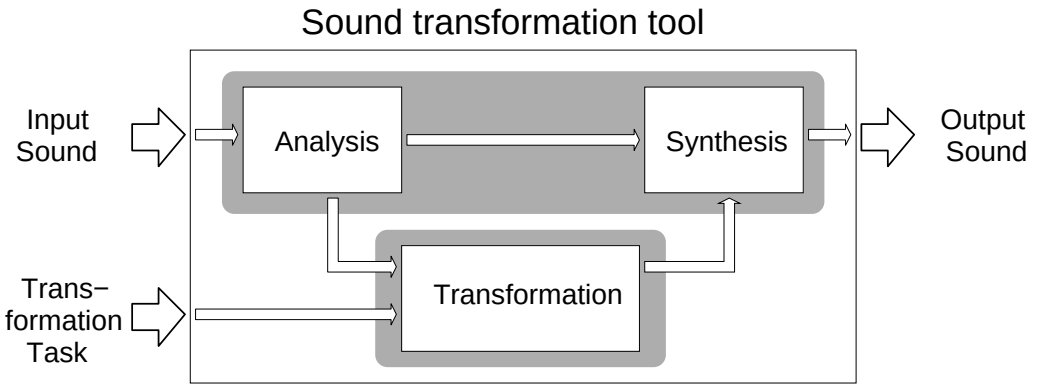


Figure 4.2: Diagram of the system architecture

and synthesis tools that are being developed within the *Tabasco*, and *ProMusic* projects, this system should constitute a software tool for the transformation of audio files containing monophonic musical content. The tool thus consists of two subsystems, one of which handles the transformations between audio and melodic descriptions, and the other subsystem deals with the transformation of one melodic description into another, given a particular tempo transformation task (this is shown in figure 4.2).

The development of the CBR system will involve the following main subtasks:

Case Representation Finding a representation of the tempo transformation problem and its solution. In particular:

- Representing the score in terms of musical concepts like 'meter', 'rhythm', 'phrase', and 'motif', or theoretical concepts like 'time span analysis' [79] or 'implication/realization structures' [87] that can be relevant for musical expression.
- Combining the sound file description and the score information into a *performance annotation*, that conveys in which ways the performed melody differs from the notated melody (the score). That is, it should contain a description of the expressivity.

Automatic Case Acquisition Automatizing as far as possible the construction of cases and input problems, based on the information provided (a score, a performance and a desired output tempo). This includes for example the performance annotation, and the construction of the necessary analyses of the score.

Case Retrieval Defining a way to determine the usefulness of cases solved in the past, for solving new problems. This will most likely involve a similarity computation between musical scores, either represented as a sequences of notes, or using alternative representations.

Case Adaptation Defining a way to reuse the solutions of past cases to solve new problems. The goal is to capture the changes between two performances (at different tempos) of a melodic fragment, and apply these changes to the performance of another (similar) melodic fragment, to obtain a new performance for that fragment at a different tempo.

We believe that a rich case representation is essential, due to the complexity of expressive music performance as a problem domain. By describing performances and scores just as vectors of pitch and onset values, for example, one will probably not be able to express patterns and regularities in the expressivity. Ideally, the musical material should be described in terms of concepts that the musician (possibly unconsciously) employs to perform the music. The continuing research concerning the nature and regularities of music performance (some of which was reviewed in chapter 2) may be a guide for making decisions about the representation of music in the system.

Resuming the above, we can state that the practical goal of this research is the development of a CBR system for musical tempo transformations, that maintains the expressive 'integrity' of the musical performance. The motivation for this is not only the practical usefulness of the system as a software application. We hope that our efforts will also shed some light on the usefulness of particular representations musical material for content based audio transformation in general. In particular, we are interested in the representation of expressivity in performances, and high-level representations of nominal score information. Viewed from a different angle, this research is an attempt to apply CBR as a problem solving technique to the problem domain of content based audio transformation. This domain has some interesting challenges from the CBR point of view. Firstly, the solution of a problem is a compound structure that has a sequential nature. In this respect, it is somewhat related to the relatively new idea of using CBR for time series prediction problems. Secondly, although expressive music performance is definitely a knowledge-intensive problem domain, only a very small part of background knowledge is discovered by performance research, and large parts of the background knowledge is of procedural nature rather than declarative nature (this becomes clear when we realize that the only way to become an outstanding performer is by many years of practice). This is unlike typical applications of CBR like medical diagnosis, or legal reasoning, where the background knowledge is largely declarative.

Chapter 5

Proposed Approach and Architecture

In this chapter we will propose a CBR based approach to the problem of content-based audio transformation, geared towards processing of monophonic jazz performances. The proposed architecture in its current form only deals with tempo transformations. But we believe that the architecture can be extended to perform other kinds of transformations, especially generating or changing the expressive mood of a performance, similar to the task performed by SaxEx [8].

In section 5.1, we will give an overview of the system as a whole, which has been implemented as *Tempo-Express*. This chapter is partly based on articles co-authored by the author [52, 51, 4]. Subsequent sections will explain in detail the mechanisms that implement the construction of cases, retrieval, reuse, and the musical material that was used to set up the initial case base.

5.1 General Overview of the *Tempo-Express* System

In this section we briefly present the main *Tempo-Express* modules and the inference flow (see Figure 5.1). The input of *Tempo-Express* is a recording of a jazz performance at a given tempo T_i (a sound file), its corresponding score (a MIDI file) and the desired output tempo. The score contains the melodic and the harmonic information of the musical piece and is analyzed automatically in terms of the Implication/Realization Model. The recording is parsed by the performance analysis module that produces an *XML* file containing the performance melody description (comprising information like onset and offset times dynamics of performed notes). Then, the melody seg-

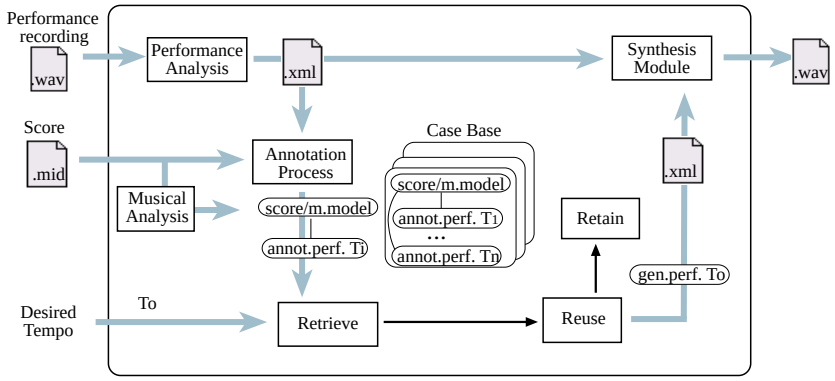


Figure 5.1: General view of *Tempo-Express* modules.

mentation is compared to its corresponding score in the annotation process. The annotated performance and the desired output tempo T_o is the input for the case-based reasoning. The task of the case-based reasoning modules is to determine a sequence of operations that achieves the desired tempo transformation with a musical expressivity that is appropriate for that tempo, while maintaining as much as possible the expressivity that was present in the original performance. These operations are applied to the internal representation of the performance, and the original XML description is modified accordingly, to reflect the changes. Finally, the output of the system is a new version of the original melody, at the desired tempo, generated by the synthesis process using the modified XML description.

5.2 Analysis and Synthesis of Audio

The audio analysis and synthesis are not part of the research reported in this thesis. These processes are being implemented by members of the Music Technology Group (MTG) of the Pompeu Fabra University using signal spectral modeling techniques (see [109, 48] for a detailed description). The output of the analysis process (and input of the synthesis process), is a description of the audio in *XML* format, that adheres to (and extends) the *MPEG7* standard for multimedia description [46].

5.3 Construction of Cases

In this section, we will explain the details of the various aspects of the construction of cases from available information. To construct a case, we need a score, typically in MIDI format. This score is represented internally as a

sequence of note objects, with the basic attributes like pitch, duration and temporal position. This score is analyzed to obtain an I/R representation. This procedure is explained in subsection 5.3.2. Furthermore, an input performance at a particular tempo is needed. The performance not stored literally, but rather a *performance annotation* is constructed to describe how the elements from the performance relate to the elements from the score. This procedure is explained in subsection 5.3.1. The performance annotation is stored as a solution, associated to a particular input description that applies to the performance (in our case, the tempo of the performance). Lastly, the desired output tempo is also included as a problem description specifying what the solution should be like.

5.3.1 Performance Annotation

To analyze a performance of a melody, a crucial problem is to identify which element in the performance corresponds to each note of the score of the melody. Especially in jazz performances, which is the area on which we will focus, this problem is not trivial, since jazz performers often favor a 'liberal' interpretation of the score. This does not only involve changes in expressive features of the score elements as they are performed, but also omitting or adding notes. Thus, one can normally not assume that the performance contains a corresponding element for every note of the score, neither that every element in the performance corresponds to a note of the score. Taking these performance liberties into account, a description of a musical performance could take the form of a sequence of operations that are applied to the score elements in order to obtain the performance.

From this perspective the edit distance, as described in the previous section, is very useful. The edit distance has been used before in the performance-to-score mapping problem by Dannenberg [30] and Large [76], among others. The application area has been score-tracking for automatic accompaniment as well as performance timing research. Other approaches to performance-to-score matching have also been proposed (e.g. [94], [57]). See [39] and [58] for an overview and comparison of several approaches.

The application of the edit distance in the context of comparing performances to scores is somewhat different from the case where scores are being compared to other scores. In the first case, we deal with sequences of different nature. The performance itself is not necessarily a discrete sequence but could be for example a continuous (audio) stream. Although matching score elements to fragments of audio data is not inconceivable, it is probably more convenient to make a transcription of the performance into a sequence of note elements before matching. The resulting sequence of note elements

is more appropriate for comparing with scores, but it must be kept in mind that transcription of audio to note sequences is a reductive procedure. For example, pitch and dynamics envelopes are usually reduced to single values.

Another difference between score-performance matching and score-score matching is more conceptual. In score-performance matching, the performance is thought to be *derived* from the score, that is, the elements of the score sequence are *transformed* into performance elements, rather than *replaced* by them. For this reason, in the context of score-performance matching it is more appropriate to talk of *transformation* operations instead of *replace* operations.

The Edit Operations

An important aspect of score-performance matching is the decision which edit operations to provide for matching. This is important since the sequence of optimal sequence of operations that constitutes the edit distance between score and performance, will be used as a representation of the performance. As such, the edit operations define the concepts that we use to describe performances. Here we propose a set of edit-operations that correspond to the variety of phenomena that we have actually encountered in a set of real jazz performances (the data referred to in section 5.4). These edit operations can be classified (as in figure 5.2) to make explicit the characteristics of their behavior. Note that all the operations refer to one or more elements in one (or both) of the sequences that are aligned. We can distinguish, within this general class of *Reference* operations, those that refer to notes in the score sequence and those that refer to elements in the performance sequence. Deletion operations refer to notes of the score sequence that are not present in the performance sequence (i.e. the notes that are not played), therefore they can be classified as *Score-Reference* operations. Conversely, insertion operations refer only to elements in the performance sequence (i.e. the notes that were added), so they form a subclass of *Performance-Reference* operations. Transformation, consolidation and fragmentation operations refer to elements from both the score and the performance and thus form a shared subclass of Score-Reference and Performance-Reference operations. We call this class *Correspondence* operations.

