UNIVERSITÄT PADERBORN

Die Universität der Informationsgesellschaft

Fakultät 5 - Elektrotechnik/Mathematik/Informatik
Arbeitsgruppe Wissensbasierte Systeme
Warburger Straße 100
D-33095 Paderborn

Studienarbeit

Problemklassen bei Automatisierungsaufgaben in der Musik

vorgelegt bei
Dr. Benno Stein und Oliver Kramer
Institut für Informatik - Wissensbasierte Systeme
am 30.01.2006

Jürgen Wall

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig und ohne unerlaubte fremde Hilfe, sowie ohne Benutzung anderer als der angegebenen Quellen angefertigt habe. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Paderborn, den 30.01.2006

_____

Jürgen Wall

Inhaltsverzeichnis

1 Einleitung **1**

2 Motivation **1**

3 Gliederung **2**

I Taxonomie der Aufgaben in der Musik **3**

4 Bereiche der Taxonomie **3**

5 Analyse **8**
5.1 Analyse akustischer Daten 8
5.1.1 Merkmalsgewinnung 9
5.1.2 Deskriptor Extraktion 9
5.1.3 Cepstralkoeffizienten Extraktion 9
5.1.4 Extraction Discovery System 10
5.2 Klassifikation und Identifikation 11
5.2.1 Klassifikation von Instrumenten 11
5.2.2 Klassifikation von Genres 11
5.3 Transkription 12
5.3.1 Quellentrennung 12
5.3.2 Segmentierung 15
5.4 Analyse symbolischer Daten 20
5.4.1 Mustererkennung und Musterextraktion 20
5.4.2 Semantische Klassifikation 21

6 Synthese **23**
6.1 Klangsynthese 23
6.1.1 Künstliche Klangsynthese 23
6.2 Musikkomposition 24
6.2.1 Stochastische Systeme 25
6.2.2 Deterministische Systeme 25
6.2.3 Evolutionäre Systeme 27
6.2.4 Neuronale und Zellulare Systeme 29

7 Schlussbemerkung **31**

II Automatisiertes Komponieren **32**

8 Aus Beispielen Komponieren lernen **32**
8.1 Suffix Bäume 33
8.2 Neuronale Netze 34
8.3 Markov Modell 35
8.4 Algorithmus 37
8.5 Pseudocode 39

9 Schlussbemerkung **46**

III Schluss **48**

10 Zusammenfassung **48**

11 Ausblick **48**

Literatur **53**

A Darstellung der Taxonomie **54**

B Implementierungen **54**

1 Einleitung

Die Musik hat den Menschen seit Urzeiten begleitet und ist heute so präsent wie kaum eine andere Kunstart. Stets war sie ein Ausdrucksmittel von Emotionen und verstand sich als eine internationale Sprache, die nicht auf Worte angewiesen ist. Im Laufe der menschlichen Entwicklung wandelten sich die Stile der Musik, ebenso wie ihre Instrumente und Kompositionsarten. Es entstanden Wissenschaften, die sich mit dem Musikbegriff auseinandersetzten und ihn zu erklären versuchten. Das Ziel war es, zu verstehen, was Musik ausmacht und was die grundlegenden Elemente sind, die akustische Ereignisse zu Musik machten. Heute sind einige dieser Fragen bereits beantwortet worden, aber viele weitere gilt es noch zu untersuchen. Mit dem Aufkommen der technischen Errungenschaften im Bereich der Informationsverarbeitung und dem fortschreitenden Verständnis für psychoakustische Phänomene der Wahrnehmung, ergeben sich neue Möglichkeiten tiefergehende Erforschungen des Musikbegriffs durchzuführen. Interessante Aufgabenfelder wie die Audioanalyse oder die automatisierte Komposition von Musik, im Bereich der Informatik, tragen dazu bei ein besseres Verständnis für Musik zu entwickeln. Während die Audioanalyse ein vergleichsweise stark ausgeprägter Forschungszweig ist, steckt die Komposition gewissermaßen noch in den Kinderschuhen. Da Letzteres eine Formalisierung der Kompositionsaufgabe fordern, die nicht leicht zu verwirklichen ist, scheint die erfolgreiche Realisierung eines automatisierten Kompositionssystems noch in weiter Ferne zu sein.

2 Motivation

Herauszufinden welche Aufgabengebiete in der Musik durch die Informatik angegangen werden und welche Verfahren zu ihrer Lösung eingesetzt werden, ist zentrales Ziel dieser Arbeit. Es sollen die wesentlichen Problemklassen bei Automatisierungsaufgaben in der Musik identifiziert und daraus eine hierarchische Struktur - eine Taxonomie konstruiert werden, die einen Überblick über die Problemgebiete und etablierte Lösungen liefern soll. Dies ermöglicht eine schnelle Einschätzungen der Komplexität neuer Fragestellungen, anhand bereits vorhandener Informationen und die Einordnung von Problemen und Algorithmen in entsprechende übergeordnete Aufgabenfelder der Taxonomie.

Neben dem breiten Überblick, den die Taxonomie bietet, wird zusätzlich ein gewähltes Kompositionssystem genauer untersucht, mit dem Ziel einen tiefen Einblick in seine Realisierung zu bekommen. Das dabei gewonnene Wissen wird in Form von kommentiertem Pseudocode angegeben.

Da die einzelnen Themenkomplexe mitunter sehr umfangreich sind, wird meist ein grober Überblick der einzelnen Verfahren gegeben und zentrale Gedanken der Lösungsansätze aufgezeigt. Für tiefergehende Studien wird stets ein Verweis auf die der Arbeit zugrunde liegende Literatur geboten.

3 Gliederung

Die Arbeit gliedert sich wie folgt: Zunächst wird in Kapitel I die „Taxonomie der Aufgaben in der Musik“ präsentiert und die Prinzipien hinter ihrer Konstruktion erläutert. Anschließend werden in Kapitel 5 und Kapitel 6 die beiden großen Bereiche der Analyse und der Synthese mit ihren, untergeordneten Komponenten beschrieben. Dabei werden beispielhaft Lösungsansätze für die angeführten Aufgabenfelder aufgezeigt. In Kapitel 8 folgt die Untersuchung eines Kompositionsalgorithmus aus [VDF04b], der mit Hilfe von neuronalen Netzen, Suffix-Bäumen und einem Markov-Modell befähigt wird aus gelernten Musikstücken, Neue zu komponieren. Beide Kapitel werden durch Schlussbemerkungen abgeschlossen. Die Arbeit endet dann mit einer Zusammenfassung und einem kleinen Ausblick.

Teil I
Taxonomie der Aufgaben in der Musik

Die im Folgenden präsentierte Taxonomie strukturiert die identifizierten Aufgabenstellungen hierarchisch, von der Wurzel bis zu den Blättern, von *allgemein* nach *speziell*. Ein Knoten ist die übergeordnete Kategorie der speziellen Unterkategorien an seinen Zweigen. Eine Kategorie entspricht einer konkreten Aufgabenstellung in der informatischen Musikwissenschaft.
Falls es nicht möglich ist, Aufgaben nur einem Oberbegriff zuzuordnen, findet die Einordnung gemäß ihrem primären Ziel statt. Daher können hierarchieübergreifende Zusammenhänge existieren, auch wenn Kategorien einem bestimmten Zweig der Taxonomie angehören. Die Aufgabe „Untersuche gegebene Musikstücken und komponiere anschließend ein neues Stück“ vereint Analyse und Synthese zu einem vollständigen Verfahren. Weil die Synthese jedoch das primäre Ziel darstellt, und die Analyse als Zwischenschritt zu sehen ist, wird dieses Verfahren der Synthese untergeordnet. Ähnliche Zusammenhänge sind auch in der Analyse zu finden. Hier werden in den speziellen Bereichen, Verfahren genutzt, die wurzelnah, also allgemeiner sind[1].

4 Bereiche der Taxonomie

Im Folgenden werden die einzelnen Bereiche der Taxonomie, an der Wurzel beginnend, präsentiert und die Aufgaben an den Knoten vorgestellt.

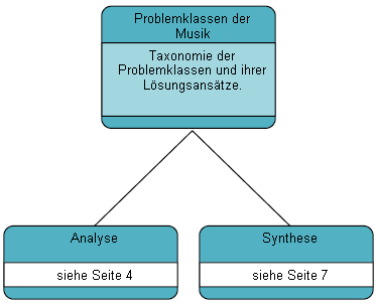

Abbildung 1: Ausgehend von der Wurzel der Taxonomie, lassen sich zwei grundlegende Aufgabegebiete definieren, die Analyse und die Synthese.

[1] Ohne das dabei Rekursionen entstehen.

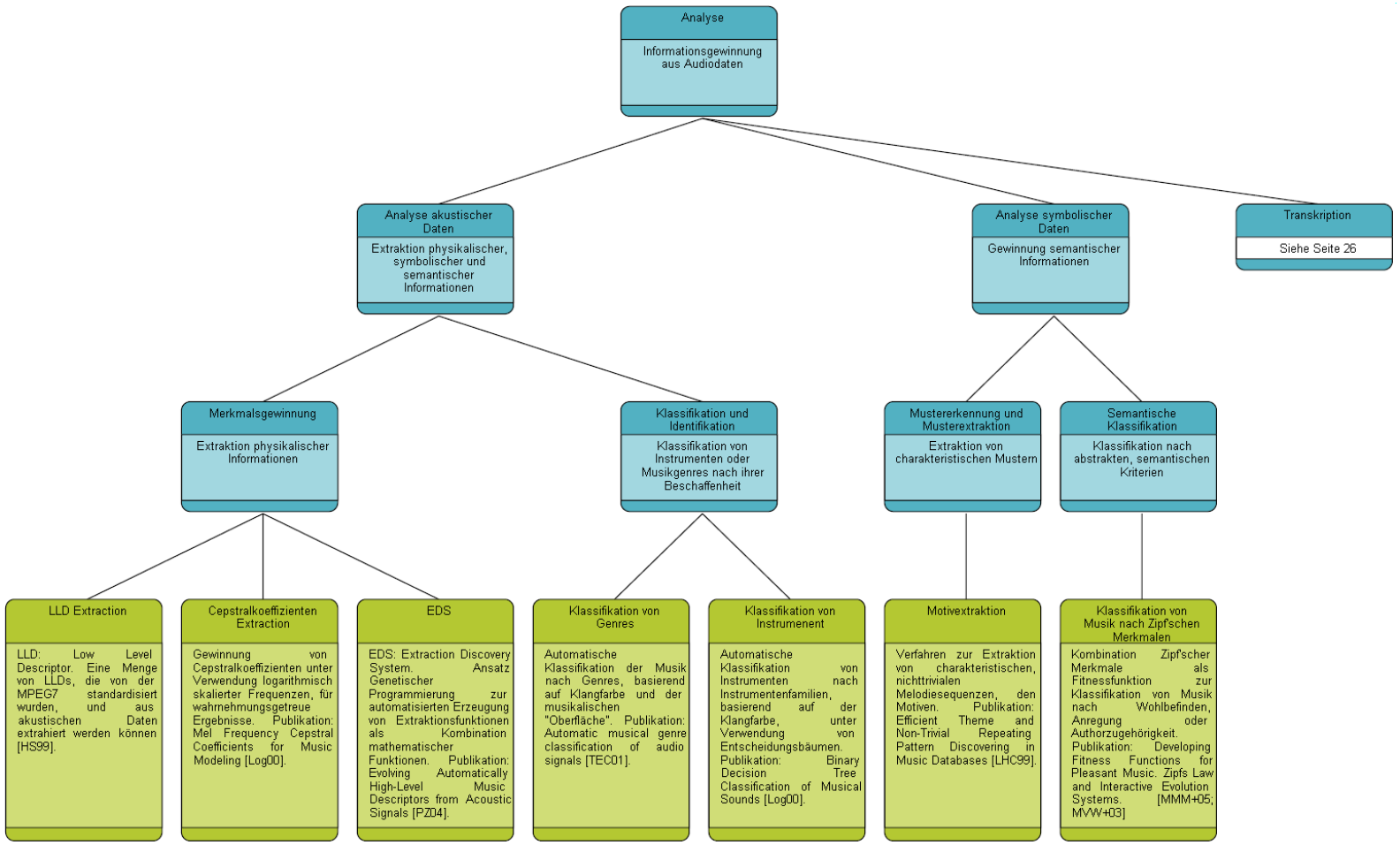

Abbildung 2: Die Analyse lässt sich in drei Bereiche unterteilen, die Analyse akustischer Daten, die Transkription und die Analyse symbolischer Daten. Während Ersteres sich primär mit der physikalischen Beschaffenheit akustischer Daten beschäftigt, wird in der Transkription versucht eine symbolische Darstellung der Musik aus akustischen Daten zu gewinnen. Die symbolische Analyse findet ausschließlich auf symbolischen Repräsentationen statt, um abstrakte semantische Informationen zu gewinnen. Siehe dazu Kapitel 5.

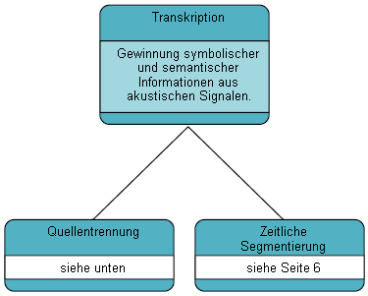

Abbildung 3: Die Gewinnung einer symbolischen Darstellung von Musik erfordert die Zerlegung polyphoner, akustischer Daten in ihre parallelen und konsekutiven Komponenten. Siehe dazu Kapitel 5.3.

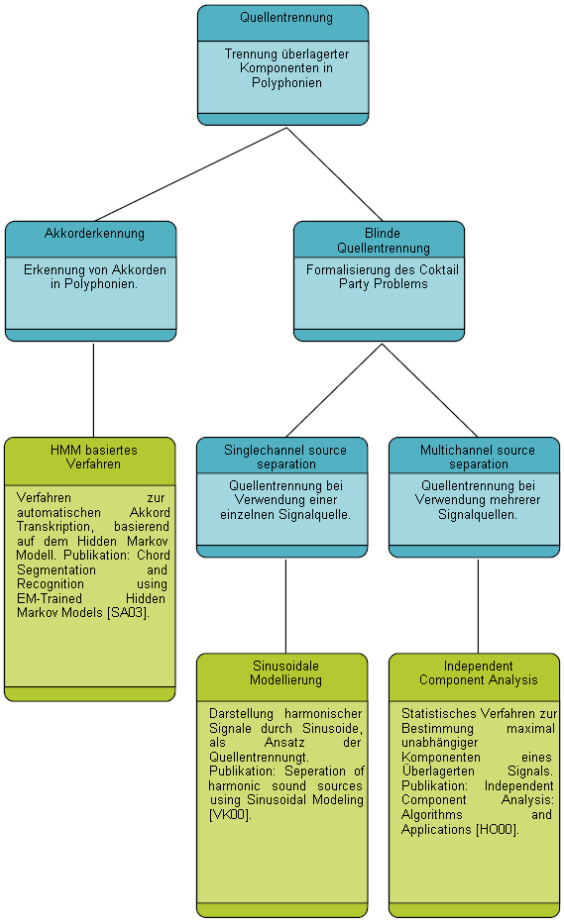

Abbildung 4: Die zentrale Aufgabe der Quellentrennung ist Zerlegung polyphoner, akustischer Signale in ihre einzelnen Komponenten. Unter Verwendung von Algorithmen der Spracherkennung und statistischer Methoden lassen sich einzelne Instrumente und Stimmen eines Musikstücks herausfiltern. Siehe dazu Kapitel 5.3.1.

