Chapter 5

From Sheet Music To Note Events

An important step for the content-based comparison of sheet music data and audio data is the generation of note events specified by absolute pitches, onset times, and durations (see Figure 2.2) from the sheet music data. This task requires several steps. First, the sheet music is converted to symbolic score data through optical music recognition (OMR). The symbolic score data we obtain from the OMR results basically consists of a notation bar sequence $\nu$ (see Figure 3.2) with each bar $\nu_b \in \nu$ containing a set of symbols, e.g., clefs, key signatures, notes, and rests. Additionally, for each page, there is a set of global symbols that are not (yet) attached to any single bar in particular, e.g., textual directives, slurs, and dynamics symbols. The task, now, is to correctly derive note events specified by onset times, pitches and durations from the symbolic score data. Presuming that the symbolic score data is complete and accurate with respect to all the symbols that affect these parameters, this task can be solved by applying the basic rules of music notation. However, in practice, not all the relevant information is output by the OMR process, and some part of the output information is inaccurate or unreliable. In this chapter, we discuss strategies for both dealing with the most critical shortcomings of the OMR output in a streamlined way and for making best use of the available data in our scenario. In Section 5.1, we discuss the task of optical music recognition and point out the most critical shortcomings for our project. In Sections 5.2 and 5.3 we present approaches to encounter some of these shortcomings via postprocessing techniques. To actually derive note events specified by onset times, durations, and pitches, one important task is to determine the sequence in which the bars are to be played, i.e., the default bar sequence $\delta$. The task of deriving an estimate $\delta^*$ by detecting repeats and jumps in the score is discussed in Section 5.4. A robust method for deriving onset times is presented in Section 5.5. The principles of deriving pitches by applying the basic rules of classical Western music notation are outlined in Section 5.6. Finally, in Section 5.7, we point out certain difficulties that can occur in orchestral scores and propose a streamlined method to manually encounter the aspects that are most critical for our application.
5.1 Optical Music Recognition

As already pointed out in Section 2.1, the term optical music recognition (OMR) is commonly used to refer to transformations from the sheet music domain to the symbolic domain. Actually, there is a wide spectrum of transformations depending on the kind of input and output data. For example, starting with a vector graphics image of a score, the task of assigning labels for musical entities such as note head, stem, treble clef, staff line to each of the shapes could already be considered as OMR. However, OMR usually covers a lot more than that. For example, when starting with a bitmap image, one crucial subtask of OMR consists in grouping the pixels to musically meaningful shapes and relating these shapes to musical entities.

OMR systems can differ in how far and deep they try to reach into the symbolic realm with their transformation. Many OMR systems stop after having identified most of the shapes as musical symbols and having interpreted their basic meaning and relations. Shapes that are not recognized are often ignored. Furthermore, higher level semantics are often neglected. For example, an OMR system might recognize repeat signs, but not necessarily does it also interpret their musical meaning, i.e., the effect on the sequence of bars that is to be followed when playing the piece. As another example, consider text-based information contained in sheet music. Most OMR systems are able to recognize text and even to distinguish between song lyrics and other text. However, the systems usually do not further distinguish other possible functions of the text elements, such as title heading, section name, tempo directive, jump directive (da capo, fine, etc.), or instrument name.

Several commercial and non-commercial OMR software systems exist. Three of the more popular commercial systems that operate on common Western classical music are SharpEye, SmartScore, and PhotoScore. Two of the more prominent examples of free OMR systems are Gamera and Audiveris. However, Gamera is actually a more general tool for document image analysis that requires to perform training on the data to be recognized [39]. An OMR module for recognizing classical Western music in Gamera called AOMR2 has been developed and used in the literature [38, 37]. However, it seems to be no longer under development or available for download. Instead, there are OMR modules specialized on lute tablature and Psaltic neume notation. Audiveris is an open-source OMR software written in Java, but, currently, is not competitive in terms of recognition rates compared to the commercial products.

Evaluating and comparing the general performance of OMR systems is a non-trivial task. In particular, it is not clear how the performance is to be measured [37]. A comparison of some aspects of OMR systems mentioned above and a detailed discussion of the problems regarding evaluation of OMR can be found in the work by Byrd et al. [18, 17]. In general,
5.1. OPTICAL MUSIC RECOGNITION

the quality of OMR results strongly depends on the quality of the input image data and the complexity of the underlying scores. More complex scores tend to result in more OMR extraction errors. Here, typical OMR errors are non-recognized symbols, missing accidentals (also in key signatures), missing beams, missing or extra barlines, erroneous time signatures, and multi-staff grand staffs that are mistakenly split apart horizontally into smaller groups or individual staffs.

A general observation is that most commercially distributed OMR systems are designed to help users to create a symbolic representation from a printed and scanned score so it can be edited, rearranged, reprinted, or archived. In that case, the main purpose of the output symbolic representation is still to serve as an image that is to be read and interpreted by human readers. Therefore, the recognition of higher-level semantics by the software such as tempo, repeats, jumps (da capo, alternative endings), or link-ups of staffs in consecutive multi-staff systems, is not important. More important, in this use case, is that the musical symbols are correctly recognized. Since recognition errors are expected to occur, users are expected to correct these errors manually. The OMR systems provide editors and tools that assist in this task.

The demands on OMR in the use case of our digital music library project are very much different from the above. Since the symbolic score data created through OMR is only used in the process of organizing the content and calculating the synchronizations, but is never directly presented to the user, we do not require the OMR results to be perfectly accurate, or to even “look” right. We only need it to be accurate enough to allow our approaches to automatic organization and synchronization of the data to work. For our scenario, a thorough manual correction of the OMR results is not an option, because the amount of sheet music is simply too large. As of October 2009, there have been 50,000 pages of sheet music in 292 sheet music books available for the project. Considering, that the number of printed sheet music books available at the Bavarian State Library in Munich exceeds 100,000, it becomes clear that a manual correction of each page is not practical.

There are two more requirements on OMR software to be useful for our scenario. For the application of linking bars that are visible on sheet music scans presented to the user to temporal regions in corresponding audio recordings, we need the OMR output to contain accurate information on the position and size of each bar in the sheet music images. Finally, the software should provide a simple way of being integrated in a fully automatic processing chain. Unfortunately, the bigger commercially distributed OMR programs are bound to graphical user interfaces that require human interaction to perform the recognition. The OMR module of the SharpEye2 Music Reader used in our project, see Section 2.3 can be run as a command-line application, which makes it easy to integrate it into the automated workflow.

Our approaches for the automatic organization and synchronization operate on two kinds of data extracted from sheet music books. The first kind of data is structural information such as title headings and indentations of grand staffs that is useful for the track segmentation and identification of sheet music books, see Chapter 6. Because commercially distributed OMR software is intended to be used with individual pieces and movement, but not with complete sheet music books, recognition of titles and boundaries for the pieces comprised in the books is not provided. The second kind of data is musical content, which, in our case, we consider to
be the note events derived from the scores. Since our content-based algorithms work on these
note events, we need to extract note events from the OMR output as accurately as possible.
We discuss strategies for extracting such information from OMR results in the subsequent
sections of this chapter.

Printed sheet music usually follows a rich set of rules, regarding aspects of engraving, no-
tation, and layout. Furthermore, there are high-level rules that dictate which constellations
of symbols are valid. Because many of these rules are not incorporated into current OMR
software, they can be used to correct errors in the OMR output data through postprocessing.
Unfortunately, the set of valid rules can differ significantly between different types of scores,
music, and instrumentation. Special types of notation and exceptions are specific to particu-
lar types of music. To be able to make use of the rules that apply to the given material, one
first has to decide which rules actually can be applied. Assuming the type of music or instru-
mention is known, OMR postprocessing can be performed to fix some of the recognition
errors. We will discuss some pragmatic approaches for that in the following section.

5.2 Postprocessing OMR Data Using Simple Heuristics

It seems clear that there still has to be put a lot of effort into the development of OMR
software until all symbols and semantics of classical Western sheet music can be extracted
reliably. Our main goal is to get as much and as accurate information out of the results
of the OMR software used in this project as is needed to make automatic organization and
synchronization work in a digital library environment. Since the OMR engine used in this
software does not seem to make use of higher-level or instrumentation-specific rules, we can
detect and correct many of the errors in the OMR output using pragmatic ad-hoc approaches
in a postprocessing step.

Several ad-hoc approaches for detecting and correcting OMR errors as well as infering impor-
tant semantic aspects have been developed and implemented. However, as pointed out in the
previous section, the set of rules that can be expected to apply for the sheet music depends on
several factors such as instrumentation. Since in the early phase of the PROBADO project
the music collection consisted of piano music only, some of the rules used in these approaches
are specific to piano music. Even though some of these rules may apply more generally, in
the remainder of this section, we assume that the underlying material is piano music.

