FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Pattern-Based MIDI Composing

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik

eingereicht von

Jakob Fellner, BSc
Matrikelnummer 0525287

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer: Ao. Univ.-Prof. Mag. Dr. Horst Eidenberger

Wien, 24.04.2013

(Unterschrift Verfasser) (Unterschrift Betreuer)

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 24.04.2013 _____

(Unterschrift Verfasser)

Danksagung

Ich danke meinen Eltern für die finanzielle und emotionale Unterstützung während der gesamten Studienzeit. Ohne eure Hilfe wäre mir diese Ausbildung verwehrt geblieben.

Außerdem danke ich meiner Lebensgefährtin Manuela für ihre tatkräftige Unterstützung.

Ganz besonderer Dank gebührt weiters meinem Diplomarbeitsbetreuer Herrn Horst Eidenberger, welcher mir mit seinem fachkundigen Wissen stets mit Rat und Tat zur Seite stand.

Ein herzliches Dankeschön geht auch an alle Damen und Herren, die sich die Zeit genommen haben, um an der erstellten Online-Umfrage teilzunehmen.

Kurzfassung

Die vorliegende Diplomarbeit beschäftigt sich mit Ansätzen des algorithmischen Komponierens. Verfahren aus diesem Bereich sollen dabei mit der Disziplin der Mustererkennung vereint werden. Das Ziel der Arbeit ist die Generierung von neuen Musikstücken, welche deutliche Parallelen zu zuvor ausgewählten Kompositionen aufweisen. Zu diesem Zweck werden kurze und prägnante Notenfolgen, sogenannte Motive, aus diversen Liedern extrahiert. Die so gewonnenen Teile dienen anschließend als Input für Verfahren der algorithmischen Komposition. Die erstellten Musikstücke sollen einen möglichst harmonischen Klangeindruck hinterlassen. Im Zuge dieser Diplomarbeit wurde ein Software-System implementiert, welches in der Lage ist, Motive in Musikstücken zu finden und diese im Anschluss zu extrahieren. Weiters wurden zwei konträre Verfahren der algorithmischen Komposition ausgewählt, welche sich in der umgesetzten Software aus den Ergebnissen der Motiv-Erkennung bedienen. Auf dieser Basis können Musikstücke im MIDI-Format generiert werden. Als Hinführung auf das Thema der Motiv-Extraktion und der algorithmischen Komposition werden zunächst unterschiedliche Verfahren und Ansätze im Detail erläutert. Im Anschluss wird das Konzept der implementierten Software vorgestellt. Am Ende der Arbeit erfolgt die Präsentation der erzielten Ergebnisse anhand der Evaluierung einer quantitativen Umfrage.

Abstract

This master thesis deals with different approaches for algorithmic composing. Such techniques are merged with methods of pattern recognition. The aim of the thesis is the generation of new pieces of music, which are derived from existing musical content. For this purpose short and concise note patterns, called motifs, are extracted from arbitrary musical input. These patterns are used as the basis for approaches of algorithmic composing. The generated musical pieces should sound as harmonic as possible. As part of this master thesis, a software-system was developed, which can perform the task of identifying and extracting motifs in music. Furthermore, two contrary techniques of algorithmic composing, which both use the extracted motifs as input, were implemented in the software. Based on those composers, new pieces of music can be generated. Those compositions employ the MIDI standard. Furthermore, as an introduction to the world of motif extraction and algorithmic composing, different approaches and techniques are explained in detail in the literature survey. Subsequently, the design of the implemented software is presented. Eventually, the results are evaluated, based on a quantitative survey.

Inhaltsverzeichnis

1 Einleitung

Die Anfänge des Automated Composings gehen bis zur Mitte des 20. Jahrhunderts zurück. Unter Zuhilfenahme des ILLIAC Computers schufen Lejaren Hiller und Leonard Isaacson 1956 die erste computergenerierte Komposition in der Geschichte. Die damals entstandene "Illiac Suite" zählt zur Kategorie des Automated Composings. Die Ausgabe der Prozedur stellte auf symbolischer Ebene Notenwerte dar. Diese mussten erst noch von einem Musiker interpretiert werden [1]. Dennoch entsprach der Output einer vollständigen Komposition. Auch heute unterscheidet man zwischen Automated Composing sowie Assisted Composing. Im Gegensatz zur zweiten Kategorie ist das Ziel des Automated Composings, ein finales Musikstück zu generieren, das keine Nachbearbeitung mehr erfordert.

Seit den Anfängen in den 50er Jahren hat sich auf dem Gebiet der computergenerierten Musik viel getan. Software, die Komponisten beim Schaffen ihrer Werke unterstützend bzw. inspirierend unter die Arme greifen soll, wird häufig eingesetzt. Vollautomatisches Komponieren hingegen hat sich im Gegensatz dazu im kommerziellen Bereich kaum durchgesetzt. Auch wenn diverse Programme durchaus gute Ergebnisse liefern, fehlt hier die menschliche Kreativität.

1.1 Zielsetzung

Der *Pattern-Based MIDI Composer (PBMC)* reiht sich vornehmlich in die Sparte des Automated Composings ein, beinhaltet jedoch in groben und abstrakten Zügen auch Ansätze des Assisted Composings. Der Output der Software soll also Musikstücken entsprechen, die möglichst harmonisch und wohlklingend sind. Eine Weiterverarbeitung ist nicht vorgesehen.

Der *PBMC* unterscheidet sich jedoch in einem Punkt ganz wesentlich von anderen Produkten bzw. Projekten auf diesem Gebiet. Es wird hier nicht von gänzlich neuen Kompositionen ausgegangen, sondern bereits Bestehendes als Ausgangsmaterial he-

rangezogen. Dieses wird aus ausgewählten Musikstücken extrahiert und als Grundlage für die weitere Verarbeitung bzw. Komposition genutzt.

In nahezu jedem bekannten Musikstück gibt es Notenfolgen, die im Verlauf des Stücks wiederkehren. Diese Repetitionen können in sich variiert oder auch transponiert sein. Ebenso ist eine unterschiedliche Anzahl von Noten möglich. Solche Teile eines Liedes werden in der Formenlehre der Musik auch als *Motive* bezeichnet. Sie bestehen üblicherweise aus wenigen Noten, können allerdings auch längere Notenfolgen umfassen. So kann z.B. jeder Refrain als Motiv angesehen werden.

Ein wesentlicher Vorverarbeitungsschritt des *PBMC* versucht mithilfe von Mustererkennungsalgorithmen solche Motive zu extrahieren. Diese werden in weiterer Folge an den Kompositionstask weitergereicht, um sie dort einfließen zu lassen. Als Ergebnis soll, wie anfangs erwähnt, kein völlig neues Musikstück generiert werden, sondern eine Abwandlung des Eingangsmaterials erfolgen. Aufgrund dieser Tatsache lässt sich der *PBMC* in weiten Zügen auch als "umgekehrter Assisted Composer" beschreiben. Es erfolgt zwar keine Weiterverarbeitung durch einen Komponisten, jedoch wird bereits Komponiertes als Grundlage herangezogen.

Ziel des Einsatzes des *PBMC* ist also das vollautomatische Generieren einer Komposition, welche deutlich erkennbare Parallelen zu ausgewählten Eingabestücken aufweisen soll. Außerdem soll diese Komposition den gängigen Gesetzen der Harmonielehre folgen. Das Ausgabemusikstück lässt sich als wohlklingende Mutation von verschiedenen existierenden Kompositionen beschreiben.

1.2 Struktur der Arbeit

Diese Arbeit gliedert sich in sieben Kapitel, die wie folgt aufgebaut sind:

- Kapitel 2 behandelt die *Grundlagen* der eingesetzten Techniken und Verfahren. Zunächst wird näher auf die Varianten von computergestützter Komposition eingegangen. Weiters werden die Grundzüge der Motiv-Erkennung erläutert.

Abschließend wird ein kurzer Einblick in den Aufbau und die Funktionsweise von MIDI geboten.

- Kapitel 3 behandelt den aktuellen *Stand der Technik* in den Gebieten Motiv-Erkennung und Computergestützte Komposition. In diesem Abschnitt findet ebenfalls Software aus dem Bereich des Automated bzw. Assisted Composings Erwähnung.
- Kapitel 4 beschäftigt sich mit dem theoretischen *Entwurf* des *PBMC*. Beginnend mit einer Anforderungsanalyse und der Beschreibung des Systemaufbaus soll hier ein guter Einstieg in die Systematik der Software ermöglicht werden. In weiterer Folge wird die Funktionsweise der einzelnen Module des *PBMC* genauer erläutert.
- Kapitel 5 beinhaltet konkrete Details zur *Implementierung* des *PBMC*. Unter diesem Aspekt erfolgen erneut Erläuterungen zu den einzelnen Teilen der Software. Verwendete Frameworks finden hier ebenso Erwähnung wie Probleme, die während der Implementierung aufgetreten sind. Abschließend wird die finale Version des *PBMC* vorgestellt und auf deren Funktionsweise eingegangen.
- Kapitel 6 beinhaltet die *Evaluierung* der Ergebnisse des *PBMC*. Zu diesem Zweck wurde ein Fragebogen zu den generierten Musikstücken und extrahierten Motiven erstellt. Die Auswertung der Ergebnisse dieser Umfrage wird hier behandelt.
- Kapitel 7 beinhaltet abschließende und zusammenfassende *Schlussfolgerungen*. Weiters wird hier auf mögliche weiterführende Arbeiten am PBMC eingegangen.

2 Grundlagen

Der Ansatz des Pattern-Based MIDI Composings vereint zwei gänzlich verschiedene Disziplinen. Zum einen werden Techniken aus dem Gebiet der computergestützten Komposition angewandt, zum anderen finden sich Grundzüge der Mustererkennung in der Motiv-Extraktion wieder. Die Verarbeitung der Musikstücke erfolgt unter Zuhilfenahme des *Musical Instrument Digital Interfaces (MIDI)*. In den folgenden Abschnitten werden die technischen und funktionellen Grundlagen dieser drei Teilgebiete erläutert.

2.1 Computergestützte Komposition

Grundsätzlich folgen Kompositionen immer gewissen Regeln, Techniken oder formalen Methoden. Zusammen mit dem kreativen Geschick eines begabten Komponisten entstehen dadurch wohlklingende Musikstücke. Aufbauend auf diesen üblichen Strukturen in der Musik lassen sich einzelne Stücke berechnen.

2.1.1 Geschichte

Wie in [2] erläutert, steht Musik seit Anbeginn im Zeichen formaler Regeln. Im weitesten Sinne hat dies bereits im 10. Jhdt. durch die Einführung der Musik-Notation begonnen. Hier wurde unter Zuhilfenahme von Notenlinien und unterschiedlich platzierten Symbolen für die verschiedenen Tonhöhen eine Umsetzung von textuell beschriebenen Notenwerten zu grafisch bestimmten Tonfolgen geschaffen.

Auch später in der Geschichte gibt es noch einige Beispiele, in denen offensichtlich erste Ansätze von algorithmischem Komponieren angewandt wurden. Besonders interessant ist die Tatsache, dass im Laufe des 18. Jahrhunderts die Idee aufkam, Tabellen aus einzelnen Musikelementen zu verwenden um auch Laien zu ermöglichen, wohl-strukturierte und harmonische Kompositionen zu kreieren. Die einzelnen Teile solcher Musik-Matrizen wurden dann zufällig aneinandergereiht [3]. Die konkrete

Zusammensetzung wurde meist durch das Werfen von Würfeln bestimmt. Diese Methode brachte zur damaligen Zeit diverse Würfelspiele hervor. Die Funktionsweise wird in [1] mit einigen simplen Schritten erläutert:

1. Tabelle mit verschiedenen musikalischen Konstellationen auswählen
2. Zufälliges Element der Tabelle auswählen
3. Ausgewähltes Element ans Ende der Komposition reihen
4. Wiederhole Schritte 2 und 3, bis die Komposition fertig ist

Eines der bekanntesten Spiele dieser Art ist das *Musikalische Würfelspiel*, das Wolfang Amadeus Mozart (1756-1791) zugeschrieben wird. Es wurde allerdings erst 1792, ein Jahr nach seinem Tod, veröffentlicht. In diesem Würfelspiel wurden zwei Tabellen verwendet. Jede davon besteht aus acht Spalten und 11 Zeilen (siehe Abbildung 2.1).

	I	II	III	IV	V	VI	VII	VIII
2	96	22	141	41	105	122	11	30
3	32	6	128	63	146	46	134	81
4	69	95	158	13	153	55	110	24
5	40	17	113	85	161	2	159	100
6	148	74	163	45	80	97	36	107
7	104	157	27	167	154	68	118	91
8	152	60	171	53	99	133	21	127
9	119	84	114	50	140	86	169	94
10	98	142	42	156	75	129	62	123
11	3	87	165	61	135	47	147	33
12	54	130	10	103	28	37	106	5

	I	II	III	IV	V	VI	VII	VIII
2	70	121	26	9	112	49	109	14
3	117	39	126	56	174	18	116	83
4	66	139	15	132	73	58	145	79
5	90	176	7	34	67	160	52	170
6	25	143	64	125	76	136	1	93
7	138	71	150	29	101	162	23	151
8	16	155	57	175	43	168	89	172
9	120	88	48	166	51	115	72	111
10	65	77	19	82	137	38	149	8
11	102	4	31	164	144	59	173	78
12	35	20	108	92	12	124	44	131

Abb. 2.1: Tabellen aus dem musikalischen Würfelspiel

Die Elemente der Tabelle beinhalten die Nummern einzelner Takte die vorgegeben waren. Wollte man nun ein Stück erstellen, würfelte man mit zwei Würfeln um so die Zeile innerhalb der ersten Tabelle zu bestimmen. In der Folge wurde der Takt mit der entsprechenden Nummer aus der gewürfelten Zeile in der ersten Spalte verwendet (siehe Abbildung 2.2).

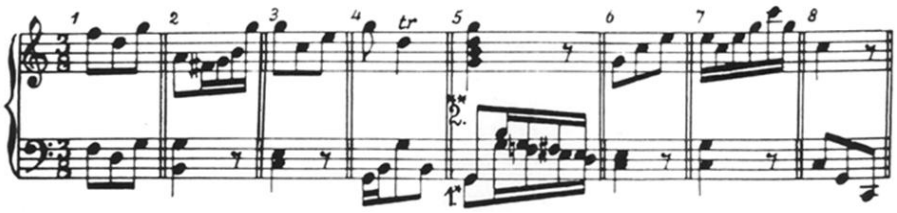

Abb. 2.2: Beispieltakte aus dem musikalischen Würfelspiel [4]

Im nächsten Wurf wählte man den entsprechenden Wert aus der zweiten Spalte. Diese Vorgehensweise wurde in Summe 16 Mal wiederholt, bis alle Spalten beider Tabellen durchlaufen waren. Jeder einzelne dieser 176 Takte war so geschrieben, dass sich die unterschiedlichen Elemente sehr gut zusammenfügten. Dieses System erlaubte eine theoretische Menge von 11^{16} möglichen Kompositionen. In der Realität waren es jedoch weniger, da einige der Takte aus der jeweils achten Spalte der Tabellen identisch waren [3].

Diese historischen Kompositionshilfen für Laien zeigen, dass es prinzipiell möglich ist, das Generieren von Musik zu formalisieren und dass dies auch damals schon bekannt war. Der erste große Durchbruch auf dem Sektor der automatisierten Komposition gelang allerdings erst im Computer-Zeitalter. Lejaren Hiller (1924-1994) gilt dabei als der Begründer der computergestützten Komposition.

Gerhard Nierhaus beschreibt in [1], welche Bedeutung die fortschreitende Entwicklung von Computern für die algorithmische Komposition hatte. Einen wesentlichen Meilenstein stellte der EDVAC (Electronic Delay Storage Automatic Computer) dar. Die Planung dieses Computers ist den Entwicklern des ENIAC I zuzuschreiben. Der EDVAC ermöglichte es später auch, Programm-Kommandos im Speicher abzulegen. Das Konzept dazu stammte von John von Neumann (1903-1957). Der Computer wurde 1952 fertiggestellt, Neumanns Überlegungen waren zu dieser Zeit allerdings schon im IAS Computer umgesetzt. Dieser wurde von 1945 bis 1960 am Princeton Institute of Advanced Studies eingesetzt. Auf Basis des IAS wurden zu dieser Zeit auch einige andere Computer gebaut. Einer davon war der ILLIAC, welcher an der Universität von Illinois verwendet wurde. Mit Hilfe dieses Rechners gelang es 1956 Lejaren Hiller und Leonard Isaacson, die erste computergenerierte Komposition zu

kreieren. Die so entstandene *Illiac Suite* stellt heute den Beginn der computergestützten Komposition dar. Sie produzierte Notenwerte auf symbolischer Ebene, die erst noch von einem Musiker interpretiert werden mussten. Trotzdem entsprach der Output einer vollwertigen Komposition. Die Arbeit von Hiller und Isaacson lässt sich also in den Bereich des Automated Composings einreihen. Genauere Erläuterungen zu den Unterschieden zwischen Automated Composing und Assisted Composing finden sich im Kapitel 2.1.2.

Mit der Einführung höherer Programmiersprachen entstanden auch die ersten dedizierten Softwaresysteme zur algorithmischen Komposition. Im Zeitraum von 1960 bis in die frühen 90er Jahre wurden hier einige Projekte geschaffen, von denen Weiterentwicklungen teilweise auch heute noch im Einsatz sind.

2.1.2 Automated versus Assisted Composing

Die Bezeichnungen des Automated bzw. des Assisted Composings werden sehr häufig als äquivalent angesehen. In Wahrheit gibt es jedoch große technische und anwendungsbezogene Unterschiede. Eine Gleichsetzung beider Ausdrücke ist also per Definition falsch. Der *PBMC* gliedert sich in die Sparte des Automated Composings ein. In [5] werden klare Definitionen zu den Unterschieden wie folgt erläutert:

> **Automated Composing** *versucht ohne menschliches Zutun, vollautomatisch generierte Musikstücke zu kreieren. Dieser Ansatz wird sehr häufig in der Forschung angewandt. Dabei wird oft versucht Kompositionen zu erstellen, die den Stil eines anderen Musikstücks oder -genres kopieren. Die Evaluierung der Ergebnisse kann hier zielführend mittels Turing-Tests durchgeführt werden. Dabei werden Probanden gefragt, ob sie eine gehörte Komposition einem menschlichen Komponisten oder einem Computer zuschreiben würden. Automated Composing wird im Wesentlichen nicht in der kommerziellen Musikbranche eingesetzt.*

Assisted Composing *hingegen dient der Unterstützung von Komponisten im Prozess des Komponierens. Es soll keine wissenschaftliche Evaluierung stattfinden und auch keine Kopien von bereits existierendem Material angefertigt werden. Hier soll der Computer tatsächlich nur die Arbeit eines Komponisten erleichtern. Erstellte Musikstücke stammen im Wesentlichen aus der Hand des Anwenders. Die Software hilft ihm dabei bestimmte Aufgaben zu bewältigen und gibt Vorschläge. Der kreative Schaffensprozess des Menschen steht hier ganz klar im Vordergrund. Assisted Composing wird sehr häufig in der Musikindustrie angewandt.*

2.1.3 Verfahren

Zu Beginn des computergestützten Komponierens stand noch eine verhältnismäßig kleine Anzahl an Möglichkeiten zur Verfügung, um diese Aufgabe zu bewältigen. Hiller verwendete für seine *Illiac Suite* diverse Regeln, um adäquate Lösungen zu generieren. Im Laufe der Zeit wurden jedoch neue Techniken und Verfahren entwickelt, die eingesetzt werden konnten. Diese sind dabei nicht speziell für das Komponieren erdacht worden, finden jedoch auch hier Einsatz. Teilweise leicht modifiziert, stellen diese Ansätze gute Grundlagen dar, um computergenerierte Musik zu erzeugen.

Nach [6] lassen sich die verschiedenen Techniken im Wesentlichen in die folgenden Kategorien gliedern:

1. Formale Grammatiken
2. Iterativer Ansatz
3. Neuronaler Ansatz
4. Stochastischer Ansatz
5. Evolutionärer Ansatz

Nachfolgend werden die diversen Ansätze im Sinne von [1, 6] näher erläutert. Beginnend wird auf formale Grammatiken, den iterativen und den neuronalen Ansatz

eingegangen. Abschließend werden der stochastische sowie der evolutionäre Ansatz im Detail erklärt. Verfahren dieser beiden Ansätze wurden im *PBMC* implementiert. Auf die konkrete Funktionsweise in der Software wird im Kapitel 4 eingegangen. Hier soll im Wesentlichen ein Überblick über die erwähnten Methoden gegeben werden.

Formale Grammatiken

Formale Grammatiken sind grundsätzlich Teil der Sprachwissenschaft. Der Begriff wurde durch Noam Chomsky geprägt. Nach seiner Theorie ist das Verstehen und Sprechen einer Sprache deshalb möglich, weil der Mensch die Fähigkeit besitzt, die formalen Grammatiken dieser zu verstehen. Er geht dabei von zwei unterschiedlichen Grammatikstrukturen aus. Zum einen existiert laut Chomsky eine Tiefenstruktur, welche die allgemeine Struktur beschreibt, die jede Sprache besitzt. Außerdem geht er von einer Oberflächenstruktur aus, welche spezifische Eigenschaften einzelner Sprachen beschreibt. Diverse Sprach- und Musikwissenschaftler glaubten, dass Chomskys Theorie sich ebenso auf Musik anwenden ließe.

Um formale Grammatiken in der Komposition zu verstehen, ist es hilfreich, zunächst deren Struktur in der Sprachwissenschaft zu erläutern. Die Umlegung auf musikalische Notationen ist dann relativ einfach. Zunächst wird davon ausgegangen, dass ein Satz (S) immer aus einem Nomensatz (NS) und einem Verbsatz (VS) besteht. Ein Nomensatz setzt sich dabei aus einem Artikel (A) und einem Nomen (N) zusammen. Ein Verbsatz kann als die Zusammensetzung eines Verbs (V) und eines Nomensatzes verstanden werden.

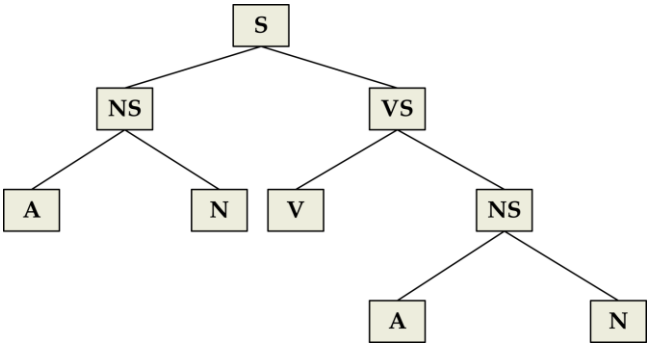

Abb. 2.3: Grammatik-Struktur in der Sprachwissenschaft [6]

Folgt man diesem grammatikalischen Aufbau, kommt man zwar zu strukturell richtigen Sätzen, jedoch kann es zu inhaltlich unsinnigen Varianten kommen. Diese Tatsache spielt bei der Komposition keine Rolle, da sich die Sinnhaftigkeit hier ausschließlich durch die Grammatik ergibt. Will man den strukturellen Aufbau (siehe Abbildung 2.3) nun auf musikalische Strukturen umlegen, ist zunächst eine Modifikation der einzelnen Elemente notwendig.

Aus diesem Grund ist es erforderlich, die einzelnen Teile der Grammatik zu bestimmen. In Anlehnung an [6] kann man von fünf unterschiedlichen Elementen ausgehen:

- Referenznote (Ref)
- Intervall zwischen zwei Noten (I)
- Richtung (R)
- Sequenz (Seq)
- Gleichzeitigkeit (G)

Eine Note wird hier als Intervall und dessen Richtung, bezugnehmend auf die Referenznote, angegeben. Die Richtung zeigt dabei an, ob die Folgenote über oder unter der Referenznote liegt. Eine Sequenz wiederum beinhaltet eine oder mehrere Noten. Mehrere Sequenzen können schließlich durch ein Gleichzeitigkeits-Element kombiniert werden. Ein beispielhafter Aufbau einer solchen Grammatik ist in Abbildung 2.4 dargestellt.