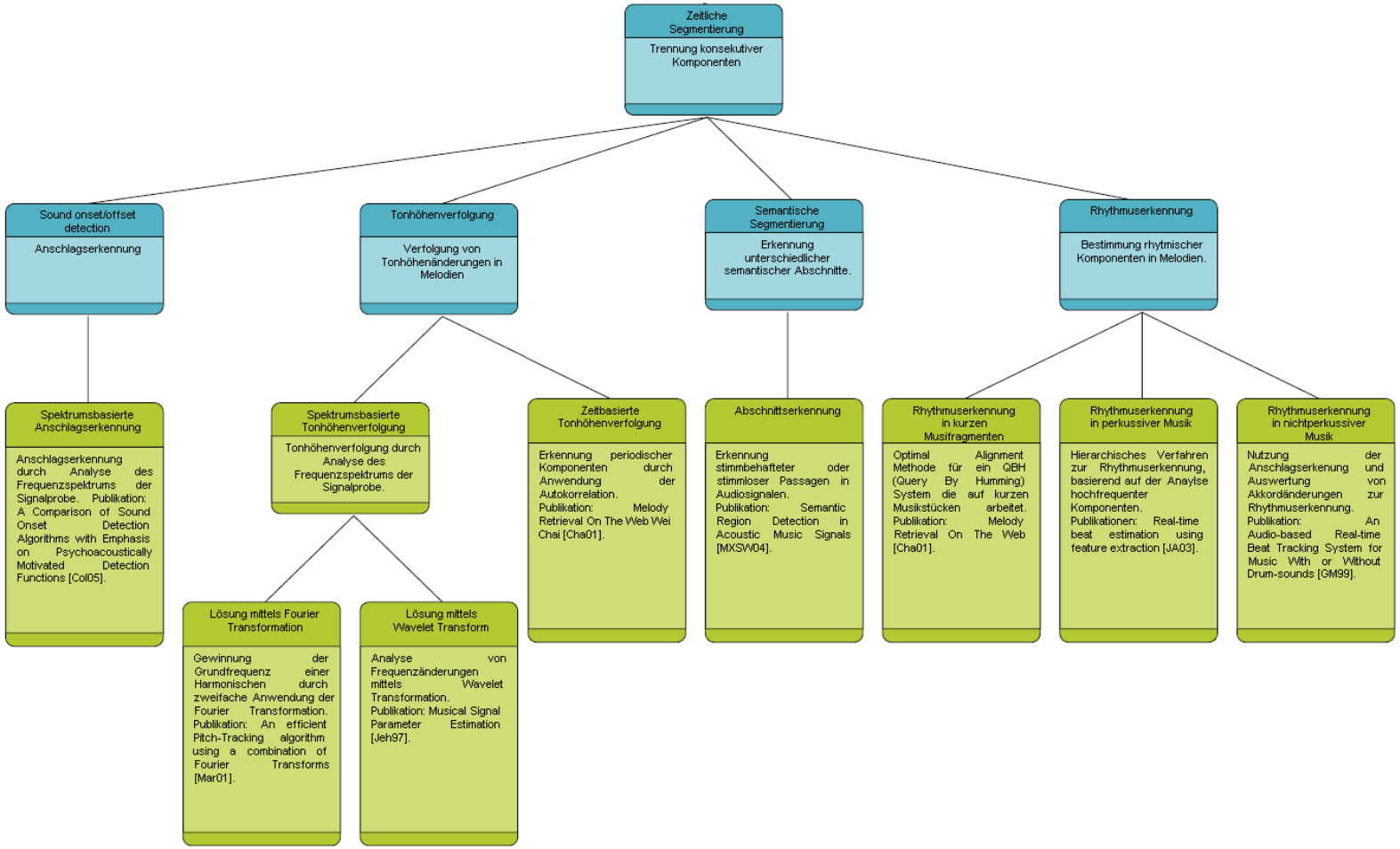

Abbildung 5: Unter zeitlicher Segmentierung lassen sich diejenigen Verfahren vereinen, die akustische Proben zeitlich analysieren und diese in konsekutive Komponenten partitionieren. Darunter fallen die Anschlagserkennung, die Tonhöhenverfolgung, die Rhythmuserkennung und die semantische Segmentierung. Siehe dazu Kapitel 5.3.2.

Abbildung 6: Die Klangsynthese und die Komposition bilden zusammen die beiden Syntheseaufgaben der Musik. Während die Klangsynthese sich primär mit der Generierung einzelener Klänge, unter Verwendung von natürlichen, elektronischen, digitalen Verfahren beschäftigt, ist die Komposition die Kreation komplexer, zusammengestetzter musikalischer Sequenzen dar. Dazu werden u.A. stochastische, deterministische, evolutionäre und neuronale Systeme eingesetzt. Siehe dazu Kapitel 6.

5 Analyse

Der vergleichsweise am besten erforschte Bereich in der IT-Musikwissenschaft ist die Analyse. Darunter versteht man die Analyse akustischer, symbolischer und auch physikalischer Informationen, also derer hierarchischer Strukturen aus denen Musik bestehen kann. Stellt man sich ein entsprechendes Schichtenmodell vor, so findet man die *Signalschicht* ganz unten, als physikalische Grundlage aller akkustischer Vorgänge. Hier sind Welleneigenschaften, Frequenzen und Amplituden wichtige Informationsträger. In der Schicht darüber befindet sich die *Klangschicht* die abstraktere Informationen wie Klangfarbe, Tonhöhen und Harmonien/Disharmonien tragen kann. Darüber befindet sich dann die *Symbolschicht*, die wiederum weiter abstrahiert und die die symbolische Darstellung von Melodien und ganzen Musikstücke erlaubt.

Nr.	Schicht	Informationen	Eigenschaften	Interpretation
3	Symbolschicht	Vertikale und horizontale Anordnung von Klängen (Komposition)	Rhythmus, Melodie, Harmonie	Musikstil
2	Klangschicht	Zusammengesetzte akustische Signale (Klänge)	Tonhöhe, Periode, Klangfarbe	Instrument, Stimme, etc
1	Signalschicht	Welleneigenschaften	Amplitude, Frequenz, Dauer	Sterile Töne

Abbildung 7: Die Unterprobleme der Analyse und der Synthese bewegen sich auf einer oder mehreren Abstraktionsebenen des vorgestellten Schichtenmodells.

5.1 Analyse akustischer Daten

In der akustischen Analyse bewegt man sich auf der physikalischen Schicht. Man versucht aus komplexen, akustischen Daten, Informationen zu gewinnen, die z.B. in der Rhythmuserkennung (Kapitel 5.3.2) oder der Segmentierung, den Teilbereichen der Transkription (Kapitel 5.3) verwendet werden können. Weiter ist es möglich, schon auf diesem Niveau eine Klassifikation diverser Klänge z.B. nach ihrer Zugehörigkeit zu bekannten Instrumentgruppen oder auch Stimmerkennung durchzuführen. Der große Bereich der Transkription beschäftigt sich damit, symbolische Informationen aus akustischen Daten zu gewinnen. Dieses ist z.B. für QBH-Systeme (Query by Humming) interessant, wo anhand einer kurzen (vom Menschen) gesummten Melodie, eine Suche nach bekannten Musikstücken in einer Datenbank gestartet werden kann. Schließlich ist es ebenfalls interessant, eine symbolische Darstellung aus Musikstücken zu gewinnen, die nur in verrauschter akustischer Form vorliegen, um sie anschließend wieder in verbesserter Qualität wiedergeben zu können.

5.1.1 Merkmalsgewinnung

Die Gewinnung bzw. Extraktion von Merkmalen beschäftigt sich mit der Gewinnung von charakteristischen Eigenschaften aus akustischen Signalen. Die gewonnenen Merkmale liefern atomare Aussagen über mathematische bzw. physikalische Eigenschaften eines Audiosignals. Eine Kombination bestimmter Merkmale kann sogar komplexe Aussagen über semantische Inhalte liefern.

5.1.2 Deskriptor Extraktion

Um einen standardisierten Satz von Merkmalen zu schaffen, wurde von der MPEG (Motion Pictures Expert Group) MPEG7 ins Leben gerufen. In diesem Projekt werden sog. „Low Level Audio Descriptoren“ (LLD) definiert, die in den Bereichen Audioanalyse und Content Retrieval verwendet werden können. Durch Auswertung gewisser Kombinationen von LLDs, lassen sich Aussagen über die physikalische Beschaffenheit der untersuchten Audiosignale machen, was semantisch relevante Informationen liefern kann. Deskriptoren werden beispielsweise zur Bestimmung von Rhythmen genutzt.
Folgende LLDs werden u.a in MPEG7 definiert:

- Amplitude sinusoidaler Komponenten
- Amplitude residualer Komponenten
- Spektrale Hülle sinusoidaler Komponenten
- Spektrale Hülle residualer Komponenten
- Harmonische Verzerrung
- Rauschen
- Spektraler Schwerpunkt
- Spektrale Neigung

Eine detaillierte Beschreibung der Deskriptoren lässt sich [HS99] entnehmen.

5.1.3 Cepstralkoeffizienten Extraktion

Ein weiteres Merkmal dass sich im Bereich der Spracherkennung etabliert hat, sind die Cepstralkoeffizienten. Ein Cepstrum ist die verdrehte Form des englischen Wortes „spectrum“ und kann als Spektrum des Spektrums bezeichnet werden. Sind große Oberwellenanteile im Spektrum eines Audiosignals enthalten, ist man mittels des Cepstrums in der Lage, eine Ermittelung der Grundfrequenz der

Oberschwingungen durchzuführen, auch wenn diese mit relativ kleinen Amplituden vorhanden sind. Einfach ausgedrückt, kann das Cepstrum den „Ursprung“ periodischer Welligkeiten in einem Spektrum aufzeigen. Die Cepstralkoeffizienten errechnen sich nun als Koeffizienten bei der Entwicklung von Fourierreihen, während eines Teilrechenschritts der Cepstral Analyse. Da in der menschlichen Sprache niedrige Frequenzen den größeren Anteil an Informationen tragen, und höhere Frequenzen den geringeren Anteil, skaliert man das Frequenzspektrum logarithmisch, um ein wahrnehmungsgetreueres Ergebnis zu erhalten. Dieses wird bei der Berechnung von MFCCs (Mel Frequency Cepstral Coefficients) berücksichtigt [Log00].

5.1.4 Extraction Discovery System

Eine andere Möglichkeit Merkmale zu extrahieren bietet das EDS „Extraction Discovery System“ [PZ04]. Es verwendet nicht wie bisher erwähnt, Kombinationen von Deskriptoren sondern bemüht genetische Programmierung um Extraktoren zu finden, die bei Anwendung in der Lage sind, Merkmale aus Audiodaten zu gewinnen. Der GA arbeitet auf einer Population von festgelegten Operatoren (ma-

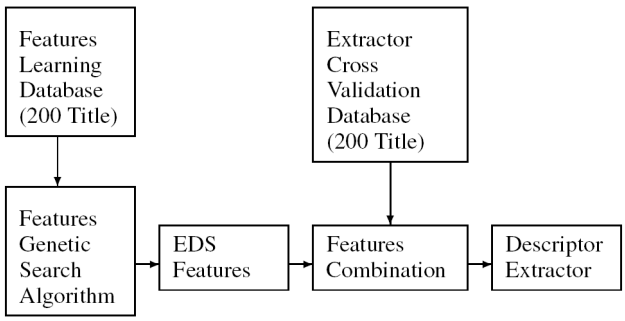

Abbildung 8: EDS System schema [PZ04]

thematische Funktionen), deren Kombination zu evolvieren ist und die sehr spezifische Informationen aus Audiodaten gewinnen kann. Die Fitness der Funktionen wird durch Korrelation zwischen den Funktionswerten und den a priori manuel gesammelten Werten bestimmt. Da die Auswertung der vielen Funktionen z.T. sehr rechenaufwendig ist und pro Suche eine Trainings-Datenbank mit Musikstücken verarbeitet werden muss, wird die Suche von einem heuristischen Algorithmus geleitet. Zusätzlich werden Vorkehrungen getroffen, unnötig komplizierte Funktionen zu vereinfachen und Mehrfachberechnungen identischer Funktionen durch caching zu vermeiden.

5.2 Klassifikation und Identifikation

Die in Kapitel 5.1.1 vorgestellten Methoden erlauben bereits in der physikalischen Analyse von Audiodaten einige interessante Einsatzmöglichkeiten. Es ist z.B. möglich, Klassifikationen von Instrumenten oder Musikgenres durchzuführen. Zwar ist die Klassifikation von Musikgenres in der symbolischen Analyse (Kapitel 5.4) sinnvoller untergebracht, aber die physikalische Schicht bietet hier schon viele Informationen, um der Aufgabe zu begegnen. Zudem ist bekannt, dass der Mensch Genres in 250ms-langen Musikstücken sicher und korrekt erkennen kann [TEC01], was bedeutet, dass genug Informationen vorhanden sein müssen um diese Aufgabe (mit genügend Vorwissen) zu lösen. Dasselbe gilt für die Zuordnung von Instrumenten zu ihren Instrumentenfamilien. Im Folgenden werden Lösungsansätze für die beiden Aufgaben aufgeführt.

5.2.1 Klassifikation von Instrumenten

In ihrer Arbeit „Binary Decision Tree Classification of Musical Sounds“ [JA99] stellen K. Jensen und J.Arnspang eine Methode vor, die mit Hilfe einer hierarchischen Klassifikation im binären Baum die Zugehörigkeit, bzw. den Verwandschaftsgrad von akustischen Signalen zu bekannten Instrumenten bestimmen kann. Dieses kann in der Transkription verwendet werden, ist aber schon auf der Ebene der Audioanalyse möglich. Die entscheidenden Klassifikationsattribute werden aus additiven Attributen gewonnen und können grob in spektrale Hüllkurven-Attribute, Hüllkurven-Attribute, und Rausch-Attribute unterteilt werden. Der Binärbaum wird auf Grundlage einer umfangreichen Instrumentendatenbank konstruiert, indem eine Menge von Fragen gestellt wird, deren Ausgang über die Spaltung der Datenmenge in Gruppen und Untergruppen entscheidet. Durch Iteration solcher Klassifikationen entsteht ein Binärbaum, an dessen Blättern Instrumentgruppen stehen. Anschließend kann entschieden werden ob ein unbekanntes Instrument z.B. zur Gruppe der Saiteninstrumente oder Blasinstrumente gehört.

5.2.2 Klassifikation von Genres

Ähnlich wie in der Klassifikation von Instrumenten werden bei der Klassifikation von Genres ausgewählte Audiomerkmale dazu verwendet, Klassifikationen durchzuführen. G. Tzanetakis, G. Essl und P. Cook stellen in diesem Zusammenhang die Analyse der „musikalischen Oberfläche“ vor [TEC01]. Ein neundimensionaler Merkmalsvektor, der die „musikalische Oberfläche“ von Musikstücken repräsentiert, wird aus selbigen errechnet und anschließend statistischen Mustererkennungsklassifikatoren zugeführt. Diese führen dann schrittweise eine hierarchische Zuordnung zu Musikgenres durch. Die Klassifikatoren wurden mit Musikstücken aus dem Radio, dem Internet und von CDs trainiert. Neben der

Genreklassifikation erlauben sie eine Unterscheidung von Musik und Sprache und erkennen geschlechtsspezifische Sprachunterschiede mit gutem Erfolg.

5.3 Transkription

Wie oben erwähnt, bildet die Transkription einen großen Bereich der akustischen Analyse. Diese Aufgabe umfasst viele Probleme, die Gegenstand aktueller Forschung sind. Das Ziel ist die Gewinnung abstrahierter, symbolischer Strukturen aus komplexen akustischen Daten. Es geht also um Kapselung von Signalen in Noten und Akkorden entsprechend eines definierten Notensystems. Grundsätzlich lassen sich zwei unterschiedliche Komponenten der Aufgabe nennen. Die erste (vertikale) Komponente ist die Trennung überlagerter Signale und ist formell als das Partyproblem bekannt. Die zweite (horizontale) Komponente ist die Segmentierung akustischer Signale in der Zeit, wie sie z.B. in der Rythmuserkennung (engl. rhythmtracking) auftritt. Auch wenn diese beiden Aufgaben grundsätzlich zwei verschiedene Aspekte der Analyse darstellen, so sind sie doch nicht von einander getrennt. Musik ist ein zeitliches Phänomen, da sie eine Abfolge akustischer Ereignisse darstellt und nicht als ganzheitliches Konstrukt, wie beispielsweise ein Gemälde, angesehen werden kann. Die beiden Aufgaben können nur zusammen, sinnvoll angegangen werden.

5.3.1 Quellentrennung

Als einer der beiden Bereiche der Transkription beschäftigt sich die Quellentrennung, auch bekannt als „Auditory Stream Segregation“, mit der Zerlegung zusammengesetzter bzw. überlagerter, akustischer Signale. Dieses ist z.B. nötig um aus einem Musikstück die einzelnen Instrumente und Stimmen herauslösen zu können, oder auch überlagerte Noten-Anschläge (Akkorde) eines Instrumentes zu erkennen. Dieses Problem ist offiziell als das Coktail Party Problem bekannt und beschreibt die Sachlage wie folgt. Auf einer Party befinden sich viele Menschen, die quer durcheinander reden. Alle Diskutierenden können sich unter einander verstehen, auch wenn sich sich nicht unbedingt sehen. Einem menschlichen Zuhörer ist es also möglich die Aufmerksamkeit auf eine einzelne Stimme zu lenken, und aus den sich überlagernden Tönen und Geräuschen eine bestimmte Stimme heraus zu filtern. Dieses fällt dem Menschen vergleichsweise leicht, da er durch die Evolution über entsprechende Mechanismen der Wahrnehmung ausgestattet wurde. Allerdings erweist sich dies in der Wissenschaft als eine schwierige algorithmische Aufgabe. Im Folgenden werden zwei Möglichkeiten vorgestellt, diese Aufgabe anzugehen.