Grand staffs in scores for piano music usually consist of two staffs. The upper staff is played
with the right hand, and the lower staff is played with the left hand. However, there are
some exceptions to this. Unlike scores for ensembles where grand staffs are simply groups of
individual staffs that each form a coordinate system for time and pitch, the two staffs for the
left and right hand can be treated as a single coordinate system as shown in Figure 5.1. Here,
a note group that is connected through a single beam is spread across the two staffs. This
mechanism is often used to avoid notes with many auxiliary lines between the two staffs.

In the following, we discuss a set of modules that detect and correct errors and derive semantics
from OMR results of piano music based on simple heuristics. Each of these modules can be
5.2. POSTPROCESSING OMR DATA USING SIMPLE HEURISTICS

Figure 5.1. Example for a beamed note group that stretches over two staffs in piano music.

applied to the data individually. Even though these modules are rather simple and naive, they can eliminate many errors that would otherwise have a negative influence on the content-based comparison of note events and audio recordings.

5.2.1 Voice Detection

Our starting point is the OMR data output by the SharpEye software. For each bar, we consider symbols representing chords, notes, and rests. In the following, we will use the term chord to refer to both chords and individual notes, i.e., we consider individual notes as a special case of a chord containing just a single note. Let us use the term voice to refer to a sequence of chords and rests that are played consecutively and without a gap. In piano music, several voices that are to be played simultaneously can coexist on the two staff system. An example bar of piano music with three voices is depicted in Figure 5.2(a). To allow a reader of the score to better differentiate between individual voices, music notation usually follows certain conventions.

(VC1) Events that are meant to occur at the same time share the same horizontal position.
(VC2) Each chord or rest belongs to one voice only.
(VC3) For two voices written in a single staff, the stems of the upper voice point upwards, and the stems of the lower voice point downwards.
(VC4) Each voice individually fills out the duration of the complete measure with either rests or notes.
(VC5) Voices usually do not intersect or change position relative to each other.

In addition to these conventions, a common visual aid for the reader is that consecutive events that belong to the same voice can often be identified by being grouped by beams, slurs, or ties. In the cases where the above conventions hold, one can use a simple algorithm to determine the voices of a given bar of sheet music. Let us consider the example depicted in Figure 5.2. The algorithm is divided into two steps. In the first step, the amount of voices present in the bar is estimated by making use of (VC1) and (VC2). To this end, we group chords and rests that roughly share the same horizontal position into so-called raster groups. In Figure 5.2(b), we see that for our example bar we get 8 of these raster groups. Then, we assume the
Figure 5.2. Example for a bar of piano music with three voices. (a) The notes and rests of each voice are highlighted with different color. (b) The amount of voices in the bar is estimated by first grouping all chords and rests with the same horizontal position and then counting the maximum group size.

amount of voices that is present in the bar to be the maximum number of chords/rests found in any of these groups. In our example, this maximum number is 3, and it is found in the first, fifth, and seventh group. The second step is to map each chord/rest to one of the voices. We start with the groups where the maximum number of voices is present. Here, we assign voice numbers by vertical position. The upper chord/rest is assigned to voice 1, the middle chord is assigned to voice 2 and the lower chord is assigned to voice 3. Now the chords/rests of the remaining groups need to be assigned. This is done by using the above conventions for stem direction and non-intersection of voices that we assumed to hold and can be assisted by considering beams, slurs, and ties.

This rather crude and simple algorithm already works well in most cases, but certainly not in all cases. In situations where following the above conventions (VC1) to (VC5) would lead to overlapping symbols or degraded readability of the notation, these conventions are often suspended. Examples for such cases are depicted in Figures 5.3 through 5.5. Even though there may be again conventions on how such cases are usually handled, in general, the engraver may prefer custom solutions for each particular case that disrespect conventions in favor of improved readability. This makes finding an algorithm that always finds the correct voices very hard. In many cases, the voices can only be inferred by deciding which interpretation of the notation makes most sense in the current context. This decision requires not only pondering existing conventions and rules on several semantic levels simultaneously, but also the anticipation of intended meaning aside from the known rules of music notation. Some curious examples for such exceptional cases of music notation have been collected by Byrd.

Incorporating more conventions and indicators, e.g., note durations, into an algorithm for voice detection may lead to improved results compared to our simple approach described above. We will discuss a more general approach of incorporating conventions and indicators into postprocessing for correcting errors and deriving semantics in Section 5.3. However, considering that in our real-world scenario, the data must additionally be expected to suffer from OMR errors such as missing or additional chords and rests as well as errors in note durations, it may be necessary to incorporate additional error detection techniques.

http://www.informatics.indiana.edu/donbyrd/InterestingMusicNotation.html
5.2. POSTPROCESSING OMR DATA USING SIMPLE HEURISTICS

5.2.2 Key Signature Correction

In piano music, key signatures are expected to be the same for the left hand and right hand staff at any time. Therefore, different key signatures in the left-hand and right-hand staff of OMR results are typically caused by OMR errors, see Figure 5.6. In case of the SharpEye OMR engine, it is much more likely that accidentals are missed out rather than extra accidentals being added. Therefore, a good strategy to handle different key signatures in the left and right hand is to choose the one which has more accidentals. Since key signatures possibly affect the pitch of many notes throughout a whole staff of music, fixing these obvious errors can significantly improve the accuracy of note events extracted from the sheet music with little effort. Note, however, that this simple algorithm does not detect cases where staff signatures in both staffs have the same amount of missing accidentals.

In the sheet music books “Beethoven Piano Sonatas Volume 1 & 2” by G. Henle, disagreeing key signatures were found in 258 of a total of 3693 grand staffs, which is a rate of roughly 7%.

Figure 5.3. Example for a voice appering within a bar of piano music. Even though in the first bar of this example at the 4-th quarter note there are three voices, the middle voice is not preceded by rests to fill up the complete measure.

Figure 5.4. Example for having three voices written in one staff in piano music. To avoid overlapping symbols, the notes that coincide in onset time are no longer aligned horizontally.
CHAPTER 5. FROM SHEET MUSIC TO NOTE EVENTS

Figure 5.5. Example for a rather complex situation that comprises 8 voices in a single bar. However, in cases like this where most of the voices consist of a single tied note event, the technical differentiation between the individual voices is no longer important. The notation may be read as simply suggesting a sequence of eighth notes where each note is held until or beyond the end of the bar. Even though the result would be the same if of the 8 eighth notes, only the 4-th and 8-th one would have been written, the beamed note groups give a nice visual hint on what is the essential part of this bar of music.

5.2.3 Time Signature Correction

Similar to the case of key signatures, time signatures in piano music are expected to be the same for the left and right hand at all times. Disagreeing time signatures found in OMR results indicate recognition errors. However, in case of time signatures it is not clear which of the two disagreeing recognition results is the correct one, if any. To deal with this issue, in our simple approach, we calculate the durations of the voices found in the 10 bars that follow the time signature and check if they agree or disagree with the durations implied by the time signatures. Since the detection of voices and the duration of notes and rests is not perfectly reliable, we use a voting mechanism to find the predominant duration among the 10 bars. If one of the time signatures agrees with the predominant voice duration, the other time signature is changed accordingly.

If there is a very clearly predominant voice duration that disagrees with both of the found time signatures, it is likely that both time signatures were recognized incorrectly. In this case, a suitable time signature is generated from the predominant voice duration and the time signatures are replaced. An example is depicted in Figure 5.6.

5.2.4 Execution Hint Removal

Another issue for the automatic generation of note events from OMR data of piano music is execution hints that, by the OMR software, are recognized as regular grand staffs. Such execution hints are usually found at the bottom of the page being linked to a particular position in the score by some marker, see Figure 5.7. If not handled, this will cause bars and note events found in the execution hint to be appended after the last regular grand staff of the page.

We use a simple heuristic to identify execution hints by searching for single-staff grand staffs
5.2. POSTPROCESSING OMR DATA USING SIMPLE HEURISTICS

Figure 5.6. Example for an OMR result for piano music with disagreeing key signatures at the beginning of the grand staff and incorrectly recognized time signatures. The key signatures can be corrected by choosing the one with more accidentals to replace the other. The time signatures can be corrected by finding the predominant voice duration found in the 10 bars following the time signature.

Figure 5.7. Example for an execution hint written at the bottom of a page. Execution hints in piano music usually consists of a single staff that is smaller than the regular staffs.

that are found below all other grand staffs and whose size is smaller than the size of the other grand staffs on the page. Grand staffs that are classified as execution hints by this heuristic are completely removed from the OMR output.