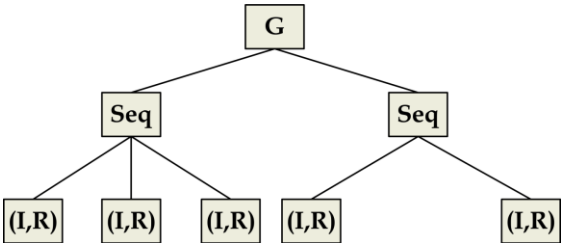

Abb. 2.4: Grammatik-Struktur in der Musik [6]

In diesem Beispiel würden durch den linken Ast drei Noten erzeugt werden. Simultan dazu bestünde, durch den rechten Ast, eine zweite Stimme aus zwei Noten.

Iterativer Ansatz

Iterative Algorithmen beschreiben mathematische Prozesse, deren Input stets vom vorangegangenen Output abhängt. Es wird dabei von einem festgelegten Startwert x_0 ausgegangen. In jedem Iterationsschritt wird der Input von einer mathematischen Regel F verarbeitet und anschließend wieder als Input der nächsten Iteration herangezogen.

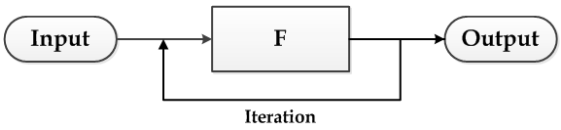

Abb. 2.5: Aufbau eines iterativen Algorithmus

Die Ergebnisse eines iterativen Algorithmus bilden einen sogenannten Orbit. Jedes Resultat des Algorithmus stellt dabei einen Punkt in diesem Orbit dar. Umgelegt auf computergestützte Komposition repräsentiert jeder dieser Punkte einen musikalischen Parameter (z.B. Tonhöhe).

Grundsätzlich können laut [6] solche Algorithmen die drei folgenden Klassen von Orbits bilden:

1. Punkte konvergieren gegen einen stabilen Wert
2. Punkte oszillieren zwischen spezifischen Elementen
3. Punkte liefern ein chaotisches Ergebnis

Einen besonders bekannten Orbit stellt die Mandelbrot-Menge dar. Diese grafisch fraktal erscheinende Menge wird durch die iterative Regel $z_{n+1} = {z_n}^2 + c$ definiert, wobei c einer komplexen Zahl entspricht. Die grafische Darstellung der Mandelbrot-Menge ist in Abbildung 2.6 ersichtlich.

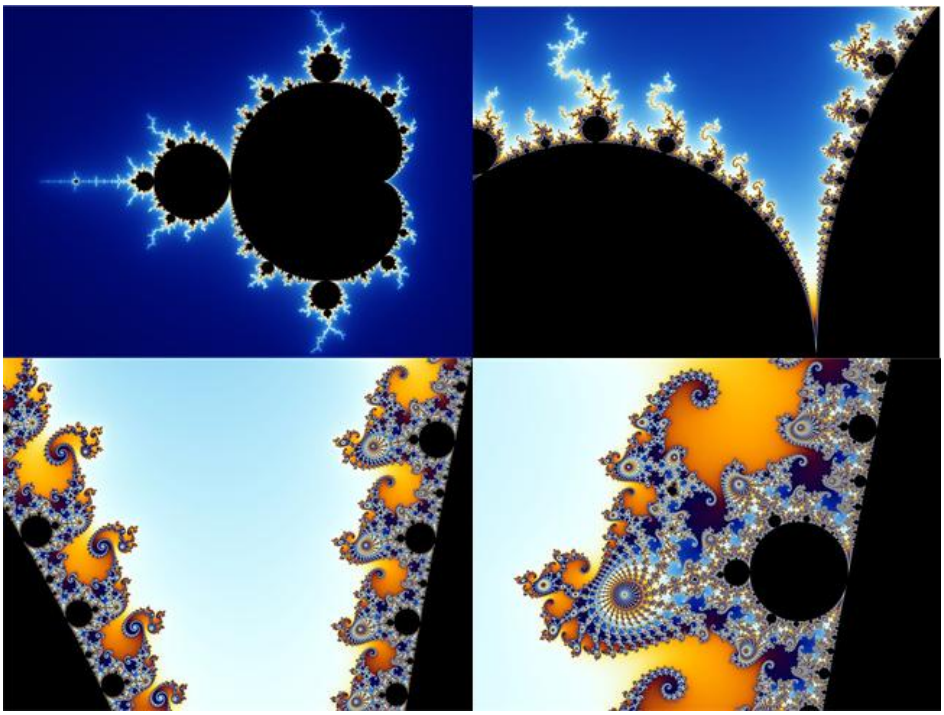

Abb. 2.6: Mandelbrot-Menge in unterschiedlichen Zoomstufen [7]

Die Orbit-Klassen 1 und 2 sind für die Generierung von Musik eher ungeeignet. Klasse 1 würde durch ihre konvergierende Form sehr eintönige und langweilige musikalische Ergebnisse liefern. Durch die Oszillation in einer hohen Frequenz wäre das Resultat von Klasse 2 wohl am ehesten als störend zu bezeichnen. Die beste Wahl zur Komposition von Musik stellt also ein chaotischer Orbit dar. Hierbei wandern die Punkte innerhalb des Orbits in einem definiert beschränkten Bereich. Ein neuer Punkt trifft kaum auf einen bereits existierenden, sondern befindet sich allenfalls in der Nähe eines solchen.

Besonders wichtig für den Ansatz der iterativen Algorithmen ist in jedem Fall ein geeigneter Startwert. Je nach Auswahl können hier kleinste Veränderungen schon große Auswirkungen haben. Die Wahl der Regel-Funktion hängt davon ab, wie viele musikalische Parameter man steuern möchte. Mehrere Parameter können entweder

durch mehrere eindimensionale Orbits bestimmt werden oder durch Regeln mit mehr als einer Gleichung. Die Bestimmung der Regel ist dabei eine Gratwanderung. Sie sollte weder zu simpel noch zu komplex sein. Es gibt bisher noch keine klare Definition, wie eine Regel aufgebaut sein sollte, um gute musikalische Lösungen zu erzeugen.

Neuronaler Ansatz

Neuronale Netze versuchen die Reizweiterleitung des Gehirns zu simulieren. Dieses besteht aus einem Netz von Neuronen, welche durch Spannungsübertragungen aktiviert werden. Neuronen können dabei auf Reize ausgehend von anderen Neuronen sowie von außerhalb reagieren. Die Neuronen stellen dabei sowohl im Gehirn wie auch in künstlichen neuronalen Netzwerken den zentralen Baustein dar.

Solche Netze müssen trainiert werden, um adäquate Ergebnisse zu liefern. Dazu werden sie mit Aufgaben gefüllt, für die die Lösung bekannt ist. Ihr Verhalten wird dadurch auf richtige Ergebnisse angepasst. Werden genug Beispiele im Training verwendet, lernt das Netz aus deren Lösungen. Ist das Training abgeschlossen und tritt ein ähnliches Problem auf, wird das zuvor gelernte Verhalten herangezogen um es zu lösen.

Zum Training neuronaler Netze gibt es die drei folgenden Herangehensweisen:

1. Überwachtes Lernen
2. Bestärkendes Lernen
3. Unüberwachtes Lernen

Beim ü*berwachten Lernen* werden die Parameter des neuronalen Netzes so angepasst, dass das Ergebnis des Trainings-Inputs der korrekten Lösung entspricht. Nachdem es nicht immer möglich ist zu jedem Input Datensatz auch das entsprechend korrekte Ergebnis zu liefern, wird beim *bestärkenden Lernen* nur das Ergebnis bewertet. Das neuronale Netz findet also selbstständig, ohne äußeren Einfluss, eine Lösung. Ist diese gut, wird das System bestärkt. Das *unüberwachte Lernen* erhält im Gegensatz zu

den beiden anderen Verfahren keine äußerlichen Einflüsse. Das Netz passt seine Parameter also selbstständig entsprechend der Trainingsdaten an.

Eines der ersten künstlichen neuronalen Netzwerke stellte das Perzeptron dar. Es wurde in den 50er Jahren von Frank Rosenblatt entwickelt. Das Perzeptron besteht aus n Input-Neuronen, die unterschiedlich gewichtet sind. Zusätzlich existiert ein Bias-Term, der die gewichteten Input-Daten nochmal erhöhen oder erniedrigen kann. Die gesammelten Input-Werte werden dann an eine Aktivierungs- oder Transferfunktion weitergegeben. Diese dient dazu, die Output-Werte auf einen definierten Bereich zu normalisieren.

Zu Komposition von Musik wurde 1999 von Kenny McAlpine ein einfaches neuronales Netz (siehe Abbildung 2.7) vorgestellt, das den Zweck hatte, monophone Melodien zu generieren. Hierfür ist nur eine Ausgabe-Einheit notwendig. Das Ergebnis dieser entspricht dabei der Tonhöhe einer Note im MIDI-Format. Dieses Netz liefert außerdem jede Ausgabe zurück zu den Eingabe-Einheiten. Aus diesem Grund sind die folgend generierten Ergebnisse stets abhängig von vorangegangenen Lösungen. Das von McAlpine erdachte Netz besitzt drei Verzögerungs-Einheiten. Diese dienen dazu, die letzten drei Ergebnisse zu sichern, um sie in die nächste Lösung einfließen zu lassen. Um mehrere musikalische Parameter zu steuern, kann man entweder mehrere Netze parallel verwenden oder zusätzliche Pfade in ein Netz einfügen.

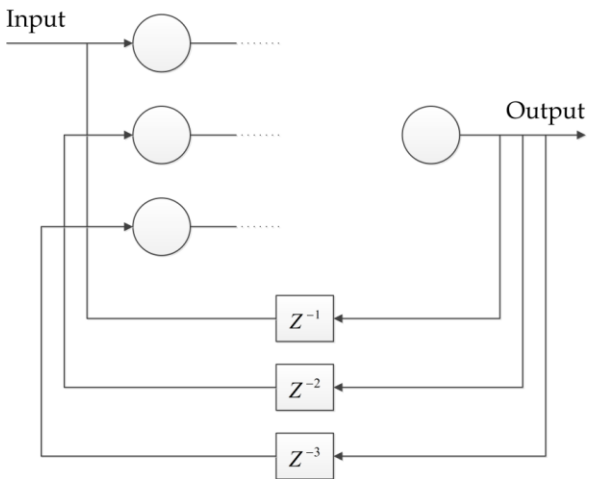

Abb. 2.7: Neuronales Netz zur Kreierung von monophonen Melodien [6]

Zum Training dieses neuronalen Netzes werden die Gewichte der einzelnen Neuronen zu Anfang zufällig bestimmt. Anschließend werden existierende Melodien Note für Note in das System eingegeben. Im Zuge dieses Prozesses werden die Gewichte der einzelnen Neuronen dann so verändert, dass die nächste berechnete Note auch der nächsten Note der Eingabe-Melodie entspricht. Es handelt sich hierbei also um *überwachtes Lernen*.

Stochastischer Ansatz

Bei der Anwendung von Wahrscheinlichkeiten, um Musik zu generieren, wird aus einem Set an möglichen Werten zufällig einer als nächster Output ausgewählt. Kommen in diesem Set an möglichen Ergebnissen keine doppelten Werte vor, spricht man von *fair trial*. Gibt es umgekehrt mehrfache Vorkommnisse eines Wertes, ist die Wahrscheinlichkeit diesen auszuwählen höher.

Markov-Ketten stellen ein Werkzeug dar, mit welchem die Auswahl eines wahrscheinlichen Ergebnisses erzielt werden kann. Der Output hängt dabei immer von einem oder mehreren vergangenen Ereignisse ab. Wie viele Ereignisse in die Berechnung der Wahrscheinlichkeiten mit einbezogen werden, wird durch die Ordnung n angegeben.

		Zukünftige Ereignisse		
		A	**B**	**C**
Vergangene Ereignisse	**A**	0.2	0.1	0.7
	B	0.6	0.3	0.1
	C	0.1	0.4	0.5

Abb. 2.8: Markov-Matrix erster Ordnung

Abbildung 2.8 zeigt den Aufbau einer Markov-Matrix der Ordnung 1. Es wird also immer nur das letzte Ereignis herangezogen, um die Wahrscheinlichkeit des folgenden Ergebnisses zu bestimmen. In der ersten Spalte werden dabei die vergangenen Ereignisse (A, B, C) angegeben, in der ersten Zeile die möglichen folgenden Ereignis-

se. Im Beispiel würde nach dem Ereignis A mit 10% Wahrscheinlichkeit das Ereignis B eintreten.

Sollen Markov-Ketten für das Generieren von Musik verwendet werden, wählt man zunächst eine Menge möglicher Noten anhand eines der musikalischen Parameter (z.B. Tonhöhe) aus. Folgend müssen nun Regeln bestimmt werden, wie Noten aufeinander folgen können. Eine solche Regel könnte besagen, dass auf C4 nur D4, E4, A4 und C5 zu jeweils 25% Wahrscheinlichkeit folgen können. Die Wahrscheinlichkeit muss jedoch nicht gleich verteilt sein, sondern kann auch variieren. Mit den so festgelegten Regeln lässt sich nun wieder eine Wahrscheinlichkeitstabelle aufstellen. Als Startwert wird zufällig ein Ereignis der möglichen Werte ausgewählt. Folgenoten werden aufgrund ihrer Wahrscheinlichkeiten ausgewählt. Auf diese Art und Weise wird eine Melodie aus n Noten generiert. Soll mehr als ein Parameter berechnet werden, können zusätzliche Markov-Ketten verwendet werden.

Evolutionärer Ansatz

Genetische Algorithmen basieren wie auch künstliche neuronale Netze auf biologischen Gegebenheiten. Hier werden Parallelen zur menschlichen DNA gezogen. Diese besteht aus Chromosomen, die sich wiederum aus einzelnen Genen zusammensetzen. Wie in Charles Darwins Evolutionstheorie beschrieben, verändert sich das Genmaterial von Lebewesen über die Zeit. So entstehen neue, angepasste Populationen von Lebewesen.

Wie im biologischen Vorbild beginnt auch ein genetischer Algorithmus mit einer Grundpopulation. Diese besteht aus n Chromosomen, die wiederum aus n Genen bestehen. In welcher Form diese Gene dem Algorithmus vorliegen, hängt vom Einsatzzweck ab. Der Zyklus einer genetischen Iteration besteht immer aus denselben Schritten (siehe Abbildung 2.9).

Ein genetischer Algorithmus terminiert, wenn entweder der errechnete Fitnesswert einer Generation einen definierten Schwellwert überschreitet oder eine maximale Anzahl an Iterationen erreicht wurde.

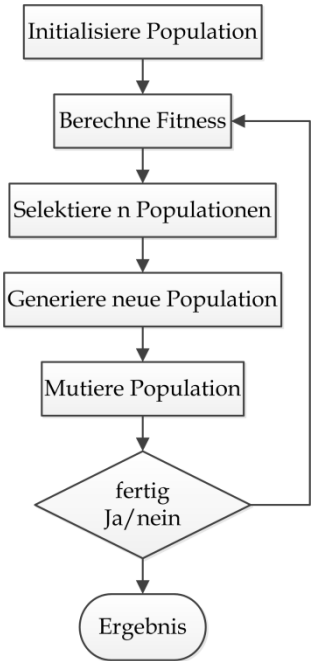

Abb. 2.9: Schematische Ablauf-Darstellung eines genetischen Algorithmus

Eine leicht abgewandelte Grundstruktur eines einfachen genetischen Algorithmus ist, wie in [8] erläutert, in Algorithmus 1 dargestellt.

```
Choose an initial population of chromosomes;
while termination condition not satified
{
    do
        if crossover condition satisfied
        {
            select parent chromosomes;
            choose crossover parameters;
            perform crossover
        }
        if mutation condition satisfied
        {
            select mutation chromosome(s);
            choose mutation points;
            perform mutation;
        }
        evaluate fitness of offspring;
    while not sufficient offspring created;
    select population;
}
```

Algorithmus 2.1: Pseudocode genetischer Algorithmus [8]

Grundpopulation: Bevor der genetische Zyklus das erste Mal durchlaufen wird, muss die Population manuell festgelegt werden. Im Regelfall geschieht dies durch eine zufällige Initialisierung der Chromosomen in der Population.

Fitness: Die Funktion zur Berechnung der Fitness eines Chromosoms ist üblicherweise eine mathematische Funktion, die die Wertigkeit eines Chromosoms beurteilt. Wie genau diese Funktion aufgebaut ist, hängt wiederum von der entsprechenden Aufgabenstellung ab.

Selektion: Zur Selektion von Chromosomen gibt es unterschiedliche Verfahren. Sehr einfach ist die Auswahl der Elemente mit dem höchsten Fitnesswert. Es gibt jedoch auch andere Verfahren, die der Selektion ein höheres Maß an Zufälligkeit verleihen. Ein Beispiel hierfür ist die Tournament-Selektion, welche n zufällige Chromosomen auswählt und davon jenes Chromosom selektiert, das den höchsten Fitnesswert besitzt.

Crossover: Um aus selektierten Chromosomen eine neue Generation zu generieren, wird bei genetischen Algorithmen das Crossover verwendet. In diesem Schritt werden zwei Chromosomen miteinander kombiniert. Dies kann wiederum mit unterschiedlichen Verfahren geschehen. Ein Beispiel ist das sogenannte One-Point-Crossover (siehe Abbildung 2.10), bei dem die erste Hälfte des ersten Chromosoms mit der zweiten Hälfte des zweiten Chromosoms verbunden wird. Ebenso möglich ist die Verknüpfung zweier Chromosomen durch Two-Point-Crossover. In dieser Variante erfolgt die Kombination durch ein Aneinanderreihen von gedrittelten Eltern-Chromosomen.

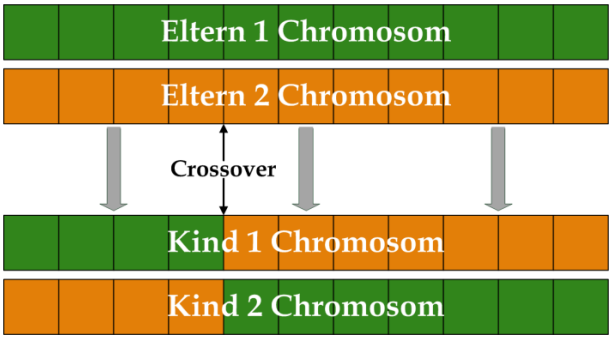

Abb. 2.10: One-Point-Crossover [9]

Mutation: Zusätzlich kann ein Chromosom einer neuen Generation auch mutieren. Mutationen können sich auf einzelne Gene eines Chromosoms auswirken (siehe Abbildung 2.11), aber auch längere Genstränge beeinflussen. Dieser Schritt trägt neben dem Crossover wesentlich dazu bei, zu verbesserten Chromosomen im Sinne der Fitness-Funktion zu gelangen.

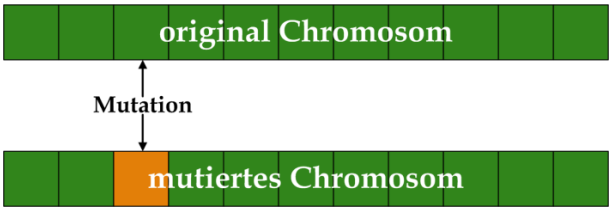

Abb. 2.11: Mutation eines Chromosoms [9]

Genetische Algorithmen werden vorwiegend zu Such- und Optimierungszwecken eingesetzt. Ein weiteres Einsatzgebiet stellt auch die Generierung von Kompositionen dar. Hierbei ist die schwierigste Aufgabe, passende Fitness-Funktionen zur Evaluierung der Wertigkeit eines Musik-Chromosoms zu finden. Die Selektion und das Crossover funktionieren hier ähnlich wie bei anderen genetischen Algorithmen. Mutationen können durch die Anwendung simpler Transformationsfunktionen auf einer Note oder Notenfolge implementiert werden. Denkbare Transferfunktionen sind z.B. die zufällige Veränderung der Notendauer, die Sortierung nach Tonhöhe innerhalb einer Notenfolge oder das Kopieren einer Notenfolge an eine andere Position im Chromosom.

2.2 Motiv-Erkennung

Nach dem zentralen Kernstück des *PBMC,* der Komposition, stellt die Motiv-Erkennung bzw. die Motiv-Extraktion den zweiten wesentlichen Bestandteil dieser Arbeit dar. Signifikante Passagen eines Musikstücks zu erkennen und deren Zusammenhänge zu verstehen, ist eine verhältnismäßig einfache Aufgabe für einen Menschen. Die verlässliche Erkennung solcher durch den Einsatz digitaler Technik kann jedoch nach wie vor eine Herausforderung darstellen. Gerade in der heutigen Zeit gibt es einen sehr breit gefächerten Bedarf, Musik nach ihrem Inhalt zu kategorisieren. Um einen Einblick in diese Thematik zu schaffen, werden im folgenden Abschnitt zunächst der Ursprung und die Definition von Motiven erläutert. Anschließend werden diverse Einsatzgebiete aufgezeigt, in denen die Erkennung solcher signifikanter Tonfolgen essenziell ist.

2.2.1 Einführung: Das Motiv

Der Begriff des *Motivs* reiht sich neben dem *Thema*, der *Melodie* und der *Figur* in den Bereich der Formenlehre der Musik ein. Laut [10] wurden Begriffe der Formenlehre dabei häufig aus der Disziplin der Kompositionslehre des 19. Jahrhunderts übernommen. Von *Motiv* und *Thema* wurde bis zu diesem Zeitpunkt nur in Bezug auf Sprache und Rhetorik gesprochen. Erst mit Beginn der Einführung der Formenlehre wurden diese Begriffe zumindest dem Sinn nach auf die Musik umgelegt. Ganz allgemein beschäftigt sich die Formenlehre mit der Gestalt von Musik. Sie versucht theoretische Ableitungen und Regeln aufzustellen, die den Aufbau eines Musikstücks beschreiben können. Mit Begriffen wie "Bewegung" und "Gleichgewicht" werden die Kraft der Musik und ihre Zusammenhänge durch Wiederkehr und Kontrast beschrieben. Da die Formenlehre als Teilgebiet der Musiktheorie sehr alt ist, beschäftigen sich ihre Ansätze vor allem mit Musik des 18. und 19. Jahrhunderts. Die Grundsätze sind jedoch bis heute erhalten geblieben und lassen sich nach wie vor auch auf aktuelle Stücke anwenden.

In [11] wird das Motiv als "der kleinste musikalische Organismus" beschrieben. Motive stellen dabei laut [10] den Hauptgedanken bzw. die thematische Einheit eines Musikstücks dar. Allerdings sind Motive niemals etwas Statisches. Sie verändern sich im Fortschreiten eines Liedes und entwickeln sich stetig weiter. Als kleinste musikalische Einheit bestehen Motive meist nur aus wenigen Noten, tragen aber dennoch den eigentlichen Charakter des Stücks in sich. Durch die Summe von Motiven setzt sich das eigentliche *Thema* zusammen, welches wiederum durch die *Melodie* ausgedrückt wird.

Clemens Kühn unterscheidet in [12] zwischen den folgenden formgebenden Mitteln:

Wiederholung: Teile eines Musikstücks kehren unverändert immer wieder.

Variation: Teile kehren in veränderter Form wieder, sie sind einander dennoch ähnlich.

Verschiedenheit: Teile heben sich voneinander ab, ohne jedoch in starkem Kontrast zueinander zu stehen.

Kontrast: Teile stehen im Gegensatz zueinander. Hiermit wird eine starke Abhebung voneinander erzielt.

Beziehungslosigkeit: Teile entbehren jeglichen Zusammenhangs. Eine Beziehung zueinander kann nicht hergestellt werden.

Diese Kategorien beziehen sich auf die Gestaltungsmöglichkeiten von Musikstücken im Allgemeinen, lassen sich jedoch größtenteils auch auf die Form von Motiven anwenden. Vor allem die Wiederholung ist zwangsläufig eines der Grundprinzipien von Motiven. In deren Vorkommen lässt sich immer ein Basismotiv ausmachen, das sich im weiteren Verlauf einer Melodie wiederholt. Wiederkehrende Motive lassen sich dann sehr häufig in eine der anderen Kategorien einteilen. Meist heben sie sich durch eine Variation ihrer Form vom Basismotiv ab.

2.2.2 Verarbeitungsmöglichkeiten von Motiven

Die Weiterentwicklung eines Basismotivs kann durch diverse Verfahren vonstattengehen. Folgend wird eine Auswahl jener Ansätze der Musikanalyse dargestellt, welche in [10] erläutert werden und auch (aber nicht nur) auf einzelne Motive angewandt werden können. Zum besseren Verständnis werden zusätzlich praktische Beispiele angeführt. Außerdem ist anzumerken, dass die unterschiedlichen Formen, in welchen Motive auftreten, auch untereinander kombiniert werden können.

