Merkmalsmodellierung für das maschinelle Lernen von Musikstilen

zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
der Fakultät für Informatik
der Universität Fridericiana zu Karlsruhe (TH)

genehmigte

D i s s e r t a t i o n

von
Karin Höthker
aus Marburg

Tag der mündlichen Prüfung:	13. Juli 2005
Erster Gutachter:	Prof. Dr. Wolfram Menzel
Zweiter Gutachter:	Prof. Dr. Werner Zorn

Vorwort

Die vorliegende Arbeit entstand im Rahmen der Forschungsgruppe „Informationsstrukturen in der Musik“ an der Universität Karlsruhe. Als ich mit meiner Dissertation begann, fand ich bereits eine breite Grundlage musikinformatischer Konzepte vor. Angefangen bei der Vorlesung „Musik und Informatik - ein Brückenschlag“[88] von Prof.Dr. Zorn über die regelbasierte Choral-Harmonisierung bis hin zu den Harmonisierungs- und Melodieumspielungssystemen HARMONET und MELONET waren in der Forschungsgruppe schon eine Vielzahl von Themen untersucht worden. Die bestehenden Systeme hatten die Modellierung von Melodien jedoch nur im Ansatz behandelt, da in Melodien eine Vielzahl unterschiedlicher Merkmale zueinander in Beziehung stehen. Die vorliegende Arbeit entwickelt Operatorgraphen als neues Beschreibungsmittel für das oftmals komplexe Beziehungsgeflecht musikalischer Kompositionen.

Ich danke meinem Hauptreferenten Prof.Dr. Menzel für die Betreuung der Arbeit und die wertvollen inhaltlichen Impulse. Prof.Dr. Zorn danke ich für die Übernahme des Korreferats. In der Musikgruppe gilt mein herzlicher Dank Dominik Hörnel, Christian Spevak und Joachim Langnickel für die hervorragende Zusammenarbeit bei der Projektarbeit ebenso wie bei der Betreuung des Seminars „Ausgewählte Kapitel der Musikinformatik“. Ebenfalls danken möchte ich Christina Anagnostopoulou und Belinda Thom für die intensive Zusammenarbeit, die ihren Niederschlag in mehreren Veröffentlichungen fand. Ferner danke ich Jean-Baptiste Goyeau und Mathieu Steelandt für ihre engagierten Diplomarbeiten. Den Kollegen der Neuro-Gruppe und des Instituts für Logik, Komplexität und Deduktionssysteme danke ich für das angenehme Arbeitsklima, insbesondere Ralf Schoknecht, Thomas Ragg und Martin Riedmiller.

Die Projekte „Informationsstrukturen in der Musik“ und „Modellierung melodischer Strukturen mit lernbasierten Verfahren“, in deren Rahmen diese Arbeit entstanden ist, wurden von der Deutschen Forschungsgemeinschaft (DFG) und der Klaus-Tschira-Stiftung gefördert.

Mein herzlicher Dank gilt Frau Prof.Dr. Wagner, die es mir nach der Emeritierung von Prof. Menzel ermöglicht hat, meine Arbeit in einem universitären Arbeitsumfeld abzuschließen. Das freundliche und offene Klima der Algo-Gruppe werde ich in guter Erinnerung behalten. Für die lustige Zeit im gemeinsamen Büro bedanke ich mich besonders bei Silke Wagner und Robert Görke.

Neubiberg, im Juli 2006

Karin Höthker

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Gegenstand dieser Arbeit ist die Analyse von Melodien mit Methoden des maschinellen Lernens.

1.1 Beispiel

Eine Melodie lässt sich auf die einfachste Weise als eine Folge von Tönen beschreiben. Ein Beispiel zeigt jedoch, dass die Darstellung einer Melodie als Tonfolge die strukturellen Verflechtungen erfasst, die innerhalb einer Melodie auftreten können. Im Thema des *Largo* aus dem *Winter* der "Vier Jahreszeiten" von Antonio Vivaldi durchdringen einander verschiedene melodische und rhythmische Bewegungen (Abbildung 1.1). In Takt 3-4 sind zwei gegenläufige Bewegungen miteinander verschränkt: Aufsteigende melodische Linien bilden eine Sequenz, die sich auf einer übergeordneten Zeitebene abwärts bewegt. Ähnlich in Takt 5-6: Dort schwingen sich Wechselnotenmotive zu Zieltönen auf, die auf einer übergeordneten Zeitebene eine aufsteigende Linie bilden. Das auftaktige rhythmische Motiv  und seine Abwandlung  ziehen sich durch die ganze Melodie und rufen damit einen Eindruck rhythmischer Geschlossenheit hervor.

Diese Beobachtungen zeigen, dass die Beschreibung einer Melodie lediglich als Tonfolge – wie in bisherigen Ansätzen der lernbasierten Melodiemodellierung weitestgehend der Fall – dem

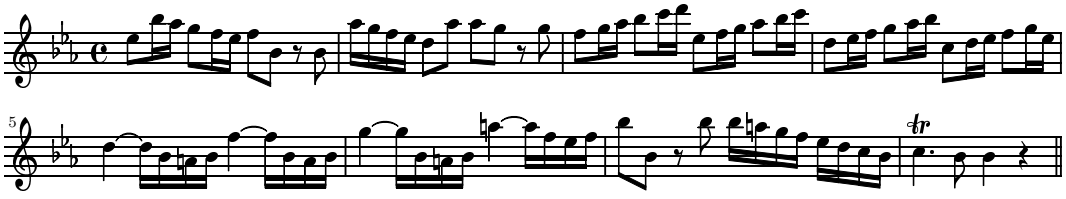


Abbildung 1.1: Antonio Vivaldi: Largo (Vier Jahreszeiten – Winter)

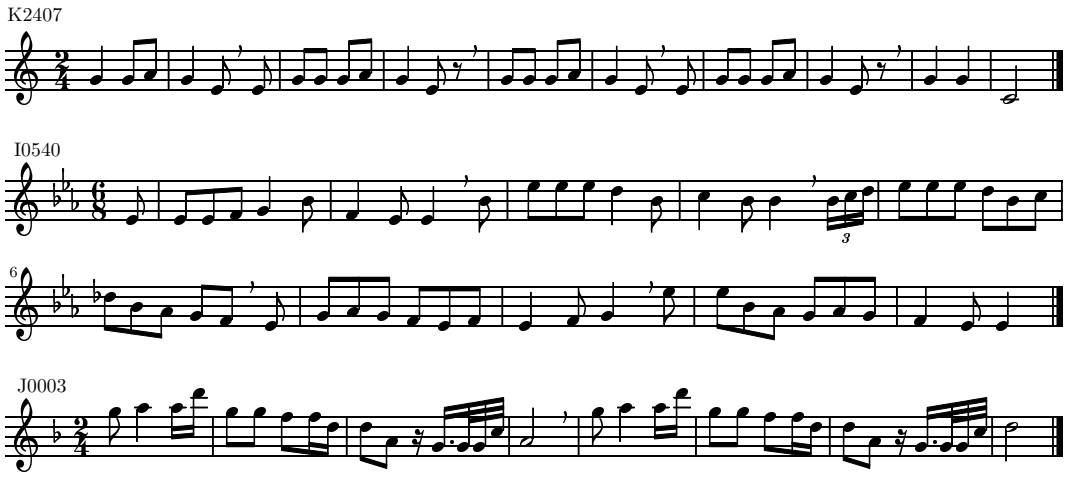


Abbildung 1.2: Drei Melodien aus der Essener Volksliedsammlung K2407: Kinderlied "Nix in der Grube", I0540: Irisches Liebeslied, J0003: Volkslied aus der chinesischen Provinz Shanxi

komplexen Beziehungsgeflecht ihrer Teilstrukturen nicht gerecht wird. Bei einer computerunterstützten Analyse musikalischer Stile, die auf die Untersuchung der inneren musikalischen Zusammenhänge von Melodien abzielt, ist es daher wichtig, musikalische Struktureigenschaften (sogenannte *Merkmale*) explizit zu repräsentieren.

Es ist nicht möglich, eine Repräsentation zu entwerfen, die alle denkbaren melodischen Merkmale abdeckt, da diese zu zahlreich sind und abhängig von Fragestellung und untersuchtem Musikstil variieren. Das oben diskutierte Beispiel zeigt aber einige allgemeine Prinzipien auf, die beim Entwurf einer Repräsentation musikalischer Merkmale berücksichtigt werden sollten: Musikalische Ereignisse wie melodische Bewegung finden auf mehreren Zeitebenen statt. Musikalische Ereignisse innerhalb einer Zeitebene müssen nicht in gleichmäßigen Abständen angeordnet sein und können einander sogar überlappen. Charakteristische Strukturen wie z.B. ein rhythmisches Motiv durchziehen eine Melodie und setzen damit zeitlich entfernte Positionen einer Melodie zueinander in Beziehung. Die Analyse unterschiedlicher Merkmale (wie z.B. rhythmische Motive und Bewegungsrichtung) beleuchtet unterschiedliche Perspektiven und liefert so verschiedene Sichten auf dieselbe Melodie.

Ein zweites Beispiel illustriert die Bandbreite an möglichen Merkmalen bei der Untersuchung von Volksliedern unterschiedlicher Herkunft, die im Vergleich mit durchkomponierten Melodien wie z.B. dem barocken Thema in Abbildung 1.1 eine eher einfache Struktur besitzen. Abbildung 1.2 zeigt drei Volkslieder aus verschiedenen Melodiecorpora der Essener Volksliedsammlung [19], einer Sammlung von etwa 10000 Volksliedern europäischer und chinesischer Herkunft, die unter Leitung des Musikwissenschaftlers Helmut Schaffrath zusammengetragen wurde.

Die erste Melodie K2407 ist ein Kinderlied aus Thüringen und Schlesien, das bei Tanz- und Laufspielen gesungen wurde. Die leicht variierte Wiederholung der kurzen zweitaktigen

Phrasen, die ihrerseits Tonwiederholungen enthalten, die Konzentration der Melodie auf den Terz/Quintbereich der Skala und der bestätigende Quintfall in den letzten beiden Takten verleihen der Melodie den Charakter eines Abzählreims.

Die zweite Melodie I0540 ist ein irisches Liebeslied, dessen melodische Entwicklung großzügiger angelegt ist als beim Kinderlied K2047. Auch hier sind die Phrasen zweitaktig, sie erscheinen jedoch durch den metrisch zweigeteilten $\frac{6}{8}$-Takt länger als bei der ersten Melodie im $\frac{2}{4}$-Takt. Jede Phrase entwickelt sich aus ihrer Vorgängerin, indem sie rhythmische und melodische Elemente aufgreift (z.B. Dreifachrepetition des Tons *es* in den Takten 3 und 5, Sekundschritte mit Richtungswechsel in den Takten 7 und 9). Der triolische Auftakt zu Takt 5 wäre in einem Kinderlied untypisch, ebenso die tiefalterierte siebte Stufe *des* in Takt 6, die eine Harmonisierung des Takts mit der Harmoniefolge Doppelsubdominante-Tonika nahelegt.

Die dritte Melodie J0003 ist ein Volkslied aus der chinesischen Provinz Shanxi. Die Melodie besteht aus zwei viertaktigen Phrasen und gehört damit zu den kurzen Shanxi-Melodien der Sammlung. Die zweite Phrase wiederholt die erste, endet aber auf *d* statt auf *a*. Eine Verzierung führt zu den Schlusstönen beider Phrasen hin. Die Folge von schneller Quarte und Quinte in Gegenbewegung an einer metrisch unbetonten Position (Takt 1 und 5) hat ebenfalls Verzierungscharakter und würde einen durchschnittlichen europäischen Volksliedsänger überfordern. Der Tonvorrat der Melodie ist die pentatonische Skala *f-g-a-c-d*. Im Vergleich zu den beiden anderen Melodien springen der große Ambitus (Duodezime) und die hohe Lage (Spitzenton: d''') ins Auge.

Die drei diskutierten Melodien weisen also unterschiedliche Merkmale auf, die man bei der Charakterisierung der zugehörigen Volksliedstile als Kandidaten für stilrelevante Merkmale verwenden kann. Gestützt auf eine größere Zahl von Melodiebeispielen kann dann mit Hilfe lernbasierter Methoden die statistische Relevanz der Kandidaten untersucht werden.

1.2 Ziele

1.2.1 Informatische Ziele

Repräsentation zeitabhängiger Merkmale. Aus Sicht der Informatik handelt es sich bei den Musikbeispielen aus dem vorhergehenden Abschnitt um diskrete Zeitreihen mit einer komplexen inneren Struktur. Wenn diese innere Struktur nur implizit gegeben ist (wie z.B. bei der Repräsentation von Melodien als Tonfolgen), ist es wenig aussichtsreich, Methoden des maschinellen Lernens einzusetzen, da sie nicht über das problemspezifische Wissen verfügen, das nötig wäre, um relevante Strukturzusammenhänge aus den Basisdaten zu extrahieren. Man benötigt daher Vorverarbeitungsverfahren, mit denen sich die Daten so aufbereiten lassen, dass den maschinellen Lernverfahren durch die Ausgangsdaten implizierte Merkmale explizit zur Verfügung stehen.

Ein informatisches Ziel dieser Arbeit besteht darin, eine Repräsentation für zeitabhängige Merkmale zu entwickeln, die die Aufbereitung diskreter Zeitreihen für das maschinelle Lernen erlaubt. Dabei ist es wichtig, dass die Berechnung von Merkmalen und Lernmustern

automatisch erfolgt, da beim maschinellen Lernen zahlreiche Trainingsbeispiele benötigt werden, die nicht mit akzeptablem Aufwand manuell bearbeitet werden können.

Wenn man diskrete Zeitreihen unterschiedlicher Herkunft untersuchen will, muss das Problemwissen einer solchen Merkmalsrepräsentation austauschbar sein. Die Beispiele im ersten Abschnitt haben gezeigt, dass die Modellierung jedes musikalischen Stils andere Merkmale erfordert. Darüberhinaus soll der entwickelte Ansatz so allgemein sein, dass er auch auf nicht-musikalische diskrete Zeitreihen wie z.B. Finanzzeitreihen anwendbar ist.

Für Algorithmen wie das Trainieren von Klassifikatoren, die Merkmalsselektion oder die musikalischen Stilunterscheidung, die die aufbereiteten Zeitreihen als Eingabe nutzen, sollte das verwendete Problemwissen nicht sichtbar sein, damit sie unabhängig von einer konkreten Aufgabenstellung implementiert werden können. Merkmale werden dann als Variable der Algorithmen behandelt. Auf diese Weise entstehen Familien von Algorithmen, die gleich arbeiten, aber unterschiedliche Merkmale verwenden. Dies stellt eine Verallgemeinerung musikspezifischer Algorithmen dar, deren Merkmale üblicherweise fest implementiert sind.

Der Nutzen der hier entwickelten Repräsentation für das maschinelle Lernen liegt also darin, dass sie nicht nur für musikalische, sondern für beliebige Merkmale eine Aufbereitung diskreter Zeitreihen mit komplexer innerer Struktur für das maschinelle Lernen ermöglicht. Dabei werden insbesondere variable Zeitebenen berücksichtigt, die sich aus unterschiedlichen Interpretationen der Daten ergeben. Finanzzeitreihen und Absatzprognosen sind nichtmusikalische Beispiele, bei denen die Betrachtung variabler Zeitebenen sinnvoll ist. Ragg [70] sagt z.B. Absatzzahlen für BILD-Zeitungen vorher, indem er Differenzen von Absatzzahlen im Wochenabstand berechnet. Dabei müssen Feiertage und besondere Ereignisse berücksichtigt werden.

Melodien sind aus der informatischen Perspektive deshalb als Untersuchungsgegenstand interessant, weil sie bei beschränktem Ausgangsmaterial besonders komplexe innere Strukturen aufweisen, die mit den bisherigen Ansätzen nicht hinreichend modelliert werden können.

1.2.2 Musikwissenschaftliche Ziele

In Abschnitt 1.1 wurden ein barockes Thema und drei Volkslieder unterschiedlicher Herkunft diskutiert. Ob die dort identifizierten Eigenschaften auch für stilistisch ähnliche Melodien gelten, lässt sich aber nicht aus einer Einzelfallbetrachtung, sondern nur anhand einer statistisch abgesicherten Analyse einer großen Zahl von Beispielen des gleichen Stils schließen. In der vorliegenden Arbeit soll untersucht werden, wie sich Verfahren des maschinellen Lernens auf die musikalische Stilanalyse anwenden lassen.

Der Einsatz maschineller Lernverfahren bietet sich besonders bei Problemen an, die sich schlecht durch Regeln formalisieren lassen, für die aber viele Lösungsbeispiele existieren. In der Musik gibt es zum Beispiel kein allgemein akzeptiertes Regelwerk zur Melodiekomposition, während Stimmführungsregeln für den traditionellen vierstimmigen Chorsatz zum musikwissenschaftlichen Grundwissen gehören.

Auffinden stiltypischer Merkmale in Melodien. In dieser Arbeit werden drei Stilmodellierungaufgaben mit Methoden des maschinellen Lernens untersucht. Bei der *Einzelstilmodellierung* werden Merkmale bestimmt, die im Kontext eines vorgegebenen Musikstils

für die Vorhersage eines weiteren Merkmals relevant sind. Bei der *Stilunterscheidung* besteht die Aufgabe darin, Merkmale zu finden, die einem lernbasierten System die Zuordnung von Musikbeispielen zu einem von zwei vorgegebenen Stilen erlauben. Die *Stilerkennung* verallgemeinert die Stilunterscheidung so, dass eine beliebige Anzahl von Stilen vorgegeben werden kann.

Eine Gemeinsamkeit dieser drei Aufgabenstellungen besteht darin, dass zur Auswahl lösungsrelevanter Merkmale ein objektives Gütekriterium zur Verfügung steht. Die Leistungsfähigkeit eines Stilunterscheiders wird z.B. ausgedrückt, indem man seine Generalisierungsleistung mit Hilfe der Stilunterscheidungsgüte auf einer Testmenge schätzt, die nicht zum Einstellen der Modellparameter verwendet wurde. Die Eingabemerkmale des leistungfähigsten Stilunterscheiders werden als besonders relevant für die Unterscheidung der untersuchten Stile ausgewählt.

Diese Vorgehensweise ermöglicht eine Analyse elementarer Zusammenhänge zwischen musikalischen Merkmalen und die automatische Optimierung lernbasierter Modelle zur Prognose musikalischer Merkmale.

Generierung neuer Melodien. Die Prognose musikalischer Merkmale lässt sich auch zur Generierung neuer Melodien einsetzen. Es stellt sich jedoch die Frage, ob ein Modell, das Melodien mit einem Ensemble objektiver Bewerter generiert, tatsächlich das musikalisch-ästhetische Konzept annähert, das Musikexperten oder Versuchspersonen von einem Musikstil haben. Genau genommen lernen die bisher angesprochenen Modelle, die Werte eines musikalischen Merkmals zu prognostizieren. Damit erfasst man, ob das betreffende Merkmal aus den Beispielen für den betrachteten Stil überhaupt gelernt werden kann, und von welchen anderen Merkmalen es abhängt. Die Identifizierung solcher "lernbaren" Merkmale ist eine notwendige Voraussetzung für die Erstellung eines lernbasierten Stilmodells; hinreichend für die Charakterisierung eines Musikstils ist sie aber nicht.

Im Vergleich zu Problemstellungen mit objektiven Gütekriterien besteht die Besonderheit bei der Modellierung eines musikalischen Stils darin, dass das musikalisch-ästhetische Zielkonzept – der musikalische Stil – sich nicht in Regeln fassen lässt. Vielmehr wird angestrebt, kognitive Konzepte eines Musikstils zu approximieren, die Musikexperten durch Studium und Hörerfahrung entwickelt haben. Um ein lernbasiertes Stilmodell mit diesem Optimierungsziel evaluieren zu können, muss das enthaltene Wissen in einer Form aufbereitet werden, die sich mit dem kognitiven Stilkonzept von Musikexperten in Verbindung bringen lässt. Hier bietet sich die Generierung neuer Melodien mit Hilfe eines lernbasierten Modells an, denn die Stiltreue generierter Melodien kann von Musikexperten oder Versuchspersonen besser bewertet werden als eine Menge von Merkmalen, die von einem Algorithmus als relevant eingestuft wurden. Die Gesamterscheinung einer Melodie spiegelt wider, ob die lernbasierten Komponenten des Stilmodells die wesentlichen Aspekte des Stils erfassen.

Ein methodischer Vorteil bei dieser Vorgehensweise besteht darin, dass man keinerlei Annahmen über die Funktionsweise des kognitiven Stilmodells der Musikexperten machen muss. Demgegenüber wäre eine Bewertung von Mengen selektierter Merkmale durch Experten, wie sie die Einzelstilmodellierung, Stilunterscheidung und Stilerkennung liefern, methodisch fragwürdiger, da die selektierten Merkmale im Kontext des kognitiven Stilkonzepts eines Experten möglicherweise keine Bedeutung haben oder sogar das Stilkonzept des Experten beeinflussen.

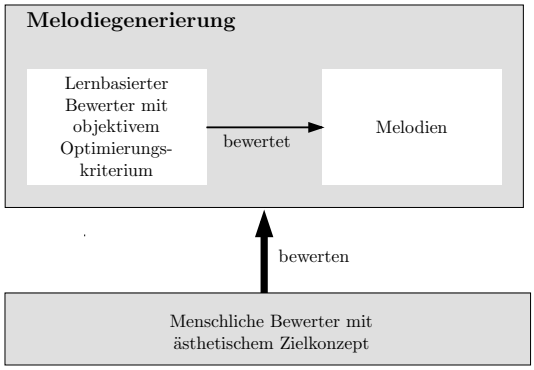


Abbildung 1.3: Zweistufige Bewertung von Melodien bei der Melodiegenerierung mit einem lernbasierten Modell.

Insgesamt wird ein lernbasiertes Stilmodell mit einem zweistufigen Ansatz bewertet (Abbildung 1.3): Es werden Melodien generiert, die ein lernbasierter Bewerter mit einem objektiven Gütekriterium optimiert. In einem übergeordneten Schritt vergleichen Musikexperten die bei der Melodiegenerierung als optimal eingestuften Melodien mit ihrer ästhetischen Zielvorstellung. Die Zusammensetzung des lernbasierten Bewerters und die Generierungsstrategie werden so lange verändert, bis das Stilkonzept der Musikexperten getroffen wird oder keine Verbesserung mehr eintritt.

Für die Musikwissenschaft stellt die Anwendung lernbasierten Verfahren auf musikalische Fragestellungen ein Werkzeug dar, mit dem die Relevanz musikalischer Merkmale statistisch fundiert untersucht werden kann. Die Auswahl der potentiell relevanten Merkmale stellt dabei bereits eine Hypothese über die Eigenschaften des untersuchten Stils dar. Da bei der Modellierung werden die musikalischen Merkmale dann formal definiert. Dadurch werden alle musikalischen Annahmen offengelegt und einer Diskussion zugänglich gemacht.

1.3 Repräsentation zeitabhängiger Merkmale

In dieser Arbeit wird eine Repräsentation für diskrete Zeitreihen entwickelt, die speziell auf die Anforderungen des maschinellen Lernens zugeschnitten ist. Die Repräsentation stellt die Grunddaten einer Zeitreihe und daraus abgeleitete Analysen dar. Sie bildet sowohl die Grundlage für die Berechnung dieser Analysen als auch für die Generierung von Lernmustern und kann an das problemspezifische Wissen unterschiedlicher Fragestellungen angepasst werden. Merkmale einer diskreten Zeitreihe werden getrennt von den Daten der Zeitreihe explizit repräsentiert, so dass sie als Variablen in Algorithmen verwendet werden können.

Ansichten. Zeitreihen bestehen aus Ereignissen, die einen Startzeitpunkt, eine Dauer und einen problemspezifischen Wert besitzen. Merkmale charakterisieren Eigenschaften einer Zeitreihe, indem sie Zusammenhänge zwischen Ereignissen aus einer festgelegten Perspektive beschreiben. In Anlehnung an die unterschiedlichen Datensichten in Softwaresystemen

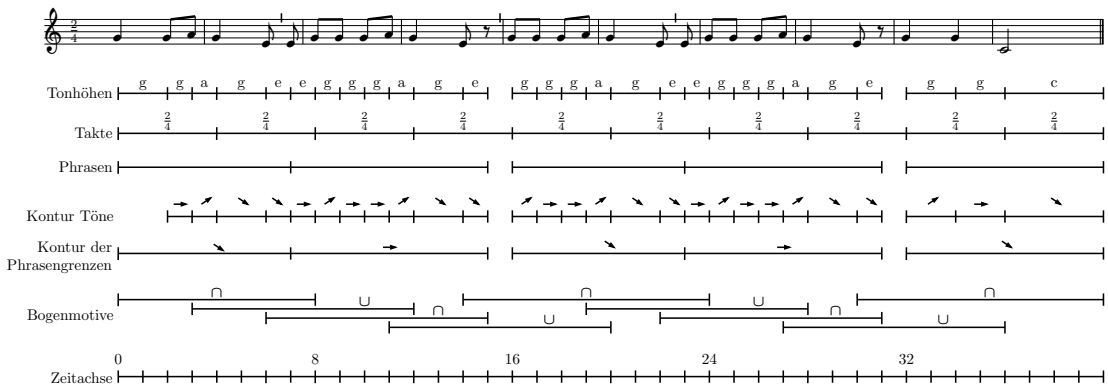


Abbildung 1.4: Fünf Melodieansichten des Kinderlieds “Nix in der Grube” (K2407)

wird die Darstellung einer Zeitreihe bezüglich eines Merkmals in dieser Arbeit als *Ansicht* (engl. *view*) der Zeitreihe bezeichnet.

Abbildung 1.4 zeigt ein Beispiel für ein Melodie, d.h. für eine musikalische Zeitreihe, und fünf Ansichten dieser Melodie. Eine Ansicht besteht aus musikalischen Ereignissen, die sich jeweils aus einem Zeitintervall (*Segment*) und einem musikalischen Symbol (*Wert*) zusammensetzen, wobei das Symbol auch fehlen kann. Die oberen drei Ansichten “Tonhöhen”, “Takte” und “Phrasen” geben die Tonhöhen, Takte und Phrasen der Melodie wieder. Sie ergeben sich direkt aus der Originaldarstellung der Melodie in der EsAC-Volkslied-Sammlung [19]. Die Phrasenansicht besteht aus Segmenten ohne musikalisches Symbol, da eine Phrase allein durch ihre zeitliche Position bestimmt wird. Die Taktansicht gibt die Position der einzelnen Takte wieder und könnte auch einen Auftakt oder Taktwechsel darstellen.

Die untere Ansicht “Bogenmotive” wurde aus der Tonhöhenansicht abgeleitet. Es handelt sich um eine einfache Analyse der Melodie, die die konvexen und konkaven Teilstücke der Melodie inklusive konstanter Randstücke wiedergibt. Diese Ansicht zeigt auch, dass bereits eine einfache Analyse einer Melodie musikalische Ereignisse beinhalten kann, die einander überlappen.

Jede Ansicht enthält musikalische Symbole eines Typs (z.B. Tonhöhen, Taktarten). Ein Beispiel zeigt, dass der Typ einer Ansicht noch nicht festlegt, welches Merkmal einer Melodie die Ansicht wiedergibt: In Abbildung 1.4 sind zwei Ansichten mit Kontursymbolen angegeben. Die erste Ansicht “Kontur Töne” zeigt die Intervallrichtung für aufeinanderfolgende Töne der Melodie. Die zweite Ansicht “Kontur der Phrasengrenzen” bildet die Kontur für den Anfangs- und Endton jeder Phrase und ordnet diese dem Segment der Phrase zu. Beide Ansichten spiegeln unterschiedliche Merkmale der Melodie wider.