The Reference, Score-Reference, Performance-Reference and Correspondence classes are abstract classes that are just conceived to express the relationships between concrete classes of operations, and are not actually identified in the matching process. The concrete classes, that are defined as real edit-operations in the matching process are depicted in figure 5.2 with light gray boxes. They are:

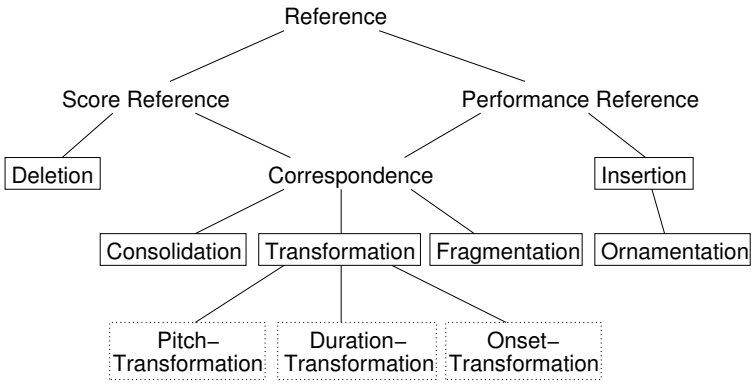


Figure 5.2: A hierarchical representation of edit operations for performance annotation. The unboxed names denote abstract classes; the light gray boxes denote concrete classes, and the dark gray boxes denote 'hyper concrete' classes.

Insertion Represents the occurrence of a performed note that is not in the score

Deletion Represents the non-occurrence of a score note in the performance

Consolidation Represents the agglomeration of multiple score notes into a single performed note.

Fragmentation Represents the performance of a single score note as multiple notes.

Transformation Represents the change of nominal note features.

Ornamentation Represents the insertion of one or several short notes to anticipate another performed note.

In the case of Transformation, we are not only interested in the one-to-one correspondence of performance elements to score elements itself, but rather in the changes that are made to attribute values of score notes when they are transformed into performance elements. Therefore, we view transformation operations as compositions of several transformations, e.g. *pitch transformations*, *duration transformations* and *onset transformations*. Since these transformations can occur simultaneously, they are not suitable for using them in an edit-distance, where each sequence element is covered by exactly one operation. Our solution is to have a single Transformation operation for the computation of the alignment, as a rough identification. In a second stage, after the alignment has been computed, the score and performance

events corresponding to Transformation operations can be compared in more detail to establish which of the pitch, duration, and onset transformation really occurred. For want of a better name, we call such operations 'hyper concrete' operations.

Following the same idea, fragmentation and consolidation operations (as described in section 3.5.2), can be elaborated by such transformation operations. For example, a consolidation operation could be composed of a duration transformation that specifies how the total duration of the consolidated score notes deviates from the duration of the corresponding performance element.

Based on the fact that the phrases in our data set were played by a professional musician and they were performed quite closely to the score, it may be thought that the performances could be described by only correspondence operations, that map a score element to a performance element, perhaps with minor adjustments of duration and onset. However, as mentioned before, most actual performances, and indeed the performances in our data set, are not such literal reproductions of the score; they contain extra notes, or lack some notes from the score. Listening to the performances revealed that these were not unintentional performance errors, since they were often found on the same places in various performances of the same phrase and the effect sounded natural. This implies that in addition to correspondence operations, insertion and deletion operations are also required. Apart from real insertions of notes, that gave the impression of an elaboration of the melody (such insertions occurred, but were rare), another type of insertion was found to occur rather often: ornamentation. By ornamentation we refer to one or more very short notes (typically about 100 or 200 ms.) that are usually a chromatic approach from below to the next score note. We have found such ornamentations to consist of one, two or three notes. Furthermore, we observed that *consolidation* (as described in the previous section) occurred in some performances. Occasionally, we found cases of *fragmentation*. Other transformations, such as sequential transposition (reversal of the temporal order of notes) were not encountered. Thus, the set of operations shown in figure 5.2 initially seem to be adequate for representing performances with respect to the scores.

The Cost Values

Once the set of edit-operations is determined, we need to decide good cost values for each of them. Ideally, the cost values will be such that the resulting optimal alignment corresponds to an intuitive judgment of how the performance aligns to the score (in practice, the subjectivity and ambiguity

that is involved in establishing this mapping by ear, turns out to be largely unproblematic). The main factors that determine which of all the possible alignments between score and performance is optimal, will be on the one hand the features of the note elements that are involved in calculating the cost of applying an operation, and on the other hand the relative costs of the operations with respect to each other.

In establishing which features of the compared note elements are considered in the comparison, we have taken the choices made by Mongeau and Sankoff [85] as a starting point. In addition to pitch and duration information (proposed by Mongeau and Sankoff), we have decided to incorporate the difference in position in the costs of the correspondence operations (transformation, consolidation and fragmentation), because this turned out to improve the alignment in some cases. One such case occurs when one note in a row of notes with the same pitch and duration is omitted in the performance. Without taking into account positions, the optimal alignment will delete an arbitrary note of the sequence, since the deletions of each of these notes are equivalent based on pitch and duration information only. When position *is* taken into account, the remaining notes of the performance will all be mapped to the closest notes in the score, so the deletion operation will be performed on the score note that remains unmapped, which is often the desired result.

It is important to note that when combining different features, like pitch, duration and onset into a cost-value for an operation, the relative contribution of each term is rather arbitrary. For example when the cost of transforming one note into another would be defined as the difference in pitch plus the difference in duration, the outcome depends on the units of measure for each feature. The relative weight of duration and pitch is different if duration is measured in seconds, than if it is measured in beats. Similarly, pitch could be measured in frequency, semitones, scale steps, etcetera. Therefore, we have chosen a radically parametrized approach, in which the relative contribution of each term in the weight function is weighted by a constant parameter value.

The other aspect of designing cost-functions is the relative cost of each operation. After establishing the formula for calculating the weights of each operation, it may be that some operations should be systematically preferred to others. This independence of costs can be achieved by multiplying the cost of each operation by a factor and adding a constant.

The cost functions w for the edit operations given below. $\mathcal{P}$, $\mathcal{D}$, and $\mathcal{O}$ are functions such that $\mathcal{P}(x)$ returns the pitch (as a MIDI number) of a score or performance element x, $\mathcal{D}(x)$ returns its duration, and $\mathcal{O}(x)$ returns its onset time. Equations 5.1, 5.2, 5.3, 5.4, 5.5, 5.6 define the costs of deletion,

insertion, ornamentation, transformation, consolidation and fragmentation, respectively.

$$w(s_i, \emptyset) = \alpha_d \cdot \mathcal{D}(s_i) + \beta_d \tag{5.1}$$

$$w(\emptyset, p_j) = \alpha_i \cdot \mathcal{D}(p_j) + \beta_i \tag{5.2}$$

$$w(\emptyset, p_j, \cdots, p_{j+L+1}) = \alpha_o \cdot \begin{pmatrix} \gamma_o \cdot \sum_{l=1}^{L} |1 + \mathcal{P}(p_{j+l}) - \mathcal{P}(p_{j+l-1})| + \\ \delta_o \cdot \sum_{l=0}^{L} \mathcal{D}(p_{j+l}) | \end{pmatrix} + \beta_o \tag{5.3}$$

$$w(s_i, p_j) = \alpha_t \cdot \begin{pmatrix} \gamma_t \cdot |\mathcal{P}(s_i) - \mathcal{P}(p_j)| + \\ \delta_t \cdot |\mathcal{D}(s_i) - \mathcal{D}(p_j)| + \\ \epsilon_t \cdot |\mathcal{O}(s_i) - \mathcal{O}(p_j)| \end{pmatrix} + \beta_t \tag{5.4}$$

$$w(s_i, ..., s_{i+K}, p_j) = \alpha_c \cdot \begin{pmatrix} \gamma_c \cdot \sum_{k=0}^{K} |\mathcal{P}(s_{i+k}) - \mathcal{P}(p_j)| + \\ \delta_c \cdot |\mathcal{D}(p_j) - \sum_{k=0}^{K} \mathcal{D}(s_{i+k})| + \\ \epsilon_c \cdot |\mathcal{O}(s_i) - \mathcal{O}(p_j)| \end{pmatrix} + \beta_c \tag{5.5}$$

$$w(s_i, p_j, ..., p_{j+L}) = \alpha_f \cdot \begin{pmatrix} \gamma_f \cdot \sum_{l=0}^{L} |\mathcal{P}(s_i) - \mathcal{P}(p_{j+l})| + \\ \delta_f \cdot |\mathcal{D}(s_i) - \sum_{l=0}^{L} \mathcal{D}(p_{j+l})| + \\ \epsilon_f \cdot |\mathcal{O}(s_i) - \mathcal{O}(p_j)| \end{pmatrix} + \beta_f \tag{5.6}$$

The costs of transformation (5.4), consolidation (5.5), and fragmentation (5.6), are principally constituted of the differences in pitch, duration and onset times between the compared elements. In the case of one-to-many matching (fragmentation) or many-to-one (consolidation), the difference in pitch is calculated as the sum of the differences between the pitch of the single element and the pitches of the multiple elements. The difference in duration is computed between the duration of the single element and the sum of the durations of the multiple elements. The difference in onset is computed between the onset of the single element and the onset of the onset of the first of the multiple elements. The cost of deletion (5.1) and insertion (5.2)is determined by the duration of the deleted element. The cost of ornamentation (5.3) is determined by the pitch relation of the ornamentation elements and the ornamented element (chromatically ascending sequences are preferred), and the total duration of the ornamentation elements. Finally, note the competitive relationship of the insertion and ornamentation operations, since they both account for performance elements that have no corresponding score element. We want very short notes with a certain pitch relationship to subsequent notes to be matched as ornamentations. If the

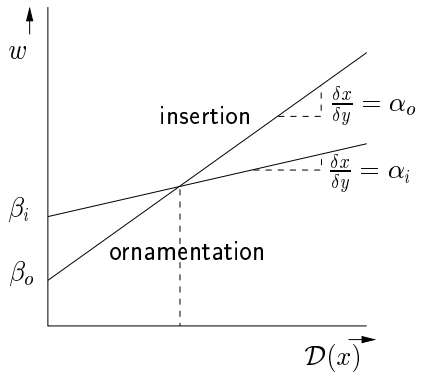


Figure 5.3: Cost values for insertion and ornamentation operations as a function of note duration.

duration of the notes exceeds a certain limit, or the pitch relation is not satisfied, the notes should be matched as insertions instead. This can be achieved by setting β_i higher than β_o and α_i lower than α_o. Figure 5.3 shows how the cost values of insertion and ornamentation vary with note duration. Before the point where the lines intersect (shorter durations), the notes will be accounted for as ornamentation; after this point (longer durations) the notes will be accounted for as insertion.