Diminution: Hier kann zwischen melodischer und rhythmischer Diminution unterschieden werden. Allgemein handelt es sich hier immer um eine Verkleinerung eines Parameters. Im Fall der rhythmischen Diminution erfolgt eine Verkürzung der Notendauern, die melodische Diminution entspricht einer Verkleinerung der Intervalle zwischen den Noten. Es wird also die Dauer oder die Tonhöhe verändert.

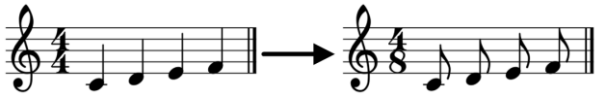

Abb. 2.12: Rhythmische Diminution

Augmentation: Die Augmentation ist das Gegenstück zur Diminution. Allerdings erfolgt hier keine Verkleinerung sondern eine Vergrößerung der Notendauern bzw. der Intervalle.

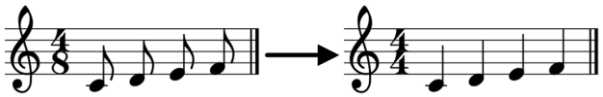

Abb. 2.13: Rhythmische Augmentation

Krebs: Hierbei handelt es sich um ein rückwärts verlaufendes Motiv im Vergleich zum Basismotiv.

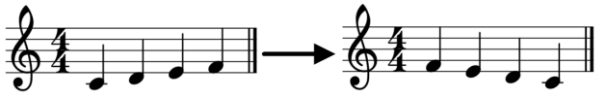

Abb. 2.14: Krebs

Spiegel: Wie der Name schon sagt, handelt es sich bei diesem Erscheinungsbild um die Spiegelung eines Basismotivs. Stellt man sich die Mitte der Notenlinien als Achse vor, werden die Noten des Basismotivs dabei um diese imaginäre Achse gespiegelt. Die so invertierten Noten können durch Intervallverengung oder -erweiterung zusätzlich variiert werden.

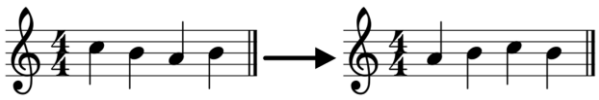

Abb. 2.15: Spiegel

Abspaltung: In dieser Darstellungsform erfolgt eine Verringerung der Notenanzahl des Basismotivs. Dabei ist eine theoretische Reduktion auf bis zu eine Note möglich.

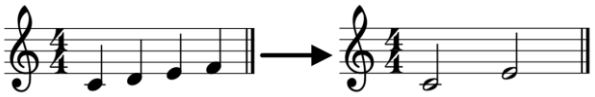

Abb. 2.16: Abspaltung und Augmentation

Erweiterung: Das Gegenstück der Abspaltung stellt die Erweiterung dar. In dieser Form der Weiterentwicklung erfolgt eine Streckung des Basismotivs durch das Hinzufügen von zusätzlichen Noten. Sowohl bei der Abspaltung als auch bei der Erweiterung muss natürlich der Takt gleichbleiben.

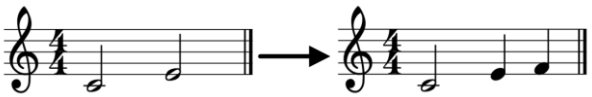

Abb. 2.17: Erweiterung

Sequenz: Die Sequenz entspricht einer einfachen Wiederholung des Basismotivs. Dieses kann dabei auch auf eine andere Tonstufe transponiert werden.

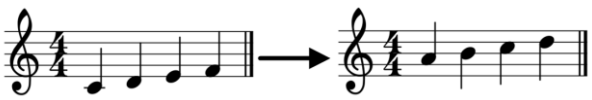

Abb. 2.18: Sequenz

Knaus und Scholz beschreiben in [10] noch weitere Verfahren, die jedoch kaum Relevanz für die Formgebung von Motiven besitzen. Die Erkennung von Motiven kann durch die Analyse eines Musikstücks mit Hilfe der beschriebenen Erscheinungsformen sehr gute Ergebnisse liefern.

2.2.3 Einsatzgebiete

Motiv-Erkennung reiht sich in die Disziplin des *Music Information Retrieval (MIR)* ein. In Zeiten, in denen digitale Kopien von Musikstücken immer mehr zunehmen, ist dieses Feld wichtiger denn je. Vor allem in Zusammenhang mit diversen Streaming-Diensten ist die Bedeutung von *MIR* klar zu erkennen. Musik wird in diesem Fall nicht mehr lokal am Rechner abgelegt, sondern online angehört. Die Möglichkeit, seine Musiksammlung selbst zu gruppieren oder zu sortieren, ist hier also nicht mehr gegeben. Stattdessen müssen die einzelnen Online-Portale sich um die Pflege ihrer riesigen Datenbestände kümmern. Dies ist eines der wesentlichsten Einsatzgebiete von *MIR*. Einige Aufgaben im Bereich von Musikdatenbanken sind:

1. Indizierung von Inhalten
2. Suche in Datenbanken
3. Gruppierung von Elementen

Beim Hinzufügen neuer Inhalte zu einer Musikdatenbank ist es notwendig, diese Musikstücke zuerst zu klassifizieren um sie danach korrekt zu indizieren. Dabei folgt eine komplette Analyse des entsprechenden Stücks, um eine musikalische Beschreibung von diesem anzufertigen. Anschließend erfolgt die Aufnahme des Liedes sowie der Beschreibung in die entsprechende Datenbank. Eine weitere essentielle Aufgabe stellt die inhaltsbasierte Suche in solchen Datenbanken dar. Ein sehr bekannter Vertreter dieser Sparte ist die Software *Shazam*, mit deren Hilfe man durch die Aufnahme kurzer Passagen eines Musikstücks innerhalb von Sekunden den Interpreten und den Titel erhält. Auch solche Systeme sind darauf angewiesen, dass die erwähnten musikalischen Beschreibungen von Stücken in deren Datenbanken vorliegen.

In [13] werden die Hauptkomponenten eines *MIR-Systems* durch die Extraktion der Beschreibung, den Vergleich mit bekannten Musik-Beschreibungen und schlussendlich das Auslesen des passenden Musikstücks beschrieben (siehe Abbildung 2.19).

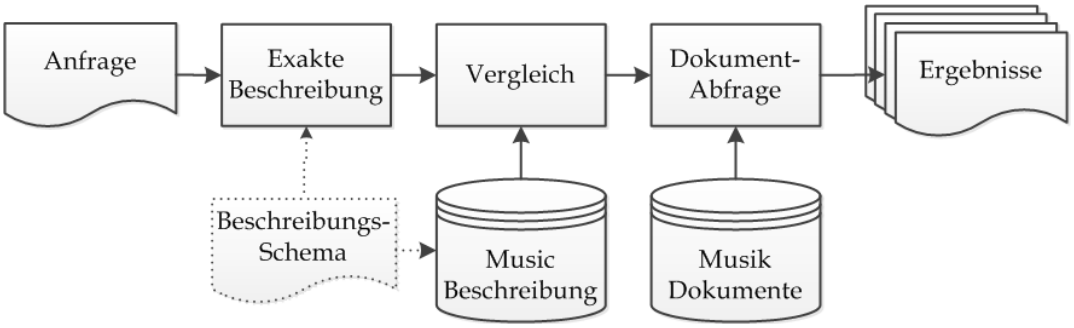

Abb. 2.19: Flussdiagramm inhaltsbasierte Musiksuche [13]

Ebenso ermöglichen *MIR-Systeme* eine Gruppierung von musikalischen Elementen. So ist beispielsweise eine Zuordnung zu passenden Genres und damit eine adäquate Sortierung von Inhalten möglich. Die hier genannten Einsatzgebiete können natürlich nicht ausschließlich durch die Erkennung und Extraktion von Motiven funktionieren. Motive stellen hier jedoch häufig auch einen wichtigen Teil einer musikalischen Beschreibung dar.

Aber nicht nur Musikdatenbanken für den Endanwender profitieren von solchen Systemen. Auch das Prüfen auf Plagiate wird durch den Einsatz von *MIR* ermöglicht. Ebenso kann es im Copyright-Management eingesetzt werden. Gerade in diesen Bereichen kann die Extraktion von Motiven sehr zielführend sein, da diese die prägnantesten Passagen von Musikstücken darstellen und somit gut für Vergleiche herangezogen werden können.

Schließlich werden Motive natürlich auch in der computergestützten Komposition eingesetzt. Vor allem im Bereich des Automated Composings werden solche einprägsamen Passagen von existierenden Liedern häufig verwendet, um die Ergebnisse in eine gewisse Richtung zu lenken. Speziell die Übertragung von Musikstil und -genre wird häufig auf diese Art und Weise durchgeführt.

2.3 Musical Instrument Digital Interfaces

MIDI stellt das zentrale Fundament dar, auf welchem der *PBMC* basiert. Die Verwendung dieses Formats ermöglicht aufgrund seiner Struktur eine einfachere Verarbeitung sowie Analyse von Musikstücken. Die Entwicklung geht bis in die 80er Jahre zurück, das Grundprinzip hat sich im Laufe der Zeit jedoch nicht verändert. Dieses Kapitel dient dazu, das Konzept von MIDI in groben Zügen zu umreißen. In den folgenden Abschnitten wird zunächst kurz auf die Entwicklungsgeschichte des Interfaces eingegangen. Abschließend werden der System-Aufbau und der eigentliche Kern, das Message System, erläutert.

2.3.1 Geschichtlicher Überblick

Wie in [14] angeführt ist, entstand *MIDI* zu Beginn der 1980er Jahre. Zu dieser Zeit wurden Computer großflächig verfügbar und digitale Synthesizer fielen stark im Preis. Diese Tatsachen ermöglichten es, Musik auf Keyboards einzuspielen, diese am Computer aufzuzeichnen und zu bearbeiten. Im Anschluss konnte man die so kreierten Stücke mit Hilfe des Synthesizers wiedergeben.

Die Einführung von *MIDI* wurde dabei von den verschiedenen Herstellern von Synthesizern angestrebt. Dazu gab es bereits 1981 erste Treffen, in denen über die Einführung eines Standard-Interfaces und dessen Spezifikation gesprochen wurde. Durch weitere Sitzungen der Arbeitsgruppe konnte bereits 1983 die *MIDI 1.0 Spezifikation* veröffentlicht werden. Zu dieser Zeit wurden erste Synthesizer hergestellt, die das neu definierte Protokoll unterstützten.

Durch *MIDI* wird jedoch keine Musik übertragen, sondern nur, wann welche Note gespielt wird. Es besitzt also nur einen beschreibenden Charakter. Aus diesem Grund ist ein Synthesizer notwendig, da dieser die Nachrichten in Töne umwandelt. Heutzutage verfügen Keyboards bereits über eingebaute Synthesizer, früher musste dafür ein externes Gerät angeschafft werden. Ebenso können Computer als Software-Synthesizer genutzt werden.

2.3.2 System-Aufbau

Die Verbindung von unterschiedlichen MIDI-kompatiblen Geräten erfolgt durch den Einsatz von speziellen Kabeln, welche allerdings nur unidirektional verwendbar sind. Jedes Gerät muss also zumindest einen In- und einen Out-Anschluss besitzen. In der Praxis existiert zusätzlich noch ein *Thru-Port*. MIDI-Kabel sind genau spezifiziert, wodurch es keine herstellereigenen Produkte mehr gibt. Um Geräte wie Keyboards an einen Computer anzuschließen, der über keine MIDI-Anschlüsse verfügt, existiert eine breite Masse an USB-MIDI-Wandlern. Die einfachsten dieser Geräte können per USB-Kabel mit dem Computer verbunden werden und besitzen ihrerseits einen MIDI In- und einen Out-Port. In Abbildung 2.20 ist der prinzipielle Aufbau eines MIDI-Setups mittels Computer und Keyboard ersichtlich.

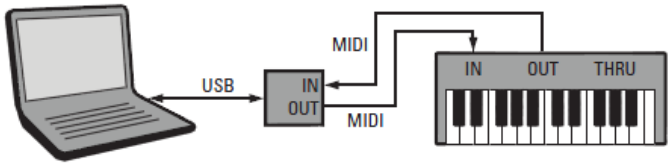

Abb. 2.20: MIDI Setup [14]

2.3.3 Message System

Das eigentliche Kernstück von *MIDI* stellt das Message System dar. Es definiert, wie Daten übertragen werden müssen, um von anderen MIDI-kompatiblen Geräten oder Programmen verstanden zu werden. Wie in der offiziellen Spezifikation in [15] zu erkennen ist, lassen sich MIDI-Messages zunächst in Channel- und System-Messages untergliedern. Unabhängig von der Kategorie, besteht eine MIDI-Message aus drei Bytes. Das erste Byte bestimmt dabei den Typ der Nachricht, das zweite und dritte Byte werden dazu genutzt, um Parameter anzugeben (siehe Abbildung 2.21). Der Typ enthält außerdem die Nummer des Kanals, der für die Ausgabe verwendet werden soll. Mit *MIDI* können 16 Kanäle parallel angesteuert werden.

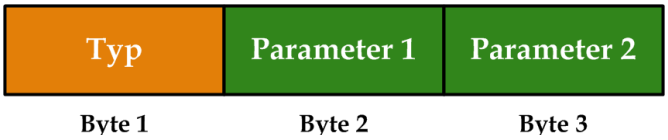

Abb. 2.21: MIDI-Message Aufbau

Channel-Messages kommen am häufigsten vor, da mit ihnen unter anderem angegeben wird, wann und wie lange eine Note gespielt werden soll. Dies geschieht mit den Nachrichten-Typen *NoteOn* und *NoteOff*, die die wesentlichsten Events zur Verarbeitung darstellen. Die beiden Messages enthalten als Parameter den Key sowie die Velocity. Ersteres gibt die Note selbst an, der zweite Wert bestimmt dabei den Anschlag der Taste und ist damit ein Maß für die Lautstärke des Tons. Ab dem Eintreffen eines NoteOn-Events wird der entsprechende Ton solange vom Synthesizer wiedergegeben bis der entsprechende NoteOff-Event eintrifft.

Im Gegensatz zu Channel-Messages sind System-Messages nicht an einen spezifischen MIDI-Kanal gebunden. Mit ihrer Hilfe können systemweite Parameter gesetzt oder Einstellungen geändert werden. Ein Beispiel stellt das Setzen des Tempos dar, welches mit System-Messages auf allen Geräten im MIDI-Setup einheitlich und gleichzeitig erfolgen kann. Diese Kategorie von Nachrichten ist jedoch für den Einsatz im *PBMC* irrelevant, da hier nicht mehrere Geräte verwendet werden.

Zur Aufnahme von Nachrichten werden Standard-*MIDI-Files (SMF)* verwendet. Das entsprechende Dateiformat ist ebenfalls in der MIDI-Spezifikation definiert. Im Wesentlichen werden darin sämtliche auftretenden Messages sequentiell in ihrer 3-Byte-Form abgelegt. Zusätzlich wird ein Timecode gespeichert, der in Millisekunden angibt, wann ein Event stattfindet. Anhand dieser Zeitangaben lassen sich die einzelnen Tondauern innerhalb eines Musikstücks auslesen. Diese Tatsache ist für die Anwendung im *PBMC* essentiell, da dieser ausschließlich mit *SMF* arbeitet. Durch den Einsatzzweck bedingt, wird hier keine Real-Time Übertragung von MIDI-Messages verwendet.

3 Stand der Technik

In diesem Kapitel soll der aktuelle Stand der Technik anhand verwandter Arbeiten und Projekte präsentiert werden. Dabei gliedert sich dieser Abschnitt in die Motiv-Erkennung sowie die computergestützte Komposition. Aufgrund der Tatsache, dass Assisted Composing in der kommerziellen Musikproduktion häufig eingesetzt wird, findet hier Software in diesem Bereich ebenso Erwähnung.

Grundsätzlich ist zu erwähnen, dass im Bereich der Motiv-Erkennung in Musikstücken verhältnismäßig wenige Arbeiten und Ansätze existieren. Im Gegensatz dazu ist das Thema der computergestützten Komposition bereits sehr gut exploriert, was sich auch im kommerziellen Einsatz wiederspiegelt.

3.1 Motiv-Erkennung und Extraktion

Wie bereits in Kapitel 2.2 erklärt, wird die Motiv-Erkennung sehr häufig in Ansätzen des *Music Information Retrieval* eingesetzt. Die hier vorgestellten Arbeiten beschränken sich dabei auf das konkrete Problem, Motive in Musikstücken zu erkennen. Die Resultate der angewandten Techniken können dann z.B. in der inhalts-basierten Musiksuche angewandt werden.

Verbeurgt et al. beschreiben in [16] einen ähnlichen Ansatz wie den des Pattern-Based MIDI Composings. Hier geht es jedoch eher darum, komplett neue Musik aus extrahierten Sequenzen zu kreieren, ohne erkennbare Parallelen zu berücksichtigen. Die vorgestellte Motiv-Erkennung basiert auf *Suffix-Trees*. Dabei wird die Baumstruktur durch eine Sequenz von Noten befüllt (siehe Abbildung 3.1). Jede mögliche Subsequenz wird in den *Suffix Tree* eingetragen. In jedem Blatt steht dabei der Startindex dieser Subsequenz. Bei mehreren möglichen Pfaden erfolgt eine Verzweigung. Findet eine solche an einem inneren Knoten nicht statt, werden die entsprechenden Elemente zu einem Knoten komprimiert. Diese Tatsache bildet den Unterschied zu *Suffix-Tries* [sic!]. Die einzelnen Knoten werden durch die Intervalle zwischen den

Noten dargestellt. Dieser Aufbau erlaubt das Finden von wiederholten Sequenzen, auch wenn diese nicht dieselben Startnoten beinhalten. Diminution und Augmentation von Motiven können erkannt werden. Kleine Intervalländerungen werden jedoch nicht berücksichtigt.

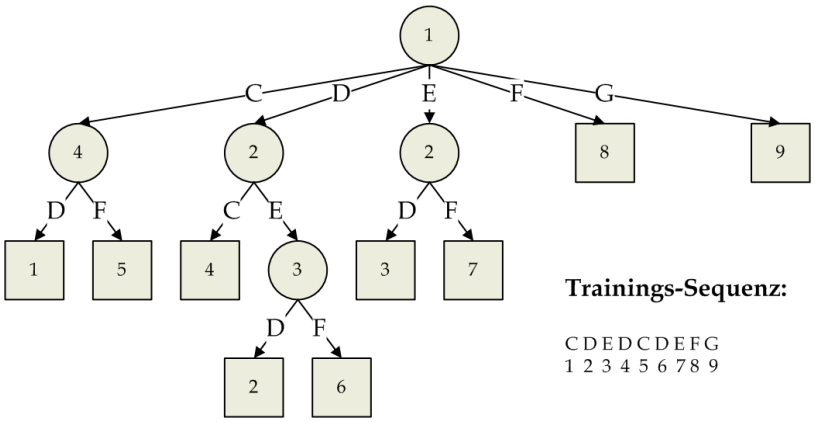

Abb. 3.1: Aufbau Suffix-Tree [16]

In [17] stellen Jiménez et al. einen Ansatz vor, um transponierte Motive in Melodien zu erkennen. Diese Arbeit basiert auf MusicXML [18], die Analyse erfolgt auf einer Sequenz von Noten. Diese stellt sich als geordnete Liste von Elementen dar, in welcher wiederum Subsequenzen gesucht werden. Der Ansatz von Jiménez et al. erlaubt, im Gegensatz zu klassischen sequenz-basierten Suchalgorithmen, auch das Finden ähnlicher Subsequenzen. Ähnliche Subsequenzen werden also ebenfalls als exakte Treffer gezählt. Der Algorithmus bestimmt dabei zunächst eine Menge von potentiell häufigen Subsequenzen. Anschließend wird gezählt wie oft diese tatsächlich vorkommen. Intervalländerungen werden nicht berücksichtigt.

Die Arbeit von Serrano und Inesta [19] beschäftigt sich mit Hanson-Intervallen (siehe Abbildung 3.2) als Grundlage der Motiv-Erkennung. Hanson Intervalle kodieren die Anzahl der Halbtonschritte mit Buchstaben. Zunächst werden Gruppen von jeweils vier kodierten Intervallen zusammengefasst und deren Häufigkeit in der Sequenz festgestellt. Durch einen iterativen Prozess gelangt man nun zu einem Motiv, das allerdings nicht genau so im ursprünglichen Material vorliegen muss. Das Ergebnis stellt eine prägnante Kombination aus den häufigsten Passagen dar.

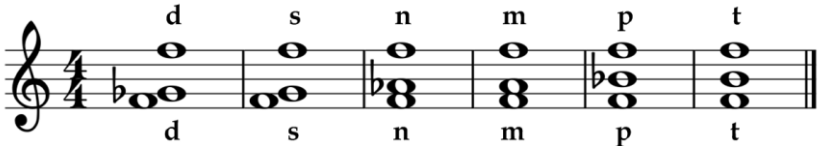

Abb. 3.2: Hanson-Intervalle

Die notwendigen Schritte um das Motiv zu generieren lauten wie folgt:

1. Sortiere die potentiellen Gruppen von Intervallen nach ihrer Häufigkeit.
2. Wähle eine Gruppe mit höchster Häufigkeit aus.
3. Verwende davon die letzten drei Intervalle.
4. Arbeite die sortierte Liste von oben nach unten durch. Bestimme in jeder Zeile das Element, dessen erste drei Intervalle den Intervallen aus Schritt drei entsprechen und erweitere die Sequenz aus Schritt 2.
5. Wiederhole die Schritte 3 bis 5 solange, bis die gefundene Intervallgruppe auf gleicher oder niedrigerer Ebene als die letzte gefundene Gruppe vorkommt.

Diese Schritte definieren nun jene Intervalle, welche am öftesten nach der am häufigsten auftretenden Initialgruppe vorkommen. Nachfolgend werden nun äquivalent dazu jene Intervalle gesucht, die am öftesten vor der Initialgruppe auftauchen. Die erwähnten Schritte sind in Abbildung 3.3 illustriert.

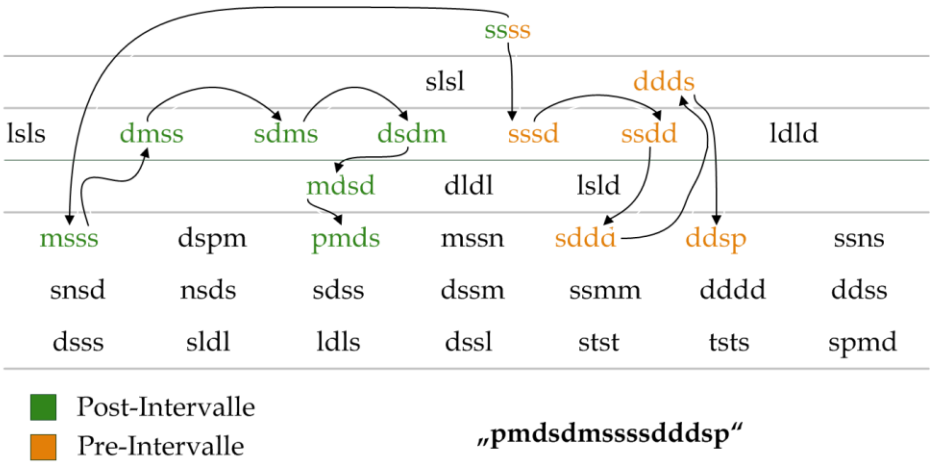

Abb. 3.3: Motiv-Generierung [19]

Die meisten Ansätze in der Motiv-Erkennung basieren auf dem Repetitionsprinzip von Notenfolgen in einer Melodie. Pinto hingegen stützt seine Arbeit [20] darauf,

dass sich Musik als ein Graph von kurzen Sequenzen darstellen lässt. Dabei berücksichtigt er sowohl rhythmische als auch melodische Strukturen. Die einzelnen Frames können kurze musikalische Passagen, beispielsweise einen Takt, beinhalten. Die Ähnlichkeit zwischen einzelnen Passagen wird durch eine Gewichtung der Kanten dargestellt, die diese Frames verbinden. Ob eine Sequenz mit einer anderen übereinstimmt, wird durch zwei Metriken $d_1$und d_2 festgelegt. Mit d_1 können Transpositionen in andere Oktaven erkannt werden, d_2 erlaubt das Finden von beliebig transponierten Tonfolgen.

Aus den genannten Arbeiten lässt sich erkennen, dass im Allgemeinen nur auf transponierte Motive eingegangen wird. Rhythmische Veränderungen wie die Diminution oder einzeln variierte Intervalle werden im Regelfall nicht erkannt.