Erweiterbare Zeitreihengrammatik. Bei der Beschreibung der Ansichten wurden verschiedene Typen musikalischer Symbole genannt (Tonhöhe, Kontur, Bogenmotiv, Taktart). Die Wahl dieser Typen hängt von der untersuchten Fragestellung ab. Die Modellierung der zeitlichen Zuordnung der Elemente einer Zeitreihe ist hingegen bei allen Zeitreihenproblemen gleich. Als Grundlage für die Definition von Ansichten und die Repräsentation von

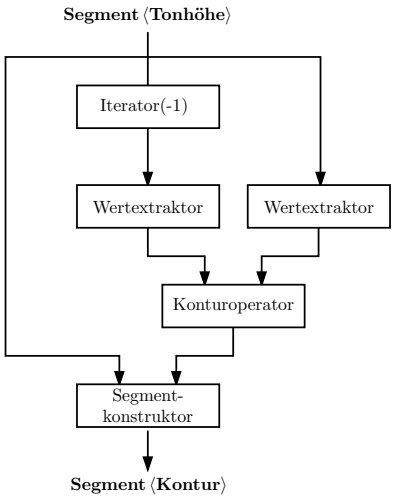


Abbildung 1.5: Operatorgraph zur Berechnung der Ansicht "Kontur Töne" aus Abb. 1.4

Merkmalen wird in dieser Arbeit eine *erweiterbare Zeitreihengrammatik* entwickelt, mit der sich die Symboltypen spezifizieren und generieren, auf denen die weiteren Modelle dieser Arbeit aufbauen. Die Zeitreihengrammatik ist erweiterbar, weil sie einen zeitreihenspezifischen Anteil vorgibt und durch die Definition eines problemspezifischen Anteils an das Anwendungsgebiet angepasst wird.

Operatorgraphen. Die beiden Ansichten vom Typ "Kontur" in Abbildung 1.4 zeigen, dass der Typ der Ansicht noch nicht festlegt, welches Merkmal einer Zeitreihe eine Ansicht wiedergibt. Vielmehr können sich selbst gleiche Ansichten in ihrer Bedeutung unterscheiden, wenn sie auf die Verknüpfung unterschiedlicher Information zurückgehen.

Um diese Mehrdeutigkeit von Ansichten zu berücksichtigen, werden Merkmale in dieser Arbeit getrennt von der Repräsentation der Zeitreihen definiert. Mit *Operatorgraphen* wird ein neues Beschreibungsmittel für Merkmale eingeführt, mit dem Berechnungsvorschriften für Symbole definiert werden. Definieren zwei Operatorgraphen unterschiedliche Funktionen, so haben auch von ihnen erzeugten Ansichten eine unterschiedliche Bedeutung, selbst wenn sie in einem konkreten Beispiel die gleichen Informationen enthalten.

Abbildung 1.5 zeigt einen Operatorgraphen, mit dem man die Kontur benachbarter Tonhöhen in Abbildung 1.4 berechnen kann. Der Operatorgraph setzt sich aus atomaren Aktionen zusammen, den sogenannten *Operatoren*, die zu einem zyklenfreien Graphen verkettet werden. Ein Segment einer Ansicht dient als Eingabe des Operatorgraphen und wird in der durch die Pfeile angegebenen Reihenfolge durch die Operatoren abgebildet und an die nachfolgenden Operatoren weitergereicht.

Im Beispiel werden die Segmente der Tonhöhenansicht aus Abbildung 1.4 als Eingabe des Operatorgraphen verwendet. Von der Eingabe ausgehend sucht der Iterator den linken Nachbarn des Eingabesegments auf. Aus diesem und aus der Eingabe werden mit Wertextrak-

toren die musikalischen Symbole extrahiert (hier Tonhöhen) und an den Konturoperator weitergereicht. Der Segmentkonstruktor verknüpft das berechnete Kontursymbol mit der Zeitinformation des Eingabesegments. Die Ausgabe des Operatorgraphen wird in der Konturansicht gespeichert. Für den zweiten Ton der Melodie, eine Achtel mit Tonhöhe g, erhält man beispielsweise die Kontur gleichbleibend ($\rightarrow$).

Auch bei der Entwicklung der Operatorgraphen werden zeitreihenspezifische von problemspezifischen Aspekten getrennt, um die Integration von neuem Problemwissen in den Ansatz zu erleichtern. Im Beispiel gibt es einen problembezogenen Operator (den Konturoperator) und drei verschiedene zeitbezogenene Operatoren, die vom betrachteten Problemfeld unabhängig sind (Iterator, Wertextraktor und Segmentkonstruktor).

Die Berechnung von Ansichten ist nicht die einzige Einsatzmöglichkeit für Operatorgraphen. So können auch Lernmuster mit Hilfe von Operatorgraphen generiert werden. Der Operatorgraph liefert dann eine numerische Ausgabe, die nicht in einer Ansicht der Datenrepräsentation, sondern in einer Lernmusterdatei abgelegt wird.

Besonders im Vergleich zur festen Implementierung musikalischer Transformationen bietet die Verwendung von Operatorgraphen den methodisch wichtigen Vorteil, dass Operatorgraphen unabhängig von ihrer Struktur immer auf dieselbe Weise ausgewertet werden. Es ist daher möglich, Algorithmen mit frei austauschbaren, von Operatorgraphen dargestellten Merkmalen zu implementieren, ohne dass auf die tatsächliche Beschaffenheit der Merkmale Bezug genommen werden muss. Operatorgraphen bilden damit eine einheitliche Schnittstelle zwischen einem generischen Algorithmus einerseits und einer problembezogenen Repräsentation musikalischer Daten andererseits.

1.4 Stilanalyse

Verallgemeinerte Musikrepräsentation. Während in früheren lernbasierten Ansätzen die Anzahl der in Frage kommenden Merkmale mit vereinfachenden Annahmen reduziert wurde, ist durch die variable Zeitdarstellung und die Austauschbarkeit von Problemwissen die Untersuchung komplexer musikalischer Strukturen nun besser möglich. Beispielsweise beziehen die neuronalen Netze bei der Choralharmonisierung in [32] ihre Eingabemerkmale nur aus einem lokalen Zeitfenster. Bei der Melodieergänzung von Kinderliedern in [89] motivierte der regelmäßige Aufbau der Melodien eine Zerlegung in Zeitsegmente gleicher Länge, wodurch die Menge der darstellbaren Merkmale eingeschränkt wurde. Diese Einschränkungen sind mit dem hier entwickelten Ansatz nicht notwendig.

Merkmalsselektion. Die mit der Verallgemeinerung der Repräsentation einhergehende deutliche Zunahme darstellbarer Merkmale macht es erforderlich, statt der bisherigen manuellen Merkmalsauswahl Merkmalsselektionsverfahren einzusetzen. Ein Merkmalsselektionsverfahren wählt aus einer vorgegebenen Menge von Merkmalen diejenigen aus, die für die Lösung einer Lernaufgabe relevant sind. Die gewonnenen Merkmale können aus Ausgangsmaterial für die Melodiegenerierung genutzt werden.

Melodiegenerierung. Wenn man musikalische Merkmale finden will, die einen musikalischen Stil angemessen beschreiben, geht man von der Grundannahme aus, dass maschinell

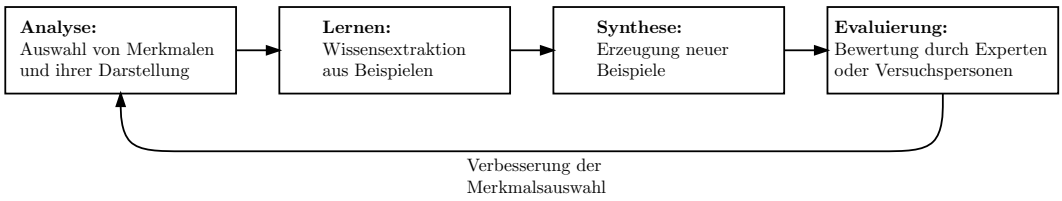


Abbildung 1.6: Zyklische Modellentwicklung mit dem Ansatz "Analyse durch Synthese"

lernbare Zusammenhänge auf die Relevanz der verwendeten Merkmale hindeuten. Der Ansatz der "Analyse durch Synthese" [101] nutzt diese Annahme zur zyklischen Verbesserung eines musikalischen Modells durch die Generierung neuer Musikstücke (Abbildung 1.6). Dabei ist ein lernbasiertes Modell mit austauschbaren Merkmalen gegeben. Nachdem man Merkmale ausgewählt und die Trainingsbeispiele entsprechend vorverarbeitet hat (*Analyse*), werden die lernbasierten Komponenten des Modells trainiert (*Lernen*). Bei der *Synthese* generiert man neue Musikstücke, deren Stiltreue bei der *Evaluierung* durch Experten oder Versuchspersonen bewertet wird. Durch die iterative Veränderung der verwendeten Merkmale und die Variierung der lernbasierten Bewerter wird die Qualität der erzeugten Stücke nach und nach verbessert. Die Generierung von Melodien dient also im Kontext dieser Arbeit der Stilanalyse, nicht der Komposition als Selbstzweck.

Operatorgraphen. Bei den Verfahren zur Stilanalyse, die in dieser Arbeit entwickelt werden, erfüllen die Operatorgraphen unterschiedliche Aufgaben. In allen Verfahren, die in Abschnitt 1.2.2 skizziert wurden, werden Operatorgraphen zur Vorverarbeitung der Lernbeispiele und zur Erzeugung von Lernmustern eingesetzt. Während die drei auf Merkmalsselektion aufgebauten Verfahren Einzelstilmodellierung, Stilunterscheidung und Stilerkennung bei der Optimierung nur auf die im Vorfeld generierten Muster zugreifen, spielen die Operatorgraphen bei der evolutionären Melodiegenerierung eine bedeutendere Rolle. Hier werden sie während der evolutionären Optimierung eingesetzt, um zufällige Melodien zu erzeugen, Melodien durch Mutation zu verändern und sie zu bewerten. Die Bewertung von Melodien mit Operatorgraphen setzt *Neurooperatoren* ein, also Operatoren, die ein neuronales Netz auswerten. Operatorgraphen erweisen sich damit als ein Werkzeug, mit dem die Interaktion lernbasierter Klassifikatoren untereinander und mit zusätzlichem Problemwissen spezifiziert werden kann. Auch eine hybride Bewertung von Melodien, die Regelwissen und lernbasierte Elemente verknüpft, wird auf diese Weise ermöglicht.

Das System MeloLab**.** Im Rahmen dieser Arbeit wurde das System MeloLab, eine flexible Entwicklungsumgebung zur Analyse und Synthese melodischer Strukturen, realisiert. Dort wurden die hier entwickelte Zeitreihenrepräsentation mit Ansichten und Operatorgraphen, sowie die untersuchten Stilanalyse- und Melodiegenerierungsverfahren implementiert.

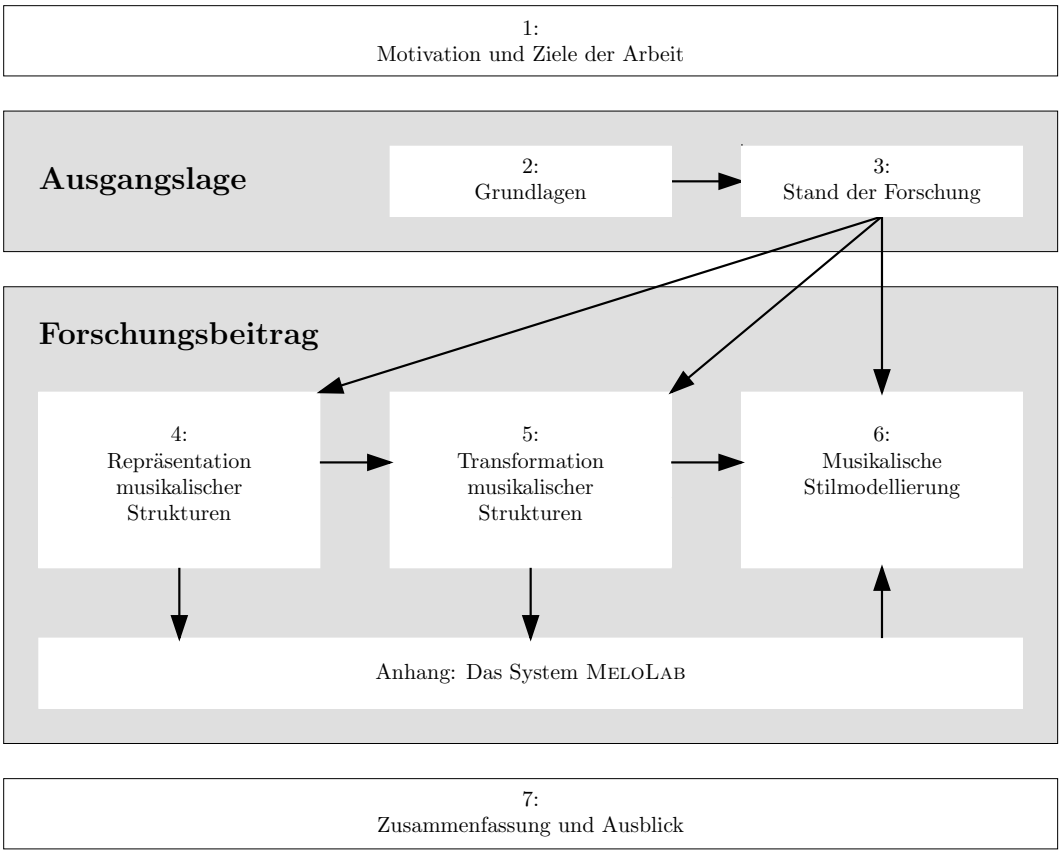


Abbildung 1.7: Aufbau der Arbeit

1.5 Aufbau der Arbeit

Der Rest der Arbeit gliedert sich wie in Abbildung 1.7 skizziert. In Kapitel 2 werden die informatischen und musikalischen Grundlagen vermittelt, die zum weiteren Verständnis notwendig sind. Das Kapitel ist als Referenz angelegt und kann beim Lesen zunächst übersprungen werden. Kapitel 3 ordnet die Arbeit in den gegenwärtigen Stand der Forschung ein. In Kapitel 4 wird eine allgemeine Repräsentation für musikalische Strukturen entwickelt. Auf dieser Basis werden in Kapitel 5 Operatorgraphen eingeführt, mit denen sich diskrete Zeitreihen unter Berücksichtigung von Hintergrundwissen für das maschinelle Lernen aufbereiten lassen. In Kapitel 6 wird gezeigt, wie Operatorgraphen bei der Merkmalsselektion und der Stilerkennung eingesetzt werden können. Die beschriebenen Algorithmen werden experimentell evaluiert. Die Ergebnisse werden in Kapitel 6.3 zusammengefasst. Anhang A skizziert das explorative neuronale System MeloLab, das den neuen Ansatz implementiert.

Kapitel 2

Grundlagen

Die vorliegende Arbeit ist im Spannungsfeld von Musikwissenschaft und Informatik angesiedelt. Dieses Kapitel stellt die Grundlagen aus beiden Bereichen vor, die zum Verständnis des Forschungsbeitrags dieser Arbeit notwendig sind. Dabei werden zwei Ziele verfolgt. Zum einen soll den Leserinnen und Lesern, deren Wissensschwerpunkt in der Musik oder in der Informatik liegt, eine abstrakte Vorstellung der Konzepte aus dem jeweils anderen Fachgebiet gegeben werden. Zum anderen werden die Nomenklatur sowie später benötigte Modelle und Algorithmen definiert. Diese Definitionen sind zum Nachschlagen beim Lesen der späteren Kapitel gedacht.

2.1 Informatisch-mathematische Grundlagen

2.1.1 Reguläre Ausdrücke

Ein regulärer Ausdruck ist ein Bildungsgesetz, das eine Menge von Zeichenketten beschreibt. Beispielsweise kann man mit regulären Ausdrücken Alternativen $A \mid B$ = "A oder B" oder regelmäßig aufgebaute Ausdrücke wie $(ab)^+ = \{ab, abab, ababab, \ldots\}$ beschreiben. Sie werden in Kapitel 4 bei der Definition der verschiedenen Grammatiken verwendet. Die folgenden Definitionen stützen sich auf [86] und [33].

Zunächst benötigt man eine Grundmenge, über der reguläre Ausdrücke gebildet werden sollen.

Definition 2.1 (Alphabet, Wort)
Ein *Alphabet* Σ ist eine endliche Menge von Symbolen. Ein *Wort* über Σ ist eine endliche Folge $a_1 \ldots a_n$ von Symbolen des Alphabets $a_i \in \Sigma$. Die Menge aller Wörter über Σ wird mit Σ^* bezeichnet. Das *leere Wort* ϵ ist das Wort der Länge $n = 0$.

* * *

Die folgende Definition beschreibt rekursiv, wie ein regulärer Ausdruck aufgebaut wird.

Definition 2.2 (Regulärer Ausdruck)
Es sei Σ ein Alphabet. Ein *regulärer Ausdruck* beschreibt eine Menge von Wörtern über Σ. Die regulären Ausdrücke $\emptyset$, ϵ und $a \in \Sigma^*$ beschreiben jeweils die leere Menge $\emptyset$, die Menge mit dem leeren Wort $\{\epsilon\}$ sowie die Menge mit einem Wort $\{a\}$. Sind α und β reguläre Ausdrücke, die Mengen L_1 und L_2 beschreiben, so erhält man folgendermaßen weitere reguläre Ausdrücke und zugeordnete Mengen:

Ausdruck	Menge	Bedeutung
$\alpha\|\beta$	$L_1 \cup L_2$	Alternative
$\alpha\beta$	$L_1 \cdot L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$	Konkatenation
α^+	$L_1^+ = \{w_1 \cdot \ldots \cdot w_k \mid w_i \in L_1 \text{ für } 1 \leq i \leq k;\ k \in \mathbb{N}\}$	ein- oder mehrmals
α^*	$L_1^* = \{w_1 \cdot \ldots \cdot w_k \mid w_i \in L_1 \text{ für } 1 \leq i \leq k;\ k \in \mathbb{N}_0\}$	kein-, ein- oder mehrmals
$[\alpha]$	$L_1 \cup \{\epsilon\}$	kein- oder einmal

wobei $\alpha|\beta$ die Alternative zwischen den regulären Ausdrücken α und β und $\alpha\beta$ das Hintereinanderschreiben von α und β ausdrückt. Der Ausdruck α^+ beschreibt die Menge der Wörter, die durch ein- oder mehrmalige Wiederholung von α gebildet werden. Nimmt man zu α^+ das leere Wort ϵ hinzu, so erhält man α^*. Der Ausdruck $[\alpha]$ steht für die Menge der von α erzeugten Wörter und das leere Wort ϵ.

* * *

Beispiel 2.3 (Regulärer Ausdruck)
Gegeben sei das Alphabet $\Sigma = \{a, b\}$.

Der reguläre Ausdruck $[a[ba]^*] = \{\epsilon, a, aba, ababa, abababa, \ldots\}$ beschreibt alle Wörter, in denen aufeinanderfolgende Buchstaben a durch ein b getrennt werden. Wenn man statt b ein Komma wählt, erhält man alle Listen, mit dem Element a: $\epsilon/a/a, a/a, a, a/\ldots$. In Kapitel 4 werden Attributlisten auf diese Weise definiert.

Wenn man sich an der Unix-Schreibweise für reguläre Ausdrücke orientiert, kann man Alternativen statt durch einen senkrechten Strich | auch durch ein Komma verbinden. Ein Intervall einer geordneten Symbolmenge wird durch das kleinste und größte Element des gewählten Ausschnitts angegeben. Der reguläre Ausdruck $A - Z, a - z, 0 - 9$ beschreibt zum Beispiel alle Groß- und Kleinbuchstaben sowie alle Ziffern. Er wird benötigt, um die Grundmengen der Grammatiken in Kapitel 4 zu definieren.

* * *

2.1.2 Grammatiken

Alle Wörter, die durch einen regulären Ausdruck beschrieben werden, werden anhand derselben Vorschrift gebildet. Eine allgemeinere Möglichkeit, die Erzeugung von Wörtern zu spezifizieren, bieten *formale Grammatiken*. Sie wurden von Noam Chomsky [15] erfunden,

um in der Linguistik die Erzeugung natürlicher Sprachen zu formalisieren. Hier betrachten wir nur *formale Sprachen*, die sehr allgemein als beliebige Mengen von Wörtern über einem Alphabet definiert sind.

Definition 2.4 (Formale Sprache)
Es sei Σ ein Alphabet. Eine Teilmenge $L \subseteq \Sigma^*$ heißt *(formale) Sprache* über Σ.

Diese Definition sagt nichts darüber aus, ob man ein kompaktes Bildungsgesetz für eine formale Sprache angeben kann, das auch Informationen über die Struktur der Sprache liefert. Ein Bildungsgesetz für rekursiv aufzählbare Sprachen kann man immer angeben, indem man die Wörter der Sprache aufzählt. Eleganter ist es, die Erzeugung einer Sprache durch eine Grammatik zu beschreiben.

Definition 2.5 (Formale Grammatik)
Eine *(formale) Grammatik* ist ein Tupel $G = (N, T, P, S)$, das aus endlichen Mengen von *Nichtterminalsymbolen* N (auch: Variablen), *Terminalsymbolen* T mit $N \cap T = \emptyset$ (auch: Terminalalphabet) und *Produktionen* $P \subset (N \cup T)^+ \times (N \cup T)^*$ (auch: Ersetzungs-, Ableitungsregeln) sowie einem *Startsymbol* $S \in N$ (auch: Axiom) besteht.

Eine Produktion $(\alpha, \beta) \in (N \cup T)^+ \times (N \cup T)^*$ wird im allgemeinen notiert als $\alpha \rightarrow \beta$. Enthält ein Ausdruck aus Terminal- und Nichtterminalsymbolen einen Teilausdruck der Form α, so kann man diesen durch β ersetzen. Man spricht von einem *Ableitungsschritt* $\alpha \Rightarrow \beta$. Mehrere aufeinanderfolgende Ableitungen bilden eine *Ableitungskette*. Entsteht dabei aus dem Ausdruck α der Ausdruck β, so schreibt man $\alpha \overset{*}{\Rightarrow} \beta$.

Die Sprache L(G), die von einer Grammatik erzeugt wird, besteht aus allen Wörtern über dem Terminalalphabet, die sich ausgehend vom Startsymbol durch Anwendung von Produktionen ableiten lassen: $L(G) = \{w \in T^* \mid S \overset{*}{\Rightarrow} w\}$.

* * *

Beispiel 2.6 (Formale Grammatik)
Hierarchische Strukturen wie Bäume können durch korrekt geklammerte Zeichenketten spezifiziert werden. Korrekt geklammert bedeutet, dass die Zeichenkette genausoviele öffnende wie schließende Zeichenfolgen enthält und dass von links kommend die Anzahl der öffnenden Klammern stets größer oder gleich der Anzahl der schließenden Klammern ist. Der Ausdruck (()(()())) ist also korrekt geklammert, (()(()()) und ()))(() hingegen nicht.

Ein Grammatik, die alle korrekt geklammerten Klammerketten erzeugt, ist gegeben durch $G = (N, T, P, S)$ mit Nichtterminalsymbol $N = \{S\}$, Terminalalphabet $T = \{(,)\}$ und einer Produktion $P = \{S \rightarrow (S) \mid SS \mid \epsilon\}$.

Da Klammern von der Produktion immer paarweise erzeugt werden, ist klar, dass nur korrekt geklammerte Zeichenketten erzeugt werden können. Es wird auch jede korrekt geklammerte Zeichenkette von der Grammatik generiert, da man bei einer korrekt geklammerten Zeichenkette von den innersten Klammerpaaren ausgehend die Produktion rückwärts anwenden kann, um die Klammern verschwinden zu lassen, bis nur noch das Startsymbol übrig ist.

Da jeder öffnenden Klammer weiter rechts eine schließende Klammer eindeutig zugeordnet ist, muss sich jeder korrekt geklammerte Ausdruck so auflösen (*parsen*) lassen.

Parsen bedeutet, dass für eine vorgegebene Zeichenkette eine gültige Ableitung gesucht wird, d.h es wird versucht die Regeln der Grammatik rückwärts anzuwenden. Wird eine Ableitung gefunden, so ist die Zeichenkette ein Wort der Sprache, die durch die Grammatik erzeugt wird.

Der korrekt geklammerte Ausdruck (()(()())) wird folgendermaßen geparst, indem in jeder Zeile die rechts angegebene Ersetzungsregel auf das am weitesten links stehende Nichtterminalsymbol der rechten Seite angewendet wird.

			Ersetzungsregel
(()(()()))	$\leftarrow$	$((S)(()()))$	$S \to \epsilon$
	$\leftarrow$	$((S)((S)()))$	$S \to \epsilon$
	$\leftarrow$	$((S)((S)(S)))$	$S \to \epsilon$
	$\leftarrow$	$(S(S(S)))$	$S \to (S)$
	$\leftarrow$	$(S(SS))$	$S \to (S)$
	$\leftarrow$	$(S(S))$	$S \to SS$
	$\leftarrow$	(SS)	$S \to (S)$
	$\leftarrow$	(S)	$S \to SS$
	$\leftarrow$	S	$S \to (S)$

Die Grammatik in Beispiel 2.6 erzeugt die Sprache der korrekten Klammerketten. Ein anderes Beispiel ist die Sprache, die durch den regulären Ausdruck $a^+[c]|b^+$ für das Alphabet $\Sigma = \{a, b, c\}$ erzeugt wird. Da ein zusammengehöriges Klammerpaar in einer korrekten Klammerkette beliebig weit auseinanderliegen darf, während die Wörter der zweiten Sprache durch 'lokale' Entscheidungen erzeugbar zu sein scheinen, gewinnt man den Eindruck, dass die Sprache der korrekten Klammerketten mächtiger sein könnte als die andere Sprache. Dies wirft die Frage auf, ob man formale Sprachen in Klassen unterschiedlicher Ausdruckskraft aufteilen kann.

In der Tat kann man die Gestalt der Produktionen einer Grammatik dazu nutzen, die erzeugten Sprachen zu klassifizieren.

Definition 2.7 (Chomsky-Hierarchie für Grammatiken)
Es sei $G = (N, T, P, S)$ eine Grammatik.

(a) **Typ 0**: Eine *Typ-0-Grammatik* hat nur Produktionen der Form

$$\alpha \to \beta$$

wobei $\alpha \in N^+$ und $\beta \in (N \cup T)^*$, d.h. $P \subset N^+ \times (N \cup T)^*$.

Typ-0-Grammatiken erzeugen (wie die allgemeinen Grammatiken aus Definition 2.5 [29, S.42]) die rekursiv aufzählbaren Sprachen [86].

(b) **Typ 1**: Eine *kontextsensitive* Grammatik hat Produktionen der Form

$$\beta A \gamma \rightarrow \beta \alpha \gamma$$

für ein Nichtterminalsymbol $A \in N$ und Ausdrücke $\alpha \in (N \cup T)^+$, $\beta, \gamma \in (N \cup T)^*$. Falls S nicht auf einer rechten Seite einer Produktion auftritt, darf zusätzlich eine Produktion der Form $S \rightarrow \epsilon$ existieren.

Im Kontext von β und γ darf das Nichtterminalsymbol A also durch α ersetzt werden. Eine kontextsensitive Grammatik erzeugt eine *kontextsensitive Sprache*.

(c) **Typ 2**: Eine *kontextfreie* Grammatik hat Produktionen der Form

$$A \rightarrow \alpha$$

für ein Nichtterminalsymbol $A \in N$ und einen Ausdruck $\alpha \in (N \cup T)^*$, d.h. $P \subset N \times (N \cup T)^*$. Auf der linken Seite einer Produktion steht nur ein Nichtterminalsymbol. Eine kontextfreie Grammatik erzeugt eine *kontextfreie Sprache.*

(d) **Typ 3**: Eine *rechtslineare* Grammatik hat nur Produktionen der Form

$$A \rightarrow a \mid A \rightarrow aB$$

für Nichtterminalsymbole $A, B \in N$ und ein Terminalsymbol $a \in T$, d.h. $P \subset N \times T(N \cup \{\epsilon\})$. Eine *linkslineare* Grammatik hat nur Produktionen der Form

$$A \rightarrow a \mid A \rightarrow Ba$$

für Nichtterminalsymbole $A, B \in N$ und ein Terminalsymbol $a \in T$, d.h. $P \subset N \times (N \cup \{\epsilon\})T$. Eine Grammatik heißt *Typ-3-Grammatik*, wenn sie rechts- oder linkslinear ist.

Rechts- und linkslineare Grammatiken erzeugen genau die durch reguläre Ausdrücke beschriebenen Sprachen. Für die Erzeugung von Bäumen sind rechts- und linkslineare Grammatiken nicht geeignet, da sie keine hierarchischen Strukturen generieren können.

* * *

Die Mengen der Sprachen, die von Grammatiken der Chomsky-Hierarchie erzeugt werden, sind ineinander enthalten: Typ-3-Sprachen $\subseteq$ Typ-2-Sprachen $\subset$ Typ-1-Sprachen $\subset$ Typ-0-Sprachen [86, Satz 5.15].