Akkorderkennung

Alexander Sheh und Daniel P.W. Ellis liefern in [SA03] ein System für die automatische Akkorderkennung. Dieses verwendet Techniken der Spracherkennung sowie das in dem Zusammenhang populär gewordene Hidden Markov Modell, das in Kapitel 8 detailliert besprochen wird. Als Erstes wird das Audiosignal in den Frequenzraum transformiert. Dann erfolgt eine intervallweise Übertragung der spektralen Intensitätswerte in den PCP-Raum (Pitch Class Profile). Man kommt auf diese Weise zu einer geeigneten Darstellung des Audiosignals, dessen überlagerte Komponenten (z.B. einzelne Töne eines Akkords) in einzelne Töne zerlegt und in dem 12-Dimensionalen „PCP-Vektoren“ gespeichert werden. Die auf diese Weise gewonnen Vektoren werden nun dazu verwendet, ein HM Modell zu konstruieren. Dabei wird als Trainingsalgorithmus das „Expectation Maximization“ Verfahren benutzt. Schließlich wird das Viterby-Verfahren angewandt, um eine Folge von Zuständen des HMM zu finden, die dem Audiosignal entsprechen würde. Damit hat man eine symbolische Darstellung der akustischen Daten gewonnen.

Blinde Quellentrennung

Die Blinde Quellentrennung (engl.: Blind Source Seperation) ist die Formalisierung des Cocktail Party Problems. Es wird in zwei Unterprobleme kategorisiert, die „Singlechannel Source Separation“ und die „Multichannel Source Separation“. Sie unterscheiden sich darin, wieviele unabhängige Signalquellen das überlagerte Signal liefern. So lässt sich eine Stimme aus der Menschenmenge erheblich leichter heraus zu filtern, wenn man dem Algorithmus Signale zweier Mikrofone, die sich an verschiedenen Orten befinden, zur selben Zeit anbieten kann. Da dies nur selten der Fall ist, beschäftigt sich die Singlechannel Source Separation mit der Variante einer einzelnen Signalquelle.

Dies ist z.B.bei der Aufnahme eines klassischen Musikstücks der Fall. Die einzelnen Instrumente zu extrahieren, ist eine schwierige Aufgabe, zu der es noch keine allgemeine Lösung gibt. Ein etwas eingeschränkter Lösungsansatz wird allerdings von T. Virtanen und A. Klapuri in [VK00] präsentiert. Die Einschränkung besteht in der Annahme, dass die ursprünglichen Signale harmonisch sind und sich in ihrer Grundfrequenz unterscheiden. Dann erlaubt der Algorithmus die Spaltung des komplexen Signals in seine Grundkomponenten, die mit ihren Originalen gut übereinstimmen. Das Vorgehen bei der Berechnung ist wie folgt: Zuerst wird die Signalprobe mittels sinusoidaler Modellierung, als Schar von Sinusoiden dargestellt. Dann werden eventuelle, kalkulationsbedingte Störungen innerhalb der Sinusoide durch Interpolation beseitigt.

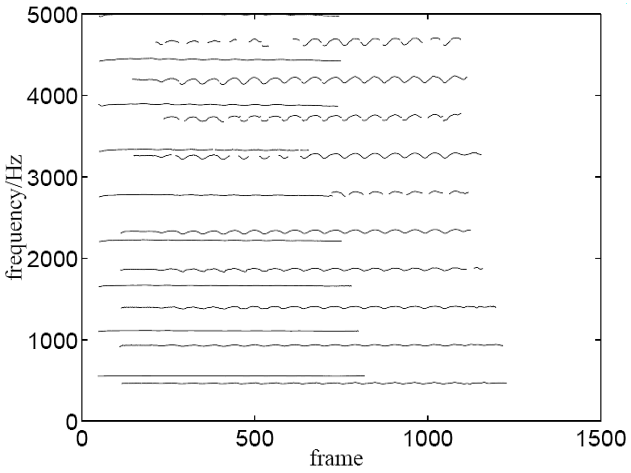

Abbildung 9: Sinusoide eines aus Oboe und Violine bestehenden Signals [VK00].

Anschließend werden ähnliche Sinusoide als zusammengehörend identifiziert und den Signalquellen zugeordnet.Eine erneute Synthese der Sinusoide liefert schließlich die unabhängigen Signale.

Im Gegensatz dazu existieren bei der Multichannel Source Separation mindestens zwei Signalquellen, die über denselben Zeitraum hinweg, unabhängig Signale aus verschiedenen Perspektiven liefern. In diesem Fall lässt sich das ICA-Verfahren (Independent Component Analysis) [HO00] anwenden. ICA ist eine allgemeine statistische Methode, um aus randomisierten Daten durch lineare Transformationen versteckte Komponenten zu ermitteln, die am schwächsten mit einander korrelieren, also von einander maximal unabhängig sind. Das Verfahren stammt ursprünglich aus der Elekrotechnik, wird aber mittlerweile sehr vielseitig eingesetzt. Angenommen wird zunächst, dass die überlagerten Komponenten nicht normalverteilt und statistisch unabhängig sind. Wenn man n solcher Signale als $s_1..s_n$ bezeichnet, dann lässt sich die Überlagerung als Summe von Signalmixturen darstellen:

$$x_i = a_{i1}s_1 + a_{i2}s_2 + a_{i3}s_3 + ... + a_{in}s_n, \forall i$$

Die i-te Überlagerung x_i ist also die Summe von n einzelnen Signalmanipulationen. Fasst man Signale $s_1..s_n$ im Vektor $\vec{s}$, Mixturen $x_1..x_n$ im Vektor $\vec{x}$ und die Koeffizienten $a_{i1}..a_{in}$ in der Matrix $\mathbf{A}$ zusammen, dann lässt sich die Überlagerung als ein Matrix-Vektor Produkt darstellen:

$$\vec{x} = \mathbf{A}\vec{s}$$

Die Lösung besteht nun darin, aus dem bekannten Vektor $\vec{x}$ die Matrix $\mathbf{A}$ und die Signale $\vec{s}$ zu gewinnen. Unter den oben genannten Einschränkungen kann $\mathbf{A}$

geschätzt werden. Durch die anschließende Berechnung der inversen Matrix

$$\mathbf{T} := \mathbf{A}^{-1}$$

und der Multiplikation mit $\vec{x}$ lassen sich dann die Signale $\vec{s}$ schätzen. Für den praktischen Einsatz existieren effiziente Implementierungen des ICA-Algorithmus.

5.3.2 Segmentierung

Bisher wurde der große Bereich der Quellentrennung besprochen. Im Gegensatz dazu beschäftigt sich das Gebiet der Segmentierung, auch als „Successive Component Analysis“ bekannt, mit der Analyse zeitlich aufeinanderfolgender Ereignisse. Die Schwierigkeit der Trennung gleichzeitiger Ereignise wird hier i.d.R. gemieden. Stattdessen wendet man sich der Analyse monophonischer Musik zu. Dieses soll auch für die weiteren Ausführungen gelten. Aufgaben der Successive Component Analysis sind die Rhythmuserkennung (Rhythmtracking), die Tonverfolgung (Pitchtracking) und die Klang-Anschlagserkennung (Sound Onset Detection). Auch die Spracherkennung gehört in diesen Bereich, da die Trennung der gesprochenen Laute, die Segmentierung zeitlicher Ereignisse darstellt.

Klang-Anschlagserkennung

Die (Klang-)Anschlagserkennung (engl.: Sound onset/offset detection) ist deshalb wichtig für die Transkription, weil es für die Übertragung akustischer Informationen in die symbolische Darstellung, unabdingbar ist Segmentierungen des Signals durchzuführen, um aus den Segmenten Atomare Komponenten gewinnen zu können. Die Erkennung der Zeitpunkte, wann die Saite einer Gitarre angeschlagen wurde und wann sie wieder verklingt, stellt eine solche Segmentierung dar. In der Publikation [Col05] der Audio Engineering Society werden einige der derzeit vorhandenen Algorithmen vorgestellt und mit einander verglichen. Die meisten Algorithmen der Anschlagserkennung bestehen aus zwei Teilen. Im ersten Teil liefert die Erkennungsfunktion Signale über die Zustandsänderung einer akustischen Probe. Der zweite Teil wertet dann das Signal aus, und berechnet aus den Positionen der Signal-Peaks den Zeitpunkt des Klang-Anschlags in der Probe. Erkennungsfunktionen können auf verschiedenen Frequenzbändern und mit unterschiedlicher Auflösung arbeiten. Auch ist es möglich, mehrere Erkennungsfunktionen für die bessere Detektion zu kombinieren. Masri und Bateman definieren zunächst einen Hochfrequenzanteil HFC (High Frequency Content) als die gewichtete Summe spektraler Intensitäten [MB96]. Das Verhältnis von HFC-Werten aufeinanderfolgender Probenabschnitten definiert dann ihre Erkennungsfunktion. Diese liefert immer dann große Werte (Peaks), wenn der Intensitätsunterschied

zwischen zwei Abschnitten groß ist, was auf den Anschlag eines Klangs hindeuten kann. Es ist auch möglich, die Differenz der HFC-Werte für die Erkennungsfunktion zu nutzen. Dies tun Jensen und Andersen in [JA03]. Dazu modifizieren sie aber zusätzlich die Definition des HFC.
Das anschließende erkennen der Peaks in den Funktionswerten der Erkennungsfunktion kann durch den Vergleich mit absoluten Schwellwerten erfolgen oder mit Hilfe von sog. adaptiven Peak-Pickern, die Vergleiche mit einem dynamischen Schwellwert durchführen, der sich aus der kummulativen Durchschnittsbildung der Intensitäten ergibt.

Tonhöhenverfolgung

In der Musik gibt es bis auf Ausnahmen keine reinen Töne, die aus nur einer Frequenz bestehen. Musik lebt von harmonischen Tönen, also Tönen die sich aus einer Grundfrequenz und vielen Harmonischen zusammensetzen. Dies sind Vielfache der Grundfrequenz. Sie verleihen dem Ton seine Klangfarbe. Gleichzeitig stellen sie aber eine Hürde für die automatisierte Tonhöhenverfolgung (engl.: Pitch Tracking) dar, da sie instrumentenspezifische Informationen tragen, und mit der Grundfrequenz überlagert sind. Das Pitch Tracking beschäftigt sich als eine der wichtigsten Zweige der Transkription mit der Erkennung und Verfolgung von Tonhöhen. Genauer gesagt geht es um die Bestimmung von Grundfrequenzen harmonischer Signale und die anschließende Zuweisung entsprechender Tonhöhen, mit dem Ziel der Gewinnung einer symbolischer Repräsentation. Ähnlich wie der Mensch im Stande ist, Tonhöhen in einem gespielten oder gesungenen Musikstück zu erkennen und wiederzugeben, soll es für einen Algorithmus möglich, sein die Grundfrequenzen gespielter Instrumente zu bestimmen und diese als Tonhöhen auszugeben. Dafür existieren mittlerweile effiziente und präzise Implementierungen. Es lassen sich dabei zwei Gruppierungen bilden. Die Erste unternimmt Spektralanalysen des Signals, also sein Frequenzspektrum. Die Zweite untersucht das Signal direkt. Beide sind für unterschiedliche Zwecke gut geeignet. Das hängt davon ab welche Informationen zur Verfügung stehen und ob der Algorithmus in Echtzeit laufen soll.

Spektrumsbasierte Tonhöhenverfolgung mittels Fourier Transformation

Sylvian Marchand präsentiert in [Mar01] seinen Ansatz für das Pitch Tracking, der sehr gute Ergebnisse liefert. Das Prinzip ist die Auswertung der „Fouriertransformierten einer Fourier-transformierten des Audiosignals“. Die erste Fourier-Transformierte zerlegt das Audiosignal in Frequenzkomponenten und stellt diese in einem Frequenzkamm dar. Dessen Peaks entsprechen den harmonischen Obertönen, die das Vielfache ihrer Grundfrequenz bilden. Die Fourier-Transformation dieser Fourier-Transformierten liefert dann wieder eine Folge von Peaks, wo-

bei der erste Peak der Grundfrequenz entspricht. Auf diese Weise kann man die Grundfrequenz auch dann bestimmen kann, wenn im ursprünglichen Audiosignal einige Obertöne gar nicht vorhanden sind oder die Grundfrequenz selbst fehlt (virtuell ist). Auch die Tatsache das diese Methode in Echtzeit erfolgreich getestet wurde und in ihrer Zuverlässigkeit den alternativen Ansatz der Autokorrelation [AD99] geschlagen hat, spricht für sich.

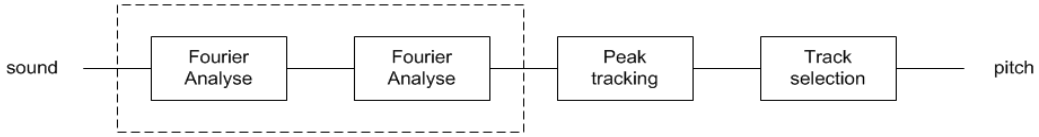

Abbildung 10: Systemübersicht [Mar01].

Spektrumsbasierte Tonhöhenverfolgung mittels Wavelet Transformation

Zur Fourier Transformation bildet die Wavelet Transformation eine gute Alternative [Jeh97]. Wavelets bieten die Möglichkeit, mathematische Funktionen verlustfrei, hierarchisch in Komponenten zu zerlegen bzw. darzustellen. Dabei kann der Detaillierungsgrad so gewählt werden, dass die Funktion von grob bis sehr fein aufgelöst wird. Es können im Gegensatz zur Kurzzeit Fourier Transformation (STFT) Funktionen mit tiefen und hohen Frequenzen gleich gut dargestellt werden. Darum sind sie für die Darstellung von instabilen und randomisierten Signalen, z.B. akustischen Signalen, gut geeignet. Um die Wavelets nun für das Pitch Tracking verwenden zu können, bedient man sich einer aus der digitalen Bildverarbeitung stammenden Methode der Kantenfindung. Es hatte sich nämlich gezeigt, dass bei der Wavelet Transformation von Bildern, lokale Maxima in der Transformierten zu beobachten waren, an der Stelle wo sich scharfe Kanten im Bild befanden. Eine scharfe Kante im Bild stellt einen krassen Farbwechsel dar. Dieses entspricht aber einem deutlichen Frequenzenwechsel, der durch die Wavelet Transformation detektiert wird. Analog lässt sich die Tonhöhenänderung als Frequenzänderung in einem akustischen Signal beschreiben. Da es hier aber nicht sinnvoll ist über alle möglichen Frequenzen zu analysieren, wird auf das Signal zusätzlich ein Wavelet-Bandpassfilter angewandt. Die Transformation gibt dann Auskunft darüber, an welcher Stelle eine Tonhöhenänderung stattfindet. Da hier noch keine absoluten Frequenzwerte geliefert werden, muss dieses im anschließenden Schritt noch erfolgen. Das gefilterte Signal wird also zwischen den Stellen der Frequenzwechsel untersucht. Dabei errechnen sich die Frequenzwerte für einen festen Zeitpunkt t aus der vorliegenden Periodendauer, also aus dem Ab-

stand zweier lokaler Maxima der Amplitude vor und nach t. Die auf diese Weise gewonnen Informationen können dann problemlos in die symbolische Darstellung überführt werden.

Pitch Tracking in Zeitbereich

Signalanalysen im Spektralbereich und im Zeitbereich haben verschiedene Vor- und Nachteile. Dieses hängt mit diversen Voraussetzungen, Annahmen und Möglichkeiten zusammen die in dem einen oder anderen Fall bestehen. Die am weitesten verbreitete, im Zeitbereich arbeitende, Methode für das Pitch Tracking ist die Autokorrelation. Es ist wiederum eine allgemeine mathematische Methode, um u.U. versteckte periodische Komponenten eines Signals zu identifizieren [Wikc]. Dabei wird das zu untersuchende Signal mit einer zeitversetzten Kopie seiner Selbst gefaltet und auf diese Weise der „Verwandschaftsgrad“ zeitversetzter Signalabschnitte bestimmt. Nichtperiodische Komponenten des Signals wie z.B. das Rauschen werden auf diese Weise gedämpft, und periodische Komponenten verstärkt. Faltet man Bereiche gleich bleibender Frequenz mit einander, so ist das Ausgangssignal gegenüber dem Eingangssignals kaum verändert. Wenn aber Bereiche unterschiedlicher Frequenz mit einander gefaltet werden, ist der Verwandschaftsgrad im einfachsten Fall gleich Null und die Autokorrelationswerte sinken entsprechend. Diese Information kann dann dazu verwendet werden, die genaue Stelle des Frequenzsprungs zu bestimmen. Die absoluten Frequenzwerte werden dann wie im vorangehenden Abschnitt berechnet. Für konkrete Implementierungsdetails sei an dieser Stelle auf [Cha01] verwiesen.

Semantische Segmentierung und Rythmuserkennung

Ein weiterer wichtiger Teil der Successive Component Analysis ist die Semantische Segmentierung. Sie beschäftigt sich mit der Einteilung von akustischen Signalen/Musik in semantische Abschnitte [MXSW04]. Dazu gehört die Einteilung in stimmbehaftete und stimmlose Abschnitte, oder die Erkennung von Rhythmen innerhalb eines Musikstücks. Ein mögliches Einsatzziel könnte z.B. die Aufzeichnung von Musikstücken, aus Rundfunkübertragungen sein, bei denen oft störende Kommentare und Werbung als solche identifiziert, und nicht aufgezeichnet werden. Daneben ist die Rhythmuserkennung wesentlicher Bestandteil der Transkription. Im Folgenden wird auf die Rhythmuserkennung näher eingegangen.