3.2 Computergestützte Komposition

Im Bereich des algorithmischen Komponierens existieren sehr viele bekannte Arbeiten. In Kapitel 2 wurden bereits diverse Ansätze erläutert. In der Praxis kommen jedoch vornehmlich evolutionäre Ansätze im Allgemeinen und genetische Algorithmen im Speziellen zum Einsatz.

Einer der wenigen Ansätze, der sich nicht damit beschäftigt, stammt von Chen und Miikkulainen [21]. Sie arbeiten hier mit einem neuronalen Netzwerk, um Melodien zu generieren. In diesem Netzwerk wird zu einem Wert zum Zeitpunkt T ein Output zum Zeitpunkt T+1 erzeugt. Note für Note wird so das gesamte Stück aufgebaut. Die Tonhöhe jedes Outputs wird relativ durch einen Offset an Halbtonschritten festgelegt. Ähnlich verhält es sich mit der entsprechenden Notendauer.

Im Gegensatz dazu beschäftigt sich Bell in [22] im Wesentlichen mit Markov-Ketten um zu einer Komposition zu gelangen. Anders als andere Ansätze in diesem Bereich wird hier jedoch zusätzlich ein genetischer Algorithmus angewandt. Dieser wird jedoch nicht für die Komposition im eigentlichen Sinne verwendet. Die Gene der Chromosomen, welche jeweils eine Generation bilden, beinhalten in diesem Fall kei-

ne Noten sondern Markov-Ketten. Weiters erfolgt die Evaluierung der besten Generation nicht automatisch, sondern der Benutzer wählt aus, welche Komposition und somit welche Markov-Kette ihm subjektiv am besten gefällt. In [16] werden ebenfalls Markov-Ketten eingesetzt, um Melodien zu kreieren. Hier kommt es jedoch zu keiner Kombination mit anderen Verfahren. Allerdings werden die Übergangswahrscheinlichkeiten in diesem Ansatz nicht durch simples Analysieren von Trainingssets bestimmt, sondern mit Hilfe von Suffix-Trees festgelegt. Dieselbe Baumstruktur wird von Verbeurgt et al. auch zum Extrahieren von Motiven verwendet. Die konkrete Vorgehensweise kann in Kapitel 3.1 nachgeschlagen werden. In jedem Fall werden hier Noten-Übergänge als wahrscheinlicher angesehen, wenn diese in repetitiven Motiven vorkommen.

Ein klassischer genetischer Algorithmus zum Generieren von Musik, dessen Struktur dem üblichen Ansatz entspricht, wird in [23] vorgestellt. Das Ziel dieser Arbeit ist das Erzeugen von kurzen Melodien. Diese besitzen üblicherweise eine Länge von etwa vier 4/4 Takten. Hervorzuheben ist, dass Matić zum Kreieren einer neuen Generation nur das Mittel der Mutation verwendet, ein Crossover zweier Chromosomen wird nicht durchgeführt. Wie in diesem Ansatz der algorithmischen Komposition üblich, werden hier mehrere Fitness-Funktionen verwendet, deren gewichtete Ergebnisse in Summe die Qualität eines Chromosoms wiederspiegeln.

In [24] und [25] werden genetische Algorithmen vorgestellt, die dazu dienen improvisierte Melodien zu generieren, wie sie häufig im Jazz vorkommen. Im Gegensatz zur Arbeit von Matić erfolgt hier sehr wohl eine Verknüpfung zweier Chromosomen durch ein Crossover. Ebenso werden einzelne Chromosomen mit festgesetzten Wahrscheinlichkeiten mutiert. Die beiden Arbeiten gehen davon aus, dass so etwas Subjektives wie Musik nur durch Fitness-Funktionen evaluiert werden kann, die ebenfalls subjektive Kriterien verwenden. Es kommen also Bedingungen zum Einsatz, die auch von Improvisationskünstlern verwendet würden. Ein Beispiel hierfür wäre, dass sich die Tonhöhe von aufeinanderfolgenden Noten nicht zu sehr unterscheiden sollte. Ebenso sprechen sich Khalifa et al. in [26] dafür aus, formale Gram-

matiken in Kombination mit evolutionären Ansätzen zu verwenden. Ihre Arbeit besteht aus zwei Schritten. Zunächst werden 16 Motive mit einer maximalen Länge von 16 Noten durch einen genetischen Algorithmus bestimmt. Die Evaluierungsfunktionen wenden hier formale Grammatiken an, um die Güte der einzelnen Chromosomen zu bestimmen. Im zweiten Schritt werden die so erzeugten Motive schließlich zusammengesetzt, um eine Melodie zu generieren.

Eine weitere Arbeit, die sich mit dem algorithmischen Erzeugen von Jazz-Improvisationen auseinandersetzt, nennt sich GenJam [27]. Biles hat seine Arbeit daran bereits 1993 begonnen und seitdem stetig erweitert und verbessert. Im Wesentlichen geht es in diesem Projekt darum, dass die Software interaktiv auf musikalischen Input reagiert und dazu passende Melodien erzeugt. GenJam basiert ebenfalls auf einem genetischen Algorithmus. Wie in [28] dargestellt, werden immer wieder Demonstrationen der Software öffentlich gezeigt. Ein ähnliches Projekt wie GenJam wird in [29] von Weinberg et al. vorgestellt. Im Zuge dieser Arbeit wurde ein Roboter entwickelt, der auf Musik-Improvisationen von Menschen eingeht und passend dazu Melodien generiert. Der Roboter besteht dabei aus zwei Armen, mit welchen er ein Xylophon spielt. Ein genetischer Algorithmus bildet wiederum die Grundlage für die entsprechende algorithmische Komposition. Auch [30] beschäftigt sich mit genetischen Algorithmen. Dostál verwendet diesen Ansatz jedoch nicht um Melodien zu erzeugen, sondern um Rhythmusspuren zu kreieren. Die verwendeten Fitness-Funktionen sind dabei auf natürliche Musik-Kriterien zugeschnitten.

In den bisher dargestellten Arbeiten erfolgt die Evaluierung einzelner Chromosomen in den genetischen Algorithmen meist durch Anwendung von musiktheoretischen Gesetzen. Einen Gegensatz hierfür stellen [31] und [32] dar. Diese Arbeiten beschäftigen sich mit dem Kreieren von Fitness-Funktionen auf Basis des Zipf'schen Gesetzes [33]. Die Bestimmung der Güte einer Population erfolgt in diesem Fall also durch ein statistisches Verfahren.

3.2.1 Software

Abschließend sollen in diesem Abschnitt einige Softwareprodukte aus dem Bereich des algorithmischen Komponierens vorgestellt werden. Diese Auflistung untergliedert sich in Programmiersprachen, um Musik zu komponieren und in klassische Kompositions- und Notations-Software. Letztere beinhalten Funktionen die den Ansatz des Assisted Composings umsetzen.

Programmiersprachen: Programmiersprachen zur Erstellung von Kompositionen arbeiten vornehmlich mit grafischen Elementen, die dann in Musik umgesetzt werden. Häufig folgt der Aufbau simplen Schemata, in denen einzelne Komponenten mittels Drag and Drop in das aktuelle Projekt gezogen und miteinander kombiniert werden können. Es existieren einige kostenlose Programme, aber auch kommerziell vertriebene Software. Ein Beispiel für ein open-source Projekt stellt *OpenMusic (IRCAM Music Representation research group)* [34] dar. Als Gegenbeispiele dienen Programme wie der *Symbolic Composer (Tonality Systems)* [35] oder *Max (Cycling 74)* [36].

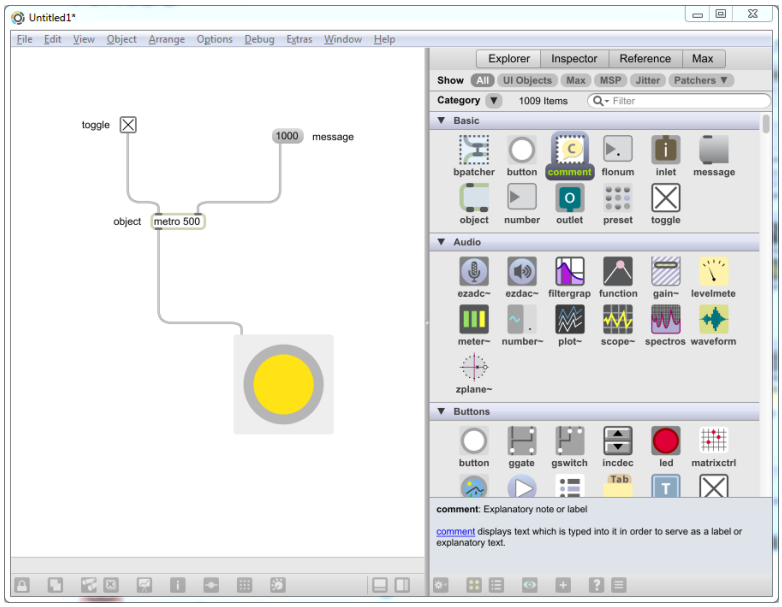

Abb. 3.4: Cycling - 74 Max

Kompositions- und Notations-Software: Diese Sparte an Software findet sehr häufig auch im professionellen Bereich Einsatz. Das Erstellen von Notenblättern gehört ebenso zum Repertoire wie die Live-Wiedergabe im Zuge des Kompositionsprozesses. Ansätze des Assisted Composings treten hier vornehmlich durch den Einsatz diverser Harmonisierungs-Funktionen auf. Mit diesem Hilfsmittel können Benutzereingaben, basierend auf der Harmonielehre oder formalen Grammatiken, harmonisch korrigiert werden. Zweck dieser Funktion ist die Unterstützung des Komponisten bei seiner Arbeit. Außerdem soll so auch ein Ideenanreiz gegeben werden. Als Vertreter dieser Art von Software können die Produkte *HarmonyBuilder (Musilogic)* [37], *Finale (MakeMusic)* [38] und *Liquid Notes (Re-Compose)* [39] genannt werden.

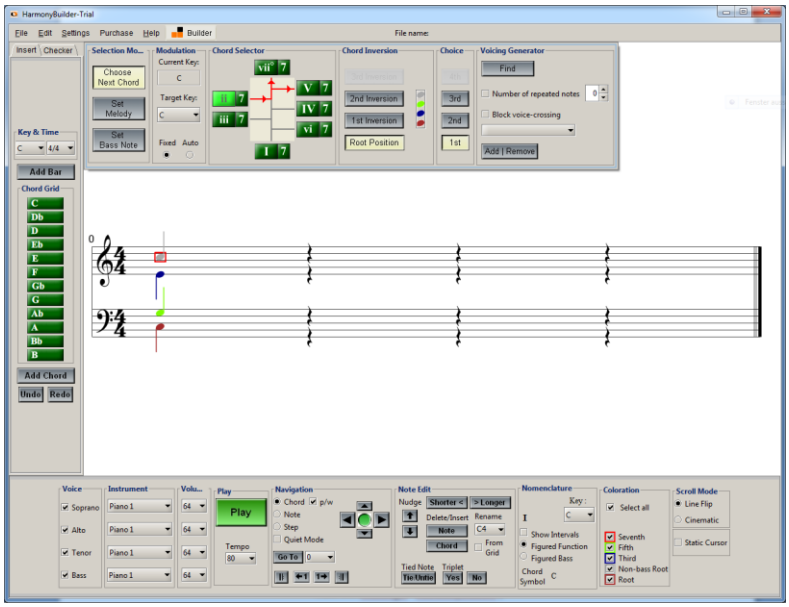

Abb. 3.5: Musilogic - HarmonyBuilder

In Kapitel 2 wurden die Grundlagen der Motiv-Erkennung und des algorithmischen Komponierens vermittelt. Weiters wurden in diesem Kapitel diverse praktische Arbeiten zu diesem Thema vorgestellt. In den folgenden Abschnitten wird nun die Planung und Implementierung des *PBMC* behandelt.

4 Entwurf

Nachdem in den letzten Kapiteln eine Einführung in die Thematik des algorithmischen Komponierens und der Motiv-Erkennung erfolgte, dient dieser Abschnitt der Erläuterung des Konzeptentwurfs des *PBMC*. Beginnend werden die Anforderungen an die Software dargestellt, um im weiteren Verlauf den Systemaufbau darzulegen. Im Anschluss werden sämtliche Komponenten und Schnittstellen näher beschrieben.

4.1 Anforderungsanalyse

Die Anforderungen an die Software, die im Zuge dieser Arbeit implementiert wurde, lassen sich zunächst grob in externe und interne Abläufe gliedern. Externe Anforderungen stellen jene Teile des Programms dar, welche auf den ersten Blick klar erkenntlich sind. Umgekehrt repräsentieren die internen Anforderungen jene, die dem strukturellen Aufbau der Software dienen. Da es sich beim *PBMC* um eine prototypische Implementierung handelt, muss weniger Wert auf die allgemeine Benutzerfreundlichkeit gelegt werden. Sämtliche funktionellen Anforderungen sind jedoch zu berücksichtigen.

Aus der Kombination aller externen und internen Software-Ziele lassen sich in weiterer Folge einzelne Module ableiten, welche im *PBMC* enthalten sein müssen. Die Summe dieser Module bildet schließlich den strukturellen Systemaufbau der Software, welcher im Abschnitt 4.2 erläutert wird.

4.1.1 Externe Anforderungen

Wie bereits erwähnt, stellen die externen Anforderungen jene dar, die nach außen hin klar ersichtlich sind. Der allgemeine Aufbau ist in Abbildung 4.1 illustriert.

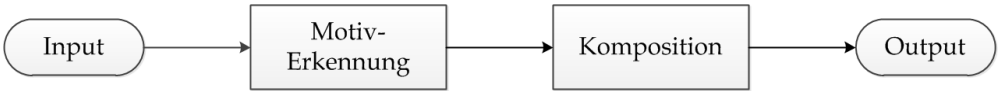

Abb. 4.1: Externe Software-Anforderungen

Das Flussdiagramm zeigt, dass diese Anforderungen sehr klar und einfach gestaltet sind. Folgend werden die einzelnen Elemente dieses Diagramms näher erläutert.

1. **Input:** Die Dateneingabe soll durch eine freie Auswahl von Musikstücken im MIDI-Format möglich sein.
2. **Motiv-Erkennung:** Ein ausgewähltes Musikstück soll analysiert werden, um darin enthaltene Motive erkennen zu können.
3. **Komposition:** Die Ergebnisse aus der Motiv-Erkennung sollen herangezogen werden, um ein neues Musikstück zu kreieren. Die Resultate sollen dabei klare Parallelen zu den verwendeten Motiven aufweisen. Weiters soll sich das komponierte Stück möglichst harmonisch und wohlklingend anhören.
4. **Output:** Wie schon beim Input angeführt, soll auch das resultierende Musikstück wieder im MIDI-Format vorliegen.

Die Punkte eins bis vier stellen die Anforderungen, die an den *PBMC* gestellt werden, sehr oberflächlich und nur in Grundzügen dar. Der sequentielle Aufbau spiegelt sich auch in der Praxis wieder. Jedes einzelne dieser Elemente bringt weitere Anforderungen hervor, die im folgenden Abschnitt erläutert werden.

4.1.2 Interne Anforderungen

Die internen Anforderungen an den *PBMC* ergeben sich aus den zuvor genannten externen Anforderungen. Im Wesentlichen stellen sie also eine detailliertere Variante dieser dar. Teilweise lassen sie sich nochmal in untergeordnete Aufgaben zerlegen (siehe Abbildung 4.2).

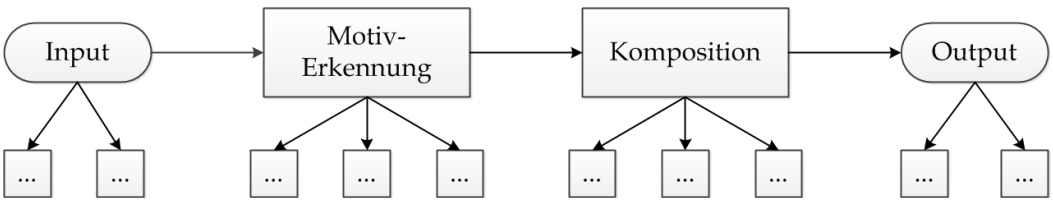

Abb. 4.2: Interne Software-Anforderungen

Folgend wird ein Überblick über die wichtigsten internen Anforderungen gegeben. Die einzelnen Ziele sind dabei entsprechend ihres Ursprungs gruppiert.

Input

- Verschiedene Typen des MIDI-Formats sollen eingelesen werden können.
- Eingelesene Musikstücke sollen in einer repräsentativen Art und Weise aufbereitet werden.

Motiv-Erkennung

- Es sollen möglichst viele unterschiedliche Varianten von Motiven erkannt werden. Zusätzlich zur üblichen Erkennung von transponierten Notenfolgen sollen auch variierte oder rhythmisch veränderte Sequenzen erkannt werden.
- Bereits detektierte Motive sollen für eine spätere Verwendung gespeichert werden können.
- Die Detektion soll zumindest auf monophone Melodien angewandt werden können.

Komposition

- Es soll zumindest ein Ansatz der algorithmischen Komposition implementiert werden.
- MIDI-Parameter wie Tempo oder Instrument sollen auch nach der Generierung einer Komposition noch verändert werden können.
- Zur kreierten Melodie sollen zusätzliche rhythmisch passende Tonspuren erzeugt werden.

Output

- Die resultierende Komposition soll innerhalb der Software wiedergegeben werden können.
- Ein optionales Speichern der Komposition soll möglich sein.

4.2 Systemaufbau

Aus der Summe der Software-Anforderungen lassen sich drei Module ableiten, die der *PBMC* beinhalten muss. Diese bilden den allgemeinen Systemaufbau (siehe Abbildung 4.3).

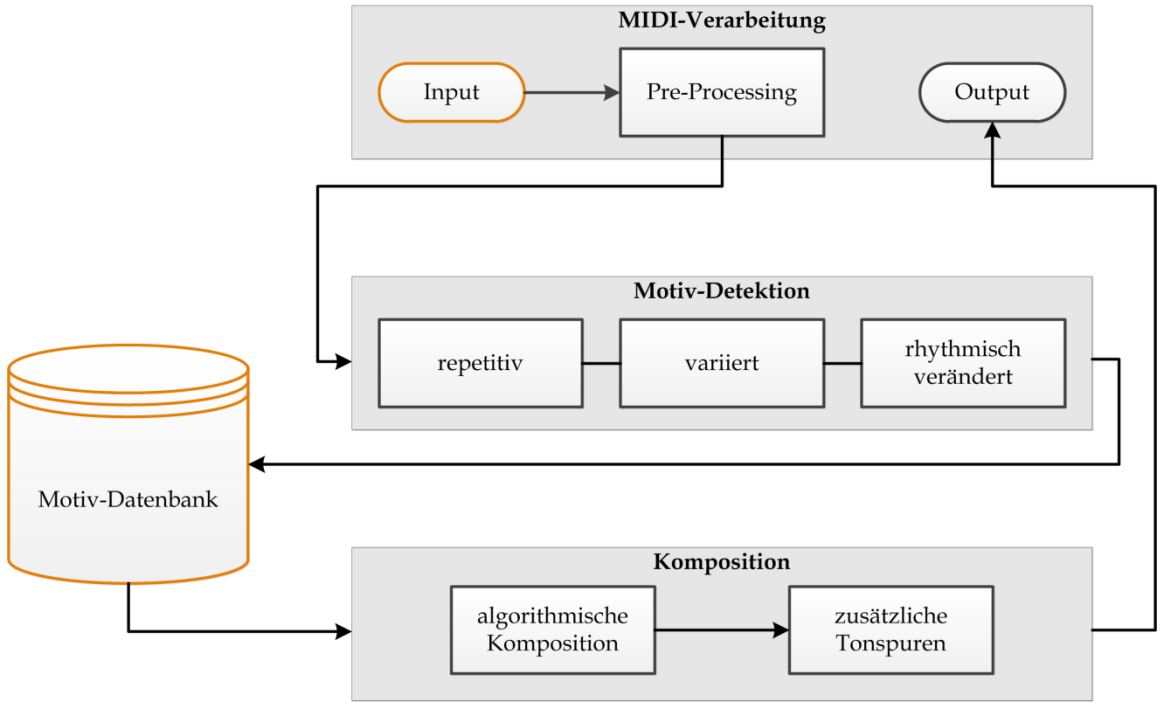

Abb. 4.3: Systemaufbau PBMC

Weitestgehend entsprechen die Module dem Aufbau der externen Anforderungen. Allerdings untergliedern sich diese Komponenten noch weiter. Zusätzlich wird eine Systemschicht hinzugefügt. Dabei handelt es sich um jenen Teil, welcher für die interne Datenstruktur verantwortlich ist, in der die eingelesenen MIDI-Stücke abgelegt werden. Diese Datenschicht wird im Wesentlichen an zwei Stellen im Ablauf gefüllt. Zum einen erfolgt eine Eingliederung der eingelesenen MIDI-Dateien in diese, zum anderen werden etwaig gespeicherte Motive bei deren Verwendung in dieser Datenstruktur abgelegt. Im Zuge der weiteren Verarbeitung innerhalb der Software wird einheitlich dieses Format verwendet. Eine genauere Erläuterung zur Struktur, in der die MIDI-Daten abgelegt und verarbeitet werden, findet sich im Abschnitt 4.3.

Grundsätzlich lässt sich sagen, dass jedes Modul für sich einer abgeschlossenen Einheit entspricht. Allerdings sind alle Komponenten voneinander abhängig. So dienen die Ergebnisse der MIDI-Verarbeitung als Input der Motiv-Detektion. Weiters bilden die erkannten Motive dieses Moduls den Input der Komposition. Wie in Abbildung 4.3 dargestellt, lässt sich der grundsätzliche Aufbau des PBMC durch die folgenden Module beschreiben.

MIDI-Verarbeitung (optional)

1. MIDI Ein-/Ausgabe
2. Pre-Processing

Motiv-Detektion (optional)

1. Repetitions-Detektor
2. Variations-Detektor
3. Rhythmik-Detektor

Komposition

1. Algorithmische Komposition
2. Erweiterung durch zusätzlich Tonspuren

Auf die einzelnen Komponenten wird in den folgenden Abschnitten dieses Kapitels noch näher eingegangen. An dieser Stelle ist jedoch anzumerken, dass die Module *MIDI-Verarbeitung* sowie *Motiv-Detektion* nicht zwangsläufig durchlaufen werden müssen. Der *PBMC* ist so konzipiert, dass das Kompositions-Modul sich aus bereits früher erkannten Motiven bedienen kann, welche in einer Datenbank abgelegt sind. Aus diesem Grund ist es nicht zwingend erforderlich, neue Motive zu extrahieren, wenn gewünschte Notensequenzen bereits vorliegen. Die möglichen Einstiegspunkte, um eine neue Komposition zu kreieren, sind in Abbildung 4.3 orange umrahmt.

4.3 Datenrepräsentation

Im Zuge der Erläuterungen zum Systemaufbau des *PBMC* wurde bereits eine Datenschicht erwähnt, in der Musikstücke intern abgelegt und weiterverarbeitet werden können. Um diese Struktur ausführlicher zu behandeln, ist es zunächst notwendig, die Anforderungen an diese zu definieren. Sie ergeben sich aus den Eigenschaften, die dieses Konstrukt aufweisen muss, welche wie folgt formuliert werden können:

1. Einfachheit
2. Vergleichbarkeit
3. Interoperabilität
4. Robustheit

Die ***Einfachheit*** der Datenstruktur setzt einen simplen und klaren Aufbau voraus. Zu verarbeitende Musikstücke müssen logisch abgelegt werden, um auf verschiedenste Weise eingesetzt werden zu können. Weiters müssen die Daten so abgelegt sein, dass sie als Ganzes oder Teile davon miteinander ***vergleichbar*** sind. Dieses Kriterium ist besonders wichtig um ein Funktionieren der Motiv-Detektion zu gewährleisten. Daten müssen außerdem so gespeichert werden, dass eine ***Interoperabilität*** gegeben ist. Die verwendete Struktur soll sowohl in der Motiv-Detektion als auch im Zuge der algorithmischen Komposition eingesetzt werden. Zusätzlich soll die Möglichkeit existieren, aus den abgelegten Informationen wiederum MIDI-Strukturen zu erzeugen. Die letzte Eigenschaft, die ***Robustheit***, ist dann gegeben, wenn sämtliche vorkommenden Datenkonstellationen in der Datenstruktur abgelegt werden können, ohne Änderungen daran vornehmen zu müssen. Diese Eigenschaft bildet eine wichtige Anforderung, welche in der MIDI-Eingabe essentiell ist. Musik kann viele Formen aufweisen. Alle Möglichkeiten sollen dabei ohne Schwierigkeiten eingelesen werden können. Nachdem nun die prinzipiellen Eigenschaften definiert sind, welche eine Struktur aufweisen muss, um darin Musik für die Verwendung im *PBMC* abzulegen, werden diese nachfolgend näher erläutert.