Die Sprache der korrekten Klammerketten ist kontextfrei, aber nicht regulär. Wären sie regulär, so könnte man ihre Wörter mit einer rechtslinearen Grammatik z.B. von links nach rechts erzeugen. Bei der Ableitung eines einer Klammerkette müßte man die Anzahl der bisher erzeugten öffnenden Klammern 'zählen', um später die richtige Anzahl schließender Klammern ergänzen zu können. Bei einer beliebig tief schachtelbaren Klammerung ist dies aber mit einer endlichen Anzahl von Nichtterminalsymbolen nicht möglich, da man für jede Schachtelungsstufe ein anderes Nichtterminalsymbol bräuchte.

Die Sprache in Beispiel 2.9 ist rechtslinear, also regulär. Die in dieser Arbeit eingeführten Grammatiken sind kontextfrei, aber i.a. nicht regulär.

Da in Kapitel 5 Funktionen auf Teilmengen von Sprachen definiert werden, die durch eine kontextfreie Grammatik erzeugt wurden, stellt sich die Frage, ob sich eine solche Sprache in natürlicher Weise zerlegen lässt.

Definition 2.8 (Teilsprache, Teilsprachendiagramm)

(a) Es sei Σ ein Alphabet und $L \subseteq \Sigma^*$ eine Sprache über Σ. Eine *Teilsprache* L' von L ist eine Sprache über Σ, die in L enthalten ist: $L' \subseteq L$.

(b) Es sei $G = (N, T, P, S)$ eine kontextfreie Grammatik. Für ein Nichtterminalsymbol $X \in N$ ist die von X erzeugte *Teilsprache* $L_G(X)$ in der Grammatik G definiert durch

$$L_G(X) = \{w \in L(G) \mid S \stackrel{*}{\Rightarrow} X \stackrel{*}{\Rightarrow} w)\}.$$

Die Menge $\{L_G(X) \mid X \in N\}$ aller Teilsprachen einer Grammatik wird mit $\mathcal{L}_G$ bezeichnet. Der Index G wird auch weggelassen, wenn die verwendete Grammatik aus dem Kontext ersichtlich ist. Das *Teilsprachendiagramm* von G ist definiert als die Relation

$$\mathcal{D}_L \subseteq N \times N \text{ mit } (X, Y) \in \mathcal{D}_L \iff L(X) \subseteq L(Y).$$

Wenn die Teilmengenrelationen eines Teilsprachendiagramms durch einen gerichteten Graphen dargestellt werden, werden die transitiven Kanten der Übersichtlichkeit halber weggelassen.

* * *

Die von einer Grammatik erzeugte Sprache lässt sich in Teilsprachen zerlegen, wenn man Nichtterminalsymbole der eigentlichen Erzeugung von Zeichenketten vorschaltet.

Beispiel 2.9 (Teilsprache)
Gegeben sei die Grammatik $G = (N, T, P, S)$ mit Nichtterminalsymbol $N = \{S, A, B, C\}$, Terminalalphabet $T = \{a, b, c\}$ und Produktionen

$$P = \left\{ \begin{array}{lcl} S & \to & A \mid B \\ A & \to & aA \mid aC \mid a \\ B & \to & bB \mid b \\ C & \to & c \end{array} \right\}$$

Die Grammatik erzeugt die reguläre Sprache $L(G) = L_G(S) = \{a\}^+ \cup \{a\}^+c \cup \{b\}^+$. Die erste Produktion spielt die Rolle einer Verzweigung zu den Teilsprachen $L_G(A)$ und $L_G(B)$. Die von den Nichtterminalsymbolen erzeugten Teilsprachen sind $L_G(A) = \{a\}^+ \cup \{a\}^+c$, $L_G(B) = \{b\}^+$ und $L_G(C) = \emptyset$. Das Beispiel zeigt, dass nicht jedes Nichtterminal eine nichttriviale Teilsprache induziert. In diesem Beispiel ist $L_G(C)$ leer, weil es keine Ableitungskette $S \stackrel{*}{\Rightarrow} C$ gibt. Man kann zwar die Menge aller Wörter $\{c\}$ bestimmen, die von C abgeleitet

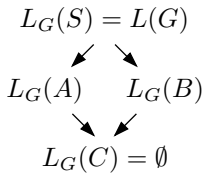


Abbildung 2.1: Teilsprachendiagramm für Beispiel 2.9

werden können, doch ist sie keine Teilsprache von $L(G)$. Eine Teilsprache $L_G(X)$ ist per Definition immer in der gesamten von G erzeugten Sprache $L(G)$ enthalten. Das Teilsprachendiagramm in Abbildung 2.1 gibt die Inklusionen zwischen den Teilsprachen graphisch wieder.

* * *

Weiterführende Informationen zum Thema Grammatiken findet man in [33] und [86].

2.1.3 Graphen

Graphen spielen sowohl bei der Repräsentation als auch bei der Transformation musikalischer Strukturen eine zentrale Rolle. Den musikalischen Objekten in Kapitel 4 liegen gerichtete Bäume zugrunde. Operatorgraphen in Kapitel 5 sind gerichtete, azyklische Graphen, deren Knoten Operatoren enthalten. Die folgenden Definitionen stützen sich auf [9] und [85].

Definition 2.10 (Graph)

(a) Ein *(ungerichteter) Graph* ist ein Tripel $G = (V, E, \psi)$ mit einer Menge von *Knoten* V, einer Menge von *Kanten* E mit $V \cap E = \emptyset$ und einer *Inzidenzfunktion* $\psi : E \to \{V' \subseteq V \mid 1 \leq |V'| \leq 2\}$, die jeder Kante aus E ein ungeordnetes Knotenpaar zuordnet.

Zwei Knoten, die durch eine Kante verbunden sind, heißen *adjazent*. Zwei Kanten, die mit einem gemeinsamen Knoten verbunden sind, heißen *inzident*. Der *Grad* eines Knotens ist definiert als die Anzahl der Kanten, die mit dem Knoten inzident sind.

Ein Graph $G' = (V', E', \psi')$ ist ein *Teilgraph* eines Graphen $G = (V, E, \psi)$, wenn $V' \subseteq V$, $E' \subseteq E$ und $\psi'(e) = \psi(e)$ für alle $e \in E'$.

Ein Graph ist *zusammenhängend*, wenn je zwei Knoten durch eine Folge inzidenter Kanten verbunden sind. Ein *Zyklus* oder *Kreis* ist ein zusammenhängender Graph mit Knotengrad 2.

(b) Ein *gerichteter Graph* oder *Digraph* (engl. *directed graph*) ist ein Tripel $G = (V, E, \psi)$ mit einer Menge von *Knoten* V, einer Menge von *gerichteten Kanten* E mit $V \cap E = \emptyset$ und einer *Inzidenzfunktion* $\psi : E \to V \times V$, die jeder Kante aus E ein geordnetes Knotenpaar zuordnet. Lässt man die Ordnung der Knotenpaare außer Acht, so erhält man den *untergeordneten ungerichteten Graphen*.

Sind zwei Knoten v, w durch eine gerichtete Kante vw verbunden, so heißt der Knoten v *Vorgänger* von w. Der Knoten w heißt *Nachfolger* von v. Der *Innengrad* eines Knotens

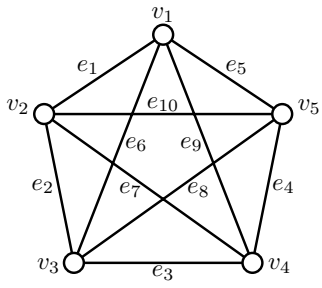


Abbildung 2.2:
Der vollständige Graph K_5

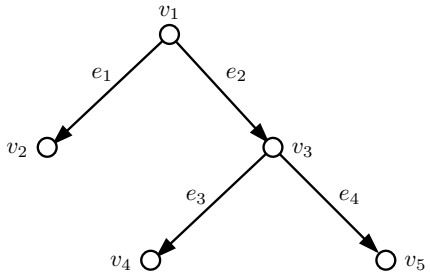


Abbildung 2.3:
Ein orientierter Wurzelbaum

Abbildung 2.4:
Schlingen und
Mehrfachkanten

ist die Anzahl der gerichteten Kanten, die in den Knoten einlaufen. Der *Außengrad* eines Knotens ist die Anzahl der Kanten, die aus dem Knoten herauslaufen.

Eine Kantenfolge $e_1, \ldots, e_n \in E$ für $n \geq 1$ heißt *gerichteter Pfad*, wenn es Knoten $v_1, \ldots, v_{n+1}$ gibt mit der Eigenschaft $\psi(e_i) = (v_i, v_{i+1})$, $1 \leq i \leq n$.

Ein *gerichteter Kreis* ist ein zusammenhängender Digraph, dessen Knoten Innen- und Außengrad 1 haben.

* * *

Beispiel 2.11 (Graph)
Abbildung 2.2 zeigt den vollständigen Graphen mit fünf Knoten K_5. Der Graph hat die Knoten $V = \{v_1, v_2, v_3, v_4, v_5\}$, die Kanten $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$ und die Inzidenzfunktion $\psi(e_1) = (v_1, v_2)$, $\psi(e_2) = (v_2, v_3)$, $\psi(e_3) = (v_3, v_4)$, $\psi(e_4) = (v_4, v_5)$, $\psi(e_5) = (v_5, v_1)$, $\psi(e_6) = (v_1, v_3)$, $\psi(e_7) = (v_2, v_4)$, $\psi(e_8) = (v_3, v_5)$, $\psi(e_9) = (v_4, v_1)$, $\psi(e_{10}) = (v_5, v_2)$.

Der Digraph in Abbildung 2.3 ist ein gerichteter Wurzelbaum mit den Knoten $V = \{v_1, v_2, v_3, v_4, v_5\}$ und den Kanten $E = \{e_1, e_2, e_3, e_4\}$. Die Lage der Kanten ist durch die Inzidenzfunktion $\psi(e_1) = (v_1, v_2)$, $\psi(e_2) = (v_1, v_3)$, $\psi(e_3) = (v_3, v_4)$, $\psi(e_4) = (v_3, v_5)$ definiert.

Das Ergebnis der Inzidenzfunktion wird bei ungerichteten Graphen als ungerichtete Kante und bei gerichteten Graphen als gerichtete Kante interpretiert. Meistens wird die Inzidenzfunktion eines Graphen aber nicht explizit angegeben, sondern wie in den Abbildungen 2.2 und 2.3 graphisch dargestellt.

Ein Graph kann auch *Schlingen*, also Kanten von einem Knoten zu sich selbst oder *Mehrfachkanten* besitzen (vgl. Abbildung 2.4). Solche Graphen kommen im weiteren Verlauf dieser Arbeit nicht vor.

* * *

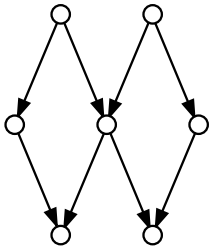


Abbildung 2.5: Ein azyklischer Digraph, der kein orientierter Wurzelbaum ist.

Definition 2.12 (Baum, orientierter Wurzelbaum, azyklischer Digraph)

(a) Ein *Baum* ist ein zusammenhängender Graph, der keinen Kreis besitzt. Ein Knoten mit Grad ≥ 2 heißt *innerer Knoten*; ein Knoten mit Grad 1 heißt *Blatt*.

(b) Ein *orientierter Wurzelbaum* ist ein Digraph, dessen zugrundeliegender ungerichteter Graph ein Baum ist und der einen Knoten besitzt, von dem aus alle anderen Knoten auf orientierten Pfaden erreichbar sind. Dieser Knoten heißt *Wurzel*. Ein Knoten mit Außengrad > 0 heißt *innerer Knoten*; ein Knoten mit Außengrad 0 heißt *Blatt*.

Ein *azyklischer Digraph* ist ein Digraph, der keinen gerichteten Kreis besitzt. Ein Knoten mit Innengrad 0 heißt *Quelle*. Ein Knoten mit Außengrad 0 heißt *Senke*.

* * *

Beispiel 2.13 (Wurzelbaum und ayzklischer Digraph)
Eine Wurzel in einem orientierten Wurzelbaum hat Innengrad 0, da es andernfalls einen orientierten Weg zum Anfangsknoten einer einlaufenden Kante der Wurzel geben müßte, der zusammen mit der einlaufenden Kante einen (verbotenen) Kreis bilden würde. Weiterhin gibt es in einem orientierten Wurzelbaum nur eine Wurzel, da eine zweite Wurzel, die wie gerade gesehen Innengrad 0 hat, im Widerspruch zur Definition von der ersten Wurzel aus nicht erreichbar wäre.

Ein azyklischer Digraph ist nicht notwendigerweise ein orientierter Wurzelbaum. Während Abbildung 2.3 einen gerichteten Baum mit einer Wurzel und drei Blättern darstellt, sieht man in Abbildung 2.5 einen azyklischen Digraphen, der kein gerichteter Baum ist. Er besitzt zwei Quellen und zwei Senken.

Die musikalischen Objekte, die von einer Erweiterbaren Zeitreihengrammatik (Definition 4.10) generiert werden, sind orientierte Wurzelbäume. Abbildung 4.16 zeigt ein Beispiel für ein musikalisches Objekt. Die Kanten sind implizit von oben nach unten orientiert.

Operatorgraphen (Definition 5.8) sind azyklische Digraphen, bei denen die Orientierung der Kanten die Durchflussrichtung der bearbeiteten Information angibt. Die Abbildungen 5.7 und 5.19 zeigen Beispiele für Operatorgraphen.

* * *

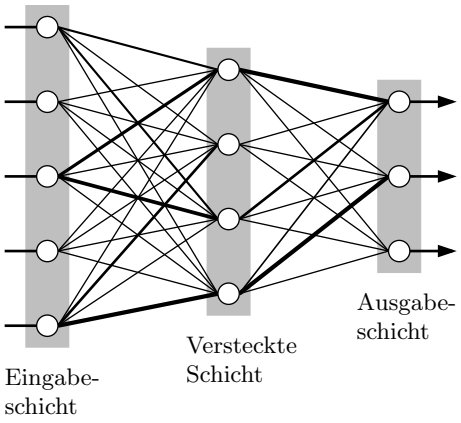


Abbildung 2.6: Ein neuronales Netz mit einer Eingabeschicht, einer versteckten Schicht und einer Ausgabeschicht. Jede Schicht besteht aus Neuronen, die mit gewichteten Verbindungen verbunden sind.

2.1.4 Neuronale Netze

Klassifikations- und Regressionsprobleme. Neuronale Netze sind lernbasierte Verfahren, die vorwiegend zur Lösung von Klassifikations- und Regressionsproblemen eingesetzt werden. Bei einem *Regressionsproblem* soll eine reellwertige Funktion möglichst gut durch ein Modell approximiert werden. Bei einem *Klassifikationsproblem* ist das Ziel, eine Funktion mit einem endlichen Wertebereich zu lernen, dessen Werte als Zuordnung der Funktionseingabe zu Klassen interpretiert werden. Wenn man zur Lösung dieser Aufgaben ein lernbasiertes Modell einsetzt, werden seine Parameter mit Hilfe einer endlichen Menge von Trainingsbeispielen eingestellt. Durch die Wahl der Parameter wird das Verhalten des Modells festgelegt.

In dieser Arbeit werden neuronale Netze ausschließlich zum Lösen von Klassifikationsproblemen eingesetzt (Kapitel 6). Aufbauend auf musikalischen Originalbeispielen werden musikalische Eingabemerkmale einem Ausgabemerkmal zugeordnet. Auf diese Weise wird ausgedrückt, in welchem musikalischen Kontext ein Ausgabemerkmal im untersuchten Musikstil auftritt.

Neuronale Netze. Künstliche neuronale Netze sind ein stark vereinfachtes Modell des Gehirns. Die Funktionsweise ihrer Berechnungselemente, der *Neuronen*, orientiert sich an den Nervenzellen des Gehirns. Wenn die Information an den Eingängen eines Neurons einen Schwellwert überschreitet, feuert das Neuron, d.h. die Eingangsinformation wird mit Hilfe einer *Aktivierungsfunktion* auf einen zulässigen Wert abgebildet, der durch *gewichtete Verbindungen* an die nachfolgenden Neuronen übermittelt wird. Die Gewichte eines neuronalen Netzes regulieren, wieviel Information vom Ausgang eines Neurons zum Eingang des nächsten Neurons gelangt. In den Gewichten ist das Wissen des neuronalen Netzes verteilt gespeichert. Das bedeutet, dass einem bestimmten Gewicht keine feste Bedeutung zugeordnet werden kann, sondern das Wissen in sehr kleinen Portionen über die Gewichte des Netzes

verstreut ist. Die Robustheit neuronaler Netze gegenüber dem Ausfallen einzelner Neuronen (*Fehlertoleranz*) ist eine Folge dieser verteilten Speicherung von Information.

Eine häufig verwendete Netzstruktur ist das *Feed-Forward-Netz*. Die Neuronen sind in Schichten angeordnet, wobei eine Schicht vollständig mit der nächsten Schicht verbunden ist (Abbildung 2.6). Nachdem ein Muster an die Eingangsschicht angelegt wurde, wird berechnet, mit welcher Aktivität die Neuronen dieser Schicht feuern. Die Ausgabewerte der ersten Schicht werden gewichtet und als Eingabe an die nächste Schicht übermittelt, deren Ausgabe nun berechnet und weitergeleitet wird. Wenn alle Schichten sukzessive ausgewertet worden sind, kann das Ergebnis an den Ausgängen der Ausgabeschicht abgelesen werden.

Bei einem Klassifikationsproblem mit zwei Klassen verwendet man nur ein Ausgabeneuron, das Werte in einem festen Intervall, z.B. $[0, 1]$ annimmt. Die Ausgabeaktivierung des Neurons wird als Zugehörigkeit zu einer der beiden Klassen interpretiert, je nachdem, ob der Wert kleiner oder größer als 0.5 ist. Bei einem Klassifikationsproblem mit mehr als zwei Klassen ordnet man jedem Neuron der Ausgabeschicht eine Klasse zu. Wenn man für ein Eingabemuster die Ausgabeaktivierungen des Netzes berechnet hat, wird die Position des Ausgabeneurons mit der größten Aktivierung als vom Netz gewählte Klasse interpretiert. Haben mehrere Neuronen die maximale Aktivierung, so wird heuristisch eine der zugehörigen Klassen ausgewählt, beispielsweise diejenige, die in den Daten die höchste a-priori-Wahrscheinlichkeit besitzt.

Lernen mit Backpropagation. Neuronale Netze besitzen die Fähigkeit, aus Beispielen zu lernen und diese zu verallgemeinern. Lernen bedeutet, dass die Gewichte der Verbindungen zwischen den Neuronen so eingestellt werden, dass das Netz zu den Eingabemustern einer Trainingsmenge die gewünschten Ausgabemuster möglichst genau berechnet. Ob das Netz die Trainingsbeispiele verallgemeinern kann, wird anhand einer Testmenge von Beispielen überprüft, die nicht zum Lernen verwendet wurde. Ein Standard-Lernverfahren für Feed-Forward-Netze ist der Backpropagation-Algorithmus, dessen Funktionsweise anhand des folgenden Beispiels erläutert werden soll. Detaillierte Informationen zu verschiedenen Lernverfahren finden sich in [8] und [74].

Beispiel XOR-Problem. Ein neuronales Netz soll erkennen, ob sich zwei binäre Eingabewerte voneinander unterscheiden. Die Menge der möglichen Eingabevektoren umfasst (0,0), (0,1), (1,0) und (1,1). Die Ausgabe soll 1 sein, wenn sich die Eingabewerte voneinander unterscheiden, andernfalls 0.

Das Training des Netzes geschieht folgendermaßen: Am Anfang werden die Gewichte zufällig initialisiert. Als nächstes wird für alle Beispiele aus einer Trainigsmenge die Netzausgabe berechnet und mit der Sollausgabe verglichen. Aus der mittleren quadrierten Differenz zwischen Netzausgabe und Sollausgabe auf der Trainingsmenge erhält man einen Fehlerwert, der zur Anpassung der Netzgewichte verwendet wird (Abbildung 2.7). Betrachtet man den Fehler als Funktion, die von den Netzgewichten abhängt, so besteht die Lernaufgabe darin, eine Belegung der Gewichte zu finden, die den Fehlerwert auf der Trainingsmenge minimiert. Mit Hilfe eines *Gradientenabstiegsverfahrens* werden die Gewichte entgegen der Richtung des Gradienten der Fehlerfunktion angepasst, indem die Änderung des Fehlers schichtweise rückwärts von der Ausgabeschicht bis zur Eingabeschicht auf die Gewichte verteilt wird.

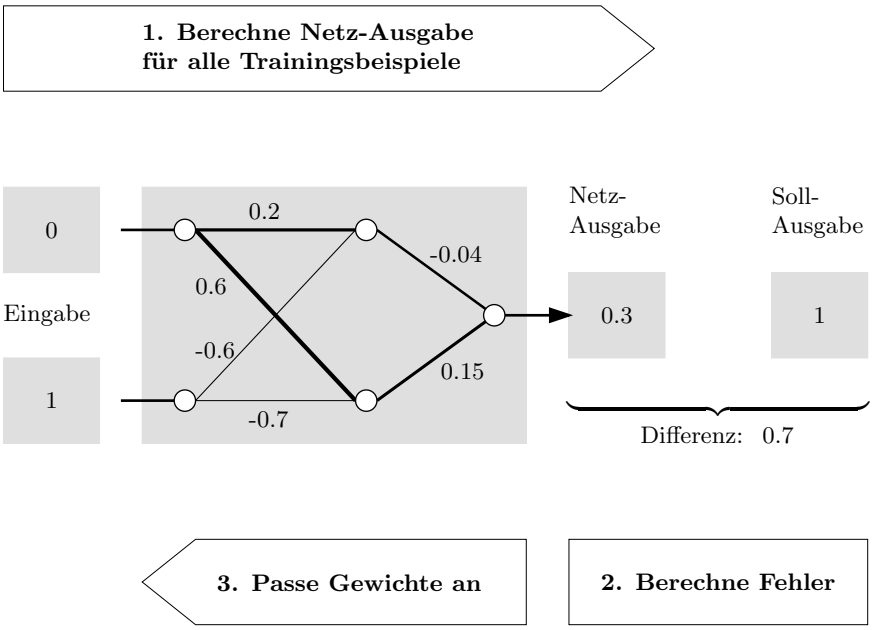


Abbildung 2.7: Backpropagation-Algorithmus am Beispiel des XOR-Problems. Trainingsbeispiele (Eingabe, Eingabe; Sollausgabe): $(0, 0; 0), (0, 1; 1), (1, 0; 1), (1, 1; 0)$

Diesem Anpassungsprozess verdankt der Backpropagation-Algorithmus seinen Namen. Die Berechnung des Fehlers und die Anpassung der Gewichte werden solange wiederholt, bis der Fehler hinreichend klein ist. In dieser Arbeit wird das Lernverfahren RPROP eingesetzt, eine Weiterentwicklung der Backpropagation-Algorithmus mit beschleunigtem Gradientenabstieg [72].

Generalisierung. Beim Training neuronaler Netze interessiert man sich nicht nur für die Klassifikationsleistung eines Netzes auf den Trainingsdaten, sondern vor allem dafür, inwieweit ein Netz sich auch bei unbekannten Beispielen sinnvoll verhält. Nach dem Training testet man die Klassifikationsgüte eines Netzes daher auf einer Testmenge, die nicht zum Trainieren benutzt wurde. Die *Generalisierungsleistung* eines Netzes kann anhand der Klassifikationsrate auf dieser Testmenge geschätzt werden. So kann man einschätzen, ob ein Netz eine Gesetzmäßigkeit erfasst hat, die den Daten zugrunde liegt, oder ob es lediglich die Trainingsdaten auswendig gelernt hat.

Overfitting. Das „Auswendiglernen“von Trainingsdaten kann vorkommen, wenn ein neuronales Netz im Verhältnis zur Anzahl der Trainingsdaten zu viele freie Gewichte besitzt, beispielsweise weil man zu viele Eingabemerkmale oder Neuronen in den versteckten Schichten verwendet. Ein überdimensioniertes Netz wird beim Training nicht gezwungen, eine kompakte Darstellung der Lernbeispiele zu finden und speichert stattdessen die Lernbeispiele direkt. Eine solche Überanpassung an die Trainingsdaten äußert sich darin, dass das neuronale Netz unbekannte Testdaten nicht einordnen kann. Sie werden häufiger falsch klassifiziert als die

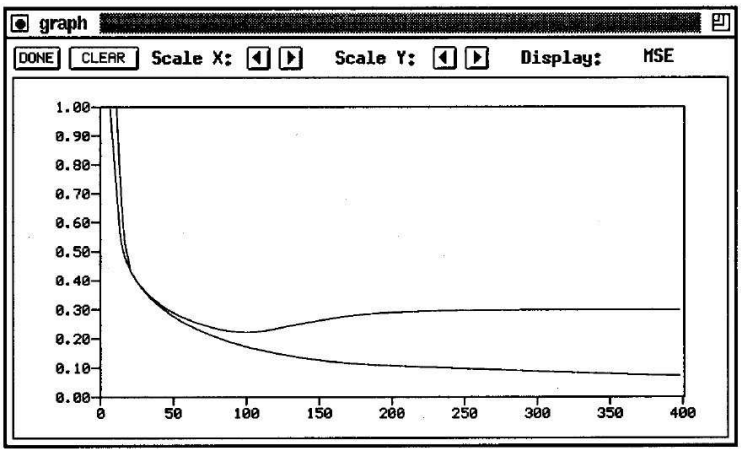


Abbildung 2.8: Typischer Verlauf von Trainingsfehler (untere Kurve) und Validierungsfehler (obere Kurve) beim Trainieren eines Netzes.

Trainingsdaten, und die Generalisierungsleistung des Netzes sinkt. Die Überanpassung eines Modells an eine Trainingsmenge heißt *Overfitting*.

Vermeidung von Overfitting. Es gibt verschiedene Möglichkeiten, dem Overfitting-Effekt entgegenzuwirken. Zum einen kann man die Anzahl der Netzgewichte verringern, indem man die versteckten Schichten verkleinert oder sie weglässt. Ein Netz ohne versteckte Schicht ist linear und kann nur lineare Zusammenhänge zwischen Ein- und Ausgabemerkmalen erfassen. Eine weitere Möglichkeit ist die Regularisierung der Netzgewichte. Dabei wird ein Strafterm zum Trainingsfehler addiert, der in das Lernverfahren eingeht. Der Strafterm ist proportional zur Summe des Quadrats aller Netzgewichte und bewirkt, dass kleine Netzgewichte bevorzugt werden. Dadurch wird es dem Netz erschwert, sich durch abrupte Richtungswechsel der gelernten Funktion besonders stark an einzelne Datenpunkte anzupassen. Der Einfluss des Strafterms wird durch Multiplikation mit einer Konstanten eingestellt. Wenn man die Netztopologie zu stark verkleinert oder die Netzgewichte zu stark regularisiert, ist ein neuronales Netz möglicherweise nicht mehr in der Lage, seine Aufgabe zu erfüllen.

Bei einer geeigneten Anzahl von Neuronen wird das in der Trainingsmenge enthaltene Wissen so komprimiert, dass es mit den vorhandenen Neuronen dargestellt werden kann. Diese Kompression führt zu einer Abstraktion von den Trainingsdaten und damit zu einer Verallgemeinerung der Beispiele. Eine geeignete Netztopologie kann z.B experimentell, mit Methoden der Informationstheorie oder mit evolutionären Algorithmen bestimmt werden. In dieser Arbeit wird mit Merkmalsselektionsverfahren aus einer Menge potentiell relevanter Merkmale eine geeignete Teilmenge ausgewählt. Auf diese Weise wird die Eingabeschicht und damit das gesamte neuronale Netz verkleinert.

Neben der Netztopologie und der Gewichtung des Regularisierungsterms beeinflusst auch die Trainingsdauer, wie stark die Gewichte des Netzes an die Beispiele in der Trainingsmenge angepasst werden. Auch bei einer zu langen Trainingsdauer kann es zu einer Überanpassung

der Netzgewichte an die Trainingsdaten kommen. Eine geeignete Trainingsdauer lässt sich bestimmen, indem während des Trainings nicht nur der mittlere Fehler auf der Trainingsmenge, sondern auch der mittlere Fehler auf einer unabhängigen sogenannten Validierungsmenge errechnet wird (Abbildung 2.8). Wenn das Lernverfahren wesentliches Wissen aus der Trainingsmenge extrahiert, wird sich auch der Validierungsfehler verringern. Wenn das Netz in die Phase der Überanpassung an die Trainingsmenge eintritt, wird der Validierungsfehler wieder ansteigen, da das Lernverfahren den Validierungsfehler bei der Veränderung der Gewichte nicht berücksichtigt. Das Abbrechen des Trainigs vor dem Anstieg des Validierungsfehlers ist als *Early Stopping* [8] bekannt.