Rhythmuserkennung

Musik hat in den meisten Fällen einen Ryhtmus, der vom zugrundeligende Takt abhängig ist. Der in der westlichen Welt, am meisten verbreitete Takt ist der 4/4 Takt. Im Osten wird eher der 5/8 Takt präferiert. Daneben gibt es den vornehm-

lich bei Walzern anzutreffenden 3/4 Takt. Darüber hinaus gibt es auch andere, sehr seltene Taktungen. Das abzudeckende Spektrum ist also sehr groß, und eine Rhythmuserkennung für alle möglichen Musikstile ist wahrscheinlich schwer zu realisieren. Momentan findet eine Beschränkung der Methoden und Algorithmen auf westliche Stile statt. Insbesondere existiert eine Zweiteilung der Aufgabe in die Rhytmuserkennung an Hand von Schlagzeugschlägen, sog. „Beat Tracking“ und die Rhytmuserkennung bei Musikstücken ohne Schlagzeug. Die erste Variante ist leichter zu realisieren, da das Schlagzeug oft den Takt direkt vorgibt und in einem bestimmten Frequenzbereich vorzufinden ist. Wenn aber kein Schlagzeug eingesetzt wird, z.B. in einem Violine Solo, müssen komplexere Berechnungen angestellt werden, um rhythmische Strukturen zu erkennen.

Rhythmuserkennung in perkussiver Musik

Einen Algorithmus, der die Rhythmuserkennung sogar in Echtzeit ermöglicht, liefern Kristoffer Jensen und Tue Haste Andersen in [JA03]. Ihr Ansatz basiert auf der in Kapitel 5.3.2 bereits erwähnten Extraktion des HFC und der Ton-Anschlagserkennung. Zunächst wird mit Hilfe des extrahierten HFC, die Berechnung der Tonanschläge von Schlagzeug Instrumenten durchgeführt. Darauf aufbauend wird mit Hilfe eines Beat-Wahrscheinlichkeits-Vektors der wahrscheinlichste Beatintervall zwischen den HFC-Peaks geschätzt. Für die Berechnung eines Intervalls werden die vorangegangenen Intervalle mit einbezogen und der Vektor immer wieder aktualisiert. Dies dient der Kompensation ausbleibender Beats und der Unterdrückung „falscher“ Beats. Auf diese Weise kann eine präzise und stabile Echtzeit-Rhythmuserkennung realisiert werden. Immerhin wurden Beatintervalle bei sechs von zehn zufällig ausgewählten, populären Musikstücken richtig geschätzt. Auch wurde der Algorithmus bereits in zwei Software Produkten implementiert.

Rhythmuserkennung in nichtperkussiver Musik

Der Mensch ist im Stande, auch ohne die Existenz von Bass-Schlägen und Schlagzeug, den Rhythmus musikalischer Stücke zu erkennen und diesen zu Klatschen oder zu Schnippsen. Dies führt im Falle von Masataka Goto und Yoichi Muraoka [GM99] zur Annahme, dass alleine die Melodieänderung und die Progression von Akkorden, schon genug Informationen für eine verlässliche Rhythmusgewinnung liefern kann. Auf dieser These aufbauend, realisierten Goto und Muraoka ein System zur automatischen Rhythmuserkennung für Musikstücke ohne Schlagzeuginstrumente. Nach ihrem Top-Down Verfahren werden Musikstücke mit Hilfe der FFT spektral transformiert und die Anschlagszeiten berechnet. Durch die Annahme eines zugrunde liegenden 4/4 Taktes können die Anschlagszeiten für eine Segmentierung in Viertel- und Achtelnoten verwendet werden. Anschließend erfolgt

eine Aufteilung des Frequenzspektrums in Bänder und die Erzeugung von Histogrammen Bandabhängiger Signalintensitäten. Dieses geschieht für jeden Takt. Von der Tatsache ausgehend, dass Akkordänderungen meistens im Takt erfolgen und nicht zwischendurch, können adjazente Histogramme auf ihre Diffenrenzen hin untersucht werden, und können Schlüsse auf den Beat-Intervall gezogen werden. In der konkreten Implementierung wird ein Multiagentensystem verwendet, das die Anschlagserkennung und die Verfolgung von Akkordänderungen auf paralleler Ebene erlaubt. Mehrere, unterschiedlich parametrisierete Agenten liefern ihre Thesen, über die Anschlagszeiten und Rhythmuskalkulationen einem Managementsystem, dass entgültige Entscheidungen über Rhythmusausgaben trifft.

5.4 Analyse symbolischer Daten

Bisher wurden Aufgaben im Feld der Analyse akustischer Daten diskutiert. Es wurde besprochen, wie physikalische Eigenschaften von akustischen Signalen dazu verwendet werden können, semantische Informationen zu gewinnen. Der große Bereich der Transkription mit dem Ziel der Überführung akustischer Darstellung von Musik in die symbolische Darstellung wurde behandelt. Nun folgt die Diskussion über Problemfelder, die bei der Analyse von Musik in symbolischer Repräsentation existieren. Ausgangspunkt ist das Vorhandensein aller relevanter Informationen zur erfolgreichen Synthese, also der Erzeugung der akustischen Darstellung von Musik. Wir bewegen uns nur noch in der obersten Schicht im Schichtenmodell aus Kapitel 5 und untersuchen Fragestellungen der semantischen Analyse.

5.4.1 Mustererkennung und Musterextraktion

Durch die rasant ansteigende Menge an Musik im Internet, entsteht ein großer Bedarf an automatischen Musik-Suchsystemen, die an Hand einer gesummten Melodie, das dazugehörige Musikstück aus einer Datenbank wiederzufinden im Stande sind. „Content based Audio Retrieval“ und „Query by Humming“[2] sind hier Schlüsselbegriffe. Oft passiert es, dass weder der Interpret, noch der Titel eines Stücks bekannt sind, sondern nur noch Teile der Melodie vorhanden sind. Die Schwierigkeit die fehlenden Informationen nun wieder zu gewinnen, liegen in der Erkennung der, vom Menschen gesummten Melodie-Passage, und der Zuordnung zu einem in der Datenbank vorhandenen Musikstück. Vorausgesetzt wird hier, dass die Stücke in symbolischer Darstellung vorliegen und für Suchanfragen vorbereitet wurden. Damit ist gemeint, dass jeweils charakteristische Passagen

[2]QBH-Systeme haben die Aufgabe Melodien anhand von gesummten Passagen zu erkennen, um durch entsprechende Suchanfragen das zugehörige Stück in einer Datenbank wiederzufinden.

aus den Stücken extrahiert wurden und für einen Vergleich herangezogen werden können. Eine solche Vorverarbeitung ist durch den Menschen zwar machbar, aber für einige Millionen Titel nicht wirtschaftlich. Darum werden Methoden der automatischen Mustererkennung und Musterextraktion gesucht. Während der erste Teil, die Melodieerkennung als ein Problem der Transkription angesehen werden kann, besteht die Schwierigkeit des zweiten Teils in der Extraktion semantisch aufschlussreicher Daten aus den gespeicherten Titeln. Eine solche Methode stellen Chih-Chin Liu et. al in [LHC99] vor. Das Prinzip besteht darin sog. nicht-triviale Muster bzw. das Motiv eines Stücks zu extrahieren. Ein nicht-triviales Muster ist eine Notensequenz, die mehrfach vorkommt und nicht Teil eines anderen Musters ist, welches genauso häufig vorkommt. Die Auffindung der Muster wird mit Hilfe des „String-Join“ Algorithmus bewerkstelligt, welcher stets nach den Sequenzen sucht, die aus jeweils zwei kürzeren Sequenzen besteht. Auf diese Weise werden zwar nicht alle Muster entdeckt, aber auf jeden Fall die nicht-trivialen. Alle nicht-trivialen Muster werden schließlich in einem RP-Baum abgelegt, von dem sie für eine Suchanfrage herangezogen werden können. Die String-Join Methode hat alternative Muster-Extraktionsmethoden, die auf Basis von Suffixbäumen und Korrelationsmatrizen arbeiten, in der Effizienz um Längen geschlagen.

5.4.2 Semantische Klassifikation

Ein weiteres Problem, welches in der Audioanalyse anzutreffen ist, ist die automatische Bewertung und Klassifikation von Musik. Während es für Menschen ein leichtes ist, Musik nach ästhetischen Gesichtspunkten zu bewerten, und es für erfahrene Musiker sogar möglich ist, den Author eines ihnen unbekannten Stücks heraus zu hören, so lässt sich diese Aufgabe nur schwer so formulieren, dass sie algorithmisch erfasst werden könnte. Dies liegt nicht zuletzt an dem sehr subjektiven Umgang mit Musik.

Klassifikation nach Zipf-basierten Merkmalen

Bill Manaris et. al [MMM+05] ist es gelungen, genau diese Hürde zu nehmen. Sie bedienen sich dabei einer Gesetzmässigkeit, die aus der Linguistik bekannt ist. So kommt das Wort „das“ in der deutschen Sprache wesentlich häufiger vor als das Wort „Taxonomie“. Ähnlich verhält es sich in der Musik mit Konsonanz und Dissonanz. Das Erste kommt häufiger vor als das Zweite. Da aber Dissonanz in der Musik i.A. nur als Spannungsphase und Übergang zu Konsonanz gesehen wird, und der Mensch, Musik mit vorwiegend konsonantem Anteil als angenehm empfindet, lässt sich diese Beobachtung u.a. als Grundlage für die Bewertung der aesthetischen Qualität heranziehen. Das Zipf’sche Gesetz beschreibt die besagte Häufigkeitsverteilung von Ereignissen abhängig von ihrer Rangordnung. Es

schreibt eine zur Rangordnung umgekehrt proportionale Wahrscheinlichtkeit vor, mit der ein Ereignis eintreten könnte [MMM+05].

$$P(f) \sim 1/f^n$$

Angewandt auf die Musik könnte man die Häufigkeit von Tonhöhen, harmonischen Perioden und anderer statistisch erfassbarer Ereignisse untersuchen. Um z.B. festzustellen ob die melodischen Intervalle, Chopin's „Revolutionary Etude" dem Zipfschen Gesetz folgen, zählten Manaris et. al die Vorkommen von Halbton- und Ganzton-Schritten aufwärts und abwärts und Bereiche gleichbleibenden Tones. Anschließend wurden die Ereignisse entsprechend ihrer Häufigkeit absteigend sortiert und die Häufigkeiten mit den Werten des Gesetzes von Zipf verglichen. Es zeigte sich eine Korrelation zwischen den Messwerten und dem Zipf'schen Zusammenhang. Um diese Beobachtung zu erhärten wurden zwölf weitere MIDI[3]-Stücke untersucht und von unabhängigen Personen a priori bewertet. Den Probanden wurden die zwölf Stücke vorgespielt, die sie dann im Hinblick auf „Anregung" und „Wohlbefinden", bewertet haben. Anschließend wurden aus elf von zwölf Musikstücken passagenweise, 81 Zipf-relevante Merkmale extrahiert. Diese bildeten zusammen mit den zugehörigen zeitlichen Positionen im Stück und den Bewertungen der Probanden eine Trainingsmenge, die einem Neuronalen Netz zugeführt wurde. Das zwölfte Stück bildete die Testmenge. Auf diese Weise wurde das Neuronale Netz trainiert, Wohlbefinden und Anregung auf Grundlage von Zipfschen Merkmalen zu lernen. Untersuchungen des Experiments zeigten, das das Neuronale Netz dieses auch gelernt, und die meisten Stücke mit 100 prozentiger Genauigkeit bewertet hatte. Als erfolgreich wurde die Klassifikation dann angesehen, wenn sie 68% der menschlichen Bewertung entsprach, also 32% der Probanden anders als das Neuronale Netz entschieden haben. Verwandte Experimente zeigten, dass es sogar möglich ist, mit hoher Genauigkeit den Author oder das Genre eines MIDI-Stücks mit Hilfe des Zipfschen Gesetzes zu bestimmen [MVW+03].

[3]MIDI: Musical Instrument Digital Interface ist eine standardisierte, befehlsbasierte Form der Darstellung und Synthese digitaler Musik.

6 Synthese

Die Synthese beschäftigt sich mit der Erzeugung akustischer Ereignisse. Sie lässt sich in zwei Unterkategorien zerlegen, die Erzeugung von Klängen, der sog. Klangsynthese und der Komposition von ganzen Musikstücken. Während die Klangsynthese eher auf der physikalischen und der Akustischen Schicht zum Tragen kommt, wird bei der Komposition meist auf der Symbolischen Schicht gearbeitet.

6.1 Klangsynthese

Der Bereich der Klangsynthese ist relativ gut erforscht und man beherrscht heute eine Vielzahl von Methoden um natürliche und künstliche Klänge zu erzeugen. Der Bereich lässt sich in zwei Abschnitte unterteilen, in die „natürliche“ und die „künstliche“ Klangsynthese. Diese Unterscheidung ist eher der Vollständigkeit halber gemacht worden, da die natürliche Erzeugung von Klängen nicht Aufgabe der Informatik ist. Es sei an dieser Stelle auf entsprechende Literatur der Musik verwiesen.

6.1.1 Künstliche Klangsynthese

Während zu Beginn der Entwicklung in den vergangenen Jahrzehnten Musik noch ziemlich künstlich und steril wirkte, lassen sich heute sehr natürlich wirkende Klänge synthetisieren. Die moderne Popmusik lebt fast ausschließlich von der künstlichen Klangsynthese, da hier auf einfache Weise ein sehr breites Spektrum von Klängen und Effekten erzeugt werden kann, was vorher unvorstellbar war.

Ursprünglich stammen die sog. Synthesizer aus der analogen Elektronik, die es schon früh ermöglicht hat, mit Schwingungen zu arbeiten und diese vielfältig zu manipulieren. Das Prinzip bestand darin einige Grundschwingungen die von Oszillatoren geliefert wurden zu überlagern, und durch den Einsatz diverser Module, Verstärkungen, Dämpfungen oder Verzerrungen einzelner Frequenzbänder zu ermöglichen. Da die Reihenfolge der Operationen und unzählige Möglichkeiten der Modifikation der Module einen großen Einfluss auf das Endprodukt haben, entsteht ein riesiger Raum für künstlerische Entfaltungsfreiheit. Dementsprechend viel Wissen und Erfahrung sind erforderlich, um gezielt bestimmte Effekte und Töne zu erzeugen.

Mit der Entwicklung der Computer halten auch die Synthesizer Einzug in die digitale Welt. Neben den bewährten Prinzipien der Klangsynthese sind neue hinzugekommen. Einige von ihnen sind digitale Simulationen analoger Vorgänge.

Andere sind tatsächlich nur mit Hilfe von Computern realisierbar.
Zu den heute bekannten Synthesemethoden gehören:

- Subtraktive Synthese
- Additive Synthese
- Granularsynthese
- Wavetable Synthese
- FM-Synthese
- Physical Modelling
- Sampling

6.2 Musikkomposition

Wie bereits erwähnt, findet Komposition meist auf symbolischer Ebene statt. Ein Komponist entwickelt kognitiv musikalische Strukturen und fasst diese in Notenschrift ein. Dabei entstehen hierarchische Strukturen, sich wiederholende Sequenzen und Variationen eines gewissen Schemas, welches dem Stück eigen ist, das Motiv. Die Bildung von Hierarchien ist in der Musik ganz wesentlich. Die meisten Stücke lassen sich in Abschnitte fassen wie z.B. *Intro, Strophe, Refrain, Strophe, Refrain, Outro*. Die einzelnen Abschnitte sind ihrerseits aus Phrasen und ihren Verknüpfungen aufgebaut. Phrasen werden wiederum durch einzelne Noten gebildet, die entlang der Zeitachse (horizontal) oder gleichzeitig (vertikal) angeordnet werden können. Tonlängen und Pausen zwischen ihnen bilden zusammen den Rhythmus. Die Entwicklung von Harmonie, Melodie und Rhythmus ist genauso Ziel der Komposition, wie die Verwendung stilistischer Komponenten und der Vermittlung von Intentionen und Emmotionen. Während Ersteres Ziel der aktuellen Forschung ist, erscheint Letzteres derzeit noch nicht erreichbar.

Mittlerweile sind viele Ansätze im Zusammenhang mit algorithmischer Komposition untersucht worden. Zu den wesentlichen Ansätzen gehören die folgenden Systeme [McC96]:

- Stochastische Systeme
- Deterministische Systeme
- Evolutionäre Systeme
- Neuronale und Zellulare Systeme

Sie alle haben in speziellen Bereichen ihre Vorteile. Ihre ausschließliche Nutzung kann jedoch problematisch sein, da sie jeweils nicht alle Aspekte abdecken, die bei der Komposition erforderlich sind. Deshalb stößt man meistens auf kombinierte Verfahren, die die Stärken der Systeme vereinigen, um ein besseres Gesamtergebniss zu erhalten.

So eignen sich z.B. Deterministische Systeme dafür, hierarchische Strukturen zu erzeugen [McC96] oder auch Harmonien zu bilden. Auf der anderen Seite sind sie nicht im Stande, so etwas wie Kreativität nachzuahmen. Dafür eignen sich Stochastische oder Evolutionäre Systeme viel besser, da sie verstärkt mit Zufallswerten arbeiten.