Einfachheit: Da die Struktur möglichst simpel gehalten werden soll, wird der grundsätzliche Aufbau eines MIDI-Musikstücks übernommen. Die einzelnen Teile davon sind in Tabelle 4.1 dargestellt.

Element	Definition
Score	Der Score repräsentiert das gesamte Musikstück mit sämtlichen Instrumenten, Tonspuren und in weiterer Folge den Noten.
Part	Ein Part steht immer für eine Tonspur, welche ein bestimmtes Instrument verwendet.
Phrase	Durch die Kombination mehrerer Phrasen innerhalb eines Parts kann eine polyphone Darstellungsweise erzielt werden. Pro Phrase kann immer nur eine Note gleichzeitig erklingen.
Note	Die Note stellt das atomare Element der Struktur dar. Pro Phrase kann eine beliebige Anzahl an Noten abgelegt sein. Eine Note wird zumindest durch Tonhöhe und Tondauer definiert.

Tabelle 4.1: Elemente der Musik-Datenstruktur

Wie in obiger Tabelle zu erkennen ist, hängen die Elemente voneinander ab. Ein Score ist also definiert durch eine Menge an Parts, ein Part besteht wiederum aus einer oder mehreren Phrases, welche dann aus n Notenelementen bestehen. Durch den klar strukturierten Aufbau ist die Eigenschaft der *Einfachheit* gegeben.

Vergleichbarkeit: Um Notenfolgen sinnvoll miteinander vergleichen zu können, kommt nur eine Gegenüberstellung der Intervalle zwischen einzelnen Noten in Frage. Anders ist keine Möglichkeit gegeben, z.B. transponierte Motive zu detektieren.

Im *PBMC* wird zu diesem Zweck eine Abwandlung der Hanson-Intervalle [40] verwendet, welche schon 1960 von Howard Hanson definiert wurden. Es handelt sich dabei um eine textuelle Darstellungsform von Notenintervallen. Diese werden per Definition mit den Buch-

staben *d-s-n-m-p-t* codiert. Jeder dieser Buchstaben entspricht dabei einem Halbtonschritt. Die ursprüngliche Kodierung wird nun durch diverse Modifikationen verändert um sie besser geeignet für die Motiv-Erkennung zu gestalten [41]. Zunächst wird die Darstellung mit den Vokalen *a-e-i-o-u* erweitert, um mehr als sechs Halbtonschritte definieren zu können (siehe Abbildung 4.4).

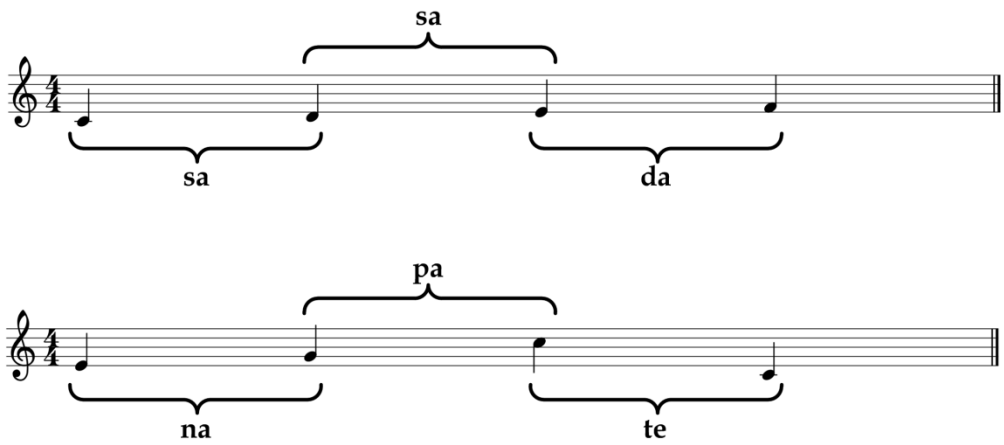

Abb. 4.4: Hanson-Intervalle mit Vokal-Erweiterung

Ein *a* an zweiter Stelle der Codierung gibt also ein Intervall von maximal sechs Schritten an, ein *e* bedeutet ein Intervall zwischen sieben und 12 Intervallen usw. Um einen Vergleich einfach zu gestalten, kann ebenso die Richtung, in welche die Noten verlaufen, angegeben werden. Deshalb erfolgt eine Codierung mit den Buchstaben *l* (Abstieg) und *r* (Anstieg) (siehe Abbildung 4.5).

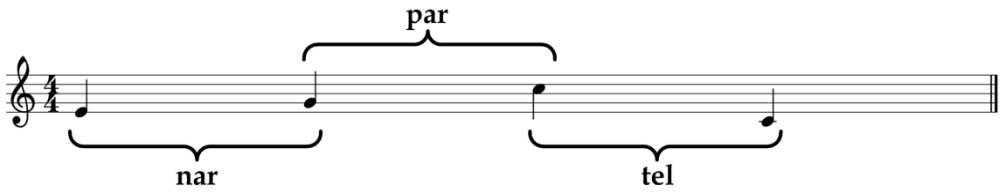

Abb. 4.5: Hanson-Intervalle mit Form-Erweiterung

Da die Tondauer von essentieller Wichtigkeit ist (siehe Einfachheit), muss eine Möglichkeit geschaffen werden, um auch diese in einem Hanson-String unterzubringen. Aus diesem Grund wird an dessen En-

de die Dauer durch die Summe der Tondauern beider das Intervall betreffender Noten angehängt. Dabei repräsentiert der Wert 1.0 eine Viertel. Eine Verdoppelung dieses Wertes wird als eine Verlängerung der Notendauer angesehen, eine Halbierung zeigt die Verkürzung dieser an (siehe Abbildung 4.6).

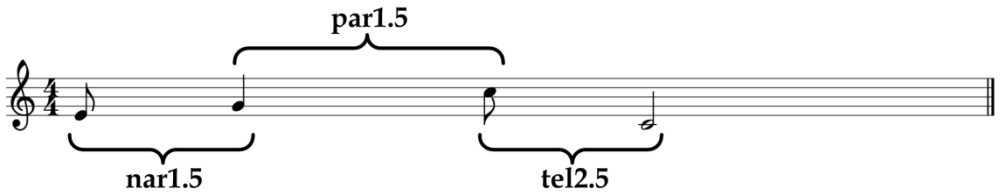

Abb. 4.6: Hanson-Intervalle mit Tondauer-Erweiterung

Weiters muss eine Darstellungsform existieren, die es erlaubt ein Intervall der Größe Null zu codieren. Eine solche Konstellation wird durch das Schlüsselwort *loo* repräsentiert. Wie im Standardfall wird auch hier die Intervalldauer durch die Kombination der Längen beider beteiligter Noten ans Ende des Strings gesetzt (siehe Abbildung 4.7).

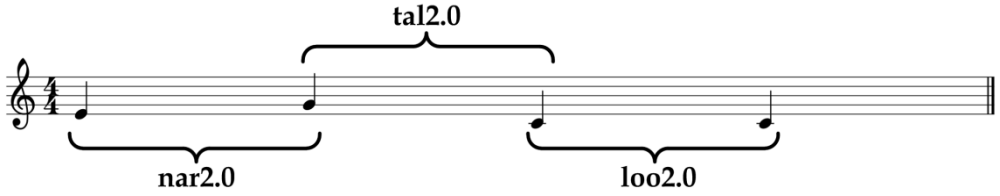

Abb. 4.7: Hanson-Intervall mit Intervallgröße Null

Interoperabilität: Durch das gemeinsame Speichern von essentiellen Noten-Parametern (Tonhöhe, Tondauer) und der entsprechenden Hanson-Codierung wird ein umfassendes Maß an Interoperabilität gewährleistet. Beide wesentlichen Module des *PBMC* können mit den abgelegten Daten bedient werden. Während die Motiv-Detektion die entsprechenden Intervalle heranzieht, um Übereinstimmungen zu finden, verwendet das Kompositions-Modul die tatsächlichen Noten-Informationen, um neue Melodien zu kreieren. Ebenso ist eine einfache Konvertierung der beschriebenen Datenstruktur in das MIDI-Format möglich, weil die Struktur aus diesem abgeleitet ist.

Robustheit: Die Tatsache, dass sich eine Übertragung eines Musikstücks in die Datenstruktur als sehr einfach darstellt, zeichnet deren Robustheit aus. Die Aufteilung in unterschiedliche Parts und Phrases innerhalb einer MIDI-Datei ist jeweils vom Ersteller abhängig. Durch die Ableitung der Basis-Struktur aus dem MIDI-Format kann es beim Befüllen der Struktur jedoch zu keinerlei Problemen kommen. Weiters ist das Konvertieren der Noten-Parameter in Hanson-Intervalle ein sehr robustes Verfahren, da durch die Vokal-Erweiterung 35 anstatt sieben Standard-Intervallen möglich sind. So ist im Praxisfall eine Überschreitung der maximal zulässigen Notenabstände ausgeschlossen.

Wie in den Software-Anforderungen erwähnt, soll es auch möglich sein, erkannte Motive für eine spätere Verwendung speichern zu können. Im Systemaufbau wurde bereits gezeigt, dass der PBMC zwei mögliche Einstiegspunkte aufweist, um eine Komposition anzustoßen. Hier soll nun die zweite Variante, die Motiv-Datenbank, näher erläutert werden. Um keine Inkonsistenz zwischen beiden Möglichkeiten zu schaffen, sollen gefundene Motive immer diese Datenbank passieren. Aus diesem Grund werden die entsprechenden Notenfolgen auch hier in der erläuterten Form abgelegt. Um ein einfaches und schnelles Wiederfinden von gespeicherten Motiven zu ermöglichen, ist auch eine Untergliederung in verschiedene, benennbare Kategorien angedacht.

Grundsätzlich soll die Möglichkeit geschaffen werden, gefundene Motive sowohl online als auch lokal zu speichern. Aus diesem Grund soll ein einfaches *relationales Datenbank-Management-System* eingesetzt werden, um die gefundenen Notenfolgen abzulegen. Um auch auf eine mögliche globale Datenbank für mehrere Benutzer vorbereitet zu sein, soll außerdem eine Anmeldung durch Benutzername und Passwort möglich sein. Anhand des nachfolgend dargestellten Datenbankmodells wird der Aufbau der Tabellen veranschaulicht.

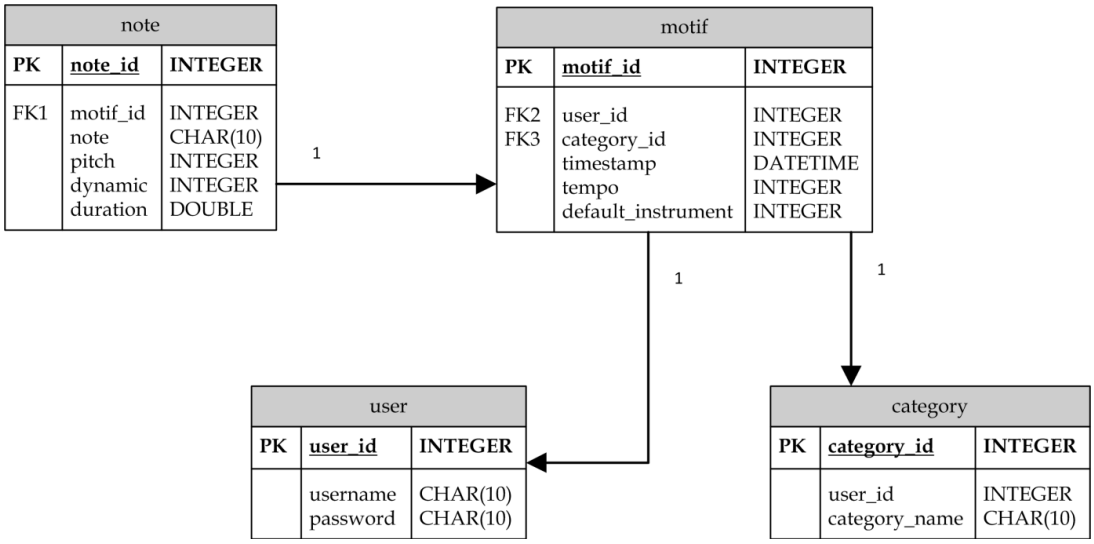

Abb. 4.8: Datenbankmodell Motiv-Datenbank

4.4 MIDI-Verarbeitung

Wie bereits im Zuge der Erläuterungen zum Systemaufbau erwähnt, hat dieses Modul im Wesentlichen die Aufgabe, sich um den In- und Output zu kümmern. Weiters werden notwendige Pre-Processing Schritte in diesem Teil des *PBMC* durchgeführt.

In Anlehnung an die System-Anforderungen stellt das Öffnen, Wiedergeben und Speichern von Musikstücken keine gravierenden Hindernisse dar. Diese Aufgaben werden durch die Auswahl eines adäquaten MIDI-Frameworks einfach ermöglicht. Das verwendete Framework wird im Kapitel *Implementierung* näher erläutert.

Die zentrale Aufgabe der MIDI-Verarbeitung stellt jedoch das Pre-Processing dar, welches zwingend erforderlich ist, um eingelesene Musikstücke weiterverarbeiten zu können. In diesem Schritt ist zunächst eine Analyse der Melodie erforderlich, um sie in das interne Datenformat (siehe Kapitel 4.3) zu übertragen. Weiters müssen die Daten entsprechend normalisiert werden, um nachfolgende Arbeiten durchführen zu können.

Analyse: Die Analyse des MIDI-Stücks erfolgt immer pro Phrase. Entsprechend der Software-Anforderungen werden die Daten nur monophon betrachtet, abgelegt und verglichen. Dies ist mit ein Grund, warum die

allgemeine Datenstruktur bereits in einzelne Phrasen gegliedert wird. In diesem Schritt werden die Noten in der Struktur sukzessive abgearbeitet. In jeder Iteration werden die relevanten Parameter einer Note ausgelesen und in die Datenstruktur übernommen. Zusätzlich wird anhand zweier aufeinanderfolgender Noten das entsprechende Hanson-Intervall ermittelt und ebenfalls im Noten-Objekt abgelegt. Nachdem die interne Datenstruktur auf diese Weise aufgebaut wurde, können die darin enthaltenen Daten auf die weitere Verarbeitung vorbereitet werden. Dieser Normalisierungsprozess wird im folgenden Punkt theoretisch erörtert. Eine detailliertere Erläuterung zur Umsetzung findet sich im Kapitel *Implementierung*.

Normalisierung: Während sich verändernde oder gleichbleibende Noten-Intervalle eine sehr gute Basis für das Auffinden von repetitiven Motiven darstellen, ergeben sich aus vorhandenen Pausen größere Schwierigkeiten im Zuge der Intervall-Kodierung. In der verwendeten Datenrepräsentation erfolgt die Kalkulation der Notenlänge anhand der Zeitstempel der MIDI-Datei. Dies begründet sich aus der Tatsache, dass in MIDI keine diskrete Angabe von Notenlängen vorgesehen ist. Existieren im eingelesenen Musikstück jedoch Pausen zwischen zwei aufeinanderfolgenden Noten, so werden diese in der Datenstruktur als "stille" Noten abgelegt. Da jedoch eine Intervallbestimmung zwischen einer tatsächlich vorkommenden Note und einer darauffolgenden Pause nicht durchführbar ist, stellt sie die Hanson-Codierung in diesen Sonderfällen mit dem Schlüsselwort *Rest* dar. Existieren nun sehr viele Pausen innerhalb eines Musikstücks, ist keine Differenzierung der Intervalle mehr möglich. Ein Vergleich wird so unmöglich gemacht.

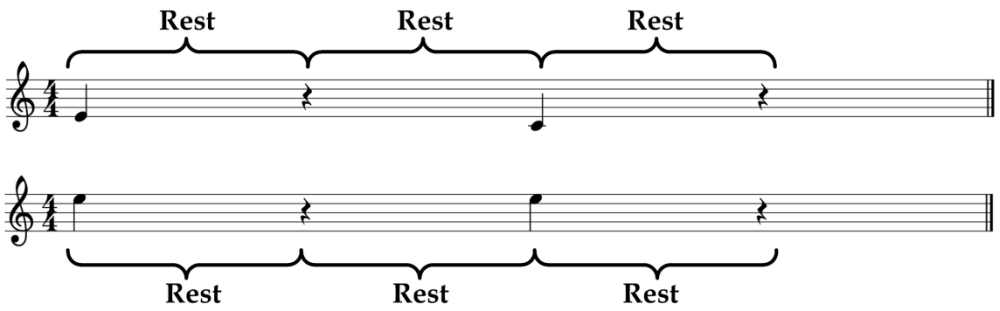

Abb. 4.9: Hanson-Intervalle mit Pausen

In Abbildung 4.9 sind zwei Notenfolgen zu je vier Vierteln dargestellt, welche beide Pausen beinhalten. Die vorhandenen Noten unterscheiden sich dabei drastisch in ihren Intervallen. Durch die vorhandenen Pausen erfolgt jedoch in beiden Varianten und sämtlichen vorkommenden Intervallen eine Kodierung mittels des Schlüsselwortes *Rest*. Ein Vergleich beider Notenfolgen würde also eine falsch-positive Übereinstimmung ergeben.

Um diesem Problem entgegenzuwirken, sollen im Zuge des Normalisierungs-Prozesses sämtliche vorkommenden Pausen entfernt werden, wenn deren Länge unterhalb eines festgelegten Schwellwertes liegt. Da bei simplem Löschen von Pausen jedoch der korrekte Takt des Musikstücks zerstört würde, wird die entsprechende Pausenlänge zur vorherigen Notendauer addiert. Mit dieser Technik lassen sich Pausen unter Beibehaltung des Taktes eliminieren. Im Anschluss können die Intervalle wieder regulär miteinander verglichen werden. Ein falsch-positiver Vergleich ist somit ausgeschlossen (siehe Abbildung 4.10).

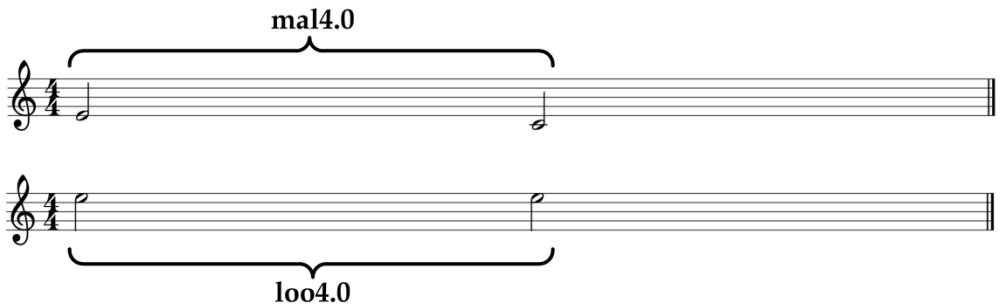

Abb. 4.10: Normalisierte Hanson-Intervalle

4.5 Motiv-Detektion

Neben dem Kompositions-Modul stellt die Motiv-Detektion einen wesentlichen Bestandteil des *PBMC* dar. Sie bildet den Kern, auf dem die gesamte Arbeit aufbaut. Grundsätzlich lässt sich sagen, dass die erzielten Ergebnisse der Software umso besser sind, je aussagekräftiger sich die Resultate der Motiv-Erkennung darstellen. Aus der Anforderungsanalyse geht hervor, dass der eigentliche Zweck dieser Arbeit darin besteht, Kompositionen zu kreieren, welche deutlich erkennbare Parallelen zu ausgewählten Musikstücken aufweisen. Arbeitet also dieses Modul nicht hinreichend gut, werden die Ergebnisse des Kompositions-Moduls das gesteckte Ziel nicht erreichen.

Die meisten Arbeiten im Bereich der Motiv-Erkennung beschränken sich auf das Suchen und Finden von sich wiederholenden, transponierten Notenfolgen (siehe Kapitel 3.1). Im Gegensatz dazu soll hier der Ansatz verfolgt werden, eine Fülle von möglichen Motiv-Übereinstimmungen zu finden. Dazu zählen nicht nur repetitive Noten-Muster, sondern auch variierte oder rhythmisch veränderte Motive. Um dieses Ziel zu erreichen, ist es zunächst notwendig, die möglichen Motiv-Strukturen zu kennen. Die gängigsten Verarbeitungsmethoden wurden bereits im Kapitel 2.2.2 vorgestellt. Diese stellen gleichzeitig die Basis der Motiv-Detektion im *PBMC* dar. Ein automatisierter Prozess, wie der hier beschriebene, wird natürlich nie in der Lage sein sämtliche möglichen Kombinationen von Motiv-Strukturen zu detektieren. Kann aber zumindest das Auftreten der genannten Motiv-Formen erkannt werden, ist bereits ein breites Spektrum an möglichen Variationen abgedeckt.

Die Grundlage dieses Moduls bilden die im Pre-Processing generierten Hanson-Intervalle. Da diese keinen Bezug zur eigentlichen Tonhöhe haben, sondern nur die Intervalle zwischen einzelnen Notenpaaren beschreiben, kann anhand dieser Darstellungsform ein einfacher Vergleich zwischen Notenfolgen durchgeführt werden. Das Konstrukt der Motiv-Erkennung gliedert sich im *PBMC* in die folgenden Komponenten.

- Basis-Detektor
- Repetitions-Detektor
- Variations-Detektor
- Rhythmik-Detektor

Der Basis-Detektor dient der Aufbereitung von potentiellen Motiven. Die übrigen drei Komponenten führen die eigentlichen Vergleiche der Notenfolgen durch. Nachfolgend werden die unterschiedlichen Detektoren näher erläutert.

Basis-Detektor: Der Basis-Detektor liefert die Grundlage der Vergleiche. Er bedient die drei spezifischen Detektoren mit Notenfolgen, deren eigentlicher Vergleich erst dort stattfindet. In dieser Komponente wird zunächst die interne Datenstruktur einer einzelnen Phrase Element für Element abgearbeitet, um alle möglichen Kombinationen aus zusammenhängenden Notenfolgen zu bestimmen, die dort vorkommen. Jeweils zwei der potentiellen Motive werden im nächsten Schritt an die weiteren Detektoren weitergereicht, wo diese miteinander verglichen werden. Dabei stellt immer eine der beiden Notenfolgen das Basismotiv dar, welchem die andere gegenübergestellt wird. Jedes der vermerkten potentiellen Motive wird im Zuge der Vergleichszyklen also einmalig als Basismotiv verwendet.

Repetitions-Detektor: Wie auch die beiden anderen spezifischen Detektoren führt diese Komponente konkrete Motiv-Vergleiche durch. Der Repetitions-Detektor beschränkt sich dabei auf das Finden von repetitiven Motiven. Wiederholte Tonfolgen können dabei transponiert sein oder sich in derselben Tonlage befinden.

Variations-Detektor: Diese Komponente ist darauf spezialisiert, intervallvariierte Motive zu erkennen. Solche Notenfolgen können als Spiegel oder Krebs (siehe Kapitel 2.2.2) auftreten. Außerdem wird eine Motiv-

Übereinstimmung auch im Falle geringfügig abweichender Intervalle festgestellt.

Rhythmik-Detektor: Im Gegensatz zu den beiden anderen spezifischen Detektoren erlaubt dieser das Auffinden von Motiven, welche rhythmisch verändert auftreten. Konkret werden hier Augmentation und Diminution, sowie erweiterte oder verkürzte Notenfolgen erkannt. In letzterer Variante bleibt die Gesamtlänge des erkannten Motivs unverändert, die Anzahl der Noten darin kann jedoch variieren.

4.6 Komposition

In diesem abschließenden Abschnitt soll nun näher auf den Entwurf des Kompositionsmoduls eingegangen werden. Dieses stellt den finalen Schritt dar, um zu einem generierten Musikstück zu gelangen. Das primäre Ziel dieser Komponente besteht darin, eine Melodie zu erzeugen, welche in ihren Grundzügen eine deutliche Ähnlichkeit zu zuvor ausgewählten Kompositionen aufweist (siehe Kapitel 4.1). Die Grundlage, um dieses Ziel zu erreichen, wurde bereits durch die Motiv-Extraktion geschaffen. Die Aufgabe dieser Software-Komponente ist es nun, die zuvor isolierten Motive in die Komposition einer neuen Melodie einfließen zu lassen.