5.2.5 Grandstaff Indentation Detection

In sheet music books that contain several songs or movements, the beginning of a new song or movement is usually indicated by an indented grand staff such as depicted in Figure 5.8. Since the OMR software does not detect the boundaries of individual songs or movements, we use these indentations as indicators for track boundaries, see also Chapter 6.

To detect indentations, we compare the left boundary of all grand staffs found on a page.
Assuming, that the smallest left margin found on the page indicates a non-indentled grand staff and that margin is roughly the same for all non-indentled grand staffs on a page, indentations are revealed as significant deviations towards the right. For piano music, the recognition of indentations is pretty stable. However, the resulting segmentation of sheet music books into songs or movements does not always coincide with the segmentation used on audio CD collections. This matter will be further discussed in Chapter 6.

5.3 Iterative Approach for Reducing Inconsistency

For the incorporation of many of the higher-level rules of music notation and semantics, the mechanism of considering simple heuristics in isolated modules, as done in the previous subsections, might not be sufficient. The reason for that is that changes made to the OMR output data might affect many rules simultaneously. Finding the correct changes to make may require considering how these changes affect the consistency of the data with possible many different rules across many different semantic levels simultaneously.

In the following, we describe a simple iterative approach for reducing inconsistency of the OMR output data with given sets of rules. Let us use the symbol $\Omega$ to refer to a given set of symbolic score data obtained via OMR. Rules are specified and collected as elements in a so-called rule set $R$. Each rule $r \in R$ provides two basic functionalities. Firstly, given the score data $\Omega$, a rule must be able to identify a set of violations $V = V(\Omega \mid r)$. Each violation $v \in V$ is assigned a cost $c(v \mid \Omega)$, which is used as a measure for the inconsistency between the
5.3. **Iterative Approach for Reducing Inconsistency**

rule and the data $\Omega$ caused by the violation $v$. We choose the notation $c(v \mid \Omega)$ to implicate that the violation and its cost depends on the context that is given through the OMR data $\Omega$. Secondly, for each violation $v \in V$, the corresponding rule must be able to provide a set of candidate modifications $M_v$ that would eliminate or at least attenuate the violation $v$ when being applied to $\Omega$.

Given a rule set $R$ and some score data $\Omega$, the iterative algorithm makes changes to the data $\Omega$ trying to minimize the inconsistency of the data with the rule set by performing the following steps:

1. Set the iteration counter $k := 0$ and initialize the score data $\Omega^0$ after 0 iterations by setting $\Omega^0 := \Omega$.
2. Calculate the violations found in the score data $\Omega^k$ based on the rule set $R$ and store them as $V^k = V(\Omega^k \mid R) = \bigcup_{r \in R} V(\Omega^k \mid r)$.
3. Calculate the inconsistency measure $c^k = c(V^k \mid \Omega^k) = \sum_{v \in V^k} c(v \mid \Omega^k)$.
4. For each violation $v \in V^k$, calculate the set of candidate modifications $M^k_v$ and collect them in the set $M^k := \bigcup_{v \in V^k} M^k_v$.
5. For each modification $m \in M^k$, apply the modification to the data and obtain $\Omega^k_m := \text{modify}(\Omega^k, m)$.
6. For each $m \in M^k$, calculate the violations found in the modified score data $\Omega^k_m$ based on the rule set $R$ and store them as $V^k_m = V(\Omega^k_m \mid R)$.
7. For each $m \in M^k$, calculate the inconsistency measure for the modified score data $c^k_m = c(V^k_m \mid \Omega^k_m)$.
8. Find the minimum inconsistency after $k + 1$ iterations $c^{k+1} := \min_{m \in M^k} c^k_m$.
9. If there has been an improvement, i.e., $c^{k+1} < c^k$, then set $\Omega^{k+1} := \Omega^k_{m_0}$ with $m_0 = \arg \min_{m \in M^k} c^k_m$, increase the iteration counter $k := k + 1$, and continue with Step 2. Otherwise, terminate the algorithm and output $\Omega^k$.

The above algorithm iteratively checks the given data for rule violations, tries out all modifications that are proposed by the violated rules, and then chooses the modification that brings the most improvement with respect to the complete set of rules. If a modification fixes a violation for one rule but causes new violations for other rules, the improvement for fixing the first violation will be diminished or even eliminated by the cost for the newly caused violations. Only modifications that result in an overall improvement concerning all the rules included in the rule set will be performed. A big advantage of this approach is that one no longer needs to worry about how to decide which rules to consult to fix which aspects of the data. One can simply collect all rules and conventions that seem relevant in a rule set $R$ and have the
algorithm do the rest. The rules are specified independently from how the decisions about performing corrections are made. It is even possible to add rules that output violations, but that do not make suggestions for candidate modifications. Such rules may penalize certain constellations and can help to decide which modifications should be performed.

The following list gives some examples for rules that are useful for correcting OMR data for piano music. Each item in the list is written as a single sentence set in italic letters whose function is to give the reader an idea of what the rule says. Below this sentence, some comments and examples are given to further illustrate the idea of the rule. After that, a list of types of candidate modifications is given that might be proposed by the rule to eliminate or attenuate a violation. Note that the purpose of this list only is to give a rough idea about how the proposed concept works. In practice, a rule is specified as Java code that, given OMR data as an input, finds violations and creates candidate modifications that concretely specify which objects in the OMR data are to be changed what way.

(R1) *Each staff must begin with a clef.*

This rule is violated if the first (leftmost) symbol on a staff is not a clef. A violation could, for example, be cause by the OMR software misclassifying a clef as a different symbol (e.g., a note or rest) or omitting the clef completely, as depicted in the example illustration below.

![Example Illustration](image)

Proposed data modifications:

1. Insert a clef before the first symbol. (Since we consider piano music, this can be either a treble clef or a bass clef. This yields two candidate modifications.)
2. Change the type of the first symbol on the staff to a clef (again two candidate modifications).

(R2) *Usually, the left hand staff uses a bass clef and the right hand staff uses a treble clef.*

Obviously, this rule is not very strict. Even though the situation depicted in the figure below poses a violation to this rule, the two treble clefs at the beginning of the grand staff are actually correct. In fact, this rule is to be understood as a kind of penalty that can drive the algorithm to the right decision if no stronger rule dictates which modification to follow. By using a relatively small violation cost, one can make the algorithm slightly favor a treble clef in the left hand and a bass clef in the right hand in situations where, for example, a clef is to be inserted due to a violation of (R1). By having this rule not propose any candidate modifications, one can avoid this rule to cause any unwanted changes to the data, as in the example depicted...
5.3. ITERATIVE APPROACH FOR REDUCING INCONSISTENCY

5.3.1. Proposals for Reducing Inconsistency

(R3) *No chord/rest can precede or overlap with the horizontal region occupied by the heading clefs, key signatures, or time signatures at the beginning of a grand staff.*

In the left example depicted below, three notes violate this rule, because they overlap with a heading key signature of the upper staff. In fact, these notes are the result of the OMR software misclassifying accidentals of type “sharp” as note heads. A box in the right illustration indicates the horizontal region occupied by the heading clefs, key signatures, and time signatures, inside which no chords or rests are allowed by this rule.

Proposed data modifications:

1. Remove the violating chord/rest.

(R4) *Certain pairs of symbols must not overlap.*

One of the most important tasks in typesetting of Western music notation is to optimize the readability for a human reader. Since overlapping symbols can pose a significant visual obstruction, there are best practices for minimum spacings between certain types of symbols. Even though slightly touching symbols can appear in some cases, situations as depicted below would pose a clear violation of this rule.

Proposed data modifications:

1. Remove/move one of the involved symbols.

(R5) *Time signatures in left hand and right hand staff must match.*
In the example below, the left hand staff declares a 6/8 time signature, while the right hand staff declares a 4/4 time signature, which is considered a violation of this rule.

Proposed data modifications:
1. Change the left hand time signature.
2. Change the right hand time signature.

(R6) **Voice durations must match the time signature.**

In most cases, the durations of the chords and rests within each voice contained in a bar are expected to add up to the duration declared by the time signature that is valid for that bar. Exceptions to this rule are offbeat bars that can occur at the beginning and end of a musical section and voices that begin or end in the middle of a bar (see Figure 5.3). In the example depicted below, a missing beam causes the duration of the voice in first bar of the lower staff to add up to 6/4 instead of just 2/4 as suggested by the time signature.

Proposed data modifications:
1. Modify the duration of a chord/rest in the voice.
2. Insert or remove a chord/rest.
3. Modify voices/membership in voices.
4. Modify the time signature.
5. Insert/remove a time signature.

(R7) **Membership in the same raster group implies having the same onset time.**

If the onset time of chords is determined by the duration of the preceding chords and rests in the same voice, errors in the detected durations or missed out chords or rests may cause this onset time to drift apart for different voices in the same raster group. An example is depicted below. Here, at the beginning of Voice 2, a beamed group of 4 eighth notes has erroneously been recognized as 3 individual quarter notes. This causes violations for this
rule at three raster groups which are highlighted in the picture. In the first violation, the onset time of the note in Voice 1 is 1/4 but the onset time in Voice 2 is already 2/4. The error propagates to all subsequent raster groups where other voices share a member with Voice 2.

Proposed data modifications:

1. Modify the duration of a chord/rest that precedes the violation to adjust the onset time.
2. Modify voices/membership in voices.
3. Modify raster groups/membership in raster groups.

(R8) No two chords/rests of a single raster group can be member of the same voice.

The voice assignment depicted in the example below would cause two violations for this rule, because Voice 1 has two member chords/rests at the highlighted raster groups.

Proposed data modifications:

1. Modify voices/membership in voices.

(R9) Each chord/rest in a raster group should belong to one and only one voice.

A voice assignment as illustrated in the example below would cause three violations to this rule. Two highlighted chords are not assigned to any voice.
One chord is assigned to two voices simultaneously.

Proposed data modifications:

1. Modify voices/membership in voices.

(R10) **Key signatures in left hand and right hand staff must match.**

In the example below, three of four $b$s of the key signature in the right hand have not been recognized. This leads to a violation, because now the left hand staff declares a 4$b$ key signature, while the right hand staff declares a 1$b$ key signature.

Proposed data modifications:

1. Change the left hand key signature.
2. Change the right hand key signature.

(R11) **If a key signature changes at the beginning of a new line, this change must be “announced” at the end of the previous line.**

An example of such an “announced” key signature change is depicted in Figure 5.9. A case, where this rule is violated is depicted below. This violation is caused by the OMR software missing out the same amount of accidentals from the key signatures at the beginning of the second line.

Proposed data modifications:
5.3. ITERATIVE APPROACH FOR REDUCING INCONSISTENCY

1. Alter the key signature at the beginning of the new line, so that there is no longer a change in key.
2. Alter the key signature at the beginning of the previous line, so that there is no longer a change in key.
3. Insert an “announcing” key signature change at the end of the previous line.

(R12) There should be no unnecessary empty space that stretches across all staffs of a grand staff.

Empty space, as highlighted in the example below, is usually avoided in sheet music typesetting. In the example, the space is caused by a key signature change showing 4 naturals that has been missed out by the OMR software.

Proposed data modifications:

1. Insert a symbol inside the empty space. (Since there are too many choices for symbols that could be added inside empty space, this rule might best be combined with suggested modifications of other rules that propose the insertion of a particular symbol.)

Rules (R1)-(R4) can fix missing clefs and remove objects that lie in the scope of the staff header and have erroneously been classified as chords/rests such as is the case in the example shown in Figure 5.9. Using rules (R5)-(R9) one might be able to simultaneously find the most likely voice assignments, fix broken time signatures, and correct errors in chord/rest durations. Since all of these aspects are interdependent, the combined optimization of all these aspects
CHAPTER 5. FROM SHEET MUSIC TO NOTE EVENTS

can yield better results than optimizing each aspect individually. Using rules (R10)-(R12) the key signatures can be fixed even beyond the level of what the module described in Subsection 5.2.2 does. Using the combination of rules, it becomes possible to fix cases where the key signature of both the left hand and right hand staff have been recognized incorrectly. It even becomes possible to infer the insertion of key signatures that have been missed out by the OMR software by following a suggested insert operation of rule (R11).

The approach is rather naively trying each candidate modification and chooses to perform the one with the biggest improvement. This may be seen as a steepest decent method for minimizing a high-dimensional inconsistency function. As is always the case with such methods, there is the chance of getting stuck in a local minimum of the function to minimize. In our case this would mean that after the algorithm has finished, a lower inconsistency could still be achieved, but it would require performing more than one modification in a single step to reach it.

The shape of the inconsistency function depends on which rules are included in the rule set and on the tuning of the violation cost produced by these rules. Some rules are known to be very strict, e.g., (R3), so a violation of that rule should produce a relatively high cost. Other rules are simply saying that certain constellations are more likely than others, but with exceptions occurring from time to time. In such cases, the cost for a violation should be set to a lower value. While adding rules for this approach is rather simple, tuning the violation cost so that the inconsistency function indeed has its global minimum at the desired configuration of the data is a delicate task. However, by carefully tuning violation cost and adding rules for expressing even minor preferences and details of music notation, some local minima can be eliminated.

The major drawback and limitation of this approach is its runtime behavior. Naively trying out each candidate modification and recalculating the violations for each try leads to an expected computational cost that scales with the cube of both the length of the input OMR data and the number of rules, see Appendix A. Obviously, the algorithm wastes a lot of resources by recalculating the violation cost for the complete data when checking the improvement of a particular modification. This might in fact not be necessary, because a modification might only affect violations in some local neighborhood. To speed up the algorithm, one should introduce some sense of locality to avoid unnecessary calculations of violation cost at locations that have no effect on the location of change. If the local neighborhood that affects violations can be assumed to be of constant size in general, this would reduce the complexity from being cubic to being quadratic in the length of the input OMR data.

Using suitable collections of rules, many OMR errors can be detected and corrected. However, there clearly is a limitation to what can be fixed. In some cases, the OMR data might be obstructed beyond repair through postprocessing. For example, missing notes are hard to infer from rules only. Even if it can be inferred that some object occupying time in a voice needs to be inserted, it will be hard to choose between inserting rests, notes, or chords with different pitches. Some rules regarding continuation of musical patterns or melody lines suited to the underlying musical style could be used to assess which options are more likely than others. However, in extreme cases, one would end up having to guess complete sections of a musical composition.
Instead of inferring corrections based on erroneous OMR results, a much more powerful approach would be to incorporate the consideration of high-level rules such as those listed above directly into the OMR process. This way, the rules could be used to increase the certainty of decisions on the recognition and classification of symbols and semantics before erroneous decisions are made and important information is lost. The main problem with putting this idea into practice is that rules are spread across very different semantic levels. For example, some rules might work on the level of pixels, others might work on the level of shapes and symbols, and again others might work on note events and voices. However, without having already derived shapes and symbols, one cannot apply the rules that work on this level, and without having already derived note events and voices, one cannot apply the rules that work on that level, too. Clearly, all of these levels are strongly interconnected, and to infer decisions with high certainty, all of these levels have to be taken into account simultaneously.

5.4 Detecting Repeats and Jumps

At this point, the symbolic score data obtained from the OMR results do not contain any information other than the plain notation bar sequence $\nu$ about the sequence in which the bars are to be played. However, for converting the symbolic score data to note events, we want to use the sequence of bars as it is suggested by the score for a performance, i.e., we want to find out the default bar sequence $\delta$. Differences between the default bar sequence $\delta$ and the notation bar sequence $\nu$ are caused by repeats and jumps. Since repeats can be seen as a special case of jumps, we will only speak of jumps in the following. In general, due to OMR errors, we will not be able to always find the correct default bar sequence from the sheet music. Therefore, the goal pursued in this section is to derive an estimate $\delta^*$ that is as close to $\delta$ as possible. The general approach to calculate $\delta^*$ is to start with the first bar of the track, i.e., the first bar in $\nu$ and then follow $\nu$ until some symbol or text indicates that a jump to a different bar than the successor in $\nu$ is to be performed. We will discuss the possible kinds of jumps in the following. Note that in order to represent $\delta^*$, it is sufficient to store a list of jumps that is to be performed with respect to the notation bar sequence $\nu$. Besides these jumps, $\delta^*$ always follows the notation bar sequence $\nu$. A jump is specified by a jump source $\nu_s \in \nu$ and a jump target $\nu_t \in \nu$ with $t \neq s + 1$.

The most simple type of jump is a repeat indicated by repeat signs. An example is illustrated in Figure 5.10(a). When reaching the bar with closing repeat signs, one has to jump back in $\nu$ until either to the closest bar with opening repeat signs or the beginning of the track or a titled subsection of the track is reached, see also Figure 5.10(b). A regular repeat sign indicates a single repeat only and, therefore, is ignored the second time it is reached. The recognition of repeat signs by the OMR software is quite reliable. In a test dataset of 5 sheet music books consisting of piano sonatas by Beethoven, Mozart, and Schubert, as well as string quartets by Mozart, 564 of 570 repeat signs were recognized correctly, which is a rate of about 98.95%. The cases, where the repeat signs are not recognized by the OMR software, are usually either caused by the repeat signs being surrounded by parantheses or the dots of the repeat signs touching nearby staff lines, see Figure 5.11.
Figure 5.10. Illustration of several types of repeats and jumps in sheet music. For each example, the resulting default bar sequence $\delta$ is indicated below the staff by a folded arrow with solid and dashed sections. The solid sections indicate bars that are to be played and the dashed sections indicate jumps. (a) Simple repeat indicated by opening and closing repeat signs, (b) example for a titled subsection acting as a jump target for a simple repeat, (c) simple repeat with two alternative endings indicated by number brackets, (d) a repeat with 4 runs and 3 alternative endings, (e) a da capo jump with a “fine” ending, (f) a dalsegno jump with a “fine” ending, (g) a da capo jump and a coda jump, (h) example for a textual jump directive that references the titles of subsections.
5.4. DETECTING REPEATS AND JUMPS

Figure 5.11. Examples of double bar lines with repeat signs that are mistakenly recognized by the OMR software as plain double bar lines without repeat signs. In the case of type (a), the recognition error is probably caused by one of the dots being slightly misaligned and touching the staff line below. In the cases of type (b), the recognition of the repeat signs probably fails because of the additional brackets.

Our algorithm for finding jumps induced by repeat signs works as follows. Stepping through the notation bar sequence $\nu$, we always keep track of the most recent candidate jump target $\nu_t$. A bar $\nu_t$ can serve as a candidate jump target if it is the first bar of the track, the first bar of a new titled subsection (e.g., “Trio” in a track that consists of a section titled “Scherzo” followed by a section titled “Trio”), or a bar with opening repeat signs. Once a closing repeat sign is reached at bar $\nu_s$, we add a jump with source bar $\nu_s$ and target bar $\nu_t$ to the list of jumps. Unfortunately, titled subsections are hard to recognize from the OMR results because the recognition of text that might serve as title is very unreliable and often it is not clear whether a such a text is to be interpreted as a title, a tempo directive, or even both. An example for a subsection title can be seen in Figure 5.9. Therefore, instead of relying on the recognition of titled subsections, we make the assumption that the jump target of a repeat never lies before the closing repeat sign of a previous repeat. Using this assumption, we simply add all bars that immediately follow closing repeat signs in $\nu$ as candidate jump targets. In our tests, this method did not only cover the cases of titled subsections, but was also able to repair cases where opening repeat signs had not been recognized. In total, 350 of 353 jumps that are induced by repeat signs were recognized correctly on the test dataset, which is a rate of 99.15%.

A second type of jump is induced by repeats with alternative endings. Such alternative endings are usually indicated by number brackets above the bars of the alternative sections, see Figure 5.10(c). In most cases, there is a single repeat with two alternatives where the first alternative (indicated by the number 1) is to be played in the first run and the second alternative (indicated by the number 2) is to be played in the second run. This manifests as one additional jump besides the one induced by the repeat signs that is performed to skip the first alternative in the second run of the repeated section. When reaching the bar $\nu_s$ preceding the first alternative in the second run, a jump has to be performed to the bar $\nu_t$ that is the first bar of the second alternative. Despite the cases of having two alternatives and a single repeat, there can be cases with more than two alternatives and more than a single repeat, see Figure 5.10(d). In the test dataset, a total of 97 repeats with alternative endings are found. Unfortunately, the OMR software used in this project did not recognize the number brackets that indicate alternative endings at all. Therefore, this type of jump cannot be recognized and taken into account in our calculation of $\delta^*$. In Chapter 7, however, we briefly outline a strategy for trying to capture this type of jump through content-based
As a third category, we consider jumps that appear in the context of so-called da capos and dal segnos. Da capo is Italian for “from the beginning” and instructs the player to jump back to the beginning of the track to play another run, which we call da capo run. Usually, in the da capo run, repeat signs are ignored and for repeats with alternative endings, the last alternative is chosen, see Figure 5.10(e). Dal segno is Italian for “from the sign/marker”. This directive appears in combination with special marker symbols such as ☞ or ☞ that specify the bar that acts as the jump target in the dal segno jump, see Figure 5.10(f). The ☞ symbol, furthermore, is often used to indicate a jump from a da capo run to a so-called “Coda” section that acts as the final section of the track, see 5.10(g). Da capo and dal segno jumps are often indicated by textual directives, jump marker symbols, or a combination of the two. Textual jump directives are usually written below the intended jump source bar, which is expected to end with some sort of double bar line indicating the end of the piece or a musical subsection. The directives for da capo and dal segno jumps can be written in many ways. Often, abbreviations are used such as “D.C.” for da capos and “D.S.” for dal segnos. Da capo and dal segno jumps can be extended by additional parameters. Using the word fine, which is Italian for “end”, a piece can be declared to end at a bar that is different from the last entry in the notation bar sequence υ. An example usage of this keyword can be found in Figure 5.9 and in Figure 5.10(e)-(f). In such a case, the target end bar is marked by the keyword “fine” and the textual jump directive might read, for example, “da capo al fine”. However, the “fine” marker can also be used to indicate the end of the piece without being explicitly mentioned in the textual jump directive. Textual jump directives can use subsection titles as parameters. For example a directive “Menuetto D.C.” indicates that the subsection titled “Menuetto” is to be repeated from the beginning. A single textual directive can, furthermore, imply more than a single jump. For example, the directive “Allegretto D.C. e poi la Coda” implies, first, a jump back to the beginning of the subsection titled “Allegretto” and, second, a jump from the end of the Scherzo to the beginning of the subsection titled “Coda”, see Figure 5.10(h). Note that some of the indicators for jumps are bound to certain conditions. For example, if a bar contains both a repeat sign and a dal segno directive, as in Figure 5.10(f), the dal segno directive is only to be followed in the second run of the repeat. As another example, the coda jump illustrated in Figure 5.10(g) is only to be followed in the da capo run.

To detect da capo and dal segno jumps from the OMR results, the detected text elements are searched for keywords such as “da capo”, “d.c.”, “fine”, and “coda”. On the test dataset, 20 of 29 da capo jumps and 11 of 15 cases where the track ended at a “fine” keyword were correctly recognized. In the remaining cases, either the OMR software failed to recognize the existence of text at all or the keyword was obstructed by errors in the recognized text. Experiments on the test data showed that using the edit distance for a fuzzy comparison to account for possible inaccuracies in the text recognition of the OMR software did not significantly improve the results without causing false positives, i.e., song lyrics or other text erroneously being classified as one of the keywords. The detected da capo jumps and “fine” endings are taken into account in our estimated default bar sequence δ*. Since segno and coda marker symbols are not detected by the OMR software, the corresponding jumps are not taken into consideration in δ*.

As a final category of jumps, we consider cases that do not fall in any of the three categories
5.5. **ESTIMATION OF ONSET TIMES AND TEMPO**

In this section, we focus on how to extract temporal information such as onset times and durations of note events from the OMR results. Since in common Western music notation, the score is usually divided into bars that are played consecutively, it is sufficient to determine the onset times and durations relative to the beginning of each bar individually. The absolute onset times can then be derived by concatenating the bars according to a suitable bar sequence, such as $\delta^*$ derived in the previous section. Onset times and durations are usually measured in musical units such as full notes, half notes, and quarter notes. The musical duration of notes and rests is specified by their shape and by the attachment of beams or flags to note stems. How these musical units translate to physical units such as seconds is discussed above. This category covers exceptional cases and free-form textual directives that are very hard to interpret computationally. A nice example of such a case is depicted in Figure 4.14, where a jump marker is used together with a textual description in four different languages that points out under what condition a jump to a matching marker found elsewhere in the score is to be performed. A somewhat less extreme example that, however, still falls outside the expected conventions is the use of the textual directive “Attacca il Menuetto subito” to indicate a da capo jump without using any of the corresponding keywords, as seen in Figure 5.12. Furthermore, fermata symbols above and below a double bar line are used instead of the “fine” keyword to mark the end of the track after the da capo run. As a last example, consider the situation depicted in Figure 5.13, where a pair of $\|$ markers is used to indicate that a section should be repeated multiple times (one time for each verse of the lyrics). Furthermore, a fermata in the last bar is used to imply the piece is supposed to end at that position within the bar after the last verse is played. Cases of this category are not taken into consideration in our automatic computation of $\delta^*$. 

![Example Figure](image-url)
determined by what is called the \textit{tempo}. The tempo is often specified as a numerical value called \textit{beats per minute} (BPM). A BPM value says how many subsequent quarter notes would fit in one minute of time. Given the tempo $T$ in BPM, the duration $s_{\text{quarter}}$ of a quarter note in seconds can, therefore, be calculated as

$$s_{\text{quarter}} = \frac{60}{T}.$$ 

Within a bar of music, the onset time in musical units of a note is usually determined by the duration of the preceding notes in the same voice. Unfortunately, this approach is not very robust in our scenario due to two reasons. Firstly, the recognition of beams and flags and, therefore, the recognition of note durations is not very reliable. Secondly, notes that have completely been missed out fail to contribute any duration at all. Therefore, simply adding up note durations for each voice individually to determine onset times could lead to voices drifting apart temporally. However, the aspect of simultaneity is a very important characteristic of the music. We, therefore, propose a method that tolerates some inaccuracies in relative onset times in order to robustly preserve the simultaneity of note onsets. Instead of accumulating voice durations, onset times of notes in a bar are determined by the relative horizontal position of their respective raster group and are measured not in musical units but as fractions of the bar width. Assuming that the duration of the bar is known, these fractional units can easily be converted to musical or physical units. Figure 5.14 illustrates how the fractional units are determined for an example bar of piano music. Let us assume that, in general, we have $R$ raster groups ($\rho_1, \ldots, \rho_R$) which are sorted by horizontal position and with their horizontal distances from the left bar line being denoted by $d_1 < \ldots < d_R$. Furthermore, we denote the horizontal distance of the right bar line from the left bar line as $d_{\text{bar}}$. Even though the leftmost raster group $\rho_1$ is usually spaced apart from the left bar line for layout reasons, the first raster group is understood to mark the temporal beginning of the bar. Since the spacing between the leftmost raster group and the left bar line does not correspond to any temporal duration, we define the width $w$ of the bar as the horizontal distance between the leftmost raster group and the right bar line:

$$w := d_{\text{bar}} - d_1.$$
Then, each raster group in the bar is assigned the fractional onset time
\[ f_r = \frac{d_r - d_1}{w}, \quad r \in [1 : R]. \]

If the duration of the bar in seconds \( s_{\text{bar}} \) is known, the onset times \( f_r \) in fractional units are simply converted to onset times \( t_r \) in seconds through
\[ t_r = f_r \cdot s_{\text{bar}}. \]

In most cases, the horizontal positions and distances are roughly proportional to the musical onsets and durations within a bar. However, requirements of layout, especially rule (R4) but also rule (R12), are given priority over preserving the proportionality. This leads to inaccuracies in relative onset times. For example, in a run of short notes that are written with little horizontal distance from each other, if one of the note has an accidental, some extra space is added before that note in the layout to keep the accidental from overlapping with the previous note. This situation is depicted in Figure 5.15.

As stated earlier, the fractional units can be converted to musical or physical units when the duration of the bar in musical or physical units is known. The duration of a bar in musical units is determined by the time signature. For example, a time signature \( \frac{4}{4} \) implies a bar duration of four quarter notes. We denote the musical duration of a bar in quarter notes by the symbol \( q_{\text{bar}} \). For a time signature with numerator \( n \) and denominator \( k \) it is calculated as
\[ q_{\text{bar}} = \frac{n}{k} \cdot 4. \]

The bar duration can also be determined by the duration of the voices it contains, and, in theory, this value should agree with the value obtained from the time signature. In practice...
CHAPTER 5. FROM SHEET MUSIC TO NOTE EVENTS

Figure 5.15. Notes that are usually written with little horizontal distance are spaced apart from the previous note more widely to avoid accidentals from overlapping with neighboring notes.

of our scenario, however, the recognition of voices and their durations is not very reliable, and voices in bars with multiple voices can disagree about the correct duration. We, therefore, choose to use the time signature as a criterion. The voice durations may have been already taken into account for validating or correcting the time signatures through OMR postprocessing.

To convert musical onset times and durations to physical onset times and durations, we need to determine the tempo in BPM. In practice, the scores in the repertoire of the project most often do not specify a tempo in BPM directly. Instead they use text labels that, by convention, are mapped to rough tempo classes. The vocabulary used for tempo labels depends on the individual piece and edition. Many scores use a mostly standardized vocabulary of Italian labels such as Allegro, Andante, or Adagio. However, in general, any language could be used and the use of conventions is optional. For example, one may encounter German tempo labels such Geschwind, Mäßig, or Etwas Langsam, which, without additional knowledge about context, composer, and period, will give only a very rough idea of what tempo might be appropriate. Considering this uncertainty in the vocabulary and the possible coarseness of its meaning, it is no surprise that the OMR software does not output any information about tempo at all. Textual tempo directives, if recognized at all, are output as general purpose text elements without further classification. However, for our scenario, it would be beneficial to have at least a rough estimate of the tempo, because large discrepancies between the estimated tempo of the score data and the actual tempo of the audio recording lead to additional difficulties in the content-based comparison. Unfortunately, the recognition of text elements in the OMR software is too unreliable to expect to properly solve this issue by classifying the recognized text elements to find out the tempo directives ourselves. Furthermore, due to the freedom of vocabulary and the context-dependency of its meaning, the translation from textual tempo directives to tempo estimates in BPM is a non-trivial task, which, in the context of this thesis, is seen as future work. For our scenario, we simply use a fixed tempo estimation for all of the OMR data and move the problem of dealing with the tempo differences to the content-based comparison techniques.

Experiments with the Beethoven piano sonatas, for which the recorded performances use tempos ranging from about 25 to 300 BPM, have shown that even when assuming that the performance sequence $\pi$ is known the global synchronization using DTW can break down if the differences in tempo become too large. This behavior was observed for a passage part of which is depicted in Figure 5.16. The tempo of the recorded interpretation by Alfred Brendel uses a tempo of approximately 25 BPM for this section. Using a fixed tempo estimation for
5.5. ESTIMATION OF ONSET TIMES AND TEMPO

the OMR data of 100 BPM, the difference in tempo is a factor of 4. To deal with this issue, we introduce an adaptive tempo estimation strategy to replace the fixed tempo estimation. In the adaptive estimation, the tempo is estimated individually for each bar. The estimated tempo depends on how many raster groups are found inside a bar. Bars with many raster groups usually indicate the presence of notes with a short duration in musical units, e.g., sixteenths or shorter. Manual inspection revealed that the performances that use the extremely low tempos, like in the example case, correspond to pieces that are written using these rather short note duration. Therefore, the adaptive tempo estimation is designed to deliver a lower tempo for bars that contain many short notes in contrast to bars that contain only few notes with longer duration.

The adaptive estimation of the tempo for a bar of sheet music is performed in three steps. In a first step, the duration of the bar \( s_{\text{bar}} \) in seconds is calculated as a base duration \( s_{\text{base}} \) in seconds multiplied by the amount of raster groups \( R \), i.e., \( s_{\text{bar}} = R \cdot s_{\text{base}} \). In our scenario, we use a base duration \( s_{\text{base}} = 0.2 \) seconds. Then, in a second step, the bar duration \( s_{\text{bar}} \) is converted to a tempo \( T \) in BPM by means of the musical bar duration in quarter notes \( q_{\text{bar}} \) specified by the time signature

\[
T = 60 \cdot \frac{q_{\text{bar}}}{s_{\text{bar}}} = 60 \cdot \frac{q_{\text{bar}}}{R \cdot s_{\text{base}}}.
\]

In a third step, the tempo is limited to a maximum BPM value, in our case 100 BPM, to avoid the tempo being estimated too high for bars that only contain very few raster groups, e.g., bars with only a single full note for each voice. In the example case depicted in Figure 5.16, the time signature is written as 6/16 and there are 12 raster groups per bar. This leads to a bar duration in seconds of \( s_{\text{bar}} = R \cdot s_{\text{base}} = 12 \cdot 0.2 = 2.4 \). Taking the time signature into account, the duration of a single quarter notes amounts to 1.6 seconds which is equivalent to a tempo estimation of 37.5 BPM.

If the performance sequence \( \pi = (\pi_1, \ldots, \pi_K) \) is known in advance, a mean bar duration \( \overline{s}_{\text{bar}} \) in seconds can be estimated from the duration of a corresponding audio recording \( s_{\text{audio}} \) in seconds and the total number of bars \( K \) by means of the formula

\[
\overline{s}_{\text{bar}} = \frac{s_{\text{audio}}}{K}.
\]
When additionally knowing the time signature for each bar, the duration of the score track in quarter notes \( q_{\text{score}} \) can be derived as \( q_{\text{score}} = \sum_{k=1}^{K} q_{\text{bar}}(\pi_k) \) and used to compute a mean tempo \( T \) of the score track by means of

\[
T = 60 \cdot \frac{q_{\text{score}}}{s_{\text{audio}}}
\]

Since \( \pi \) is usually not known, \( \delta \) or \( \delta^* \) can be used instead. Even though this may lead to inaccurate values for \( q_{\text{score}} \), the error can usually be expected to be small and is very likely to be smaller than a factor of 2, which for our application is considered not critical. We will make use this strategy for the tempo estimation in Chapter 7. What remains problematic are score tracks for which the tempo varies by large factors within the track. This is, for example, the case in Beethoven’s Piano Sonata No. 26, Opus 81a, “Les adieux”, where the tempo starts with Adagio and then changes to Allegro. Clearly, for such cases, the estimation of a single average tempo for a single score track cannot be accurate.

### 5.6 Deriving Pitches

Our method for determining pitches of note events derived from the symbolic score data is to follow the basic rules of Western music notation. These basic rules state that the pitch of a note event is determined by its vertical position on the staff, the currently valid clef and key signature, as well as accidentals occurring in the bar where the note event starts. For piano music, these rules are usually sufficient to derive accurate pitch information. For other types of music, in particular orchestral music, additional difficulties arise, some of which are discussed in Section 5.7. Furthermore, music notation can comprise many exceptions and conventions that make determining accurate pitch information very hard, see [15]. Throughout this section, we assume that for the type of symbolic score data given, the basic rules stated above are sufficient.

Our goal is to determine the pitches of note event within each bar in form of MIDI pitch numbers \( p \in [0 : 127] \). A list of MIDI pitch numbers and their corresponding frequencies and musical names can be found in [147]. To determine the pitches of note events within each bar, we step through the bars of the score in order of the given bar sequence \( \delta^* \). For each staff, we scan the corresponding symbols in left-to-right order and keep track of the symbols that affect the pitch, i.e., clefs, key signatures, and accidentals. Clefs stay valid until the end of the track unless they are replaced by another clef in the same staff before the end of the track is reached. Key signatures stay valid until the end of the line is reached unless they are replaced by another key signature before that. Accidentals directly attached to note heads are usually valid until the end of the bar. When a note symbol is found, its pitch is determined using the clef, key signature, and accidentals currently valid for this staff. To this end, the following steps are performed.

1. The vertical position \( v \in \mathbb{Z} \) of the note head is obtained from the OMR data. Here, the value \( v = 0 \) means that the note head is on the middle line of the 5-line staff, see Figure 5.17. The value of \( v \) increases with lower vertical positions, e.g., \( v = 1 \) means that the
5.6. DERIVING PITCHES

Figure 5.17. Top Left: Illustration of note head positions and their corresponding vertical position \( v \in \mathbb{Z} \). Bottom Left: Examples of clefs and their corresponding reference pitch class offset \( r \). Right: Two examples for determining the MIDI pitch \( p \) of a note given its vertical position, a clef, a key signature and possibly an accidental.

Note head is between the third and the fourth line from the top, \( v = 2 \) means that it is on the fourth line and so on. If the note head is above the middle line, \( v \) becomes negative.

2. Using the currently valid clef, a pitch class \( q \in Q = \{ C, D, E, F, G, A, B \} \) and an octave number \( o \in \mathbb{Z} \) is determined based on the vertical position \( v \). To this end, for each clef, a characterizing reference pitch class offset \( r \in \mathbb{Z} \) must be known that determines how a vertical note position \( v \) translates into pitch class and octave. Then, the pitch class and octave number are computed as:

\[
q = f((r - v) \mod 7) \quad (5.1)
\]

\[
o = \left\lfloor \frac{r - v}{7} \right\rfloor, \quad (5.2)
\]

with \( f \) being the canonical bijective mapping from the numbers \([0 : 6]\) to the pitch classes \( f : 0 \mapsto C, 1 \mapsto D, 2 \mapsto E, 3 \mapsto F, 4 \mapsto G, 5 \mapsto A, 6 \mapsto B \). A list of common clefs along with their reference pitch class offsets \( r \) is depicted in Figure 5.17.

3. From the pitch class \( q \) and the octave number \( o \) a base MIDI pitch number \( p' \) is derived through

\[
p' = g(q) + 12 \cdot o \quad (5.3)
\]

with \( g \) being the following mapping from the pitch classes to corresponding base MIDI pitch numbers \( g : C \mapsto 60, D \mapsto 62, E \mapsto 64, F \mapsto 65, G \mapsto 67, A \mapsto 69, B \mapsto 71 \).

4. Each pitch class \( q \in Q \) can be modified by both the accidentals of the key signature and individual accidentals that are attached to note heads. Each accidental affects one particular pitch class. Which pitch class that is can be determined by their vertical position in the same way as for note heads. If both the key signature and an individual accidental affect the same pitch class, the individual accidental overrules the accidental of the key signature. To take the key signatures and accidentals into account in our calculation of pitches for note events, we keep track of the currently valid pitch modifier.
$m_q \in [-2 : 2]$ for each pitch class $q \in Q$ while stepping through the symbols in a strictly left-to-right order. The list below shows the commonly used accidental symbols and their corresponding pitch modifiers $m_q$:

- $\#$ (sharp) $\mapsto +1$
- $b$ (flat) $\mapsto -1$
- $\natural$ (natural) $\mapsto 0$
- $\sharp$ (double sharp) $\mapsto +2$
- $\boldsymbol{b}$ (double flat) $\mapsto -2$.

To determine the MIDI pitch $p$ of a note event, the pitch modifier $m_q$ is added to the base MIDI pitch number $p'$

$$p = p' + m_q. \quad (5.4)$$

Two examples for determining the MIDI pitch $p$ of notes based on their vertical position, the clef, the key signature and possibly accidentals are illustrated in Figure 5.17.

### 5.7 Orchestral Scores

Compared to scores for piano music, OMR results of orchestral scores raise additional difficulties for extracting accurate pitches and onset times for generating note events. An example page of orchestral music is depicted in Figure 5.18. One of the difficulties in orchestral music is the possible presence of transposing instruments. These are instruments for which the nominal pitch that is written in the score differs from the pitch that sounds when the instrument is played. Common examples for transposing instruments are clarinets, horns, and trumpets.

Usually, a transposing instrument is declared to be written in a certain key. The declared key specifies the pitch that sounds when the pitch $C$ is written. From this information, one can determine the amount of semi-tones by which the written pitch must be shifted to match the sounding pitch. For example, for a clarinet written in $B\flat$, the written pitch $C$ sounds as a $B\flat$, which is two semitones below the $C$. This means that, as long as the transposition does not change, all pitches for that clarinet sound two semitones lower than they are written.

Throughout this thesis, we specify transpositions by giving the amount of semitones that need to be added to the written pitch in order to get the sounding pitch. For example, the clarinet written in $B\flat$ would have a transposition of $-2$.

The declaration of the key for transposed instruments specifies the transposition only up to the ambiguity of octave shifts. For example, instead of a transposition of $-2$, an instrument written in $B\flat$ could as well have a transposition of $+10$, or $-14$, or $-2 + 12k$, for $k \in \mathbb{Z}$, in general. The particular octave shift $k$ depends on the particular instrument class and is assumed to be known by the reader of the score. To our application of comparing note events to audio recordings, the octave shift is not critical, because the chroma features used for the comparison discard this information anyway. However, the non-octave part of the
transposition is critical to our application, because using systematically wrong pitch classes will systematically compromise the similarity between the chroma features created from the note events and the chroma features extracted from the audio recordings.

In addition to the issue raised by transposing instruments, orchestral scores often suffer more strongly from recognition errors of symbols such as key signatures and clefs than piano music. Reasons for that might be that printed staff sizes of orchestral scores tend to be smaller, which causes the degrading effect of printing resolution and artifacts to be stronger. Another reason could be different contrast and thresholding techniques in the conversion to black and white. Certainly, these effects depend on the printing quality of the particular score. A side-by-side comparison of image quality for an extract of piano music with higher quality and an extract of orchestral music with lower quality is depicted in Figure 5.19. Note that the issues caused by printing quality cannot be solved by simply increasing the scanning resolution. Another problem of OMR that becomes more relevant with orchestral scores is the grouping of staffs to grand staffs. While in piano music, each grand staff always consists of two staffs, in orchestral scores, grand staffs are usually far more complex. Depending on the kind of orchestration, the amount of staffs in a grand staff can easily exceed 20 or even pass 30. Furthermore, the selection of instruments that are included in a grand staff can change with each new grand staff. Often, the staffs for instruments that do not play any notes throughout a grand staff are simply omitted to save space and to avoid unnecessarily printing staffs that contain nothing but rests. The elevated complexity of grand staffs in orchestral scores leads to OMR errors in the grouping of staffs to grand staffs becoming a relevant issue for our application. Errors in the grouping of staffs usually manifest as grand staffs erroneously being split horizontally into two or more smaller grand staffs, see also Figure 5.18. The effect on the resulting OMR data is that in place of the subsequence of the notation bar sequence $\nu$ that would resemble the original grand staff, we obtain multiple repeats of this subsequence with each repeat being played by a different subset of instruments. Not only does this lead to compromised note events for this subsequence, but it would also be very confusing for users to look at in the application of highlighting recognized bars in the sheet music.

To encounter the challenges posed by orchestral scores through postprocessing, we need to identify rules that can help us to detect and to fix errors based on the OMR results. In fact, there are some rules which make this task seem possible. Here are some examples:

(OR1) *Certain combinations of instrument groups and transpositions are more common than others.*

For example, french horns always use a transposition of $-7$ (F), clarinets usually use $-2$ ($B\flat$), $-3$ (A) or $+3$ ($E\flat$), and string instruments such as violins and celli use a transposition of 0 (C). More information on which instruments commonly use which transpositions can be found in the literature such as [69].

(OR2) *The transposition key is usually indicated as part of the staff labels that are attached to the first grand staff of each score track.*

Some examples found on the page depicted in Figure 5.18 are “2 Klarinetten in B”, “Baßklarinette in A”, “4 Hörner in F”, and “2 Trompeten in B”.
Figure 5.18. Example page of an orchestral score showing the beginning of the Breitkopf & Härtel edition of Liszt’s Dante Symphony. This score uses special staffs with only one staff line for the percussion instruments. A special problem for the OMR software is posed by the text interrupting the vertical line that marks the left boundary of the grand staff. In the OMR recognition results, this grand staff is erroneously split into two consecutive grand staffs with the first one containing all staffs from the “Kleine Flöte” to the “Pauken in F. C.” and the second one containing all staffs from the “Harfe” to the “Kontrabässe”. The one-line staffs for the percussive instruments are not recognized by the OMR software.
5.7. ORCHESTRAL SCORES

Figure 5.19. Side-by-side comparison of image quality for an extract of piano music with higher quality (top) and an extract of orchestral music with lower quality (bottom).

(OR3) *The transposition key of an instrument group is usually constant throughout a score track.*

Even though this might hold in the majority of cases, exceptions to this rule are quite common. An example of changing transpositions is depicted in Figure 5.20.

(OR4) *The first grand staff of each score track includes staff labels declaring which staffs are to be played by which instrument groups.*

An example for this is shown in Figure 5.18.

(OR5) *Even though staffs for certain instrument groups can be omitted, the initial order of the instrument groups does never change throughout a score track.*

This is a very powerful rule, because, if the correct instrument groups are known for only some of the staffs in a grand staff, it allows us to infer or at least narrow down the instrument groups of the remaining staffs.

(OR6) *Differences in transposition can manifest in differences in key signatures between instrument groups.*
For example, having a key signature with 2 sharps in staff A and a key signature with 3 sharps in staff B might indicate, that the transposition used in staff B is a perfect fifth lower (−7) than the one in staff A (see the top two staffs in Figure 5.21). However, this rule has to be used with caution, because for some instruments, key signatures might be omitted completely in favor of using accidentals directly with each note (as in the sixth line in Figure 5.21).

(OR7) Certain instrument groups only use a particular clefs.

For example, violins always use a treble clef and piano only uses treble or bass clefs.

(OR8) Each instrument group has a known limited pitch range.

For some instruments the pitch range is limited in both directions, e.g., on a standard 88-key piano, the lowest pitch is A'' (also: A0, MIDI number 21) and the highest pitch is c''''' (C8, MIDI number 108). For many other instruments, only a lower pitch boundary can be properly specified. For example, the lowest pitch of a violin is g' (also: G3, MIDI number 55).

(OR9) If the selection of instrument groups in a grand staff is different from the previous grand staff, the new selection is indicated by staff labels attached to the new grand staff.

An example for this can be seen in Figure 5.21. Here, the staff labels are actually abbreviations of the instrument names.

The above rules show that transposition, key signatures, and clefs are all connected through the mapping between staffs and instrument groups. Even though these rules suggest the possibility of finding this mapping automatically in postprocessing of the OMR data, the chance of succeeding in this strongly depends on the completeness and reliability of the OMR output. In particular, the application of the above rules relies heavily on the accurate recognition of staff labels. Unfortunately, the OMR software used in this project is very unreliable regarding the recognition of staff labels. Labels that are printed on the left side of staffs are usually not recognized at all, and abbreviated staff labels that are found above or between staffs are often missed out or very low in accuracy of the recognized text. Since the recognition of key signatures and clefs is also not very reliable, either, rules (OR6) and (OR7) cannot help much to improve the situation. Furthermore, there can be exceptions to many of these rules. For example, regarding (OR2), for certain instrument groups no transposition key is written and a standard transposition for that instrument group is assumed. Regarding (OR3), there are cases where the transposition does change for certain instrument groups within a track, e.g., in the Breitkopf & Haertel edition of Liszt’s Faust Symphony, see Figure 5.20 The relationship between transposition and differences in key signature is not very strict. Instead of modifying key signatures according to the transposition, composers or editors can choose to completely omit key signatures for some instrument groups and use individual accidentals for each note instead. Furthermore, certain instruments, e.g., timpani have special conventions regarding pitch and transposition. Exception to rule (OR9) can also occur in practice. In such cases, the selection and order of instruments groups included in a grand staff is assumed to be implicitly understood by means of the other rules and through similarity to previous types of grand staffs.
Figure 5.20. Example for changes in transposition that take place within a score track. The changes are indicated by textual instruction starting with “muta in”, which means “change to”. In the example, three changes are indicated. In the third staff from the top, the clarinets that originally were written in C change their transposition key to A. In the 5-th and 6-th staff, the horns change from F to E. Such changes are usually achieved by changing either a crook or the complete instrument.

The above considerations lead to the conclusion that the recognition of transposing instruments and the correction of key signatures, clefs, and grand staffs, in our scenario, cannot be achieved reliably through postprocessing only. A successful approach to this task will require incorporating more advanced OMR software or combining current OMR results with further analyses of the original image data. This leads to interesting research topics, which we get back to in our discussion of future work in Chapter 9. Instead of trying to solve all of these issues fully automatically, we pursue a pragmatic method for handling the issues by a computer-assisted manual correction of the most critical errors. Using this method, we enable the integration of orchestral scores into the workflow of the project, and we lay a foundation for automatic approaches to replace the manual steps in the future. The experience and ground truth data that is collected with our pragmatic approach will prove useful in the development of automatic approaches. In the above discussion of properties of orchestral scores, we have identified that important characteristics of the staffs on each page are the grouping to grand staffs, transposition, as well as the clefs and key signatures found at the beginning of each staff. For each page, we denote these characteristics as the staff signature. An example for staff signature information can be found in Figure 5.21.

In our pragmatic approach, staff signature information is extracted automatically and saved to a plain text file for each page of the score. Each staff found on the page is represented by a single line of information in the text file. Grand staffs are separated by an empty line. Each line that represents a staff comprises columns for transposition, clef, and key signature. Optionally, additional information can be given, e.g., a flag indicating that two staffs act as a single coordinate system such as for piano or harp, an identifier for the instrument group to which the staff belongs, or information about brackets grouping certain instrument groups at the left border of grand staffs in orchestral music. The automatically generated files can be corrected manually by comparison with the original sheet music image. After that, the corrected staff signatures can be reimported to the OMR data, and any changes made, such as merging erroneously split grand staffs or adding missing transposition information are applied automatically. Unfortunately, the manual correction of the staff signatures still take a lot of effort. The correction of score books of about 150 pages took several hours of manual
CHAPTER 5. FROM SHEET MUSIC TO NOTE EVENTS

Clef | Key Signature | Transposition
---|---|---
treble | +2 | 0
| +3 | -1
| -1 | -3
| +4 | -2
| 0 | -7
| +2 | 0
| +2 | 0
| +2 | 0
| +2 | 0

Figure 5.21. Staff signature annotation for an example grand staff taken from a score of the “Symphony to Dante’s Divina Commedia S109 - Inferno” by Franz Liszt. Positive key signature values count the number of sharps, negative values count the number of flats. Transposition values are specified as the amount of semitones the pitch has to be modified with to sound correctly.

work. With the use of manually corrected staff signatures, the accuracy of the pitches of the note events extracted from the OMR data is expected to be sufficient for our scenario. The remaining inaccuracy will mostly be caused by individual accidentals being missed out, which is not expected to affect enough note events to be considered critical.

Besides the issues that can be fixed by correcting staff signatures, orchestral scores can include many other kinds of problematic behaviour that require special attention. For example, 1-line percussive staffs like the ones found on the example in Figure 5.18 are not recognized by the OMR software used in this project. Another example are in-line execution hints and alternatives that appear as additional staffs inside a grand staff directly above the staff they affect. Such in-line execution hints are recognized by the OMR software as regular staffs that are filled by empty bars to span the whole width. Fortunately, both of these issues are not critical to our scenario. The missing events of percussive instruments are not important to our chroma-based comparison of content, and the additional events introduced by in-line execution hints and alternatives are usually very similar to the regular content, so they have no negative effect on the chroma-based comparison either.