Evolutionäre Kompositionssysteme sind oft interaktive Systeme [McC01], deren Fitness danach gemessen wird, wie sehr ihre Ausgaben einem menschlichen Zuhörer imponieren. Dies kann zum Nachteil werden, wenn eine automatisierte Lösung für die Kompositionsaufgabe gesucht ist. Der große Vorteil ist jedoch, das die explizite Problemlösung dem Algorithmus überlassen wird und man sich nur um die Rahmenbedingungen kümmern muss.

Zum Teil gilt dies auch für Systeme, die mit Neuronalen Netzen arbeiten. Allerdings kann man sich vorstellen, dass mit Hilfe von Analyseverfahren und der Klassifikation nach Zipf'schen Merkmalen, wie in Kapitel 5.4.2 beschrieben, eine automatische Fitnessbewertung machbar ist.

6.2.1 Stochastische Systeme

Der wesentliche Aspekt stochastischer Kompositionssysteme ist der Nichtdeterminismus. Durch die Verwendung von Zufallsprozessen ist der Ausgang der Komposition nicht vorhersagbar. Wahrscheinlichkeiten entscheiden über die Ausgabe von einzelnen Noten bzw. ganzen Noten-Sequenzen und deren Konkatenation. In diesem Zusammenhang werden z.B. Markov Modelle (8.3) eingesetzt, welche eine Erweiterung von endlichen Automaten darstellen. Ein solches System wurde für die tiefergehende Analyse im zweiten Teil dieser Arbeit ausgewählt. Das dort vorgestellte, kombinierte Verfahren vereint ein Stochastisches und ein Neuronales System. Deshalb sei an dieser Stelle auf Kapitel 8 verwiesen.

6.2.2 Deterministische Systeme

Regelmengen und Grammatiken werden in Deterministischen Systemen eingesetzt, um Harmonien oder Hierarchien zu bilden. In „Understanding Music with AI“ [Max92] schlägt J. Maxwell 55 Regeln für die Bildung von Harmonien und

Übergängen zwischen ihnen vor. Durch die Gewichtung der Regeln kann ein Bestrafungsverfahren den Kompositionsalgorithmus veranlassen Regelverstöße zu minimieren.

Grammatiken formaler Sprachen bestehen aus Ersetzungsregeln bzw. Produktionen, welche auf Sätzen aus Terminalen und Nicht-Terminalen arbeiten. Durch einfache Produktionen (Ersetzungsregeln), die die Ersetzung von Nicht-Terminalen durch andere Nicht-Terminale bzw. Terminale beschreibt, lassen sich komplexe Hierarchien erzeugen, welche als musikalische Strukturen interpretiert werden können, wie später noch gezeigt wird. In diesem Zusammenhang werden oft Fraktale erwähnt, die auf Grund ihrer geometrischen Formen in der Kunst und der Mathematik einen ästhetischen Stellenwert haben. Auch sie sind Ergebnisse mehrfacher, rekursiver Anwendung einfacher Regeln. Durch die Wiederholung des gesamten Musters in Teilbereichen desselben, mit leichten lokalen Modifikationen, sind sie sehr interessant für die Komposition. Eine Einführung in Fraktale und ihre Eigenschaften kann in [Wika; Nel93; Mau] gefunden werden.

Im Falle von J. McCormack's „Grammar Based Music Composition" [McC96] wird auf Basis einer Lindenmayer-Grammatik (L-Grammatik) ein Kompositionssystem konstruiert, welches die sinnvolle musikalische Strukturierung mit den probabilistischen Eigenschaften eines Markov Modells vereint. L-Grammatiken wurden ursprünglich für die Modellierung von natürlichen Objekten wie nichthölzernen Pflanzen, tierischen Gangarten und menschlichen Organen verwendet. Sie sind im Stande komplexe, partiell wiederholbare, hierarchische Muster zu erzeugen. Hier wird eine L-Grammatik derart ausgebaut, dass Produktionen um Anwendungswahrscheinlichkeiten erweitert werden. Dabei können Produktionen auch parallel angewandt werden, was Symmetrien in der Strukturierung begünstigt.

Beispiel: Die Grammatik **G** besteht aus dem geordneten Tripel <Σ,P,α> [McC96].

Dabei ist Σ das Alphabet, Σ^* die Menge der Worte über dem Alphabet Σ, und Σ^+ die Menge der nicht-leeren Worte über Σ.
P ist die Menge der Produktionen
$\alpha \in \Sigma^+$ ist das nicht-leere Startsymbol.
Die Menge der Produktionen ist wie folgt definiert: $P := \{$
$p_1 : C \longrightarrow E$
$p_2 : E \longrightarrow CGC$
$p_3 : G \longrightarrow \epsilon$
$\alpha : C$
$\}$

Durch Anwendung der Produktionenfolge $[\alpha, p_1, p_2, p_3, p_2, p_1, p_3]$ wird die Ausgangszeichenkette „C“ über „E“→“CGC“→“EE“→“CGCCGC“ nach „E E E E“ überführt. Durch Konkatenation der einzelnen Zeichenketten entsteht „C E C G C E E C G C C G C E E E E“ (Man achte auf die sich wiederholenden Elemente). Diese lässt sich dann als eine Folge von Noten interpretieren und abspielen.

Als Eingabe dient dem Verfahren also eine Grammatik und eine Zeichenkette als Ausgangspunkt. Der Algorithmus transformiert die Zeichenkette mehrmals unter Anwendung der Ersetzungsregeln. Die resultierende Zeichenkette wird anschließend als eine Folge von Noten interpretiert und im MIDI Format gespeichert. Ähnlich wie die Noten, werden weitere MIDI-Steuerbefehle, die u.A. die Tondauer und Klangfarbe betreffen, kodiert.

Die Ausgabe der erzeugten Musik ist prinzipiell in Echtzeit möglich, dies ist aber stark von der Komplexität der Grammatik abhängig. Produktionen die exponentiellen Wachstum der Ausgabe verursachen, können Verzögerungen in der Ausgabe erzeugen.

6.2.3 Evolutionäre Systeme

Eine weitere interessante Variante algorithmischer Komposition stellen Evolutionäre Kompositionssysteme dar. Evolutionäre Algorithmen sind ein Bereich der Künstlichen Intelligenz, der bereits einige interessante Lösungen geliefert hatte. Durch die Anwendung der evolutionären Operatoren wie Selektion, Rekombination und Mutation, lassen sich Individuen entwickeln, deren Fitness nach einer definierten Eigenschaft bewertet wird. In der Natur können Schnelligkeit, Stärke oder Intelligenz solche Eigenschaften darstellen. Durch die natürliche Selektion wird bewirkt, dass nur die Starken und Schnellen in einer rauen Umwelt überleben, während die weniger angepassten Artgenossen aussterben. Rekombinationen und Mutationen der Individuen sorgen für Fortschritt in der Entwicklung der Population.

Auf ähnliche Art und Weise lassen sich in einer künstlichen Umgebung Bedingungen schaffen, unter denen die Entwicklung einer definierten Eigenschaft begünstigt wird. Im Falle der künstlichen Komposition, können z.B. Konfigurationen von mathematischen Funktionen (als Individuen), entwickelt werden, die melodische Klänge produzieren. Da es jedoch schwierig ist die Güte melodischer Klänge algorithmisch zu bewerten, greift man oft zu interaktiven Verfahren, die den Menschen als äußere Bewertungsinstanz nutzen.

In Mc Cormack's „Eden: An Evolutionary Sonic Ecosystem" [McC01] wird eine künstliche Evolutionsplattform beschrieben, deren Individuen um das Interesse der interagierenden Besucher kämpfen. Das System ist als Exponat konzipiert, das von Besuchern als *lebendiges*, interaktives Kunswerk betrachtet werden kann.

Grundlage des Systems ist eine zweidimensionale Matrix, deren Zellen verschiedene Objekte beinhalten können. Dies können die Individuen sein, oder aber Dinge wie Biomasse (Futter) und Wände bzw. Felsen. Die Individuen können sich zwischen den Felsen frei bewegen. Sie können die Biomasse aufnehmen, um ihre Lebensenergie zu steigern und es ist ihnen möglich, sich gegenseitig zu elliminieren oder sich zu paaren. Sie sind im Stande zu sehen, zu hören und zu singen. Jedoch kostet jede Aktion Lebensenergie und wirkt sich auf die Eigenschaften des Individuums aus. So bewirkt dauernde Nahrungsaufnahme eine Massenzunahme, worauf hin das Individuum in der Bewegung langsamer wird.

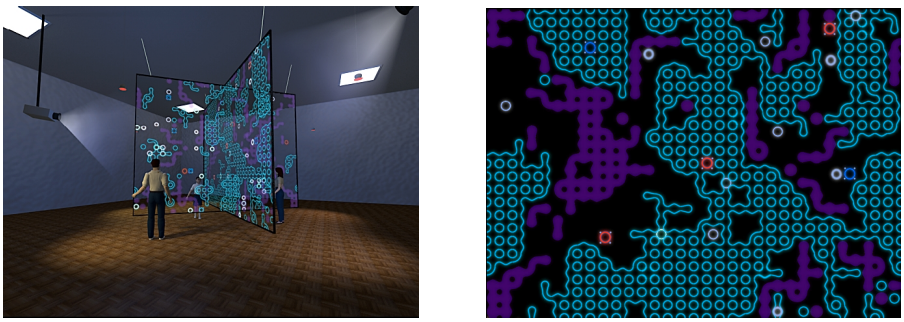

Abbildung 11: Links: Das Exponat ist in einer X-Form aufgebaut. Auf den senkrecht zu einer stehenden Ebenen ist jeweils ein Ausschnitt der Simulationsumgebung dargestellt. Besucher sind Teil des Evolutionssystems, während sie diese beobachten. Rechts: Die Visualisierung der Matrix mit ihren Bewohnern (rote und weise Ringe) und der sie umgebenden Landschaft aus Felsen (Pupur) und der Biomasse (Cyan) [McC03].

Das Verhalten der Individuen wird von einem regelbasierten System gesteuert, welches sensorische Eingabewerte auf entsprechende Aktionen überträgt. Die Sensoren liefern Werte für Lebensenergie, den Nährwert von Biomasse, Schmerz sowie visuelle und akustische Werte. Sie sind zusammen mit den möglichen Aktionen wie Bewegung, Essen, Schlagen, Paaren, Hören und Singen statisch und können sich nicht weiterentwickeln. Dagegen ist das interne Regelwerk eines Individuums der Evolution unterworfen. Diese basiert auf dem Lamarkschen Prinzip. Das bedeutet, dass bei der Rekombination diejenigen Regeln der beiden Individuen in die Folgegeneration einfließen, die am häufigsten angewendet wurden und zu einer Energiesteigerung geführt haben.

Die Interaktion wird auf zweifache Weise realisiert. Das lokale Vorkommen von Biomasse, ist abhängig von der Anzahl der Besucher, die sich eine Region des Exponats anschaut. Wenig besuchte Bereiche stagnieren, während gut besuchte Bereiche gedeihen. Diese Abhängigkeit beruht auf der Annahme, dass die Individuen durch interessantes Verhalten Besucher anlocken. Dies kann hauptsächlich durch interessante Melodien bewirkt werden. Diese würden durch ihre erhöhte Anzahl wiederum zur lokalen Steigerung der Biomasse beitragen, was sich positiv auf die Lebensenergie der Individuen auswirkt.

Weiter wirkt sich die Bewegung der Besucher auf die Mutationsrate der Individuen aus. Eine erhöhte lokale Bewegung bewirkt eine höhere Mutationsrate bei der Paarung. Diese Abhängigkeit beruht auf der Annahme, dass uninteressante Bereiche eine kurze Verweildauer der Besucher zu Folge haben und diese weitergehen. Interessante Bereiche dagegen locken Besucher an und die Mutationsrate sinkt.

Die Klänge, die bei diesem Verfahren entstanden sind, haben keinen kulturellen Ursprung und folgen keinen definierten Regeln. Sie werden von den Individuen eingesetzt, um mit einander zu kommunizieren, oder um Besucher anzulocken. Zweiteres ist maßgeblich für die lokalen Lebensbedingungen. Hörproben ausgewählter Melodien lassen sich unter [McC01] finden.

6.2.4 Neuronale und Zellulare Systeme

Neuronale Netze sind ein allgemeines Werkzeug der Informatik. Auch in der automatisierten Komposition können sie eingesetzt werden. Da sie und ihre Verwendung jedoch im zweiten Teil der Arbeit in Kapitel 8.2 beschrieben werden, wird hier auf eine eingehende Erläuterung verzichtet. Es folgt nun ein Verfahren, welches einen Zellularen Automaten und weitere Modelle in einem Kompositionssystem vereint.

Zellulare Automaten und Homeostasis

A.C. Eldridge stellt in „Adaptive Systems Music: Musical Structures from Algorithmic Process“ das AdSym Kompositionssystem vor [Eld02], welches unter Verwendung von Zellularen Automaten und einem Homeostat versucht, Musik durch Nachahmung natürlicher Selbstregulierungsprozesse, in Echtzeit zu komponieren. Das Verfahren beinhaltet drei Algorithmen, die verschiede Aspekte der Musik betreffen. Im ersten Algorithmus werden Harmonien erzeugt, im zweiten die Melodie und im dritten der Rhythmus.

Während der Zellulare Automat für den Rhythmus zuständig ist, sorgt der Homeostat für die Melodie und die harmonische Begleitung. Das Verfahren liefern schließlich eine MIDI Darstellung der erzeugten Musik.

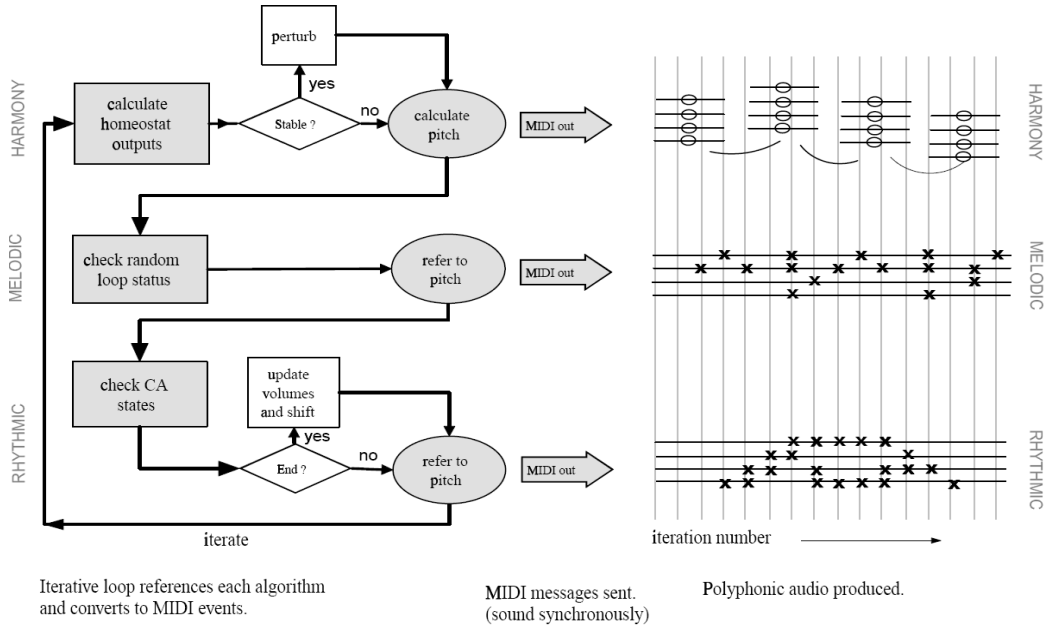

Abbildung 12: Abschnitte des AdSyM Systems zur Erzeugung von Harmonie, Melodie und Rhythmus [Eld02].

Der Zellulare Automat (CA) besteht aus einer Reihe von binären Zellen, deren Wert simultan, durch Anwendung einer Reihe von Regeln, iterativ verändert wird. Die Regeln definieren den Wert $z_i(t)$ einer Zelle z_i zum Zeitpunkt t, abhängig vom eigenen Wert $z_i(t-1)$ und den Werten benachbarter Zellen $z_{i-1}(t-1), z_{i+1}(t-1)$ in der vorhergehenden Iteration. Diese lokal angewandten Regeln verursachen eine globale Ordnung, und erzeugen auf diese Weise ein binäres Muster, welches für die Rhythmuserzeugung verwendet wird. Die Interpretation des Binären Musters geschieht durch reihenweises Auslesen der Werte. Dabei bedeutet 0=spielen, 1=nicht spielen.