Bezugnehmend auf die Anforderungsanalyse soll zumindest ein Ansatz der algorithmischen Komposition angewandt werden. Um jedoch eine Vergleichsbasis zu schaffen, wurden im Zuge des Entwurfs des Softwarekonzeptes zwei unterschiedliche Verfahren angedacht. Konkret handelt es sich dabei um ein stochastisches und ein evolutionäres Verfahren. Als Vertreter der ersten Variante werden im *PBMC* Markov-Ketten eingesetzt, ein genetischer Algorithmus stellt das Gegenstück dazu dar. In den folgenden beiden Abschnitten wird auf den Aufbau beider implementierter Ansätze eingegangen.

Eine weitere Anforderung an das Kompositionsmodul des *PBMC* besagt, dass generierte Melodien durch weitere, rhythmisch passende Tonspuren ergänzt werden sol-

len um eine vollwertige Komposition zu erstellen. Zu diesem Zweck wurden drei zusätzliche Phrasen angedacht, welche optional aktivierbar sind. Jeder dieser Zusatz-Phrasen kann ein eigenes Instrument zugeordnet werden, wobei die dritte Phrase eine reine Rhythmusspur darstellt. Die Struktur dieser zusätzlichen Tonspuren wird grundsätzlich durch eine Analyse der zuvor generierten Melodie festgelegt. Die erste Phrase (Counterphrase) übernimmt sämtliche Noten der Komposition, variiert dabei jedoch die Tondauern. So kann beispielsweise eine Viertel zu zwei Achteln geändert werden. Ebenso ist das Zusammenfassen mehrerer Noten zu einem längeren Ton möglich. Wie die Noten variiert werden, ist dabei zufallsbestimmt. Zusätzlich erfolgt noch eine Transposition um eine Oktave nach oben. Die zweite Phrase (Answerphrase) übernimmt ganz einfach die Struktur der komponierten Melodie und transponiert diese eine Oktave nach unten. Die dritte und letzte optionale Phrase (Rhythmus) ist völlig unbeeinflusst von der aktuellen Komposition. Hier erfolgt eine schlichte Beat-Ausgabe im 4/4 Takt. Die Form der ersten beiden Zusatzphrasen leitet sich aus der Entwicklungsform der Fuge aus der Musiktheorie ab [11].

4.6.1 Markov-Ketten

Markov-Ketten stellen eine sehr einfache, aber wirkungsvolle Methode dar, um eine computergenerierte Komposition zu kreieren. Die prinzipielle Funktionsweise dieses Ansatzes wurde bereits im Kapitel 2.1.3 ausführlich erläutert. In diesem Abschnitt sollen nun die Unterschiede zur dort angeführten Grundform beschrieben werden.

Die allgemein übliche Vorgehensweise beim Einsatz von Markov-Ketten stützt sich auf die Festlegung von Regeln, welche die Transitions-Wahrscheinlichkeiten bestimmen. Das heißt, dass sich der Anwender eines solchen Verfahrens zunächst darüber im Klaren sein muss, welche Art von Musik erzeugt werden soll. Aufgrund dieser Auswahl müssen dann passende Grammatiken erdacht werden, welche die Notenabfolge in die beabsichtigte Richtung lenken.

Im Gegensatz dazu soll der *PBMC* jedoch keine völlig unbeeinflussten Kompositionen hervorbringen, sondern die zuvor extrahierten Motive in seine Berechnungen einfließen lassen. Aus diesem Grund ist hier auch kein Regelwerk vonnöten, um Übergangswahrscheinlichkeiten zu bestimmen. Diese werden ausschließlich durch die Analyse von ausgewählten Motiven festgelegt. Als Basis dieser Analyse werden zunächst alle ausgewählten Motive in einer Phrase aneinander gereiht (siehe Abbildung 4.11).

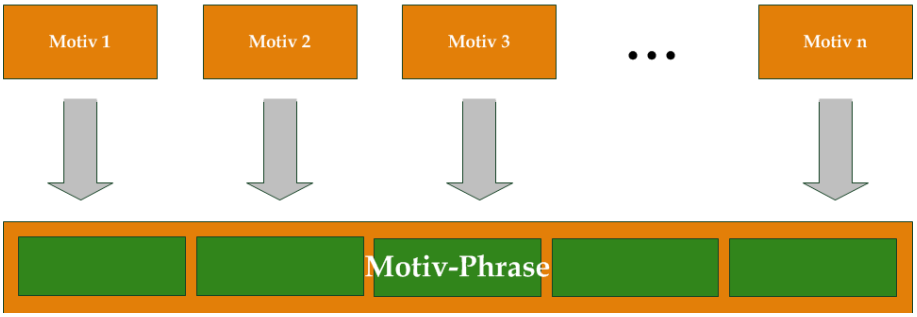

Abb. 4.11: Aufbau Motiv-Phrase

Ausgehend von den dort vorkommenden Noten werden im Anschluss die möglichen Ereignisse bestimmt (Tonhöhe, Tondauer, Dynamik). Die entsprechenden Übergänge innerhalb der Motiv-Phrase werden dann gezählt und entsprechend die Übergangswahrscheinlichkeiten festgelegt. Dieses Verfahren ersetzt also die andernfalls notwendigen Transitions-Regeln.

Die übrige Vorgehensweise entspricht wiederum der Standardanwendung. Ein beliebiges Ereignis wird als Startwert herangezogen, aufgrund der Übergangswahrscheinlichkeiten werden dann die Folgeereignisse ausgewählt. Häufig vorkommende Transitionen werden also sehr wahrscheinlich auch in der generierten Komposition auftreten. Zudem kann die entsprechende Markov-Matrix sehr schnell berechnet werden, da keine Regeln abgearbeitet werden müssen. Die Analyse kürzerer Musikstücke nimmt kaum Zeit in Anspruch.

4.6.2 Genetischer Algorithmus

Wie die Markov-Ketten wurde auch der Ansatz der genetischen Algorithmen ausführlich in Kapitel 2.1.3 erläutert. Hier sollen wiederum nur die spezifischen Änderungen und Besonderheiten beschrieben werden, welche für den Einsatz im *PBMC* notwendig sind. Der grundsätzliche Aufbau des Algorithmus entspricht dabei der üblichen Herangehensweise. Die gravierendsten Abweichungen betreffen den Inhalt der Grundpopulation. Folgend werden sämtliche Komponenten der Implementierung beschrieben. Das wesentlichste Kriterium für einen gut funktionierenden Algorithmus stellt dabei die Fitness-Funktion dar.

Grundpopulation

Üblicherweise wird die Grundpopulation in einem genetischen Algorithmus durch zufällig aufgebaute Chromosomen festgelegt. Im Laufe von n Iterationen werden diese modifiziert um ein möglichst optimales Ergebnis zu erzielen.

Da eine wesentliche Anforderung an die Software (siehe Kapitel 4.1) jedoch besagt, dass deutliche Parallelen zu ausgewählten Input-Motiven bestehen sollen, erfolgt die Zusammensetzung der Grundpopulation im *PBMC* anders. Wie schon beim Einsatz von Markov-Ketten angewandt, werden auch hier sämtliche ausgewählte Motive in zufälliger Reihenfolge aneinander gereiht. Entsprechend der Länge der einzelnen Chromosomen kann es hier durchaus auch zu mehrfachem Vorkommen einzelner Notenfolgen kommen. Weiters wird eine Vor-Harmonisierung bereits bei der Generierung der Grundpopulation vorgenommen, indem die einzelnen Motive auf dieselbe Tonart transponiert werden (siehe Abbildung 4.12). Der genetische Algorithmus im *PBMC* arbeitet hierbei auf Basis der C-Dur. Diese Vorgehensweise vereinfacht die Bewertung durch die Fitness-Funktion und trägt dazu bei, dass bereits nach relativ wenigen Iterationen ein möglichst harmonisches Ergebnis erzielt werden kann.

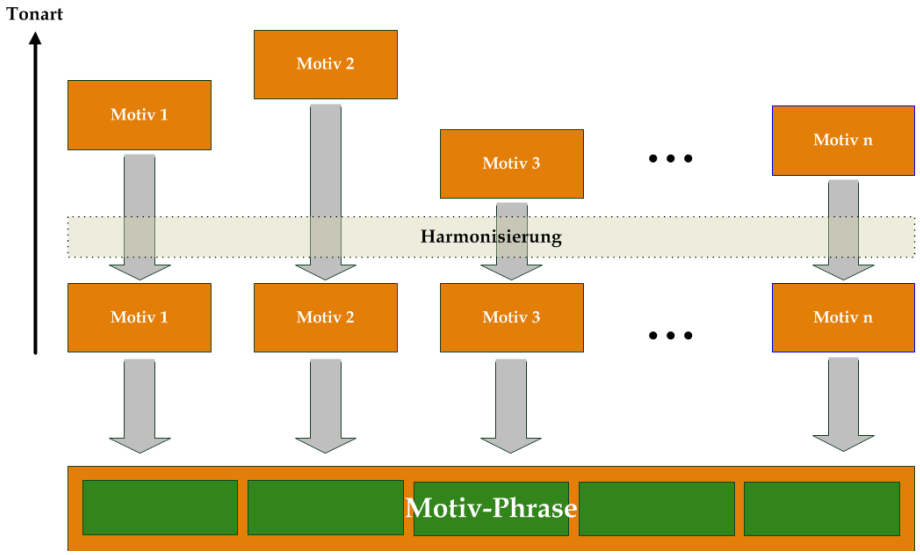

Abb. 4.12: Motiv-Phrase und Harmonisierung

Fitness Funktion

Nachdem die Grundpopulation angelegt wurde, erfolgt im nächsten Schritt die Bewertung der einzelnen Chromosomen. Dies stellt den wichtigsten Schritt dar, um zu einem möglichst optimalen Resultat zu gelangen. Je besser die Fitness-Funktion ausgewählt wurde, umso besser hört sich auch das Ergebnis des Algorithmus an.

Musik basiert in ihren Grundzügen auf einigen konkreten Regeln. Diese lassen sich jedoch nicht durch eine einzige Fitness-Funktion ausdrücken. Aus diesem Grund erfolgt hier die Evaluierung der Chromosomen durch die Berechnung eines gewichteten Mittels mehrerer Funktionen. In diesem genetischen Algorithmus werden dazu neun Funktionen verwendet. Stellt $F(x)$ den gemittelten Fitnesswert für ein Chromosom x dar und sei $f_n(x)$ entsprechend eine der neun Fitness-Funktionen, so ergibt sich die Evaluierung der Güte eines Chromosoms durch die Formel $F(x) = (\sum_1^9 f_n(x))/9$. Die erwähnten neun Funktionen werden folgend näher beschrieben.

Letzte Note: Allgemein lässt sich sagen, dass das Ende einer Melodie harmonisch ausklingt, wenn die letzte Note eine längere Tondauer aufweist. Aus diesem Grund liefert diese Fitness-Funktion einen höheren

Wert zurück, wenn diese Note wenigstens einer Viertel oder mehr entspricht.

C-Dur: Wie in den Erläuterungen zur Grundpopulation erwähnt, erfolgt bereits beim Zusammenstellen dieser eine Transposition aller Motive in C-Dur. Durch später vorkommende Mutationen in den Chromosomen kann dieser Sachverhalt jedoch wieder verändert werden. Im Zuge der Untersuchung eines Chromosoms durch diese Funktion werden solche Ausreißer erkannt. Das zurückgelieferte Resultat ist dabei höher, je mehr Noten sich in C-Dur befinden.

Noten-Zusammenhänge: In dieser Funktion werden die Intervalle zwischen aufeinanderfolgenden Noten ermittelt. Je mehr davon zwischen ein und drei Halbtonschritten liegen, desto besser wird ein Chromosom bewertet. Häufig gleichbleibende Tonhöhen wirken langweilig und schneiden deshalb schlechter in der Beurteilung ab. Im Gegensatz dazu ist ein Notenunterschied von vier oder mehr Schritten zu sprunghaft, weshalb auch dieser Fall weniger gut bewertet wird.

Melodie-Intervall: Vor Durchführung des genetischen Algorithmus wird festgelegt, in welchen Oktaven sich die resultierende Melodie bewegen soll. In dieser Funktion werden Chromosomen umso besser bewertet, je mehr Noten davon innerhalb der festgelegten Grenzen liegen.

Takt: Der hier eingesetzte Algorithmus geht immer von einem 4/4 Takt für die generierte Melodie aus. Hier wird evaluiert, wie exakt dieser im entsprechenden Chromosom eingehalten wird. Je mehr Ausreißer gefunden werden, umso schlechter schneidet das Chromosom in der Bewertung ab.

Kontur: Die Kontur ist ein wichtiges Ausdrucksmittel in der Musik. Diese Fitness-Funktion zieht zur Beurteilung jeweils vier aufeinanderfolgende

Noten heran. In diesen Notenfolgen wird untersucht ob die Form gleichbleibend, schwingend, absteigend oder aufsteigend ist. Je mehr Notenfolgen zu letzteren beiden Formen zählen, umso besser schneidet das entsprechende Chromosom ab.

Anfangs-/Endnote: Eine Melodie, vor allem wenn diese eher kurz ist, klingt harmonischer, wenn deren erste und letzte Note dieselbe Tonhöhe aufweisen. Dieser Fall wird hier durch einen höheren Fitnesswert belohnt.

Große Notenintervalle: Die Intervalle zwischen aufeinanderfolgenden Noten wurden bereits in einer anderen Funktion evaluiert. Da durch die Zusammensetzung der unterschiedlichen Motive aber auch sehr große Sprünge von mehr als vier Halbtonschritten auftreten können, erfolgt eine zweite Fitness-Auswertung für besonders große Intervalle. Je mehr von diesen über vier Schritte aufweisen, desto weiter wird der Fitnesswert nach unten gedrückt.

Tondauer: In dieser Bewertungsfunktion geht es darum, drastische Wechsel von aufeinanderfolgenden Tondauern zu untersuchen. Je mehr bzw. je größere Wechsel in der Tondauer auftreten, umso schlechter wird ein Chromosom bewertet.

Selektion

Zur Selektion der jetzt bewerteten Chromosomen wird im genetischen Algorithmus des *PBMC* die sogenannte Tournament-Selektion durchgeführt. Im Vergleich zur sonst häufig angewandten Auswahl der besten Chromosomen, ist dieses Verfahren stärker zufallsabhängig. Dabei werden n noch nicht verwendete Chromosomen der Population beliebig ausgewählt und davon das am besten bewertete selektiert. Dieser Vorgang wird so lange wiederholt, bis die selektierte Menge groß genug ist um eine neue Generation hervorzubringen.

Crossover

In diesem Schritt erfolgt eine Kombination aus den zuvor selektierten Chromosomen. Dieser Algorithmus verwendet dazu ausschließlich ein One-Point-Crossover. Das Verfahren wird genau so angewandt, wie es im Grundlagenkapitel 2.1.3 erläutert wurde. Die erste Hälfte eines Elternteils wird also mit der zweiten Hälfte des zweiten Elternteils zusammengesetzt, woraus ein neues Kind-Chromosom resultiert. An welcher Stelle die Teilung der Elternchromosomen vorgenommen wird, wird zufällig bestimmt.

Mutation

Eine Mutations-Funktion erlaubt es einem genetischen Algorithmus Gene zufällig zu verändern. So wird nicht nur Erbgut von existenten Chromosomen verwendet, um neue Generationen hervorzubringen, sondern es kann auch zu einer spontanen Erbgutveränderung kommen. Wie auch in der Natur, treten Mutationen jedoch nicht in jedem Chromosom jeder Generation auf, sondern nur zu einer gewissen Wahrscheinlichkeit. Diese ist im *PBMC* per Parameter einstellbar. Das heißt, dass ein Chromosom nach dem Crossover mutiert werden kann, aber nicht muss.

Der Aufbau des Mutations-Algorithmus erfolgt in diesem speziellen Fall immer auf einem Teilfragment des Chromosoms. Dazu wird in einem ersten Schritt ein zufälliges Stück der Melodie ausgewählt. Mit Hilfe von neun möglichen Mutationen werden einzelne oder mehrere Noten dieses Fragments anschließend zufällig verändert. Die Mutationsmöglichkeiten lauten wie folgt.

1. Zufällige Transposition des Fragments
2. Zufälliges Vertauschen der Tonhöhen
3. Zufälliges Verändern von mehreren Tonhöhen
4. Zufälliges Verändern von genau einer Tonhöhe
5. Zufälliges Verändern von Tonhöhe und Tondauer von genau einer Note
6. Aufsteigendes Sortieren nach Tonhöhen

7. Absteigendes Sortieren nach Tonhöhen
8. Invertieren des Fragments
9. Zufälliges Vertauschen der Tondauern

Im Falle einer Mutation wird eine diese Möglichkeiten zufällig ausgewählt. In Summe können jedoch bis zu vier zufällige Fragmente ausgewählt werden, auf denen Mutationen durchgeführt werden. Im Anschluss an das Verändern dieses Teilstücks der Melodie erfolgt entweder ein Kopieren dieses Fragments an eine andere beliebige Position im Chromosom oder ein Vertauschen mit einem beliebigen anderen Fragment gleicher Länge.

5 Implementierung

Nachdem im letzten Kapitel vorwiegend die theoretische Struktur des *PBMC* definiert und erläutert wurde, werden in diesem Abschnitt relevante Details zur Implementierung aufgegriffen. Diese werden zum Teil anhand von Pseudocode-Ausschnitten erklärt. Die entwickelte Software wurde in Java programmiert. Aus diesem Grund ist der *PBMC* plattformunabhängig.

5.1 MIDI-Framework

Für die Verwendung mit Java existieren einige MIDI-Frameworks, welche die Ein- und Ausgabe von MIDI-Dateien ermöglichen. Die Verwendung eines adäquaten Frameworks bietet außerdem die Möglichkeit, die Bearbeitung von eingelesenen Musikstücken zu vereinfachen. Viele musikrelevante Funktionen, wie etwa die Transposition, sind üblicherweise bereits implementiert, was den eigenen Entwicklungsaufwand deutlich verringert.

Zwei der umfangreichsten und bekanntesten Java-Bibliotheken zur Verarbeitung von MIDI stellen *JFuge* [42] und *JMusic* [43] dar. Im *PBMC* kommt zweitere zum Einsatz. Dies hat zwei Gründe. Zum einen werden die Noten-Parameter in diesem Framework als numerische Werte abgelegt, zum anderen besteht die Struktur bereits im Wesentlichen aus den Komponenten, welche auch in der hier entwickelten Software angedacht sind (siehe Kapitel 4.3). Im Gegensatz dazu verwendet *JFuge* String-Pattern um Musik zu kreieren. Diese Tatsache würde Vergleiche zwischen unterschiedlichen Fragmenten schwieriger gestalten und sich negativ auf die Geschwindigkeit auswirken. Weiters wäre eine Konvertierung zur Datenstruktur des *PBMC* um ein Vielfaches aufwändiger.

5.1.1 Verwendung

Das eingesetzte Framework wird in erster Linie im MIDI-Verarbeitungs-Modul eingesetzt. Hier übernimmt es das Einlesen und Speichern von Dateien sowie die Wiedergabe von Kompositionen innerhalb der Software. Eine Übertragung der *JMusic*-eigenen Datenstruktur in die des *PBMC* ist aufgrund des ähnlichen Aufbaus sehr einfach.

Über die erwähnten Aufgaben hinaus werden Funktionen der Bibliothek auch in beiden Ansätzen des Kompositions-Moduls angewandt. Innerhalb des genetischen Algorithmus werden Modifikations-Funktionen von *JMusic* benutzt, um gewisse Mutationen auf einzelnen Chromosom-Fragmenten durchzuführen. Das Framework stellt hier eine breite Palette an Hilfsmitteln zur Verfügung, um beispielsweise eine Phrase zu transponieren, zu invertieren oder die Reihenfolge der enthaltenen Noten zufällig zu verändern. Umgekehrt basiert der komplette stochastische Ansatz auf *JMusic*. Die MIDI-Bibliothek beinhaltet bereits eine Implementierung zur Komposition mit Hilfe von Markov-Ketten. Hier musste nur noch der Input für den Algorithmus, wie in Kapitel 4.6.1 beschrieben, kreiert werden.

5.1.2 Probleme

Trotz aller Vorteile, die die Verwendung von *JMusic* mit sich bringt, ergaben sich im Verlauf der Implementierung größere Schwierigkeiten damit. Diese beziehen sich auf das korrekte Einlesen von MIDI-Dateien, weshalb gewisse Modifikationen am Open-Source Framework durchgeführt werden mussten.

Da MIDI per Definition keine diskreten Notendauern im Dateiformat anbietet, müssen diese bei der Verarbeitung durch eine Analyse der Zeitstempel einzelner Events berechnet werden. Mit Hilfe einer Subtraktion des Timecodes eines NoteOff-Events von dem eines NoteOn-Events kann also die Dauer einer Note in Millisekunden berechnet werden. Nachdem Musik jedoch niemals auf Zeiteinheiten basiert, sondern in Beats angegeben wird, muss an dieser Stelle eine Umrechnung erfolgen. Wie auch

andere Frameworks verwendet *JMusic* hierzu die PPQN-Clock (Pulses per Quarter Note). Der Wert des PPQN gibt an, wie lange eine Viertelnote in Millisekunden dauert. Finden sich innerhalb einer MIDI-Datei absolut korrekte Zeitstempel, ist die Umrechnung problemlos durch eine einfache Division möglich. Allerdings kommt dies eher selten vor, da MIDI oft von Keyboards aufgezeichnet wird. Dadurch wird die Notendauer manuell durch die Länge des Tastendrucks bestimmt, wodurch es zu Zeit-Diskrepanzen kommen kann. Finden sich aufgrund dieser Tatsache abweichende Timestamps in der MIDI-Datei wieder, kommt es zu Fehlinterpretationen von Notenlängen. Bei der Wiedergabe der Originaldatei fallen solche minimalen Zeitabweichungen kaum auf. Diese werden jedoch deutlich verstärkt, sobald die Datei eingelesen wurde, da es bei der Umrechnung von Millisekunden auf tatsächliche Notendauern zu Rundungsfehlern kommt. Beim Abspielen von solchen falschen Notendefinitionen werden die Fehler deutlich hörbar und wirken sehr störend. Außerdem schließen aus diesem Grund Noten oft nicht direkt aneinander an, obwohl sie so angedacht sind. Es entstehen ungewollte Pausen in der internen Datenstruktur.

Um dem Problem entgegenzuwirken, wurden jene Teile von *JMusic* umgeschrieben, welche im *PBMC* zum Einsatz kommen. Durch kleinere Modifikationen legt das Framework die eingelesenen Noten nicht mehr mit seiner internen, umgerechneten Tondauer ab, sondern mit der konkreten Zeitdauer in Millisekunden. Das ursprüngliche Timing bleibt also bestehen. Wo nötig erfolgt ein gezieltes Mapping von Zeit auf Musiknotation. Diese genannten Änderungen beheben die Timing-Probleme bei der Wiedergabe und beim Speichern in Dateien. Ungewollte Pausen können dadurch jedoch nicht gänzlich vermieden werden. Aus diesem Grund ist eine Normalisierung der Daten notwendig, wie im Kapitel 4.4 beschrieben.

5.2 Datenrepräsentation

Die Hanson-Intervalle, welche bereits vor der Motiv-Detektion ermittelt werden, werden als Strings abgelegt. Die Form entspricht der Darstellung in Kapitel 4.3. Anfangs wurden Vergleiche zwischen einzelnen Motiven auf Basis solcher Strings

durchgeführt. Um diesen Prozess zu optimieren, wurde im weiteren Verlauf ein rein numerisches Format festgelegt. Textuell abgelegte Kodierungen können dabei sehr einfach in diese Form gebracht werden, und umgekehrt.

In der numerischen Hanson-Kodierung werden sämtliche Buchstaben aus der ursprünglichen Formatierung als Ziffern dargestellt. So befindet sich an der Tausenderstelle eine Zahl zwischen 0 und 5, die Hunderterstelle beinhaltet einen Wert zwischen 0 und 4 und die Zehnerstelle wird durch 0 oder 1 dargestellt. Durch Addieren der Notendauer des Hanson-Strings wird die Darstellung komplettiert (siehe Abbildung 4.13).