From a musical perspective, it can be argued that a the cost matching two elements with different pitches should not depend on the difference of the absolute pitches, but rather on the different roles the pitches play with respect to the underlying harmonies, or their scale degree, as these features have more perceptual relevance than absolute pitch difference. This would certainly be essential in order to make good alignments between scores and performances that very liberally paraphrase the score (e.g. improvisations on a melody) and also in the case where alignment is constructed for assessing the similarity between different scores. In our case however, we currently deal with performances that are relatively 'clean' interpretations of the score (rather than improvisations). As such, changes of pitch are very uncommon in our data. Still, it is desirable to have a more sophisticated pitch comparison approach, to accommodate more liberal performances in the future.

Evolutionary Optimization of the Annotation Process

The introduction of parameters inherently comes with the problem of finding appropriate values for those parameters. Although manual tuning would yield good results for small sets of performances, the accuracy of annotation with manual tuning did hardly rise above the accuracy of random parameter

settings for larger sets of performances. Therefore, we have employed a *genetic algorithm* to try to obtain parameter settings that would have validity for larger sets of performances.

The idea of evolutionary optimization of the parameter values is rather simple: an array of parameter values (one value for each parameter) can be treated as a chromosome. The number of errors produced in the annotation of a set of performances using that set of parameter values, is inversely related to the fitness of the chromosome. By evolving an initial population of (random) chromosomes through crossover, mutation and selection, we expect to find a set of parameter values that minimizes the number of annotation errors, and thus improves automatic performance annotation.

We are interested in two main questions. The first is whether it is possible to find a parameter setting that works well in general. That is, can we expect a parameter setting that worked well for a training set to perform well on unseen performances? The second question is whether there is a single setting of parameter values that optimizes the annotations. It is also conceivable that good annotations can be achieved by several different parameter settings.

We have run the genetic algorithm with two different (non-overlapping) training sets, both containing twenty performances. These were (monophonic) saxophone performances of eight different phrases from two jazz songs (*Body and Soul*, and *Once I Loved*), performed at different tempos. For each of the performances, the correct annotation was available. The fitness of the populations was assessed using these annotations.

The fitness of the chromosomes is calculated by counting the number of *annotation errors* using the parameter values in the chromosome. For example, assume that the correct annotation of a melodic fragment is 'T T C T', and the annotation of that fragment obtained by using the parameter values of the chromosome is 'T T T D T' (that is, a consolidation operation is confused with an transformation and a deletion operation). The 'C' doesn't match to an element in the second sequence, and the 'T' and 'D' don't match to elements in the first sequence and thus three errors occur. To count the errors between the correct and the predicted annotations (which are represented as sequences of symbols), we use the edit-distance (don't confuse this use of the edit-distance to *compare* annotations with the use of the edit-distance to *generate* annotations).

For a given set S of performances (for which the correct annotations are known), we define the fitness of a chromosome c as:

$$\mathit{fit}(c) = \frac{1}{E(c,S)+1}$$

where $E(c,S)$ is the total number of errors in the predicted annotations

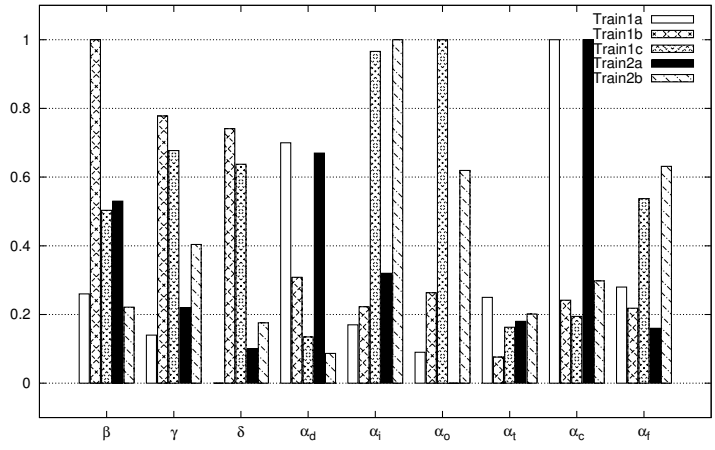


Figure 5.4: Estimated parameter values for two different training sets (Tr1 and Tr2). Three runs were done for each set (a, b, and c). The x-axis shows the nine different parameters of the cost functions (see section 5.3.1). For each parameter the values are shown for each run on both training sets

for S using the parameter values in c. The fitness function *fit* ranges from zero to one. Obviously, a fitness value of one is the most desirable, since it corresponds to zero annotation errors.

For each of the two training sets, Tr1 and Tr2, the evolution algorithm was run three times. The resulting parameter settings are shown in figure 5.4. Table 5.1 shows the number of annotation errors each of the parameter settings produced on the training sets, and on a test set (a set of 35 performances, none of which occurred in Tr1 or Tr2). The average number of annotation errors on the test set is about 32 on a total of 875 annotation elements in the test set, an error-percentage of 3.66%. This is only slightly higher than the error-percentages on the training sets: 2,60% for Tr1, and 2,37% for Tr2 (averaged over three runs), and substantially lower than the average error-percentage of random parameter settings on the test set, which is about 13,70%.

Table 5.2 shows the pair-wise correlations of the values. As can be seen from the cross-correlations in the table, the parameter settings did not all converge to the same values. Nevertheless, there were some cases in which the parameters were highly correlated. In particular the solutions found in runs Tr1a, and Tr2a are highly similar (this can be easily verified by eye in figure 5.4. A rather strong correlation is also observed between the solutions found in Tr1c and Tr2b, and those in Tr1b, and Tr2c. It is interesting that the correlated solutions were obtained using non-overlapping sets of performances. This is evidence that the solutions found are approximations of a

	Tr1a	Tr1b	Tr1c	Tr2a	Tr2b	Tr2c
Errors on Train	19 (3.89)	9 (1.84)	10 (2.05)	11 (2.30)	12 (2.51)	11 (2.30)
Errors on Test	19 (2.17)	26 (2.97)	30 (3.43)	19 (2.17)	32 (3.66)	65 (7.43)

Table 5.1: Annotation errors produced by the obtained solutions for three different runs (denoted by the letters a, b, and c) on two different training sets (Tr1 and Tr2) and a test set. The first row shows the number of errors on the set that the solutions were trained on, and the corresponding percentages in parentheses (Tr1 contained 488 annotation elements in total, and Tr2 contained 479). The second row shows the number of errors on the test set (875 elements), with percentages in parentheses

single parameter setting that is valid for the performances in both training sets. In the case of the solutions of Tr1a and Tr2a, the approximated parameter setting may also have a more general validity, since both solutions have a low error number of annotations on the test set as well (see table 5.1).

	Tr1a	Tr1b	Tr1c	Tr2a	Tr2b	Tr2c
Tr1a	1.00	-0.32	-0.70	0.92	-0.32	-0.28
Tr1b	-0.32	1.00	0.17	-0.02	-0.33	0.68
Tr1c	-0.70	0.17	1.00	-0.61	0.76	0.07
Tr2a	0.92	-0.02	-0.61	1.00	-0.33	-0.12
Tr2b	-0.32	-0.33	0.76	-0.33	1.00	-0.47
Tr2c	-0.28	0.68	0.07	-0.12	-0.47	1.00

Table 5.2: Cross-correlations of the parameter values that were optimized using two different training sets (Tr1 and Tr2), and three runs for each set (a, b, and c)

Concludingly, we can state that all solutions from different trials on two non-overlapping sets of performances substantially improved the quality of annotation of a test set over untuned parameter settings. Moreover, cross-correlations were found between some parameter settings that were optimized for different training sets. This suggests that they are approximations of a parameter setting that works well for a larger group of performances. In general however, the solutions did not all converge to a single set of parameter values.

5.3.2 Musical Analysis: An I/R Parser for Monophonic Melodies

The score input to the system is analyzed to obtain an abstract description of the music (which will be used in later problem solving steps). We use Narmour's Implication/Realization Model (see section 3.5.1) to analyze the

score, since it provides a categorization and segmentation of the musical surface that corresponds to the perception of the melody by human listeners (this is of course a bold claim, see section 3.5.1 for a discussion).

The I/R model gives rise to the analysis of melodies into a sequence of (either basic or derived) structures, the theoretical constructs of the model. Narmour supposes the simultaneous activity of top-down and bottom-up processes, both guiding the listener's expectations in the perception of music. The bottom-up process, Narmour argues, is innate, subconscious and reflexive. It leads from musical/auditive input to expectations about subsequent input in a mechanistic way, following if-then rules. In addition, the top-down process at the same time actuates learned musical schemata, and interferes with the expectations generated by the bottom-up process. This behavior can appropriately be conceived of as the imposition of except-clauses to the if-then rules that the bottom-up process follows. The top-down contribution to expectations of the listener represents her knowledge of musical style, the idiosyncratic features that characterize a musical style (called *extra opus style* in the I/R model), or more specifically, a particular piece of music (*intra opus style* in the I/R model).

Based on the I/R model, we have built a parser for monophonic melodies, that automatically generates the corresponding sequences of I/R structures. The algorithm implements most of the *bottom-up* part of the I/R model. The *top-down* part is not implemented at this moment. Nevertheless, we believe that the I/R analyses generated by the parser have a reasonable degree of validity (the bottom-up part being regarded as the main contribution of the model [106]) and are useful for our purposes.

The parser takes as input a melody, represented as a sequence of notes, having pitch, duration and position attributes. Additionally, meter is known. The process of parsing this melody to obtain a sequence of I/R structures, can be divided into a small number of straight-forward steps:

1. Tag pairs of intervals with labels to categorize them as I/R structures
2. Detect positions where closure occurs
3. Detect positions where inhibition of closure (non-closure) occurs
4. Categorize the interval pairs within the sequence segments in between closures
5. Assimilate chained P and D structures

In the following subsections, we will explain these steps in more detail.

Tagging Interval Pairs

This step consists of checking every subsequent pair of intervals of the melody and tagging them with two boolean labels: *intervallic similarity*, and *registral sameness*. These two features are of primary importance in the classification of a sequence of notes as a particular I/R structure.

In the I/R model, two subsequent intervals are said to have intervallic similarity if their *intervallic differentiation* (i.e. the difference between these intervals) is less or equal to a minor third. The registral sameness feature is present if subsequent intervals have the same registral direction, which can be either upward, downward, or lateral (in the case of a prime interval).

Detecting Closure

In the I/R model, *closure* is the term for the inhibition of implication. This inhibition can be established by several conditions. The most prominent are rests, *durational cumulation* (where a note has significantly longer duration than its predecessor), metrical accents, and resolution of dissonance into consonance. These conditions can occur in any combination. Depending on the number of dimensions in which closure occurs and the degree of closure in each dimension, the overall degree of closure will differ.

The closure detection step in the parser currently detects three kinds of conditions for closure: rests, durational cumulation, and metrical accents. On each point in the sequence, the level of closure is computed for each of the dimensions.

Detecting Non-Closure

Although the conditions mentioned in the previous subsection principally imply closure, there are circumstances where closure is inhibited. Examples of these circumstances are:

- The absence of metrical emphasis
- The envelopment of a metric accent by a P process in the context of additive (equally long) or counter-cumulative (long to short) durations
- The envelopment of a metric accent by a D process in the context of additive (equally long) durations
- The occurrence of dissonance on a metric accent

Currently, the first three of these circumstances are recognized during the step of detecting conditions for non-closure. In these cases, closure due to metrical emphasis that was detected in the previous step, is canceled.

Categorizing the Interval Pairs

When the final degree of closure is established for each point in the sequence, the next step is the categorization of the interval pairs, based on the information gathered during the first step. The information about the degree of closure is also used in this step. In between two parts of closure, the structures are overlapping, sharing two notes (i.e. one interval), due to the lack of closure. Where closure occurs, two contiguous structures either share one note, or none, depending on the degree of closure.

This procedure generally leads to structures consisting of three notes, except in cases where closure occurs on two subsequent notes, or when a note after a closural note is followed by a rest. In the first case a two note structure, or *dyad* results; in the second case a single note structure, or *monad* results.

Assimilating Chained P and D Structures

In the I/R model, there are two of the basic structures that can span more than three notes, viz. P and D structures. P Structures occur when three or more notes form a ascending or descending sequence of similar intervals; D structures occur when three or more notes continue in lateral direction, i.e. a note is repeated. A further condition is the absence of closure, for example by durational cumulation.

In the sequence of structures, only structures of three notes have been identified. Yet this is not problematic, since possible instants of longer P and D structures necessarily manifest themselves as a chain of P or a chain of D structures, overlapping each other by two notes. Subsequent P or D structures that share only one note, cannot be regarded as a single P or D structure, since the fact that the overlap is less than two notes, indicates that there is some (weak) degree of closure.

So the last step of the construction of an I/R analysis is simply to 'reduce' any chain of overlapping D or P structures into single structures.

The final I/R analysis consists of a sequence of structures. The structures of this sequence is represented by the parser as symbols denoting structural type, with references to the notes of the melody that belong to that structure.

5.4 The Case Base

As explained in the previous section, a case is represented as a complex structure containing three different kinds of information: (1) the representation

of the musical score (notes and chords), (2) a musical model of the score (automatically inferred from the score using Narmour's Implication/Realization model as background musical knowledge as described in section 5.3.2), and (3) a collection of annotated performances. An example of how the different kinds of information are structured is shown in figure 5.5. The relations (like 'belongs-to', 'underlying-harmony', 'score-reference', and 'performance-reference') allow us to 'navigate' through the information of the case. For example, given a particular note of the melody, it is possible to retrieve the other notes that belong to the same I/R structure, using the 'belongs-to', 'first-note', 'middle-note' and 'last-note' relations.

For the case acquisition, saxophone performances were recorded of 5 jazz standards, each consisting of about 4 distinct phrases. The performances were played by a professional performer, at about 11 different tempos per phrase. From this, the initial case base was constructed, containing 20 scores of musical phrases, each with about 11 annotated performances.

5.5 Problem Solving

In this section we will explain the process of problem solving in TempoExpress. We have adopted Constructive Adaptation (CA, see subsection 3.3.3, page 40) as a specific approach to reuse cases. We believe this method offers good opportunities to generate performances that satisfy predefined constraints regarding their form and characteristics. Although CA has been presented as a reuse technique, in fact CA invites to interleaving retrieval and reuse steps. This is because the solution is gradually constructed by elaborating or composing partial solutions. Therefore, it seems natural that the retrieval of cases (and hence the decision of what cases are relevant) is not irrevocably done before the construction of a solution. Instead, it seems more appropriate to use the retrieval step to filter out clearly unusable/irrelevant cases and prepare the remaining data, thus generating a smaller data set to work with in the reuse step.

5.5.1 Retrieval

The goal of the retrieval step is to form a pool of relevant cases, that can possibly be used in the reuse step. This done in the following three steps: firstly, cases that don't have performances at both the input tempo and output tempo are filtered out; secondly, those cases are retrieved from the case base that have phrases that are I/R-similar to the input phrase; lastly, the retrieved phrases are segmented. The three steps are described below.

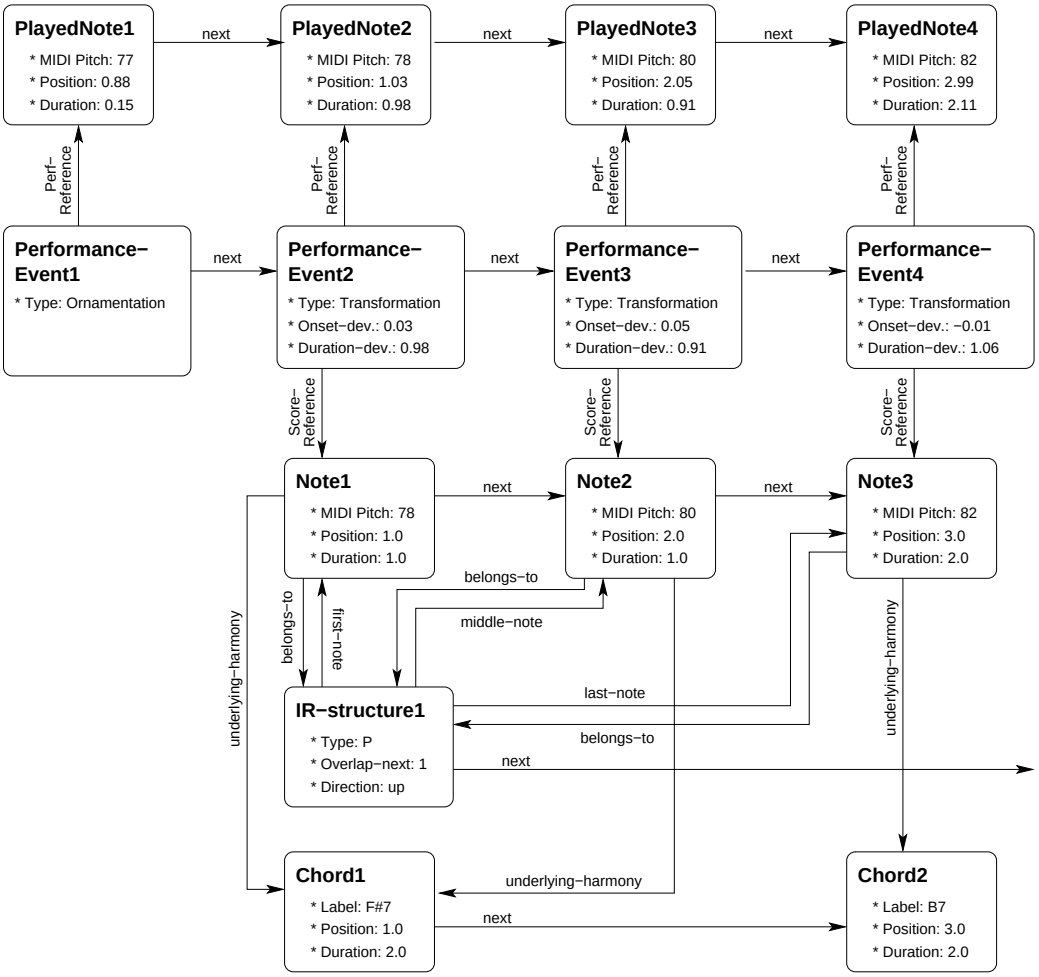


Figure 5.5: The internal structure of a case. Different kinds of information, like score-melody and harmony, I/R analysis, performance and performance annotation are structured through different kinds of relations.

Case filtering by tempo

In the first step, the case base is searched for cases that have performances both at the tempo the input performance was played, and the tempo that was specified in the problem description as the desired output tempo. The matching of tempos need not be exact, since we assume that there are no drastic changes in performance due to tempo within small tempo ranges. For example, a performance played at 127 beats per minute (bpm) may serve as an example case if we want to construct a performance at 125 bpm.

I/R based retrieval

In the second step, the cases selected in step 1 are assessed for melodic similarity to the score specified in the problem description. In this step, the primary goal is to rule out the cases that belong to different styles of music. For example, if the score in the problem description is a ballad, we want to avoid using a bebop theme as an example case. Note that the classification of musical style based on just melodic information (or derived representations) is far from being an established issue. Nevertheless, there is some evidence [51] that the comparison of melodic material at different levels of abstraction yields different degrees of discriminatory power. For example comparing on the most concrete level (comparing individual notes) is a good way to find out which melodies in a set are nearly identical to a particular target melody. But if the set of melodies does not contain a melody nearly identical to the target, the similarity values using this measure are not very informative, since they are highly concentrated in a single value. On the other hand, comparisons based on more abstract descriptions of the melody (e.g. melodic contour, or I/R analyses), tend to produce a distribution of similarity values that is spread out through the spectrum more equally. Thus, these measures tell us in a more informative way *how* similar two melodies are (with respect to the other melodies in the set), even if they are considerably different. As a consequence, a melodic similarity measure based on an abstract representation of the melody seems a more promising approach to separate different musical styles.

We use the I/R analysis of the melodies to assess similarities. The measure used is an edit distance. The edit distance measures the minimal cost of transforming one sequence of objects into another, given a set of edit operations (like insertion, deletion, and replacement), and associated costs. We have defined edit operations and their corresponding costs for sequences of I/R structures (see [51] for more details). The case base is ranked according to similarity with the target melody, and the subset of cases with similarity values above a certain threshold are selected. The resulting set of cases will contain phrases that are roughly similar to the input score.

Segmentation

In this step, the melodies that were retrieved in the second step are segmented. The motivation for this twofold. Firstly, using complete melodic phrases as the working unit for adaptation is inconvenient, since a successful adaptation will then require that the case base contains phrases that are nearly identical as a whole to the input phrase. Searching for similar phrase

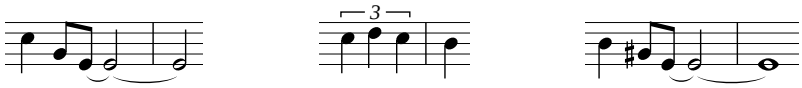


Figure 5.6: Segmentation of the first phrase of 'All of Me', according to I/R structures. The segments correspond to single I/R structures, or sequences of structures if they are strongly chained (see subsection 3.5.1)

segments will increase the probability of finding a good match. Secondly, the segmentation is motivated by the intuition that the way a particular note is performed does not only depend of the attributes of the note in isolation, but also on the musical context of the note. Therefore, rather than trying to reuse solutions in a note-by-note fashion, it seems more reasonable to perform the reuse segment by segment. This implies that the performance of a retrieved note is only reused for a note of the input phrase if their musical contexts are similar.

Melodic segmentation has been addressed in a number of studies (e.g. [113][18]), with the aim of detecting smaller musical structures (like motifs) within a phrase. Many of them take a data driven approach, using information like note inter onset intervals (IOI) and metrical positions to determine the segment boundaries. Our method of segmentation is based on the I/R representation of the melodies. This may seem quite different from the approach mentioned above, but in essence it is similar. The melodies are split at every point where the overlap of two I/R structures is less than two notes (see subsection 3.5.1). This overlap is determined by the level of closure, which is on its turn determined by factors like metrical position and IOI. The resulting segments usually correspond to the musical motifs that constitute the musical phrase, and are used as the units for the stepwise construction of the output performance. As an example, figure 5.6 displays the segmentation of the first phrase of 'All of Me' (the complete phrase is shown in figure 3.6).

5.5.2 Reuse

In the reuse step a performance of the input score is constructed at the desired tempo, based on the input performance and the set of retrieved phrase segments. As mentioned above, this step is realized using constructive adaptation. In this subsection, we will first explain briefly how the reuse step can in general be realized as best-first search, and then we will explain how we implemented the functions necessary to make the search-algorithm operational in the context of performance transformation.

In constructive adaptation, partial solutions of the problem are represented as states. Furthermore, a function HG must be defined for generating a set of successor states for a given state. The state space that emerges from this function and the state that represents the empty solution (generated by a function Initial-State), is then searched for a complete solution that satisfies certain constraints (through a function Goal-Test). The resulting state is transformed to a real solution by a function SAC. The order of expansion of states is controlled by a function HO that orders the states in a best-first manner. The search process is expressed in pseudo code below.

```
Initialize OS = (list (Initial-State Pi))
Function CA(OS)
        Case (null OS) then No-Solution
        Case (Goal-Test (first OS)) then (SAC (first OS))
        Case else
                Let SS = (HG (first OS))
                Let OS = (HO (append SS (rest OS)))
                     (CA OS)
```

Figure 5.7: The search process of constructive adaptation expressed in pseudo code. Functions HG and HO are Hypotheses Generation and Hypotheses Ordering. Variables *OS* and *SS* are the lists of Open States and Successor States. The function SAC maps the solution state into the configuration of the solution. The function Initial-State maps the input problem description *Pi* into a state. From Plaza and Arcos [93]

We explain our implementations of the functions Initial-State, HG, HO, Goal-Test, and SAC below.

Initial-State

The function Initial-State returns a state that is used as the starting point for the search. It takes the input problem description (the score, analysis, input-performance, and desired output tempo) as an argument. In our case, the state contains a sequence of score segments, and a slot for storing the corresponding performance segments (none of which is filled in the initial state, obviously). Furthermore, there is a slot that stores the quality of the partially constructed performance, as a number. We will explain the derivation of this number in the next subsection. Figure 5.8 shows the initial state for a short musical fragment (containing two segments).

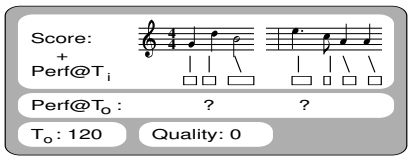


Figure 5.8: Example of an initial state in Constructive Adaptation. T_i is the tempo of the input performance; T_o is the desired output tempo

Hypothesis-Generation (HG)

The Hypothesis-Generation function takes a state as an argument and tries to find a sequence performance events for one of the unprocessed score segments in the state. We will illustrate this procedure step by step, using the first segment of the initial state in figure 5.8 as an example. The steps are presented graphically in figure 5.9.

The *first step* is to find the segment in the pool of retrieved melodic segments that is most similar to the input score segment. The similarity is assessed by calculating the edit distance between the segments (the edit distance now operates on notes rather than on I/R structures, to have a finer grained similarity assessment). A mapping between the input score segment and the best matching retrieved segment is made.

In the *second step*, the performance annotation events (see subsection 5.3.1 and [4]) corresponding to the relevant tempos are extracted from the retrieved segment case and the input problem specification (both the input tempo T_i and the output tempo T_o for the retrieved segment case, and just T_i from the input problem specification).

The *third step* consists in relating the annotation events of the retrieved segment to the notes of the input segment, according to the mapping between the input segment and the retrieved segment, that was constructed in the first step. For the notes in the input segment that were mapped to one or more notes in the retrieved segment, we now obtain the tempo transformation from T_i to T_o that was realized for the corresponding notes in the retrieved segment. It is also possible that some notes of the input segment could not be matched to any notes of the retrieved segment. For such notes, the retrieved segment can not be used to obtain annotation events for the output performance. Currently, these gaps are filled up by directly transforming the annotation events of the input performance (at tempo T_i) to fit the output tempo T_o (by scaling the duration of the events to fit the tempo). In the future, more sophisticated heuristics may be used.

In the *fourth step*, the annotation events for the performance of the input score at tempo T_o are generated. This is done in a note by note fashion, us-

ing rules that specify which annotation events can be inferred for the output performance of the input score at T_o, based on annotation events of the input performance, and the annotation events of the retrieved performances (at T_i and T_o). To illustrate this, let us explain the inference of the Fragmentation event for the last note of the input score segment (B) in figure 5.9. This note was matched to the last two notes (A, A) of the retrieved segment. These two notes were played at tempo T_i as a single long note (denoted by the Consolidation event), and played separately at tempo T_o. The note of the input segment was also played as a single note at T_i (denoted by a Transformation event rather than a Consolidation event, since it corresponds to only one note in the score). To imitate the effect of the tempo transformation of the retrieved segment (one note at tempo T_i and two notes at tempo T_o), the note in the input segment is played as two shorter notes at tempo T_o, which is denoted by a Fragmentation event (F).

In this way, adaptation rules were defined, that describe how the tempo transformation of retrieved elements can be translated to the current case. In figure 5.9, two such rules are shown. If the antecedent part matches the constellation of annotation events, the tempo transformation in the consequent part can be applied. It can occur that the set of rules contains no applicable rule for a particular constellation, in particular when the performances at T_i of the retrieved note and the input note are too different. For example, if the score note is played as a Transformation event, but the retrieved note is deleted in the performance at T_i, then the performances are too different to make an obvious translation. In this case, the annotation events from the input performance are transformed in the same way as in the case where no corresponding note from the retrieved segment could be found (see the third step of this subsection).

The mismatch between the input segment and the retrieved segment and the inability to find a matching adaptation rule obstructs the use of case knowledge to solve the problem and forces *TempoExpress* to resort to default mechanisms. This will affect the quality of the solution. To reflect this, the value of the quality slot of the state (see figure 5.8) is calculated as the number of input score notes for which annotation events could be inferred from retrieved cases, divided by the total number of notes processed so far (that is, the sum of all notes in the processed input segments, including the current input segment).

Hypothesis-Ordering (HO)

The Hypothesis-Ordering function takes a list of states (each one with its partial solution) and orders them so that the states with the most promising

partial solutions come first. For this ordering, the quality value of the states is used. In our current implementation, the quality value is only determined by one factor, roughly the availability of appropriate cases. Another factor that should ideally influence the quality of the states is the 'coherence' of the solution. For example, if the notes at the end of one segment were anticipated in time (as a possible effect of a Transformation event), then anticipation of the first notes of the next segment will not have the typical effect of surprise, since the listener will experience the performance as being shifted forward in time, instead of hearing a note earlier than expected. We are currently incorporating the detection and evaluation of such phenomena into the Hypothesis-Ordering function, so that this functionality will soon be available.

Goal-Test

The Goal-Test function is called on the best state of an ordered list of states to test if the solution of that state is complete and satisfies the constraints imposed upon the desired solution. The completeness of the solution is tested by checking if all segments of the input score have a corresponding segment in the performance annotation for the output tempo. The constraints on the solution are imposed by requiring a minimal quality value of the state. In our case, where the quality value represents the ratio of notes for which annotation events were obtained using retrieved cases (a value between 0 and 1), the quality value is required to be superior or equal to 0.8.

State-to-Solution (SAC)

The State-to-Solution function takes the state that passed the goal-test and returns a solution to the input problem. This step consists in building a complete performance annotation from the annotation events for the score segments (basically concatenation of the events). The new performance annotation is used to adapt the XML description of the original audio file, by changing attribute values, and possibly deleting and inserting new note descriptors. Finally, the audio transformation module (which is under development) generates a new audio file, based on the new XML description.

5.5.3 Retain

When the solution that was generated is satisfying to the listener, and when the quality of the solution is high (that is, default adaptation operations have

been scarcely used, or not at all), it is retained as a case that includes the input score, the input performance, and the newly generated performance.

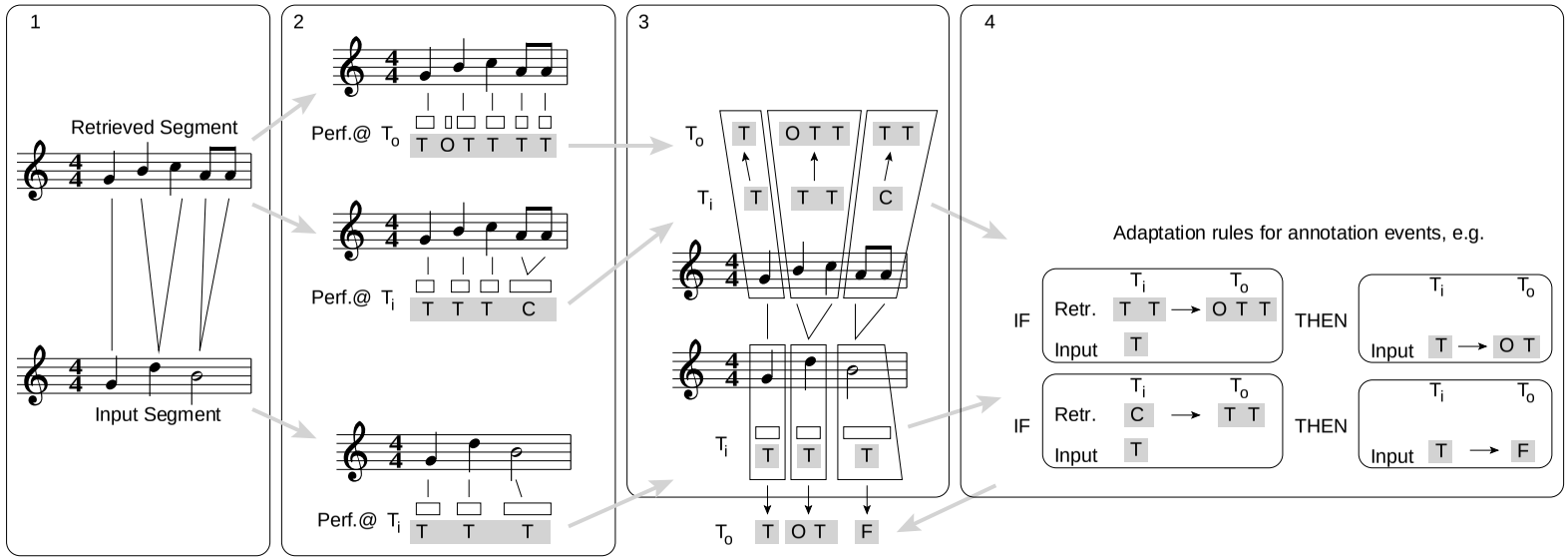


Figure 5.9: The process of hypothesis generation. In step 1, a mapping is made between the input score segment and the most similar segment from the pool of retrieved segments. In step 2, the performance annotations for the tempos T_i and T_o are collected. In step 3, the performance annotation events are grouped according to the mapping between the input score and retrieved score. In step 4, the annotation events are processed through a set of rules to obtain the annotation events for a performance at tempo T_o of the input score segment

Chapter 6

Conclusions and Planned Research

6.1 Resume

In the first two chapters of this research work, we have given an overview of prominent literature about expressive music performance. Firstly, we paid attention to fundamental and methodological issues. Secondly, we reviewed studies of expressive music performance that are geared towards expressive performance rendering.

Chapter 3 was devoted to Case Based Reasoning and its application in the field expressive music performance. In this chapter we also reviewed some kinds of music specific domain knowledge, like melodic similarity and theoretical frameworks for the analysis of music, that are likely to be useful in the application of CBR to music processing.

The goals of this research were discussed in chapter 4. In this chapter we briefly sketched our perspective on content based music processing, as the broader context of this research. The actual goals were limited to the design of a system for a particular kind of content based musical processing – global tempo transformations of monophonic jazz performances.

In chapter 5, we have presented a Case Based Reasoning system for applying global musical tempo transformations. This system operates on melodic descriptions of monophonic audio. Together with a sound analysis and sound synthesis component (in development in the context of the Tabasco and PROMUSIC projects), it provides a high-level music content processing tool for audio.

6.2 Current Results and Contributions

In this section, we briefly mention the current results of our research, as presented in several publications [4, 47, 52, 53, 55, 50, 54].

6.2.1 Comparison of Melodic Distance Measures

We have compared four edit-distance based melodic similarity measures. Each of the measures compared melodies using a different representation. The melodies were represented respectively as sequences of notes, melodic intervals, melodic directions, and I/R structures. With each of the measures, the pairwise distances were calculated for a set of about 125 phrases from about 35 songs from the Real Book of Jazz. The main conclusions were that the note representation had a relatively low discriminative power (defined as the entropy of the distance distribution of the measure) for the whole data set, but was good at recognizing phrases of the same song that were close variants of each other. The interval, direction and I/R representations had similar discriminative power, but the I/R representations gave better results for the recognition of phrases from the the same song. Since the interval and direction representations only contain information derived from pitch and do not capture any rhythmic information (which can be argued to be too abstract), we adapted the representations to include note durations. This decreased the discriminatory power, but increased the recognition of phrases from the same song. Using these two criteria, the interval representation with durational information had slightly better results than the I/R representation.

These results are useful for the retrieval part of our CBR system, where melodic similarity is assessed as a criterion for retrieval. But the results may also be relevant for music retrieval in general, as for example in *query-by-humming* systems, where melodic similarity is frequently used for retrieval.

Related publications:

- M. Grachten and J. Ll. Arcos. Using the Implication/Realization Model for Measuring Melodic Similarity. In *Proceedings of the 16th European Conference on Artificial Intelligence, ECAI 2004*. IOS Press, 2004.

- M. Grachten, J. Ll. Arcos, and R. López de Mántaras. Melodic similarity: Looking for a good abstraction level, 2004. To appear in Proceedings of ISMIR 2004.

6.2.2 Performance Annotation

We have presented a basic annotation scheme to represent expressive information from performances. The annotation consists of a sequence of *performance events*, that describe how the performance relates to the score. These events capture in some sense the musical behavior of the musician while performing a melody. For example, if the musician ornamented a certain notes while playing, this is represented by an *Ornamentation Event*. Alternatively, changing the timing, duration or other attributes of notes, is reflected by *Transformation Events*. The events are not just labels, but complex structures that have references to elements from the score and the performance, and can hold additional information such as the quantity of timing or duration deviations (in the case of *Transformation Events*). Other performance events are *Consolidation Events*, *Fragmentation Events*, *Insertion Events*, and *Deletion Events*. The annotations are constructed by mapping the performance events to corresponding edit operations. With this set of edit operations, an optimal alignment between score and performance (using the edit distance) is computed. The alignment (which is a sequence of edit operations), can easily be converted in a performance annotation (which is a sequence of performance events), through the correspondence between edit operations and performance events.

This annotation scheme should be regarded as an improvement for expressive performance rendering, in the sense that it allows a variety of phenomena (such as ornamentation, consolidation etcetera) to be treated as expressive gestures. Contrastingly, most existing expressive performance rendering/analysis systems limit the degrees of freedom of expressive change to varying the timing, duration and dynamics of notes.

Related publications:

- J. Ll. Arcos, M. Grachten, and R. López de Mántaras. Extracting performer s behaviors to annotate cases in a CBR system for musical tempo transformations. In *Proceedings of the Fifth International Conference on Case-Based Reasoning (ICCBR-03)*, 2003. (ICCBR-03 Best Paper Award)

6.2.3 Evolutionary Optimization of the Performance Annotation Process

We proposed a further enhancement of the performance annotation process just presented. This enhancement consists in the application of a genetic algorithm to tune the parameters of the cost functions that determine the cost

of edit operations. Since the optimal alignment obtained by computing the edit distance between two sequences is fully dependent on the cost functions of the edit operations, the cost functions should be designed in such a way that the resulting alignments correspond to the correct performance annotations. We noticed that untuned (that is, with randomly set parameters) cost functions lead to an average accuracy of about 76% when annotating a set of 40 performances. Manual tuning did improve the accuracy for small sets of performances, but we were not able to improve the accuracy on the complete set of 40 performance by manual tuning. Genetic algorithms (GA) were run several times on two training sets of 20 performances (for which the correct annotations were provided), and the parameter settings found by the GAs, resulted in annotations accuracies ranging from 92% to 97% for the test set. This is a substantial improvement over untuned settings.

Additionally, we were interested in the question whether the parameter settings obtained through different trials of the GAs with different training sets, all converged to a single set of values. This did not seem to be the case in general, but there were some trials (using different training sets!) that lead to very much resembling parameter settings (with correlations up to 97%). A statistical test is needed however to conclude whether this correlation is significant.

Related publications:

- M. Grachten, J. Ll. Arcos, and R. López de Mántaras. Evolutionary optimization of music performance annotation. In *CMMR 2004*, Lecture Notes in Computer Science. Springer, 2004.

6.2.4 Development of a Prototype of the System Architecture

Finally, we have developed a prototype of the system architecture described in chapter 5. The system loads a MIDI file containing a monophonic phrase, and an XML file containing a melodic description of some performance of that phrase (the tempo of that performance should be specified – it is not inferred from the performance). Furthermore a desired output specified. The input problem specification is constructed from those data automatically: an I/R analysis of the MIDI score is made, and the performance is annotated. Then the CBR system is consulted to solve the problem. The result is a new performance annotation, which is used to convert the old melodic description into to a new one. The XML file containing the new melodic description can be used as the input to a sound synthesis module (such as the saxophone synthesizer *Salto* [56]).

Related publications:

- M. Grachten, J. Ll. Arcos, and R. López de Mántaras. Tempo-express: tempo transformations preserving musical expression. In *Working notes of the IJCAI03 Rencon Workshop*, 2003. Online: *http://shouchan.ei.tuat.ac.jp/˜rencon/IJCAI-03/IJCAI_Rencon2003_FINALLAST.pdf*.

- M. Grachten, J. Ll. Arcos, and R. López de Mántaras. TempoExpress, a CBR Approach to Musical Tempo Transformations. In *Advances in Case-Based Reasoning. Proceedings of the 7th European Conference, ECCBR 2004*, Lecture Notes in Computer Science. Springer, 2004.

6.3 Future Directions

6.3.1 Better Performance Representation

One aspect of the system that needs more attention is the representation of the performances. Through the performance annotation, performed notes are related to the notes in the score, and note onset, note durations, and pitch deviations are calculated where appropriate. An indication of dynamic intensity is given by a mean energy value per performed note. This representation is limited in the sense that it does not incorporate the following aspects of expressivity:

Articulation An important aspect of expressive performances is the articulation of notes, that is, whether two consecutive notes are played *legato*, or *staccato* and to what extent. Since this feature intrinsically belongs to note transitions rather than notes, a natural extension of the melodic representation of performances might be the inclusion of *note transition* objects in addition to just *note* objects. The note transition objects reflect the duration of the micro pause between two notes in case of a staccato transition, and in case of a legato transition, it represents the dynamic intensity level of the transition.

Within Note Dynamics Another aspect of expressivity in the dimension of dynamic intensity is the evolution of the dynamic curve through the sounding of notes. This curve conveys the characteristics of note attack, steady state and decay. Since the raw dynamic curve is probably hard to work with in the problem solving process, a good approach may be to try to segment the curves for a set of notes and try to cluster the curve segments into a small number of groups. Alternatively, dynamic curves could be approximated by e.g. spline models or Bezier curves,

and the parameters of the approximation can serve as a representation of the dynamic curve.

Glissando and Vibrato Just like the dynamic curves, the fundamental frequency curves of the notes convey expressive information, e.g. *glissando* (a continuous approach to the note pitch from above or below), or *vibrato* (the oscillation of the note pitch around a certain value). Glissandi might be represented by onset time, onset frequency, offset time (defining the offset time as the moment the frequency curve has reached the frequency corresponding note pitch).

Timbre Finally, the timbre of the notes should be characterized in order to have a more complete description of the musical expressivity. This is a more remote possibility improving the performance representation. It may be possible to use the spectral centroid curve for deriving timbre characteristics.

6.3.2 Tools for Evaluating the Quality of the Case Base

As will have become clear from chapter 3, and subsection 3.3.6, the size and contents of the case base determine to a large extent the performance of the system as a whole. In the current implementation of the system, there is no mechanism for case base evaluation or maintenance.

The case base can be metaphorically conceived of as a geographical map of the problem domain. Each of the cases in the case base 'illuminates' a small region of the map, together forming the area of the problem domain where good solutions can be found. Our aim is to determine where are the dark areas on the map, and to add cases to the case base in a directed manner, to illuminate those remaining dark areas.

The topology of the map can be interpreted in multiple ways. One way is to conceive of the map as representing melodic distances between melodies of the cases. The goal is then to provide tempo transformation examples for a wide range of melodies. Another interpretation of the topology is to take the map as representing different phenomena in the performances of the melodies. For example, it may that we observe that there are too little examples of note insertions to use incorporate this type of performance event into a newly constructed performance. In that case, it may be useful to instruct a performer to play melodies in such a manner that he intentionally inserts notes where he or she thinks it is appropriate, and add the performances to the case base. This is a straight-forward manner of amplifying particular areas of implicit case knowledge.

The other direction of evaluating the quality of the case base is to measure its redundancy. If particular musical fragments are performed many times in rather similar ways at the same tempo, it is unnecessary to keep all performances, since one performance will suffice to serve as an example. The additional value of the redundancy – the level of confidence in the validity of the performance, can be maintained by adding a confidence value to each performance in the case base, which increases with the redundancy encountered for that performance.

6.3.3 System Evaluation

Until now, we have mainly concentrated on design issues and the development of the various components of the system (like automatic case acquisition, retrieval methods and adaptation). Most of those subsystems have been evaluated individually. The accuracy of the performance annotation process (being part of automatic case acquisition) was evaluated by checking the estimated annotations against manual annotations. The I/R parser (the other major part of automatic case acquisition) was informally validated by checking its analyses for some example scores reviewed by Narmour [87]. The retrieval step, employing different melodic similarity measures, was partly evaluated by presenting a comparison of the various similarity measures, and stating their merits and disadvantages for particular kinds of use. The adaptation part, used for inferring performances for the problem score from performances of retrieved score fragments, needs more detailed evaluation.