Der sog. Homeostat ist ein Modell für biologische Selbstregulierungsprozesse, die vielfach bei Mensch und Tier vorkommen. Die Idee ist dabei eine Möglichkeit zu schaffen, ein System im internen Gleichgewicht zu halten (in diesem Fall sind es harmonische Verhältnisse zwischen den Zellen des Homestats) auch wenn die Umweltbedingungen sich drastisch verändern. Der Homeostat besteht aus vier Zellen, die mit einander unidirektional und rekursiv verbunden sind. Der Output einer Zelle ist die gewichtete Summe aller eingehenden Inputs. Durch diese Art

von Rückkopplung ist es dem Homeostat möglich, zwei verschiedene Zustände zu erreichen, den stabilen Zustand, indem die Werte der Zellen sich einem gewissen Zyklus anpassen, und den Zustand der Fluktuation, in dem die Werte sich unentwegt auf neue Weise verändern. Das harmonische (weil gegenseitig abhängige) Verhältnis der Zellen zu einander ist der Ausgangspunkt für die Ausgabe harmonischer Tonhöhen. Damit immer wieder komplexere Harmonien entstehen können, muss dafür gesorgt werden, dass der Homeostat nicht im stabilen Zustand verweilt.

Der für Melodien zuständige Algorithmus bedient sich der Ausgaben des CA und des Homeostats. Zunächst wählt er vier Zufallszahlen aus. Diese werden dann mit Beats des vom CA ausgegebenen Rhythmus assoziiert. Anschließend erfolgt die Ausgabe von vier Tönen, im Takt des Beats, in den vom Homeostat definierten Tonhöhen.

7 Schlussbemerkung

Während in der Analyse schon viele etablierte Forschungzweige existieren und man immer tiefer in die Materie eindringt, kämpft die Synthese noch mit grundlegenden Problemen der Aufgabendefinition. Es wurden unterschiedliche Versuche unternommen, die Problematik aus verschiedener Sicht zu beleuchten. Man bemüht Fraktale, Grammatiken und lernfähige Systeme, um Musik in der Art zu erzeugen, wie Mnschen es gewohnt sind, und in der Qualität die als angenehm empfunden werden kann. Leider gelingt dies nur mit mäßigem Erfolg. Es ist noch nicht ganz klar, welche Voraussetzungen zu erfüllen sind und welche Modelle zum gewünschten Ziel führen können.

Da nicht alle Problemklassen und Methoden in der Taxonimie abgebildet werden können, wurde eine Möglichkeit geschaffen Änderungen an ihrem Inhalt leicht durchzuführen und Erweiterungen zu realisieren. Siehe dazu Anhang 11.

Teil II
Automatisiertes Komponieren

Im ersten Teil der Arbeit wurde ein breiter Überblick über diverse Themenkomplexe gegeben. In diesem Teil richtet sich der Fokus auf ein einzelnes Verfahren zur automatisierten Komposition. Das Verfahren wird tiefergehend analysiert, mit dem Ziel einen Pseudocode für die Realisierung anzugegeben. Ausgewählt wurde ein Verfahren welches zwei Publikationen von Karsten Verbeurgt et. al umfasst [VDF04a; VDF04b]. In den nächsten Kapiteln folgt eine informelle Beschreibung der Methode, einige notwendige Grundlagen, gefolgt von formalen Ausführen und dem gewonnenen Pseudocode. Abgeschlossen wird die Analyse mit der persönlichen Bewertung des Verfahrens.

8 Aus Beispielen Komponieren lernen

Da Musik ihren Ursprung in der Kultur findet, sind immer mit ihr Erwartungen an gewohnte stilistische Eigenschaften verknüpft. Die Schwierigkeit bei der automatischen Komposition ist die erforderliche Definition und Beeinflussung des Musikstils und der musikalischen Strukturierung auf formaler Ebene. Viele der bereits unternommenen Versuche, Musik algorithmisch zu komponieren, erzeugen gemäß dem angehäuften Wissen über Musik und den gefundenen Regeln, eine Form von Musik, welche sich schnell als künstlich (also nicht direkt menschlichen Ursprungs) identifizieren lässt. Oft fehlt es ihr an Tiefe, da die hierarchische Struktur zu schwach oder zu stark ausgeprägt ist. Dadurch wirkt die Musik zu langweilig oder zu kompliziert.

Eine Möglichkeit diesen Schwierigkeiten aus dem Weg zu gehen, ist die Anwendung der Gesetzmäßigkeiten, die aus analysierten Stücken implizit gewonnen werden. Die zentrale Idee ist dabei, vorhandene Musik in ihre „Bestandteile“ zu zerlegen, und später neu zusammen zu setzen. Dadurch lassen sich charakteristische Komponenten erhalten, und durch Permutation zu neuen Gebilden zusammensetzen.

8.1 Suffix Bäume

Für die formale Beschreibung des Verfahrens müssen einige Begriffe eingeführt werden. Im Folgenden werden daher Suffix Bäume, Neuronale Netze und das Markov Modell kurz beschrieben.

Suffix Bäume sind eine Datenstruktur die die Lösung vieler Zeichenketten-basierter Probleme wie die effiziente Suche nach Wörtern in langen Texten oder das Auffinden identischer Sequenzen in verschiedenen Texten, darstellt. Sie speichern die Menge der Suffixe eines Textes derart, dass auf jedes Suffix von der Wurzel aus zugegriffen werden kann. Der Aufbau eines Suffix Baums kostet nach dem Algorithmus von Esko Ukkonen [Ukk95] Zeit $O(m)$, wenn m die Länge des Textes ist. Das Auffinden eines Wortes der Länge n ist in Zeit $O(n)$ realisierbar.

Definition: 1 *Für eine Zeichenkette* $S := [s_1, s_2, .., s_i, .., s_n]$ *der Länge* n*, sind Zeichenketten* $S_i := [s_i, .., s_n]$ *für alle* $i \in \{1..n\}$, Suffixe *von* S.

Beispiel: Das Wort S:=”CDEDCDEFG” hat die folgenden acht Suffixe:

S_1 = DEDCDEFG	S_5 = DEFG
S_2 = EDCDEFG	S_6 = EFG
S_3 = DCDEFG	S_6 = FG
S_4 = CDEFG	S_8 = G

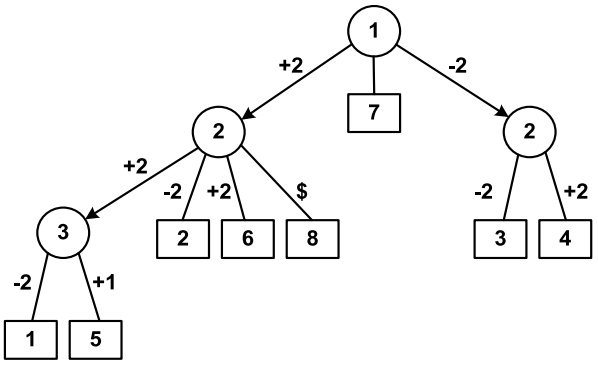

Abbildung 13: Der Suffix Baum nach dem Einfügen der Sequenz **”+2+2-2-2+2+2+1+1”** gewonnen aus **CDEDCDEFG**. Hierbei soll ”+2” oder ”-2” usw. jeweils nur ein Symbol darstellen [VDF04b].

Ein Suffix Baum erlaubt nun das effiziente abspeichern der Suffixe, sodass diese schnell wiedergefunden werden können. Dabei liegen, sich mehrfach wiederhole Muster, auf den inneren Kanten. Einfach vorkommende Muster liegen auf den Kanten der Blätter im Baum.

8.2 Neuronale Netze

Künstliche Neuronale Netze (engl: Artificial Neural Network bzw. ANN) sind eine Nachahmung kognitiver Lernprozesse eines Gehirns. Sie bestehen aus Neuronen, die mit einander in einem Netzwerk verbunden sind und Informationen verarbeiten können. Sie werden in den Bereichen der Klassifikation, Inter- und Extrapolation, Clustering und Visualisierung verwendet [KBK]. Ein Neuronales Netz ist in Schichten (Layer) organisiert, welche aus eindimensionalen Neuronenarrays bestehen. Layer lassen sich in drei Kategorien unterteilen, die Input-Layer(Eingabeschicht), die Output-Layer (Ausgabeschicht), und die dazwischen liegenden, verborgenen Schichten, die Hidden-Layer.

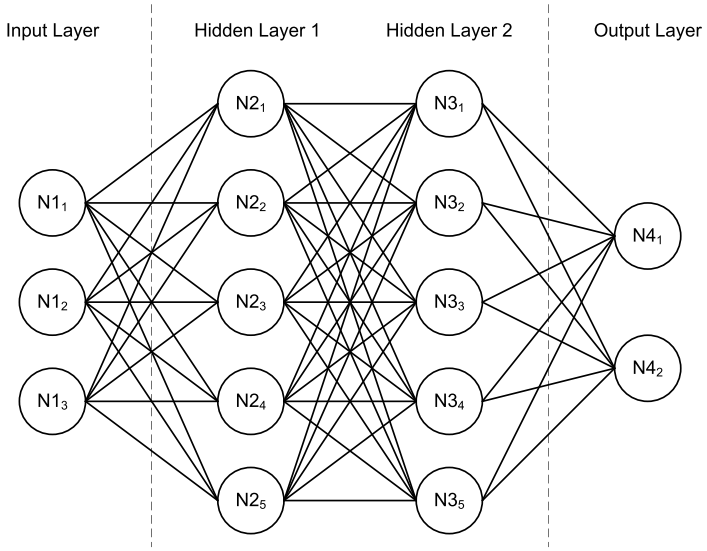

Abbildung 14: Die Datenverarbeitung finden von links nach rechts statt.

Ein Perzeptron wie in Abbildung 15 dargestellt, ist die einfachste Form eines Neuronalen Netzes. Er besteht aus einem Neuron mit m Eingabesignalen und einem Ausgangssignal. Eingangssignale sind gewichtet durch Koeffizienten $w_1..w_m$. Auf die Summe der gewichteten Eingangssignale wird die Schwellwertfunktion angewandt, um die Ausgabe des Neurons zu bilden. Diese Form des Netzes kann linear separierbare Klassifikationsprobleme lösen, wie z.B. binäre Operationen **AND** und **OR**. Dazu gehört die **XOR**-Operation nicht, da sie ein nicht linear-separierbares Problem darstellt. Durch Verwendung mehrerer Hidden Layer können auch nichtlineare Probleme wie das **XOR** gelernt werden. Das Lernen geschieht durch stete Anpassung der Signalgewichtung am Neuroneingang, abhängig vom Fehler, der bei der Ausgabe produziert wird. Der Fehler ist die Summe der

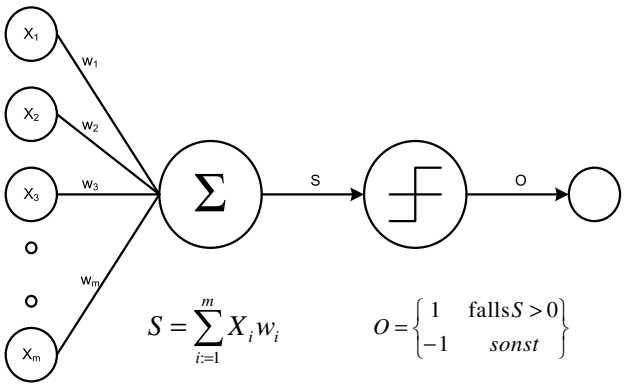

Abbildung 15: Die Ausgabe o eines Perzeptrons ist 1 wenn die Summe der gewichteten Eingaben positiv ist [KBK].

quadratischen Abweichungen des tatsächlichen Ausgabewertes vom gewünschten. Durch die Anwendung des Gradientenabstiegs lassen sich beim Perzeptron die Gewichte so justieren, das Optima in der Fehlerfunktion erreicht werden können. Bei merhschichtigen Netzen wird das Backpropagation Verfahren eingesetzt, um den Fehler vom Ausgang bis zum Eingang weiter zu vermitteln, und die Gewichte entsprechend justieren zu können. Die Schwellwertfunktion wird dann allerdings durch eine differenzierbare Sigmoid-Funktion ersetzt.

8.3 Markov Modell

Das Markov Modell ist ein von Andrei Andrejewitsch Markow definiertes, stochastisches Modell welches als Erweiterung des Endlichen Automaten angesehen werden kann [Wikd; Sie]. Es wird durch Mengen von Zuständen und Transitionen beschrieben, wobei einige Zustände als Startzustände mit einer Startwahrscheinlichkeit ausgezeichnet werden. Zusätzlich werden Übergangswahrscheinlichkeiten auf den Transitionen definiert. Das Modell erlaubt die mathematische (Re)Kostruktion stochastischer Ereignisfolgen.

Definition: 2 *Das Markov-Modell besteht aus dem Tupel* (S, P, T).

$S := \{s_1, s_2, ..., s_N\}$, *Menge der Zustände*
$P := \{p_1, p_2, ..., p_N\}$, *Menge der Startwahrscheinlichkeiten*
$T := \bigcup_{i,j \in \{1..N\}} t_{ij}$, *Menge der Übergangswahrscheinlichkeiten, mit* t_{ij}
als Wahrscheinlichkeit von Zustand S_i *nach Zustand* S_j *zu wechseln.*

Die spezielle *Markov-Eigenschaft* ist dadurch gegeben, dass Vorhersagen über die Zukunft des Modells, abhängig vom aktuellen Zustand, nicht durch zusätzliche Kenntnisse seiner Vergangenheit verbessert werden können. Die Folge $(s_1, s_2, ..., s_i)$ mit dem Zustandsraum $E = \{e_1, e_2, ..., e_i\}$ ist also eine Markov-Kette wenn

$$P(s_{i+1} = k|s_i = e_i) = P(s_{i+1} = k|s_1 = e_1, s_2 = e_2, ..., s_i = e_i)$$

erfüllt ist [Sie]. Die Summe aller Transitionswahrscheinlichkeiten eines Zustandes beträgt immer Eins, es sei denn, der Zustand hat keine ausgehenden Transitionen. In diesem Fall existiert die Summe nicht bzw. ist die Summe Null.

Durch die Erweiterung der Markov-Kette um die Möglichkeit von probabilistischen Emmisionen, ensteht das *Hidden Markov Model* (Verborgenes Markov Modell). Dieses modelliert Prozesse, welche keinen Zugriff auf interne Zustände, sondern nur die Arbeit mit ihren Ausgaben erlauben. Statistische Berechnungen an Hand emittierter Informationen können dann zur Schlussfolgerung auf interne Zustände und Zustandswechsel führen [Wikb].

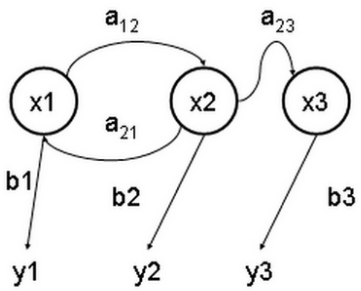

Abbildung 16: Das HMM definiert Zustände x_i, Transitionswahrscheinlichkeiten a_{ij}, Emmisionswahrscheinlichkeiten b_{ij} und mögliche Ausgabesymbole y_i [Wikb].

8.4 Algorithmus

Das Verfahren lässt sich in zwei logische Abschnitte, die Lernphase und die Kompositionsphase aufteilen. In der Lernphase wird eine Menge vorhandener Musikstücke analysiert. Dabei gewonnene Informationen werden abgespeichert. In der zweiten Phase dienen diese Informationen der Komposition eines neuen Musikstücks.

PHASE I

- **Einlesen und Analysieren der Dateien**
  1. Durchlaufe alle Musikstücke der Datenbank
  2. Erzeuge jeweils für jede Spür eines Stücks eine Zeichenfolge mit relativen Notenschritten.
  3. Füge die Zeichenfolge in den Suffix-Baum ein.
  4. Zähle wie häufig eine Sequenz auf eine andere folgt.
  5. Lasse Neuronales Netz Grundtonübergänge zwischen den Sequenzen lernen.
- **Abbilden der Informationen**
  1. Füge Sequenzen (Pfade) aus dem Suffix-Baum als Knoten ein, die eine Mindestlänge von 3 Noten haben.
  2. Berechne Übergangswahrscheinlichkeiten gewichtet mit der Länge der Sequenzen.
  3. Lasse ANN die Baseline definieren.
  4. Menge der Startzustände im Markov-Modell besteht aus Eröffnungssequenzen der gelernten Musikstücke. Benutze Gleichverteilung über Startzustände.

PHASE II

- **Komponieren**
  1. Durchlaufe die Markov Kette von einem der Startzustände aus.
  2. Spiele an jedem Knoten die zugehörige Sequenz ab.
  3. Breche nach einer festen Anzahl von Transitionen ab.

Verbeurgt et. al verwenden einen Suffix-Baum, um Musikstücke auf Muster (sich mehrfach wiederholende Sequenzen) hin zu untersuchen, und diese abzuspeichern. Dabei werden wie in Abbildung 17 dargestellt, nicht die absoluten Tonhöhen, sondern nur Differenzen zwischen Notenschritten im Suffix-Baum abgelegt. Absolute Tonhöhen der jeweils ersten Noten eines Musters bilden Grundtöne, welche bei einer erneuten Synthese als Ursprung für relative Tonänderungen im Verlauf der ausgegebenen Sequenz interpretiert werden.