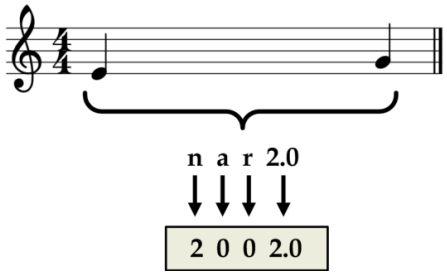

Abb. 5.1: Numerische Hanson-Kodierung

Durch diese umgewandelte Form der Repräsentation lassen sich nun Gegenüberstellungen einzelner Intervalle mittels Double-Vergleichen durchführen. Auf diese Weise ist eine höhere Effizienz als bei der Verwendung von Strings gewährleistet.

5.3 Modulbeschreibung

In den folgenden beiden Abschnitten werden die Module, welche für die Motiv-Extraktion bzw. die Komposition zuständig sind, unter dem Aspekt der konkreten Implementierung beleuchtet. Im Zuge dessen werden auch hier Probleme aufgezeigt, die während der Umsetzung aufgetreten sind.

5.3.1 Motiv-Extraktion

Wie bereits im Kapitel 4.5 erläutert wurde, besteht das Konstrukt der Motiv-Erkennung im *PBMC* aus vier wesentlichen Komponenten. Die eigentlichen Verglei-

che zwischen zwei Notenfolgen werden von den drei spezifischen Detektoren erledigt. Die Aufbereitung der Melodie und das Finden von potentiellen Motiven erfolgt im Basis-Detektor. Da es sich bei der entwickelten Software nur um eine prototypische Umsetzung handelt, wurde hier wenig Wert auf Geschwindigkeit gelegt. Die Suche nach zusammenpassenden Notenfolgen wird intern nach dem Brute-Force-Schema gehandhabt. Es werden also sämtliche mögliche Notenfolgen, die sich innerhalb von definierten Maximallängen bewegen, miteinander verglichen. Diesem Ansatz kommt die Nutzung von numerischen Intervall-Kodierungen entgegen, da eine Vielzahl an Vergleichen angestellt wird. Im folgend dargestellten Algorithmus wird der prinzipielle Aufbau des Basisdetektors veranschaulicht.

```
size = MIN_MOTIF_LENGTH
while size <= MAX_MOTIF_LENGTH
{
      for i = 0; i < numberOfIntervals; i++
      {
            add potentialMotif (interval i ... size)
      }
}

while potentialMotifs left to match
{
      for i = 0; i < numberOfIntervals; i++
      {
            match(potentialMotif, interval i ... motifSize)
      }
}
```

Algorithmus 5.1: Aufbau Basis-Detektor

Die spezifischen Detektoren arbeiten mit einem simplen Vergleich der einzelnen Intervalle der potentiell zusammenpassenden Motive. Algorithmus 5.2 zeigt den Aufbau anhand des Vergleichs eines repetitiven Motivs.

```
for i = 0; i < motifSize; i++
{
      if baseIntervals[i] != potentialMatchIntervals[i]
            return false //motifs not equal
}
return true //motifs equal
```

Algorithmus 5.2: Aufbau Repetitions-Detektor

Der Vorteil dieses Verfahrens ist, dass der Algorithmus sehr einfach und klar ist. Alle Motive, welche den Kriterien der spezifischen Detektoren entsprechen, werden im Musikstück bzw. in einer Phrase davon gefunden. Allerdings bringt diese Herangehensweise auch Nachteile mit sich. Einerseits ist natürlich offensichtlich, dass die Suche in sehr langen Musikstücken auch entsprechend mehr Zeit kostet. Das ergibt sich ganz einfach aus dem ineffizienten Suchalgorithmus. Andererseits ist jedoch auch die Überschneidung von gefundenen Motiven möglich. In einer Folge aus mehreren identen Noten können mehrere Motive mit unterschiedlicher Länge gefunden werden. Dieser Fall wird nicht abgefangen. Vielmehr wird dem Benutzer des *PBMC* am Ende des Suchalgorithmus eine Liste der gefundenen Motive präsentiert. Aus dieser ist dann manuell auszuwählen, welche der Notenfolgen in die Motiv-Datenbank übernommen werden sollen. Dieser Vorgang erlaubt es, trotz möglicherweise sehr vieler Suchergebnisse, eine selektierte Auswahl vorzunehmen. Einem doppelten Vorkommen von potentiellen Basismotiven wird entgegengewirkt, indem beim Erstellen der Liste untersucht wird, ob ein identes Motiv bereits enthalten ist. Dadurch werden nur Notenfolgen aufgenommen, die noch nicht als potentielles Motiv vorliegen.

5.3.2 Komposition

Dieser Abschnitt beschäftigt sich im Wesentlichen mit dem umgesetzten genetischen Algorithmus. Der Ansatz zur Komposition mit Markov-Ketten ist im verwendeten MIDI-Framework *JMusic* implementiert und soll deshalb hier nicht näher erläutert werden. Der Aufbau dieses Ansatzes folgt den in Kapitel 4.6.1 beschriebenen Strukturen. Im folgenden Algorithmus wird der prinzipielle Ablauf dargestellt.

```
for pitch, duration, dynamic
{
      detect unique prefixes

      for each prefix
      {
            count specific events after prefix
      }

      translate counter to relative values

      return weightMatrix
}
```

Algorithmus 5.3: Aufbau Markov-Composer

Die Struktur und Verwendung des Markov-Composers ist sehr einfach und geradlinig. Im Gegensatz dazu basiert die Implementierung eines genetischen Algorithmus zur Komposition sehr stark auf Ausprobieren. Nachdem die grundsätzliche Struktur festgelegt ist, gilt es, unterschiedliche Grundpopulationen und Fitness-Funktionen zu testen. Hier lässt sich, gerade bei Musik, im Vorhinein keine klare Aussage treffen, wie man die besten Resultate erzielen kann.

Die verwendeten Fitness-Funktionen standen bereits sehr früh fest. Die initiale Population des genetischen Algorithmus im *PMBC* wurde dagegen erst nach einigen Versuchs-Iterationen finalisiert. Zu Beginn der Entwicklungsarbeit bestand diese noch aus zufälligen Noten innerhalb eines definierten Bereichs. In diese randomisierte Notenfolge wurden dann die ausgewählten Motive an beliebige Stellen kopiert. Ein Mehrfachauftreten war nicht möglich. Mit dieser Grundpopulation konnten zwar rhythmisch und melodisch gute klingende Resultate erzielt werden, die Parallelen zu den entsprechenden Motiven waren jedoch nicht oder nur in geringem Ausmaß vorhanden. Die ursprünglich in die initiale Population kodierten Motive wurden im Laufe der Generationen meist durch Crossover und Mutationen derart zersetzt, dass sie in der finalen Melodie kaum noch erkennbar waren. Aus diesem Grund wurde im nächsten Schritt die Möglichkeit geschaffen, als Mutationen einzelne Original-Motive an beliebige Stellen in einem Chromosom zu kopieren. Als Ergebnis litt dabei jedoch die Harmonie in der Komposition, da häufig ein markanter Sprung vor auftretenden Motiven zu hören war. Im Gegensatz zur ersten Implementierung waren Ähnlichkei-

ten zu den Inputmusikstücken jedoch wesentlich deutlicher zu hören. Von nun an wurden die ausgewählten Motive, wie auch beim Markov-Composer, in zufälliger Reihenfolge aneinander gereiht, um so die ersten Chromosomen zu kreieren. Anfangs wurden diese noch nicht harmonisch aufeinander abgestimmt. Eine Transposition auf dieselbe Tonart und einen festgelegten Oktavenbereich wurde noch nicht vorgenommen. Das Ergebnis waren klar zu erkennende Parallelen zu den ursprünglichen Musikstücken, aus welchen die Motive extrahiert worden waren. Vom rhythmischen bzw. harmonischen Standpunkt aus gesehen, wurde das Resultat jedoch erst durch eine Harmonisierung der Grundpopulation im Vorfeld besser. Algorithmus 5.4 zeigt den finalen Aufbau des genetischen Algorithmus im *PBMC*.

```
initialize population
while iterations < MAX_ITERATIONS
{
      calculate fitness of all chromosomes
      select fittest chromosomes

      for selected chromosomes
      {
            crossover
            for each crossover
            {
                  if should be mutated
                  {
                        choose mutation-function
                        mutate
                  }
            }
      }
      iterations ++
}
```

Algorithmus 5.4: Aufbau genetischer Composer

Der Entwicklungsprozess des genetischen Composers im *PBMC* durchlief im Laufe der Implementierung also vier Iterationen, bis der finale Stand erreicht wurde. Die übrigen Teile des Algorithmus stellten keine gravierenden Schwierigkeiten dar.

5.4 PBMC - Die Software

Anhand der folgenden Darstellungen wird eine kurze Einführung in den Aufbau der Benutzeroberfläche und die grundsätzliche Bedienung des *PBMC* gegeben.

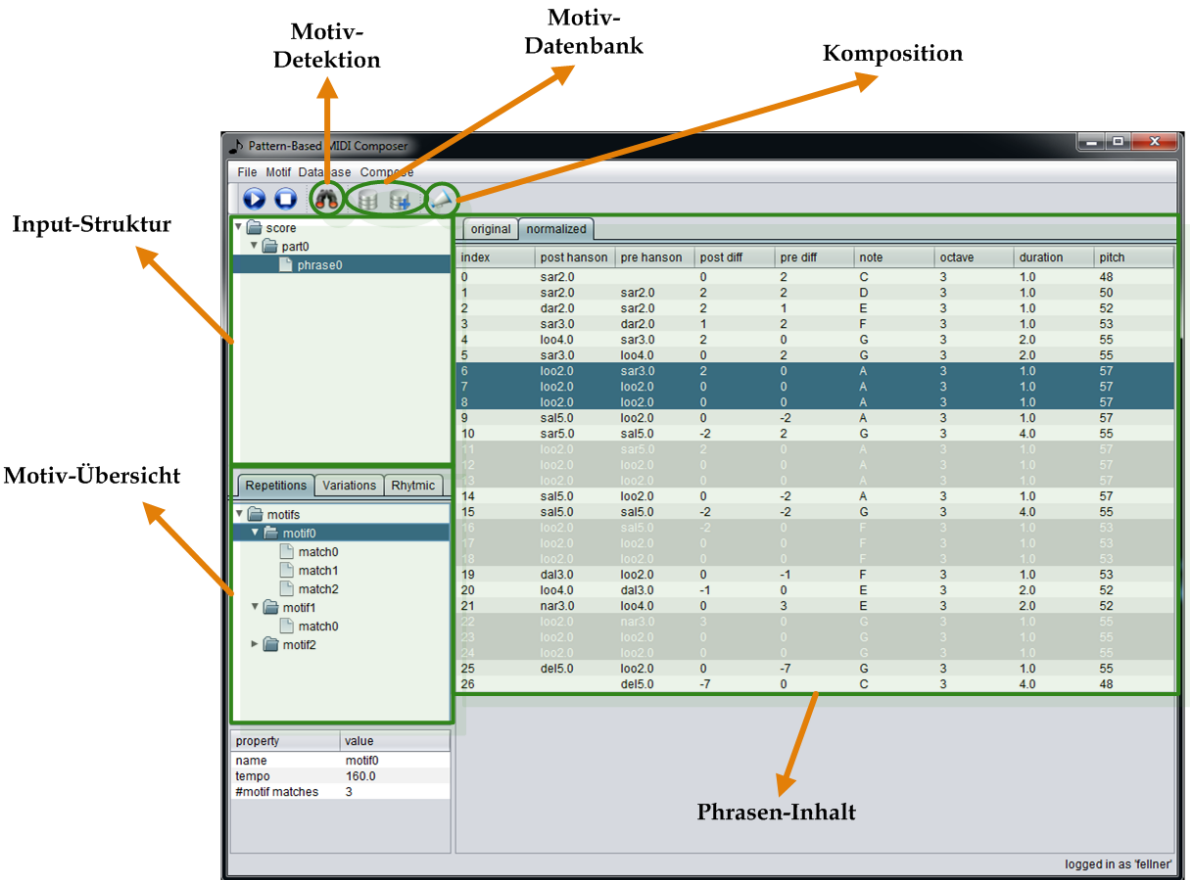

Abb. 5.2: PBMC Benutzeroberfläche

Wie in der Abbildung 5.2 ersichtlich ist, gliedert sich die Oberfläche des *PBMC* im Wesentlichen in drei Anzeigebereiche. Auf der linken Seite wird einerseits die Struktur des geladenen MIDI-Musikstücks angezeigt und andererseits werden gefundene Motive, gegliedert nach Detektor, dargestellt. Der Hauptbereich dient der tabellarischen Übersicht einer ausgewählten Phrase. Hier werden sämtliche enthaltene Noten mit allen ihren Parametern angezeigt. Die farbliche Hinterlegung einzelner Zeilen markiert das ausgewählte Basismotiv inklusive seiner Übereinstimmungen.

Die Steuerung der wichtigsten Funktionen erfolgt über das Programmmenü und die Werkzeugleiste. Dort können die Motiv-Detektion angestoßen und die Motiv-Datenbank sowie der Kompositionsdialog aufgerufen werden. Außerdem können ausgewählte Motive dort in die Datenbank übernommen werden. Diese ist in Abbildung 5.3 dargestellt.

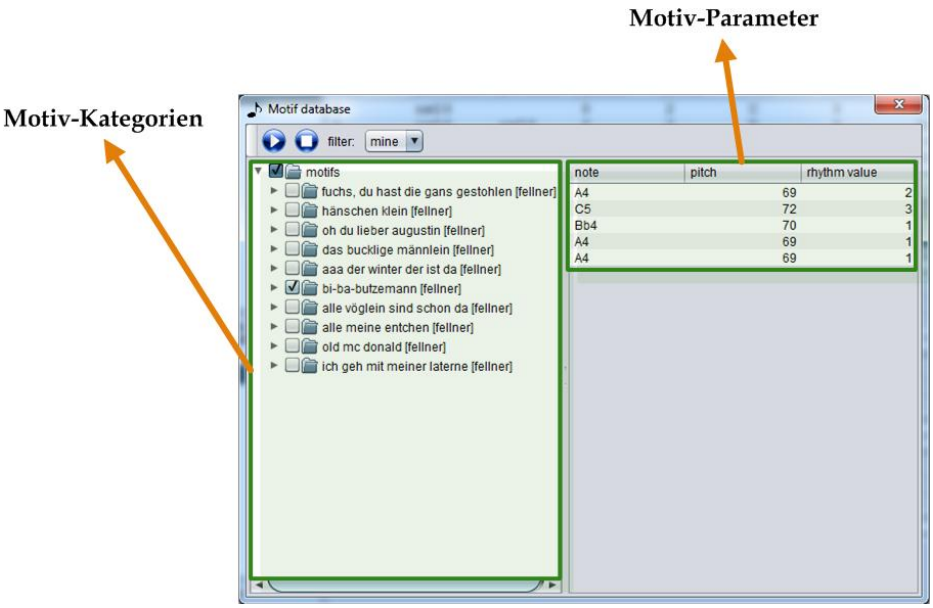

Abb. 5.3: GUI Motiv-Datenbank

Im linken Teil des Dialoges werden sämtliche Kategorien angezeigt, welche in der Datenbank vorliegen. Diese können erweitert werden, um so sämtliche darin befindlichen Motive dargestellt zu bekommen. Im rechten Teil des Dialoges werden die relevanten Parameter eines ausgewählten Motivs angezeigt. Selbstverständlich ist auch das Anhören jeder einzelnen Notenfolge möglich. Durch das Aktivieren der Checkboxen werden die entsprechenden Motive für die Verwendung in einer Komposition aktiviert.

Der Kompositionsdialog ist in Abbildung 5.4 dargestellt. Im oberen Bereich des Fensters werden sämtliche Parameter der Komposition angezeigt. Diese sind composerspezifisch und wechseln je nach Auswahl des verwendeten Ansatzes. Im unteren Teil des Dialogs finden die Einstellungen der Instrumente und Zusatzphrasen Platz.

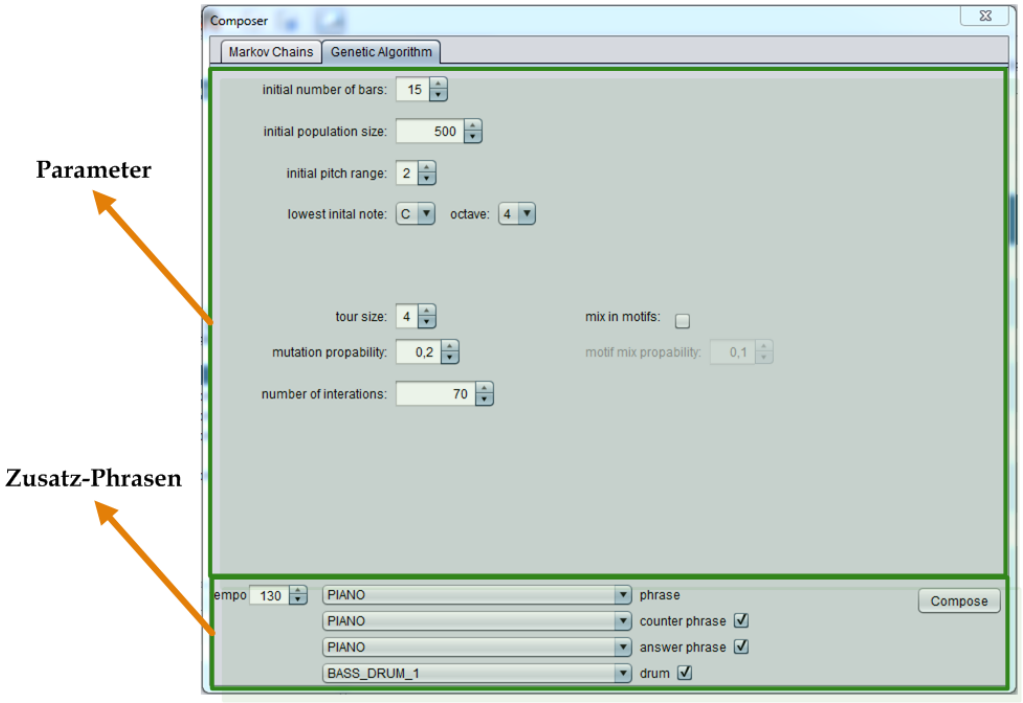

Abb. 5.4: Kompositionsdialog

Die Verwendung des PBMC lässt sich in die folgenden Arbeitsschritte untergliedern:

1. MIDI-Datei öffnen
2. Phrase auswählen
3. Motiv-Detektion durchführen
4. Motive auswählen und in die Datenbank übertragen
5. Gewünschte Motive in der Datenbank markieren
6. Ansatz im Kompositionsdialog auswählen und Parameter definieren
7. Komposition starten

Wenn sich die gewünschten Motive bereits in der Datenbank befinden, kann auch bei Schritt 5 begonnen werden, um eine Komposition zu erstellen.

5.4.1 Parameter

In diesem Abschnitt werden sämtliche Parameter zusammengefasst und erklärt, die in den beiden Kompositions-Verfahren eingesetzt werden. Dabei verwendet der Markov-Composer nur zwei einstellbare Parameter, die wie folgt definiert sind.

Tiefe: Wie in Kapitel 2.1.3 erläutert, gibt der Wert dieses Parameters die Anzahl der zusammenhängenden, vergangenen Ereignisse an, nach denen zu einer gewissen Wahrscheinlichkeit ein bestimmtes Folgeereignis eintritt. Je größer der Wert, umso ähnlicher ist der Output des Markov-Composers seinem Input.

Anzahl der Noten: Mit diesem Parameter kann die Anzahl der Noten gesteuert werden, die im Output vorliegen soll. Der Wert hat keinen Einfluss auf die zeitliche Länge der Komposition.

Zusätzlich lässt sich festlegen, welche Notenparameter durch die Markov-Ketten beeinflusst werden sollen. Mögliche Optionen sind die Tonhöhe, die Tondauer und der Dynamik-Wert. Entsprechend der Auswahl werden ein, zwei oder drei Markov-Matrizen aufgebaut.

Das Ergebnis des genetischen Algorithmus kann mit zehn Parametern beeinflusst werden. Davon sind zwei optional. Die Menge der Einstellungsmöglichkeiten spiegelt den Komplexitätsgrad dieses Ansatzes wieder. Die Parameter werden nachfolgend erläutert.

Taktanzahl: Mit diesem Parameter kann die Anzahl der gewünschten Takte in der Population festgelegt werden.

Initiale Populationsgröße: Die initiale Populationsgröße wird mit dieser Einstellung verändert. Der Wert bezieht sich auf die Anzahl von Noten in den Chromosomen.

Oktaven-Intervall: Das Oktaven-Intervall gibt einen Bereich an, in dem sich die Noten der einzelnen Chromosomen bewegen sollen. Der Wert bezieht sich auf die Anzahl von Oktaven, die dieses festgelegte Intervall umfasst.

Niedrigste Note: Dieser Parameter steuert die niedrigste, vorkommende Note in den Chromosomen. Gemeinsam mit dem Oktaven-Intervall wird der gültige Notenraum festgelegt.

Oktave: Die hier eingestellte Oktave bezieht sich auf die niedrigste Note.

Tour-Größe: Die Tournament-Selektion wurde bereits im Kapitel 4.6.2 erläutert. Dieser Parameter steuert die Anzahl der zufällig ausgewählten Chromosomen, aus welchen das beste ausgewählt wird.

Mutationswahrscheinlichkeit: Die Mutationswahrscheinlichkeit wird in Prozent angegeben. Durch sie wird die Häufigkeit von auftretenden Veränderungen der Chromosomen festgelegt.

Anzahl der Iterationen: Die einzelnen Schritte eines genetischen Algorithmus werden n mal durchgeführt. Dieser Wert gibt die Anzahl der Durchläufe an.

Verwendung von Motiv-Kopien: In Kapitel 5.3.2 wird die Entwicklung in der Generierung der Grundpopulation beschrieben. Das Kopieren zufälliger Motive an beliebige Positionen war einer der verwendeten Ansätze. Auch wenn Motive nun auf andere Weise in die Initialpopulation kodiert werden, ist das ursprüngliche Verfahren noch als optionaler Parameter anwendbar.

Wahrscheinlichkeit von Motiv-Kopien: Diese Einstellung bezieht sich auf den zuletzt genannten Parameter. Durch eine Veränderung dieses Wertes kann die Wahrscheinlichkeit festgelegt werden, mit der ein beliebiges Motiv im Zuge einer Mutation in ein Chromosom kopiert wird.

6 Evaluierung

Dieses Kapitel dient der Präsentation der Ergebnisse des *PBMC*. Zunächst wird auf die Resultate der Motiv-Extraktion eingegangen. Im Anschluss erfolgt eine Gegenüberstellung der beiden angewandten Kompositions-Ansätze. Weiters wurde im Zuge dieser Arbeit ein Online-Fragebogen erstellt, dessen Ergebnisse am Ende des Kapitels diskutiert werden.

6.1 Ergebnisse der Motiv-Extraktion

Um die Qualität der Motiv-Extraktion evaluieren zu können, wurde ein Test-Set an Musikstücken angelegt. Der *PBMC* ist dazu ausgelegt, in jeglichem Input eine große Anzahl an Motiven finden zu können. Da diese sehr verlässlich in beinahe jedem bekannten Kinderlied vorkommen, bestehen die Testdaten ausschließlich aus solchen. Weiters ist bei der Verwendung von Kinderliedern auch der Umstand gegeben, die Ergebnisse sehr einfach manuell überprüfen zu können. Dies wird durch die meist geringe Länge der Melodien und den strukturell einfachen Aufbau zusätzlich erleichtert.

Die Resultate der Motiv-Suche sind sehr umfassend, was aufgrund des verwendeten Brute-Force-Ansatzes auch zu erwarten war. Sämtliche Notenfolgen, welche den Suchparametern entsprechen, werden erkannt. Die Sinnhaftigkeit der Ergebnisse wird nicht programmatisch evaluiert, sondern erfolgt durch die manuelle Auswahl jener Motive, welche in die Datenbank übernommen werden sollen.

6.2 Vergleich der verwendeten Kompositionsansätze

Bei den verwendeten Kompositionsansätzen handelt es sich um zwei grundverschiedene Verfahren. Dem stochastischen Ansatz unter Verwendung von Markov-Ketten steht ein evolutionärer Ansatz gegenüber, welcher durch einen genetischen Algorithmus umgesetzt wurde. Beide Methoden besitzen Vor- und Nachteile. All-

gemein lässt sich sagen, dass der genetische Composer des *PBMC* bessere Ergebnisse im Sinne der Harmonielehre produziert. Im Gegensatz dazu besticht der Markov-Composer durch seine Geschwindigkeit und die Erzeugung von klar abgewandelten Basis-Motiven. In der folgenden Tabelle werden die wesentlichsten Unterschiede zwischen den beiden Verfahren anschaulich dargestellt.