The next step is to evaluate the performance of the system as a whole. The question how to do this is far from trivial, and depends on the statement of the goals. For example, a system that is supposed to generate or transform performances in such a way that the resulting performance is musically satisfactory, or pleasing to the ear, should obviously be evaluated in a different way than a system that is supposed to predict as accurately as possible the characteristics of a human performance given certain constraints (such as global tempo). Which is the goal of our tempo transformation system? This is not a fully settled issue, but we tend to favor the criterion of 'musical acceptability' over the criterion of predictive accuracy. The main reason for this is that intuitively, predictive accuracy is not a definitive evaluation criterion, since it is commonly accepted that there are many different ways to perform a particular piece of music. The origins of these differences may be partly tractable (e.g. they may be explained by different structural interpretations of the score), but as long as these differences cannot be fully accounted for, a failure of the system to predict the characteristics of a *particular* human performance accurately, may not invalidate the prediction as

a good performance of the piece.

The problem with musical acceptability as a criterion is of course that this term is blatantly undefinable. It is impossible to describe what aspects of a performance make it musically (un)acceptable, since what may be acceptable to one person may be unacceptable to another. A more easily measurable criterion, related to musical acceptability, is *indiscernibility*. The performance of the system can be evaluated using this criterion by doing a blind test, where human listeners are to judge whether a particular performance was generated by the system or rather by a human performer. The percentage of machine generated performances that were mistaken for human performances can be taken as a measure for the performance of the system. As a final evaluation of our tempo transformation system, we intend to carry out such an experiment.

Bibliography

[1] A. Aamodt. Towards expert systems that learn from experience. In *Case Based Reasoning Workshop*, pages 181–187, Pensacola Beach, 1989. DARPA, Morgan Kaufmann.

[2] Agnar Aamodt. Knowledge-intensive case-based reasoning and sustained learning. In *European Conference on Artificial Intelligence*, pages 1–6, 1990.

[3] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1):39–59, 1994. Online: *http://www.iiia.csic.es/People/enric/AICom_ToC.html*.

[4] J. Ll. Arcos, M. Grachten, and R. López de Mántaras. Extracting performer's behaviors to annotate cases in a CBR system for musical tempo transformations. In *Proceedings of the Fifth International Conference on Case-Based Reasoning (ICCBR-03)*, 2003.

[5] J. Ll. Arcos, R. Lopez de Mantaras, and X. Serra. Saxex : a case-based reasoning system for generating expressive musical performances. In *Proceedings of the International Computer Music Conference 1997*, pages 329–336, 1997.

[6] Josep Lluís Arcos. Saxex: un sistema de raonament basat en casos per a l'expressivitat musical. Master's thesis, Institut Universitari de l'Audiovisual. Universitat Pompeu Fabra, 1996.

[7] Josep Lluís Arcos. *The Noos representation language*. PhD thesis, Universitat Politècnica de Catalunya, 1997. online at `www.iiia.csic.es/~arcos/Phd.html`.

[8] Josep Lluís Arcos, Ramon López de Mántaras, and Xavier Serra. Saxex : a case-based reasoning system for generating expressive musical per-

formances. In *International Computer Music Conference (ICMC'97)*, 1997.

[9] Josep Lluís Arcos and Enric Plaza. Reflection in Noos: An object-centered representation language for knowledge modelling. In *IJCAI-95 Workshop on Reflection and Meta-Level Architectures and their Applications in AI*, 1995.

[10] Josep Lluís Arcos and Enric Plaza. Noos: An integrated framework for problem solving and learning. In *Knowledge Engineering: Methods and Languages*, 1997.

[11] K. D. Ashley. *Modeling Legal Arguments: Reasoning with Cases and Hypotheticals*. MIT Press, Bradford Books, Cambridge, MA, 1990.

[12] I. Bengtsson and A. Gabrielsson. Methods for analyzing performance of musical rhythm. *Scandinavian Journal of Psychology*, 21:257–268, 1980.

[13] J. Bilmes. A model for musical rhythm. In *ICMC Proceedings*, pages 207–210. Computer Music Association, 1992.

[14] A. Binet and J. Courtier. Recherches graphiques sur la musique. L'année Psychologique (2), 201–222, 1896.

[15] K. Branting. Exploiting the complementarity of rules and precedents with reciprocity and fairness. In *Case Based Reasoning Workshop*, pages 39–50, Washington DC, 1991. DARPA, Morgan Kaufmann.

[16] A. S. Bregman. *Auditory Scene Analysis*. MIT Press, Cambridge, MA, 1990.

[17] R. Bresin and A. Friberg. Emotional coloring of computer-controlled music performances. *Computer Music Journal*, 24(4):44–63, 2000.

[18] E. Cambouropoulos. The local boundary detection model (lbdm) and its application in the study of expressive timing. In *Proceedings of the International Computer Music Conference (ICMC'2001)*, Havana, Cuba, 2001.

[19] Sergio Canazza, Giovanni De Poli, Stefano Rinaldin, and Alvise Vidolin. Sonological analysis of clarinet expressivity. In Marc Leman, editor, *Music, Gestalt, and Computing: studies in cognitive and systematic musicology*, number 1317 in Lecture Notes in Artificial Intelligence, pages 431–440. Springer, 1997.

[20] Sergio Canazza, Giovanni De Poli, and Alvise Vidolin. Perceptual analysis of musical expressive intention in a clarinet performance. In Marc Leman, editor, *Music, Gestalt, and Computing: studies in cognitive and systematic musicology*, number 1317 in Lecture Notes in Artificial Intelligence, pages 441–450. Springer, 1997.

[21] Jaime Carbonell. Derivational analogy and its role in problem solving. In *Proceedings of the national conference on Artificial Intelligence*, pages 64–69, 1983. AAAI-83.

[22] Jaime Carbonell. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning*, volume 2, pages 371–392. Morgan Kaufmann, 1986.

[23] R. Cilibrasi, R. de Wolf, and P. Vitányi. Algorithmic clustering of music, 2003. Submitted. Online: *http://arxiv.org/archive/cs/0303025*.

[24] E. F. Clarke. Generative principles in music. In J.A. Sloboda, editor, *Generative Processes in Music: The Psychology of Performance, Improvisation, and Composition*. Oxford University Press, 1988.

[25] E. F. Clarke. Expression and communication in musical performance. In Johan Sundberg, Lennart Nord, and Rolf Carlson, editors, *Music, Language, Speech and Brain*. MacMillan Academic and Professional Ltd, 1991.

[26] S. Cohen. Finding color and shape patterns in images. Technical report, Stanford University, 1999. STAN-CS-TR-99-1620.

[27] G. L. Collier and J. L. Collier. An exploration of the use of tempo in jazz. *Music Perception*, 11:219–242, 1994.

[28] Computational Intelligence. Blackwell Publishers, 2001. Volume 17, Nr. 2.

[29] L. L. Cuddy and C. A. Lunney. Expectancies generated by melodic intervals: Perceptual judgments of melodic continuity. *Perception & Psychophysics*, 57:451–462, 1995.

[30] R. Dannenberg. An on-line algorithm for real-time accompaniment. In *Proceedings of the 1984 International Computer Music Conference*. International Computer Music Association, 1984.

[31] DARPA. *Case Based Reasoning Workshop*, Pensacola Beach, 1989. Morgan Kaufmann.

[32] G. De Poli, Canazza S., A Rodà, A. Vidolin, and P. Zanon. Analysis and modeling of expressive intentions in music performance. In *Proceedings of the International Workshop on Human Supervision and Control in Engineering and Music*, Kassel, Germany, September 21–24 2001.

[33] I. Deliège. Similarity in processes of categorisation: Imprint formation as a prototype effect in music listening. In M. Ramscar, U. Hahn, E. Cambouropoulos, and H. Pain, editors, *Proceedings of the Interdisciplinary Workshop on Similarity and Categorisation*, pages 59–65, Edinburgh, 1997. University of Edinburgh.

[34] I. Deliège. Introduction: Similarity perception $\leftrightarrow$ categorization $\leftrightarrow$ cue abstraction. *Music Perception*, 18(3):233–244, 2001.

[35] P. Desain and H. Honing. Towards a calculus for expressive timing in music. *Computers in Music Research*, 3:43–120, 1991.

[36] P. Desain and H. Honing. Tempo curves considered harmful. In "Time in contemporary musical thought" J. D. Kramer (ed.), Contemporary Music Review. 7(2), 1993.

[37] P. Desain and H. Honing. Does expressive timing in music performance scale proportionally with tempo? *Psychological Research*, 56:285–292, 1994.

[38] P. Desain and H. Honing. The formation of rhythmic categories and metric priming. *Perception*, 32(3):341–365, 2003.

[39] P. Desain, H Honing, and H. Heijink. Robust score-performance matching: Taking advantage of structural information. In *Proceedings of the 1997 International Computer Music Conference*, pages 337–340, San Francisco, 1997. International Computer Music Association.

[40] A. Friberg. Generative rules for music performance: A formal description of a rule system. *Computer Music Journal*, 15 (2):56–71, 1991.

[41] A. Friberg and G. U. Battel. Structural communication. In R. Parncutt and G.E. McPherson, editors, *The science and psychology of music performance*, chapter 13, pages 199–218. Oxford University Press, 2002.

[42] A. Gabrielsson. Once again: The theme from Mozart's piano sonata in A major (K. 331). A comparison of five performances. In A. Gabrielsson, editor, *Action and perception in rhythm and music*, pages 81–103. Royal Swedish Academy of Music, Stockholm, 1987.

[43] A. Gabrielsson. Expressive intention and performance. In R. Steinberg, editor, *Music and the Mind Machine*, pages 35–47. Springer-Verlag, Berlin, 1995.

[44] A. Gabrielsson. The performance of music. In D. Deutsch, editor, *The Psychology of Music*, chapter 14, pages 501–602. Academic Press, 1999.

[45] D. Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, 1983.

[46] E. Gómez, F. Gouyon, P. Herrera, and X. Amatriain. Using and enhancing the current mpeg-7 standard for a music content processing tool. In *Proceedings of Audio Engineering Society, 114th Convention*, Amsterdam, The Netherlands, 2003.

[47] E. Gómez, M. Grachten, X. Amatriain, and J. Ll. Arcos. Melodic characterization of monophonic recordings for expressive tempo transformations. In *Proceedings of Stockholm Music Acoustics Conference 2003*, 2003.

[48] E. Gómez, A. Klapuri, and B. Meudic. Melody description and extraction in the context of music content processing. *Journal of New Music Research*, 32(1), 2003.

[49] M. Grachten. Summary of the Music Performance Panel, MOSART Workshop 2001, Barcelona. Unpublished. Online: *http://www.iua.upf.es/mtg/mosart/panels/music-performance.pdf*.

[50] M. Grachten and J. Ll. Arcos. Using the Implication/Realization Model for Measuring Melodic Similarity. In *Proceedings of the 16th European Conference on Artificial Intelligence, ECAI 2004*. IOS Press, 2004.

[51] M. Grachten, J. Ll. Arcos, and R. López de Mántaras. A comparison of different approaches to melodic similarity, 2002. Second International Conference on Music and Artificial Intelligence (ICMAI).

[52] M. Grachten, J. Ll. Arcos, and R. Lopez de Mantaras. Tempo-express: tempo transformations preserving musical expression. In *Working notes of the IJCAI03 Rencon Work-*

shop, 2003. Online: *http://shouchan.ei.tuat.ac.jp/˜rencon/IJCAI-03/IJCAI_Rencon2003_FINALLAST.pdf*.

