Knowledge and Learning Based Segmentation and Recognition of Rhythm Using Fuzzy-Prolog

Tillman Weyde

Research Department of Music und Media Technology

University of Osnabrück

Osnabrück, Germany

tweyde@uos.de

Abstract This paper introduces an architecture for rhythm recognition and comparative analysis. A fuzzy system is used to rate segmentation and structural assignment produced by combinatorial pattern-matching. The fuzzy system can be trained by examples. It provides fault tolerant, context sensitive and adaptive recognition of musical rhythm with a description of temporal and structural deviations.

1 Introduction

Fuzzy logic allows for the modeling systems with uncertain and incomplete knowledge. Fuzzy systems can be robust and show graceful degradation on their limits while their function is still interpretable, it is not just a black box. So it is surprising that the use of fuzzy logic is rare in the field of music theory and musical applications[1], since our knowledge of processes involved in musical activity is far from being exact or complete.

The immediate recognition of auditory rhythmic structures in one of the key features of human auditory perception, necessary for understanding speech as well as music. Although there is a large body of research on the properties of auditory perception of temporal patterns[2], a coherent paradigm or theoretical framework to support computer models and applications is not yet established. This is a problem for musical computer applications like production tools, tutorial programs an musical database applications which should interact with the user in a musically meaningful way.

This paper introduces a framework for modeling the recognition of musical rhythmic structure. It provides a basis for integration of prior knowledge from empirical experiments and

[1] There are a few examples like [Kostek, 1999] who uses fuzzy logic and there are some custom system designs like [Schottstead, 1989] which contain elements similar to fuzzy systems.

[2] E.g. [Deutsch, 1986], [Handel, 1989], [Desain and Windsor, 2000].

music theory with a learning by example approach.

1.1 Musical pattern processing

Musically meaningful structures are constrained by the auditory perception of music since the main form of consuming and appreciating music is listening (rather than reading). So the integration of properties of aural perception is obligatory. Also knowledge from music theory can be integrated (which is dependent on cultural context).

There are two main aspects of rhythmic structure which need to be considered for recognition and analysis: *segmentation* and *similarity*. The segmentation process divides a sequence of events into groups. These groups correspond to musical motifs which are combined to higher level patterns like musical phrases. The similarity of groups determines the internal structure of these patterns or the relation to a known model (e.g. a given rhythm which should be played by a student). Groups of notes are assigned to groups in a model or within a piece of music on the basis of similarity. Segmentation and assignment processes are highly interdependent and both depend strongly on context.

2 System architecture

The initial motivation for this work was to bring more musical intelligence into music tutorial applications. The user should try to play rhythms after hearing or reading them. Our aim is to give differentiated and musically meaningful responses to the user about how her or his input differs from the task and what she or he should take care of. This is the more difficult the larger the differences between task and input are since it is not clear which part of the input corresponds to which part of the task. Missing or extra notes or groups, wrong order of groups, changing tempo and timing deviations can only be noticed by the system if it recognizes which groups were played by the user, even if they are temporally or structurally distorted or misplaced.

Our system currently supports three modes: segmentation only, matching patterns and matching structures. The segmentation mode emulates segmentation by a listener. It can be seen as a partial implementation of the grouping rules by [Lerdahl and Jackendoff, 1983] restricted to the rhythm domain. The pattern matching mode emulates recognition of a single rhythmic group. It assigns the notes in the matched patterns and gives thus detailed information on the differences between task and input groups. The structure matching mode combines and extends the former two. It assigns the groups and allows a detection and description of structural changes like omissions, insertions or changed order of groups.

The general scheme is to combinatorially generate segmentations, assignments of groups, and assignments of notes and filtering them. Filtering reduces complexity by removing perceptually implausible interpretations. Features are extracted from segmentations and assignments and a rating is calculated by the Fuzzy-Prolog program. The alternative rated best determines system output. A Fuzzy-Prolog program can be viewed as a neural net, where rules and facts correspond to network nodes. So by node we mean just a different view on a fuzzy rule or fact.