	Markov-Composer	**Genetischer Composer**
Geschwindigkeit	lineare Laufzeit bei Tiefe eins	polynomielle Laufzeit in Abhängigkeit der Anzahl der Takte und Iterationen
Parametrisierbarkeit	zwei Parameter	zehn Parameter
Rhythmus	keine Überprüfung von korrekten Takten	Fitness-Funktionen wirken falschen Taktverschiebungen entgegen
Harmonie	abhängig von den verwendeten Motiven, schlechter Input impliziert schlechten Output	Fitness Evaluierung erzeugt auch bei schlechtem Input adäquate Harmonie
Motiv-Verwendung	meist klare Ableitungen der verwendeten Motive hörbar	Motiv-Abwandlungen sind fast immer zu erkennen, Anzahl der hörbaren Übereinstimmungen variiert

Tabelle 6.1: Gegenüberstellung der Kompositions-Implementierungen

6.3 Fragebogenauswertung

Musik ist sehr subjektiv. Diese Tatsache spiegeln zum Teil auch die Ergebnisse der Online-Umfrage wieder. Der Fragebogen gliederte sich in zwei Abschnitte, welche der Evaluierung der Ergebnisse der Motiv-Extraktion und der beiden Kompositions-

Verfahren entsprechen. Vor allem die Bewertungen der unterschiedlichen Kompositionsverfahren zeigen den Einfluss der verwendeten Parameter auf diese.

Von den insgesamt 91 befragten Personen waren 60% männlich und 40% weiblich. Die Altersverteilung der Stichprobe ist in Tabelle 6.2 dargestellt. Der Großteil der Teilnehmer war zwischen 30 und 39 Jahre alt. Dies entspricht etwa 42% des gesamten Stichprobenumfangs.

	absolute Häufigkeit	**relative Häufigkeit (in Prozent)**
15 - 19 Jahre	1	1
20 - 29 Jahre	20	22
30 - 39 Jahre	38	42
40 - 49 Jahre	16	18
50 - 59 Jahre	12	13
60 - 64 Jahre	4	4
Gesamt	**91**	**100**

Tabelle 6.2: Altersverteilung der Stichprobe

Außerdem wurde nach dem musikalischen Vorwissen der Teilnehmer gefragt. Etwa 36% der Befragten sagen von sich selbst, sehr musikalisch zu sein. Dagegen behaupten 24% das Gegenteil. Herausstechend ist, dass knapp 76% aller Teilnehmer Noten lesen können. In dieser Frage war eine mehrfache Auswahl von Antworten möglich. Die relativen Häufigkeiten beziehen sich auf den gesamten Stichprobenumfang. Die Aufschlüsselung des musikalischen Vorwissens ist in Tabelle 6.3 dargestellt.

	absolute Häufigkeit	**relative Häufigkeit (in Prozent)**
Kann Noten lesen	69	76
Spiele ein Instrument	54	59
Singe in einem Chor	14	15
Komponiere selbst	14	15
Bin sehr musikalisch	33	36
Bin nicht musikalisch	22	24

Tabelle 6.3: Musikalisches Vorwissen der Teilnehmer

Der visuelle Aufbau des Fragebogens wird in Abbildung 6.1 veranschaulicht. Die Darstellung bezieht sich auf die Evaluierung der Qualität der Motiv-Erkennung.

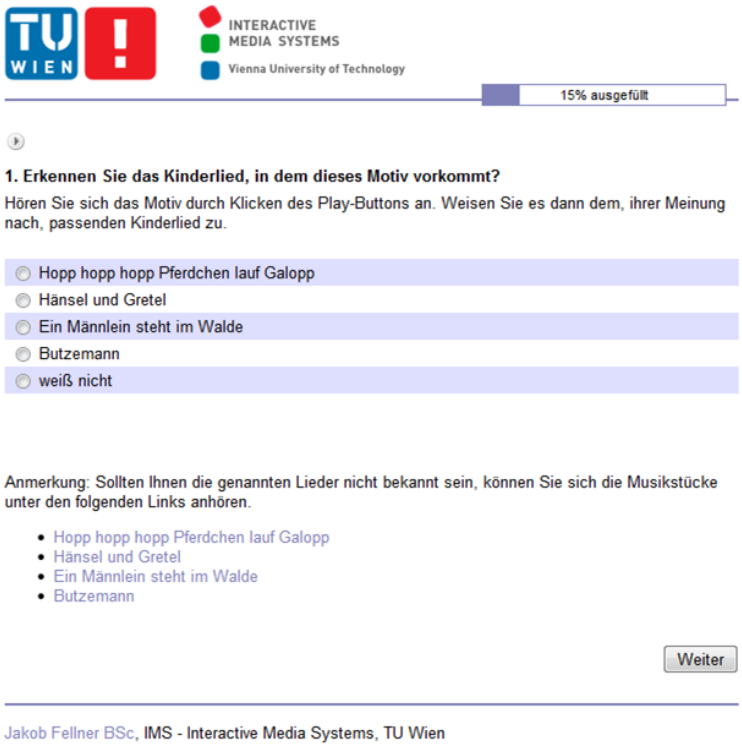

Abb. 6.1: Auszug aus dem Online-Fragebogen

In jenem Abschnitt, welcher die Motiv-Erkennung betrifft, wurden den Teilnehmern in sechs Fragen vom *PBMC* extrahierte Motive verschiedener Herkunft vorgespielt.

Diese kurzen Musik-Ausschnitte sollten jeweils einem bekannten Kinderlied zugeordnet werden. Von den insgesamt 546 gegebenen Antworten waren 64% richtig und 16% falsch. In 20% der Fälle konnte keine Lösung gefunden werden. Die Häufigkeitsverteilung im Bezug auf die sechs Motive ist in Abbildung 6.2 dargestellt.

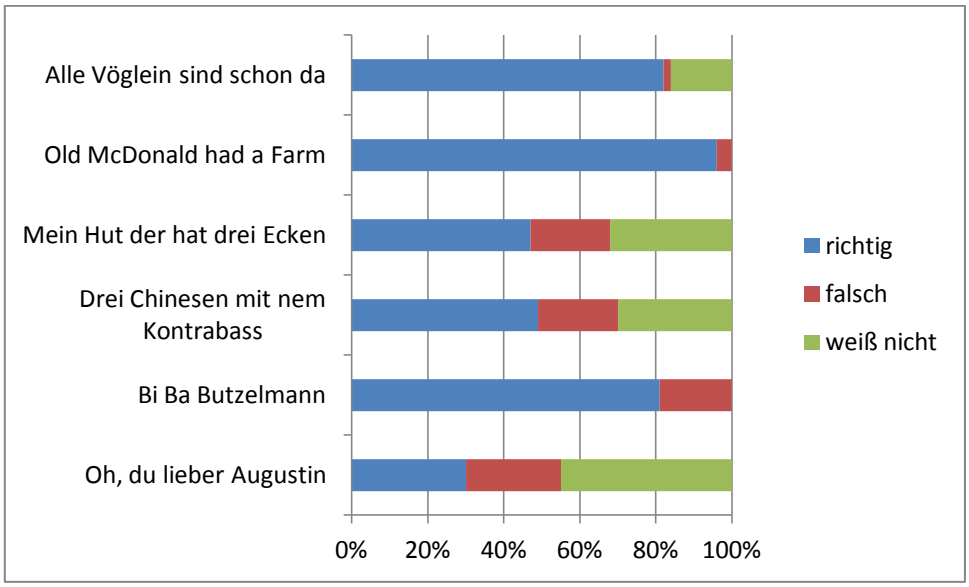

Abb. 6.2: Antwortverteilung Motiv-Erkennung

Aus diesem Verteilungsdiagramm lässt sich ableiten, dass natürlich die Auswahl der entsprechenden Motive eine wesentliche Rolle spielt. Notenfolgen, welche aus dem Refrain des entsprechenden Musikstücks gewonnen wurden, konnten sehr gut zugeordnet werden. In jedem Fall zeigt diese Statistik, dass der *PBMC* meist in der Lage ist, sehr gut erkennbare Motive zu extrahieren.

In der Umfrage wurden die Teilnehmer außerdem gebeten, generierte Kompositionen mit 1 ("gefällt mir nicht") bis 6 ("gefällt mir sehr gut") zu bewerten. Für die Erstellung der Musikstücke wurden mehrere Motive aus jeweils fünf Kinderliedern ausgewählt. Die Komposition wurde dann sowohl mit dem Markov- als auch dem genetischen Composer erstellt. Die Parameter beider Verfahren wurden in den jeweils fünf Kompositionen variiert. Im stochastischen Ansatz wurde eine Tiefe (siehe Kapitel 5.4.1) zwischen eins und vier gewählt. Der genetische Algorithmus arbeitete mit 70, 500, 750 und 1000 Iterationen. In den Abbildungen 6.3 und 6.4 wird die Bewertung der Kompositionen getrennt nach Verfahren dargestellt.

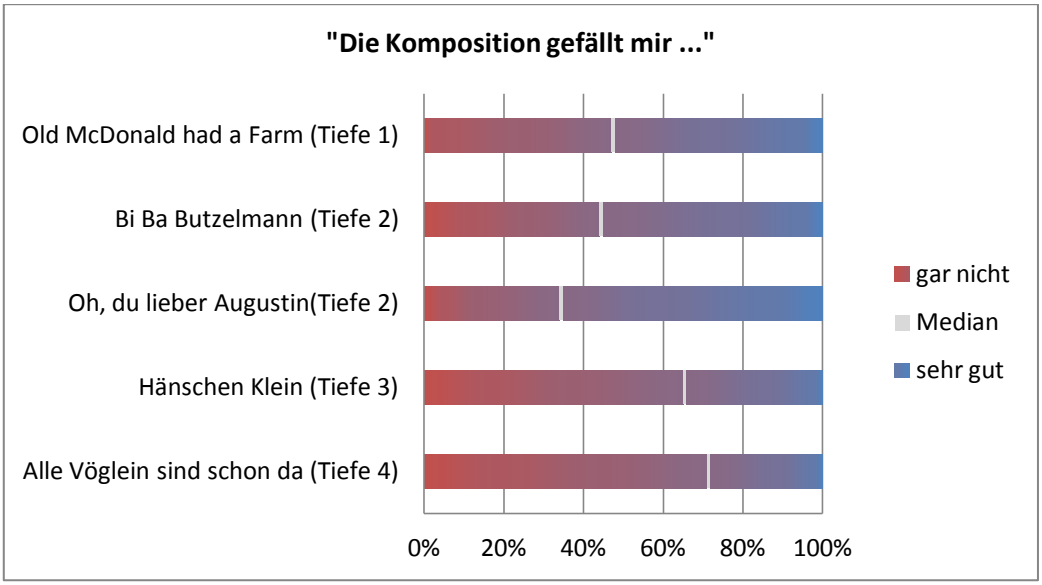

Abb. 6.3: Bewertung Markov-Kompositionen

Wie zu erwarten war, wirkt sich die Wahl der Tiefe, die zur Erstellung der Markov-Matrix verwendet wurde, signifikant auf die Bewertungen der Umfrage-Teilnehmer aus. Während die Komposition mit einer Tiefe von eins sich im mittleren Bewertungsbereich sehr gut verteilt, verschiebt sich die Qualität der Kompositionen bei einer Tiefe von zwei in den positiven Bereich. Steigt der Parameter noch stärker an, tritt ein gegenteiliger Effekt ein.

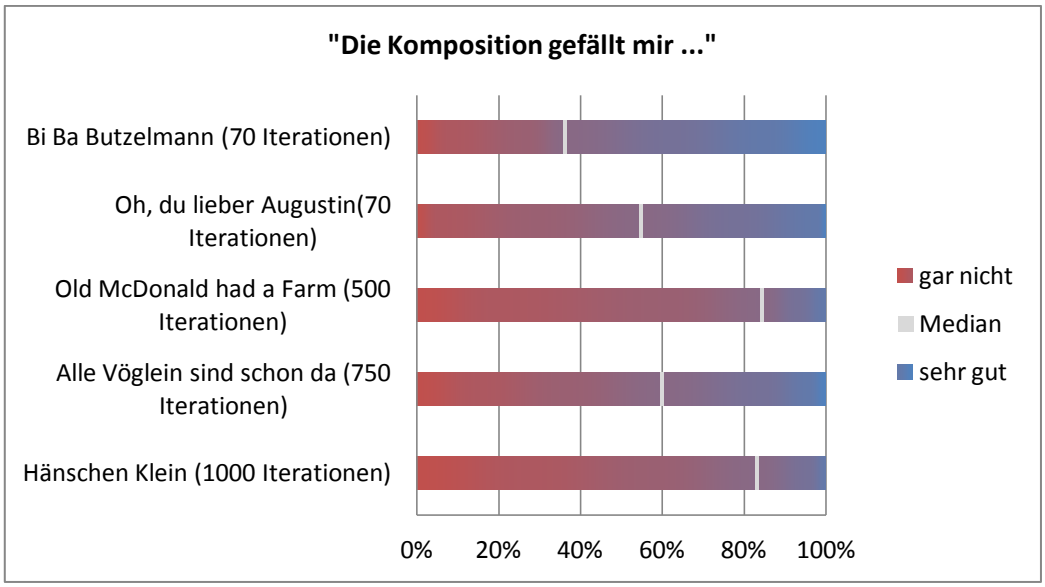

Abb. 6.4: Bewertung genetische Kompositionen

Der genetische Algorithmus schneidet mit 70 Iterationen deutlich besser ab, als wenn zur Generierung eine signifikant höhere Anzahl an Durchläufen verwendet wird. Dies lässt sich damit begründen, dass bei weniger Durchläufen bereits eine Harmonisierung der musikalischen Struktur eintritt, diese jedoch durch Crossover und Mutationen noch nicht soweit variiert ist, dass der rhythmische Aufbau der zugrundeliegenden Motive bis zur Unkenntlichkeit verändert wird.

Die Ergebnisse auf die Frage, ob die Teilnehmer Motive in den einzelnen Kompositionen wiederkennen können, wiesen eine hohe Streuung auf. Die Resultate scheinen nur bedingt von der Wahl der Kompositionsparameter abzuhängen. Allgemein lässt sich sagen, dass die verwendeten Motive im Output des stochastischen Ansatzes mit durchschnittlich 60% sehr gut in der jeweiligen Komposition erkannt wurden. Die Wahl des Tiefen-Parameters scheint hier keinen Einfluss zu haben. Im Gegensatz dazu war der Wiedererkennungswert von Motiven in Kompositionen des genetischen Composers schlechter als erwartet. Stücke, welche in 500 Iterationen und mehr generiert wurden, schneiden zwar nachvollziehbar schlecht ab, jedoch wurden auch Motive in Kompositionen mit 70 Iterationen nur zu durchschnittlich etwa 40% wiedererkannt. Weiters ist kein Zusammenhang zwischen den subjektiven Bewertungen der generierten Musikstücke und der Wiedererkennbarkeit von Motiven gegeben.

Eine Evaluierung der besten Parameter hätte mehrfache Kompositionen mit demselben Basismaterial, aber unterschiedlichen Einstellungen vorausgesetzt. Um jedoch den Fragebogen ansprechend und spannend zu gestalten, fiel bei der Erstellung die Wahl auf verschiedene Basis-Lieder. Optimale Einstellungen der Composer konnten so zwar nicht eindeutig festgestellt werden, es ist jedoch eine Tendenz zu erkennen. Weiters ist durch die Auswahl der Musikstücke in jedem Fall ein Vergleich beider Kompositions-Ansätze möglich.

Allgemein lässt sich sagen, dass beide Verfahren mit optimalen Parametereinstellungen ähnlich gut von den Umfrage-Teilnehmern bewertet wurden. Eine suboptimale Wahl der Parameter wirkt sich in beiden Ansätzen auf die Qualität der Komposition

aus. Jedoch scheinen diese einen wesentlich stärkeren Einfluss bei der Verwendung des genetischen Algorithmus zu haben. Die tatsächliche Güte der Ausgabe beider Verfahren lässt sich anhand der Umfrage nur schwer bewerten. Die Subjektivität in der Wahrnehmung von Musik trägt ihren Teil dazu bei.

7 Schlussfolgerungen und Ausblick

Der Zweck dieser Arbeit war es, ein Software-System zu implementieren, welches Ansätze der algorithmischen Komposition aufgreift, um erkennbare Ableitungen von bekannten Musikstücken zu kreieren. Dazu wurden signifikante Notenfolgen, die Motive, aus ausgewählten Melodien extrahiert und als Input für die Komposition genutzt. Wie die Evaluierung des Online-Fragebogens (siehe Kapitel 6.3) zeigt, ist der implementierte *PBMC* in der Lage, solche prägnanten Motive zu finden und zu extrahieren. Ebenso kann festgehalten werden, dass beide implementierten Kompositions-Verfahren ähnlich gute Ergebnisse produzieren können. Die Qualität der Ausgabe wird dabei deutlich von den entsprechend verwendeten Parametern beeinflusst. Eine gute Basis für optimale Resultate stellt außerdem die Auswahl von gut geeigneten Motiven dar. Die Parallelität zu den ausgewählten Musikstücken konnte durch die quantitative Evaluierung in den meisten Kompositionen nachgewiesen werden.

Die im *PBMC* implementierten Ansätze stellen eine solide Grundlage für eine potentielle Weiterentwicklung dar. Durch diverse Erweiterungen der einzelnen Module ergibt sich ein enormes Verbesserungspotential. In der Motiv-Extraktion könnte ein zeitoptimiertes Suchverfahren verwendet werden. Somit würde die Motiv-Suche auch in längeren Musikstücken deutlich rascher Ergebnisse liefern. Weiters kann die Analyse von polyphonen Melodien angedacht werden. Um eine aussagekräftigere Vergleichsbasis zu schaffen, könnten im Zuge einer Weiterentwicklung noch andere Kompositionsansätze implementiert werden. Die Resultate des eingesetzten Markov-Composers würden durch eine Normalisierung der Takte noch optimiert werden. Ebenso könnten die Ergebnisse des genetischen Algorithmus unter Umständen durch eine Überarbeitung der verwendeten Fitness-Funktionen verbessert werden. Im Speziellen wäre eine Erhöhung der Güte eines Chromosoms bei partieller Übereinstimmung mit entsprechenden Motiven vorteilhaft. Eine solche Änderung könnte dem allgemein eher schlechteren Wiedererkennungswert von Motiven in solchen Kompositionen entgegenwirken.

Quellen

[1] G. Nierhaus, *Algorithmic Composition*: Springer Verlag, 2009.

[2] M. Edwards, "Algorithmic Composition: Computational Thinking in Music," *Communications of the ACM,* vol. 54, no. 7, pp. 58-67, Juli 2011, 2011.

[3] I. Peterson. "Mozart's Melody Machine," 6. April 2013; http://www.sciencenews.org/view/generic/id/1915/description/Mozarts_Melody_Machine.

[4] "Abjad Documentation," 6. April 2013; http://abjad.mbrsi.org/examples/mozart/index.html.

[5] T. Anders, "Composing Music by Composing Rules: Computer Aided Composition employing Constraint Logic Programming," Sonic Arts Research Centre, Queens University Belfast, 2003.

[6] E. Miranda, *Composing Music with Computers*: Focal Press, 2001.

[7] "Mandelbrot Set," 26. April 2013; http://commons.wikimedia.org/wiki/Mandelbrot_set.

[8] C. R. Reeves, and J. E. Rowe, *Genetic Algorithms - Principles and Perspectives: A Guide to GA Theory*, p.^pp. 26: Kluwer Academic Publishers, 2002.

[9] "House Evolution 2," 10. April 2013; http://web.arch.usyd.edu.au/~rob/applets/house/House2.html.

[10] H. Knaus, and G. Scholz, *Formen in der Musik: Herkunft Analyse Beschreibung, Band 1*: Österreichischer Bundesverlag, 1988.

[11] Lemacher, and Schroeder, *Formenlehre der Musik*: Musikverlage Hans Gerig, 1962.

[12] C. Kühn, *Formenlehre der Musik*, 7. Auflage ed.: Bärenreiter-Verlag, 1987.

[13] M. A. Casey, "Content-Based Music Information Retrieval: Current Directions and Future Challenges," *Proceedings of the IEEE,* vol. 96, no. 4, pp. 668-696, April 2008, 2008.

[14] D. Hosken, *A Introduction to Music Technology*: Routledge, 2011.

[15] M. M. Association. 12. April 2013; http://www.midi.org/techspecs/midimessages.php.

[16] K. Verbeurgt, M. Dinolfo, and M. Fayer, "Extracting Patterns in Music for Composition via Markov Chains," *Innovations in Applied Artificial Intelligence*, pp. 1123-1132: Springer, 2004.

[17] A. Jiménez, M. Molina-Solana, F. Berzal, and W. Fajardo, "Mining transposed motifs in music," *Journal of Intelligent Information Systems,* vol. 36, no. 1, pp. 99-115, Februar 2011, 2011.

[18] "MusicXML," 15. April 2013; http://www.musicxml.com/.

[19] J. F. Serrano, and J. M. Inesta, "Music Motive Extraction Through Hanson Intervallic Analysis." pp. 154-160.

[20] A. Pinto, "Relational motif discovery via graph spectral ranking." pp. 102-109.

[21] C.-C. J. Chen, and R. Miikkulainen, "Creating Melodies with Evolving Recurrent Neural Networks." pp. 2241-2246.

[22] C. Bell, "Algorithmic music composition using dynamic Markov chains and genetic algorithms," *Journal of Computing Sciences in Colleges,* vol. 27, no. 2, pp. 99-107, Dezember 2011, 2011.

[23] D. Matić, "A Genetic Algorithm for Composing Music," *Yugoslav Journal of Operations Research,* vol. 20, pp. 157-177, April 2010, 2010.

[24] E. Özcan, and T. Erçal, "A genetic algorithm for generating improvised music." pp. 266-277.

[25] G. Papadopoulos, and G. Wiggins, "A Genetic Algorithm for the Generation of Jazz Melodies."

[26] Y. M. A. Khalifa, B. K. Khan, J. Begovic, A. Wisdom, and A. M. Wheeler, "Evolutionary music composer integrating formal grammar." pp. 2519-2526.

[27] J. A. Biles, "Improvizing with Genetic Algorithms: GenJam," *Evolutionary Computer Music*, pp. 137-169: Springer, 2007.

[28] J. A. Biles. "GenJam," 15. April 2013; http://igm.rit.edu/~jabics/GenJam.html.

[29] G. Weinberg, M. Godfrey, A. Rae, and J. Rhoads, *Computer Music Modeling and Retrieval. Sense of Sounds* p.^pp. 351-359: Springer, 2008.

[30] M. Dostál, "Genetic Algorithms as a Model of Musical Creativity – On Generating of a Human-Like Rhythmic Accompaniment," *Computing and Informatics,* vol. 22, pp. 321-340, 2005.

[31] B. Manaris, D. Vaughan, C. Wagner, J. Romero, and R. B. Davis, "Evolutionary music and the zipf-mandelbrot law: developing fitness functions for pleasant music." pp. 522-534.

[32] B. Manaris, P. Machado, C. McCauley, J. Romero, and D. Krehbiel, "Developing fitness functions for pleasant music: zipf's law and interactive evolution systems." pp. 498-507.

[33] "Encyclopaedia Britannica," 26. April 2013; http://www.britannica.com/EBchecked/topic/709359/Zipfs-law.

[34] "OpenMusic," 15. April 2013; http://repmus.ircam.fr/openmusic/home.

[35] "Symbolic Composer," 15. April 2013; http://www.symboliccomposer.com/page_main.shtml.

[36] "Max," 15. April 2013; http://cycling74.com/products/max/.

[37] "HarmonyBuilder," 15. April 2013; http://www.harmonybuilder.com/index.html.

[38] "Finale," 15. April 2013; http://www.finalemusic.com/products/finale/.

[39] "Liquid Notes," 15. April 2013; http://www.re-compose.com/liquid-notes-music-software.html.

[40] H. Hanson, *Harmonic Materials of Modern Music*: Appleton-Century-Crofts, 1960.

[41] J. F. Serrano, and J. M. Inesta, "Comparison of Hanson Intervallic Representations for Music Information Retrieval." pp. 147-153.

[42] "JFuge," 22. April 2013; http://www.jfugue.org/.

[43] "JMusic," 22. April 2013; http://sourceforge.net/projects/jmusic/?sort=date#reviews-n-ratings.

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmen