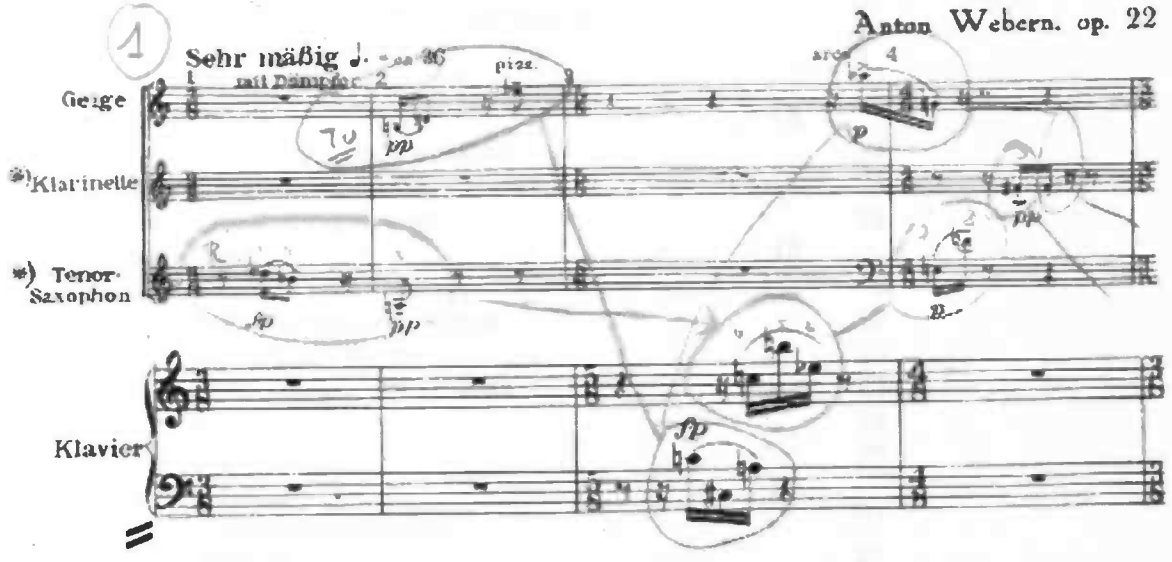
Grouping in *Language and Music*, two faces of the same problem?



Fragment from *Quartett*, Op. 22 by Anton Webern with comments from René Leibowitz
from: www.muspe.unibo.it

An Optimality Theory – based approach to musical problems in grouping structure as a case-study of the similarities between music and language and their theoretical approach

Master Thesis, 2002 – 2003
Sybrand van der Werf, stud.nr. 0923699
Bergstraat 63, 9717 LS Groningen, sybrand.v.d.werf@freeler.nl

Supervisor: Petra Hendriks

Kunstmatige Intelligentie
Rijks*universiteit* Groningen

968.

Index

Index **1**
I. Introduction **3**
II. Research Question **4**
III. Theoretical Background **5**
III.1. Groups in music 5
III.2. The GTTM-model of grouping 5
III.3. Optimality Theory 9
III.4. OT and Generative Musicology 12
IV. Methods **18**
IV.1. Implementing the model 18
IV.2. Testing the model 18
IV.3. Comparing results 18
V. Implementation **19**
V.1. Prolog 19
V.2. Well-formedness Rules 19
V.3. Preference Rules 20
V.4. GEN 23
V.5. EVAL 24
V.6. Conclusion 25
VI. Experiment **27**
VI.1. Experimental setup 27
VI.2. Stimuli 27
VI.3. Subjects 28
VI.4. Group-results 29
VI.5. Discussion group-results 30
VI.6. Results inter-subject differences 34
VI.7. Discussion inter-subject differences 35
VI.8. Conclusion 37
VII. Comparison of results **39**
VII.1. Predictions of the implementation 39
VII.2. Conclusions group-results 41
VII.3. Conclusions ISD 41
VIII. Conclusions **42**
VIII.1. Grouping in music 43
VIII.2. The relation to language 43
VIII.3. Two faces of the same problem? 44
IX. Literature **45**
IX.1. Literature 45
IX.2. Consulted websites 46

APPENDIX 1 – Code of "GTTM.ARI" **47**
APPENDIX 2 – Stimuli **52**
APPENDIX 3 – Individual Results of experiment **57**

...there is no art without constraint.
Abraham Moles

I. Introduction

Music and language share many properties. They are both time-structured soundpatterns, they both use a very elaborate syntax and share a lengthy vocabulary. For centuries, however, the theoretical approaches were totally different. With the rise of cognitive sciences and psychophysics the parallels are more and more used to obtain insights in both areas.

This thesis aims at uncovering some of the parallels between language and music. In order to shed light on these parallels a very specific case-study is performed: modelling the process that governs the grouping of musical phrases and the use of *Optimality Theory* for implementing a *musical parser*. The foundation for this parser is laid by the theory of *Generative Musicology*. The ideas governing this theory are in many ways comparable to a more or less recent development in linguistics: Optimality Theory, OT for short. In OT, a system of violable rules is used to describe a myriad of linguistic phenomena. In this thesis, the violable rules that govern the musical parsing come from generative musicology and the system of processing from OT. The resulting model is implemented using Prolog and this model is tested in a small experiment. At the end of this thesis the question will be considered in what ways music and language are comparable, not only in this particular case, but also in a wider perspective.

Here I wish to thank all the people that participated in my experiment. The discussions that took place with many of them after performing the experiment were often very valuable. The meetings with "the band": Maartje Schreuder, Menno van Zaanen and Dicky Gilbers, stood at the beginning of this thesis. I thank these people for their great support. Last but not least I would like to thank Petra Hendriks. Without her both pragmatic and critical, but always constructive remarks I never would have completed this thesis.

Samenvatting hoofdstuk *Introduction*
In deze scriptie zal een verband gelegd worden tussen taal (taalkunde) en muziek (musicologie). Deze onderzoeksgebieden hebben veel overeenkomsten. Aan de hand van een model uit de muziekpsychologie in combinatie met een taalkundige theorie zal blijken in hoeverre deze overeenkomsten bruikbaar zijn op beide terreinen.

II. Research Question

Together with an exploration of the correspondences between language and music, a theoretical model of how listeners parse musical surfaces in different groups will be the central objective of this thesis.

The research question falls apart in two separate questions:

- How do people parse a musical structure? How does this compare to the parsing of language?
- What can musicology and linguistics learn from each other – in particular at a cognitive level of consideration?

The research that has to be done for the thesis will consist of three parts: a study of relevant literature, implementing a musical parser and performing an experiment. Based on the literature a model is made, this model will be implemented and then an experiment will be performed, including both the model and human subjects, in order to test the underlying model. In the last part of the thesis, parallels between music and language will be discussed.

Samenvatting hoofdstuk *Research Question*

De onderzoeksvraag van deze scriptie bestaat uit de volgende twee gedeeltes:

- Hoe delen mensen muziek in groepen op en hoe verhoudt zich dat tot de manier waarop dat in taal gebeurt?
- Wat kunnen taalkunde en musicologie van elkaar leren, in het bijzonder vanuit een cognitieve zienswijze?

III. Theoretical Background

A study of the parallels between music and language must involve a study of the fields of research connected to these subjects. Theory of music is a very broad term which incorporates subjects like history of music, composition-techniques and psychology of music. In paragraph III.1 the notion of groups in music will be explained.
With the rise of cognitive science a new field has emerged: that of cognitive musicology. In particular, the rise of *Generative Musicology* (GM) as sketched in Lerdahl and Jackendoff (1983) is of interest to us. III.2 deals with their model of grouping in music.
Parallel to this development the study of general linguistics became more and more interested in cognitive insights, too, leading to cognitive and computational linguistics. One of the latest theories is *Optimality Theory* (OT), which will be given attention in section III.3.
In section III.4 we will turn to an OT-based approach of the grouping-process in music.

III.1. Groups in music

First of all, we must formalize the notion of a "musical group". What exactly is meant by a group in music? With a group we mean these notes that are more related than others. This relationship can be one of pitch, timbre, intonation, articulation, etc. An example from the Gestalt-psychology on grouping:

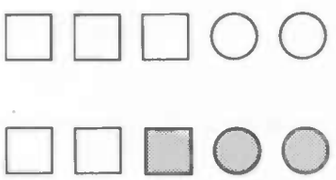


Figure 1
Conflicting grouping-cues

In figure 1 two strings of objects can be seen: squares and circles. The first string falls naturally apart in two groups, depending on the shape of the objects. When an other dimension or "cue" for grouping objects -for instance shading- is used also and the boundaries of the groups defined by both cues do not coincide (as can be seen in the second string), ambiguity arises. This can be compared to the grouping processes in music. Shape, colour, size etc. are replaced by interval, timbre, loudness, etc.
Duration plays a different role when grouping a musical surface. The difference between duration (or proximity in time) and other dimensions is that proximity is the only cue that is not based on equivalence. Where the same shape, colour, size, etc. binds objects together, (fysical) distance does not. This can be seen in figure 2.

Figure 2
Effect of spatial difference

Here, three groups are perceived by most people: first one square, then another one and than a group consisting of one square and two circles. The three squares all share the same proximity, but this does not bind them together in one group. This also holds for distance in time (read: length of notes).
Boundaries in music are comparable to the above Gestalt-examples. Often, the exact determination of a boundary is a intuitive and therefore personal judgement.

III.2. The GTTM-model of grouping

In *A Generative Theory of Tonal Music* Fred Lerdahl and Ray Jackendoff construct a (generative) theory of musical grouping. Lerdahl and Jackendoff indentify rules that coördinate the process of grouping pitch-events into larger-scale units. They distinguish two

types of rules: grouping well-formedness rules (GWFR's) and grouping preference rules (GPR's). GWFR's can't be violated and define all perceptual possible grouping-structures. GPR's are soft and may be violated in order to match other GPR's. They define a prefered structure. Optimality Theory (discussed in section III.3) makes use of violable rules in a comparable way.
An important feature of GTTM is that the theory uses an essentially input-driven or bottom-up approach. A certain musical surface is "fed" to the model, producing a prefered structure. The rules themselves, however, are not restricted in their direction of use and could be used top-down (judging a certain structure on its probability).
The GWFR's are as follows:

GWFR 1

Any contiguous sequence of pitch-events, drum beats, or the like can constitute a group, and only contiguous sequences can constitute a group.

GWFR 2

A piece constitutes a group.

GWFR 3

A group may contain smaller groups.

GWFR 4

If a group G_1 contains part of a group G_2, it must contain all of G_2.

GWFR 5

If a group G_1 contains a smaller group G_2, then G_1 must be exhaustively partitioned into smaller groups.

These rules define the possible structuring of musical surfaces. In a sense, a kind of tree-structure is described here. The first three rules define what groups should be made of: pitch-events or groups themselves. Here, the possibility of embeddedness is created: groups are allowed to be made of groups.
An objection might be made about the requirement in GWFR 1 that only contiguous events can constitute a group. When a sequence of tones jumps rapidly up and down between different frequency regions, a phenomenon known as *sequential integration* occurs (Bregman (1990)). If the alternation is fast enough and the frequency-spacing sufficiently large, one hears two different melodic streams (one of repeating notes in a low range of frequency, and one of repeating notes in a high one) instead of only one (a very rapid alternation of notes). This is a special case of *auditory stream segregation*. Although they are not contiguous, notes in the upper stream are grouped apart from those in the lower group. However, the effect of stream segregation is that a piece of music becomes polyphonic (more than one melodic stream at a time) instead of homophonic (only one melody). Lerdahl and Jackendoff explicitely point out that their theory is inadequate for this kind of music. Acknowledging this, we can preserve GWFR 1.
GWFR 4 prohibits the possiblity of overlapping groups. When part of a group is found to be part of another group, the whole group has to be contained by this last group. Two examples illustrate the effect of this rule. Both examples are discarded by GWFR 4.

Figure 3
Examples of groups violating GWFR 4

In both examples, the solid-lined groups contains part of the dashed-lined group but not all of it. In some cases, Lerdahl and Jackendoff argue, experienced listeners might judge a certain pitch-event as being both the last of a certain group and the beginning of another one. This could lead to the conclusion that GWFR 4 is actually a preference rule instead of a well-formedness rule, but this approach is left out of scope in this thesis.
GWFR 5 prohibits "empty" parts of groups. Note that it does not prohibit subdivisions of certain groups where others are not further divided. These structures are actually quite common.

Now we turn to the –more interesting– preference rules. We can distinguish two kinds of GPR's: those acting on the first level (the actual pitch-events), and those acting on groups.

First level grouping:

GPR 1 (SINGLES)

Strongly avoid groups containing a single event (Avoid analyses with very small groups – the smaller, the less preferable).

Rule 1 states that one will ideally group a single element together with adjacent events in the flow of music. A desired side-effect of this rules is that segmentation in a large amount of small groups is avoided: very small-scale grouping perceptions tend to be marginal.

GPR 2 (PROXIMITY)

Consider a sequence of four notes $n_1n_2n_3n_4$. All else being equal, the transition $n_2 - n_3$ may be heard as a group boundary if

a. (slur/rest) the interval of time from the end of n_2 to the beginning of n_3 is greater than that from the end of n_1 to the beginning of n_2 and that from the end of n_3 tot the beginning of n_4, or if
b. (attack-point) the interval of time between the attack points of n_2 and n_3 is greater than that between the attack points of n_1 and n_2 and that between the attack points of n_3 and n_4.

In rule 2 breaks in the musical surface in terms of proximity in time are detected. The first half, rule 2a, handles slurs and rests. When two notes are slurred, they are closer together in time: the end of the first note is closer to the beginning of the next as opposed to the case when they wouldn't be slurred. The same is true for a rest between two notes. Being closer together in this sense means there is more evidence to assign these notes to one and the same group.
Another type of proximity is that of the beginning of notes, the attack-points. When two notes have attack-points that are close in time and a third note begins after a longer period of time, the first two notes tend to be grouped together, apart from the third. This is stated in rule 2b.

GPR 3 (CHANGE)

Consider a sequence of four notes $n_1n_2n_3n_4$. All else being equal, the transition $n_2 - n_3$ may be heard as a group boundary if

a. (register) the transition $n_2 - n_3$ involves a greater intervallic[1] distance than both $n_1 - n_2$ and $n_3 - n_4$, or if
b. (dynamics) the transition $n_2 - n_3$ involves a change in dynamics and $n_1 - n_2$ and $n_3 - n_4$ do not, or if
c. (articulation) the transition $n_2 - n_3$ involves a change in articulation and $n_1 - n_2$ and $n_3 - n_4$ do not, or if
d. (length) n_2 and n_3 are of different lengths and both pairs n_1, n_2 and n_3, n_4 do not differ in length.

[1] The word "interval" is used both for temporal distance and distance in frequency. When used without explicit reference to time, "interval" always refers to change in frequency.

GPR 3 formalizes the intuition that notes with the same properties are grouped, or, stated in terms of boundaries, a boundary is placed between notes that differ with respect to their properties. These properties are register (frequency-range: when notes fall in the same range of frequency they tend to be grouped), dynamics (loudness), articulation (the qualitative character of the sound) and length (of time).
The rules stated above define how pitch-events themselves are grouped. Four additional rules are proposed that coördinate the process of grouping groups themselves.

Larger-level grouping:

GPR 4 (INTENSIFICATION)

Where the effects picked out by GPR 2 and GPR 3 are relatively more pronounced, a larger-level group boundary may be placed.

When the same boundary is being marked by multiple applications of GPR 2 or 3 or both, GPR 4 says to mark this boundary as one of a larger (i.e. containing other groups) group.

GPR 5 (SYMMETRY)

Prefer grouping analyses that most closely approach the ideal subdivision of groups into two parts of equal length.

GPR 5 reflects the slight bias towards binary structures. Groups should ideally be structured as containing two parts of equal length.

GPR 6 (PARALLELISM)

Where two or more segments of the music can be construed as parallel, they preferably form parallel parts of groups.

In GPR 6 a very important feature of musical groups is expressed: when certain groups can be seen as being "tied together" (motivic parallelism, same rythmic pattern, etc.), they should ideally occupy the same place in larger-level groups.

GPR 7 (TIME-SPAN AND PROLONGATIONAL STABILITY)

Prefer a grouping structure that results in more stable time-span and/or prolongational reductions.

As always, the venom is in the tail: GPR 7 states a very complex constraint for prefered groups. In their (very elaborate) exposition of their theory, Lerdahl and Jackendoff introduce the concepts of *time-span reduction* (assigning to the pitches of a piece a hierarchy of structural importance with respect to their position in grouping and metrical structure) and *prolongational reduction* (assigning to the pitch-events a hierarchy that expresses harmonic and melodic tension and relaxation, continuity and progression). Both concepts relate to the intuition that certain events in music are more important than other ones. Without corrupting the musical surface in a gross manner, we could leave the less important events out. Repeating this process, whole symphonies can be reduced to a single chord. *A Generative Theory of Tonal Music* provides an extensive explanation of both notions.

When implementing the above sketched processes, a certain conflict-resolution-system is needed when GPR's are in conflict. Mostly, the rules won't clash because all rules act on different aspects of a musical phrase (e.g. pitch, intensity, etc.). When rules do collide, Lerdahl and Jackendoff point out that this is due to ambiguity in the surface. This ambiguity is experienced by listeners, too. They argue this psychological reality should be kept intact and therefore the problem of conflict-resolution is not resolved. When discussing Optimality Theory in the next section, strict hierarchy of rules is introduced as a means of conflict-resolution.

In this thesis only first-level grouping is considered. This is done for a number of reasons.

- Higher-level grouping leans more heavily on knowledge than first-level grouping. This can alone be seen from the GPR's governing both processes. GPR's 1 to 3 act on notes, GPR's 4 to 7 lean on more complex aspects such as symmetry, harmonic tension, etc. This means the prior knowledge probably plays a greater role in higher-level grouping. The same can be seen with visual inputs. Small dots arranged in a line are perceived as being a line by all spectators, but depending on their knowledge and experience they might "group" the dots into a bridge or (even larger-grouped) a painting by Georges Seurat[2].
- As will be seen in section V.4, one of the difficulties arising with the implementation of the theory is computational complexity. Sticking to first-level grouping alone keeps the problem tractable.
- The larger the structure, the less evidence there is to keep it active in memory. Listeners can keep track of short pieces and remember the heard events, but as the piece progresses, loss of memory will make the "housekeeping" that is necessary to keep large groups together impossible[3].

For these reasons (and the more pragmatical one that modelling all GPR's would satisfy for a PhD-scholarship) only first-level grouping will be considered.

III.3. Optimality Theory

The rise of *Generative Linguisitics* since the first publications of Noam Chomsky in the 1960's and 70's, lateron evolving into his *Principles and Parameters Theory* (PPT) meant a breakthrough in linguistics. In PPT sets of rules (or principles) are postulated that form the *Universal Grammar* (UG) shared by all languages. This UG is supposed to be able to distinguish between grammatical and ungrammatical structures. To provide an explanation for the many differences between different languages, all principles have certain parameters that are language-specific[4]. To give a parallel example from physics: "every object in vacuüm falls according the gravitational law (displacement = $\frac{1}{2} g \cdot time^2$)" can be said to be a principle, the exact value of the gravitational constant g differs from planet to planet and is therefore a parameter to be set.

One of the problems often mentioned in relation to PPT is its immense complexity. Using laws in a complex domain such as language the same way one would use them in physics means defining a large amount of exceptions. In reaction to PPT a linguist and a physicist, Alan Prince and Paul Smolensky, proposed a totally new approach to the theory of language (Prince & Smolensky (1993)). Instead of a no-compromise rule-based approach as was advocated by PPT they proposed a system of violable rules that select an optimal structure from a number of candidates: *Optimality Theory* (OT). This optimal structure corresponds to the grammatical one as was seen in PPT. The approach originated in phonology, but soon found its way to other regions of linguistics as well. In only ten years time OT developed into a full-fledged theory of language, accompagnied by implementations in various fields.

The basic of OT is formed by the following routines:

- GEN
  In this part of OT, all possible candidates are generated given an initial input. This input is for instance a lexical entry that has to be pronounced, an unparsed sentence, etc. A

---

[2] Georges Seurat (1859 – 1891), French pointillistic painter. The pointillists only use small, coloured dots in their (figurative) paintings.

[3] An anecdote illustrates the effect of training: Wolfgang Amadeus Mozart (1756 – 1798) heard the *Miserere* by Gregorio Allegri (1582 – 1652) only one time and reproduced the complete score in his hotelroom hours later. The *Miserere* has a duration of approximately 13 minutes.

[4] Another aspect of PPT is the claim that the UG is innate. Instead of having to construct a complete grammar during childhood, everyone is born with knowledge about language (principles) with only parameters to be set. This explains the emphasis laid on acquisition.

very important aspect of the theory is that this set of candidates is, in principle, infinite in size. This is because the set of candidates should contain all imaginable structures. In – for example– the case of parsing words into syllables this means GEN has to encompass the processes of epenthesis (insertions of other phonemes) and deletion of phonemes[5]. Because of the strict distinction between the generation- and the evaluation-process it is important that the set of candidates *is* infinite. Decreasing the size of this set would inevitably mean doing some sifting-work in GEN.

- EVAL
  The candidates generated by GEN are passed to the evaluation process, EVAL. EVAL selects from the set of candidates the optimal one given a set of constraints, CON. These constraints are violable and ordened in a strict hierarchy. Violability means that even the optimal candidate can violate certain constraints. The strict hierarchy means that the optimal candidate never violates a higher ranked constraint in order to satisfy any number of lower ranked constraints.
- CON
  The set of constraints is shared by all languages and linguistic variation is due to the ranking of the constraints. Two important categories of constraints can be distinguished: faithfullness-constraints and markedness constraints. Faithfullness means that the optimal candidate should be as close as possible to the initial input. In our syllabification example that means no epenthesis and deletion. "Markedness" means in what extend a property (described in the constraints) is very specific for a certain language or found in many other languages. Unmarked properties are shared by almost all languages and correspond with higher ranked (and therefore rarely violated) constraints, marked properties are language-specific, corresponding with lower ranked constraints.

An example of syllabification will illustrate the working of OT. A syllable falls apart in three structural elements: an onset, a nucleus and a coda. In the nucleus a syllable reaches the top of its sonority or intrinsic loudness. In principle, the nucleus consists of a vowel, although some languages permit certain voiced consonants (/l/, /r/, /n/, /m/) to serve as a nucleus. The nucleus can be preceded by a one or more consonants (an onset) and/or followed by one or more consonants (a coda).
The following example concerns the syllabification of the Dutch word /auto/ (car). The table below (an OT-tableau) will provide insight in the working of the OT-processes and the constraints.

	/auto/	FAITH	ONSET
☞ a	au.to		*
b	tau.to	*!	
c	aut.o		**!
d	taut.o	*!	*

Figure 4
Example of an OT-tableau, syllabification of the word */auto/*

The first column of the tableau holds the candidates. A period marks the boundary between syllables, epentheses are in bold face (the epenthetic t is just an example, it could have been any consonant). The other columns hold the different constraints. In this tableau, FAITH and ONSET are used, stating the following:

[5] An example of both is the Dutch verb *werken* (to work). It is often pronounced as /wɛruku/ (u denotes the schwa). Here, we can distinguish an epenthetic schwa (given in bold-face) and the last /n/ is deleted.

FAITH

Pronounce everything as it is.

ONSET

Syllables must have onsets.

ONSET is a markedness constraint: an optimal syllable should have a consonant in the first position. In this example, FAITH is higher in the hierarchy than ONSET. This shows from the ordering of columns: stronger constraints are on the left, weaker on the right. As can be seen, candidate a is chosen to be optimal, because it doesn't violate FAITH and violates ONSET only once. The optimal candidate is marked with ☞, violations by an asterisk and the violation that eventually rules out a specific candidate with an exclamation-mark. When a candidates is ruled out, the cells corresponding to the applications of lower-ranked rules are shaded.

To account for possible syllabification structures we need at least one more constraint (NOCODA):

NOCODA

Syllables end with a vowel.

Now we can distinguish the following four hierarchies:

1. FAITH » { NOCODA, ONSET }
2. { NOCODA, ONSET } » FAITH
3. ONSET » FAITH » NOCODA
4. NOCODA » FAITH » ONSET

Note that ONSET and NOCODA act on different constituents and don't clash. Therefore, their mutual ordering is of no relevance and they can be seen as occupying the same place in the hierarchy when they are in contiguous hierarchic places (as can be seen in the first two orderings). The given hierarchies predict four types of syllabification structures and, significantly, each of these four types (and exactly these) do occur. Examples taken from Archangeli & Langendoen (1997) (O = Onset, N = Nucleus, C = Coda, elements between parentheses are possible but not required):

1.	(O)N(C)	English
2.	ON	Senufo
3.	ON(C)	Yawelmani
4.	(O)N	Hawaiian

In the first hierarchy, the pressure to pronounce the segments of the input the way they are outweighs having an onset or missing a coda. Therefore, all syllables are of a (O)N(C)-form: both onset and coda may be present, depending on whether they are present in the input. This holds for English. In –for example– the fourth hierarchy onsets are optional and codas strictly disallowed (the input-structure may be corrupted in order to satisfy this constraint) resulting in a (O)N-language such as Hawaiian.

Note that, due to the intrinsic mechanisms of OT, all constraints are formulated as general as possible: "Every syllable has to be ...". In short: rules are stated in a hard way and processed in a soft way.

A short summary of OT in the form of a tableau:

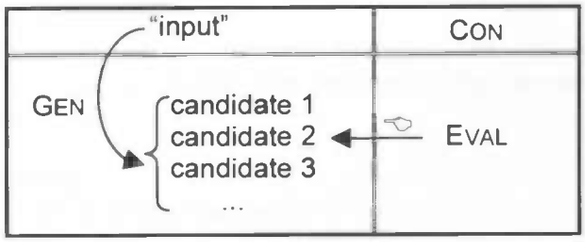


Figure 5
Schematic outline of Optimality Theory

Given a certain input, GEN generates candidates, out of which EVAL chooses an optimal one depending on the strict ordered constraints in CON.
Optimality theory, with its violable rules ranked in a strict hierarchy, provides a consistent framework with successes in various fields such as phonology, syntax and semantics. We will now turn to the correspondences between OT and the theory outlined in section III.2 concerning the GTTM-model of grouping.

III.4. OT and Generative Musicology

The violability of rules as seen in OT is reminiscent of the preference rules from GTTM. The GPR's must be violable in order to describe the possibilities and variation inherent to the grouping-process and look like OT-constraints in that sense. OT can provide the "toolkit" (GEN and EVAL) to complete an OT-based theory of grouping in music.
But is the process of grouping in music really an OT kind of process? When listening to music, the grouping process unfolds in time. During listening, hypotheses for possible groups rise and fall. Other than in standard OT[6], where the process is essentially output-driven, the total output is not yet known when hearing music. Rather, listeners tend to judge each note whether this note belongs to the former group or should be the beginning of a new one. This judgement however leans heavily on the expectations one has of the continuation of the piece. This expectation, together with the already heard and analyzed material constitutes the context in which a note is being processed and this total context fullfills the same role as output in OT.
Building an expectation of the development of a certain piece is a highly complex task. It incorporates rythm, harmonic development, melodic consistancy and also higher-level knowledge such as knowledge about form, counterpoint and instrumentation. To give an example: when a certain "classical" piece for orchestra and soloist reaches a *ff* I^6_4 chord (typically at the end of the first part of a three-part piece), a high-level expectation could be that the cadenza (a passage for only the soloist, originally an improvisation on before-used themes) is reached. Over-sophisticated as it might appear, this could have an effect on the grouping of the mentioned chord and surrounding pitch-events.
The rules proposed by Lerdahl and Jackendoff are not the same as constraints in OT. In order to fit in an OT-framework, these rules have to be modified into a form where we can determine whether the rule is violated. When this is done, simple OT-tableau's for musical grouping can be made. This means the rules have to be strengthened: they must define the prefered structure instead of judging the preference itself. The violable character of the rules is not in the rules itself (as is the case with Lerdahl and Jackendoff), but in the processing of them. Below, we will modify the GPR's into constraints. For every GPR, this alterated version is given (in section III.2 the original rule is stated) and a couple of examples are given to provide insight into the precise nature of the rule.

[6] Fanselow (1999) describes how a OT-parser might be hypothesized that derives parsing preferences in an incremental fashion as new input becomes available. Müller (2002) suggests an even more limited approach where only local optimization is used (and not even earlier processed material as with Fanselow).

GPR 1, SINGLES

This rule is a gradual one. Because OT assumes that rules are violated or not, this gradual component has to be removed. As Lerdahl and Jackendoff do themselves at first, we restrict our scope to one note only.

GPR 1 (SINGLES) – *modified*

Groups never contain a single element.

Given this rule, the fact that a two-note group might be prefered above a three-note group is neglected. A solution to this problem might be the following: we replace GPR 1 by a – theoretically infinite – number of rules stating the following:

GPR 1 – *1* Groups never contain one element.
GPR 1 – *2* Groups never contain two elements.
GPR 1 – *3* Groups never contain three elements.
...

The hierarchy of rules as proposed in OT works in our advantage here: we can place every next rule lower in the hierarchy. Rule 1 – *200* may exist, but is always violated. Because it must have a very low position in the hierarchy, this won't be a problem.

GPR 2, PROXIMITY

GPR 2 falls apart in two different cases, presented here as GPR 2a and GPR 2b. In the original form stated in GTTM, the rule marks a boundary between two groups; given the fact that two notes in a sequence are "closer" together than two following notes, mark the transition from the second to the third note as a boundary.
As was stated in the first section of this paragraph, we can only determine whether a note belongs to the already processed group or marks the beginning of a new one. In GPR 2, Jackendoff and Lerdahl take four notes and decide whether or not we should mark a boundary between note 2 and 3. The note actually being processed here is note 3: does this note mark the beginning of a new group or doesn't it? When listening to music and being confronted with this task, note 4 hasn't been heard yet. Therefore, we can't include this event in our decision. Apart from re-writing the preference rules into constraints, we lose the fourth note in our rules, too.

GPR 2a (PROXIMITY SLUR/REST) – *modified*

No group contains a contiguous sequence of three notes, such that the interval of time from the end of the second note to the beginning of the third is greater than that from the end of the first note to the beginning of the second.

GPR 2b (PROXIMITY ATTACKPOINTS) – *modified*

No group contains a contiguous sequence of three notes, such that the interval of time between the attackpoints of the second and the third note is greater than that between the attackpoints of the first and the second note.

The following OT-like tableau illustrates the use of the rules (note that the ordering of the rules has not been established yet, so the ordering in the tableau is arbitrary):

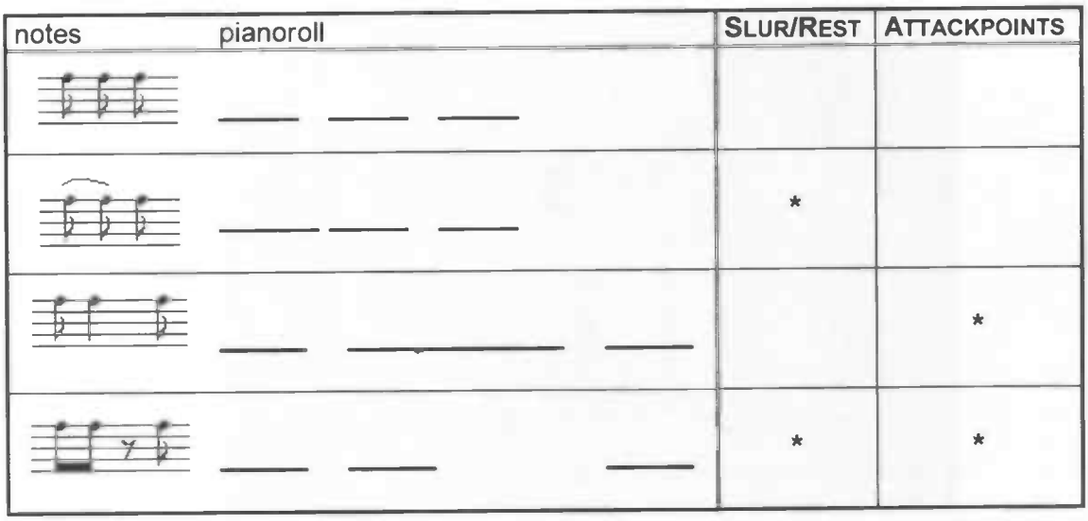


Figure 6
Working of GPR's 2a and 2b

Given that the notes in the first column constitute a group, the asterisks mark which rule is violated. The pianoroll-notation, where length of the line corresponds to duration of the note makes it clear how the rules apply. The first example obviously doesn't violate any of the two rules. The notes are equally spaced and of equal length. The second group violates GPR2a: the end of the first note is closer to the beginning of the second than the end of the second note is to the beginning of the third. In the actual case of a slur the transition of note 2 to 3 will not have a noticable rest at all. Because the onsets of the notes are still equally spaced, rule 2b isn't violated. This doesn't hold for the third example. The gaps between the notes are now of equal length (so rule 2a has no objections), but the onsets differ in length. In the fourth example both rules are violated. Clearly, these notes can not share the same group given GPR 2.

GPR 3, CHANGE

The last of the lower-level grouping preference rules falls apart in four different cases. The third case, change of articulation, is left out of scope because articulation is a very complex feature. In a score articulation can be marked with special symbols such as · (staccato; note is played very short), ^ (martelé; short and strong), ¯ (portato; slightly longer with emphasis) or ı (wedge; short with emphasis). In most cases, however, articulation is left to the performer. Therefore, articulation varies from note to note.

When looking at the audio-signal, a note can be seen as a summation of sine-waves. The feature that all (or at least the most prominent) sine-waves are multiples of the same fundamental frequency makes the signal a musical one. These sines are called *harmonics*. The fundamental frequency (often being the first harmonic, this however is not obligatory) is identified by the listener as the pitch of the signal. Articulation depends on the exact configuration of the harmonics.

In figure 7, three spectra of a note of 440 Hz can be seen. The instrument used is a viola. The first signal is a staccato note, the second a portato one and the third is played pizzicato (plucked). Time is on the horizontal axis, frequency on the vertical. The three notes are played immediately after one another.

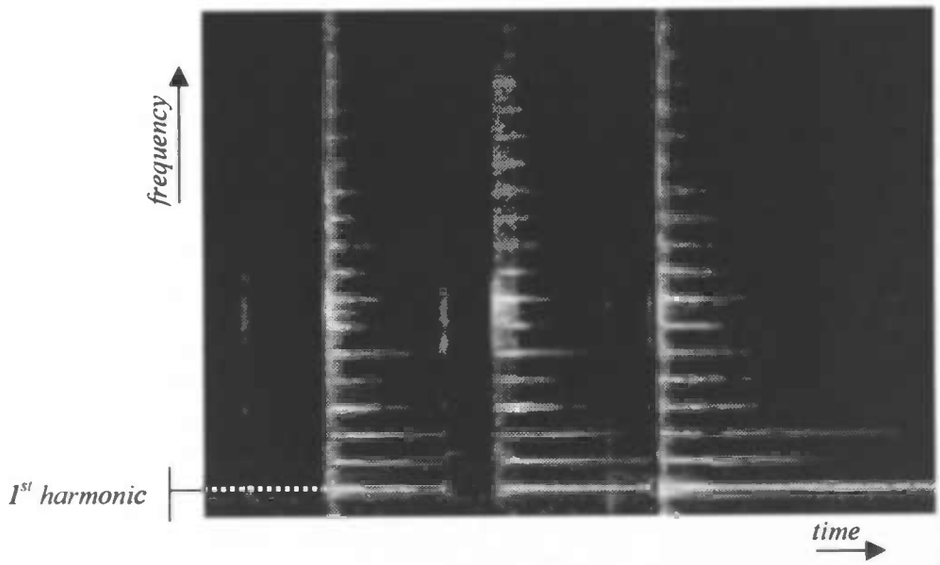


Figure 7
Spectrum of three tones differing in articulation

The harmonic structure can clearly been seen as horizontal stripes in the spectrum. The fundamental frequency corresponds to the separation of the harmonics and equals the first harmonic (in this case): 440 Hz. The only virtually deducible feature that could point the second signal out as being the portato one is that is is slightly more "fuzzy" at the beginning over the whole range of frequency. This is due to the fact that a bowed note often has a slight noisy quality. It is important to keep in mind that all three notes are very different in quality for every listener. This spectrum clearly illustrates the virtual absence of distinguishable features in the signal that could keep the three (perceptual totally different) kinds of articulation apart. Therefore, GPR 3c is left out of our model.

In their theory Lerdahl and Jackendoff focus on the score instead of the actual signal. GPR 3 points to the practical advantage of this approach. In short, this preference rule marks a boundary when two notes are the same and a third is not. As opposed to the score, when looking at the physical signal, no two notes will ever be the same. The timing of a note (and therefore length) might differ, the root-mean-square value of the signal (corresponding to intensity) fluctuates etcetera. In order to use the concept of "equal" some adjustments have to be made to the model. All parts of this rule ask for a different line of attack. The approach chosen might not be the only nor the best one, but follows Lerdahl and Jackendoff as closely as possible, with this important difference that we consider only three notes, as we did with rule 2. The modified formulations of rules 3a, 3b and 3d are given below.

GPR 3a (CHANGE REGISTER) – *modified*

No group contains a contiguous sequence of three notes, such that the interval from the second note to the third is bigger than that from the first note to the second.

GPR 3b (CHANGE DYNAMICS) – *modified*

No group contains a contiguous sequence of three notes, such that the first two share the same dynamics, different from the third.

GPR 3d (CHANGE LENGTH) – *modified*

No group contains a contiguous sequence of three notes, such that the first two share the same length, different from the third.