Fehlerfunktion für Klassifikationsprobleme. Die Ausgabe eines neuronalen Netzes besteht aus einem reellwertigen Vektor, der je nach Lernaufgabe unterschiedlich interpretiert wird. Löst ein Netz eine Regressionsaufgabe, so wird die Netzausgabe als Funktionswert der gelernten Funktion interpretiert. Der Fehler der Netzausgabe ist der euklidische Abstand zwischen der tatsächlichen Netzausgabe und dem Zielwert, der durch die Lernbeispiele vorgegeben ist. Abbildung 2.7 zeigt diese Situation.

Wenn man eine Klassifikationsaufgabe löst, ist der Abstand zwischen Ist- und Sollausgabe eines Netzes für die Klassifikationsgüte unerheblich. Hier zählt allein die Entscheidung für die korrekte Klasse, unabhängig davon, wie knapp sie ausfällt. Bei Klassifikationsproblemen setzt man daher beim Anpassen der Netzgewichte die Kreuzentropie-Fehlerfunktion

$$-\sum_{n=1}^{N}\sum_{k=1}^{c} t_k^n \ln y_k(x^{(n)})$$

ein. Dabei bezeichnet N die Anzahl der Lernmuster, c die Anzahl der Ausgabeklassen, $x^{(n)}$ die Eingabemuster, t_n die Zielwerte und $y_k()$ die Ausgabe des k-ten Ausgabeneurons des Netzes. Die Ermittlung der vorhergesagten Klasse für ein einzelnes Eingabemuster $x^{(n)}$ als maximale Position der Netzausgabe $\text{maxarg}_k\{y_k(x^{(n)}) \mid 1 \leq k \leq c\}$ ändert sich durch die Verwendung der Kreuzentropie-Fehlerfunktion nicht.

Die Kreuzentropie-Fehlerfunktion misst den Abstand zwischen Wahrscheinlichkeitsverteilungen [21, S.318] und nimmt ihr Minimum an, wenn die Netzausgabe und der Zielwert gleich sind. Die wahrscheinlichkeitstheoretische Motivation der Kreuzentropie-Fehlerfunktion findet man in [8].

Kreuzvalidierung. Beim Lernen einer Klassifikationsaufgabe aus Beispielen geht man von der Annahme aus, dass die Beispiele von einer unbekannten Zufallsvariablen generiert wurden, die die wahre Verteilung der Beispiele widerspiegelt. Da man bei realen Problemen keinen Zugriff auf die wahre Verteilung der Beispiele hat, ist man darauf angewiesen, die Qualität eines Klassifikators anhand einer Testmenge zu schätzen, die nicht beim Training verwendet wurde. Die Zuverlässigkeit dieser Schätzung lässt sich verbessern, wenn man die verfügbaren Daten aufteilt und den Klassifikator mehrmals trainiert. Bei der n-fachen *Kreuzvalidierung* (engl. *cross validation*) teilt man die verfügbaren Lernbeispiele in n disjunkte Teilmengen auf ($n \in \mathbb{N}$). Der Klassifikator wird mit jeder Kombinationen von $n-1$ Teilmengen trainiert und auf der jeweils verbleibenden Teilmenge validiert. Der Mittelwert der Klassifikationsgüten auf den Validierungsmengen ist eine zuverlässigere Schätzung der

Generalisierungsleistung eines Klassifikators als das Trainieren mit einer Trainings- und Validierungsmenge [21]. Typischerweise wählt man für n bei der n-fachen Kreuzvalidierung Werte zwischen 3 und 10. Wenn man nur sehr wenige Daten zur Verfügung hat, kann man n auch als die Anzahl der Trainingsdaten wählen. In diesem Fall wird nur ein Datenpunkt zum Validieren zurückgehalten. Die Methode ist als "Leave-one-out Cross-Validation" bekannt. Da jeder Datenpunkt bei der Kreuzvalidierung sowohl in $n-1$ Trainingsmengen als auch in einer Validierungsmenge vorkommt, kann die Schätzung der Generalisierungsleistung zu optimistisch ausfallen. Daher sieht man eine weitere Testmenge vor, die während der Kreuzvalidierung nicht verwendet wird und eine bessere Schätzung der Generalisierungsleistung erlaubt.

Komitees. Bei der Kreuzvalidierung wird die Schätzung der Generalisierungleistung dadurch verbessert, dass mit den verschiedenen Trainingsmengen mehrere Klassifikatoren trainiert werden, die dasselbe Problem lösen. Die Klassifikatoren werden zu einem *Komitee* kombiniert, indem die Ausgabevektoren der Komiteemitglieder gemittelt werden und die gewählte Klasse wie zuvor aus der Position des maximalen Eintrag des gemittelten Vektors bestimmt wird. Auf diese Weise lässt sich die Generalisierungsleistung steigern, denn ein Komitee kann eine bessere Performanz haben als das beste seiner Mitglieder [8, S. 365]. In [8, S. 365] wird auch gezeigt, dass man die Leistung eines Komitees weiter verbessern kann, wenn man die Ausgaben der Komiteemitglieder mit Hilfe der Fehlerkorrelationsmatrix gewichtet, die die Korrelation zwischen den Fehlern der Komiteemitglieder wiedergibt.

Codierung der Ein- und Ausgabemerkmale. Um ein lernbasiertes Modell erfolgreich trainieren zu können, ist die Wahl einer geeigneten Codierung der Eingabedaten von großer Bedeutung. Eine Standardmethode bei der Vorverarbeitung der Trainingsdaten ist die *Skalierung* unterschiedlicher Merkmale, so dass die codierten Werte vergleichbarer Merkmale eine ähnliche Größenordnung haben. Wenn man etwa bei der lernbasierten Erkennung von Fischsorten Länge und Umfang in Millimetern bzw. Metern angibt, erhält die Länge der Fische dadurch zu Unrecht beim Lernen eine höheres Gewicht als der Umfang.

Eine weitere Vorverarbeitungsmethode ist die Standardisierung eines Merkmals (engl. *whitening*). Dabei werden die Daten so verschoben, dass ihr Mittelwert bei 0 liegt. Anschließend wird ihre Varianz auf 1 normiert.

Bei der Untersuchung musikalischer Daten ist es sinnvoll, musikalische Symbole so zu codieren, dass ein Klassifikator Zusammenhänge zwischen Merkmalen herstellen kann, zwischen denen tatsächlich eine inhaltliche Verbindung besteht. Ein Beispiel ist die *harmonische Toncodierung*, die in Abschnitt 5.2.3.3 ausführlich besprochen wird.

Mehr Informationen zum maschinellen Lernen findet man in [8, 60, 21, 18]

2.1.5 Merkmalsselektion

Merkmal. In der Informatik definiert man ein *Merkmal* als Eigenschaft einer Sache, einer Person oder einer Struktur, die durch Beobachtung, Messung oder Vorverarbeitung gewonnen werden kann. Merkmale eines Fischs sind z.B. seine Länge und sein Gewicht. Man unterscheidet zwischen der abstrakten Beschreibung eines Merkmals (Länge, Gewicht) und

den Werten, die das Merkmal annehmen kann (90cm, 8kg). Aus statistischer Sicht betrachtet ist ein Merkmal eine Zufallsvariable, die Werte aus einer vorgegebenen Menge – dem *Wertebereich* des Merkmals – annimmt [8, S.2]. Wenn keine Verwechslungsgefahr besteht, wird das Wort "Merkmal" häufig synonym für "Merkmalswert" verwendet.

Merkmalsselektion und maschinelles Lernen. Merkmalsselektionsverfahren zielen darauf ab, für eine vorgegebene Lernaufgabe eine möglichst kleine Menge relevanter Eingabemerkmale zu finden. Dies ist einerseits sinnvoll, weil kleine lernbasierte Modelle bei gleicher Anzahl von Trainingsbeispielen tendenziell robuster sind als größere Modelle. Gleichzeitig kann man durch die Merkmalsselektion redundante Eingabemerkmale herausfiltern. Eine Schwierigkeit bei der Merkmalsselektion für praktische Probleme bilden versteckte Beziehungen zwischen Eingabemerkmalen, z.B. in Form korrelierter Merkmale. Wenn man in einem Selektionsverfahren ein Merkmal entfernt, kann dies abhängig von den Wechselwirkungen zwischen den sonst vorhandenen Eingabemerkmalen zu unterschiedlichen Veränderungen des Optimierungskriteriums führen. Wenn man nicht alle Merkmalskombinationen durchsucht, findet man daher im allgemeinen nicht die globalen Optima für das Merkmalsselektionsproblem. Dennoch wird mit den hier verwendeten lokalen Suchverfahren bereits eine wesentliche qualitative Verbesserung des trainierten Modells erreicht.

Merkmalsselektionsverfahren werden in Kapitel 6 eingesetzt, um musikalische Merkmale zu finden, die für einen Musikstil relevant sind. Als Quelle für die in diesem Abschnitt beschriebenen Verfahren dienen [45], [48], [68] und [78].

Relevanz eines Merkmals. Die Relevanz eines Merkmals beschreibt die Bedeutung, die einem Merkmal in einem bestimmten Kontext zukommt. Wenn man z.B. nach Unterscheidungsmerkmalen zwischen musikalischen Stilen sucht, so sind die Merkmale relevant, die nicht für alle betrachteten Stile charakteristisch sind und daher zur Erkennung von Musikstilen verwendet werden können. Die Gesamtheit der betrachteten Stile stellt hier den Kontext dar, in dem die Relevanz musikalischer Merkmale untersucht wird.

Für das maschinelle Lernen werden die Begriffe der Relevanz und des Kontexts genauer spezifiziert. Es soll untersucht werden, ob die Eingabemerkmale eines Klassifikators es ermöglichen, durch das Training mit einer festen, ausreichend großen Menge von Beispielen, eine gute Lösung für eine Klassifikationsaufgabe zu finden. Da man für die Lösung irrelevante Merkmale herausfiltern möchte, sucht man nach einer minimalen Menge von Eingabemerkmalen, mit denen sich eine gute Lösung erzielen lässt. Hauptkriterium für die Bewertung der Relevanz eines Eingabemerkmals ist seine Wechselwirkung mit den weiteren Merkmalen bei der Lösung einer Lernaufgabe: Manche Merkmale kann man nie, andere kann man immer weglassen. Weitere Merkmale tragen zwar zur Lösung der Lernaufgabe bei, sind aber miteinander korreliert, so dass man teilweise auf sie verzichten kann.

Verfahren zur Merkmalsselektion. Es gibt verschiedene Ansätze zur Merkmalsselektion. Wenn man ein monotones Optimierungskriterium zur Verfügung hat, findet das Branch-and-Bound-Verfahren [63, 50] eine optimale Merkmalskombination mit exponentiellem Aufwand [48]. Ein solches Kriterium steht bei der Optimierung neuronaler Netze aber nicht zur Verfügung, da sowohl das Einfügen als auch das Entfernen von Eingabemerkmalen die Qualität eines neuronalen Netzes verbessern oder verschlechtern kann.

Viele der weiteren Merkmalsselektionsverfahren fallen in die Gruppen der Wrapper- und der Filteransätze [45]. Ziel beider Ansätze ist es, ein Lernmodell durch die Selektion geeigneter Eingabemerkmale zu optimieren. Bei *Wrapperverfahren* wird für Kombinationen von Eingabemerkmalen je ein Lernmodell trainiert, dessen als Bewertung der Merkmalskombination dient. Das Lernmodell, dessen Eingabemerkmale optimiert werden sollen, geht hier direkt in das Merkmalsselektionsverfahren mit ein. Bei *Filterverfahren* wird die Merkmalsauswahl unabhängig vom Lernmodell mit Hilfe informationstheoretischer Methoden optimiert [47]. Hier filtert die Merkmalsselektion Merkmale aus einer Menge von Kandidaten heraus, bevor ein Lernmodell trainiert wird. Wrapperverfahren sind aufwendiger als Filterverfahren, da zur Bewertung der einzelnen Suchpunkte ein lernbasiertes Modell trainiert werden muss. Sie sind jedoch besser in der Lage, die spezifischen Eigenschaften des verwendeten Lernmodells zu berücksichtigen [45]. Daher werden in dieser Arbeit Wrapperverfahren zur Merkmalsselektion eingesetzt.

Wrapperverfahren. Die folgende Definition eines Wrapperverfahrens stammt aus [45]. Gegeben sei eine geordnete Menge $\mathcal{X}$ von Merkmalen, aus der eine Teilmenge $\mathcal{X}' \subseteq \mathcal{X}$ mit maximaler Bewertung und minimaler Größe gewählt werden soll. Die Merkmale $\mathcal{X}$ stellen eine Vorauswahl potentiell relevanter Merkmale dar, in die der Anwender sein a-priori-Wissen über das Problemfeld einfließen lassen kann.

Eine Auswahl von Merkmalen aus $\mathcal{X}$ wird durch einen binären Vektor aus $\{0,1\}^{|\mathcal{X}|}$ repräsentiert. Dabei bedeutet eine 1, dass ein Merkmal als Eingabemerkmal eines Modells aktiv ist; eine 0 bedeutet, dass es inaktiv ist. Jede Kombination von Merkmalen aus $\mathcal{X}$ kann durch einen solchen Vektor dargestellt werden. Der *Suchraum* $\{0,1\}^{|\mathcal{X}|}$ für das Wrapperverfahren enthält damit $2^{|\mathcal{X}|}$ unterschiedliche Suchpunkte. Im folgenden werden der Vektor $(0,0,\ldots,0)$ als *leerer Zustand* oder *Nullzustand* und der Vektor $(1,1,\ldots,1)$ als *Einszustand* bezeichnet. Im Nullzustand sind alle Merkmale inaktiv, im Einszustand sind alle Merkmale aktiv.

Um sich durch den Suchraum zu bewegen, benötigt man eine Nachbarschaftsbeziehung zwischen den Suchpunkten. Hier wird die in binären Räumen gebräuchliche Hamming-Distanz verwendet. Der Hamming-Abstand zwischen zwei binären Vektoren gleicher Länge ist definiert als die Anzahl der unterschiedlichen Einträge der Vektoren. Direkt benachbart (Hamming-Distanz 1) sind z.B. zwei Vektoren, die sich nur an einer Position unterscheiden.

Jeder Suchpunkt wird mit Hilfe einer *Gütefunktion* bewertet. Beim Wrapperverfahren wird die Gütefunktion für einen Suchpunkt $x \in \{0,1\}^{|\mathcal{X}|}$ berechnet, indem man ein lernbasiertes Modell trainiert und testet, wie gut anhand der aktiven Merkmale von x das vorgegebene Ausgabemerkmal prognostiziert wird. Zum Trainieren und Testen des Modells werden immer die gleichen Datenmengen verwendet. Um die Schätzung der Testgüte des Modells zu verbessern, wird es mit Kreuzvalidierung trainiert.

Bei der Durchführung eines Wrapperverfahrens wählt man einen Zustand des Suchraums als *Anfangszustand*, bewertet ihn und setzt ihn als aktuellen Zustand. Mit einer der weiter unten beschriebenen *Suchstrategien* generiert man aus dem aktuellen Zustand Kandidaten für neue Suchpunkte und bewertet sie. Aus allen bewerteten Punkten wird ein bester Punkt ausgewählt, der als nächster aktueller Suchpunkt dient. Solange das *Abbruchkriterium* für die gewählte Suchstrategie nicht erfüllt ist, werden aus dem aktuellen Zustand neue Kandidaten generiert und bewertet.

Das Wrapperverfahren ist vom Typ der gewählten Klassifikators (Neuronales Netz, Nächster Nachbar, Support Vector Machine, ...) unabhängig; es wird lediglich eine Schnittstelle zu einem Klassifikator benötigt. Da die Klassifikatoren unterschiedliche Modelle realisieren, hängt das Ergebnis eines Selektionslaufs vom gewählten Klassifikator ab.

Suchstrategien. Eine Suchstrategie für das Wrapperverfahren wird festgelegt durch einen Anfangszustand, eine Vorschrift zur Bildung von Suchpunktkandidaten, eine Vorschrift zur Auswahl des nächsten Suchpunkts und ein Abbruchkriterium. Im Rahmen dieser Arbeit werden aus Laufzeitgründen nur die Vorwärts- und die Rückwärtssuche sowie für ein kleines Beispiel die vollständige Suche verwendet. Weitere Suchstrategien finden sich in [48, 68, 78].

Vollständige Suche. Die vollständige Suche bewertet alle Zustände des Suchraums und wählt diejenigen mit maximaler Bewertung aus.

	Vollständige Suche (`ExS`, engl. *exhaustive search*)
Anfangszustand:	Nullzustand
Suchpunktkandidaten:	Lexikographischer Nachfolger
Nächster Suchpunkt:	Lexikographischer Nachfolger
Abbruchkriterium:	Einszustand erreicht

Die vollständige Suche findet die globalen Maxima für das Merkmalsselektionsproblem. Sie ist jedoch für große Merkmalsmengen im allgemeinen zu aufwendig, da bei n Merkmalen 2^n Zustände ausgewertet werden müssen. Für jede Bewertung muss in einem Wrapperverfahren ein Klassifikator trainiert werden, so dass die Wahl der Modellparameter (z.B. Wahl des Lernverfahrens, Anzahl der trainierten Epochen, Topologie eines neuronalen Netzes, Anzahl der Kreuzvalidierungsmengen) den absoluten Zeitaufwand für ein Selektionsverfahren entscheidend beeinflusst. Die vollständige Suche wird eingesetzt, um heuristische Verfahren für kleine Merkmalsmengen evaluieren zu können.

Sequentielle Suche. Da die vollständige Suche aufwendig ist, verwendet man für Probleme mit großen Merkmalsmengen suboptimale Heuristiken. Schnelle Heuristiken sind die sequentielle Vorwärts- und Rückwärtssuche, bei denen man von einem aktuellen Suchpunkt aus zum besten Nachbarn übergeht bis keine Verbesserung mehr erzielt wird oder der Null- bzw. Einszustand erreicht ist. Es handelt sich also um ein Gradientenverfahren mit einer einfachen, lokalen Suchheuristik.

	Sequentielle Vorwärtssuche (`SFS`, engl. *sequential forward search*)
Anfangszustand:	Nullzustand
Suchpunktkandidaten:	Alle Zustände, bei denen ein Bit mehr aktiviert ist als im aktuellen Zustand
Nächster Suchpunkt:	Bester Suchpunktkandidat
Abbruchkriterium:	Aktueller Zustand ist gleich gut oder besser bewertet als bester Suchpunktkandidat (alternativ: Einszustand erreicht)

	Sequentielle Rückwärtssuche (`SBS`, engl. *sequential backward search*)
Anfangszustand:	Einszustand
Suchpunktkandidaten:	Alle Zustände, bei denen ein Bit weniger aktiviert ist als im aktuellen Zustand
Nächster Suchpunkt:	Bester Suchpunktkandidat
Abbruchkriterium:	Aktueller Zustand ist gleich gut oder besser bewertet als bester Suchpunktkandidat (alternativ: Nullzustand erreicht)

Wie alle Gradientenverfahren bleibt die sequentielle Vorwärts- und Rückwärtssuche häufig in lokalen Maxima stecken. Mit einem vergleichsweise geringen Aufwand von $\mathcal{O}(n^2)$ bei $n \in \mathbb{N}$ Merkmalen eignen sich die sequentiellen Suchverfahren aber gut für schnelle Voruntersuchungen von Merkmalsmengen oder Untersuchungen großer Merkmalsmengen. Wenn man das alternative Abbruchkriterium verwendet, wird die sequentielle Suche fortgesetzt, bis alle Merkmale eingefügt oder entfernt wurden. Dadurch werden die Merkmale sortiert und man erhält einen ersten Eindruck von der relativen Wichtigkeit der einzelnen Merkmale. Ergebnis des Suchverfahrens ist dann der Zustand, der im Verlauf der Suche am besten bewertet wurde.

Evolutionäre Suche [60, 21]. Im Gegensatz zu den bisher vorgestellten Suchstrategien handelt es sich bei der evolutionären Suche um ein paralleles Suchverfahren. Ausgehend von einer Population, die zu Beginn n zufällig gewählte Suchpunkte enthält, werden durch Mutations- und Rekombinationsoperationen neue Suchpunktkandidaten generiert. Die am besten bewerteten Suchpunkte der Population und der Suchpunktkandidaten werden in die nächste Epoche übernommen und wiederum durch Mutation und Rekombination verändert. Das Verfahren bricht ab, wenn eine vorgegebene Anzahl von Epochen oder eine obere Schranke für die Bewertung erreicht wurde. Der Aufwand für die evolutionäre Suche ist proportional zu $p + eq$), wobei e die Anzahl der Epochen, p die Größe der Population und q die Anzahl der Nachkommen ist, die in jeder Epoche gebildet werden.

	Evolutionäre Suche (`ES`, engl. *evolutionary search*)
Anfangszustand:	n zufällige Suchpunkte
Suchpunktkandidaten:	Mutation und Rekombination der Population
Nächster Suchpunkt:	n beste Punkte der Population und der Suchpunktkandidaten
Abbruchkriterium:	Vorgegebene Anzahl von Generationen oder obere Schranke für Bewertung erreicht

Die Evolutionäre Suche wird in Abschnitt 6.2 zur Generierung von Melodien verwendet.

2.2 Musikalische Grundlagen

In diesem Abschnitt wird der Begriff der Melodie (Abschnitt 2.2.1) und des musikalischen Stils (Abschnitt 2.2.2) erläutert, weil sie im Kontext der computergestützten Stilanalyse allgemeiner definiert werden müssen als in der musikwissenschaftlichen Literatur üblich. Abschnitt 2.2.3 gibt Beispiele für die musikalischen Merkmale, die in späteren Kapiteln

der Arbeit vorkommen. Für die Definitionen elementarer musikalischer Grundbegriffe wie Notenschrift und Harmonielehre sei auf [58, 59, 75] verwiesen.

2.2.1 Melodie

Das Wort "Melodie" stammt vom Griechischen *mélos* und bedeutet "Lied, Gesang, gegliederte Weise". Darin drücken sich zwei definierende Aspekte einer Melodie aus: Sie ist einstimmig, da sie in ihrer Urform von einer Person gesungen wird. Sie weist eine Struktur auf, die einem Rezipienten hilft, einen musikalischen Sinn in der Melodie wahrzunehmen.

Die Einstimmigkeit und die Strukturiertheit einer Melodie findet man auch in musikwissenschaftlichen Definitionen des Melodiebegriffs wieder:

> Eine Melodie besteht aus aufeinanderfolgenden musikalischen Noten, die so in einem bestimmten rhythmischen Muster angeordnet sind, dass sie eine wiedererkennbare Einheit bilden. [75].

Nicht nur die rhythmische Struktur berücksichtigt diese Definition aus einem Musiklexikon vom Anfang des 19. Jahrhunderts:

> ⟨940⟩ Man bezeichnet damit theils eine Folge der Töne überhaupt, theils und insbesondere eine solche Tonreihe, die aus abwechselnden Stufen einer zum Grunde liegenden Tonart bestehet, in eine bestimmte Taktart eingetheilt ist, und gewisse Ruhepunkte des Geistes enthält, wodurch sie in einzelne Glieder aufgelöset werden kann.[...] Die Melodie muß auch [...] so beschaffen seyn, dass sie einer mannigfaltig abwechselnden, und der Beschaffenheit ihres Ausdruckes angemessenen Harmonie fähig ist.

Auch hier wird zwischen einer allgemeinen Definition – einer Folge von Tönen, die Melodien von anderen musikalischen Strukturen abgrenzt – und den wünschenswerten Eigenschaften einer Melodie wie z.B. "Ruhepunkten des Geistes", also einer ansprechenden Phrasierung, getrennt. Desweiteren soll die Melodie bewegt, tonal und ihrem Ausdruck entsprechend harmonisierbar sein. In dieser Charakterisierung drückt sich das klassisch-romantische Melodieideal aus, das jedoch für Melodien anderer Epochen wie z.B. Gregorianische Gesänge oder atonale Melodien nicht gleichermaßen zutrifft.

Bei der computergestützten Analyse von Melodien unterschiedlicher Epochen und Gattungen ist es daher sinnvoll, zwischen der notwendigen Eigenschaft der Einstimmigkeit einer Melodie und wünschenswerten Eigenschaften einer Melodie wie ihrer Strukturiertheit zu unterscheiden. Letztere sind stilabhängig und damit Untersuchungsgegenstand der Stilanalyse. Es wäre methodisch fragwürdig, sie in der Definition der Problemstellung vorauszusetzen.

In dieser Arbeit wird daher eine allgemeine, nicht wertende Definition einer Melodie gewählt, die auch "Melodiekandidaten" wie Zufallsmelodien einschließt. Eine *Melodie* wird als eine Folge nicht-überlappender Töne definiert, die jeweils eine Einsatzzeit, eine Dauer und eine Tonhöhe besitzen. Bei Bedarf wird diese Darstellung durch weitere Merkmale ergänzt wie z.B. rhythmische Symbole für die Dauern, Angaben zu Taktarten, Position der Takte, Tonart, Phrasierung, Dynamik und Verzierungen.

2.2.2 Musikalischer Stil

In diesem Abschnitt wird diskutiert, wie sich der Begriff des musikalischen Stils im Kontext des maschinellen Lernens definieren lässt, und wie die hier entwickelte Methodik in die musikwissenschaftliche Herangehensweise bei der Stilanalyse eingebettet ist.

Der Begriff des *musikalischen Stils* ist der musikalischen Stilkunde zuzurechnen, die "gattungsgeschichtliche Merkmale [untersucht], die über das Einzelwerk hinaus Gültigkeit haben und den musikalischen Stil einer Gattung oder einer Epoche, eines Komponisten oder einer Schule manifestieren" [58]. Die Stilkunde (oder Stilanalyse) beschreibt also prägende Eigenschaften von Musikstücken, die aufgrund äußerer, häufig nicht-musikalischer Kriterien als zusammengehörig angesehen werden.

Dies führt zu einer konkreteren Formulierung der lernbasierten Stilanalyse in dieser Arbeit: Es soll untersucht werden, ob man eine stilistische Gruppierung von Musikstücken, die nach außermusikalischen Kriterien vorgenommen wurde, mit Mitteln des maschinellen Lernens im Notentext nachweisen kann und welche musikalischen Merkmale einen solchen Nachweis ermöglichen.

Eine zweite Definition des Stilbegriffs gibt Auskunft über den Geltungsbereich und die Gewinnung musikalischer Merkmale:

> "Über den *musikalischen* Stilbegriff lässt sich etwa folgendes zusammenfassend sagen. Der Begriff, gültig für Musik von der Antike bis zur Gegenwart, wurzelt in geisteswissenschaftlichem Denken, wo er von Philosophie, Ästhetik und Psychologie als ein empirisch erfassbares Erlebnisganzes zu deuten ist, in das eine Fülle charakteristischer Merkmale eingelagert ist [...] Sein Ausstrahlungsvermögen wird erfasst durch gemeinsame Merkmale, die methodisch durch Beschreibung, Vergleich und Urteil gewonnen werden." [30, S.24f.]

Wie in der ersten Definition wird auch hier ein musikalischer Stil durch charakteristische Merkmale beschrieben, allerdings definieren sie den Stil nicht, sondern dienen der empirischen Beschreibung eines vor der Analyse vorhandenen "Erlebnisganzen".

Damit kann die Herangehensweise bei der Stilanalyse in Analogie zur "Analyse durch Synthese" (Kapitel 1) folgendermaßen strukturiert werden: Eine Musikanalytikerin hat zunächst aufgrund ihres Vorwissens eine gewisse Vorstellung der charakteristischen Eigenschaften eines Musikstils. Sie konkretisierte diese Vorstellung, indem sie potentiell stilprägende Merkmale definiert. Durch die Analyse von Notenbeispielen versucht sie dann, die Bedeutung dieser Merkmale empirisch zu erhärten. Dies kann durch eine manuelle Analyse oder computerunterstützte Methoden erfolgen. Die Interpretation der Ergebnisse der Analyse führt dann zu einer veränderten Wahrnehmung des untersuchten Stils und gibt Anlass zu neuen Hypothesen über stilprägende Merkmale.