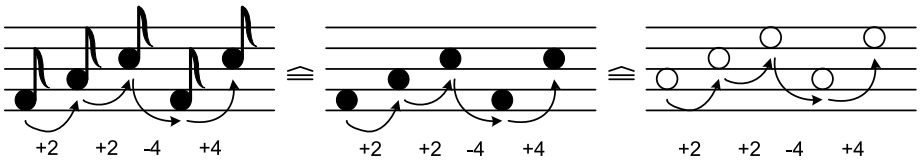

Abbildung 17: Sequenzen die sich nur durch Tonhöhe oder Tondauer unterscheiden, aber dieselben relativen Notensprünge aufweisen, werden als Ausprägungen desselben Musters erkannt.

Ein Neuronales Netz lernt Übergänge zwischen Grundtönen der gespeicherten Muster, um sie später in Abhängigkeit vom Fortschritt des Kompositionsprozesses auszugeben.

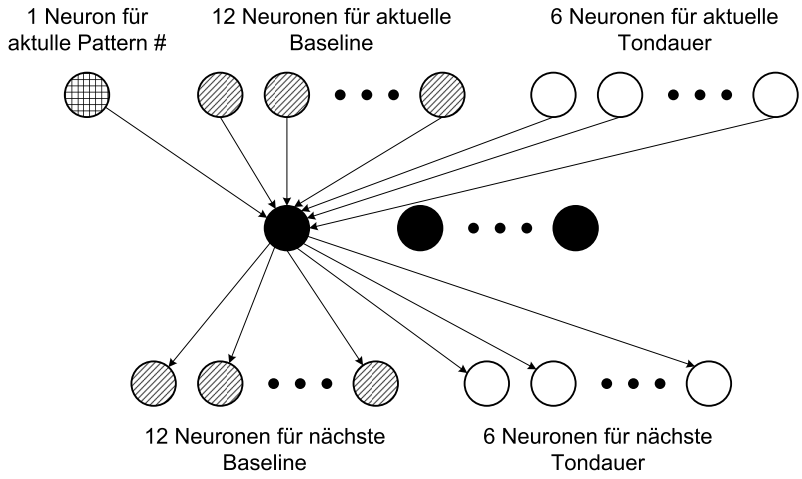

Abbildung 18: Die Eingabeschicht besteht aus 19 Neuronen. Die Neuronenanzahl in der verborgenen Schicht ist aus den Publikationen nicht ersichtlich. 18 Neuronen bilden die Ausgabeschicht.[VDF04b]

Die Tonhöhe ist unär kodiert, das heißt für die 12 möglichen Noten der chromatischen Skala gibt es 12 Neuronen, von denen jeweils einer den Wert 1 erhält und die anderen den Wert 0. Analog wird die Tondauer kodiert. Die aktuelle Pattern Nummer, repräsentiert den Zustand des Markov Modells. Die Berechnung der Tonhöhe und Tondauer erfolgt in Abhängigkeit vom Pattern, von dem aus gewechselt werden soll. Sie ist dezimal kodiert.

Anschließend wird das Markov Modell erzeugt und die gespeicherten Muster aus dem Suffix Baum als Knoten in die Markov Kette eingefügt. Die eigentliche Komposition findet während der Bewegung durch die Markov Kette statt. An jedem besuchten Knoten wird eine Sequenz von Tönen ausgegeben, deren Grundton vom Neuronalen Netz vorgegeben wird. Transitionen zwischen den Knoten werden abhängig von der, ihnen zugewiesenen Wahrscheinlichkeit, benutzt.

8.5 Pseudocode

Das Ziel der Analyse war die Rekostruktion eines Pseudocodes für das Verfahren in [VDF04a; VDF04b]. Dieses gelang bis auf einzelne, nicht dokumentierte Details. Der aus sechs Funktionen bestehende Pseudocode wird mit eingestreuten Kommentaren in blauer Schrift und zusätzlichen Erklärungen auf den folgenden Seiten präsentiert.
Die Hauptroutine **LearnToComposeFromExamples(**S**)** [Algorithmus 1] umfasst Funktionsaufrufe, welche vier Schritte des Algorithmus darstellen. Sie erwartet das Argument S. Dies ist eine Menge von Musikstücken, z.B. im MIDI Format.
Als Erstes wird **FindPatterns(**S**)** [Algorithmus 2] aufgerufen um die Zeichenfolge der Notenschritte zu ermitteln. Alle Musikstücke M der Menge S werden dabei durchlaufen und jede Spur des Stücks wird analysiert. Es entsteht eine Folge von Halbnotenschritten wie etwa " $+1+2-2+4+3-2-2...$".

Diese Zeichenfolge dient dem **SuffixTreeInsert** [Algorithmus 3] als Eingabe. Sie wird dort in den Suffix Baum eingefügt. Dies kann mit Hilfe des Algorithmus von Ukkonen [Ukk95] geschehen. Dadurch werden mehrfach vorkommende Sequenzen gefunden und auf internen Kanten des Baums platziert.

Es muss dafür gesorgt werden dass, jedes Suffix im Baum einen eindeutigen Verweis auf das zu Grunde liegende Musikstück und die Tracknummer enthält. Dies ist für das Lernen der Grundtöne durch **LearnBaselines(**N**)** [Algorithmus 4] notwendig.

Algorithm 1 Hauptroutine

```
1: function LearnToComposeFromExamples(S)
   ▷ Globale Variablen
2:     Global st ← SuffixTree
3:     Global nn ← NeuralNetwork
4:     Global mm ← MarkovModell

   ▷ Berechnen von Notenschritten und Einfügen in den Suffix Baum
5:     FindPatterns()

   ▷ Lasse ein ANN baselines der im SuffixBaum gespeicherten Muster lernen
6:     LearnBaselines(20)

   ▷ Übertrage Muster aus dem Suffix Baum ins Markov Modell
7:     BuildMarkovModel(st)

   ▷ Komponiere neues Stück aus 100 Mustern
8:     newmidi ← Compose(100)
9: end function
```

Algorithm 2 Auslesen der Musikstücke

```
1: function FindPatterns
2:     for all M ∈ S do                          ▷ Wiederhole für alle MIDI Dateien
3:         for all T ∈ M do                      ▷ Wiederhole für jeden Track
4:             steps ← " "                       ▷ Initialisiere Zeichenkette
5:             n_i ←FirstNote(T)                 ▷ Lese erste Note
6:             n_j ←NextNote(T)                  ▷ Lese nächste Note
7:             while n_j ≠ null do               ▷ Solange es weitere Noten gibt
8:                 Δn ←Pitch(n_j)-Pitch(n_i)     ▷ Tonhöhenunterschied Δn
9:                 steps ←steps + Δn             ▷ Füge Δn an
10:                n_i ← n_j
11:                n_j ←NextNote(T)              ▷ Lese nächste Note
12:            end while
   ▷ Füge steps in den Suffix Baum ein.
   ▷ Markiere dabei neue Blätter mit Namen und Track der MIDI
13:            SuffixTreeInsert(Name(M),Number(T),steps)
14:        end for
15:    end for
16: end function
```

Algorithm 3 Erzeugen des Suffix Baums aus den Notenschritten

1: **function** SUFFIXTREEINSERT($mname, trackNr, steps$)
Require: $mname \neq null$, $trackNr \neq null$, $steps \neq null$

2: **if** $st = null$ **then**
3: $st \leftarrow$ **new** SuffixTree
4: **end if**

5: UkkonenInsert($st, mname, trackNr, steps$) ▷ z.B. Algorithmus von E.Ukkonen [Ukk95].
▷ UkkonenInsert muss derart angepasst werden, das neu entstandene Blätter neben dem Verweis auf die Startposition des Suffix in der Zeichenkette $steps$, auf das zugehörige Musikstück $mname$ und die Tracknummer $trackNr$ verweisen. Zusätzlich müssen Knoten eindeutig durchnummeriert werden. Später wird auf die Nummer per $Nr(Knoten)$ zugegriffen.
6: **end function**

Anschließend wird das Neuronale Netz aus Abbildung 8.4 in **LearnBaselines**(N) [Algorithmus 4] trainiert, Grundtonübergänge zwischen den Mustern zu lernen. Dabei wird das Backpropagation Verfahren [KBK] angewendet

Leider lässt sich die Architektur des Neuronalen Netzes nicht vollständig rekonstruieren. So ist z.B. die Größe der verbogenen Schicht nicht angegeben. Im Falle einer Implementierung ließe sich diese aber experimentell bestimmen.

Im nächsten Schritt werden durch **BuildMarkovModel()** [Algorithmus 5] die im Suffix Baum gespeicherten Muster, ins Markov Modell übertragen. Sequenzen die kürzer als 3 Noten sind, werden nicht zugelassen, um häufige, krasse Sprünge im Melodieverlauf zu vermeiden. Je öfter eine Sequenz auf eine andere folgt, desto höher ist die Transitionswahrscheinlichkeit zwischen entsprechenden Zuständen im Markov Modell. Zu Startzuständen werden alle Zustände die mit Eröffnungssequenzen der Stücke aus der Trainingsmenge beginnen.

Algorithm 4 Lernen der Muster-baselines durch das ANN

1: **function** LEARNBASELINES(N)
Require: $S \neq null, st \neq null$

▷ Falls nn nicht existiert → neu anlegen
2: **if** $nn = null$ **then**
3: $L_{\text{in}} \leftarrow$**new** ANNLayer(19 Neurons) ▷ Input layer
4: $L_{\text{h}} \leftarrow$**new** ANNLayer(h Neurons) ▷ Hidden layer
5: $L_{\text{out}} \leftarrow$**new** ANNLayer(18 Neurons) ▷ Output layer
6: $nn \leftarrow$ **new** ANN(L_{in}, L_{h}, L_{out}); ▷ Verschaltung der Layer im Netz
7: **end if**

▷ Trainingsmenge der Größe N
8: **for** $n \leftarrow 1$ **to** N **do**
9: $p_i \leftarrow$SuffixTreeRandom(st) ▷ Wähle zufällig einen Knoten aus
10: $p_j \leftarrow$SuffixTreeNext(st, p_i) ▷ Wähle nachfolgenden Knoten aus

▷ Finde Quellen aus denen Muster p_i und p_j stammen und lese jeweils deren erste Note
11: $n_i \leftarrow$ LookupMIDI(p_i.mname,p_i.trackNr,p_i.offset)
12: $n_j \leftarrow$ LookupMIDI(p_j.mname,p_j.trackNr,p_j.offset)

▷ Konstruiere Eingabe für Input Layer
13: $L_{\text{in}}[1] \leftarrow$Nr($p_i$) ▷ Neuron 1: Nr. des Musters
14: $L_{\text{in}}[2..13] \leftarrow$Pitch($n_i$) ▷ Neuron 2..13: Tonhöhe
15: $L_{\text{in}}[14..19] \leftarrow$Duration($n_i$) ▷ Neuron 2..13: Tondauer

▷ Gewünschter Output für nächstes Muster
16: $L_{\text{out,must}} \leftarrow$**new int** $[18]$
17: $L_{\text{out,must}}[1..12] \leftarrow$Pitch($n_j$) ▷ Neuron 1..12: Tonhöhe
18: $L_{\text{out,must}}[13..18] \leftarrow$Duration($n_j$) ▷ Neuron 13..18: Tondauer

19: TrainANN(nn,L_{in},$L_{\text{out,must}}$) ▷ Lasse nn lernen
20: **end for**
21: **end function**

Algorithm 5 Abbilden der Suffix Baum Knoten auf Knoten im Markov Modell

1: **function** BUILDMARKOVMODEL
Require: $st \neq null$
2: $S \leftarrow \emptyset$ ▷ Menge der Zustände
3: $A \leftarrow \emptyset$ ▷ Menge der Startzustände
4: $T \leftarrow \emptyset$ ▷ Menge der Transitionen
5: $mc \leftarrow$ **new** MarkovModel(S, A, T) ▷ Erzeuge das Markov Modell
6: $s_i \leftarrow$**new** State(null) ▷ Leerer Startzustand
7: $S \leftarrow S \cup s_i$
8: $A \leftarrow A \cup s_i$

▷ Wiederhole für alle inneren Knoten, deren Muster länger als 2 ist
9: **for all** $p_i \in st$: Parent(p_i)$\neq null \wedge$Children(p_i)$\neq null \wedge$Length(p_i)>2 **do**
10: $s_i \leftarrow$**new** State(p_i) ▷ Zustand s_i erhält Inhalt von Muster p_i
11: **if** $s_i \notin S$ **then**
12: $S \leftarrow S \cup s_i$
▷ Wenn das Muster eine Eröffnungssequenz darstellt, wird der Zustand zum Startzustand
13: **if** p_i.offset=1 $\wedge$ $s_i \notin A$ **then**
14: $A \leftarrow A \cup s_i$
15: **end if**
16: **end if**

▷ ... und für alle Unterknoten
17: **for all** $p_j \in st : Parent(p_j) = p_i \wedge$Length($p_j$)>2 **do**
18: $s_j \leftarrow$**new** State(p_j) ▷ Zustand s_j erhält Inhalt von Muster p_j
19: **if** $s_j \notin S$ **then** ▷ Füge Knoten ins Modell ein
20: $S \leftarrow S \cup s_j$
21: **end if**
▷ Füge neue Transitionen ins Modell ein
22: $t_{ij} \leftarrow$**new** Transition(s_i, s_j)
23: **if** $t_{ij} \notin T$ **then**
24: $T \leftarrow T \cup t_{ij}$
25: Counter(t_{ij}) $\leftarrow$0 ▷ Initialisiere Transitions-Zähler für t_{ij}
26: **else**
▷ zähle wie oft Übergang benutzt wird
27: Counter(t_{ij}) $\leftarrow$Counter(t_{ij}) + 1
28: **end if**
29: **end for**
30: **end for**

▷ Fortsetzung auf Seite 44

Algorithm 6 Abbilden der Suffix Baum Knoten auf Knoten im Markov Modell

▷ Fortsetzung von Algorithmus 5 auf Seite 43

▷ Wiederhole für alle Zustände im Markov Modell
31: **for all** $s \in S$ **do**
32: $sum \leftarrow 0$ ▷ Summe der ausgehenden Transitionen

▷ Zähle ausgehende Transitionen
33: **for all** $t_{ij} \in T : s_i, s_j \in S \wedge s_i = s$ **do**
34: $sum \leftarrow sum+$ Counter(tij)
35: **end for**

▷ Bilde Übergangswahrscheilichkeiten für Transitionen
36: **for all** $t_{ij} \in T : s_i, s_j \in S \wedge s_i = s$ **do**
37: $P(t_{ij}) \leftarrow$ Counter(t_{ij})$/sum$
38: **end for**
39: **end for**
40: **end function**

Sobald das Markov Modell vollständig ist und das Neuronale Netz trainiert ist, kann ein neues Stück mittels **Compose()** [Algorithmus 7] komponiert werden. Dies geschieht durch Bewegung im Markov Modell.

Beginnend in einem der Startzustände, deren Startwahrscheinlichkeiten gleichverteilt sind, werden gemäß den errechneten Transitionswahrscheinlichkeiten Übergänge zu benachbarten Zuständen durchgeführt. In jedem Zustand wird das zugehörige Muster abgespielt. Dabei liefert das Neuronale Netz, abhängig von der aktuellen Grundtonhöhe, der Grundtondauer und dem aktuellen Zustand, in dem sich das Markov Modell befindet, die nachfolgende Grundtönhöhe und die Grundtondauer. Darauf aufbauend, bildet das Markov Modell absolute Tonhöhen, durch Addition der Grundtonhöhe mit den Halbtonschritten der Sequenz. Die Grundtondauer gilt für alle Noten der Sequenz.

Die auf diese Weise konstruierte, ”neue” Notenfolge wird in einer MIDI Datei mit nur einer Spur abgelegt. Sie ist zum Abspielen bereit.