Rule 3a deals with melodic interval. Every normal melody contains many different intervals. As was seen before, the rule actually states that when notes fall in the same range of frequency they tend to be grouped. Here the rule is formulated in a negative and –more important– more narrow way: no group contains a leap in frequency larger than the preceding one. When only considering three notes, the order of the notes becomes more important. The constraint now does not prohibit groups where the intervals decrease. This means groups can still contain large leaps in pitch, as long as the next interval isn't larger. These last groups are not assumed to be optimal in GTTM. Sticking to the principle that not yet heard notes should be left out of consideration deviates from GTTM here.
The next fragments show the difference between GTTM and the above stated constraint:

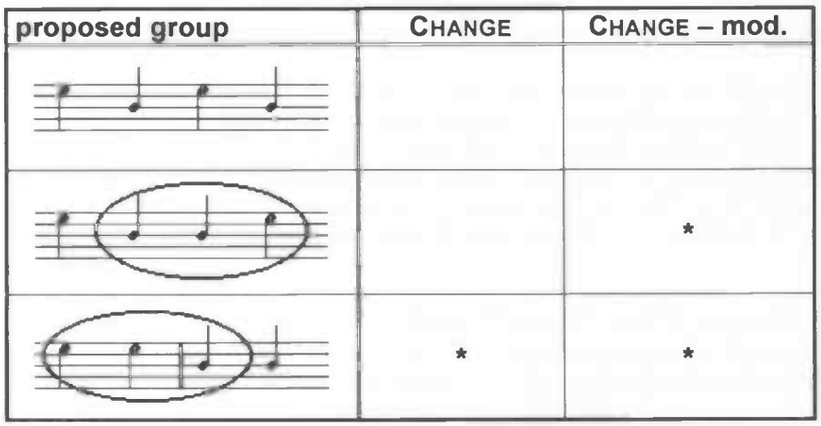


Figure 8
Working of CHANGE REGISTER in GTTM and the current model

Where a boundary is detected by the original rule (only in the third fragment), a vertical line is drawn in the score. The circles give fragments of the group that violate the modified constraint. Because the modified version works on three notes only and the original rule requires both intervals adjacent to the boundary to be smaller than the interval across the boundary, the modified version is slightly more conservative than the original one.
Problems might arise when intervals are the same. Because the exact frequency, given in Hertz, is often a rounded estimate, intervals that are equal might not return exactly the same numerical result. We turn to this problem concerning quantisation in section V.3 when discussing the implementation of the preference rules.

The same problem rises when treating dynamics or intensity of the signal. The solution is simple: instead of taking the actual, physical intensity of the signal (in dB or phones, for example), the dynamics-mark from the score is used. When viewed in a score, intensity has only a limited set of possibilities ranging from pianisissimo (very very soft) to fortisissimo (very very loud): *ppp, pp, p, mp, mf, f, ff, fff*. Rare examples exist of *pppp* and *ffff* is even more scarce[7]. Given the fact that no term apart from "four-double forte" exists for these marks, they can be safely left out of scope. When using these marks for dynamics in our model instead of numerical scores, GPR 3b can remain unaltered using this "terraces" of dynamics, using (in)equality of mark. In the actual, physical signal, intensity is a continuous variable corresponding to amplitude. This would mean we would have to define a measure for

[7] The first movement of Piotr Illich Tchaikovsky's (1840 – 1893) Symphony no. 6 (the *Pathétique*) provides an example of both rare dynamic marks. On first glance, *ffff* only appears in the seventh symphony of Gustav Mahler (1860 – 1911) and more contemporary works.

difference in intensity and a treshold should be defined. When do we leave the *pp*-region and enter the *p*-region? A side-effect of the use of a symbolic representation of dynamics is the inability of dealing with (de)crescendo's.

In order to use GPR 3d the input has to be assumed to be quantized so small deviations in length will have no effect and a quaver can be assumed to have exactly the same length as all other quavers.

Samenvatting hoofdstuk *Theory*

Het eerste gedeelte van dit hoofdstuk behandelt het concept van een *groep* in muziek. Onder een groep wordt verstaan die noten die meer bij elkaar horen dan anderen. Uit de Gestalt-psychologie met betrekking tot visuele groeperingen werden enkele parallellen aangehaald.

Vervolgens komt de Generatieve Theorie van Tonale Muziek (GTTM) van Lerdahl & Jackendoff aan de orde. Zij geven een op taalkundige principes gebaseerde theorie van de manier waarop een luisteraar een bepaald muziekstuk in groepen opdeelt. Hierbij wordt onderscheid gemaakt tussen welgevormdheids-regels (GWFR's) en voorkeurs-regels (GPR's). GWFR's definiëren waaraan mogelijke groeperingen in ieder geval moeten voldoen, GPR's geven aan welke groeperingen de meeste voorkeur genieten. De GPR's zijn schendbare regels.

De schendbaarheid van regels die een voorkeur weergeven voor bepaalde linguistische structuren is een van de pijlers van de Optimaliteits Theory (OT) van Prince & Smolensky. Deze theorie beschrijft hoe mensen op basis van een bepaalde mentale invoer een (in theorie oneindig grote) groep kandidaten genereren en daar vervolgens op basis van schendbare selectievoorwaarden een optimale kandidaat uit kiezen.

Tot slot worden de regels uit GTTM opnieuw gegeven, nu in de vorm van OT-regels.

IV. Methods

IV.1. Implementing the model

The first step in testing whether the rules proposed by Lerdahl and Jackendoff do what they are designed for, modelling the way people "parse" music, is implementing these rules. A few remarks should be made here concerning the nature of the process. As was seen above, the process of grouping music has an individual component. That is, different subjects might judge the same structure differently. An ideal model should be able to predict different outputs only by setting a small set of parameters. The ordering of rules is such a parameter. When a different ordering is able to predict inter-subject differences without altering the rules themselves, the latter gain a lot of credibility. During implementing the model the theory of Lerdahl and Jackendoff is followed as close as possible.

IV.2. Testing the model

The rules proposed by Lerdahl and Jackendoff are based upon their own musical intuitions. Although these may very well be correct and sometimes even seem to be trivial, the authors have taken no effort to prove their actual existence as rules that govern the psychological process of musical grouping. In order to do this, the rules by themselves are not enough. Constraints based on the rules from GTTM in combination with the assumptions of strict hierarchy and violability of constraints (taken from Optimality Theory) as described in this thesis, form a coherent model of how people "parse" music that allows for experimental testing. In order to test the model on its psychological reality an experiment is performed with a set of ten subjects.
Because the constraints often interact, (it's almost impossible to construct stimuli that violate only one constraint in all candidates), it is hard to test their validity itself. Therefore, the main aim of this experiment is the hierarchy of constraints. The constraints used are as formulated in section III.4. Both group-results and inter-subject differences are considered.

IV.3. Comparing results

The resulting hierarchy of rules is implemented in the computational model, that will produce the OT-tableaux corresponding to the reaction given by the subjects. On this basis, we can determine whether the rules proposed really exist, whether the model should be altered or whether a totally different approach should be taken.

Samenvatting hoofdstuk *Methods*
In dit hoofdstuk wordt kort uiteen gezet dat, ten einde een antwoord te geven op de onderzoeksvragen, een experiment zal worden uitgevoerd en een computationeel model gemaakt zal worden.

V. Implementation

V.1. Prolog

In language-orientated programming, the logical language Prolog is very popular. A very important reason for this is the goal- and object-orientated behaviour of Prolog. The language is centered around a small set of basic mechanisms such as pattern matching, tree-based data structuring and automatic backtracking. This makes it very suited for problems involving objects and their relationships. Because many problems encountered in language processing can be expressed in these terms, Prolog is frequently used. Special procedures are even incorporated that handle grammar rules, for example. All statements in Prolog are predicates. For example, the knowledge that J.S.Bach was born in 1685 could be stated as `born(bach,1685)`. This example uses the predicate `born` with as first argument the name and second the year of birth. The convention to denote the number of arguments after a slash when refering to the predicate (e.g. `born/2`) is used throughout this thesis. Variables are given an upper-case beginletter, facts are lower-case (e.g. `Composer` might be instantiated by `bach`).

V.2. Well-formedness Rules

Using Prolog-lists as representation of grouped structures seems a logical choice. But doing this, a lot of assumptions are already made. These assumptions are exactly the ones made by GWFR's 1 to 4. GWFR 5 doesn't automatically follow from the nature of Prolog-lists, but will always be met when dealing with first-level grouping as will be shown below. All GWFR's are discussed below, showing how they are automatically implemented in Prolog.

GWFR 1 – *Any contiguous sequence of pitch-events, drum beats, or the like can constitute a group, and only contiguous sequences can constitute a group.*
One property of Prolog-lists is the impossibility to count non-contiguous elements as belonging to one and the same list. Only when the elements in between also share the same list, all these elements constitute a list.

GWFR 2 – *A piece constitutes a group.*
When we are dealing with an organised data-structure such as a list, there has to be a top-level. In the case of a musical piece, we call the top-level the piece itself, when talking about lists, it is just the list that includes all others lists.

GWFR 3 – *A group may contain smaller groups.*
In Prolog, lists are allowed, but not required to contain smaller lists. This is exact the same criterium as seen with the Lerdahl and Jackendoff's groups.

GWFR 4 – *If a group G_1 contains part of a smaller group G_2, it must contain all of G_2.*
Cross-references are not possible with Prolog-lists. For example: `[1,[2,3],4]` is interpreted as being a bedded list instead of two intersecting lists. A language where these intersecting lists are possible (but not allowed) is HTML:`<A> 1 <B> 2 3 </A> 4 </B>`. This structure is an example of two cross-referred lists.

GWFR 5 – *If a group G_1 contains a smaller group G_2, than G_1 must be exhaustively partitioned into smaller groups.*
The following Prolog-code implements exactly the above stated rule:

```
gwfr5(Structure) :-
        not containslist(Structure).
gwfr5(Structure) :-
        containslist(Structure),
        listlist(Structure).
```

With `containslist(A)` is true when A contains a list and `listlist(A)` is true when A is made of lists. A is a list. The first part of the rule is a check whether a structure contains a list. When it does, the second parts provides that this structure should be composed of lists entirely.
Note that GWFR's 3 to 5 can only be violated when dealing with larger-level grouping structures, therefore they are not of direct interest to our goals. All the grouping well-formedness rules are automatically hold when using Prolog-lists and *only* these assumptions are implicitely made. We will turn to the implications of this when discussing GEN in section V.4. Because the well-formedness is inherent in the use of Prolog-lists, one should use a different representation (or even programming-environment) when considering ill-formed structures, too. We will now turn to the implementation of the preference rules.

V.3. Preference Rules

In section III.4 the rules given by Jackendoff and Lerdahl were translated into OT-constraints. Now we have to translate these constraints into Prolog-code. In order to understand how the code works, we have to have an understanding of how notes are represented in the program. The representation of the notes, specially created for this application, are 4-tuples of on-time, off-time, frequency and a dynamics-mark. The on- and off-time are both measured in milliseconds relative to the beginning of the piece, frequency is given in Hertz. Dynamics can range from *ppp* to *fff*, as was argued above.
An example is taken from Mozart Symfony nr. 40, the first part of the main theme. A quarter-note (one beat) = 120 bpm (beats per minute, corresponds in this case to a rate of 2 quarters per second). This fragment lasts up to 3500 milliseconds. This particular example is frequently used in Lerdahl & Jackendoff because its grouping-structure is very straightforward.

Score of the fragment (this is a simplification of the actual music, where first three beats rest are encountered and more slurs):

Representation (`n[ontime, offtime, frequency, dynamics]`):

```
[
n(   0, 200,622,p), n( 250, 450,587,p), n( 500, 950,587,p),
n(1000,1200,622,p), n(1250,1450,587,p), n(1500,1950,587,p),
n(2000,2200,622,p), n(2250,2450,587,p), n(2500,2950,587,p),
n(3000,3450,932,p)
]
```

The passage begins with two eight notes (both 200 ms), first e flat (622 Hz) and then d (587 Hz), etc. The dynamics of all notes is *piano*. Note that there is a small pause of 50 ms between all notes, known as the inter-note-interval (INI). This interval decreases when notes are slurred and increases in the case of a rest. The value of 50 ms taken here corresponds to real values.

Now the representation of notes is clear we can turn to the grouping preference-rules. All GPR's are of the following form:

```
gprX(Struc,1) :-
        violate(Struc),
        !.
gprX(_,0).
```

The predicate `gprX/2` returns a value of 1 (in the second place of the predicate) when the given structure violates this particular rule. The _ in the second part of the rule is a special variable that can be instantiated by anything (also by violating structures). The first half of the code makes use of the cut-operator !. The cut is a means of controlling back-tracking. When a ! is encountered, Prolog stops trying to find other instantions of the predicate. The meaning in natural language of the above code is: "When `violate(Struc)` succeeds, return 1, 0 *otherwise*". This works because Prolog handles rules in a serial order, trying higher placed rules before lower placed ones. Instead of having to define both violating and prefered structures we can stick to the violating cases only, followed by a !. Prolog returns the value of 1 immediately, not bothering to consider the non-violating case. Only when `violate(Struc)` fails, the second part of the rule is considered. Because _ can be instantiated by anything, we don't have to bother to describe exactly what non-violating structures look like.
Now the GPR's are considered. Each rule is repeated in more or less informal form.

GPR 1 – *Groups never contain a single element.*
As was seen in section III.4, GPR 1 poses the problem of being a gradual rule: the smaller a group, the less prefered. Instead of using a clear one-element-only strategy, we can define an infinite number of rules stating the maximum length of a structure. In this implementation, only GPR 1-1 was used:

```
% GPR 1: SINGLES
gpr1([n(_,_,_,_)],1) :- !.
gpr1(_,0).
```

But we could include all other length-rules:

```
% GPR 1-2
gpr12(Struc,1) :-
        length(Struc,2),
        !.
gpr12(_,0).

% GPR 1-3
gpr13(Struc,1) :-
        length(Struc,3),
        !.
gpr13(_,0).

% ...
```

GPR 2 – *Groups never contain notes with a large amount of time in between.*
Other than in rule 1, a bit of calculation has to be done here. GPR 2a compares the intervals of time between notes (only the "violate"-part of the code will be shown from now on):

```
        subset(Struc,[n(_,Off1,_,_),n(On2,Off2,_,_),n(On3,_,_,_)]),
        Int1 is On2 - Off1,
        Int2 is On3 - Off2,
        Int2 > Int1,
```

The predicate `subset/2` is true when the structure contains a list of notes of the form formulated in the second argument of the predicate. This subset contains contiguous elements only. Here, Prolog "reads" off- and on-times from this subset, calculates their differences and compares them. Int1 is the interval of time between note 1 and 2 and int2 that from note 2 to note 3. The rule checks whether the second interval is larger than the first one. When this is true the rule is violated, as was required in section III.4. Note that the code

stated here defines the terms on which the rules are *violated* instead of describing prefered groups.

GPR 2b compares the intervals of time between attackpoints in the same way:

```
subset(Struc,[n(On1,_,_,_),n(On2,_,_,_),n(On3,_,_,_)]),
Int1 is On2 - On1,
Int2 is On3 - On2,
Int2 > Int1,
```

Here the intervals between the attackpoints are compared. When the second interval is larger, the rule is violated.

GPR 3 – *Groups never contain a change in interval, dynamics or length.*
Rule 3a deals with change of interval ("interval" is used here to refer to change in frequency). To determine the interval, the predicate `int/3` is called. This predicate returns the intervallic distance between two frequencies, expressed as their ratio. Frequency has a logarithmic relationship to pitch: an octave higher (or adding twelve semitones) corresponds to multiplying the frequency with a factor two. Therefore, subtraction of the frequencies will not yield the correct value of intervallic distance and the ratio is used.

```
subset(Struc,[n(_,_,F1,_),n(_,_,F2,_),n(_,_,F3,_)]),
int(F1,F2,Int1),
int(F2,F3,Int2),
Int2 > Int1,
```

In GPR 3b, another type of change is used: that of dynamic mark. GPR 3b is violated when the dynamics of the first two notes in a subset of three match and the third does not.

```
subset(Struc,[n(_,_,_,Dyn),n(_,_,_,Dyn),n(_,_,_,Oth_dyn)]),
not Dyn = Oth_dyn,
```

This comparison of properties is almost the same as in rule 3d, but here the lengths have to be obtained first from the on- and offsettimes:

```
subset(Struc,[n(On1,Off1,_,_),n(On2,Off2,_,_),n(On3,Off3,_,_)]),
Dur1 is Off1 - On1,
Dur1 is Off2 - On2,
Dur3 is Off3 - On3,
not Dur1 = Dur3,
```

The variable `Dur1` corresponds to the length of both the first and the second note. In the last clause of the rule, the lengths of note 1 and 3 are compared. Note that the rule is violated when both durations are not exactly the same. This means the input is supposed to be quantised. Quantisation is a process where notes of approximately the same length are rounded to equal values. Where, for example, two quarter notes are written, not a single performer will succeed in playing the two notes with exact equal length. Still, listeners perceive them as sharing the same length or, to be more precise, the same rythmic category: "quarters". Quantisation rounds notes of the same category to exactly the same duration. When the input is unquantised, a treshold should be used in order to let this rule be of any significance[8].
Now the preference rules are defined, they can be used by the EVAL-process, described in section V.5.

[8] In Temperley (2001) so-called "pips" are used: frames of 15 ms. All events falling within a pip are seen as occuring at the same moment.

V.4..GEN

GEN is the OT-mechanism that generates a set of candidate-structures. When actually implementing OT, an important problem has to be solved. As was seen when discussing OT, GEN theoretically generates an infinite set of candidates. This of course poses a computational impossibility and heuristics must be used to limit the size of the set of candidates. As was seen in section V.2, using Prolog-lists limits our representation of structures to well-formed ones only. This means that GEN will only produce well-formed structures. The set of candidates structures is now finite. This obviously has a computational advantage, but is in contradiction with the assumptions made in OT.
Two remarks have to be made here:

- When listening to music, two kinds of groups could be distinguished: psycho-physic groups (notes that are grouped by a non-conscious process, done so on basis of the physical signal; corresponding to well-formed structures) and cognitive groups (groups that are made based on higher-level knowledge of music, such as counter-point; corresponding to prefered structures). The exact line between these groups is thin and there might even be argument whether the first kind of groups exists. It is, however, quite safe to state that certain structures of the first kind are impossible to perceive, while this does not hold for structures of the second kind. These are exactly the structures that violate the GWFR's. Why should EVAL consider structures that are impossible to perceive?
- In language-related OT, candidates that are likely to exist are considered by EVAL. The theoretical framework requires the set of these candidates to be infinite, but in practice higher-level knowledge is used in OT-research to limit the size of this set. For instance, when looking at a syllabification-problem, we wouldn't bother to epenthesize the whole *Divina Commedia* in between two letters. It is impossible to define where to draw the line and therefore, from a theoretical point of view, infinity of the set of candidates is required. Lerdahl and Jackendoff, however, provide us with the distintion between well-formed and ill-formed structures that can be used by GEN to avoid nonsense-structures to be evaluated.

For these reasons, GEN can be permitted to generate well-formed structures only.
The input to GEN is a list of notes. This list of notes is fed to the procedure `gen/1`, that uses a failure-driven loop to write all well-formed structures to the file "candidates.txt". A failure-driven loop is a use of repeating the same operation over and over again that makes use of the process of backtracking incorporated in Prolog. The loop used here looks like this:

```
gen(String) :-
        not generate(String).
generate(String) :-
        glue(Structure,String),
        write_to_file(Structure),
        fail.
```

What happens is the following: `gen` calls the negation of `generate`. So it asks whether `generate(String)` is NOT true. In `generate`, the predicate `glue` creates a certain structure, based upon `String`. This structure is written to file. Now Prolog encounters a fail and backtracking begins. In backtracking, `glue` is triggered again and asked for a different solution. This new solution is written to file and Prolog reaches the fail again. Again the program starts backtracking and so forth. The process ends when `glue` can't find any other solutions any more. Then `generate` fails at last and therefore `gen` succeeds. The file "candidates.txt" now contains all well-formed grouping-structures with the notes from `String`.

In `glue`, the real process of forming structures takes place. `glue(B,A)` is true when A is a list composed of all sublists of B "glued" together in the original order. Used the other way round it does exactly what we want it to do: creating a list of lists (B) whose elements combine into A.
The complexity of `glue` is of the following form:

$$m = 2^{(n-1)}$$

With m the total number of candidates and n the number of notes in the input-string. Because the number of candidates grows exponentially and all these structures have to be written to file, a problem with memory-size arises when using longer fragments. Therefore, the current model is not suited for input-strings exceeding 10 notes. Since only five-note fragments are used in the experiment this doesn't pose a problem here.

V.5. EVAL

The evaluation of the structures generated in GEN begins with reading these from file. Every structure is tested on every preference rule. For every candidate an ordered vector is made in which the values returned by the rules (1 or 0) are stored. These vectors combine into an OT-tableau. This tableau is written to the file "output.txt". An example of what this tableau looks like:

example input (three notes of 440 Hz, one of 1 sec, one of 2 sec and then again one of 1 sec, all *forte*): `[n(1000,2000,440,f),n(2000,4000,440,f),n(4000,5000,440,f)]`

candidates	violations
3	[0,0,1,0,0,0]
1+2	[1,0,0,0,0,0]
1+1+1	[1,0,0,0,0,0]
2+1	[1,0,0,0,0,0]

Figure 9
Example of grouping-structures and reponse of the implementation

Here we have four structures (corresponding to a three-note input-string, as given by the complexity-equation in section V.4). Because the first structure in "candidates.txt" is processed and its tableau written to the file "output.txt" first too, both files are structurally equivalent. In the first column of the table shown above, every structure is denoted by the number of notes in every group. For example: 1+2 corresponds to a group of one note, followed by a group of two notes. The vectors in the second column give the values returned by the preference rules. The ordering in the vector used here is simply the ordering given by the numbers of the rules by Lerdahl & Jackendoff: [1 2a 2b 3a 3b 3d]. Modelling individual subject behaviour might involve changing this ordering. We will turn to this issue when discussing the results of the experiment in chapter VI.
In this example, we can see how SINGLES (rule 1) is violated by the last three structures. Because we need a string of at least three notes for every other GPR, only the first structure shows a violation of GPR 2b.
The model used here makes use of strict ordering of rules. That gives us a computational advantage. Providing that the ordering in the tableau-vector corresponds to the hierarchy of rules, we can "truncate" the vectors into a number ([0,0,1,0,0,1] becomes 1001, [1,0,0,0,0,0] 100000). This is done by the predicate `trunc/2`. The larger the number, the more important the rules that are violated. So all we have to do is find the smallest number in "output.txt" and select the corresponding structure from "candidates.txt" in order to select the prefered candidate.

At the end of the process, the prefered structure is returned graphically to the user together with the corresponding OT-tableau. In our little example this response would be:

```
Prefered structure:
[ --- ]

OT-tableau:
|   1  2a 2b 3a 3b 3d |
|>       *            |
```

The prefered structure is the one of three notes. The tableau states that only rule 2b is violated, instead of rule 1.

V.6. Conclusion

Before discussing the results of the application a short summary of the above outline is given in the form of a block-diagram of the program. Rectangles with round corners contain file-names or input from prompt, rectangles OT-routines with important predicates named. The diamond contains the decision made by the program.

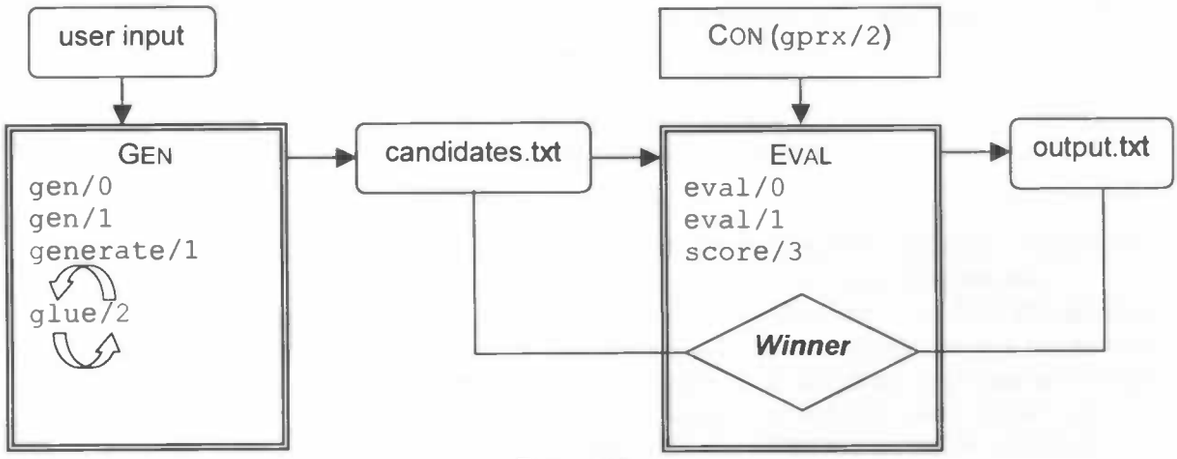


Figure 10
Block-diagram of the implementation

The existence of the parser as described provides evidence that implementing a musical grouping-parser on the basis of Optimality Theory is possible. Implementing a strict hierarchic OT-parser with as many restrictions on GEN as in this case is even a straightforward case. Given the fact that the set of candidates is finite in size, the mechanisms in OT (such as the use of tableaux) are easily translated in computational algorithms.

The implementation in its current form is not as effective as it could be. An implementation described in Hammond (1997) provides examples of ways to improve the performance of the model. It concerns an OT-based syllabification-parser. Hammond places great emphasis on the efficiency of the model as this is one of the main problems when constructing an OT-based parser (to syllabify a word of 7 elements with only 5 constraints, theoretically $1{,}86 \times 10^{10}$ candidates need to be considered when we restrict our scope to the epenthesis of one element only).

First of all, Hammond's parser does not consider violations of the FAITH constraint, which means no epenthesis is considered and the set of candidates is finite in size as with the musical parser. Second, Hammond uses a cyclic application of EVAL. This means that first the effect of the strongest constraint is examined. Candidates violating this constraint will not be processed further. The remaining candidates are judged on their violations of the second-strongest constraint and again violating candidates are discarded, etcetera. Third, the

concept of local coding is considered. Instead of examining all possible *configurations* of structural possibilities (every element might be an onset, a nucleus, a coda or remain unparsed – deletion) every element is judged locally on its structural function, independent from other elements. An example clarifies this procedure. When the possible structures for a word consisting of two elements is generated, all of the structural classifications of elements (an element can be an onset, a nucleus, a coda or remain unparsed) must be considered: {oo, on, oc, ou, no, nn, nc, nu, co, cn, cc, cu, uo, un, uc, uu}. This leads to a set of 4^n candidates, n being the number of elements. But when coded locally, the possible structures are: $\{o,n,c,u\}_1$, $\{o,n,c,u\}_2$. Instead of 4^n we now have 4n candidates, a considerable gain in simplicity.

The musical parser should preferrably preserve the transparency concerning judgments made by the constraints. The information accessible in the file "output.txt" makes it possible to draw conclusions about the constraints a certain candidate does or doesn't violate. When the program would adopt the cyclic use of CON/EVAL, it loses this capacity because violations of lower ranked constraints are not considered when the same candidate already violates stronger constraints. Therefore Hammond's approach is not a prefered method to increase efficiency.

Local coding might improve the prestations of the model, but in order to use it, the constraints as well as the representation of groups should be altered. Instead of focusing on groups the program should focus on structural position of individual notes. The representation of a four-note group should become something like:

`[ n1, n2, n3, n4 ]`	→	`n1/b, n2/m, n3/m, n4/e`
old representation		proposed new representation

The notes in the second grouping are separated from their structural function (b = begin, m = middle and e = end) by a slash. The problem is now to "rewrite" strings of notes like `n/[b,n,e]` (all structural functions possible) into strings of notes like `n/b`. This makes local coding possible as described by Hammond. Because this local coding results in a parser that is defined in terms of boundaries instead of groups themselves, a further improvement might be to make the parser an incremental one. Instead of assuming the complete input to be known, the implementation would gain perceptual credibility when the possible grouping-structures change as more and more information becomes available. This approach is taken in Fanselow (1999). He argues that OT-based parsers are especially suited to deal with incomplete information.

Samenvatting hoofdstuk *Implementation*
In het hoofdstuk over de implementatie van het model als beschreven in hoofdstuk III worden nogmaals de GPR's van Lerdahl & Jackendoff besproken. Hier worden de regels herschreven in Prolog, een op predicatenlogica gebaseerde programmeertaal. Aan GWFR's wordt automatisch voldaan dankzij de manier waarop groeperingen in Prolog worden gerepresenteerd. Ook het OT-proces dat de te beoordelen kandidaten moet produceren (GEN) en het beoordelings-mechanisme zelf (EVAL) worden besproken en de manier waarop ze in het programma vorm zijn gegeven. Tot slot wordt het resulterende programma vergeleken met een eveneens op OT gebaseerd programma voor syllabificering van woorden en worden er op basis van deze vergelijking voorstellen tot verbetering aangedragen.

VI. Experiment

VI.1. Experimental setup

The experiment was conducted using a HTML-application. The advantage of this web-based approach is that subjects can decide for themselves whether they want to continue to a next stimulus. Furthermore the use of audio in HTML is very easy. Before starting the actual experiment, all subjects are presented with two introductory pages. To continue to a new page or a new stimulus, the subject has to click a button with the mouse. The first introductory page explains the overall goals of the experiment. The second introductory page gives an example of a response (again Mozart 40):

Figure 11
Example of response

As can be seen, subjects are asked to group the notes by circles. After the introduction, the subject is presented with 20 recordings of musical phrases, 5 notes in length, together with the same phrases in printed score. The value of 5 stays within the 7 ± 2 items that people can keep in memory (Miller (1956)). Every recording is played twice. The stimulus on paper contains no measures nor indication of time in order to avoid all possible grouping-cues other than the notes. The audio-fragment is presented with a headphone and played at a appropriate level so that all notes can easily be heard.
After all stimuli have been presented, two questions are asked to get an impression of musical experience of every subject:

- How long have you been playing an instrument?
- For how long did you take lessons on that instrument?

In appendix 3 a complete summary of individual results is given.

VI.2. Stimuli

The stimuli used in the experiment are series of notes in MIDI in combination with a written score. MIDI is an audio-format that includes on-time, off-time, pitch, instrument and intensity. The instrument chosen has to have the same intensity and spectrum during the whole note to avoid the note has "died out" before the off-time is reached (e.g. an piano or harp). The MIDI-instrument "ocarino"[9] looks very much like a constant, perfect sine-wave and was therefore used in the stimuli. Each score of a stimulus was printed on a separate paper using Sibelius® music notation software. The pitches of the stimuli are taken from the *Thema Regis* from the *Musical Offering* by Johann Sebastian Bach (1685 – 1750).
The main concern in constructing the stimuli was to provide insights in the hierarchy of constraints. Different strategies can be used to obtain the eventual ordering of constraints given optimality judgments. Tesar and Smolensky (1998) propose an algorithm, based on constraint promotion and demotion, that models the way children are supposed to infer the correct constraint-hierarchy from utterances in their environment. Their approach, however, is more concerned with the acquisition-process itself. In this thesis a basic logical procedure is used that determines the hierarchy based on the hierarchic order of two individual constraints at a time.
In order to establish the hierarchy of constraints, 17 of the 20 stimuli are constructed based on conflicting constraints. When in a certain stimulus two constraints are in conflict, the ordering of rules can be determined depending on which of the candidates is chosen. Let's first consider two abstract cases. The first uses only two constraints: CON 1 and CON 2. On

[9] An ocarino is a small flute made of pottery, originating from Italy.

the basis of the response the ordering of these two constraints can be obtained. The following tableaux illustrates the way this is done.

	CON 1	CON 2
candidate 1	*!	
☞ candidate 2		*

	CON 2	CON 1
☞ candidate 1		*
candidate 2	*!	

Figure 12a & b
Establishing the order of rules from the response

Figure 12 shows how the ordering of CON 1 and 2 can be obtained on basis of the response. In this abstract example, candidate 1 violates CON 1 and candidate 2 violates CON 2. When a subject responds with candidate 2 as is the case in figure 12a, CON 1 is obviously the stronger one because it is not violated when choosing this candidate to be optimal. The resulting order is therefore CON 1 » CON 2. When candidate 1 (figure 12b) is chosen to be optimal, the hierarchy has to be CON 2 » CON 1.
In most cases, more than one constraint at a time is violated by a certain canididate. The next example shows what we can infer from that.

	CON 1	CON 2	CON 3
☞ can 1		*	*
can 2	*!		

	CON 2	CON 3	CON 1
can 1	*	*	
☞ can 2			*

Figure 13a & b
Violating multiple constraints

Candidate 1 in this example violates two constraints: CON 2 and CON 3, and candidate 2 only one: CON 1. In figure 13a, the situation is given where a subject responded with candidate 1. This response is only possible when constraint 1 is stronger than BOTH constraint 2 and constraint 3. We therefore can conlude: CON 1 » CON 2 & CON 3 (the use of the ampersand is a short notation for [CON 1 » CON 2] ∧ [CON 1 » CON 3]). When the response is as in figure 13b, the conclusions are somewhat weaker: now either constraint 2 is stronger than 1 OR constraint 3 is: CON 2, CON 3 » CON 1 (with the comma as shorthand notion for [CON 2 » CON1] ∨ [CON 3 » CON 1]). The mutual ordering of CON 2 and CON 3 can not be derived from this example, therefore no exclamation-mark was placed with one of the two violations and the cells were not shaded.
When we can determine the order of all pairs of constraints, based on their conflict, we can establish the total hierarchy. This is done using a decision-grid. We will turn to this exact procedure when discussing the group-results.
Three stimuli were added where (only) one candidate violates no constraint at all: A, D and U. The constraints that are violated by the other candidates in these stimuli are respectively [1,3a], [1,3b] and [1,2b,3d]. Not all preference rules could be tested in this "1 to 1"-way, because often constraints act on the same structures. When our OT-approach is correct, the candidates that violate no contraints will always be chosen to be optimal, because violating no constraint at all is prefered above violating even the lowest ranked constraint.
One should refer to appendix 2 for all stimuli and corresponding tableaux.

VI.3. Subjects

Given the use of scores in the stimuli, the subjects are required to be able to read notes. More particularly, subjects should be able to follow a given written melody. This requires a certain amount of musical training. This training could interfere with the experiment because when a subject has a very elaborate understanding of groups in music, the judgements given

can't be assumed to be based entirely on perception, but on prior knowledge also. The ideal subjects for this experiment are therefore non-musically trained subjects that are able to read notes.
An attempt has been made to explain the principles of written music to people without musical training. However, after half an hour of training, the reached level of understanding was not sufficient to perform the experiment. Therefore, subjects with intermediate musical experience (no professional musicians) were asked to participate in the experiment.
Probably, a certain amount of meta-knowledge can't be avoided.
No subject reported having problems with hearing.

VI.4. Group-results

The most given responses for every stimulus are collected in the vector **R**. This vector is of length 20, the number of stimuli. We state this resulting vector with percentages of response in bold-faced integer-notation. This notation gives the number of notes in each group, seperated by +. E.g.: "2+3" means the first two notes are grouped, followed by a group of three notes.

A	**2+3**	60%	L	**2+2+1**	60%
B	**2+2+1 / 2+3**	40%	M	**2+3**	60%
C	**2+3**	60%	N	**2+3**	80%
D	**2+3**	70%	O	**4+1 / 3+2**	30%
E	**2+2+1**	60%	P	**4+1**	50%
F	**2+3**	80%	Q	**2+2+1**	30%
G	**2+3**	50%	R	**3+2**	40%
H	**2+3**	70%	S	**2+3**	30%
I	**2+2+1**	60%	T	**3+2**	60%
K	**3+2**	50%	U	**3+2**	30%

Figure 14
Most given responses for each stimulus

When two different responses are given the same number of time, they both appear in the above table, separated by a slash. To give an example: stimulus L is grouped by 60% of the subjects (6 people) as 2+2+1. Stimulus O is grouped by 30% as 4+1 and by 30% as 3+2. Other groupings occur less often. To give some insight in the types of responses given, figure 15 gives all encountered grouping-structures with their frequency of response. Never given responses were: 1+2+2, 3+1+1, 1+3+1, 1+2+1+1 and 1+1+1+2.

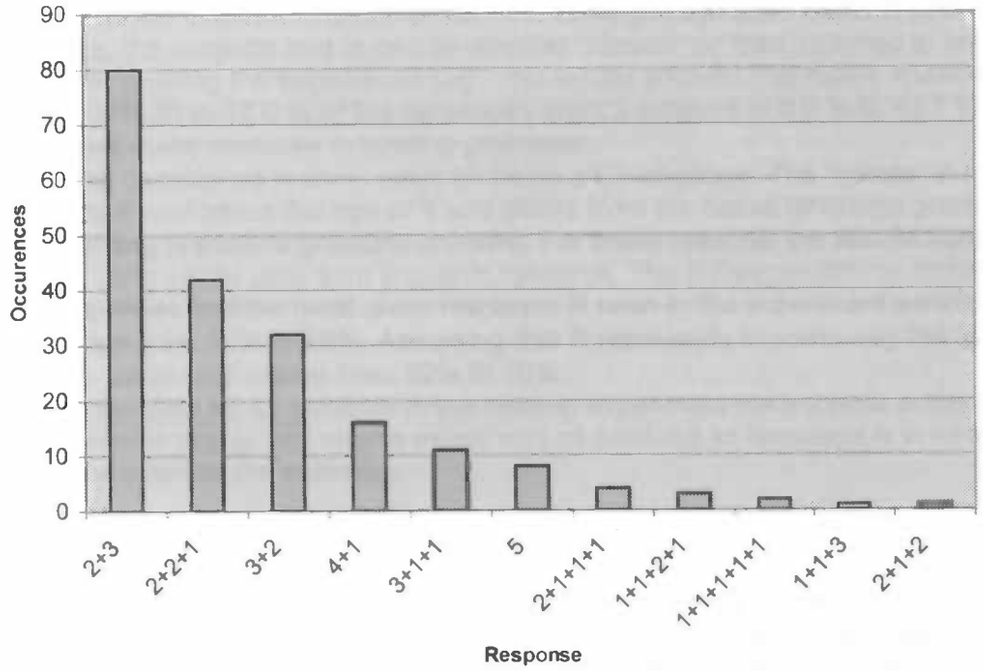


Figure 15
Given responses

VI.5. Discussion group-results

Significance

The significance of the percentages is not easy to determine. Given that we have 16 possible grouping-structures (from the theory of Lerdahl & Jackendoff), the probability for n subjects (out of a total of m) to make the same decision out of 16 different options on basis of chance is given by the following equation:

$$P = (^{1}/_{16})^{n} \cdot (^{15}/_{16})^{(m-n)} \cdot (^{m!}/_{n!\,(m-n)!})$$

This probability[10] drops quickly and is for a score of 50% (m (total number of subjects) = 10, n (number of subjects that make the same decision)= 5) only 0,000174.