Diese Beschreibung macht deutlich, dass sich die empirische Untersuchung musikalischer Merkmale mit lernbasierten Methoden in natürlicher Weise in den Erkenntniszyklus der Stilanalyse einfügt. Auch aus musikwissenschaftlicher Sicht wird eine empirische Herangehensweise befürwortet:

> "Der Stil sucht das Typische, sucht Merkmale, die er zur Gemeinsamkeit zu binden vermag. Der Einzelfall liefert noch kein stilistisches Bild. Er ist atypisch und entzieht sich einer zusammenfassenden Betrachtung. Die Bedingungen, mit deren Hilfe Stilbegriffe entwickelt werden, erfordern den Nachweis gleicher Merkmale, die zahlenmäßig mehrfach auftreten müssen. Durch vorsichtige Beschreibung, vergleichende Betrachtung, Herauslösung von gemeinsamen Kennzeichen entsteht die Struktur des Stilbegriffs, der alles Zufällige und Singuläre abstreift und im Grunde auf Dauer ausgerichtet ist." [30, S.15]

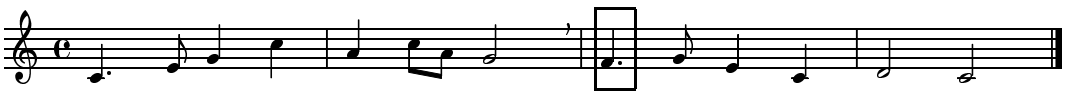
Der hier entwickelte Ansatz unterstützt einen Musikanalytiker dabei in den drei Bereichen der Merkmalsgewinnung "Beschreibung, Vergleich und Urteil". Das Merkmalsspezifikationswerkzeug der Operatorgraphen (Kapitel 5) erlaubt eine formale Definition musikalischer Merkmale, das Training lernbasierter Modelle für verschiedene Stile oder verschiedene Eingabemerkmale ermöglicht den Vergleich verschiedener Merkmalsmengen bzw. Stile. Die Leistungsfähigkeit eines lernbasierten Modells bildet die Grundlage, um zu beurteilen, ob die vom Modell verwendeten Merkmale einen Stil angemessen charakterisieren.

Eine weitere Gemeinsamkeit zwischen der musikwissenschaftlichen und der lernbasierten Stilanalyse liegt darin, dass sowohl im musikalischen als auch im informatischen Fall Merkmale ihre Bedeutung erst durch den Kontext erhalten, in dem sie betrachtet werden: Stilcharakteristika "dürfen niemals absolut in ihrer Gültigkeit betrachtet und bewertet werden, sondern sind in übergeordnete Zusammenhänge einzugliedern" [30, S.15]. Auch bei lernbasierten Merkmalsselektionsalgorithmen kann von der Relevanz der ausgewählten Merkmale nur in Bezug auf die zum Training verwendeten Beispiele gesprochen werden. Ändern sich die Daten, so wird ein Selektionsverfahren im allgemeinen andere Merkmale als wichtig auswählen.

Da die hier entwickelte Methodik musikunabhängig ist und die Fülle existierender Kategorisierungen von Musikstilen auch nicht angemessen berücksichtigt werden könnte, wird für diese Arbeit eine pragmatische Stildefinition gewählt. Ein *Stil* wird beschrieben durch eine Menge von Musikstücken, die aus Sicht eines Anwenders in einem musikalischen Bedeutungszusammenhang stehen. Unter dem Begriff Stil werden auch Einteilungen nach Genre oder musikalischen Filterregeln subsumiert, da es für die Anwendung der hier entwickelten Methoden unerheblich ist, nach welchen Kriterien die untersuchten Daten zusammengestellt wurden. Desweiteren vermeidet diese Definition, Stilkategorisierungen vorwegzunehmen, die sich eigentlich erst aus den Untersuchungsergebnissen ergeben sollten.

Die Beschreibung eines musikalischen Stils durch Beispiele ist durch die statistische Sichtweise des maschinellen Lernens motiviert. Man nimmt an, dass ein "wahrer" Prozess Daten generiert, die eventuell mit einem Rauschen behaftet sind. Beim maschinellen Lernen besteht die Aufgabe nun darin, den generierenden Prozess mit einem Modell zu approximieren, das ausgehend von einer endlichen Menge von Beispielen eingestellt wird. Ob eine Zusammenstellung von Beispielen tatsächlich für einen musikalischen Stil repräsentativ ist, ist eine Frage, die nur ein Anwender mit einem semantischen Verständnis des betrachteten Musikstils beurteilen kann.

In der Praxis hat es sich für die Leistungsfähigkeit lernbasierter Modelle als zielführend erwiesen, Datenmengen zu untersuchen, die in Bezug auf die Lernaufgabe eine gewisse Homogenität aufweisen. Beispielsweise werden in Kapitel 6 Kinderlieder betrachtet, die nach



Merkmal	Positionen	Wert
Tonhöhe	$t-2$, $t-1$, t, $t+1$, $t+2$	a, g, f, g, e
Tonhöhe pentatonisch	$t-2$, $t-1$, t, $t+1$, $t+2$	a, g, dissonant, g, e
Alteration der Tonhöhe	$t-2$, $t-1$, t, $t+1$, $t+2$	keine
Oktave der Tonhöhe	$t-2$, $t-1$, t, $t+1$, $t+2$	eingestrichen
Dauer (bzgl. ganzer Note)	$t-2$, $t-1$, t, $t+1$, $t+2$	$\frac{1}{8}$, $\frac{1}{2}$, $\frac{3}{8}$, $\frac{1}{8}$, $\frac{1}{4}$,
Intervall	$[t{-}2, t{-}1]$, $[t{-}1, t]$, $[t, t{+}1]$, $[t{+}1, t{+}2]$	große Sekunde, große Sekunde, große Sekunde, kleine Terz
Kontur	$[t{-}2, t{-}1]$, $[t{-}1, t]$, $[t, t{+}1]$, $[t{+}1, t{+}2]$	ab, ab, auf, ab
Dauernverhältnis	$[t{-}2, t{-}1]$, $[t{-}1, t]$, $[t, t{+}1]$, $[t{+}1, t{+}2]$	$0.25, 1.33, 3, 0.5$
Metrisches Gewicht	$t-1$, t, $t+1$	0.7, 1, 0.1
Rhythmusklasse	Takt	$Prototyp : (0, 0.375, 0.5, 0.75)$
Implizite Harmonie	Takthälften	D^7, T
Harmonisches Gewicht	Takt	1.125, 0.75
Ambitus	Phrase	Quinte
Erster, letzter Phrasenton	Phrase	c', f'
Intervall Phrasengrenzen	Phrase	Quarte abwärts
Phrasenform	Phrase	abwärts, flach
Bewegtheit	Phrase	2.6
Phrasenrichtung	Phrase	abwärts
Letzter Ton (Pentatonisch)	Global	c
Existiert eine Synkope?	Global	nein
Existiert eine Punktierung?	Global	ja
Existiert ein Auftakt?	Global	nein
Mittlere Phrasendauer	Global	zwei Takte
Anzahl der Phrasen	Global	2

Tabelle 2.1: Musikalische Merkmale für das Volkslied "Alle Vögel sind schon da". Die aktuelle Position t ist im Notenbeispiel eingerahmt.

Tongeschlecht gefiltert wurden. Aus musikalischer Sicht gehören alle Kinderlieder zwar demselben Stil an. Melodien desselben Tongeschlechts bauen aber auf einer gemeinsamen Skala auf, so dass alle von der Tonhöhe abgeleiteten Merkmale für ein Lernverfahren besser interpretierbar als wenn sie sich auf unterschiedliche Skalen beziehen.

2.2.3 Beispiele für musikalische Merkmale

In diesem Abschnitt werden Beispiele für die musikalischen Merkmale gegeben, die in späteren Kapiteln verwendet werden. Tabelle 2.1 zeigt das Volkslied "Alle Vögel sind schon da" mit einer eingerahmten aktuellen Position. Die Merkmale in der Tabelle werden unterteilt in

lokale Merkmale, die unmittelbar von der Position t des aktuellen Tons abhängen (Tonhöhe – Metrisches Gewicht), abstrahierende Merkmale, die mittelbar von der aktuellen Position t abgeleitet sind (Takt- und Phrasenmerkmale) sowie (melodie-)globale Merkmale, die die ganze Melodie betreffen (Letzter Ton – Anzahl der Phrasen).

Tonhöhen. Die Tonhöhe ist neben der Einsatzzeit, der Dauer und der Oktavzugehörigkeit Grundbestandteil eines Tons. Tonhöhen lassen sich auf vielerlei Weise darstellen: Die absolute, enharmonisch verwechselte Tonhöhe ist z.B. Grundlage des MIDI-Formats [76], das jede Tonhöhe durch eine Zahl darstellt. Das eingestrichene C wird z.B. durch die Zahl 60 repräsentiert, das eingestrichene Cis durch 61 usw. Die enharmonische Verwechslung alterierter Tonhöhen vereinfacht einerseits die Repräsentation, macht es aber andererseits unmöglich, alterierte Tonhöhen in einem harmonischen Kontext eindeutig zu interpretieren. Dafür ist eine Stufenrepräsentation besser geeignet, die aus der diatonischen Stufe eines Tons, einer eventuellen Alteration und der Oktavzuordnung besteht. In A-Dur befindet sich das eingestrichene Cis z.B. auf der dritten Stufe und ist hochalteriert.

Nicht immer ist es sinnvoll, den gesamten Tonvorrat in eine Tonhöhenrepräsentation aufzunehmen. Wenn man z.B. einen pentatonisch geprägten Stil wie chinesische Volkslieder untersucht, kann eine Repräsentation hilfreich sein, die basierend auf einer chinesischen Skala die Stufen I, II, III, V, VI explizit darstellt, alle nicht-pentatonischen Tonhöhen jedoch zu einem Wert "dissonant" zusammenfasst. (vgl. Merkmal "Tonhöhe pentatonisch" in Tabelle 2.1). Eine solche komprimierte Tonhöhenrepräsentation gibt zwar den Notentext nicht eindeutig wieder, kann einem maschinellen Lernverfahren aber helfen, die Funktion verschiedener Tonhöhen zu berücksichtigen.

Tonhöhen werden beim maschinellen Lernen auf allen Abstraktionsebenen berücksichtigt: Lokale Merkmale sind die aktuelle Tonhöhe an einer Position t und ihre Nachbarn. Auf einem mittleren Abstraktionsniveau bewegen sich der erste und letzte Ton der aktuellen Phrase. Ein globales Merkmal ist der letzte Ton einer Melodie, der hier aufgenommen wurde, weil er in den Melodiemengen der Essener Volkslieddatenbank unterschiedliche Häufigkeitsprofile aufweist. Während 98.1% der Kinderlieder in Dur einen tonika-eigenen letzten Ton haben (davon 71.2% Grundton), enden bis auf eine Ausnahme alle irischen Volkslieder auf dem Grundton. Im Shanxi-Corpus ist die Quinte als letzter Ton vorherrschend (68.7%), gefolgt von Sekunde (12.3%), Grundton (11.7%) und Sexte (6.4%). Die Schlusstöne variieren hier stärker als in europäischen Melodien. Besonders das Auftreten von Sekunden und Sexten als Schlusston kann darauf zurückgeführt werden, dass chinesische Skalen und Modi unter anderem durch den letzten Ton einer Melodie charakterisiert werden [22].

Rhythmus und Metrum. Aus Rhythmus und Metrum einer Melodie lassen sich verschiedene Merkmale ableiten. Die Dauer, die hier auf die Länge einer ganzen Note bezogen wird, ergibt sich direkt aus dem rhythmischen Wert einer Note. Die Dauer einer Viertelnote ist 0.25. Man könnte die Dauer einer Note stattdessen auch auf den Grundschlag der Taktart beziehen. Dann betrüge die Dauer einer Viertelnote in einer Melodie im $\frac{4}{4}$-Takt 1 und in einer Alla-Breve-Melodie 0.5. Die zweite Interpretation der Dauer würde sich beim Vergleich der Altdeutschen Balladen und der Kinderlieder aus der Essener Volksliedsammlung anbieten, weil die Altdeutschen Balladen auch Halbe und Ganze als Zählzeit verwenden.

Die Taktart einer Melodie impliziert unterschiedliche Betonungen der Taktpositionen. In einem $\frac{4}{4}$-Takt ist die erste Zählzeit am stärksten und die dritte am zweitstärksten betont,

gefolgt von der zweiten und vierten Zählzeit. Die dazwischenliegenden Achtelpositionen sind noch schwächer betont. In der europäischen Musik erhält man die Betonung durch die hierarchische Aufteilung der Taktdauer in zwei oder drei Teile gleicher Größe[1]. Wenn man jeder Betonungsstärke eine Zahl zuweist (z.B. $(1, 0.7, 0.3, 0.1)$ für die vier Zählzeiten in einem $\frac{4}{4}$-Takt), ergibt sich das *metrische Gewicht* eines musikalischen Ereignisses aus der Betonungsstärke seiner Einsatzzeit. Die Töne an den Positionen $t-1$, t und $t+1$ in Tabelle 2.1 haben z.B. das metrische Gewicht 0.7, 1 und 0.1.

Auch globale rhythmusbezogene Ereignisse wie die Existenz einer Synkope, einer Punktierung oder eines Auftakts sind Kandidaten für stilcharakteristische Merkmale.

Um den Rhythmus der Töne in einem Takt zu klassifizieren, bestimmt man zunächst, welche Rhythmen am häufigsten in den untersuchten Daten auftreten. Dazu wird die Dauer jedes Takts auf das Intervall $[0, 1]$ abgebildet. Das *rhythmische Muster* eines Takts ist der Vektor der auf $[0, 1]$ abgebildeten Einsatzzeiten der Töne. Die Häufigkeit der rhythmischen Muster in einer Melodiemenge zeigt an, welche Muster für diese Menge typisch sind. In der Kinderliedmenge der Essener Volksliedsammlung weisen z.B. 32.2% der Takte ein regelmäßiges Vierermuster $(0, 0.25, 0.5, 0.75)$ auf. Die Rhythmusklasse eines Takts bestimmt man, indem man eine selbstgewählte Anzahl von häufigsten rhythmischen Mustern als Prototypen einer *Rhythmusklasse* definiert und jedes neue rhythmische Muster dem ähnlichsten Prototyp zuordnet. Der ähnlichste Prototyp wird ermittelt, indem man eine Zuordnung zwischen den Punkten des rhythmischen Musters und jedes Prototyps vornimmt, bei der die Differenz zwischen zugeordneten Punkten und die Anzahl der nicht zugeordneten Punkte minimal sind. Das rhythmische Muster wird der Klasse des Prototypen mit der besten Bewertung zugeordnet. In Tabelle 2.1 ist das rhythmische Muster des aktuellen Takts zufällig selbst ein Prototyp.

Relationale Merkmale. Wenn man zwei gleichartige Merkmale in Beziehung zueinander setzt, entsteht ein relationales Merkmal. Die Differenz zweier Tonhöhen liefert z.B. ein *Intervall*. Die *Kontur* eines Intervalls abstrahiert von seiner Größe und gibt nur die Bewegungsrichtung aufeinanderfolgender Tonhöhen wieder. Doch auch multiplikative Relationen können von Nutzen sein. Das *Dauernverhältnis* zweier Ereignisse ist definiert als Quotient zwischen zwei Dauern, wobei man davon ausgeht, dass es keine Ereignisse ohne zeitliche Ausdehnung gibt. Das Dauernverhältnis beschreibt die Beschleunigung bzw. Verlangsamung einer Melodie. Tabelle 2.1 gibt das Dauernverhältnis für die Positionspaare $[t-2, t-1]$, $[t-1, t]$, $[t, t+1]$, $[t+1, t+2]$ an.

Phrase. Es gibt eine Vielzahl von Merkmalen, um eine Phrase zu charakterisieren. Eine Phrase hat eine Dauer, das Intervall zwischen erstem und letztem Ton beschreibt grob die Richtung der Phrase. Genauer ist die Analyse der *Phrasenform*, die hier durch die Untermerkmale "Kontur zwischen den Randtönen der Phrase" (auf, ab, etwa gleichbleibend) und der Abweichung der Phrasentöne von der Kontur (bauchig, flach) implementiert wurden. Die *Bewegtheit* einer Phrase gibt ihre mittlere absolute Intervallgröße in Halbtonschritten an ($\frac{1}{5}(2+3+4+2+2) = 2.6$). Die *Phrasenrichtung* beschreibt den Trend der Phrase, wobei im Gegensatz zur Richtung bei der Berechnung der Phrasenform hier alle Töne der Phrase

[1] Ausgenommen sind ungewöhnliche Taktarten wie der $\frac{5}{4}$-Takt, der additiv in 3+2 oder 2+3 Viertel zerlegt wird.

Corpus	I	I#	II	IIIb	III	IV	IV#	V	V#	VI	VIIb	VII
Kinder (Dur)	18.1	0.0	13.1	0.02	19.9	8.4	0.02	29.2	0.01	8.7	0.1	2.4
Irland (Dur)	25.5	0.0	13.8	0.0	14.9	9.4	0.3	18.05	0.0	9.7	1.6	6.8
Shanxi	17.3	0.01	18.7	0.01	17.0	2.1	0.05	23.4	0.1	17.7	0.1	3.6

Tabelle 2.2: Relative Häufigkeiten der enharmonisch verwechselten, chromatischen Tonhöhen in drei Volksliedcorpora (in Prozent).

in die Berechnung der Richtung einbezogen werden. Die *mittlere Dauer einer Phrase* ist ein globales Merkmal, das die Phrasenstruktur einer Melodie kennzeichnet.

Harmonik. In europäischer Musik impliziert auch eine (einstimmige) Melodie eine harmonische Interpretation, da europäische Zuhörer eine dur-moll-tonale Hörweise gewohnt sind. Die *implizite Harmonik* [36] einer Menge von Tönen wird in dieser Arbeit folgendermaßen berechnet: Zunächst wird ein Vorrat an harmonischen Funktionen festgelegt, z.B. Tonika, Dominante mit Septime und Subdominante. Für jede harmonische Funktion werden nun die oben definierten metrischen Gewichte der Töne addiert, die in der Harmonie vorkommen. Die harmonische Funktion mit der höchsten Bewertung wird als implizite Harmonie der bewerteten Tonmenge angesehen. Im Beispiel impliziert die erste Takthälfte die Dominante G-Dur und die zweite die Tonika C-Dur. Das *harmonische Gewicht* einer Menge von Tönen ist die höchste Bewertung, die eine harmonische Funktion bei der Berechnung der impliziten Harmonie erzielt, im Beispiel 1.125 und 0.75 für die beiden Hälften des aktuellen Takts.

Statistische Analyse von Merkmalen. Um sich einen ersten Eindruck von der Zusammensetzung einer Datenmenge zu verschaffen, kann man die Häufigkeit eines Merkmals für einen Melodiecorpus berechnen.

Die *relative Häufigkeit* eines Merkmals erhält man, indem man die *absolute Häufigkeit* jedes Merkmalswerts im Corpus zählt und durch die Summe der absoluten Häufigkeiten teilt. Als Beispiel sind in Tabelle 2.2 die relativen Häufigkeiten der enharmonisch verwechselten, chromatischen Tonstufen für die Kinderlieder (Dur), irischen Volkslieder (Dur) und die chinesischen Volkslieder der Provinz Shanxi aus der EsAC-Volksliedsammlung angegeben. Auffällig ist, dass alle Corpora einen sehr geringen nicht-diatonischen Anteil von Tonhöhen besitzen (Kinder: 0.144%, Irland: 1.910%, Shanxi: 0.311%). Der Anteil der pentatonischen Töne (Stufen I, II, III, V, VI) ist aber bei den Shanxi-Liedern mit 94.015% größer als bei den beiden anderen Melodiemengen (Kinder: 89.080%, Irland: 81.972%), was vermutlich darauf zurückzuführen ist, dass – auch wenn jeder Ton in chinesischen Volksliedern auftreten kann – das pentatonische Gerüst des Tonvorrats in diesem Melodiestil eine herausragende Rolle spielt. Bemerkenswert ist ebenfalls, dass im Shanxi-Corpus die zweite Stufe nach der fünften Stufe der wichtigste Ton ist. Dies ist durch die Tatsache zu erklären, dass die chinesischen Volkslieder nicht dur-moll-tonal konzipiert sind, sondern auf Skalen basieren, von denen eine Skala die zweite Stufe als zentralen Ton verwendet [22].

Kapitel 3

Stand der Forschung

In dieser Arbeit werden die Problemstellungen Stilmodellierung, Unterscheidung und Erkennung musikalischer Stile sowie evolutionäre Melodiegenerierung behandelt. In Kapitel 4 entwickeln wird eine Repräsentation entwickelt, die auf das Lernen von diskreten Zeitreihen zugeschnitten ist. Darauf aufbauend wird in Kapitel 5 ein neues Verfahren zur Spezifikation und Berechnung zeitreihenspezifischer Merkmale und Lernmuster vorgestellt, das dann in Kapitel 6 in musikalischen Anwendungen eingesetzt wird.

Dieses Kapitel beschreibt, welche Ansätze es zu den genannten Problemstellungen und Anwendungen bereits gibt, wo vorhandene Methoden wie z.B. maschinelle Lernverfahren und Merkmalsselektionsverfahren verwendet werden können und wo Entwicklungsbedarf besteht. Letzteres ist insbesondere bei der systematischen Spezifikation und Berechnung von Merkmalen für das Lernen musikalischer Zeitreihen der Fall, also beim inhaltlichen Schwerpunkt der vorliegenden Arbeit.

Der Aufbau dieses Kapitels folgt dem inhaltlichen Aufbau der Arbeit. Nachdem Abschnitt 3.1 die Wahl der informatischen Methodik begründet, wird in Abschnitt 3.2 ein Überblick über die Repräsentation und Transformation von Merkmalen in der Musikinformatik gegeben. Abschnitt 3.3 fasst Arbeiten zusammen, deren Fragestellung mit den in Kapitel 6 untersuchten Anwendungen übereinstimmt. Schließlich werden in Abschnitt 3.4 die relevanten Vorarbeiten skizziert, die in der Forschungsgruppe “Informationsstrukturen in der Musik” an der Fakultät für Informatik der Universität Karlsruhe entstanden sind.

3.1 Andere Verfahren des maschinellen Lernens

3.1.1 Neuronale Netze und Support-Vektor-Maschinen

Da diese Arbeit im Rahmen der Forschungsgruppe "Neuronale Netze" am Institut für Logik, Komplexität und Deduktionssysteme der Universität Karlsruhe entstanden ist, war es naheliegend, neuronale Netze als Methode des maschinellen Lernens einzusetzen. Die behandelten Klassifikationsaufgaben könnten aber auch mit anderen Klassifikationsverfahren gelöst werden. Daher wird im folgenden nicht von neuronalen Netzen, sondern allgemein von Klassifikatoren die Rede sein.

Bei früheren Ansätzen zum maschinellen Lernen wurden neuronale Netze zur Lösung einer Vielzahl ganz unterschiedlicher Problemstellungen erfolgreich eingesetzt, beispielsweise zur Modellierung des olfaktorischen Lernens bei der Honigbiene, bei Anwendungen aus der Regelungstechnik, bei Brettspielen, zur Musikmodellierung und zur Prognose von Finanzzeitreihen [57]. In den letzten Jahren haben Support-Vektor-Maschinen [84] als maschinelles Lernverfahren an Bedeutung gewonnen, so dass die Frage aufgeworfen wurde, ob diese besser als neuronale Netze zur Modellierung musikalischer Stile geeignet seien.

Im Rahmen einer Studienarbeit [67] wurde die Leistungsfähigkeit von Support-Vektor-Maschinen und neuronalen Netzen für das Lernen von Choralharmonisierungen verglichen. Dabei standen die Generalisierungsfähigkeit und die Laufzeit der Klassifikatoren sowie die musikalische Angemessenheit der Ergebnisse im Zentrum des Interesses. Gegenüber den neuronalen Netzen, die in dem von der Forschungsgruppe "Informationsstrukturen in der Musik" früher entwickelten Choralharmonisierungssystem HARMONET [32] und dem Melodieumspielungssystem MELONET [35] zum Einsatz kommen, bieten Support-Vektor-Maschinen mit Gausskern, linearem, polynomiellem oder sigmoidem Kern keine signifikante Verbesserung der Generalisierungsleistung. Die Anwendung der verschiedenen Kerne führte trotz unabhängiger Modelloptimierung zu weitgehend gleichen Resultaten. Auch die Anzahl der jeweils zur Modellbildung verwendeten Supportvektoren war bei allen Kernen ähnlich. Sie betrug jeweils mehr als die Hälfte der Anzahl der Trainingsbeispiele und war somit sehr hoch, was zu einer aufwendigen Auswertung der Support-Vektor-Maschinen führte.

Die in der Studienarbeit untersuchten Modelle arbeiteten aus diesem Grund wesentlich langsamer als die neuronalen Netze, die bezüglich ihrer Effizienz unabhängig von der Kardinalität der Lernstichprobe sind. Bei einer versteckten Schicht von 20 Neuronen wurden die Feed-Forward-Netze, die in HARMONET die harmonische Funktion berechnen, bei vergleichbarer Generalisierungsleistung und unter Anwendung des Lernverfahrens RPROP mit der Bibliothek N++ [10] etwa um den Faktor 140 schneller ausgewertet als die Support-Vektor-Maschinen der Bibliothek LIBSVM [14]. Da sich die neuronalen Netze in [67] im Vergleich zu den Support-Vektor-Maschinen bei vergleichbarer Leistung als erheblich effizienter herausstellten und die Laufzeit eines Merkmalsselektionsverfahrens maßgeblich vom Aufwand für die Berechnung des Gütekriteriums durch Klassifikatoren bestimmt wird, werden in dieser Arbeit neuronale Netze als Lernmodell eingesetzt.

3.1.2 Verfahren zur automatischen Merkmalsextraktion

Bisher existiert noch keine Theorie zum Auffinden charakteristischer Merkmale in Datenmengen. Es gibt jedoch viele Verfahren mit dem Ziel, Merkmale aus Daten zu extrahieren. Dabei sind die Daten als Punkte eines Vektorraums gegeben, wobei jede Koordinate als ein Merkmal des Vektors verstanden wird.

Ein verbreiteter statistischer Ansatz zum Auffinden wichtiger Merkmale ist die Komponentenanalyse [21]. Unter diesem Begriff werden verschiedene Verfahren zusammengefasst. Die Hauptkomponentenanalyse (*principal component analysis*, PCA) ist ein unüberwachtes Verfahren zur Dimensionsreduktion der Daten durch eine lineare Koordinatentransformation, die die Varianz der einzelnen Merkmale berücksichtigt. Ziel der Faktoranalyse (*factor analysis*) ist es, eine ebenfalls niedrigerdimensionale Darstellung der Daten zu finden, die die Korrelation zwischen den Merkmalen ausnutzt. Bei der Analyse unabhängiger Komponenten (*independent component analysis*, ICA) werden Signalquellen getrennt, indem die Daten in ein Koordinatensystem mit stochastisch möglichst unabhängigen Koordinaten transformiert werden.

Ein explorativer Ansatz zur Relevanzbestimmung von Merkmalen ist die Merkmalsselektion (vgl. Abschnitt 2.1.5), bei der ein Suchverfahren unterschiedliche Merkmalskombinationen mit einer Strategie durchläuft, die im Verlauf der Suche ein vorgegebenes Gütekriterium zur Bewertung von Merkmalskombinationen optimiert. Alle genannten Verfahren gehen von Datenmengen aus, die in einen Vektorraum eingebettet sind. Sie verwenden kein Wissen über den modellierten Problembereich und setzen voraus, dass die analysierten Daten im Vorfeld geeignet aufbereitet wurden.

In einen zweidimensionalen euklidischen Vektorraum eingebettete torusförmige Punkteklassen sind ein einfaches Beispiel, bei dem eine durch Problemwissen motivierte Transformation der Daten – hier z.B. von euklidischen Koordinaten in Polarkoordinaten –, eine Aufgabe wie die Klassifikation der Datenpunkte erleichtern kann. Solche Transformationen sind nur schwer zu finden, wenn man die Beschaffenheit der zu analysierenden Daten nicht wie im Beispiel des Torus geschlossen beschreiben kann.

Vor einem solchen Problem steht man bei der Analyse musikalischer Daten. Um dennoch für eine Fragestellung relevante Aspekte aus musikalischen Daten herauszuarbeiten, wendet man auf die Daten mehrere einfache Operationen an, die jeweils einen Aspekt des Problemwissens modellieren. Wenn man geeignete Operationen kombiniert, ist eine bessere Lösung der betrachteten Fragestellung mit maschinellem Lernen möglich. Diese Idee liegt der Transformation von diskreten Zeitreihen mit Operatorgraphen zugrunde.

In dieser Arbeit wird zur Datenanalyse die Kombination von Merkmalsselektionsverfahren und neuronalen Netzen verwendet, weil dieser Ansatz die Selektion mehrdimensional codierter Merkmale erlaubt, während bei den Verfahren zur Komponentenanalyse einzelne Dimensionen betrachtet werden. Der hier entwickelte Ansatz kann jedoch auch zur Vorverarbeitung von Daten für die eingangs beschriebenen Merkmalsextraktionsverfahren verwendet werden.

3.2 Repräsentation und Transformation musikalischer Strukturen

3.2.1 Datenformate zur Repräsentation von Musikdaten

Es gibt eine Fülle von Datenformaten, um Musikdaten darzustellen bzw. auszutauschen [76]. Diese Vielfalt ist einerseits darin begründet, dass die Bemühungen um eine Normierung von Musikdatenformaten erst in den vergangenen Jahren begonnen haben. Andererseits dienen die Datenformate ganz unterschiedlichen Zielen. Aus Sicht des maschinellen Lernens, wie es in dieser Arbeit verwendet wird, ist man daran interessiert, über die musikalischen Basisdaten (Einsatzzeiten, Tonhöhen, Tondauern) hinaus komplexe musikalische Zusammenhänge darstellen zu können, die den Melodien innewohnen. Im folgenden wird kurz auf einige Musikdatenformate und ihre Eignung für die hier behandelte Aufgabenstellung eingegangen.

MIDI [41] ist das Akronym für *Musical Instrument Digital Interface* und steht für eine Hardwareschnittstelle ebenso wie für ein binäres Dateiformat und eine Spezifikation für die Simulation von Instrumenten. Es dient zur ereignisbasierten Darstellung von Tönen und deren Übertragung über eine digitale Schnittstelle. Töne werden auf einer chromatischen Skala als ganze Zahlen dargestellt. Damit ist keine Unterscheidung zwischen enharmonisch äquivalenten Tönen (z.B. *cis* und *des*) möglich. Das MIDI-Format ist für den Austausch musikalischer Basisdaten geeignet, nicht aber für die Annotation komplexer musikalischer Analysen.

Eine andere Zielsetzung haben Sound- und Notensatzformate. Soundformate (z.B. Csound, MP3) dienen der Repräsentation von Audiodaten und konzentrieren sich damit auf die klanglichen Eigenschaften von Tönen, während bei Notensatzformaten (z.B. NIFF, Guido) die graphische Darstellung des Notentexts im Vordergrund steht.

Für diese Arbeit interessant ist das EsAC-Format [19], ein ASCII-basiertes Datenformat zur Darstellung monophoner Melodien. Es wurde vom Musikwissenschaftler Helmut Schaffrath entwickelt, um Forschungsaktivitäten im Bereich Ethnomusikologie und Musikanalyse zu unterstützen. Die EsAC-Volksliedsammlung [19] ist eine der wenigen umfangreichen systematischen Sammlungen von Melodien. Sie enthält über 10.000 Volkslieder gruppiert nach unterschiedlichen Stilen und unterschiedlicher Herkunft (z.B. deutsche Kinderlieder, irische Volkslieder, chinesische Lieder aus der Provinz Shanxi). Im EsAC-Format sind auch einige übergeordnete Strukturen darstellbar, z.B. die Takt- und Phrasenstruktur der Lieder.

Allgemeiner als EsAC ist das Kern-Format [40], in dem sich auch mehrstimmige Musikstücke darstellen lassen. In Kombination mit Humdrum [31], einer Sammlung von Werkzeugen für die Musikanalyse, lassen sich komplexere Strukturen – beispielsweise harmonische Analysen – annotieren. Durch seine Erweiterbarkeit ist Kern offen für die Integration neuer Merkmale und Analysemethoden. Allerdings ist Kern ebenso wie EsAC nicht für die Darstellung überlappender Zeitintervalle geeignet.

3.2.2 Analytische Musikstrukturmodelle

Während Datenformate dazu dienen, die musikalischen Basisinformationen eines Musikstücks wie Töne, Dauern und Notationssymbole darzustellen, sollen analytische Melodiestrukturmodelle dabei helfen, die semantischen Beziehungen musikalischer Strukturen und Prozesse zu erkunden.

Prozessorientierte, sequentielle Ansätze. Prozessmodelle konzentrieren sich auf die zeitliche Abfolge der Ereignisse. Ausgangspunkt dieser Modelle ist die Beobachtung, dass eine Folge von Ereignissen beim Hörer Erwartungen hinsichtlich der Fortsetzung der Ereignisfolge hervorruft. Die Erfüllung oder Nichterfüllung dieser Erwartungen stellt einen wichtigen Bestandteil der musikalischen Wahrnehmung dar [6].

Ein Beispiel hierfür ist das IR-Modell (*Implication-Realization Model*) von Narmour [64, 65], durch das Teilsequenzen der Melodie gemäß ihrer Struktur klassifiziert werden. Dabei sind die Merkmale, anhand derer die Strukturen klassifiziert werden, bereits festgelegt – beispielsweise steht ein *Process* für ähnliche, auf- oder absteigende Intervalle in gleicher melodischer Richtung, *Duplication* bezeichnet die Wiederholung von Tönen. Das IR-Modell berücksichtigt auch explizit die Überlappung von Teilsequenzen.

Kritisiert wurde an dem IR-Modell die starke Konzentration auf wenige, vorab festgelegte und zeitlich lokale Merkmale. Dies macht eine Berücksichtigung stiltypischer Merkmale, wie sie in anderen, insbesondere außereuropäischen Kulturen auftreten, schwierig [81].

Reduktionistische, hierarchische Ansätze. Reduktionsmodelle versuchen die Ereignisse eines Musikstücks in eine Hierarchie der strukturellen Wichtigkeit nach bestimmten Gesichtspunkten (z.B. metrischen und harmonischen Kriterien) einzuordnen. Die Einzelereignisse werden schrittweise zu größeren Einheiten zusammengefasst, bis das Stück auf ein Gesamtereignis reduziert ist.

Prominentes Beispiel hierfür ist die GTTM (*Generative Theory of Tonal Music*) von Lerdahl und Jackendoff [51], bei der eine Melodie schrittweise reduziert wird. Die Länge eines Astes des sich aus der Reduktion ergebenden Baums entspricht dabei der strukturellen Bedeutung des zugehörigen Tons. Das Reduktionsmodell liefert eine streng hierarchische Struktur, bei der keine Überlappung der Segmente möglich ist. Desweiteren unterscheidet die GTTM zwischen Wohlgeformtheitsregeln (allgemeine Eigenschaften einer Reduktion) und Präferenzregeln, mit denen Besonderheiten eines Musikstils beschrieben werden. Merkmale sind dadurch weniger festgelegt als bei Narmour.

Die Schenker-Analyse [24] geht auf den österreichisch-polnischen Musiktheoretiker Heinrich Schenker (1868-1935) zurück und ist vor allem in den USA populär geworden. Sie versucht die gesamte tonale Musik auf eine einfache, auf dem Tonikadreiklang basierende Tonfortschreitung (den *Ursatz*) zurückzuführen. Diese besteht aus einem melodischen Prototyen (der *Urlinie*) und einem harmonischen Prototypen (der *Bassbrechung*). Die Schenker-Analyse betrachtet infolgedessen jedes tonale Werk lediglich als Verzierung des Ursatzes.

Mathematische Ansätze. Einen grundsätzlich anderen Ansatz verfolgt die mathematische Musiktheorie MaMuth [55, 56], die algebraische und geometrische Methoden zur mathematischen Modellierung von Musik anwendet. Elementare musikalische Strukturen wie Töne,

Motive und Harmonien werden durch ihre Einbettung in mehrdimensionale Vektorräume definiert. Die Abstraktion dieser und anderer musikalischer Grundbegriffe von ihrer musikwissenschaftlichen Verwendung ermöglicht eine neue Betrachtungsweise musikalischer Strukturen, erschwert aber andererseits die in dieser Arbeit angestrebte Integration von Problemwissen für Anwender mit musikwissenschaftlichem Schwerpunkt.

Perspektivische Ansätze. Mit Hilfe des *Multiple Viewpoint Model* [16] lassen sich Sichten (viewpoints) definieren, um aus der Oberflächenstruktur eines Musikstücks übergeordnete Strukturen zu gewinnen. Sichten sind mathematische Funktionen, die durch den Musikanalytiker definiert werden können und auf der Basisdarstellung des Musikstücks operieren. Sie modellieren bestimmte Typen von musikalischen Merkmalen, z.B. eine Melodiekontur, Intervalle oder Dauern. Ein Musikstück wird auf diese Weise aus der Basisdarstellung in eine abgeleitete Darstellung transformiert, die eine bestimmte Perspektive auf das Stück repräsentiert. Mehrere Sichten können hintereinander auf die Ausgangsdaten angewendet werden.

Multiple Viewpoint Models stehen in engem Bezug zu den in Kapitel 5 definierten Operatorgraphen. Auch bei Operatorgraphen besteht die Grundidee darin, aus der Basisdarstellung eines Musikstücks eine abgeleitete Darstellung zu gewinnen. Eine Sicht des Multiple Viewpoint Models entspricht einem Operatorgraphen der vorliegenden Arbeit. Die Verkettung von Sichten beim Multiple Viewpoint Model folgt dem funktionalen Programmierparadigma, so dass Daten und Funktionen miteinander vermischt werden. Im hier entwickelten Ansatz wird die Repräsentation von Musikstücken mit Ansichten von ihrer Transformation mit Operatorgraphen konzeptionell getrennt. Dies ist im Hinblick auf das maschinelle Lernen von Vorteil, da durch Analysen annotierte Musikstücke und Lernmuster unabhängig von der Methode verfügbar sind, mit der sie generiert wurden.

Ein weiterer Unterschied dieses Ansatzes zu Multiple Viewpoint Models besteht darin, dass hier Klassen von Operatoren vorgegeben werden, deren Funktionalität teils zeit- und teils anwendungsbezogen ist. Dadurch wird eine bessere Austauschbarkeit von Problemwissen gewährleistet als bei einem Ansatz der die Definition beliebiger Funktionen erlaubt. Desweiteren stellen Operatorgraphen einen Informationsfluss graphisch dar, wohingegen die Verkettung von Sichten im Multiple Viewpoint Model für einen Anwender weniger transparent ist.

3.3 Stilmodellierung

3.3.1 Musikalische Merkmalsselektion

Bei der Merkmalsselektion geht es darum, die zur Lösung eines Problems wichtigen Merkmale aus einer Menge von Merkmalen herauszufiltern. Im Kontext der Stilmodellierung besteht die Aufgabe darin, die für einen Stil typischen Merkmale zu ermitteln bzw. gegenüber anderen Stilen abzugrenzen. Bislang sind Verfahren zur Merkmalsselektion kaum auf musikalische Problemstellungen angewandt worden. Die vorliegende Arbeit will dazu einen Beitrag leisten.

3.3.2 Melodiegenerierung

Historische Verfahren. Die Geschichte des algorithmischen Komponierens lässt sich bis in die klassische Antike zurückverfolgen. Im 18. Jahrhundert entwickelten viele Komponisten musikalische Würfel- oder Kartenspiele [71], mit denen z.B. 16-taktige Walzer durch Würfeln komponiert werden konnten. Dabei ist ein 16-teiliges Taktraster vorgegeben. Für jeden Takt steht ein Vorrat an sorgfältig zusammengestellten Takten zur Verfügung, die jeweils in einer Harmonie komponiert wurden und sich gut an die benachbarten Takte anfügen. Durch Würfeln wird aus jeder der 16 Mengen ein Takt ausgewählt und in das Raster eingesetzt. Bassbrechung

Grammatikbasierte, hierarchische Verfahren. Die ersten Versuche, musikalische Strukturen mit formalen Methoden zu beschreiben, entstanden aus den Erfolgen der Linguistik [15] und dem Interesse für natürlichsprachige Erkennungssysteme innerhalb der Informatik. Dies führte zu konkreten Fragestellungen zur Anwendbarkeit linguistischer Konzepte und Hilfsmittel auf die Musik, insbesondere zur Darstellung musikalischer Strukturen mit Hilfe formaler Grammatiken, die in vielerlei Arbeiten Ausdruck fanden [73].

Ein möglicher Kompositionsmechanismus zur Realisierung des reduktionistischen Ansatzes (vgl. Abschnitt 3.2.2) besteht darin, die Strukturelemente der einzelnen Hierarchiestufen von oben nach unten festzulegen. Bei der Umsetzung dieses Top-down-Mechanismus bieten sich formale Grammatiken an, die beispielsweise in [52, 3, 53] zur Komposition einfacher Melodien (Bachmelodien, Volkslieder) eingesetzt werden.

Lernbasierte, sequentielle Verfahren. Der sequentielle Ansatz lässt sich zur Melodiegenerierung einsetzen, indem die Vorhersage musikalischer Ereignisse in Melodien als ein Problem des Lernens und der Erzeugung von Symbolfolgen aufgefasst wird. Eine Vielzahl unterschiedlicher Methoden zur zeitlichen Musterverarbeitung wurden dementsprechend auf musikalische Problemstellungen angewandt, insbesondere neuronale Netze [82, 62, 83]. Dabei zeigt sich, dass die Betrachtung als Folge von Einzeltönen nicht ausreicht, um wesentliche Eigenschaften von Melodien zu erfassen [61]. Das Auffinden geeigneter abstrahierender Darstellungen für stilprägende Merkmale ist daher von zentraler Bedeutung für die Melodiemodellierung.

Evolutionäre Verfahren. Eine dritte Methode zur Komposition musikalischer Strukturen sind evolutionäre bzw. genetische Algorithmen. Sie lassen sich als heuristische Optimierungsverfahren verwenden, um eine Population von Melodien solange zu verändern, bis eine Melodie mit möglichst guter Bewertung (Fitness) gefunden wurde [26, 43, 7].

Pattern-Matching-Verfahren. Natürlich gibt es eine Vielzahl weiterer Verfahren zur Generierung musikalischer Strukturen. David Cope [17] verwendet spezielle Pattern-Matching-Verfahren, um den Stil eines Komponisten nachzubilden und diese bei der Komposition ganzer Sinfonien beispielsweise im Stil Mozarts zu verwenden. Die Verfahren werden bei Cope als Hilfsmittel für das algorithmische Komponieren, weniger zur Stilanalyse und -modellierung eingesetzt.

3.3.3 Stilunterscheidung und Stilerkennung

Das Problem der Stilerkennung und -unterscheidung wurde bereits mit unterschiedlichen statistischen und lernbasierten Verfahren untersucht. In [44] wird ein neuronales Netz zur Erkennung des Bach'schen Choralstils trainiert. Der Ansatz erfordert jedoch eine erhebliche manuelle Vorverarbeitung musikalischer Merkmale. Andere Autoren interpretieren die Ausgabeaktivierungen [6, 5] oder die Energie [4] eines trainierten neuronalen Netzes als Maß für den Grad musikalischer Erwartungshaltungen. Die Übereinstimmung der Erwartungen von Versuchspersonen mit den durch ein neuronales Netz erzeugten Aktivierungen scheint die Interpretation der Netzausgaben als musikalische Erwartungen zu bestätigen.

Ein anderer Ansatz mit unüberwachten Lernverfahren wird in [42] verfolgt. Hier werden basierend auf Ton- und Tonübergangsverteilungen selbstorganisierende Karten (self-organizing maps) [46] mit Melodien in zehn unterschiedlichen Stilen trainiert. Die Ergebnisse zeigen gute Übereinstimmungen mit den musikwissenschaftlichen Beschreibungen der jeweiligen Stile.

3.4 Vorarbeiten der eigenen Forschungsgruppe

3.4.1 Methodischer Ansatz: Analyse durch Synthese

Grundlage der Vorarbeiten der Forschungsgruppe *Informationsstrukturen in der Musik* ist ein zyklisches Verfahren zur Modellentwicklung, mit dem die Beschreibung eines musikalischen Gegenstands schrittweise verbessert werden soll [101]. Diese als *Analyse durch Synthese* bezeichnete Vorgehensweise beinhaltet die Schritte *Analyse*, *Lernen*, *Synthese* und *Evaluierung* (siehe Abbildung 1.6 im Einleitungskapitel).

Es folgt ein Überblick über die wichtigsten Forschungsbeiträge zu den einzelnen Schritten der zyklischen Modellentwicklung.

3.4.2 Analyse von Merkmalen

Klassifikation melodischer Strukturen. Die Betrachtung von Motiven spielt eine zentrale Rolle bei der Analyse melodischer Strukturen. Ein Beispiel dafür ist die sogenannte paradigmatische Analyse [66]. Die Aufgabe besteht darin, ein Musikstück in Segmente aufzuteilen und diese Segmente hinsichtlich ähnlichkeitsbasierter Kriterien zu klassifizieren [2, 13]. Die Aufgabe der Motivklassifizierung besteht darin, alle Motive einer Motivfolge bezüglich der Ähnlichkeit ihrer Struktur einer Menge von Motivklassen zuzuordnen. Jede Motivklasse besteht dann aus einem Prototypen und dessen mehr oder weniger veränderten Vorkommen in der Motivfolge.

Beide Teilprobleme sind anspruchsvoll und besitzen keine eindeutige Lösung. In [90] wird die Aufgabe durch die Vorgabe einer festen Segmentierung vereinfacht. Damit konzentriert sich die Untersuchung auf die Klassifizierung von Motiven gleicher Dauer, die den jeweiligen Segmenten zugeordnet sind. Es werden verschiedene Motivrepräsentationen und Clusterverfahren eingesetzt und die Ergebnisse mit einer musikwissenschaftlichen Experten-Analyse

verglichen. Dabei zeigt sich, dass die Motivrepräsentation einen deutlich größeren Einfluss auf das Klassifikationsergebnis hat als die Wahl des Clusterverfahrens. In dem durchgeführten Experiment erweist sich die Kontur als das wichtigste Merkmal zur Klassifizierung motivischer Strukturen. Die Unterschiede hinsichtlich der Algorithmen sind nur geringfügig.

Untersuchung von Segmentierungsverfahren. Aufgabe von Segmentierungsverfahren ist die Unterteilung von Melodien in Segmente beliebiger Länge. Dies stellt eine wesentliche Verallgemeinerung gegenüber einer festen Segmentierung dar.

In [96, 97] werden für das Problem der Phrasen- und Motivfindung zwei Segmentierungsverfahren für Melodien miteinander verglichen: der *Grouper*-Algorithmus von Temperley und Sleator [80] und das *Local Boundary Detection Model* von Cambouropoulos [12]. Da es i.a. keine eindeutige Lösung bei der Analyse der Phrasen- und Motivstruktur einer Melodie gibt, um das Ergebnis der beiden Segmentierungsverfahren zu bewerten, wurden 17 Musiker gebeten, einen Corpus mit zehn Melodien unterschiedlicher Stilrichtungen in musikalisch sinnvolle Abschnitte zu unterteilen. Die erhobenen Daten bestätigen die Vermutung, dass häufig mehrere musikalisch sinnvolle Segmentierungen einer Melodie existieren. Die Ergebnisse der Algorithmen wurden nun durch ihren mittleren "Abstand" von den Segmentierungen der musikalischen Experten bewertet. Dieses Gütemaß hat den Vorteil, dass die inhärente Mehrdeutigkeit der Segmentierungsaufgabe berücksichtigt wird. Im Mittel stimmen die Ergebnisse des *Grouper*-Algorithmus besser mit den Segmentierungen der Experten überein als die des *Local Boundary Detection Model*.

3.4.3 Lernen musikalischer Strukturen mit maschinellen Verfahren

Im Verlauf des Forschungsprojekts wurde ein Vielzahl unterschiedlicher musikalischer Problemstellungen mit Verfahren des maschinellen Lernens untersucht.

Harmonisierung vierstimmiger Choräle mit neuronalen Netzen. HARMONET [32] ist ein Musikharmonisierungssystem, das Melodien im Stil eines Komponisten mit neuronalen Netzen vierstimmig harmonisieren lernt. Es wurde ursprünglich entwickelt, um Choralharmonisierungen im Stil von J. S. Bach zu lernen, ist aber auch auf andere Musikstile erweitert worden [39]. Am Beispiel von Choralkompositionen von Komponisten verschiedener Epochen (J. S. Bach, M. Reger, S. Scheidt) konnte gezeigt werden, dass sich auf diese Weise Harmonisierungen in den gelernten Stilen generieren lassen.

Entwicklung mehrstufiger Verfahren zum Lernen übergeordneter Strukturen. Während sich harmonische Strukturen recht gut durch Betrachtung einer festen Zeitstufe (eine Harmonie pro Zählzeit) aus Beispielen lernen lassen, scheiterten alle Versuche, selbst einfache melodische Strukturen als Abfolge von Einzeltönen zu modellieren. Der Grund besteht darin, dass Einzeltonmodelle nicht in der Lage sind, übergeordnete, auf verschiedenen Zeitstufen gleichzeitig auftretende Strukturen wie Harmonien, Motive und Phrasenstrukturen zu erfassen.

Hier setzt die Arbeit von Hörnel [36] an, in der mehrstufige Modelle zum Lernen übergeordneter Strukturen eingesetzt werden. Die Grundidee besteht darin, eine adäquatere Beschreibung zeitlicher Prozesse dadurch zu finden, dass ein Teil des zu lösenden Lernproblems auf

eine höhere Zeitstufe verlagert wird. Dazu werden die Symbole einer Sequenz – beispielsweise einer Tonfolge – zu größeren Segmenten, also z.B. zu Tongruppen zusammengefasst. Lernen findet dann auf verschiedenen Zeitstufen statt.

Lernen von Melodieumspielungen im Stil von J. Pachelbel. Mehrstufige Modelle werden in MELONET [37] dazu eingesetzt, Melodieumspielungen im Stil von J. Pachelbel zu erlernen und zu reproduzieren. Das Lernen von Melodievariationen stellt in seiner Komplexität gewissermaßen eine Vorstufe zum Lernen ganzer Melodien dar. Durch die Vorgabe einer zu umspielenden Melodie bzw. einer Harmoniefolge, über die variiert bzw. improvisiert werden soll, liegt über den auszufüllenden Zeitraum bereits eine musikalische Grundstruktur vor, an welcher sich der Komponist bzw. der improvisierende Musiker orientieren kann. Die Herausforderung der Aufgabe besteht darin, eine Umspielung zu finden, die zu eben diesen Vorgaben passt, d.h. insbesondere die Abhängigkeit zwischen harmonischen und melodischen Beziehungen berücksichtigt.

3.4.4 Verfahren zur Melodiegenerierung

Regelbasierte Verfahren. In [34] wird ein System zur Analyse und automatischen Erzeugung klassischer Themen am Beispiel von Klaviersonaten Mozarts und Beethovens entwickelt. Ein Thema wird zunächst in seine Bestandteile (Parameter) zerlegt, die auf Strukturen basieren, welche durch die musikalische Wahrnehmung motiviert sind. Diese Parameter werden vollständig durch natürlichsprachliche Regeln beschrieben. Dabei werden allgemeingültige (d.h. für eine Klasse von Musikstücken gültige) und themaspezifische (nur für ein Thema gültige) Regeln unterschieden. Durch Weglassen themaspezifischer Regeln lassen sich dann neue ähnliche Themen erzeugen.

In [49] werden die in der rechten Hand liegenden Melodielinien der Klavierwalzer Frédéric Chopins durch probabilistische Regeln generiert, bei der die einzelnen musikalischen Parameter wie Rhythmus, Tonhöhe, Tonlänge, Zielnoten, Hauptnoten und Nebennoten jeweils eine eigene Klasse bilden. Auf diese Weise können Melodien zweier verschiedener Kategorien (ein Figurationsmodell mit Achtelketten und ein Kantilenenmodell mit Achteln und Vorhaltsvierteln) beschrieben und erzeugt werden. Die Großform und die taktweise Harmonisierung wird von einer regulierten, kontextsensitiven und probabilistischen Grammatik vorgenommen, wobei die Großform durch die Variations- und Wiederholungsmuster der Melodielinien der rechten Hand bestimmt wird.

Improvisatorische und kompositorische lernbasierte Verfahren. Bei der Generierung von Musikstücken unterscheidet [36] zwischen improvisatorischen und kompositorischen lernbasierten Verfahren. Bei kompositorischen Verfahren ist die Reihenfolge der Veränderungen an einer Komposition nicht vorbestimmt und getroffene Entscheidungen können revidiert werden. Als improvisatorisch werden Verfahren bezeichnet, bei denen musikalische Entscheidungen ad hoc getroffen werden und nicht mehr rückgängig gemacht werden können. Diese Art sequentieller Musikkomposition ist vergleichbar mit der Vorgehensweise eines improvisierenden Organisten oder eines Jazzmusikers. Der improvisatorische Ansatz ist beispielsweise im Echtzeit-Harmonisierungssystem HarmoTheater [93] realisiert. Dort spielt ein Benutzer eine einstimmige Melodie auf einem Keyboard. Jeder Ton wird mit einer Echtzeitversion von HARMONET vierstimmig harmonisiert und sofort abgespielt. Dabei geht der

bisherige Verlauf der Harmonisierung in die Harmonisierung eines neuen Tons ein; anders als bei der Offline-Version von HARMONET kann der zukünftige Verlauf der Melodie bei der Echtzeit-Harmonisierung aber nicht berücksichtigt werden. Als improvisatorisch konzipiertes System ist HarmoTheater nicht an eine feste Tonart gebunden. HarmoTheater moduliert, indem es zu jedem gespielten Ton diejenige harmonische Interpretation sucht, die das harmonische Geschehen der näheren Vergangenheit am besten erklärt.

Vervollständigung einfacher Volksliedmelodien. Ein weiteres Beispiel für ein improvisatorisches Kompositionsverfahren ist das System MELONET II [39], in dem ein vorgegebener Melodieanfang zu einer vollständigen Melodie ergänzt wird – eine typische Aufgabe in Melodielehrbüchern [20]. MELONET II verwendet bereits ein mehrstufiges Modell zur Erkennung und Vorhersage übergeordneter Strukturen. Unüberwachtes Lernen wird zur Klassifizierung musikalischer Motive eingesetzt. Neuronale Netze sagen die Abfolge von Motiven und Motivtönen vorher.

Melodievervollständigung mit evolutionären Verfahren. Bei dem in MELONET II verwendeten improvisatorischen Verfahren wird die Komposition auf der Zeitachse streng "von links nach rechts" durchgeführt. Damit können einmal getroffene Entscheidungen nicht mehr rückgängig gemacht werden. Diese Vorgehensweise hat den Nachteil, dass die Komposition in "Sackgassen" steckenbleiben kann, in denen keine musikalisch sinnvolle Fortführung mehr möglich ist. Das stetige Verhalten der neuronalen Netze sorgt dafür, dass ähnliche harmonische bzw. melodische Vorgaben auch ähnlich umspielt werden, trotzdem werden zeitlich weiter auseinanderliegende Abschnitte der Komposition nicht direkt zueinander in Beziehung gesetzt. Im Gegensatz zum improvisatorischen Ansatz lässt sich die Melodiekomposition viel eher als "Entwicklungsprozess" auffassen: Beginnend mit einer aus einem oder mehreren Motiven bestehenden Anfangsidee werden verschiedene Abschnitte der Melodie solange verändert, bis die Melodie ihre endgültige Gestalt erhält.

Die Betrachtung des Kompositionsvorgangs als einen evolutionären Prozess bildet den Ausgangspunkt des Systems MeloGeNet [89]. Zur Entwicklung lokaler Abschnitte einer Melodie bzw. einer Harmonisierung werden evolutionäre Algorithmen verwendet. Die Bewertungs- oder Fitnessfunktion, durch welche die Evolution gesteuert wird, ergibt sich aus den Beziehungen zwischen den Strukturelementen einer Melodie, die mit Hilfe neuronaler Netze gelernt werden. Auch hierbei werden verschiedene Zeitstufen berücksichtigt, so dass sich eine mehrstufige Gesamtbewertung ergibt.

Auch wenn der in MeloGeNet entwickelte Ansatz ein vergleichsweise flexibles Verfahren zur Exploration melodischer Suchräume darstellt, unterliegt er doch einigen Beschränkungen. Zum einen sind die Merkmale, mit denen die zur Berechnung der Fitnessfunktion eingesetzten Netze trainiert werden, fest vorgegeben. Zum anderen liegt der Melodiebeschreibung ein *äquiformes* Modell zugrunde, d.h. alle betrachteten Zeitstufen (Tonebene, Harmonieebene, Motivebene) besitzen eine Segmentierung fester Länge (Achtel, Viertel, Taktdauer). Die in dieser Arbeit eingesetzte Melodiegenerierung ist eine Fortentwicklung des Systems MeloGeNet, die auf diese Beschränkungen verzichtet. Außerdem werden hier Mutationsoperatoren mit musikalischem Wissen und Strategien zur Diversifikation der evolvierten Population eingesetzt. Erstere ermöglichen eine zielgenauere Suche, letztere beheben das

in MeloGeNet ungelöste Problem des Populationskollapses, bei dem die Population nach einigen hundert Suchepochen nur noch aus Kopien des am besten bewerteten Suchpunkts besteht und eine weitere Verbesserung ausbleibt.

3.4.5 Evaluierung durch Stilerkennung

Bei den im Rahmen des Forschungsprojekts "Informationsstrukturen in der Musik" entwickelten Modellen wurden stets Musikstücke eines bestimmten Stils betrachtet, also z.B. Choräle von J. S. Bach oder Kinderliedmelodien. Insofern galt es bei der Evaluierung der Modelle darauf zu achten, inwieweit sie in der Lage waren, wichtige stilistische Eigenschaften der zugrundeliegenden Musikstücke zu berücksichtigen.

Stilerkennung. Ein objektives Gütekriterium zur Evaluierung der entwickelten Modelle bietet die Stilerkennung. Im Kontext des maschinellen Lernens besteht die Methode darin, mehrere neuronale Netze mit verschiedenen Stilen zu trainieren. Ein unbekanntes Musikstück wird dann demjenigen "neuronalen Stilexperten" zugeordnet, der die größte Erkennungsgüte, d.h. die größte Anzahl korrekt klassifizierter Muster pro Musikstück erzielt.

In [37] werden mit verschiedenen Musikstilen trainierte neuronale Stilexperten dahingehend untersucht, ob sie in der Lage sind, unbekannte Choräle dem richtigen Harmonisierungsstil (J. S. Bach, S. Scheidt, M. Reger) zuzuordnen. Es zeigt sich, dass das Bach- und das Scheidt-Netz alle Choräle perfekt voneinander unterscheiden können, d.h. alle Testchoräle werden dem richtigen Komponisten zugeordnet. Schlechter ist hingegen die Erkennungsgüte des Reger-Netzes. Hier werden von 6 Chorälen nur 4 richtig klassifiziert, die anderen zwei werden fälschlicherweise J. S. Bach zugeordnet.

Der Ansatz differenziert nicht zwischen Komposition und Stilerkennung, d.h. es werden für beide Problemstellungen die gleichen Merkmale verwendet, die nur den linken Kontext zur Bestimmung des aktuellen Harmonieübergangs berücksichtigen, nicht aber zeitlich entferntere Beziehungen oder globale Merkmale.

Stilanalyse. Mit der Methode der *vergleichenden Stilanalyse* [38] wird ein theoretisches Fundament gelegt, um die gelernten Musikstrukturen auf stilunabhängige und stiltypische Wendungen hin zu untersuchen. Durch Vergleich der musikalischen Erwartungen der einzelnen neuronalen Stilexperten lassen sich stilabhängige oder stilunabhängige Passagen in den Stücken aufspüren. Bei hinreichend umfangreichen Datenmengen können diese Passagen automatisch in eine Menge von probabilistischen Regeln transformiert werden.

Durch die Anwendung der vergleichenden Stilanalyse wurde unter anderem ein Choral aus dem Bach-Choralbuch gefunden, den die neuronalen Stilexperten nicht J. S. Bach, sondern einem frühbarocken Stil zuordneten, und der nach musikwissenschaftlichen Untersuchungen mit großer Wahrscheinlichkeit von Bach nicht selbst komponiert, sondern aus einem älteren Choralbuch übernommen wurde.

Experimente mit Versuchspersonen. Neben der Bewertung durch Experten bieten auch Untersuchungen mit Versuchspersonen die Möglichkeit, die Leistungsfähigkeit der entwickelten Modelle zu testen.

Das mit Unterstützung des *Heinz Nixdorf MuseumsForums (HNF)* in Paderborn entwickelte didaktische, interaktive Exponat "Komponieren wie Bach – künstlich oder künstlerisch?" [99] erlaubt es, die Leistungsfähigkeit eines Systems, das musikalische Strukturen mittels neuronaler Netze lernt, an einer sehr großen, zufällig zusammengesetzten Personengruppe zu evaluieren. In einem musikalischen Turingtest, der einen Teil des Exponats "Komponieren wie Bach" bildet, hatten die Besucher die Aufgabe, zwischen Originalchorälen von J. S. Bach und automatisch mit dem neuronalen System HARMONET harmonisierten Chorälen zu unterscheiden. Dabei wurden dem Besucher Paare aus je einer Original- und einer Computerkomposition vorgelegt, wobei jeder Choral als Notentext sichtbar war und die Aufnahme eines Organisten beliebig oft abgespielt werden konnte. Die Aufgabe der Besucher bestand darin, die Originalkomposition zu identifizieren.

2744 Besucher haben insgesamt 6325-mal den musikalischen Turingtest absolviert. Besucher, die sich als "Laien" einschätzten, klassifizierten 57% der Choräle korrekt. Bei den Besuchern, die sich als "Experten" einschätzten, betrug die Trefferquote 61%. Die Verbesserung der Erfolgsquote durch die Stilkennis der Besucher im Vergleich zu einer zufälligen gleichverteilten Auswahl mit 50% Erfolgsquote machte also 7% bzw. 11% der getesteten Choralpaare aus. Das neuronale System HARMONET ist also in der Lage, Choräle zu komponieren, die von vielen Besuchern, die den musikalischen Turingtest absolviert haben, nicht zuverlässig von Originalchorälen unterschieden werden können. Weiterhin ist interessant, dass die Trefferquote bei den einzelnen Choralpaaren stark zwischen 35% und 77% variiert, was bei einer Stichprobe von mehreren Tausend Rateversuchen eine signifikante Streuung darstellt. Der musikalische Turingtest ist also ein geeignetes Instrument, um eine Rangliste für die Stiltreue künstlich erzeugter Musikstücke zu erstellen.

Kapitel 4

Repräsentation musikalischer Strukturen

Ziel und Motivation. In diesem Kapitel wird eine Repräsentation für diskrete Zeitreihen entwickelt, die die Aufbereitung von Musikstücken für das maschinelle Lernen unterstützt. Die entwickelte Repräsentation ermöglicht es, ein Musikstück und seine Analysen aus verschiedenen Blickwinkeln zu beschreiben. Die Generierung solcher Beschreibungen mit Hilfe von *Operatorgraphen* ist dann Thema von Kapitel 5. Operatorgraphen sind Funktionen, die Musikstücke transformieren und Lernmuster generieren. Ziel dieses Kapitels ist es, Definitions- und Wertebereiche für die Operatorgraphen zu definieren, die auf das Lernen diskreter Zeitreihen zugeschnitten sind.

Die Grundidee bei der Definition der Datenstrukturen besteht darin, hierarchisch aufgebaute *Objekte* mit einer generativen Grammatik zu erzeugen. Die von der Grammatik erzeugte Sprache zerfällt in hierarchisch angeordnete Teilsprachen, die die Grundlage für die Definitions- und Wertebereiche der Operatoren bilden.

Vorgehensweise. Das Ergebnis dieses Kapitels ist die Definition einer speziellen generativen Grammatik, der *Erweiterbaren Zeitreihengrammatik*. Um die verschiedenen Eigenschaften einer Erweiterbaren Zeitreihengrammatik transparent zu machen, werden zunächst drei vorbereitende Grammatiken definiert: die *Objektgrammatik*, die *Typgrammatik* und die *Templategrammatik*. Die Erweiterbare Zeitreihengrammatik vereinigt die Eigenschaften von Typ- und Templategrammatik, die ihrerseits spezielle Objektgrammatiken sind (Abbildung 4.1).

Objektgrammatik. In Abschnitt 4.2.2 werden Objektgrammatiken definiert, die baumförmige *Objekte* erzeugen. Die inneren Knoten der Objekte sind benannt, die Blattknoten bestehen in der Regel aus einem einfachen Datum wie einer Zahl oder Zeichenkette. Damit eine Objektgrammatik nur baumförmige Objekte erzeugt, wird ihre Struktur durch *Wohlgeformtheitsregeln* festgelegt.

Im Gegensatz zu gewöhnlichen Grammatiken ist die von einer Objektgrammatik erzeugte Sprache zusätzlich in eine Hierarchie von Teilsprachen aufgeteilt, die entsteht, wenn man

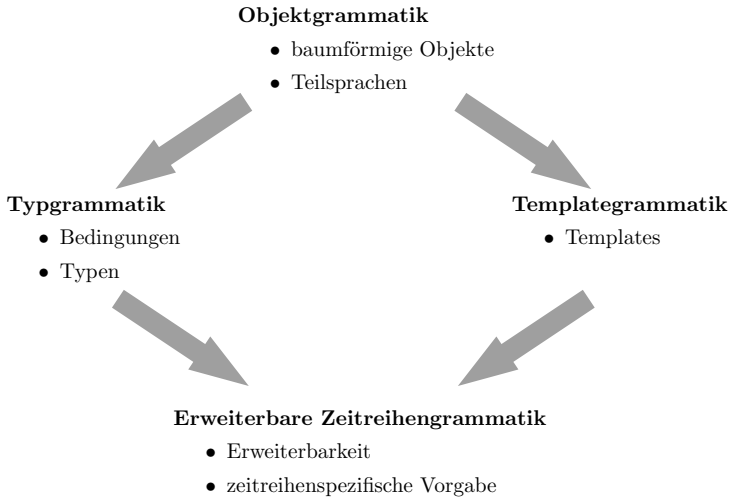


Abbildung 4.1: Generative Grammatiken in Kapitel 4

Objekte mit gleich benanntem Wurzelknoten zu einer *Teilsprache* der Grammatik zusammenfasst.

Typgrammatik. Um Teilsprachen einer Objektgrammatik gezielt auf wenige sinnvolle Werte einzuschränken, werden in einer Typgrammatik Bedingungen eingeführt, die die Baumstruktur eines Objekts festlegen. Beispielsweise ist es sinnvoll, bei der Definition eines musikalischen Takts als Nenner nur Zweierpotenzen zuzulassen. Eine Typgrammatik ist eine Objektgrammatik, bei der die Teilsprachen durch Bedingungen auf *Typen* eingeschränkt werden.

Templategrammatik. Bei der Analyse musikalischer Strukturen begegnet man häufig musikalischen Objekten, die eine variable Anzahl von Objekten desselben Typs enthalten. Ein Beispiel sind die in einer Phrase oder in einem musikalischen Motiv enthaltenen Töne. Um solche Strukturen darstellen zu können, werden parametrisierte Teilsprachen ähnlich den Containerklassen in objektorientierten Programmiersprachen eingeführt. Um parametrisierte Teilsprachen zu erzeugen, wird im ersten Schritt eine parametrisierte Version einer Objektgrammatik definiert – ein sogenanntes *Grammatiktemplate*. Dieses Grammatiktemplate wird zu einer *Templategrammatik* expandiert, indem man alle erlaubten Ersetzungen für die Parameter vornimmt.

Erweiterbare Zeitreihengrammatik. Aus der Typ- und der Templategrammatik wird im letzten Schritt die Erweiterbare Zeitreihengrammatik zusammengesetzt. Während die Typ- und die Templategrammatiken weder zeitreihen- noch anwendungsspezifisch sind, unterscheidet man bei der Erweiterbaren Zeitreihengrammatik eine zeitreihenspezifische Teilgrammatik, die durch einen anwendungsspezifischen Teil ergänzt wird. Diese Zweiteilung ermöglicht, zeitreihenspezifische Operatoren unabhängig vom untersuchten Problembereich zu definieren und erleichtert zudem durch den modularen Aufbau die Anpassung des Ansatzes an

neue Problemstellungen. Dadurch lässt sich der Ansatz auch auf andere nicht-musikalische Anwendungen wie etwa Finanzzeitreihen übertragen.

Musikalische Strukturen. Mit den Objekten einer Erweiterbaren Zeitreihengrammatik werden musikalische *Kompositionen* repräsentiert. Die Repräsention einer Komposition setzt sich aus mehreren *Ansichten* zusammen, die jeweils einen bestimmten Aspekt einer Komposition widerspiegeln, etwa die Tonhöhen einer Stimme, eine motivische Analyse oder die Phrasenstruktur. Jede Ansicht setzt sich aus Objekten eines Typs zusammen. Die Repräsentation eignet sich nicht nur zur Darstellung (einstimmiger) Melodien und ihrer Analysen, sondern auch zur Darstellung beliebiger mehrstimmiger Musikstücke, indem man für jede Stimme eine eigene Ansicht definiert. Die Darstellung von Kompositionen mit Hilfe von Ansichten ist die grundlegende Datenstruktur, auf der Operatorgraphen operieren.

4.1 Musikalische Zeit

Ein Musikstück besteht, wenn es erklingt, aus zeitlich angeordneten Klangereignissen. Während man ein Bild zu einem Zeitpunkt als Ganzes ansehen und dann in selbstgewählter Reihenfolge Details genauer betrachten kann, hört man von einem Musikstück immer nur den gegenwärtig erklingenden Ausschnitt. Die Abfolge der Klangereignisse ist vom Zuhörer nicht beeinflussbar. Das Gedächtnis erlaubt jedoch, das Gehörte durch Erinnerung und Erwartung zu strukturieren. Wenn man verstehen möchte, wie man ein Musikstück hörend strukturiert, kann man sich zunächst fragen, wie die Zeitwahrnehmung funktioniert.

Newton unterscheidet zwischen *objektiver* und *subjektiver Zeit*, wobei er die objektive Zeit als absolut und gleichmäßig fließend und die subjektive Zeit als ein durch die Sinnesorgane vermitteltes Maß der absoluten Zeit beschreibt [69, S.15]. Um die subjektive Zeitwahrnehmung zu charakterisieren, ermittelt Pöppel experimentell verschiedene kognitive Konstanten, darunter die Dauer der subjektiven Gegenwart [69]. Was sich in diesem Zeitfenster von 2.5-3 Sekunden abspielt, wird als gegenwärtig und potentiell als Einheit wahrgenommen. Ein längerer Strom von Ereignissen wird hingegen segmentiert, abstrahiert und zu symbolischen Wahrnehmungsgestalten zusammengefasst.

Zeitrepräsentation. Von diesen Beobachtungen ausgehend definieren wir die Zeitdarstellung im zu entwickelnden Modell. Grundlage der Zeitdarstellung ist ein *Zeitgitter*, das aus kleinsten Zeiteinheiten fester Länge besteht. Es entspricht Newtons “objektiver Zeit”.

Die Position und die Dauer eines musikalisches Ereignisses werden mit einem Intervall auf dem Zeitgitter – einem sogenannten *Segment* – beschrieben. Eine *Segmentierung* eines Musikstücks ist eine Folge von Segmenten, die Einsatzzeiten und Dauern von musikalisch zusammengehörenden Ereignissen (z.B. Phrasen oder Motiven) eines Musikstücks wiedergeben. Die Begriffe Segment und Segmentierung werden in Anlehnung an die Aufgabe der Melodiesegmentierung gewählt, bei der eine Melodie in musikalisch sinnvolle Einheiten zerlegt werden soll.

Die Definition eines Segments in Definition 4.1 ist vorläufig und beinhaltet nur die Zeitinformation, nicht aber die musikalische Interpretation eines Ereignisses. Die Definition wird benötigt, um die Zeitdarstellung des hier entwickelten Ansatzes zu diskutieren. In der

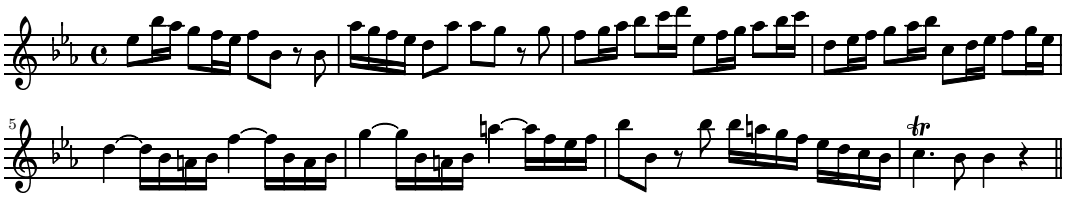


Abbildung 4.2: Antonio Vivaldi: Largo (aus: Vier Jahreszeiten – Winter, op. 8, Nr. 4)

endgültigen Version (Definition 4.10) wird auch das musikalische Ereignis selbst als Teil eines Segments betrachtet. Fasst man Segmente zusammen, die den gleichen *Typ* musikalischer Ereignisse (etwa Motive in einer Stimme) enthalten, so erhält man die bereits erwähnte Ansicht eines Musikstücks.

Nun werden einige zeitbezogene Grundbegriffe eingeführt, die zur Definition einer Repräsentation von Zeitreihen im nächsten Abschnitt benötigt werden. Die Zeitachse bildet den Übergang vom Zeitkontinuum in ein diskretes Modell.

Definition 4.1 (Zeitbezogene Grundbegriffe)
Eine *diskrete Zeitachse* $\mathbb{T}_r$ ist definiert als

$$\mathbb{T}_r = \mathbb{Z}[r] = \{zr | z \in \mathbb{Z}\},$$

wobei $r \in \mathbb{Q}^{>0}$ die *Auflösung* der Zeitachse bezeichnet. Ein *Segment* ist ein Paar $s = (t, d)$, das aus einer *Einsatzzeit* $t \in \mathbb{T}_r$ und einer *Dauer* $d \in \mathbb{T}_r^+ = \{nr | n \in \mathbb{N}_0\}$ besteht. Eine endliche Menge $S \subset \mathbb{T}_r \times \mathbb{T}_r^+$ von Segmenten heißt *Segmentierung*.

* * *

Beispiel. Abbildung 4.2 zeigt das Thema des Largo aus dem "Winter" der "Vier Jahreszeiten" (op. 8, Nr. 4) von Antonio Vivaldi. Ein Segment kann die Zeitinformation eines beliebigen Merkmals der Melodie beschreiben. Wenn man im Beispiel eine Zeitachse mit Sechzehntelauflösung wählt, ist der ganzen Melodie das Segment $(0, 124)$ und der ersten Phrase bis zur ersten Achtelpause das Segment $(0, 12)$ zugeordnet. Die Noten des ersten Takts haben die Segmentierung

$$\{(0, 2), (2, 1), (3, 1), (4, 2), (6, 1), (7, 1), (8, 2), (10, 2), (14, 2)\}.$$

Die erste Achtelnote *es* setzt zum Zeitpunkt 0 ein und dauert zwei kleinste Zeiteinheiten, d.h. zwei Sechzehntel: $(0, 2)$. Zur Einsatzzeit 2 folgt darauf eine Sechzehntel *b* mit einer Dauer von einer Zeiteinheit: $(2, 1)$, usw. Die Lücke zwischen dem vorletzten und dem letzten Segment $(10, 2)$ und $(14, 2)$ im ersten Takt stellt die Achtelpause im Notenbeispiel dar. In diesem Beispiel wurde $r = 1$ gesetzt und festgelegt, dass die kürzeste Dauer r einer Sechzehntel entsprechen soll.

Im folgenden wird immer $r = 1$ gesetzt und r weggelassen. Aus praktischer Perspektive kann man auf die Auflösung r aber nicht verzichten, weil sie erlaubt, die darstellbaren Einsatzzeiten und Dauern zu verändern, ohne zuvor berechnete Segmentierung anpassen zu müssen.

	Rhythmus	Segmentierung	Auflösung	Zeitgitter
1:		$(0,1),(1,1)$	$r = 1$	$\mathbb{T}_r = \mathbb{Z}$
2:		$(0,\frac{1}{2}),(\frac{1}{2},\frac{1}{2}),(1,1)$	$r = \frac{1}{2}$	$\mathbb{T}_r = \mathbb{Z}[\frac{1}{2}]$
3:		$(0,\frac{1}{4}),(\frac{1}{4},\frac{1}{4}),(\frac{1}{2},\frac{1}{4}),(\frac{3}{4},\frac{1}{4}),(1,1)$	$r = \frac{1}{4}$	$\mathbb{T}_r = \mathbb{Z}[\frac{1}{4}]$

Tabelle 4.1: Beispiel für das zeilenweise Einlesen feiner werdender Rhythmen. Wenn ein neuer kürzester Notenwert gelesen wird, wird die Auflösung r angepasst. Frühere Segmentierungen bleiben trotzdem gültig.

Auf diese Weise kann man große Mengen von Musikstücken in nur einem Durchlauf einlesen, selbst wenn die kürzeste benötigte Dauer nicht von vorneherein bekannt ist. Tabelle 4.1 zeigt dies an drei Rhythmen. Der kürzeste Wert von Rhythmus 1 ist eine Halbe; man startet mit der Auflösung $r = 1$ und berechnet die Segmentierung in der zweiten Spalte der ersten Zeile. Die Dauer 1 ist wird von nun an als Halbe interpretiert. Nun bestimmt man den kürzesten rhythmischen Wert des zweiten Rhythmus, eine Viertelnote, passt die Auflösung zu $r = \frac{1}{2}$ an und kann die Segmentierung bezüglich des neuen Zeitgitters $\mathbb{Z}[\frac{1}{2}]$ berechnen. Die Segmentierung des ersten Rhythmus bleibt trotz der Veränderung der Auflösung gültig. Diese Vorgehensweise ermöglicht es, einen bestehenden Corpus musikalischer Stücke jederzeit zu ergänzen, ohne die Repräsentation der bereits vorhandenen Stücke ändern zu müssen.

Vergleich von ereignisorientierter und gerasterter Zeitdarstellung. Segmentierungen bilden die zeitliche Grundlage der hier entwickelten musikalischen Repräsentation. Sie bestehen aus Segmenten, die unterschiedliche Längen haben, einander überlappen können und nicht direkt aneinander anschließen müssen. Da jedes Segment sich auf ein musikalisches Ereignis bezieht, handelt es sich um eine *ereignisorientierte Zeitdarstellung*. Die *gerasterte Zeitdarstellung* orientiert sich ebenfalls am Zeitgitter, stellt ein musikalisches Ereignis jedoch nicht mit Hilfe eines Zeitintervalls, sondern durch eine Folge miteinander verbundener kürzester Intervalle des Zeitgitters dar [82, 36].

Während die Rasterdarstellung durch die Verwendung von Zeitintervallen einer einzigen

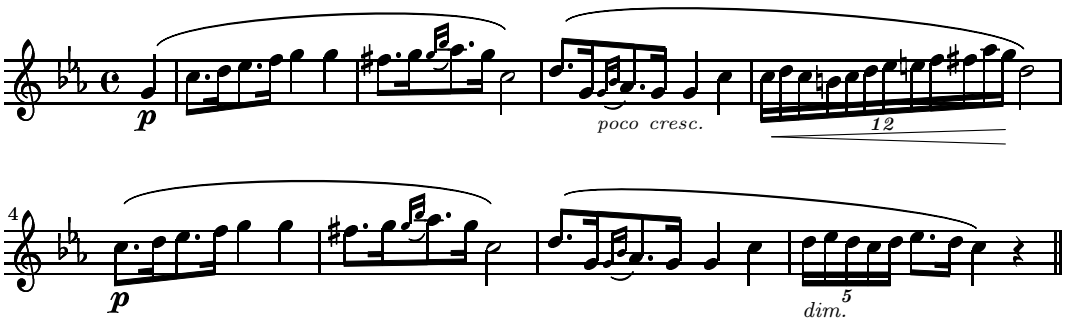


Abbildung 4.3: Frédéric Chopin: Andante

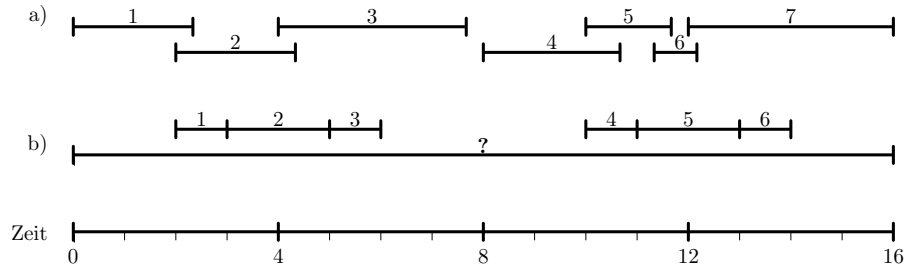


Abbildung 4.4: Lexikographisch geordnete Segmente.
a) Überlappende, leicht zu ordnende Segmente
b) Segmente ohne eindeutige, zeitlich lineare Ordnung. Wie soll das lange Segment in die Folge der kurzen Segmente eingeordnet werden?

Dauer leichter zu handhaben ist und sich zur Modellierung von Musikstilen mit einer einfachen rhythmischen Struktur wie etwa Kinderlieder oder Choralmelodien eignet, bietet die ereignisorientierte Zeitdarstellung Vorteile bei der Modellierung allgemeinerer Melodiestile wie z.B. die Möglichkeit, überlappende Ereignisse darzustellen. Wenn ein Segment *einem* musikalischen Ereignis entspricht, gibt es zudem die Nachbarschaften zwischen musikalischen Ereignissen besser wieder als die Rasterdarstellung und ermöglicht dadurch einem maschinellen Lernverfahren eine bessere Einordnung zeitlicher Zusammenhänge. Desweiteren kann eine ereignisorientierte Zeitdarstellung beispielsweise bei der Erzeugung von Lernmustern effizienter durchlaufen werden, wenn sehr unterschiedliche Dauern in einem Musikstück auftreten. Ein Beispiel zeigt den Unterschied. Die Melodie von Frédéric Chopin in Abbildung 4.3 besteht aus 64 Tönen, wenn man die Vorschlagnoten berücksichtigt. Das Zeitgitter muss zwar mit einer Auflösung von 240-tel Noten[1] sehr fein gewählt werden, um alle Notenwerte bis auf die Vorschläge darstellen zu können, doch braucht man nur 64 Schritte, um die Melodie zu durchlaufen. Wenn man bei einer Rasterdarstellung das gleiche Gitter verwendet, hat jedes Segment die kürzeste vorgesehene Dauer von einer 240-tel Note, so dass 1920 (= 8 Takte*240 kleinste Einheiten) Schritte zum Durchlaufen der Melodie erforderlich sind.

Vorschlagnoten sind ein Beispiel für musikalische Ereignisse, die sich nicht auf natürliche Weise in eine zeitliche Anordnung integrieren lassen. Repräsentiert man sie mit einer kurzen Dauer, die von der vorhergehenden Dauer abgezogen wird, so ändert sich der symbolische rhythmische Wert dieser Note willkürlich. Repräsentiert man Vorschläge überlappend mit ihren Nachbarn, so wird eine einstimmige Melodie mehrstimmig. Weist man den Vorschlägen keine Dauer zu, so spiegelt die Repräsentation nicht mehr den Höreindruck. Es handelt sich dabei jedoch um ein Problem, das bei jeder Einbettung von Notentext in einen Zeitstrahl auftritt und nicht spezifisch für den hier gewählten Ansatz ist.

Anordnung von Segmentierungen. Die Segmente einer Segmentierung können einander

[1] Es sollen Sechzehntel, Viertelsextolen und Viertelquintolen gleichzeitig dargestellt werden. Eine Viertel muss in das kleinste gemeinsame Vielfache von vier (für die Sechzehntel), sechs und fünf zerlegt werden können, also in 60 Teile. Daher muss man ein Zeitgitter verwenden, dessen kleinste Einheit einer 240-tel Note entspricht. Die Vorschlagnoten haben eine unbestimmte Dauer und werden in der Rechnung nicht berücksichtigt.

überlappen, zur gleichen Einsatzzeit beginnen oder auch ineinanderliegen. Wenn man nun einen Algorithmus zur Berechnung von Merkmalen oder zur Erzeugung von Lernmustern angeben will, kann das Ergebnis davon abhängen, in welcher Reihenfolge die Segmente einer Segmentierung durchlaufen werden. Dies ist z.B. der Fall, wenn die Berechnung eines Merkmals für ein gegebenes Segment von dessen Vorgänger in der Segmentierung abhängt. Welches Segment soll als Vorgänger angesehen werden, wenn es mehrere Segmente mit gleicher Einsatzzeit, aber unterschiedlicher Dauer gibt?