2.1 Combinatorial processing

Input data

Input data represents musical notes based on MIDI data. The input events have three values that we use: onset time, duration, and key velocity (loudness). If there is a task to which the input is to be compared to, the task is encoded in the same way.

Segmentation and group assignments

There are perceptual constraints to the length and the duration of perceptual groups. The number of events in a group is restricted as is well known since [Miller, 1956]. The maximal number is a adjustable parameter in our system and a setting of 4 or 5 has shown to be adequate which agrees with the literature ([Handel and Todd, 1981], [Swain, 1986]). We model these constraints by generating all segmentations within the given range of group lengths and filtering out those that do not meet the constraints.

Empirical evidence suggests that durations of perceptual groups lie in a range of approximately 0.5 to 2 seconds ([Seifert et al., 1995]). It is also known that temporal proximity of events plays a role, relatively long distances between events tend to end a group ([Handel, 1973]). Since grouping by temporal proximity is dominant over accent grouping ([Deutsch, 1986]) we can filter out segmentations that grossly contradict grouping by proximity. We also assume that groups containing only one element should not occur unless they have considerable distance to the neighbor notes ([Lerdahl and Jackendoff, 1983]) else they are filtered out.

On the structure level input groups are assigned to task groups based on input and task segmentation. All possible assignments for all combinations of segmentations are calculated. Currently we do not filter group assignments.

Note assignments and tempo variants

In a match of input and task groups notes of the input are assigned to notes of the task or marked as additional (leaving the temporal relations of the other notes intact) or inserted (the rest of the group being moved by the amount of time occupied by note), task notes are marked as assigned, subtracted, and removed respectively.

For every assigned input group tempo variants are calculated based on pairs of assigned notes as anchor points. Different possibilities for the two anchor notes yield different tempo variants. The process is visualized in figure 1. The result allows measuring the deviation of the group from the expected position, the tempo deviation and deviations of the individual notes concerning timing and loudness. Only the tempo variant which produces the best similarity rating is used for further calculation in order to reduce calculation time.

2.2 Feature extraction

On both the group level and on the structure level features are extracted that form the basis for the similarity and segmentation rating.

For segmentations we calculate ratings based on their length and number of notes. For

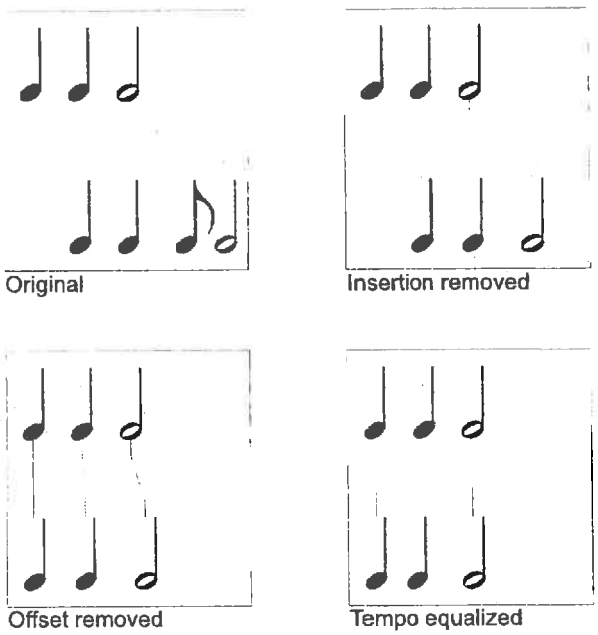

Figure 1: Calculation of tempo variants

length values we use four input nodes corresponding to different length. This design allows good fitting to a preference curve on the considered range of group lengths. The same design is used for the number of notes in a group. We have five different input nodes corresponding to note numbers from one to five. The regularity of group lengths (number of notes) as well as of group intervals (distance between the beginning of groups) is also rated by calculating their variance. For each group we have a node representing whether the inter-onset-interval (IOI) between the last note and the next group is larger than IOIs within the group, and whether the first note is relatively loud.

For similarity on the group and structure level nodes representing different kinds of imprecisions and inaccuracies are used as facts for the Fuzzy-Prolog program. We use nodes for early notes, late notes, too loud notes, too soft notes, added notes, inserted notes, subtracted notes, deleted notes, tempo stability, and tempo plausibility. The calculation of their values is not described here due to space limitations[3].

A feature specific for the structure level is the order of groups which is calculated as

$$\frac{i}{n \cdot (n-1)/2}$$

where i is the number of the intersections in the assignment graph and n the number of group assignments (see figure 2). The other nodes on structure level are calculated as combinations of the corresponding nodes per group.

[3]Some more details can be found in [Weyde, 2000] and [Enders and Weyde, 1996]

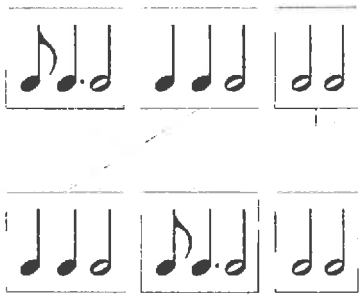

Figure 2: Group assignments for a rhythmic structure

3 Segmentation and similarity ratings using Fuzzy-Prolog

3.1 Fuzzy-Prolog

The module for rating segmentation quality and group similarity is based on *Fuzzy-Prolog* which was described by [Nauck et al., 1996]. Fuzzy-Prolog is a programming language similar to *Prolog*. A Fuzzy-Prolog program consists of a set of rules in the following form:

$$\text{conclusion} \leftarrow \text{premise} \wedge \ldots \wedge \text{premise}$$

Operators have an evaluation function with values in the interval $[0, 1]$. For evaluation of the implication the *Goguen-Implication* is being used. The truth value of other rules can be derived from those by using the *modus ponens* generalized for fuzzy logic [4].

3.2 Operators

We have conjunction and disjunction operators. There are several options for the evaluation function of disjuction and conjunction, well known ones are *min* for $\wedge$ and *max* for $\vee$. Since we need a differentiable function for learning by backpropagation we use this evaluation functions for conjunction:

$$f_{\wedge}(\|\varphi_1\|, \ldots, \|\varphi_n\|) = \left(\frac{\sqrt[q]{\|\varphi_1\|} + \ldots + \sqrt[q]{\|\varphi_j\|}}{n} \right)^q$$

and analogously for the disjunction:

$$f_{\vee}(\|\varphi_1\|, \ldots, \|\varphi_n\|) = \sqrt[q]{\frac{\|\varphi_1\|^q + \ldots + \|\varphi_j\|^q}{n}}$$

where $\|\varphi_n\|$ denotes the truth value of rule φ_n. These functions allow for a certain amount of compensation between the operands that can be adjusted by the q parameter. For these functions we use a q value of 2. For $q \to \infty$, $f_{\wedge}$ approaches *min* and $f_{\vee}$ approaches *max*.

[4] See [Nauck et al., 1996].

3.3 Fuzzy rules

Not all the rules that we use can be shown here due to space limitations. But we will look at an example to try and give an idea of how the system works. The rules for similarity on the structure level are based among others on the segmentation rating for input and task if present. E.g. the predicate for the input, CInputSegmentation, returns the quality of an input segmentation based on rules and facts on group and structure level:

$$\text{CInputSegmentation}(gs) \leftarrow \text{CInputSegment}(gs) \wedge \text{CEqualGroupDistance}(gs) \wedge \text{CEqualGroupLength}(gs) \quad (1)$$

$$\text{CInputSegment}(gs) \leftarrow \text{GInputSegment}(g_1) \wedge \ldots \wedge \text{GInputSegment}(g_n), \text{ where } gs = [c_1, \ldots, c_n] \quad (2)$$

$$\text{GInputSegment}(g_i) \leftarrow \text{GGroupDuration}(g_i) \wedge \text{GGroupLenth}(g_i) \wedge \text{GGroupEndLong}(g_i) \wedge \text{GGroupStartLoud}(g_i) \quad (3)$$

$$\text{GGroupDuration}(g_i) \leftarrow \text{GGroupDuration05}(g_i) \vee \text{GGroupDuration10}(g_i) \vee \text{GGroupDuration15}(g_i) \vee \text{GGroupDuration20}(g_i) \quad (4)$$

$$\text{GGroupLength}(g_i) \leftarrow \text{GGroupLength1}(g_i) \vee \text{GGroupLength2}(g_i) \vee \text{GGroupLength3}(g_i) \vee \text{GGroupLength4}(g_i) \vee \text{GGroupLength5}(g_i) \quad (5)$$

The facts of the form GGroupDuration*XX* (rule 4) are the inputs features mentioned earlier which return values close to one if the length of the respective group is near to 0.5, 1.0, 1.5 or 2.0 seconds. Similarly the rules GGroupLength*X* (rule 5) represent a group length of 1, 2, 3, 4, and 5. Relatively loud events tend to mark a group beginning and long events tend to end a group which is reflected by facts on group level (rule 3)

We do not know in advance how many groups there will be in a structure. The number of groups varies for different segmentations of the same input. This made it necessary to extend Fuzzy-Prolog with a list processing feature. In rule 2 the transition from the structure level to the group level takes place, where a variable number of inputs is combined to one node. We call these nodes with variable numbers of connections multi-nodes.

The output node CInterpretationQual returns the rating for a whole structure assignment that comprises segmentation, similarity of groups, and assignment of groups. The stucture of the net generated from the whole Fuzzy-Prolog program is shown in figure 3.

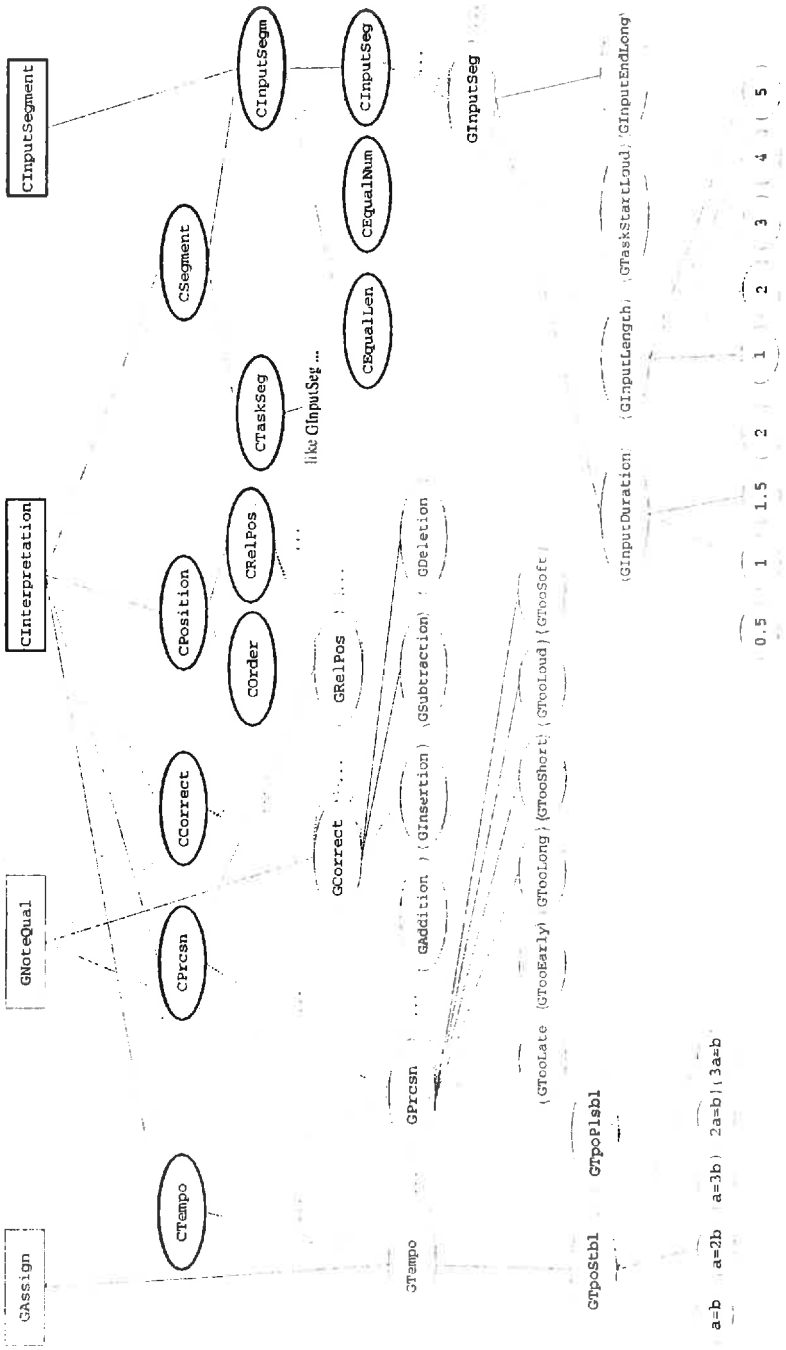

Figure 3: Structure of the neuro-fuzzy program system: Nodes on the top are output nodes (rectangles), Nodes on the bottom are net inputs Fuzzy-Prolog facts. Nodes with bold border operate on the structure level, all other nodes operate on the group level. Nodes with connections marked with '···' are transitions from group to structure level.

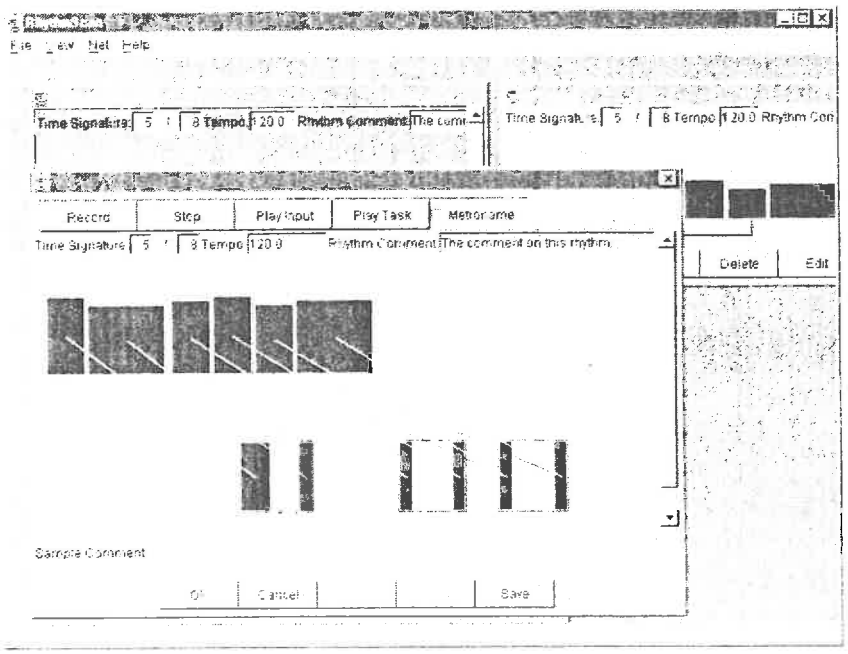

Figure 4: *RhythmScan* user interface: black lines show segmentation and assignments calculated by the system, yellow lines show input by an expert user

3.4 Learning

When trying to model how the rules contribute to the output in a fuzzy system, the rules must be weighed individually. These weights can be ad hoc estimates, but they have to be adjusted to achieve good performance. The idea is now to automate this process by optimizing the weights using rating examples. It is shown in [Nauck et al., 1996] that Fuzzy-Prolog programs are equivalent to feed-forward neural nets. They can therefore be trained with a version of the backpropagation algorithm adapted to the evaluation functions of the fuzzy logical operators. For the multi-nodes weight-sharing is used.

The system is trained with relative ratings following the method described by [Braun, 1997]. This means pairs of assignment or segmentation examples are given and one of each pair is to be rated better. When learning was successful the system prefers better assignments since they receive higher ratings.

4 Application

We have implemented an experimental application of the system described in this paper called *RhythmScan* which allows to generate samples, test, and train the system. It is a Java application with a Fuzzy-Prolog interpreter written in C and uses XML data format. The system can be interactively tested and trained by giving preferred segmentations and assignments. The training samples consist of a system output and the result preferred by the (expert) user. Differing segmentations and assignments by system and user can be seen in figure 4.

After training the system reacts adequately to segmentation tasks e.g. modeling subjective rhythmization and to perform structural comparison of user input for a given task like in interactive music tutorials. The system is able to detect delayed inputs, interrupted and repeated input as well as tempo and timing deviations.

5 Conclusions

Fuzzy-Prolog is a powerful means of modeling musical knowledge in combination with machine learning. The weights in the system trained by examples give also insight to the relevance of rules, although the sample sets have been to small to draw any generalizable conclusions yet. The system can be easily modified or extended by adding, changing or removing rules or input features.

The system works well on selected tasks but it has to be tested more intensively and systematically to explore the system capabilities. Further challenges are the integration of pitch and metrical information into the system as well as the extension to streaming mode and optimization of the computational performance.

References

[Braun, 1997] Braun, H. (1997). *Neuronale Netze*. Springer, Berlin Heidelberg.

[Desain and Windsor, 2000] Desain, P. and Windsor, L., editors (2000). *Rhythm Perception and Production*. Swets and Zeitlinger, Lisse.

[Deutsch, 1986] Deutsch, D. (1986). Auditory pattern recognition. In Boff, K. R., Kaufman, L., and Thomas, J. P., editors, *Handbook of Perception and Human Performance: Cognitive Processes and Performance*, volume 2, chapter 32, pages 32–1–49. John Wiley and Sons, New York.

[Enders and Weyde, 1996] Enders, B. and Weyde, T. (1996). Automatische Rhytmuserkennung und -vergleich mit Hilfe von Fuzzy-Logik. *Systematische Musikwissenschaft*, IV(1-2):101–113.

[Handel, 1973] Handel, S. (1973). Temporal segmentation of repeating auditory patterns. *Journal of Experimental Psychology*, 101:46–54.

[Handel, 1989] Handel, S. (1989). *Listening: An Introduction to the Perception of Auditory Events*. MIT Press, Cambridge, Massachusetts.

[Handel and Todd, 1981] Handel, S. and Todd, P. (1981). Segmentation of sequential patterns. *Journal of Experimental Psychology: Human Perception and Performance*, 7(1):41–55.

[Kostek, 1999] Kostek, B. (1999). *Soft computing in acoustics : applications of neural networks, fuzzy logic and rough sets to musical acoustics*, volume 30 of *Studies in fuzziness and soft computing*. Physica-Verlag, Heidelberg.

[Lerdahl and Jackendoff, 1983] Lerdahl, F. and Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. The MIT Press, Cambridge, Mass.

[Miller, 1956] Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2):81–97.

[Nauck et al., 1996] Nauck, D., Klawonn, F., and Kruse, R. (1996). *Neuronale Netze und Fuzzy-Systeme*. Computational Intelligence. Vieweg, Braunschweig, 2 edition.

[Schottstead, 1989] Schottstead, W. (1989). Automatic counterpoint. In Mathews, M. V. and Pierce, J. R., editors, *Current Directions in Computer Music Research*, volume 2 of *System Develpment Foundation Benchmark Series*, chapter 16, pages 199–213. The MIT Press, Cambridge, Mass.

[Seifert et al., 1995] Seifert, U., Olk, F., and Schneider, A. (1995). On rhythm perception: Theoretical issues, empirical findings. *Journal of New Music Research*, 24(2):164–95.

[Swain, 1986] Swain, J. P. (1986). The need for limits in hierarchichal theories of music. *Music Perception*, 4(1):121–148.

[Weyde, 2000] Weyde, T. (2000). Recognition of rhythmic structure with a neuro-fuzzy-system. In Woods, C., Luck, G. B., Brochard, R., Seddon, F., and Sloboda, J. A., editors, *Proceedings of the Sixth International Conference on Music Perception and Cognition*, pages 1467–77, Keele, Staffordshire, UK. Department of Psychology, Keele University. CD-ROM (pdf, html).