In order to shed some light on the percentages as seen in figure 14, the data are compared with linguistic data on binding-experiments reported in Cook (1993). Binding (the principle that a reflexive like "himself" must be bound by an antecendent –e.g. "John"– and can only do so when it is C-commanded by it[11]) is seen as a central aspect of syntactic knowledge. Cook examined the difference in error-rates on binding-problems between 14 native speakers of English and 47 subjects that had English as their second language (L2 subjects, which included 14 Romance, 16 Japanese and 17 Norwegian). Binding is concerned with the possible antecedents for anaphors such as "himself" and pronomials such as "her". In an embedded sentence like:

John said [Steve$_i$ helps himself$_i$].

[10] In the discussion of this probability the bias towards certain responses was not taken in consideration. When dicussing the resulting hierarchy of constraints bias will be considered. However, the effect (and even the mere existence) of this bias can't be determined without further research.
[11] See for further discussion Haegeman (1994).

the anaphor "himself" must be bound by Steve, as indicated by the subscript i. The experiment involved different kinds of sentences, varying in syntactic kinds of binding. For each sentence, the subjects had to decide whether "himself" or "him" referred to one out of two persons by pushing the appropriate key. The results showed that native speakers made mistakes in 1,8 % (!) to 16,0 % of the sentences and L2 subjects in 0,0 % to 43,7 %. So even native speakers make mistakes in binding-problems.
Music might be considered in some ways as being a L2 language. The "syntax" in most cases is learned from about the age of 5 and differs from the native language grammar. The problem of binding is partly a grouping-problem. For these reasons, the results from Cook can serve as comparable data from linguistic research. The correspondences between individual responses and the most given response **R** seen in the experiment performed in this thesis range from 30% to 80%. Assuming that **R** represents in some way the "correct" response, the "error-rate" varies from 20% to 70%.
Compared to the data for L2 subjects in the binding-experiment the subjects in this experiment behave poorly. But maybe music isn't as hard-cut as language is in some respects[12]. For example the sentence:

*John$_i$ said [Steve helps himself$_i$].

is just ungrammatical, because "himself" can only refer to "Steve". However, as much as 9,9% of the native speakers and (on average) 15% of the L2 subjects responded "John" as being bound by the anaphor. So even in cases where most people overtly agree on the presence and validity of the rules a significant amount of mistakes is made. It can therefore be concluded that an agreement of half of the subjects on a certain grouping-structure is a significant judgement.

Ranking of constraints
The results from figure 14 are not without ambiguity. As an example of the encountered contradictions stimuli Q and R are examined here. As can been seen from the tableau in figure 16a below, we must conclude from stimulus Q that rule 3d (CHANGE LENGTH) is stronger than 1 (SINGLES). But figure 16b shows the most given response on stimulus R, which states exactly the opposite.

	stimulus Q (30%)	3d CHANGE LENGTH	1 SINGLES
	3+2	*!	
☞	2+2+1		*

	stimulus R (40%)	1 SINGLES	3d CHANGE LENGTH
☞	3+2		*
	2+2+1	*!	

Figure 16a & b
SINGLES versus CHANGE LENGTH

To avoid this problem, the responses given by more than 50% of the subjects (in order of percentage: F, N, H, C, E, I, L, M, T), that is on which more than half of the subjects agree, are first considered. Importantly, none of these stimuli are contradictory. Because in most cases different stimuli give the same information, stimuli H, C, L and M provide no new information that can't be concluded from the other more given responses. Therefore only stimuli F, N, E, I and T are processed below.
In order to obtain the hierarchy from the conflicts a decision grid is used. This is a 6 x 6 matrix: columns represent hierarchic places constraints (given in rows) can occupy:

[12] In *L'obvie et l'obtus*, Roland Barthes puts it this way: "...language [...] is the order of the general, and music [...] is the order of difference". (Taken from Scher (1992))

1	1	1	1	1	1
2a	2a	2a	2a	2a	2a
2b	2b	2b	2b	2b	2b
3a	3a	3a	3a	3a	3a
3b	3b	3b	3b	3b	3b
3d	3d	3d	3d	3d	3d

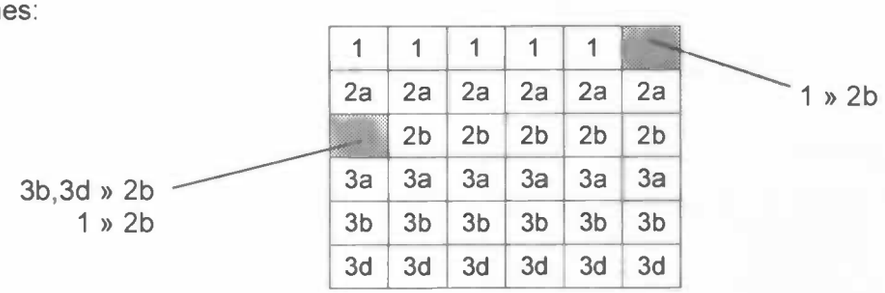
Figure 17
Decision grid

On the basis of the order of two individual constraints positions in the grid become impossible. For example: stimulus F is grouped by 80% of the subjects as 2+3. The tableau below shows what conclusions can be drawn:

stimulus F (80%)	1 SINGLES	3d CHANGE LENGTH	3b CHANGE DYNAMICS	2b PROXIMITY ATTACKPOINTS
2+3				*
3+2		*	*	
2+2+1	*!			

Figure 18

The first two candidates lead to the conclusion 3d, 3b » 2b. We can't conclude which of the two is stronger: 3d or 3b, therefore no exclamation-mark is placed. What can be concluded is that 2b can never be the strongest constraint. Therefore, the constraint is erased from the first column in the decision grid. Furthermore, we can conclude from this stimulus that 1» 2b. This means constraint 1 can't be in the lowest hierarchic position. The decision grid now becomes:



1	1	1	1	1	
2a	2a	2a	2a	2a	2a
	2b	2b	2b	2b	2b
3a	3a	3a	3a	3a	3a
3b	3b	3b	3b	3b	3b
3d	3d	3d	3d	3d	3d

Figure 19
Decision grid after processing stimulus F

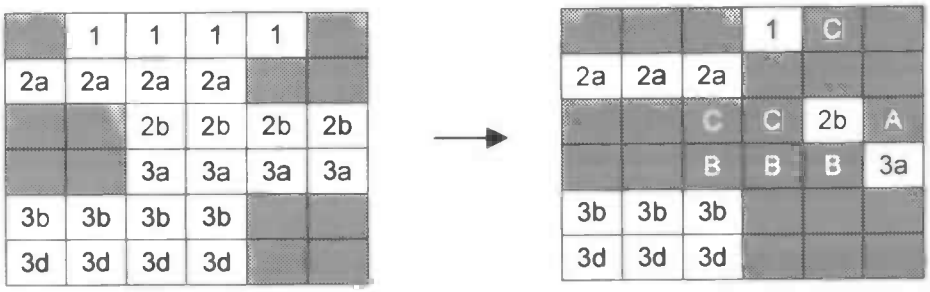
Consequent processing of stimuli N, E, I and T give the following decision grids:

1	1	1	1	1	
2a	2a	2a	2a	2a	
	2b	2b	2b	2b	2b
	3a	3a	3a	3a	3a
3b	3b	3b	3b	3b	3b
3d	3d	3d	3d	3d	3d

Stimulus N (80%): 2a » 2b, 1 » 2b & 3a

	1	1	1	1	
2a	2a	2a	2a		
		2b	2b	2b	2b
		3a	3a	3a	3a
3b	3b	3b	3b		
3d	3d	3d	3d	3d	3d

Stimulus E (60%): 2a » 1, 3b » 1
(keep 1 » 2b & 3a from stimulus N in mind)



Stimulus I (60%): 3d » 1

Stimulus T (60%): 2b » 3a

Figure 19 (both pages)
Decision grids for stimuli N, E, I and T

The last step, in which the hierarchy of 1, 2b and 3a is settled, seems to be a typical case of "...and then a miracle occurs" and should be explained in some detail. The information given by stimulus T leads to the conclusion that 2b can't occupy the lowest place in the hierarchy, because it overrules 3a, so the cell marked "A" becomes gray. Because constraints 2b and 3a were the only two ones left in this hierarchic place, we must conclude that 3a is the weakest constraint. We can therefore blank out the rest of the 3a-row (cells B). With the knowledge from stimulus F (1 » 2b) we now can infer that 2b occupies the 5th place (cells C become gray). We already knew that 2a, 3b and 3d all overruled 1, leading to the last decision grid in its final form. The resulting hierarchy is (brackets indicate that the ordering of these constraints could not be established from the taken responses):

{ 2a - PROXIMITY SLUR/REST)
3b - CHANGE DYNAMICS
3d -CHANGE LENGTH }
» 1 - SINGLES
» 2b - PROXIMITY ATTACKPOINTS
» 3a - CHANGE REGISTER

Stimulus S (30% report a 2+3 grouping-structure to be optimal) provides a bit of evidence that 3d is stronger than 3b:

Stimulus S (30%)	3d CHANGE LENGTH	3b CHANGE DYNAMICS	3a CHANGE REGISTER
2+3		*	
2+2+1	*!		*

Figure 20
CHANGE LENGTH versus CHANGE DYNAMICS

This would mean that either 3a or 3d is stronger than 3b or both. But from the obtained hierarchy it can be concluded that both 3d and 3b are stronger than 3a, therefore the exclamationmark is placed with the violation of rule 3d. Given the fact that in no response from **R** 2a is violated and both 3b and 3d are (but never in order to satisfy 2a), we assume 2a to be the top of the hierarchy. This results in the following hierarchy of constraints:

2a - PROXIMITY SLUR/REST
» 3d - CHANGE LENGTH
» 3b - CHANGE DYNAMICS
» 1 - SINGLES
» 2b - PROXIMITY ATTACKPOINTS
» 3a - CHANGE REGISTER

The resulting hierarchy has some unexpected features. Lerdahl and Jackendoff give no ordering of rules, but their order of introducing them is not entirely coincidental. The first rule, SINGLES, seems to be a tautological one; the term "group" almost always refers to more than one element. However, in the experiment SINGLES is frequently violated and ends as low as the 4th position. This might be due to bias against reproducing the same response. The only two candidates that obey constraint 1 are 2+3 and 3+2. Subjects are reported to refuse to give the same response over and over again. Still, as can be seen from figure 15, 2+3 and 3+2 together are responded more than half of the time (112 out of 200).
The top-position is held by rule 2a. This is not really a surprise, because a rest (as seen in the discussion of the Gestalt-examples in section III.1) is a "heavy" cue for beginning a new group. When looking at the original responses, a reformulation of 2a might strengthen its position even more. In short, 2a is now "Reject groups containing a sequence of three notes with a rest between the second and the third one". Considering a formulation in terms of group-boundaries instead of groups ("Prefer a boundary on the position of a rest") might prove to be more efficient in explaining the obtained data, but further research need to be done to establish this supposition.
The third conspicuous aspect of the hierarchy is the low position of 3a. At first sight, change of pitch is an important indication that a new group should be started. At least important enough to reach a somewhat higher position in the hierarchy. Apparently this does not show from the data. A reason for this might be that the stimuli are constructed, instead of being existing musical fragments. Melodic pattern is not as full-fledged as it is in "real" music. More elaborate passages might give a different ranking of this rule in particular, because it is the only one concerned with melodic structure.

One-to-one rules
The results for the stimuli A,D and U (in which one of the candidates violates no constraints) are given here for all subjects (shaded regions give the "optimal result" as predicted by our model):

	ÀB	CB	DB	DV	HS	JK	JM	KB	SH	TB
A	2+3	2+1+1+1	2+3	2+3	2+3	2+3	1+1+1+1+1	2+3	5	5
D	2+3	2+3	2+3	5	2+2+1	2+3	2+2+1	2+3	2+3	2+3
U	3+2	3+1+1	2+3	4+1	2+1+1+1	3+2	3+1+1	5	2+3	3+2

Figure 21
Responses stimuli A, D & U for all 10 subjects

This is, 15 out of 30 responses correspond with the prediction that candidate 2+3 will be considered to be optimal. A and D behave best, stimulus U breaks with the principles of the OT-based model. However, in this stimulus the "correct" response is not agreed upon, as can be seen by the different answers given.

VI.6. Results inter-subject differences
All subjects said the experiment was difficult. Since the stimuli were created on basis of conflicts, this was to be expected. To provide a measure of inter-subject difference (ISD), the

closeness of two vectors **a** and **b**, both of length *m*, is defined as being (vectors are always given in bold-face):

$$\mathbf{C}(\mathbf{a}, \mathbf{b}) \equiv \frac{\sum_{i}^{m} \delta(a_i, b_i)}{m}$$

with the altered delta-function given by:

$$\delta(x, y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$$

The equality in the delta-function can also be used with symbolic vectors (for instance, the vectors [a, apple, allegro] and [forte, sharp, allegro] would have a closeness of (0+0+1)/3). We can now use this measure for determining the closeness of individual responses (denoted by **r**) to the most given response (**R**): C(**r**,**R**), called the *closeness of response*. The results for all 10 subject of their closeness of response is given below:

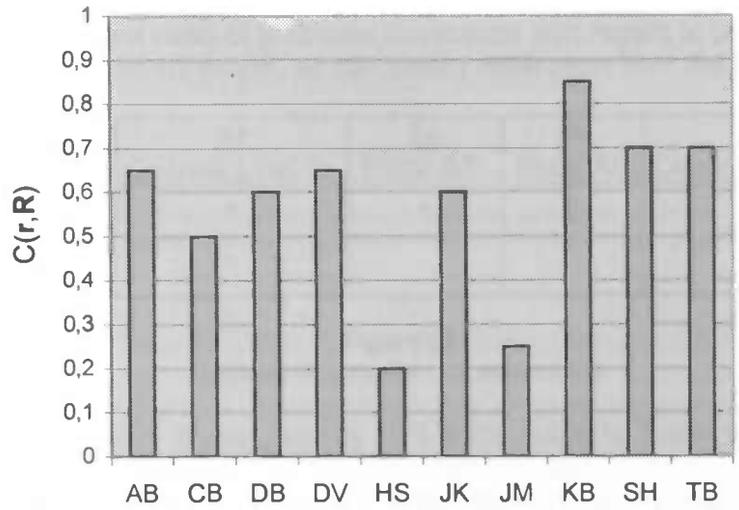


Figure 22
Closeness for the subjects to most given response **R**

The figure above gives a summary of the responses given by all subjects: the heigth of the staves correspond to the degree of which the individual response **r** and the most given response **R** are alike. The found average for the closeness of all tested subjects is 0,57 ± 0,20. This average provides a measure for ISD. When all subjects would have given the same response (no ISD), the value would be 1, when all subjects would have given a different response (maximum ISD) this would be 0,05 (because for each value of *m* at leats one subject would have one correspondence with **R**).

VI.7. Discussion inter-subject differences

One might suspect a relation between the closeness given in figure 22 and musical experience of subjects, but there is not enough evidence to support this hypothesis. When the ISD could be explained without altering the preference rules but only by changing the hierarchy that would not only provide evidence for the correctness of these constraints but also for our OT-based approach. Two subjects will be dicussed in detail here: one with intermediate closeness (DB) and one with minimum closeness to the most given response

(HS). In discussing the results for these subjects different issues will be given attention (problematic responses, use of written score, training, etc.).

*Subject DB (C(**r,R**) = 0,6)*
Two problematic responses, the ones on stimuli B and I, are examples of the two kinds of problems that our model encounters: contradictions in hierarchic structure (one response gives rise to an ordering CON 1 » CON 2, another to CON 2 » CON 1) and objections with regard to processing (e.g. favouring a candidate that violates a constraint above one that violates no constraint at all).
Stimulus B (DB responded 2+3, as did 3 more subjects) is of the first kind: it gives 3a » 2a, which is in contradiction with (for example) the combination of stimuli N (2a » 2b) and T (2b » 3a). The same could be seen when considering stimulus Q and R for the most given response. In the discussion of the group-results, frequency of response could be used to solve conflicts between contradicting outcomes. However, when considering individual responses only the number of times a conflict occurs can be used. For example, when two stimuli give the hierarchy CON 1 » CON 2 and only one gives CON 2 » CON 1, the outcome becomes CON 1 » CON 2. Processing the data of subject DB the same way as described in section VI.5 with this extra heuristic leads to the hierarchy 1 » 3d » 3b » 2a » 2b » 3a. Apart from responses B and I there were no problems with the decision grid. This can't be said for all subjects.
Stimulus I is of the second class of problems (objections with regard to processing) and can't be explained given the current model, as can clearly been seen from the tableau below:

	Stimulus I	3d CHANGE LENGTH	2a PROX.S/R	2b PROX.AP	3a CHANGE REGISTER
☞?	2+3	*	*	*	*
	3+2	*			

Figure 23
Tableau of stimulus I for subject DB

This particular response (2+3) was given by 40% of the subjects. Subject DB has given only two of these problematic responses so for this subject we can't conclude that the OT-approach is incapable of describing the data or that the rules are incomplete or wrong. For all subjects the hierarchies of constraints were obtained the same way, sometimes with less ease. These are given in Appendix 3. The difficulties with obtaining the resulting hierarchy shows as groups of constraints of which the mutual ordering couldn't be deducted, given between brackets. An example can be seen when discussing the results for subject HS. The hierarchies are used in the implemented model, facilitating interpreting the results. Section VII.1 returns to this issue.

*Subject HS (C(**r,R**) = 0,2)*
The results for subject HS are often problematic for our OT-approach. The resulting hierarchy of constraints is : { 2a, 2b, 3a, 3b } » 3d » 1. The ordering of 2a, 2b, 3a and 3d could not be determined. Problems of the second kind are even graver. An example is seen in stimulus O:

	Stimulus O	3d CHANGE LENGTH	1 SINGLES
	3+2	*	
☞?	3+1+1	*	*

Figure 24
Tableau of stimulus O for subject HS

Here can clearly been seen that the proposed model fails. The first candidate violates one constraint only. The second violates the same constraint and one more (SINGLES) but is still chosen. Further responses that can't be explained given our model for the same reason are given on stimuli D, K, Q, R, T and U.
Subject HS reported having difficulties focusing on the auditive signal instead of grouping the music on basis of the printed score. This might be an indication of interference with meta-knowledge. Another indication for meta-knowledge might be the highly complex way of grouping seen with this subject. An example of a response (stimulus L) is given below.

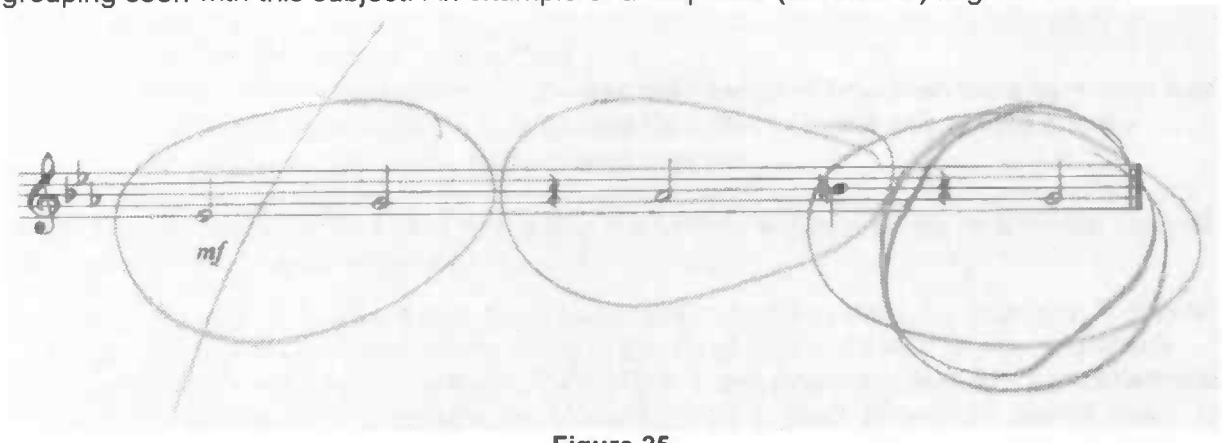


Figure 25
Example of response for subject HS

In spite of being asked to use circles only, the subject prefers to mark a boundary by means of a slash between the first and the second note apart from the use of circles elsewhere. Clearly, the subject had difficulty with grouping the last note (aside from this, all subjects reported stimulus L to be particularly difficult).
Something that should be mentioned here is the use of a written score. This score could have been used by subjects to provide extra information and therefore introduces a bias. In the discussion of rule 2b in section III.4 the use of groups of three notes in the definition of the rules instead of four was explained. While listening to music and indentifying groups the fourth note had not been heard yet and therefore this note can not play a role in the decision. In the experiment, however, subjects had access to the whole stimulus because of the written score. Subject HS reported having difficulties not to base the decision on basis of the score instead of the auditive signal (as was asked). This difficulties might explain the erratic behaviour. An alternative experimental setup without the stimuli written out on paper might give different results, more based upon perception. Subjects might be asked, for instance, to press a button when a boundary is detected. This approach, however, would involve a totally different experimental setup and the results are less easy obtained.
An important factor that might interfere with the results of this experiment is musical experience. It was to be expected that the results would depend in some degree on the amount of years a subject had played an instrument. However, because all subjects had more than 10 years of musical experience, the differences were evened. Probably effects will occur between subjects with less than 5 years of musical training and more than 10 years. The effect of experience can not be determined in this experiment. Also, the musical experience of subject HS did not deviate from the rest: the subject played an instrument for 15 years (average over all subjects: 16,7 ± 3,9 years) and took lessons for 12 years (average over all subjects: 10,5 ± 2,7 years).
The outcomes for this particular subject brings up the question whether Optimality Theory in combination with GTTM provide a good model. The conclusion will now turn to this question.

VI.8. Conclusion

It is hard to determine whether performance of the proposed model depends on the validity of the constraints, their hierarchy or on the basic assumptions of the model, such as strict

ordering. We will again turn to this problem in the next chapter. As was seen when discussing the position of constraint 2a in the hierarchy, a reformulation of the rules might be needed. When listening to music, every note should be judged as to whether it belongs to the former group or whether it is the first of a new one. Instead of focussing on groups, it might therefore be better to look at the ways boundaries are recognized. This is consistent with the idea of local coding as seen in section V.6: for every note a classification is made whether the note is at the beginning, in the middel or at the end of a group. An example of a reformulation of rule 2a (also incorporating local coding) is:

GPR 2a (PROXIMITY SLUR/REST) – *modified*

> When in a contiguous group of notes the interval of time from the end of note n to the beginning of note (n+1) is greater than that from the end of note n to the beginning of note (n–1), appoint note n as n/e.

Recall that the notation n/e means that n is in the endposition of a group. In a similar way, all constraints could be reformulated.

A certain amount of problems aside, there is no reason to conclude that Optimality Theory is incapable of describing the processes found in grouping music. As little as five responses (F,N, E, I and T) shared by (on average) 68% of the subjects already lead to a consistent and almost complete hierarchy of constraints. Modelling inter-subject differences seems less succesfull because a considerable amount of data is ambiguous. The assumption that this model is more suited for explanation of group-results leads to the conclusion that further research with a larger pool of subjects needs to be done to establish this hypothesis. Further conclusions:

- A flexible, individual process like grouping in music is hard to describe with strict rules. Optimality Theory provides a flexible yet consistent framework to model this process.
- Contradictions (both in hierarchic structure and in processing) can probably be solved by changing the constraints instead of the framework itself.
- Especially for the group-result a consistent hierarchy appeared, proving that the experimental results can lead to an OT-based model.

When the implemented model is compared with the results of the experiment, more explicit details about the correctness of the model are revealed. In the next chapter, the comparison of the implementation with the experimental data will provide further evidence for the OT-based approach.

Samenvatting hoofdstuk *Experiment*
In dit hoofdstuk wordt het experiment beschreven dat uitgevoerd is om te bepalen in hoeverre de combinatie van OT en GTTM een vruchtbare is. Aan een groep van 10 proefpersonen worden muziekfragmenten voorgelegd die zij moeten groeperen. Uit de gegeven antwoorden wordt bepaald welke hierarchie van regels de proefpersonen gebruikt lijken te hebben. De gemiddelde overeenkomst van de antwoorden van onderlinge proefpersonen met de meest gegeven antwoorden bedraagt (op een schaal van 0,05 tot 1) 0,57 ± 0,20. Er bestaan dus grote verschillen tussen proefpersonen onderling.
De vraag is of deze grote variatie te wijten is aan de architectuur van ons model of aan de regels waarvan het gebruik maakt. Er is geen reden aan te nemen dat de op OT gebaseerde aanpak niet werkt, maar een herdefiniëring van de regels in termen van grenzen tussen groepen in plaats van groepen zelf zou een oplossing kunnen zijn. Verder onderzoek, eventueel gebruik makend van een andere experimentele opzet, zou veel dingen duidelijker kunnen maken.

VII. Comparison of results

VII.1. Predictions of the implementation

In order to compare the results of the experiment with the computational model, both the group-result and each individual subject is being modelled in the OT-framework. The parameter to be set is the hierarchy of constraints. The hierarchy for the group-result obtained from the experiment (2a » 3d » 3b » 1 » 2b » 3) is given to the model. Implementing this order of constraints results in a model **M** with a closeness of response to **R** of C(**M**,**R**) = 0,7. This closeness is well above the average of 0,57 found in the experiment. The value competes with the score for closeness of individual subjects.

In order to give some insight in the overall performance of the model, the *hierarchic robustness* of constraints will be considered. The hierarchic robustness of a constraint is defined as the dependance of the models' performance on the hierarchic position this particular constraint occupies. The following figure is obtained from varying hierarchic place of only one constraint from the place highest in hierarchy (position 1) to the place lowest in hierarchy (position 6), keeping the rest of the hierarchy intact and calculating the closeness of the resulting model to the vector **R**. This provides a measure of the hierarchic robustness for each constraint.

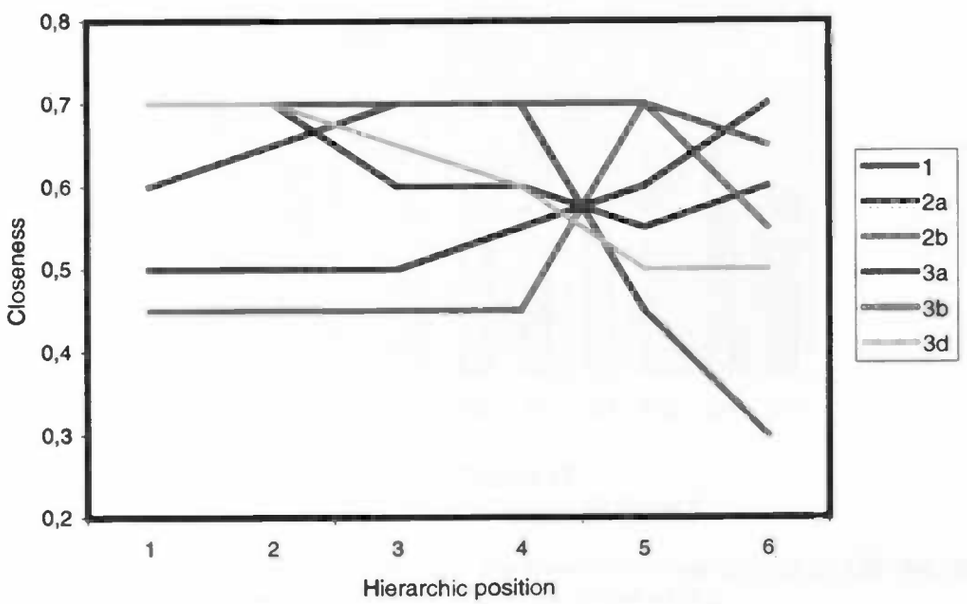


Figure 26
Hierarchic robustness of constraints

The height of the lines in this diagram correspond to resulting closeness of response of the model **M** (that follows from the order with the corresponding constraint in the given position in the hierarchy) to the response being modelled: **R** (most given response).

This figure provides insights in the significance of the rules and therefore the quality of the model. A constraint with high hierarchic robustness shows as a sharp peak in the figure, as can be seen in some degree with rule 2b (PROXIMITY ATTACKPOINTS). The performance of the model is highly dependent on this constraint occupying the 5th position in the hierarchy. Hierarchic robustness for constraint 3b (CHANGE DYNAMICS) is very small. The model behaves the same for all positions occupied by 3b but the last.

From this graph not only robustness shows, but both the hierarchy of the constraints and the conflicts seen in this ordering are easily deducted. For the last place in the hierarchy, all constraints drop (1 (SINGLES) even dramaticly), except for constraint 3a (CHANGE REGISTER). Therefore, 3a should occupy the last place in the hierarchy. Following the curve of closeness

of rule 2b, the only logical place is the second last, etcetera. It is no accident that the hierarchy of the three most hierarchically robust constraints (1, 2b and 3a) was easily determined from the most given responses, while the other constraints needed less secure information in order to obtain the total hierarchy (section VI.5).
The overall performance of the model can be deducted from the shape of this diagram. A perfect model (all the constraints are maximally selective and the hierarchy is the only one possible) would show a sharp peak from 0,05 (minimal closeness) to 1 (maximal closeness) for every rule on a different hierarchic place. The more "flat" this diagram is, the less selective the constraints and the therefore less exact the model.

For the modelling of individual results in order to explain inter-subject differences we can again use closeness. Processing the data of all subjects provides a hierarchy of constraints for each subject. This hierarchy is again given to the implementation, resulting in an individual model, **m**. The closeness of this model to the actual response for each subject (**r**) can be calculated to provide a measure of accuracy: C(**r**,**m**). As opposed to closeness of response this measure is called *closeness of model*. In figure 27a, the closeness of model is given for each subject.

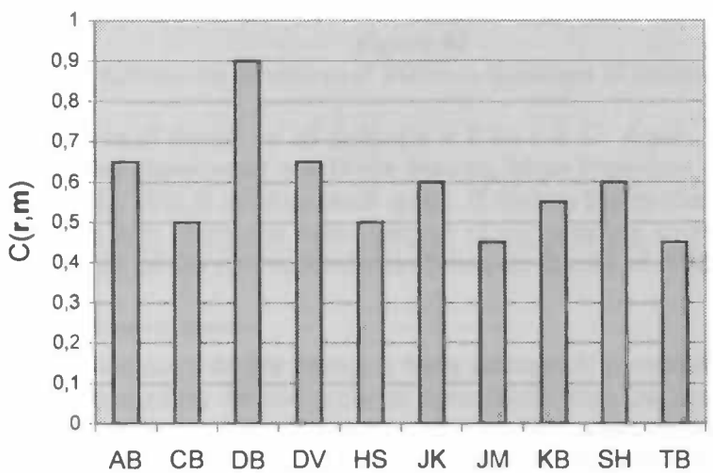


Figure 27
Closeness of model for all subjects.

So for example the model that results from the experiment performed by DB results in a model that has a closeness to the actual response of DB of 0,9.
Figure 28 summarizes both figures 22 and 27: the closeness of model (on X-axis) is given against the closeness of response (on Y-axis). The open circle in this figure corresponds to the model for the most given response. Seen as a "new" subject, it of course has a value of 1 for closeness of model (C(**M**,**M**) = 1, C(**M**,**R**) = 0,7).

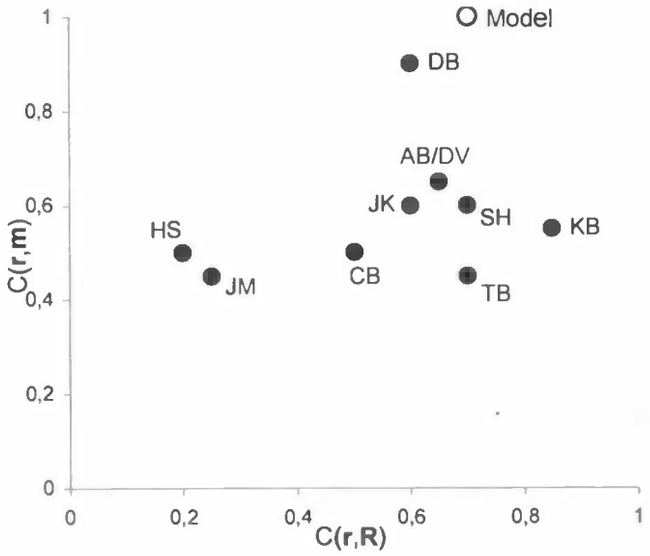


Figure 28
Dependence closeness of model to closeness of response.

The average closeness of model for all subjects is 0,59 ± 0,13. Again, no correlation can be found between musical experience and these figures. More important, there is no clear dependance on C(**r**,**R**). This is an important result. It means the model is capable of modelling all subjects with about the same degree of success (as shows from the fairly small deviation), independent of the correspondence between results of different subjects.

VII.2. Conclusions group-results

The last section pointed out that the model is fairly succesfull in modelling group-results. The problems seen with deducting the hierarchy of constraints from the results of the experiment are predicted by the analysis of hierarchic robustness. This analysis depends on the model, because without a computational model the closeness for different positional hierarchies could never be calculated.
The resulting model for the group-result did not predict the results with complete accuracy, but the responses generated were close enough to the most given response **R** to compete with the different subjects. Subjects SH and TB scored the same closeness of response and only KB exceeds this value (seen in figure 22). Because the vector **R** is averaged over all subjects, this is clearly a good result: even being based on the results from the subjects themselves, the model performs well above their average closeness.

VII.3. Conclusions ISD

The treshold for success when modelling inter-subject differences lies somewhat higher compared to group-results. In modelling the group-result, concluding that the model behaves comparable with the subjects could suffice. When explaining (or predicting) ISD the model must keep closely track with individual results. As can be seen from figure 27a, only the results for subject DB gives a substantial closeness of response.
An argument in favour of the capability of modelling inter-subject difference is that closeness of model is not dependent on closeness of response. This means that subjects that behave "out of the ordinary" and subjects very close to the general agreement of response are equally close modelled (e.g. HS and KB, respectively). This means that the architecture of the model might be correct, but the rules need to be adapted. When the architecture itself (OT-based violability of strict ranked constraints) would be wrong, the results for modelling ISD would show more deviation. Alteration and/or adding of constraints might increase the

performance of the model. For instance the addition of a rule that is ranked in a low position by subject DB (because this subject is adequately modelled), but somewhat higher by other subjects might solve the problem. In order to establish this supposition, further research need to be done.

Samenvatting hoofdstuk *Comparison of results*
In dit hoofdstuk wordt geprobeerd de resultaten uit het experiment te modelleren aan de hand van het ontwikkelde computationele model. Het model gebaseerd op het groeps-resultaat scoort boven het gemiddelde van de proefpersonen. Om de individuele verschillen tussen proefpersonen te verklaren is het model minder geschikt, maar dat is nog geen reden om de opzet van het model te veranderen: alleen een aanpassing van de regels kan in principe voldoende zijn.

VIII. Conclusions

VIII.1. Grouping in music

In this thesis, the process of grouping in music has been considered in great detail. The framework as sketched in a *Generative Theory of Tonal Music* by Lerdahl and Jackendoff formed the theoretical background. This theory leans on a set of rules, from which three were taken and modified into soft constraints. Optimality Theory, introduced by Prince and Smolensky, provided this theory with an elaborate toolkit that made it possible to model the grouping-process in a coherent way. Implementing the theory into a working model proves that such a model exists and testing the model (or at least corner-stones of its architecture) provided experimental evidence of its validity. Not all aspects of the model survived the experiment without a scratch. Certain limitations were placed on its application and during the process of implementing and testing the model several suggestions for improvement were formulated, not only for practical implementation, but on some of the theoretical topics, too. Further development and research might result in a succesfull theory.

The resulting implementation is a consistent one and Optimality Theory is suited to form the basis of this computational model. Especially in Prolog, with its built-in backtracking procedure, most OT-processes are implemented in a simple and comprehensible way. The experiment showed the limitations of the model in its current form more clearly. In order to become a complete and relevant theory, some work has to be done. An approach focussing on the boundaries between groups instead of on groups themselves gained credibility from both the implementation and the experiment. However, there is no reason to abandon the main assumptions of the theory: 1) parsing a musical surface is not a coincidental process and is governed by constraints, 2) these constraints can be violated and 3) are ordered in a strict hierarchy.

One of the most promising improvements is local processing: instead of looking at whole groups, the model should be restricted to identifying boundaries between groups. Evidence to support this come from computational considerations (local coding means a considerable gain of efficiency), conclusions from the experiment (the strenght of 2a is explained when it is reformulated in terms of boundaries) and parallels with similar research in linguistics (e.g. Müller (2002)).

VIII.2. The relation to language

The success of a theory from linguistics in musicology supports the assumption that both fields of research are comparable. Throughout this thesis, conclusions were compared with linguistic issues. This provided backgrounds and new insights that were of great use in developing the model. The question arises how the process of grouping in music relates to similar processes in language. And –more speculative– what other subjects are comparable?

As was mentioned in chapter III, the model focusses on first-level-grouping only. Syllabification is similar with respect to this approach. This process describes how boundaries are marked between groups of letters. In some ways, however, musical grouping might be closer to syntactic parsing. The complete theory of Lerdahl & Jackendoff incorporates tree-structures that have many similarities with syntactic X-bar theory. The function of constituents in a syntactic analysis depends on their meaning. E.g.: verbs have different syntactic properties from nouns. "Meaning" in the linguistic sense is not present in music. This is an important difference.

The degree of variation might be considered as another difference between language and music. The experiment showed great inter-subject variation as opposed to what one might expect in language. However, the results from Cook (1993) show that inter-subject variation in language can not be neglected either. In vernacular style (speech in ordinary conversational context instead of an experimental setup as in the Cook-experiment) this variation will probably increase even further.

VIII.3. Two faces of the same problem?

The application of a linguistic theory to music is not new. Bedřich Smetana (1824 – 1884), for instance, a reknown Czech composer, developed a style of composing for vocal music based upon his own research of prosodic speech-patterns in Czech. Most of his latter vocal works and opera's are composed in this style. In most cases, however, the cross-pollination goes the other way, especially in scientific matters. A good example is Gilbers & Schreuder (2002) on prosodic variability. They use GTTM in order to provide explanations for the variation of stress-patterns with tempo of speech. Their research leads them to the conclusion that all temporally ordered behaviour (speech, music, even dance) are governed by the same structures and principles. The research in this thesis provides more evidence to support this hypothesis.

Music and language are different facets of human behaviour. Language is primarily concerned with communication, music with expression. But the means by which the two achieve their goals is essentially the same: high-structured sound (when confined to played – or sung – music and speech). In order to "understand" this sound, listeners of both music and language are faced with the same problem: uncovering the underlying structure. This thesis shows that this core-process shared by both phenomena might be modelled using similar techniques.

Because music is being thought of as a highly individual, unpredictable form of art and a chaotic field of research, scientists seem to be afraid to treat music in the same "hard" way as they treat linguistic issues. Therefore, musicology often restricts itself to descriptions. This thesis shows that a sound constraint-based theory from the field of linguistics can be used to describe a musical process. At the other hand, it also points out that some of our ideas about language aren't as hard-cut as they seem to be. Variation in linguistic experiments shows that the individual language isn't always the same as the language that is generally agreed upon. Maybe music is more than art and linguistics more than science.

Samenvatting hoofdstuk *Conclusions*

Het laatste hoofdstuk keert terug bij de onderzoeksvragen:

1) Welke processen liggen ten grondslag aan groeperings-processen in muziek?

Op basis van de resultaten kan geconcludeerd worden, dat OT in combinatie met GTTM een veelbelovende basis vormt voor een theorie over groepen in muziek. Met enkele wijzigingen, vooral door het vormen van groepen meer lokaal te bekijken in plaats van uitspraken te doen over complete groepen zou het model te verbeteren zijn.

2) In hoeverre zijn muziek en taal vergelijkbaar?

Vaak worden de "harde" taalkunde en de "zachte" musicologie strikt gescheiden gehouden, ondanks het medium dat ze delen: gestructureerd geluid. De taalkundige optimaliteits theorie werd met redelijk succes op een muzikaal onderwerp toegepast. Dit geeft aan dat een op regels gebaseerd systeem wel degelijk gebruikt kan worden om over een artistiek fenomeen als muziek uitspraken te doen. Aan de andere kant zijn lang niet alle bevindingen binnen de taalkunde zo universeel als ze pretenderen, zoals blijkt uit verschillen in taalgebruik tussen individuen. Muziek is wellicht meer dan alleen kunst en taalkunde meer dan alleen een wetenschap.

IX. Literature

IX.1. Literature

Anderson, John R. *Learning and memory: an integrated approach*. John Wiley & Sons, 1995

Apostol, Tom M. *Calculus, Multi-Variable Calculus and Linear Algebra, with Applications to Differential Equations and Probability*. John Wiley & Sons, 1969

Archangeli, Diana & Langendoen, D. Terence, ed. *Optimality Theory. An Overview*. Blackwell Publishers,1997

Barlow, Roger J. *Statistics. A guide to the Use of Statistical Mathods in the Physical Sciences*. John Wiley & Sons, 1989

Barrow, John D. *The Artful Universe*. Oxford University Press, 1995

Bratko, Ivan. *Prolog programming for Artificial Intelligence*. Addison-Wesley Publishers Ltd., 1990

Bregman, A.S. *Auditory Scene Analysis: The Perceptual Organization of Sound*. The MIT Press, 1990

Cook, Vivian. *Linguistics and Second Language Acquisition*. The Macmillian Press Ltd., 1993

Dürr, Walter & Kobayashi, Yoshitake. *Bach-Werke-Verzeihnis. Kleine Ausgabe*. Breitkopf & Härtel, 1998

Fanselow, Gisbert, et. al. *Optimal parsing: Syntactic parsing preferences and Optimality Theory*. ROA, 1999

Gilbers, Dicky & Schreuder, Maartje. *Language and Music in Optimality Theory*. Unpublished manuscript, University of Groningen, 2002.

Haegeman, Liliane. *Introduction to government and binding theory*. Blackwell Publishers, 1994

Hammond, Michael. *Parsing syllables: modeling OT computationally*. ROA, 1997

Howell, Peter & Cross, Ian & West, Robert (ed.). *Musical Structure and Cognition*. Academic Press, 1985

Lerdahl, Fred & Jackendoff, Ray. *A Generative Theory of Tonal Music*. The MIT Press, 1983

Miller, G.A. *The magical number seven, plus or minus two: some limits on our capacity to process information*. In: *Psychological Review 63*, 1956

Müller, Gereon. *Local vs. global optimization in syntax: a case study*. ROA, 2002, To appear in Jennifer Spenader et al. (eds.), *Proceedings of the Workshop on Variation within Optimality Theory*. Stockholm University: Institute of Linguistics.

Prince, Alan & Smolensky, Paul. *Optimality Theory: Constraint Interaction in Generative Grammar*. Ms. Rutgers University, New Brunswick, and University of Colorado, Boulder. NJ, 1993

Scher, Steven Paul (ed.). *Music and text: critical inquiries.* Cambridge University Press, 1992

Temperley, David. *The Cognition of Basic Musical Structures.* The MIT Press, 2001

Tesar, Bruce & Smolensky, Paul. *The Learnability of Optimality Theory: An Algorithm and Some Basic Complexity Results.* ROA, 1998

Willemze, Theo. *Algemene Muziekleer.* Het Spectrum, 1993

IX.2. Consulted websites

http://www.classicarchives.com
A database of classical music. Music is available in MIDI and in MP3. Site founded by Pierre R. Schwob.

http://tyala.freeyellow.com/4scales.htm
A site about musical scales, relation to frequency, tuning, etc. Created by T. Tahaya Abdullah.

http://www.muspe.unibo.it
Official site of the department of music and drama of the University of Bologna.

http://roa.rutgers.edu
The Rutgers Optimality Archive is a distribution point for research in Optimality Theory. Posting in ROA is open to all who wish to disseminate their work in, on, or about OT. Managed by: Eric Bakovic.

http://www.let.rug.nl/~hendriks
Personal website of Petra Hendriks.

APPENDIX 1 – Code of "GTTM.ARI"

```
:-     write('GROUPING MUSIC'),nl,
           write('According to GTTM by Jackendoff and Lerdahl.'),nl,nl,
           write('Typing "gen." will start the GEN-process.'),nl,
           write('Typing "eval." will start the EVAL-process.'),nl,nl,
           write('During processing, the following files will be produced:'),nl,
           write('candidates.txt; containing all possible groupingstructures.'),nl,
           write('output.txt     ; containing the OT-scheme and winner.'),nl.


%%%%%%%%%%%%%%%%%%%
%| OT-PROCESSES |%
%%%%%%%%%%%%%%%%%%%


%| GEN |%

gen :-
          write('Give filename: '),
          nl,
          read(Filename),
          see(Filename),
          read(String),
          seen,
          gen(String).

gen(String) :-
          tell('candidates.txt'),
          write('['),
          not generate(String),
          write('[[end_of_file]]].'),
          told.

generate(String) :-
          glue(Structure,String),
          write(Structure),
          write(','),
          nl,
          fail.

glue([List],List) :-
          nonempty(List).
glue([Listl|Rest],BiggerList) :-
          conc(Listl,NextList,BiggerList),
          nonempty(Listl),
          glue(Rest,NextList).
```

```
%| EVAL |%

% eval/0 coördinates the evaluation process.
eval:-
          see('candidates.txt'),
          read(TempCandidates),
          seen,
          del([[end_of_file]],TempCandidates,Candidates),
          tell('output.txt'),
          write('['),
          eval(Candidates),
          write('[end_of_file]].'),
          told,
          see('output.txt'),
          read(TempScores),
          seen,
          del([end_of_file],TempScores,Scores),
          getwinner(Candidates,Scores,Maximum,Winner),
          nl,
          write('Prefered structure:'),nl,
          initwritegroups(Winner),nl,nl,
          write('OT-tableau:'),nl,
          graph(Maximum),nl.

eval([]).
eval([Candidate|Rest]) :-
          score(Candidate,_,ScoreList),
          write(ScoreList),
          write(','),
          nl,
          eval(Rest).

% Here, actual processing of the GPR's takes place.
score([],ScoreList,ScoreList).
score([Group|Rest],_,[Sc1,Sc2a,Sc2b,Sc3a,Sc3b,Sc3d]) :-
          score(Rest,[Sc1,Sc2a,Sc2b,Sc3a,Sc3b,Sc3d], ...
                  [Rest1,Rest2a,Rest2b,Rest3a,Rest3b,Rest3d]),
          gpr1(Group,Gpr1),
          max(Gpr1,Rest1,Sc1),
          gpr2a(Group,Gpr2a),
          max(Gpr2a,Rest2a,Sc2a),
          gpr2b(Group,Gpr2b),
          max(Gpr2b,Rest2b,Sc2b),
          gpr3a(Group,Gpr3a),
          max(Gpr3a,Rest3a,Sc3a),
          gpr3b(Group,Gpr3b),
          max(Gpr3b,Rest3b,Sc3b),
          gpr3d(Group,Gpr3d),
          max(Gpr3d,Rest3d,Sc3d).
```

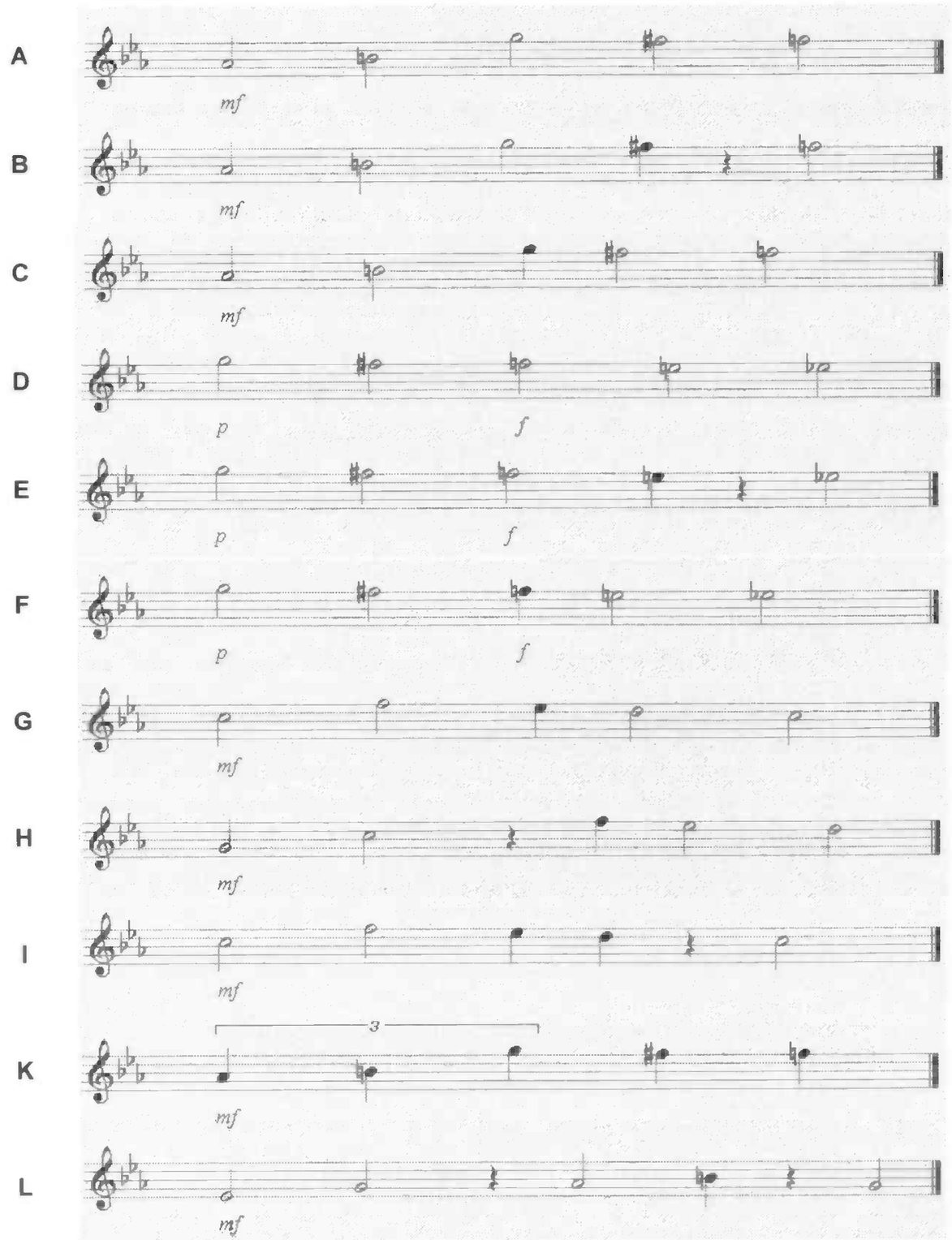
```
%| GPR's |%

% GPR 1: SINGLES
/* No group contains one element */
gpr1([n(_,_,_,_)],1) :- !.
gpr1(_,0).

% GPR 2a: PROXIMITY SLUR/REST
/* No group contains a contiguous sequence of three notes, such that the
interval of time from the end of the second note to the beginning of the
third is greater than that from the end of the first note to the begin-
ning of the second. */
gpr2a(Struc,1) :-
          subset(Struc,[n(_,Off1,_,_),n(On2,Off2,_,_),n(On3,_,_,_)]),
          Int1 is On2 - Off1,
          Int2 is On3 - Off2,
          Int2 > Int1,
          !.
gpr2a(_,0).

% GPR 2b: PROXIMITY ATTACKPOINTS
/* No group contains a contiguous sequence of three notes, such that the
interval of time between the attackpoints of the second and third note is
greater than that between the attackpoints of the first and second note.*/
gpr2b(Struc,1) :-
          subset(Struc,[n(On1,_,_,_),n(On2,_,_,_),n(On3,_,_,_)]),
          Int1 is On2 - On1,
          Int2 is On3 - On2,
          Int2 > Int1,
          !.
gpr2b(_,0).


% GPR 3a: CHANGE REGISTER
/* No group contains a contiguous sequence of three notes, such that the
interval from the second to the third note is bigger than that from the
first to the second note. */
gpr3a(Struc,1) :-
          subset(Struc,[n(_,_,F1,_),n(_,_,F2,_),n(_,_,F3,_)]),
          int(F1,F2,Int1),
          int(F2,F3,Int2),
          Int2 > Int1,
          !.
gpr3a(_,0).

% GPR 3b: CHANGE DYNAMICS
/* No group contains a contiguous sequence of three notes, such that the
first two share the same dynamics, different from the third. */
gpr3b(Struc,1) :-
          subset(Struc,[n(_,_,_,Dyn),n(_,_,_,Dyn),n(_,_,_,Oth_dyn)]),
          not Dyn = Oth_dyn,
          !.
gpr3b(_,0).

% GPR 3d: CHANGE LENGTH
/* No groups contains a contiguous sequence of three notes, such that the
first two share the same length, different form the third. */
gpr3d(Struc,1) :-
          subset(Struc,[n(On1,Off1,_,_),n(On2,Off2,_,_),n(On3,Off3,_,_)]),
          Dur1 is Off1 - On1,
          Dur1 is Off2 - On2,
          Dur3 is Off3 - On3,
          not Dur1 = Dur3,
          !.
gpr3d(_,0).
```

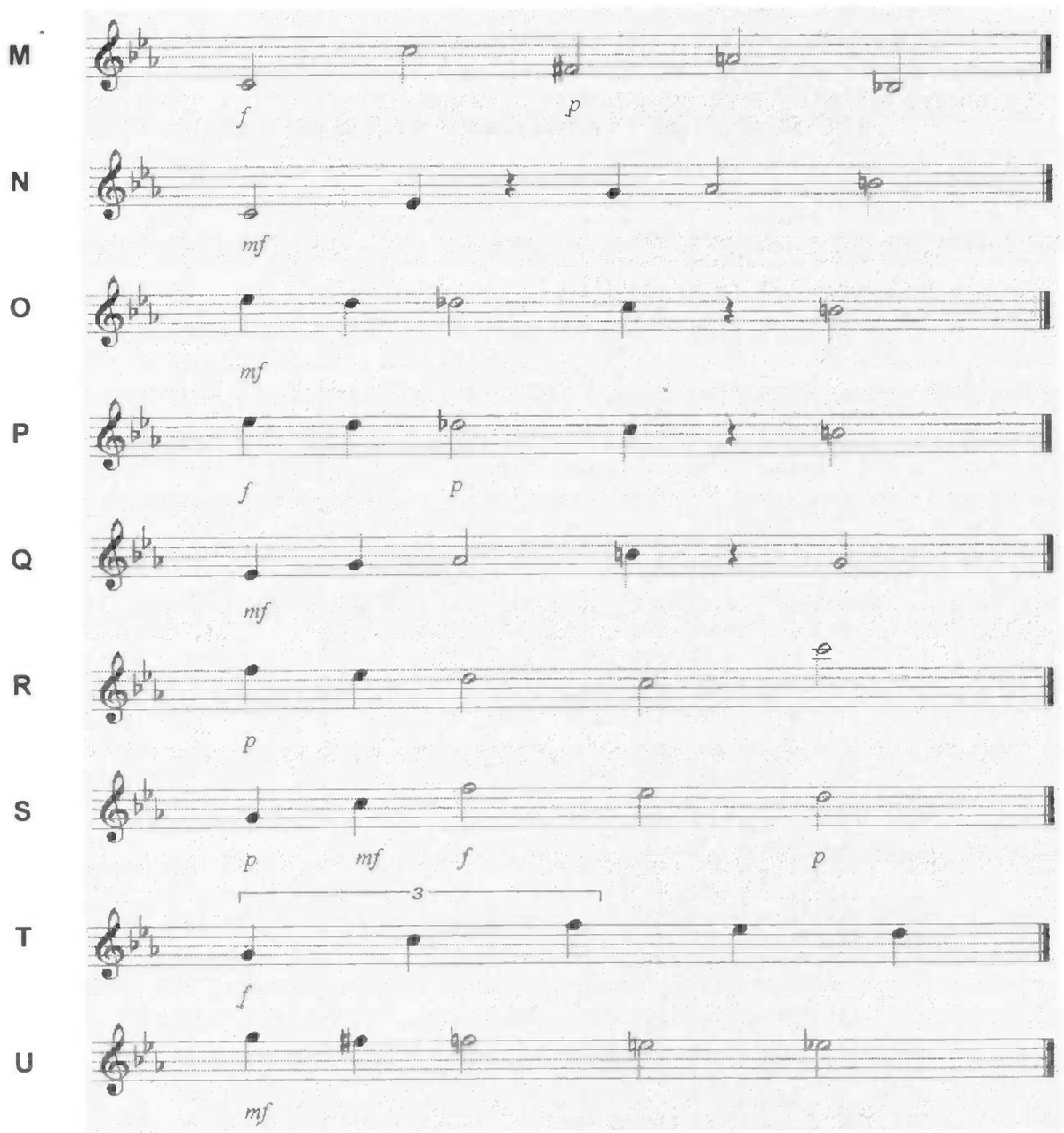
```
%%%%%%%%%%%%%%%%%
%| SUBROUTINES |%
%%%%%%%%%%%%%%%%%

nonempty(List) :-
          islist(List),
          not empty(List).
empty([]).

islist([]).
islist([Head|Tail]) :-
          islist(Tail).

containslist([Head|_]) :-
          islist(Head).
containslist([_|Tail]) :-
          containslist(Tail).

listlist([List]) :-
          islist(List).
listlist([Head|Tail]) :-
          islist(Head),
          listlist(Tail).

conc([],L,L).
conc([X|L1],L2,[X|L3]) :-
          conc(L1,L2,L3).

del(X,[X|Tail],Tail).
del(X,[Head|Tail1],[Head|Tail2]) :-
          del(X,Tail1,Tail2).

max(X,Y,Y) :-
          Y > X.
max(X,Y,X) :-
          Y =< X.

min(X,Y,Y) :-
          Y < X.
min(X,Y,X) :-
          Y >= X.

subset(List,Sub) :-
          conc(Begin,Rest,List),
          conc(Sub,End,Rest).

% int/3 gives the interval between two notes.
int(F1,F2,Int) :-
          max(F1,F2,Max),
          min(F1,F2,Min),
          Int is Max / Min.

% getbest/5 chooses from a structure/score-pair the best structure.
% That is, with smallest Numeric Score.
getbest(Item,_,Score1,Score2,Score1,Item) :-
          trunc(Score1,NumericScore1),
          trunc(Score2,NumericScore2),
          NumericScore1 < NumericScore2.
getbest(_,Item,Score1,Score2,Score2,Item) :-
          trunc(Score1,NumericScore1),
          trunc(Score2,NumericScore2),
          NumericScore1 >= NumericScore2.
```

```
% getwinner/4 chooses from structure/score-lists the winning structure.
getwinner([Winner],[Score],Score,Winner).
getwinner([Candidate1,Candidate2|Candidates],[Score1,Score2|Scores],...
                  MaxScore,Winner) :-
          getwinner([Candidate2|Candidates],[Score2|Scores],NextScore,NextWinner),
          getbest(Candidate1,NextWinner,Score1,NextScore,MaxScore,Winner).

% trunc/2 makes a list into a string.
trunc(List,Term) :-
          convert(CharList,List),
          name(Term,CharList).

% convert will connect a list of ASCII-codes with one of the actual characters.
convert([],[]).
convert([FirstCode|RestCode],[FirstChar|RestChar]) :-
          name(FirstChar,[FirstCode]),
          convert(RestCode,RestChar).


%%%%%%%%%%%%%%%%%%%%%%%%%%
%| GRAPHICAL SUBROUTINES |%
%%%%%%%%%%%%%%%%%%%%%%%%%%

% writegroups produces a graphical representation of the given group.
initwritegroups(Structure) :-
          write('[ '),
          writegroups(Structure),
          write(']').
writegroups([]).
writegroups([Head|Tail]) :-
          writehead(Head),
          write(' '),
          writegroups(Tail).
writehead([]).
writehead([Head|Tail]) :-
          write('-'),
          writehead(Tail).

% graph produces a graphic version of an OT-Tableau.
graph(Maximum) :-
          write('|   1  2a 2b 3a 3b 3d |'),nl,
          write('|> '),
          print(Maximum),
          write(' |'),nl.
print([]).
print([Head|Tail]) :-
          getsymbol(Head,Char),
          write(' '),
          write(Char),
          write(' '),
          print(Tail).
getsymbol(1,'*').
getsymbol(0,' ').
```

APPENDIX 2 – Stimuli

In this appendix all stimuli are given in costum music notation. Stimulus J is left out conform musical convention.




M
f
p
N
mf
O
mf
P
f
p
Q
mf
R
p
S
p
mf
f
p
3
T
f
U
mf

The second part of this appendix gives the corresponding OT-tableaux. There are 16 candidates for every stimulus. First the candidates are stated in a notation where numeric length of groups are seperated by a +. E.g.: "2+3" means that the first two notes are grouped and the last three too. Then the violations of constraints are given in the form of a vector. An asterix marks a violation. The ordering in the vector is [1, 2a, 2b, 3a, 3b, 3d].

5	[, , ,*, ,]	A
1+4	[*, , , , ,]	
1+1+3	[*, , , , ,]	
1+1+1+2	[*, , , , ,]	
1+1+1+1+1	[*, , , , ,]	
1+1+2+1	[*, , , , ,]	
1+2+2	[*, , , , ,]	
1+2+1+1	[*, , , , ,]	
1+3+1	[*, , , , ,]	
2+3	[, , , , ,]	
2+1+2	[*, , , , ,]	
2+1+1+1	[*, , , , ,]	
2+2+1	[*, , , , ,]	
3+2	[, , ,*, ,]	
3+1+1	[*, , ,*, ,]	
4+1	[*, , ,*, ,]	

5	[,*, ,*, ,*]	B
1+4	[*,*, , , ,*]	
1+1+3	[*,*, , , ,]	
1+1+1+2	[*, , , , ,]	
1+1+1+1+1	[*, , , , ,]	
1+1+2+1	[*, , , , ,]	
1+2+2	[*, , , , ,]	
1+2+1+1	[*, , , , ,]	
1+3+1	[*, , , , ,*]	
2+3	[,*, , , ,]	
2+1+2	[*, , , , ,]	
2+1+1+1	[*, , , , ,]	
2+2+1	[*, , , , ,]	
3+2	[, , ,*, ,]	
3+1+1	[*, , ,*, ,]	
4+1	[*, , ,*, ,*]	

5	[, ,*,*, ,*]	C
1+4	[*, ,*, , ,]	
1+1+3	[*, ,*, , ,]	
1+1+1+2	[*, , , , ,]	
1+1+1+1+1	[*, , , , ,]	
1+1+2+1	[*, , , , ,]	
1+2+2	[*, , , , ,]	
1+2+1+1	[*, , , , ,]	
1+3+1	[*, , , , ,]	
2+3	[, ,*, , ,]	
2+1+2	[*, , , , ,]	
2+1+1+1	[*, , , , ,]	
2+2+1	[*, , , , ,]	
3+2	[, , ,*, ,*]	
3+1+1	[*, , ,*, ,*]	
4+1	[*, , ,*, ,*]	

5	[, , , ,*,]	D
1+4	[*, , , , ,]	
1+1+3	[*, , , , ,]	
1+1+1+2	[*, , , , ,]	
1+1+1+1+1	[*, , , , ,]	
1+1+2+1	[*, , , , ,]	
1+2+2	[*, , , , ,]	
1+2+1+1	[*, , , , ,]	
1+3+1	[*, , , , ,]	
2+3	[, , , , ,]	
2+1+2	[*, , , , ,]	
2+1+1+1	[*, , , , ,]	
2+2+1	[*, , , , ,]	
3+2	[, , , ,*,]	
3+1+1	[*, , , ,*,]	
4+1	[*, , , ,*,]	

5	[,*, , ,*,*]	E
1+4	[*,*, , , ,*]	
1+1+3	[*,*, , , ,]	
1+1+1+2	[*, , , , ,]	
1+1+1+1+1	[*, , , , ,]	
1+1+2+1	[*, , , , ,]	
1+2+2	[*, , , , ,]	
1+2+1+1	[*, , , , ,]	
1+3+1	[*, , , , ,*]	
2+3	[,*, , , ,]	
2+1+2	[*, , , , ,]	
2+1+1+1	[*, , , , ,]	
2+2+1	[*, , , , ,]	
3+2	[, , , ,*,]	
3+1+1	[*, , , ,*,]	
4+1	[*, , , ,*,*]	

5	[, ,*, ,*,*]	F
1+4	[*, ,*, , ,]	
1+1+3	[*, ,*, , ,]	
1+1+1+2	[*, , , , ,]	
1+1+1+1+1	[*, , , , ,]	
1+1+2+1	[*, , , , ,]	
1+2+2	[*, , , , ,]	
1+2+1+1	[*, , , , ,]	
1+3+1	[*, , , , ,]	
2+3	[, ,*, , ,]	
2+1+2	[*, , , , ,]	
2+1+1+1	[*, , , , ,]	
2+2+1	[*, , , , ,]	
3+2	[, , , ,*,*]	
3+1+1	[*, , , ,*,*]	
4+1	[*, , , ,*,*]	

```
        5  [ , ,*,*, ,*]  G
      1+4  [*, ,*,*, , ]
    1+1+3  [*, ,*,*, , ]
  1+1+1+2  [*, , , , , ]
1+1+1+1+1  [*, , , , , ]
  1+1+2+1  [*, , , , , ]
    1+2+2  [*, , , , , ]
  1+2+1+1  [*, , , , , ]
    1+3+1  [*, , , , , ]
      2+3  [ , ,*,*, , ]
    2+1+2  [*, , , , , ]
  2+1+1+1  [*, , , , , ]
    2+2+1  [*, , , , , ]
      3+2  [ , , , , ,*]
    3+1+1  [*, , , , ,*]
      4+1  [*, , , , ,*]

        5  [ ,*,*,*, ,*]  H
      1+4  [*, ,*, , , ]
    1+1+3  [*, ,*, , , ]
  1+1+1+2  [*, , , , , ]
1+1+1+1+1  [*, , , , , ]
  1+1+2+1  [*, , , , , ]
    1+2+2  [*, , , , , ]
  1+2+1+1  [*, , , , , ]
    1+3+1  [*, , , , , ]
      2+3  [ , ,*, , , ]
    2+1+2  [*, , , , , ]
  2+1+1+1  [*, , , , , ]
    2+2+1  [*, , , , , ]
      3+2  [ ,*,*,*, ,*]
    3+1+1  [*,*,*,*, ,*]
      4+1  [*,*,*,*, ,*]

        5  [ ,*,*,*, ,*]  I
      1+4  [*,*,*,*, ,*]
    1+1+3  [*,*,*,*, ,*]
  1+1+1+2  [*, , , , , ]
1+1+1+1+1  [*, , , , , ]
  1+1+2+1  [*, , , , , ]
    1+2+2  [*, , , , , ]
  1+2+1+1  [*, , , , , ]
    1+3+1  [*, , , , , ]
      2+3  [ ,*,*,*, ,*]
    2+1+2  [*, , , , , ]
  2+1+1+1  [*, , , , , ]
    2+2+1  [*, , , , , ]
      3+2  [ , , , , ,*]
    3+1+1  [*, , , , ,*]
      4+1  [*, , , , ,*]

        5  [ , ,*,*, ,*]  K
      1+4  [*, ,*, , ,*]
    1+1+3  [*, ,*, , , ]
  1+1+1+2  [*, , , , , ]
1+1+1+1+1  [*, , , , , ]
  1+1+2+1  [*, , , , , ]
    1+2+2  [*, , , , , ]
  1+2+1+1  [*, , , , , ]
    1+3+1  [*, , , , ,*]
      2+3  [ , ,*, , , ]
    2+1+2  [*, , , , , ]
  2+1+1+1  [*, , , , , ]
    2+2+1  [*, , , , , ]
      3+2  [ , , ,*, , ]
    3+1+1  [*, , ,*, , ]
      4+1  [*, , ,*, ,*]

        5  [ ,*,*,*, ,*]  L
      1+4  [*,*, ,*, ,*]
    1+1+3  [*,*, ,*, , ]
  1+1+1+2  [*, , , , , ]
1+1+1+1+1  [*, , , , , ]
  1+1+2+1  [*, , , , , ]
    1+2+2  [*, , , , , ]
  1+2+1+1  [*, , , , , ]
    1+3+1  [*, , ,*, ,*]
      2+3  [ ,*, ,*, , ]
    2+1+2  [*, , , , , ]
  2+1+1+1  [*, , , , , ]
    2+2+1  [*, , , , , ]
      3+2  [ ,*,*, , , ]
    3+1+1  [*,*,*, , , ]
      4+1  [*,*,*,*, ,*]

        5  [ , , ,*,*, ]  M
      1+4  [*, , ,*, , ]
    1+1+3  [*, , ,*, , ]
  1+1+1+2  [*, , , , , ]
1+1+1+1+1  [*, , , , , ]
  1+1+2+1  [*, , , , , ]
    1+2+2  [*, , , , , ]
  1+2+1+1  [*, , , , , ]
    1+3+1  [*, , , , , ]
      2+3  [ , , ,*, , ]
    2+1+2  [*, , , , , ]
  2+1+1+1  [*, , , , , ]
    2+2+1  [*, , , , , ]
      3+2  [ , , , ,*, ]
    3+1+1  [*, , , ,*, ]
      4+1  [*, , , ,*, ]

        5  [ ,*,*,*, ,*]  N
      1+4  [*, ,*,*, ,*]
    1+1+3  [*, ,*,*, , ]
  1+1+1+2  [*, , , , , ]
1+1+1+1+1  [*, , , , , ]
  1+1+2+1  [*, , , , , ]
    1+2+2  [*, , , , , ]
  1+2+1+1  [*, , , , , ]
    1+3+1  [*, , , , ,*]
      2+3  [ , ,*,*, , ]
    2+1+2  [*, , , , , ]
  2+1+1+1  [*, , , , , ]
    2+2+1  [*, , , , , ]
      3+2  [ ,*, ,*, , ]
    3+1+1  [*,*, ,*, , ]
      4+1  [*,*, ,*, ,*]

        5  [ ,*,*, , ,*]  O
      1+4  [*,*,*, , , ]
    1+1+3  [*,*, , , , ]
  1+1+1+2  [*, , , , , ]
1+1+1+1+1  [*, , , , , ]
  1+1+2+1  [*, , , , , ]
    1+2+2  [*, , , , , ]
  1+2+1+1  [*, , , , , ]
    1+3+1  [*, ,*, , , ]
      2+3  [ ,*, , , , ]
    2+1+2  [*, , , , , ]
  2+1+1+1  [*, , , , , ]
    2+2+1  [*, , , , , ]
      3+2  [ , , , , ,*]
    3+1+1  [*, , , , ,*]
      4+1  [*, ,*, , ,*]
```

```
        5  [ ,*,*, ,*,*]  P
      1+4  [*,*,*, , , ]
    1+1+3  [*,*, , , , ]
  1+1+1+2  [*, , , , , ]
1+1+1+1+1  [*, , , , , ]
  1+1+2+1  [*, , , , , ]
    1+2+2  [*, , , , , ]
  1+2+1+1  [*, , , , , ]
    1+3+1  [*, ,*, , , ]
      2+3  [ ,*, , , , ]
    2+1+2  [*, , , , , ]
  2+1+1+1  [*, , , , , ]
    2+2+1  [*, , , , , ]
      3+2  [ , , , ,*,*]
    3+1+1  [*, , , ,*,*]
      4+1  [*, ,*, ,*,*]

        5  [ ,*,*,*, ,*]  Q
      1+4  [*,*,*,*, , ]
    1+1+3  [*,*, ,*, , ]
  1+1+1+2  [*, , , , , ]
1+1+1+1+1  [*, , , , , ]
  1+1+2+1  [*, , , , , ]
    1+2+2  [*, , , , , ]
  1+2+1+1  [*, , , , , ]
    1+3+1  [*, ,*,*, , ]
      2+3  [ ,*, ,*, , ]
    2+1+2  [*, , , , , ]
  2+1+1+1  [*, , , , , ]
    2+2+1  [*, , , , , ]
      3+2  [ , , , , ,*]
    3+1+1  [*, , , , ,*]
      4+1  [*, ,*,*, ,*]

        5  [ , ,*,*, ,*]  R
      1+4  [*, ,*,*, , ]
    1+1+3  [*, , ,*, , ]
  1+1+1+2  [*, , , , , ]
1+1+1+1+1  [*, , , , , ]
  1+1+2+1  [*, , , , , ]
    1+2+2  [*, , , , , ]
  1+2+1+1  [*, , , , , ]
    1+3+1  [*, ,*,*, , ]
      2+3  [ , , ,*, , ]
    2+1+2  [*, , , , , ]
  2+1+1+1  [*, , , , , ]
    2+2+1  [*, , , , , ]
      3+2  [ , , , , ,*]
    3+1+1  [*, , , , ,*]
      4+1  [*, ,*,*, ,*]

        5  [ , ,*,*,*,*]  S
      1+4  [*, ,*, ,*, ]
    1+1+3  [*, , , ,*, ]
  1+1+1+2  [*, , , , , ]
1+1+1+1+1  [*, , , , , ]
  1+1+2+1  [*, , , , , ]
    1+2+2  [*, , , , , ]
  1+2+1+1  [*, , , , , ]
    1+3+1  [*, ,*, , , ]
      2+3  [ , , , ,*, ]
    2+1+2  [*, , , , , ]
  2+1+1+1  [*, , , , , ]
    2+2+1  [*, , , , , ]
      3+2  [ , , ,*, ,*]
    3+1+1  [*, , ,*, ,*]
      4+1  [*, ,*,*, ,*]

        5  [ , ,*,*, ,*]  T
      1+4  [*, ,*, , ,*]
    1+1+3  [*, ,*, , , ]
  1+1+1+2  [*, , , , , ]
1+1+1+1+1  [*, , , , , ]
  1+1+2+1  [*, , , , , ]
    1+2+2  [*, , , , , ]
  1+2+1+1  [*, , , , , ]
    1+3+1  [*, , , , ,*]
      2+3  [ , ,*, , , ]
    2+1+2  [*, , , , , ]
  2+1+1+1  [*, , , , , ]
    2+2+1  [*, , , , , ]
      3+2  [ , , ,*, , ]
    3+1+1  [*, , ,*, , ]
      4+1  [*, , ,*, ,*]

        5  [ , ,*, , ,*]  U
      1+4  [*, ,*, , , ]
    1+1+3  [*, , , , , ]
  1+1+1+2  [*, , , , , ]
1+1+1+1+1  [*, , , , , ]
  1+1+2+1  [*, , , , , ]
    1+2+2  [*, , , , , ]
  1+2+1+1  [*, , , , , ]
    1+3+1  [*, ,*, , , ]
      2+3  [ , , , , , ]
    2+1+2  [*, , , , , ]
  2+1+1+1  [*, , , , , ]
    2+2+1  [*, , , , , ]
      3+2  [ , , , , ,*]
    3+1+1  [*, , , , ,*]
      4+1  [*, ,*, , ,*]
```

APPENDIX 3 – Individual Results of experiment

For every subject, the following data are given: Initials, ranking of constraints, C(**r**,**R**), C(**r**,**m**), # years of playing an instrument, # years taken lessons.

AB	
1 » 3b » 3a » 2a » 3d » 2b	
C(**r**,**R**) = 0,65	C(**r**,**m**) = 0,65
# instrument: 14 years	# lessons: 12 years

CB	
2a » 3d » 1 » 3b » 3a » 2b	
C(**r**,**R**) = 0,5	C(**r**,**m**) = 0,5
# instrument: 15 years	# lessons: 14 years

DB	
1 » 3d » 3b » 2a » 2b » 3a	
C(**r**,**R**) = 0,6	C(**r**,**m**) = 0,9
# instrument: 15 years	# lessons: 11 years

DV	
2a » 3d » 1 » 3b » 2b » 3a	
C(**r**,**R**) = 0,65	C(**r**,**m**) = 0,65
# instrument: 18 years	# lessons: 12 years

HS	
{ 2a, 2b, 3a, 3b } » 3d » 1	
C(**r**,**R**) = 0,2	C(**r**,**m**) = 0,5
# instrument: 15 years	# lessons: 12 years

JK	
1 » 3b » 3a » 3d » 2a » 2b	
C(**r**,**R**) = 0,6	C(**r**,**m**) = 0,6
# instrument: 14 years	# lessons: 10 years

JM	
{ 2a, 3b } » 2b » 3a » 3d » 1	
C(**r**,**R**) = 0,25	C(**r**,**m**) = 0,45
# instrument: 17 years	# lessons: 7 years

KB	
2a » 3a » 3d » 3b » 1 » 2b	
C(**r**,**R**) = 0,85	C(**r**,**m**) = 0,55
# instrument: 27 years	# lessons: 12 years

SH	
2a » { 3b, 3d } » 1 » 2b » 3a	
C(**r**,**R**) = 0,7	C(**r**,**m**) = 0,6
# instrument: 17 years	# lessons: 5 years

TB	
2a » 1 » 3a » 3d » 3b » 2b	
C(**r**,**R**) = 0,7	C(**r**,**m**) = 0,45
# instrument: 15 years	# lessons: 10 years