[53] M. Grachten, J. Ll. Arcos, and R. López de Mántaras. Evolutionary optimization of music performance annotation. In *CMMR 2004*, Lecture Notes in Computer Science. Springer, 2004.

[54] M. Grachten, J. Ll. Arcos, and R. López de Mántaras. Melodic similarity: Looking for a good abstraction level, 2004. To appear in Proceedings of ISMIR 2004.

[55] M. Grachten, J. Ll. Arcos, and R. López de Mántaras. TempoExpress, a CBR Approach to Musical Tempo Transformations. In *Advances in Case-Based Reasoning. Proceedings of the 7th European Conference, ECCBR 2004*, Lecture Notes in Computer Science. Springer, 2004.

[56] J. Haas. Salto - a spectral domain saxophone synthesizer. In *Proceedings of MOSART Workshop on Current Research Directions in Computer Music*, Barcelona, 2001.

[57] H. Heijink. Matching scores and performances. Master's thesis, KUN, Nijmegen, 1996. Unpublished.

[58] H. Heijink, P. Desain, H. Honing, and L. Windsor. Make me a match: An evaluation of different approaches to score-performance matching. *Computer Music Journal*, 24(1):43–56, 2000.

[59] M. T. Henderson. Rhythmic organization in artistic piano performance. *University of Iowa studies in the psychology of music*, IV:281–305, 1937. Univ. Press Iowa.

[60] T. R. Hinrichs. *Problem solving in open worlds : a case study in design*. L. Erlbaum Associates, Hillsdale, N.J., 1992.

[61] K. J. Holyoak and P. Thagard. Analogical mapping by constraint satisfaction. *Cognitive Science*, 13:295–355, 1989.

[62] H Honing. From time to time: The representation of timing and tempo. *Computer Music Journal*, 35(3):50–61, 2001.

[63] H. Honing. Structure and interpretation of rhythm and timing. *Tijdschrift voor Muziektheorie*, 7(3):227–232, 2002.

[64] V. Iyer. Embodied mind, situated cognition, and expressive microtiming in african-american music. *Music Perception*, 19(3):387–414, 2002.

[65] V. Iyer, J. Bilmes, M. Wright, and D. Wessel. A novel representation for rhythmic structure. In *ICMC Proceedings*, 1997.

[66] J. A. Johnstone. *Phrasing in piano playing*. Witmark, New York, 1913.

[67] P.N. Juslin. Communicating emotion in music performance: a review and a theoretical framework. In P.N. Juslin and J.A. Sloboda, editors, *Music and emotion: theory and research*, pages 309–337. Oxford University Press, New York, 2001.

[68] K. Koffka. *Principles of Gestalt Psychology*. Routledge & Kegan Paul, London, 1935.

[69] W. Köhler. *Gestalt psychology: An introduction to new concepts of modern psychology*. Liveright, New York, 1947.

[70] J. L. Kolodner. Reconstructive memory, a computer model. *Cognitive Science*, 7:281–328, 1983.

[71] J. L. Kolodner. An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1):3–34, 1992.

[72] P. Koton. Reasoning about evidence in causal explanations. In *Proceedings of AAAI-88*, Cambridge, MA, 1988. AAAI Press/MIT Press.

[73] P. Koton. Using experience in learning and problem solving. Technical Report 90/2, MIT/LCS, 1989.

[74] A. Lamont and N. Dibben. Motivic structure and the perception of similarity. *Music Perception*, 18(3):245–274, 2001.

[75] S. Langer. *Feeling and Form: a Theory of Art*. Scribner, New York, 1953.

[76] E. W. Large. Dynamic programming for the analysis of serial behaviors. *Behavior Research Methods, Instruments & Computers*, 25(2):238–241, 1993.

[77] F. Lerdahl. Calculating tonal tension. *Music Perception*, 13(3):319–363, 1996.

[78] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, 1983.

[79] F. Lerdahl and R. Jackendoff. An overview of hierarchical structure in music. In S. M. Schwanaver and D. A. Levitt, editors, *Machine Models of Music*, pages 289–312. The MIT Press, 1993. Reproduced from Music Perception.

[80] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.

[81] E. Lindström. 5 x "oh, my darling clementine". the influence of expressive intention on music performance, 1992. Department of Psychology, Uppsala University.

[82] L.B. Meyer. *Emotion and meaning in Music*. University of Chicago Press, Chicago, 1956.

[83] M. Minsky. A framework for representing knowledge. In P. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, 1975.

[84] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[85] M. Mongeau and D. Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24:161–175, 1990.

[86] Music Perception. An Interdisciplinary Journal. Ed. R. Gjerdingen. University of California Press, 2002. Volume 19, Nr. 3.

[87] E. Narmour. *The Analysis and cognition of basic melodic structures : the implication-realization model*. University of Chicago Press, 1990.

[88] E. Narmour. *The Analysis and cognition of melodic complexity: the implication-realization model*. University of Chicago Press, 1992.

[89] C. Palmer. *Timing in skilled music performance*. PhD thesis, Cornell University, 1988.

[90] C. Palmer. Anatomy of a performance: Sources of musical expression. *Music Perception*, 13(3):433–453, 1996.

[91] C. Palmer. Music performance. *Annual Review of Psychology*, 48:115–138, 1997.

[92] C. (L.) E. Peper, P. J. Beek, and P. C. W. Van Wieringen. Frequency-induced phase transitions in bimanual tapping. *Biological Cybernetics*, 73:301–309, 1995.

[93] E. Plaza and J. Ll. Arcos. Constructive adaptation. In Susan Craw and Alun Preece, editors, *Advances in Case-Based Reasoning*, number 2416 in Lecture Notes in Artificial Intelligence, pages 306–320. Springer-Verlag, 2002.

[94] M. Puckette and A. C. Lippe. Score following in practice. In *Proceedings, International Computer Music Conference*, pages 182–185, San Francisco, 1992. International Computer Music Association.

[95] R. Rasch. Synchronization in performed ensemble music. *Acustica*, 43:121–131, 1979.

[96] P. Reinholdsson. Approaching jazz performance empirically: Some reflections on methords and problems. In A. Gabrielsson, editor, *Action and perception in rhythm and music*, pages 105–125. Royal Swedish Academy of Music, Stockholm, 1987. Publication Nr 55.

[97] B. H. Repp. Relational invariance of expressive microstructure across global tempo changes in music performance: An exploratory study. *Psychological Research*, 56:285–292, 1994.

[98] B. H. Repp. Expresive timing in Schumann's "Träumerei": An analysis of performances by graduate student pianists. *Journal of the Acoustical Society of America*, 98(5):2413–2427, 1995.

[99] B. H. Repp. Quantitative effects of global tempo on expressive timing in music performance: Some perceptual evidence. *Music Perception*, 13(1):39–58, 1995.

[100] B. H. Repp, L. Windsor, and P. Desain. Effects of tempo on the timing of simple musical rhythms. *Music Perception*, 19(4):565–593, 2002.

[101] C. Riesbeck and R. Schank, editors. *Inside Case Based Reasoning*. Lawrence Erlbaum, 1989.

[102] M.G. Rigg. The mood effects of music: a comparison of data from former investigations. *Journal of Psychology*, 58:427–438, 1964.

[103] B. H. Ross. Some psychological results on case based reasoning. In *Case Based Reasoning Workshop*, pages 144–147, Pensacola Beach, 1989. DARPA, Morgan Kaufmann.

[104] R. Schank. *Dynamic Memory: a theory of reminding and learning in computers and people*. Cambridge University Press, 1982.

[105] E. G. Schellenberg. Expectancy in melody: Tests of the implication-realization model. *Cognition*, 58:75–125, 1996.

[106] E. G. Schellenberg. Simplifying the implication-realization model of melodic expectancy. *Music Perception*, 14(3):295–318, 1997.

[107] C. E. Seashore. *Psychology of Music*. McGraw-Hill, New York, 1938. (Reprinted 1967 by Dover Publications New York).

[108] X. Serra. Musical sound modeling with sinusoids plus noise. In C. Roads, S. T. Pope, A. Picialli, and G. De Poli, editors, *Musical Signal Processing*, pages 91–122. Swets and Zeitlinger Publishers, 1997.

[109] X. Serra, J. Bonada, P. Herrera, and R. Loureiro. Integrating complementary spectral methods in the design of a musical synthesizer. In *Proceedings of the ICMC'97*, pages 152–159. San Francisco: International Computer Music Asociation., 1997.

[110] J. A. Sloboda. The communication of musical metre in piano performance. *Quarterly Journal of Experimental Psychology*, 35A:377–396, 1983.

[111] J. Sundberg, Anders Friberg, and Lars Frydén. Common secrets of musicians and listeners: an analysis-by-synthesis study of musical performance. In P. Howell, R. West, and I. Cross, editors, *Representing Musical Structure*, Cognitive Science series, chapter 5. Academic Press Ltd., 1991.

[112] J. Sundberg, Anders Friberg, and Lars Frydén. Threshold and preference quantities of rules for music performance. *Music Perception*, 9:71–92, 1991.

[113] D. Temperley. *The Cognition of Basic Musical Structures*. MIT Press, Cambridge, Mass., 2001.

[114] P. Thagard and K. J. Holyoak. Why indexing is the wrong way to think about analog retrieval. In *Case Based Reasoning Workshop*, pages 36–40, Pensacola Beach, 1989. DARPA, Morgan Kaufmann.

[115] R. Timmers and H. Honing. On music performance, theories, measurement and diversity. In M.A. Belardinelli (ed.). Cognitive Processing (International Quarterly of Cognitive Sciences), 1-2, 1-19, 2002.

[116] N. P. Todd. A model of expressive timing in tonal music. *Music Perception*, 3:33–58, 1985.

[117] N.P. Todd. A computational model of rubato. *Contemporary Music Review*, 3 (1), 1989.

[118] R. Typke, P. Giannopoulos, R. C. Veltkamp, F. Wiering, and R. van Oostrum. Using transportation distances for measuring melodic similarity. In *Proceedings of the Fourth International Conference on Music Information Retrieval (ISMIR)*, 2003.

[119] G. Widmer. Large-scale induction of expressive performance rules: First quantitative results. In *Proceedings of the International Computer Music Conference (ICMC2000)*, San Francisco, CA, 2000. International Computer Music Association.

[120] G. Widmer. In search of the Horowitz factor: Interim report on a musical discovery project. In *Proceedings of the 5th International Conference on Discovery Science DS'02*, Lübeck, Germany, 2002. Berlin: Springer Verlag.

[121] G. Widmer. Machine discoveries: A few simple, robust local expression principles. *Journal of New Music Research*, 31(1):37–50, 2002.

[122] G. Widmer. Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence*, 146(2):129–148, 2003.

[123] W. Wilke and R. Bergmann. Techniques and knowledge used for adaptation during case-based problem solving. *IEA/AIE*, 2:497–506, 1998.

[124] David C. Wilson and David B. Leake. Maintaining case-based reasoners: Dimensions and directions. *Computational Intelligence*, 17(2):196–213, 2001.

Index