Algorithm 7 Komposition eines neuen Stücks

1: **function** COMPOSE($maxsteps$)

Require: $mc \neq null, nn \neq null, maxsteps > 0$

2: $A \leftarrow$InitialStates(mc) ▷ Menge der Startzustände
3: $s_i \leftarrow$MMRandom(A) ▷ Wähle beliebigen Startzustand

▷ Schaue welche Note zum Zustand gehört
4: $pnr \leftarrow$ Nr(s_i)
5: $p \leftarrow$ SuffixTreeFind(Nr(s_i))
6: $n \leftarrow$LookupMIDI(p_i.mname,p_i.trackNr,p_i.offset)
7: $base_pitch \leftarrow$Pitch(n)) ▷ Lese Tonhöhe
8: $base_duration \leftarrow$Duration(n))) ▷ Lese Tondauer

9: $T \leftarrow$**new** MIDITrack ▷ Lege neuen Track an
10: $T \leftarrow T + n$ ▷ Füge die Initialnote hinzu

11: $steps \leftarrow 0$ ▷ Schrittzähler
12: **while** $steps \leq maxsteps$ **do** ▷ Wiederhole $maxsteps$ mal

▷ Bilde neue Noten mit angepasster Tonhöhe und Tandauer
13: **for** $k \leftarrow 1$ **to** Length(s_i) **do**
14: $n \leftarrow$ **new** Note($base_pitch + s_i[k], base_duration$)
15: $T \leftarrow T + n$ ▷ Hänge erzeugte Noten an Track an
16: **end for**

▷ Erzeuge Eingabe für das NN bestehend aus aktuellem Zustand des Markov Modells, der aktuellen Tonhöhe und Tondauer
17: $nn_{\text{in}} \leftarrow$**new int** [19]
18: $nn_{\text{in}}[1] \leftarrow pnr$
19: $nn_{\text{in}}[2..13] \leftarrow base_pitch$
20: $nn_{\text{in}}[14..19] \leftarrow base_duration$

▷ Lasse Tonhöhe und Tondauer für den nächsten Zustand bestimmen
21: $nn_{\text{out}} \leftarrow$**new int** [18]
22: $nn_{\text{out}} \leftarrow$QueryANN($nn$,$nn_{\text{in}}$)
23: $s_j \leftarrow$NextState(mc, s_i) ▷ Der nächste Zustand
24: $pnr \leftarrow$ Nr(s_j)

▷ Fortsetzung auf Seite 46

Algorithm 8 Komposition eines neuen Stücks

▷ Fortsetzung von Algorithmus 7 auf Seite 45

▷ Speichere Ausgabewerte für nächste Iteration

$p_i \leftarrow$PatternNr(s_j)

$bl_pitch \leftarrow nn_{\text{out}}[1..12]$

$bl_duration \leftarrow nn_{\text{out}}[13..18]$

$s_i \leftarrow s_j$

$steps \leftarrow steps + 1$ ▷ Nächster Schritt

end while

▷ Der Track ist generiert → Lege ihn in der MIDI Datei ab

$newMIDI \leftarrow$ **new** MIDIFile(T);

return $newMIDI$ ▷ Gebe MIDI Datei aus

end function

9 Schlussbemerkung

Der Pseudocode wurde auf der Grundlage der beiden Publikationen [VDF04a; VDF04b] rekonstruiert. Stellenweise mussten Implementierungdetails, auf Grund fehlender Informationen, frei gewählt werden. Auch ist der Pseudocode so weit abstrakt gehalten, dass sehr spezielle Aspekte, wie z.B. der Umgang mit dem MIDI-Format, die konkrete Implementierung der Suffix Bäume, der Neuronalen Netze und des Markov Modells nicht behandelt wurden. Diese Details können in einschlägiger Literatur und unter [KBK; Sie] gefunden werden. Für die Umsetzung werden im Anhang B frei verfügbare Implementierungen der oben genannten Komponenten angegeben, welche eine Implementierung des Pseudocodes erleichtern können.
Das vorgestellte Kompositionsverfahren hat gegenüber anderen Ansätzen den Vorteil, dass bezüglich der komponierten Musik keine Richtlinien aufgestellt werden müssen, da sich diese aus den gelernten Musikstücken ergeben. Dies erlaubt eine Stil-ungebundene Komposition und somit ein breites Spektrum an möglichen Ausgaben. Außerdem können die komponierten Stücke Hierarchien mit wiederholbaren Strukturen aufweisen, wenn diese auch nicht unbedingt dem Beispiel in Kapitel 6.2 entsprechen.

Da das Verfahren lediglich monophone Ausgaben erzeugt, wie man Abbildung 19 sehen kann, kann die komponierte Musik die gelernten Stücke nur bedingt
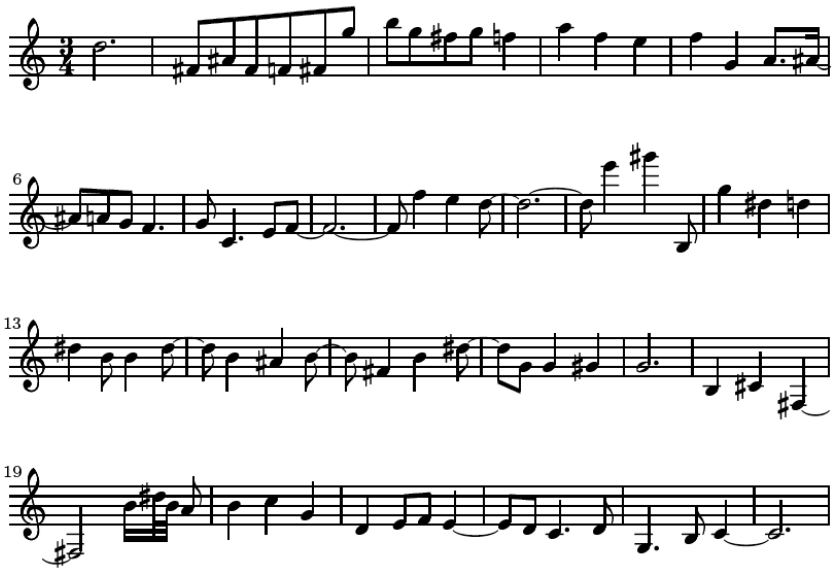
Abbildung 19: Ein mit Hilfe des vorgestellten Systems, algorithmisch komponiertes Musikstück. [VDF04b]

sinnvoll verwenden.Dieses schränkt auch den Rahmen möglicher Kompositionen stark ein, da Harmonien ein starkes Ausdrucksmittel sind und der Musik Tiefe verleihen.

Ein Problem das andere Verfahren mit dem vorgestellten teilen, ist auch, dass hinter der Komposition keine Intention steckt. Da hinter menschlichen Kompositionen meistens eine Absicht steckt, die die Musik als Medium nutzt um sich mitzuteilen, und der Zuhörer gleichzeitig eine Erwartung an diese stellt, kann davon ausgegangen werden, dass Verfahren die diesen Aspekt nicht berücksichtigen, nur bedingt erfolgreich sein können.

Teil III
Schluss

Dieses letzte, kleine Kapitel fasst kurz das zuvor Gesagte zusammen und schließt die Arbeit mit einem Resümee ab.

10 Zusammenfassung

Im ersten Teil der Arbeit wurde ein grober Überblick über existierende Methoden und Algorithmen im Zusammenhang mit der automatisierten Analyse und Synthese in der Musik gegeben. Dabei entstand eine Taxonomie der Aufgabenstellungen und einiger, verwendeter Lösungen. Es wurden wesentliche Aspekte aufgezeigt, auf die die Informatik bei der Arbeit mit Analyse und Synthese stößt. Zentrale Ideen hinter den Methoden wurden kurz erläutert.

Im zweiten Teil wurde ein spezielles, automatisiertes Kompositionsverfahren näher betrachtet. Aus den verfügbaren Quellen wurde ein Pseudocode gewonnen, der einen Einblick in die Gestalltung eines solchen Systems erlaubt. Die verwendeten Komponenten wurden angesprochen und ihre Verwendung im Pseudocode aufgezeigt.

11 Ausblick

Da die Audioanalyse von Musik auf der Informationsgewinnung beruht, und viele, bereits etablierte Methoden der Mathematik und Physik nutzt, hat sie sich im Gegensatz zur Synthese, zum stärker ausgeprägten Forschungszweig entwickelt. Viele interessante Methoden der automatisierten Audioanalyse existieren bereits und weitere werden folgen. Die Transkription ist z.B. eines der Gebiete, welches vielversprechende Perspektiven bietet. Mit einem Transkritionssystem wäre z.B. die Resynthese korrupter, verrauschter Audiodaten möglich, oder man könnte es als effiziente Kompressionsmethode verwenden, da die Daten in ihrer symbolischen Darstellung weniger Informationen tragen müssen.

Was die Synthese betrifft, sieht es derzeit nicht so aus, dass die Informatik in der Nähe einer greifbaren Lösung für die Kompositionsaufgabe stünde. Zu viele Fragen sind auf Grund der Komplexität des Sachverhaltes noch unbeantwortet. Es existiert kein klares Konzept, welches definitiv eine menschenähnliche Kompositionskunst an den Tag legen könnte. Die derzeit exisitierenden Systeme wei-

chen in ihrer Konzeption stark von einander ab und verwenden dabei viele unterschiedliche Algorithmen. Intuitiv gesehen, liegt es auf der Hand, was notwendig ist um komponieren zu können. Kreativität, kulturelles Hintergrundwissen, Musikerfahrung, Intention, die Kenntniss über physikalische Aspekte der Musik und die Fähigkeit all diese Dinge mit einander zu kombinieren. Jedoch liegt die große Schwierigkeit darin, die intuitiven Begriffe in formale Modelle und Aussagen zu gießen. Erst wenn diese Hürde überwunden wurde, besteht die Möglichkeit Musik automatisiert zu komponieren.

Literatur

[AD99] ARFIB, DANIEL und NATHALIE DELPRAT: *Alteration of the Vibrato of a Recorded Voice.* in Proc. of International Computer Music Conference, Seiten 186–189, 1999.

[Cha01] CHAI, WEI: *Melody Retrieval On The Web.*, 2001.

[Col05] COLLINS, NICK: *A Comparison of Sound Onset Detection Algorithms with Emphasis on Psychoacoustically Motivated Detection Functions*. Audio Engineering Society, 2005.

[Eld02] ELDRIDGE, A.C.: *Adaptive systems music: Musical Structures from Algorithmic Process*. In: SODDU, C. (Herausgeber): *Proceedings of Generative Art 2002*, Milan, 2002.

[GM99] GOTO, MASATAKA und YOICHI MURAOKA: *Real-time beat tracking for drumless audio signals: Chord change detection for musical decisions*. Speech Communication, 27:311–335, 1999.

[HO00] HYVÄRINEN, AAPO und ERKKI OJA: *Independent Component Analysis: Algorithms and Applications.* Neural Networks, Seiten 411–430, 2000.

[HS99] HERRERA, PERFECTO und XAVIER SERRA: *Audio Descriptors and Descriptor Schemes in the Context of MPEG-7*. ICMC99, 0:1–4, 1999.

[JA99] JENSEN, KRISTOFFER und J. ARNSPANG: *Binary decision tree classification of musical sounds*. `citeseer.ist.psu.edu/jensen99binary.html` (gelesen am 20.01.2006), 1999.

[JA03] JENSEN, KRISTOFFER und TUE HASTE ANDERSEN: *Real-time beat estimation using feature extraction.* Lecture Notes in Computer Science. Springer Verlag, 2003.

[Jeh97] JEHAN, TRISTAN: *Musical Signal Parameter Estimation.*, 1997.

[KBK] KLEINE-BÜNING, PROF. H. und DIPL.-INFORM. O. KRAMER: *Vorlesungsfolien Maschinelles Lernen: Neuronale Netze, Reinforcement Learning, Evolutionäre Algorithmen*.

[LHC99] LIU, CHIH-CHIN, JIA-LIEN HSU und ARBEE L. P. CHEN: *Efficient Theme and Non-Trivial Repeating Pattern Discovering in Music Databases*. International Conference on Data Engineering, 15:14–21, 1999.

[Log00] LOGAN, B.: *Mel Frequency Cepstral Coefficients for music modeling*. `http://overcite.lcs.mit.edu/logan00mel.html` (gelesen am 26.01.2006), 2000.

[Mar01] MARCHAND, SYLVAIN: *An Efficient Pitch-Tracking Algorithm Using a Combination of Fourier Transforms*. Digital Audio Effects, Seiten 6–8, 2001.

[Mau] MAURER, JOHN A.: *The Influence of Chaos on Computer-Generated Music*. `http://ccrma.stanford.edu/~blackrse/chaos.html` (gelesen am 20.01.2006).

[Max92] MAXWELL, H.J.: *An Expert System for Harmonic Analysis of Tonal Music*. Understanding Music with AI: Perspectives on Music Cognition, Seiten 336–345, 1992.

[MB96] MASRI, PAUL und ANDREW BATEMAN: *Improved modelling of attack transients in music analysis-resynthesis.* in Proc. of International Computer Music Conference, 1996.

[McC96] MCCORMACK, JON: *Grammar Based Music Composition*. Complexity International, 3, 1996.

[McC01] MCCORMACK, JON: *Eden: An Evolutionary Sonic Ecosystem*. Lecture Notes in Artificial Intelligence. Springer Verlag, 2159:133–142, 2001.

[McC03] MCCORMACK, JON: *Evolving Sonic Ecosystems*. Kybernetes, 32(1/2):184–202, 2003.

[MMM+05] MANARIS, BILL, PENOUSAL MACHADO, CLAYTON MCCAULEY, JUAN ROMERO und DWIGHT KREHBIEL: *Developing Fitness Functions for Pleasant Music: Zipf's Law and Interactive Evolution Systems*. Lecture Notes in Computer Science. Springer Verlag, 3449:498–507, 2005.

[MVW+03] MANARIS, BILL, DALLAS VAUGHAN, CHRISTOPHER WAGNER, JUAN ROMERO und ROBERT B. DAVIS: *Evolutionary Music and*

the Zipf-Mandelbrot Law: Developing Fitness Functions for Pleasant Music. Lecture Notes in Computer Science. Springer Verlag, 2611, 2003.

[MXSW04] MADDAGE, NAMUNU CHINTHAKA, CHANGSHENG XU, ARUN SHENOY und YE WANG: *Semantic Region Detection in Acoustic Music Signals*. In: *PCM (2)*, Band 3332, Seiten 874–881, 2004.

[Nel93] NELSON, GARY LEE: *Real Time Transformation of Musical Material with Fractal Algorithms*. `http://www.timara.oberlin.edu/~gnelson/gnelson.htm` (gelesen am 20.01.2006), 1993.

[PZ04] PACHET, FRANCOIS und AYMERIC ZILS: *Evolving Automatically High-Level Music Descriptors from Acoustic Signals*. Springer-Verlag. Lecture Notes in Computer Science, 2771:42–53, 2004.

[SA03] S. ALEXANDER, E. DANIEL: *Chord segmentation and recognition using EM-trained hidden markov models.*, 2003.

[Sie] SIEGEL, CHRISTIAN: *Facharbeit Mathematik: Markow-Ketten*. `http://www.siegel-christian.de/seiten/facharbeit/markow.html` (gelesen am 20.01.2006).

[TEC01] TZANETAKIS, G., G. ESSL und P. COOK: *Automatic Musical Genre Classification Of Audio Signals*, 2001.

[Ukk95] UKKONEN, ESKO: *On-line construction of suffix trees*. Algorithmica, 14:249–260, 1995.

[VDF04a] VERBEURGT, KARSTEN, MICHAEL DINOLFO und MIKHAIL FAYER: *Extracting Patterns in Music for Composition via Markov Chains*. Lecture Notes in Artificial Intelligence. Springer Verlag, 3029:1123–1132, 2004.

[VDF04b] VERBEURGT, KARSTEN, MICHAEL DINOLFO und MIKHAIL FAYER: *A Hybrid Neural-Markov Approach for Learning to Compose Music by Example*. Canadian AI, 3060:480–484, 2004.

[VK00] VIRTANEN, TUOMAS und ANSSI KLAPURI: *Separation of harmonic sound sources using Sinusoidal Modeling.* ICASSP, 2000.

[Wika] WIKIPEDIA: *Fraktale*. `http://de.wikipedia.org/wiki/Fraktal` (gelesen am 20.01.2006).

[Wikb] WIKIPEDIA: *Hidden Markov Model*. `http://de.wikipedia.org/wiki/Hidden_Markov_Model` (gelesen am 20.01.2006).

[Wikc] WIKIPEDIA: *Korrelation*. `http://de.wikipedia.org/wiki/Korrelation` (gelesen am 20.01.2006).

[Wikd] WIKIPEDIA: *Markow-Kette*. `http://de.wikipedia.org/wiki/Markov-Kette` (gelesen am 20.01.2006).

Anhang

A Darstellung der Taxonomie

Die Taxonomie liegt im XML Format vor. Sie kann mittels eines XSLT Transformers und einem XSL StyleSheet in eine graphische Darstellung im SVG Format (Scalable Vector Graphic) überführt werden. Das Resultat lässt mit dem Adobe SVG Viewer[4] i.d.R. ein einem Browser[5] betrachten. Da SVG v1.0 keine Unterstützung für automatischen Zeilenumbruch mitbringt, dieses aber in der Darstellung langer Texte der Taxonomie von Vorteil ist, wurde ein externes Javascript-Modul eingebunden, welches nur zur Laufzeit im Browser zum Einsatz kommt.

B Implementierungen

Für die Implementierung des Pseudocodes in Kapitel 8.5 lassen sich einige open-source Bibliotheken für C++ verwenden.

- **WildMIDI MIDI processing library**
  http://sourceforge.net/projects/wildmidi
- **MUMmer Project**
  http://sourceforge.net/projects/mummer
- **libstree — A Generic Suffix Tree Library**
  http://www.cl.cam.ac.uk/ cpk25/libstree
- **General Hidden Markov Model Library**
  http://sourceforge.net/projects/ghmm
- **Fast Artificial Neural Network Library**
  http://sourceforge.net/projects/fann

[4] zu finden unter: http://www.adobe.com/svg/viewer/install/main.html
[5] erfolgreich getestet werden konnte nur der Microsoft Internet Explorer 6.0