In vielen Fällen liefert die lexikographische Ordnung auf den Segmenten eine plausible Sortierung. Hier wird die Einsatzzeit als erstes und die Dauer als zweites Ordnungskriterium verwendet.

Definition 4.2 (Lexikographische Ordnung von Segmenten)
Gegeben seien Segmente $s_1 = (t_1, d_1), s_2 = (t_2, d_2) \in \mathbb{T} \times \mathbb{T}^+$. Das Segment s_1 ist *lexikographisch kleiner* als s_2 ("$s_1 < s_2$"), wenn

$$(t_1 < t_2) \vee ((t_1 = t_2) \wedge (d_1 < d_2))$$

und *lexikographisch größer* als s_2 ("$s_1 > s_2$"), wenn

$$(t_1 > t_2) \vee ((t_1 = t_2) \wedge (d_1 > d_2)).$$

* * *

Abbildung 4.4a zeigt Segmente, bei denen die lexikographische Ordnung eine musikalisch sinnvolle Reihenfolge liefert. Nacheinander beginnende, nur wenig überlappende Ereignisse würden sicher auch von Versuchspersonen nach ihren Einsatzzeiten geordnet werden. Wenn hingegen in einander enthaltene Klangereignisse wie in Abbildung 4.4b in eine Reihenfolge gebracht werden sollen, gibt es keine befriedigende lineare Ordnung, weil die Ereignisse nicht nur ein wenig am Anfang oder am Ende überlappen, sondern gleichzeitig stattfinden. Doch auch in solchen Fällen will man den Segmenten eine Ordnung geben, da z.B. ein Verfahren zur Merkmalsberechnung ein nicht-deterministisches Ergebnis liefern könnte, wenn man ungeordnete Segmente in beliebiger Reihenfolge bearbeitet. Die lexikographische Ordnung auf den Segmenten stellt eine mögliche Ordnung der Segmente her und dient darüber hinaus als technisches Hilfsmittel, um musikalisch sinnvollere Relationen zwischen in einander enthaltenen Segmenten aufzubauen.

Die Anordnung von Segmenten mit gleicher Einsatzzeit und Dauer innerhalb einer Segmentierung wurde in diesem Abschnitt nicht angesprochen, weil sie bei den weiteren Definitionen musikalischer Strukturen nicht zu Konflikten führt. Wenn gleichzeitige Segmente musikalische Informationen unterschiedlichen Typs enthalten, gehören sie zu verschiedenen Ansichten eines Musikstücks und damit auch zu verschiedenen zeitlichen Abstraktionen, d.h. Segmentierungen der Ansichten. Wenn gleichzeitige Ereignisse wie z.B. mehrere Tonhöhen in einem Musikstücks auftreten, verteilt man sie abhängig von der untersuchten Fragestellung auf mehrere Ansichten oder verwendet einen Symboltyp, der mehrere Ereignisse darstellen kann.

4.2 Musikalische Objekte

Nachdem wir in Abschnitt 4.1 eine Repräsentation für die musikalische Zeit kennengelernt haben, wenden wir uns der musikalischen Information selbst zu. Die Zeitinformation wird nun mit *musikalischen Symbolen* wie z.B. einer Tonhöhe, einer Taktart, einem Motiv oder einer musikalischen Phrase verknüpft. Zunächst werden anhand eines Beispiels die Schwierigkeiten skizziert, die bei der Strukturierung von musikalischem Problemwissen auftreten können.

4.2.1 Beispiel zur Strukturierung von Problemwissen

Tabelle 4.2 zeigt eine Auswahl musikalischer Symbole, die sowohl der musikwissenschaftlichen Literatur als auch digitalen Musikrepräsentationen entnommen sind. Mit Hilfe dieser Symbole wird das bereits erwähnte Largo von A. Vivaldi in Abbildung 4.5 auf mehrere Weisen beschrieben. Die Beschreibung gliedert sich in Schichten von Segmenten, denen jeweils ein musikalisches Symbol zugeordnet ist. Eine Schicht wird als *Ansicht* der Komposition bezeichnet und zeigt einen bestimmten Aspekt der Melodie, z.B. Tonhöhen, Intervalle, Konturen, Motive und Phrasen. Für die Definition von Tonhöhen gibt es verschiedene Möglichkeiten, von denen hier drei herausgegriffen werden. Zwei wesentliche Unterscheidungsmerkmale bei der Repräsentation einer Tonhöhe sind die zugrundegelegte Notenskala und Oktavlage des Tons. Die "enharmonisch äquivalenten Tonhöhen" stellen Töne gleich dar, die wie z.B. *fis* und *ges* durch Alteration benachbarter Stammtöne gewonnen werden und in temperierter Stimmung zusammenfallen (in Tabelle 4.2 mit Angabe der Oktavlage). Zum Vergleich ist in der Tabelle auch der Tonvorrat für die "chromatischen Tonhöhenklassen" angegeben, bei denen die enharmonische Verwandtschaft nicht berücksichtigt wird (hier ohne Angabe der Oktavlage). Eine Tonhöhenklasse fasst alle Tonhöhen im Oktavabstand zusammen und abstrahiert so von der Oktavlage. Die MIDI-Darstellung stammt aus dem MIDI-Dateiformat und stellt die Tonhöhen wie die erste Repräsentation enharmonisch äquivalent unter Berücksichtigung der Oktavlage dar. Die MIDI-Nummern $0 - 127$ entprechen der Teilmenge C_3-g^6 der enharmonischen äquivalenten Tonhöhen. Die pentatonische Tonhöhendarstellung stützt sich für das diskutierte Beispiel auf den Tonvorrat {es,f,g,b,c} und ist, wie die Ansicht der pentatonischen Tonhöhen in Abbildung 4.5 zeigt, nicht in der Lage, alle Töne des Beispiels wiederzugeben.

Intervalle geben den Abstand zwischen zwei Tönen bezüglich der Stammtonskala *es-f-g-as-b-c-d* einer Tonart an (hier: *es*-Dur), wobei ein Modifikator (rein, vermindert, übermäßig) die genaue Größe des Intervalls festlegt. Es wird zwischen einer übermäßigen Quarte (z.B. *c-fis*) und einer verminderten Quinte (z.B. *c-ges*) unterschieden, die zwar beide 5 Halbtonschritte umfassen, deren Abstand auf der Stammtonskala sich aber unterscheidet.

In der Intervallansicht des Beispiels wurde die Richtung der Intervalle nicht berücksichtigt. Dies geschieht in der Konturansicht, die nur die Richtung der auftretenden Intervalle angibt. Die Werte -1, 0 und 1 bezeichnen jeweils fallende, gleichbleibende und aufsteigende Intervalle. Die Motivklassenansicht zeigt die gebräuchliche Notation ähnlicher Motivklassen durch Buchstaben A, A', A", ... Bei der hier dargestellten Motivanalyse wurden nur die rhythmischen Werte der Töne berücksichtigt, nicht aber ihre Tonhöhen. Die Phrasenansicht besteht nur aus Segmenten, denen keine musikalischen Symbole zugeordnet wurden.

Musikalische Symbole	**Interpretation**
{ . . . , c1, cis1, d1, es1, e1, . . . }	enharmonische äquivalente Tonhöhen (mit Oktavangabe)
{ 0, 1, . . . , 127}	Tonhöhen als MIDI-Nummern
{ . . . , ceses, ces, c, cis, cisis, . . . , deses, des, d, dis, disis, . . . }	chromatische Tonhöhenklassen (ohne Oktavangabe)
{ es, f, g, b, c}	pentatonische Tonhöhenklassen mit Grundton es
{ es, f, g, as, b, c, d}	Stammtonskala in *es*-Dur
{ Prim, Sekunde, Terz, . . . , Oktave}	Intervalle
{ r1, k2, g2, k3, g3, r4, ü5, v5, r5, . . . , r8}	Abgekürzte Intervalle mit Modifikator: reine Prim, kleine Sekunde, große Sekunde, kleine Terz, große Terz, reine Quarte, übermäßige Quarte, verminderte Quinte, reine Quinte, . . . , reine Oktave
{ 1, 0, -1}	melodische Konturen (auf, gleichbleibend, ab)
{ A, A', A", . . . , B, B', B",. . . , C, C', C", . . . D, D', D", . . . }	Motivklassen
kein Symbol	nichtklassifiziertes Motiv
kein Symbol	Phrase
{ T, D, S, Tp, Sp, Dp, DD, DP, TP, d, VTp, SS}	Harmonische Funktionssymbole
{ C, Cis, Cisis, . . . , Des, D, Dis,. . . , c, cis, cisis, . . . , des, d, dis, . . . }	Tonarten

Tabelle 4.2: Beispiele für musikalische Symbole

Die Harmonieansicht enthält die Harmonisierung der Melodie von Vivaldi. In der Ansicht der "Harmonischen Funktionen" werden dieselben Harmonien auf das tonale Zentrum der Melodie bezogen.

Zwischen den Symbolen in Tabelle 4.2 gibt es sowohl musikalisch motivierte Beziehungen als auch Scheinbeziehungen. Man findet in der Tabelle:

- Überlappungen der unterschiedlichen Symbolmengen. Die Stammtonskala ist eine Teilmenge der chromatischen Tonhöhenklassen.

- Gleiche Symbole, die unterschiedliche musikalische Gegenstände bezeichnen: Die Konturbezeichner 0 und 1 lassen sich z.B. auch als MIDI-Nummer interpretieren. Der Buchstabe D steht sowohl für ein harmonisches Funktionssymbol als auch für eine Motivklasse.

- Gleiche Symbole, die verwandte musikalische Gegenstände bezeichnen: Für Tonarten und Tonhöhen werden teilweise die gleichen Bezeichner verwendet. Die Symbole haben eine verwandte, aber nicht dieselbe Bedeutung: Der Grundton einer Tonart ist eine Tonhöhe, die Groß- oder Kleinschreibung gibt bei den Tonarten an, ob es sich um eine Dur- oder Molltonart handelt, bei den Tonhöhen hat sie keine Bedeutung.

- Identische musikalische Gegenstände, die unterschiedlich bezeichnet werden: Absolute Tonhöhen werden sowohl durch MIDI-Nummer als auch durch enharmonische äquivalente Tonhöhen angegeben.
- Symbole, die Beziehungen zwischen musikalischen Gegenständen ausdrücken: Ein Intervall stellt zum Beispiel den Abstand zwischen zwei Tonhöhen dar. Das Symbol "Terz" abstrahiert von den Tonhöhen, aus denen die Terz berechnet wurde. Dass eine Beziehung dargestellt wird, ist ohne Hintergrundwissen nicht aus den Bezeichnungen der Intervalle erkennbar.
- Numerisch und kategorial skalierte Darstellung musikalischer Gegenstände: Die Kontur gibt die Steigung eines Intervalls numerisch an. "1" steht etwa für ein steigendes Intervall, "-1" für ein fallendes. Die harmonischen Funktionen dagegen bilden Kategorien, die sich nicht linear anordnen lassen.
- Bezeichner, die sich auf die Angabe von Zeitinformation beschränken: Phrasen werden nur durch ihre zeitliche Position und ihre Dauer beschrieben, ohne auf zusätzliche musikalische Information zu verweisen. Die Zusammenfassung der Segmente zu einer "Phrasenansicht" drückt implizit aus, dass die Zeitangaben als Phrase interpretiert werden sollen.

In der hier entwickelten Repräsentation werden *Typen* eingeführt, die musikalische Symbole mit der gleichen Semantik zusammenfassen und die es ermöglichen, die beschriebenen Mehrdeutigkeiten aufzulösen.

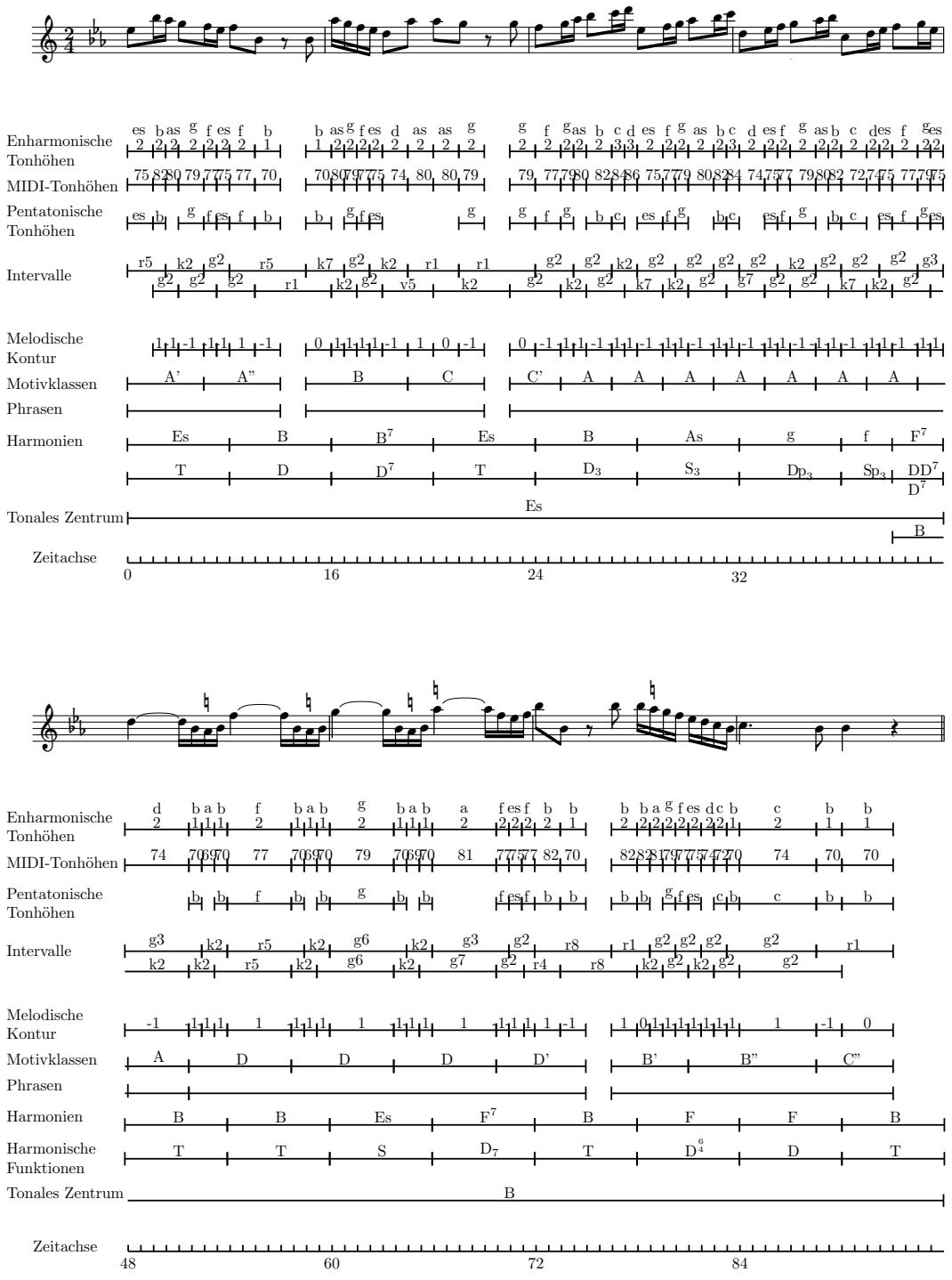


Abbildung 4.5: Ansichten für das Largo von A. Vivaldi (op. 8, Nr. 4)

Tabelle 4.3: Vergleich der Grundmengen

Objektgrammatik	**Typgrammatik**	**Template-Grammatik**	**Erweiterbare Zeitreihengrammatik**
$\Sigma_{NTS} = \{A-Z,\ a-z,\ 0-9\}$ $\Sigma_{TB} = \{\mathbf{A-Z},\ \mathbf{a-z},\ \mathbf{0-9}\}$ $\Sigma_{Str} = \{\mathrm{A-Z},\ \mathrm{a-z},\ 0-9\}$ $\Sigma_I = \{\ (\ )\ ;\ ,\ \}$ $\Sigma_W = \Sigma^*_{Str} \cup \mathbb{Q}$	$\Sigma_{NTS} = \{A-Z,\ a-z,\ 0-9\}$ $\Sigma_{TB} = \{\mathbf{A-Z},\ \mathbf{a-z},\ \mathbf{0-9}\}$ $\Sigma_{Str} = \{\mathrm{A-Z},\ \mathrm{a-z},\ 0-9\}$ $\Sigma_I = \{\ (\ )\ :\ ;\ ,\ \}$ $\Sigma_{Att} = \{a-z\}$ $\Sigma_W = \Sigma^*_{Str} \cup \mathbb{Q}$	$\Sigma_{NTS} = \{A-Z,\ a-z,\ 0-9\}$ $\Sigma_{TB} = \{\mathbf{A-Z},\ \mathbf{a-z},\ \mathbf{0-9}\}$ $\Sigma_{Str} = \{\mathrm{A-Z},\ \mathrm{a-z},\ 0-9\}$ $\Sigma_I = \{\ (\ )\ \langle\ \rangle\ :\ ;\ ,\ \preceq\ \}$ $\Sigma_{Par} = \{A-Z\}$ $\Sigma_W = \Sigma^*_{Str} \cup \mathbb{Q}$ $\Sigma = \Sigma_{NTS} \cup \Sigma_{TB} \cup \Sigma_I \cup \Sigma_W \cup \Sigma_{Par}$	$\Sigma_{NTS} = \{A-Z,\ a-z,\ 0-9\}$ $\Sigma_{TB} = \{\mathbf{A-Z},\ \mathbf{a-z},\ \mathbf{0-9}\}$ $\Sigma_{Str} = \{\mathrm{A-Z},\ \mathrm{a-z},\ 0-9\}$ $\Sigma_I = \{\ (\ )\ \langle\ \rangle\ :\ ;\ ,\ \preceq\ \}$ $\Sigma_{Att} = \{a-z\}$ $\Sigma_{Par} = \{A-Z\}$ $\Sigma_W = \Sigma^*_{Str} \cup \mathbb{Q}$ $\Sigma = \Sigma_{NTS} \cup \Sigma_{TB} \cup \Sigma_I \cup \Sigma_W \cup \Sigma_{Att} \cup \Sigma_{Par}$
		$U \subset \Sigma^+_{NTS} \backslash \Sigma_{Par}$	$U = U_A \cup U_B$ Stammsymbole U_A (gegeben) $U_B \subset \{w \in {\Sigma_{TB}}^* \mid \lvert w\rvert \geq 2\} \backslash U_A$
$N \subset \Sigma^+_{NTS}$	$N \subset \Sigma^+_{NTS} \backslash \Sigma_{Att}$	$N = N_{TP} = \mathbf{subst}(\widetilde{N}_{TP})$ $\widetilde{N}_{TP}$ (vgl. $(W_6{''})$)	$N = N_{TP}$ $N_{TP} = \mathbf{subst}(\widetilde{N}_{TPA} \cup \widetilde{N}_{TPB})$ $\widetilde{N}_{TPA}$ (gegeben) $\widetilde{N}_{TPB}$ (vgl. $(W_6{'''})$)
$T = T_{TB} \cup \Sigma_I \cup \mathbb{Q} \cup \Sigma_{Str}$ $T_{TB} = \alpha(N)$	$T = T_{TB} \cup \Sigma_I \cup \mathbb{Q} \cup \Sigma_{Str}$ $T_{TB} = \alpha(N)$	$T = T_{TP} \cup \Sigma_I \cup \Sigma_W \cup \Sigma_{Att} \cup \Sigma_{Par}$ $T_{TP} = \mathbf{subst}(\widetilde{T}_{TP})$ $\widetilde{T}_{TP} = \widetilde{\alpha}(\widetilde{N}_{TP})$	$T = T_{TP} \cup \Sigma_I \cup \Sigma_W \cup \Sigma_{Att} \cup \Sigma_{Par}$ $T_{TP} = \widetilde{\alpha}(N_{TP})$ $\widetilde{T}_{TPA} = \widetilde{\alpha}(\widetilde{N}_{TPA})$ $\widetilde{T}_{TPB} = \widetilde{\alpha}(\widetilde{N}_{TPB})$
P (vgl. (W_1))	P (vgl. (W_1))	$P = P_{TP} = \mathbf{subst}(\widetilde{P}_{TP})$ $\widetilde{P}_{TP}$ (vgl. $(W_1{''})$)	$P = P_{TP} = \mathbf{subst}(\widetilde{P}_{TPA} \cup \widetilde{P}_{TPB})$ $\widetilde{P}_{TP} = \widetilde{P}_{TPA} \cup \widetilde{P}_{TPB}$ $\widetilde{P}_{TPA}$ (gegeben) $\widetilde{P}_{TPB}$ (vgl. $(W_1{'''})$)
$S \in N$	$S \in N$	$S \in N$	$S = \mathit{Objekt}$
$\alpha : \Sigma^*_{NTS} \longrightarrow \Sigma^*_{TB}$	$\alpha : \Sigma^*_{NTS} \longrightarrow \Sigma^*_{TB}$	$\widetilde{\alpha} : U\Sigma^* \longrightarrow \alpha(U)\Sigma^*$ (vgl. $(W_7{''})$)	$\widetilde{\alpha} : U\Sigma^* \longrightarrow \alpha(U)\Sigma^*$ (vgl. $(W_7{''})$)
$L(X),\ X \in N$ $\mathcal{L}(G)$ $\mathcal{D}_L$	$\mathrm{Typ}(X),\ X \in N$ $\mathcal{T}(G)$ $\mathcal{D}_T$	$L(X),\ X \in N_{TP}$ $\mathcal{L}(G)$ $\mathcal{D}_L$	$\mathrm{Typ}(X),\ X \in N_{TP}$ $\mathcal{T}(G)$ $\mathcal{D}_T$

Tabelle 4.4: Vergleich der Wohlgeformtheitsregeln und der semantischen Filterregeln

Objektgrammatik	**Typgrammatik**	**Template-Grammatik**	**Erweiterbare Zeitreihengrammatik**
(W_1) Jede *Produktion* in P ist eine *Alternativenregel* oder eine *Definitionsregel*.	(W_1') wie (W_1)	(W_1'') Jedes *Produktionstemplate* in $\widetilde{P}_{TP}$ ist ein *Alternativenregeltemplate* oder ein *Definitionsregeltemplate*.	(W_1''') Jedes *Produktionstemplate* in $\widetilde{P}_{TPB}$ ist ein *Alternativenregeltemplate* oder ein *Definitionsregeltemplate*.
(W_2) Eine *Alternativenregel* hat die Form $NTS \longrightarrow NTS_1 \mid ... \mid NTS_k$ für paarweise verschiedene Nichtterminalsymbole NTS, NTS_1, ..., $NTS_k \in N$ und $k \geq 2$.	(W_2') wie (W_2)	(W_2'') Ein *Alternativenregeltemplate* hat die Form $NTS \longrightarrow NTST_1 \mid ... \mid NTST_k$ für ein Nichtterminalsymbol $NTS \in U \cap \widetilde{N}_{TP}$ und Nichtterminalsymboltemplates $NTST_1, ..., NTST_k \in \widetilde{N}_{TP}$, $k \geq 2$, wobei alle Symbole paarweise verschieden sind.	(W_2''') Ein *Alternativenregeltemplate* in $\widetilde{P}_{TPB}$ hat die Form $NTS \longrightarrow NTST_1 \mid ... \mid NTST_k$ für ein Nichtterminalsymbol $NTS \in U \cap \widetilde{N}_{TPB}$ und Nichtterminalsymboltemplates $NTST_1, ..., NTST_k \in \widetilde{N}_{TP}$, $k \geq 2$, wobei alle Symbole paarweise verschieden sind.
(W_3) Eine *Definitionsregel* hat die Form $NTS \longrightarrow (\alpha(NTS)\ [; AttListe])$ oder $NTS \longrightarrow (\alpha(NTS)\ ; WMenge)$ wobei $NTS \in N$ ein Nichtterminalsymbol, $\alpha(NTS) \in T_{TB}$ der NTS zugeordnete Typbezeichner, *AttListe* eine Attributliste (W_4) und $WMenge \subset \Sigma_W$ eine Wertemenge ist.	(W_3') Eine *Definitionsregel* hat die Form $NTS \longrightarrow (\alpha(NTS)\ [; AttListe\ [; Bed\ ]])$ oder $NTS \longrightarrow (\alpha(NTS)\ ; WMenge)$ wobei $NTS \in N$ ein Nichtterminalsymbol, $\alpha(NTS) \in T_{TB}$ der NTS zugeordnete Typbezeichner, *AttListe* eine Attributliste (W_4'), *Bed* eine Bedingung (W_{10}') und $WMenge \subset \Sigma_W$ eine Wertemenge ist.	(W_3'') Ein *Definitionsregeltemplate* hat die Form $NTST \longrightarrow (\widetilde{\alpha}(NTST)\ [; AttListe])$ für ein Nichtterminalsymboltemplate $NTST \in \widetilde{N}_{TP}$ (W_6''), ein Typbezeichnertemplate $\widetilde{\alpha}(NTST)$ (W_7'') und eine Attributliste *AttListe* (W_4'') oder $NTS \longrightarrow (\alpha(NTS)\ ; WMenge)$ für ein Nichtterminalsymbol $NTS \in U \cap \widetilde{N}_{TP}$, einen Typbezeichner $\alpha(NTS)$ und eine Wertemenge $WMenge \subset \Sigma_W$. Nur ein Templateparameter, der im Argument von $NTST$ deklariert wurde, kann in der Attributliste *AttListe* verwendet werden.	(W_3''') Ein *Definitionsregeltemplate* in $\widetilde{P}_{TPB}$ hat die Form $NTST \longrightarrow (\widetilde{\alpha}(NTST)\ [; AttListe\ [; Bed\ ]])$ für ein Nichtterminalsymboltemplate $NTST \in \widetilde{N}_{TPB}$ (W_6'''), ein Typbezeichnertemplate $TBT = \widetilde{\alpha}(NTST)$ (W_7'''), eine Attributliste *AttListe* (W_4''') und eine Bedingung *Bed* (W_{10}''') oder $NTS \longrightarrow (\alpha(NTS)\ ; WMenge)$ für ein Nichtterminalsymbol $NTS \in U \cap \widetilde{N}_{TPB}$, einen Typbezeichner $\alpha(NTS)$ und eine Wertemenge $WMenge \subset \Sigma_W$. Nur ein Templateparameter, der im Argument von $NTST$ deklariert wurde, kann in der Attributliste *AttListe* verwendet werden.

Objektgrammatik	**Typgrammatik**	**Template-Grammatik**	**Erweiterbare Zeitreihengrammatik**
(W_4) Eine Attributliste *AttListe* besteht aus keinem, einem oder mehreren durch Kommas getrennten Nichtterminalsymbolen $NTS \in N$: $[NTS\,[,\ NTS\,]^*]$	(W_4') Eine Attributliste *AttListe* besteht aus keiner, einer oder mehreren durch Kommas getrennten Attributdeklarationen *AttDekl* (W_5'): $[AttDekl\,[,\ AttDekl\,]^*]$	(W_4'') Eine Attributliste *AttListe* besteht aus keinem, einem oder mehreren durch Kommas getrennten Nichtterminalsymboltemplates $NTST \in \widetilde{N}_{TP}$: $[NTST\,[,\ NTST\,]^*]$	(W_4''') wie (W_4')
—	(W_5') Eine Attributdeklaration *AttDekl* hat die Form $[AttName:]\ NTS$ wobei $AttName \in \Sigma_{Att}$ ein Attributname und $NTS \in N$ ein Nichtterminalsymbol ist.	—	(W_5''') Eine Attributdeklaration *AttDekl* hat die Form $[AttName:]\ NTST$ wobei $AttName \in \Sigma_{Att}$ ein Attributname und $NTST \in \widetilde{N}_{TPA} \cup \widetilde{N}_{TPB}$ ein Nichtterminalsymboltemplate ist.
—	—	(W_6'') Ein *Nichtterminalsymboltemplate* $NTST \in \widetilde{N}_{TP}$ hat eine der Formen NTS $NTS\langle\, TBT \,\rangle$ $NTS\langle\, TPar[\preceq\ TB] \,\rangle$ für $NTS \in U$, $TBT \in \widetilde{T}_{TP}$ und $TPar \in \Sigma_{Par}$.	(W_6''') wie (W_6'') mit $\widetilde{N}_{TP} = \widetilde{N}_{TPA} \cup \widetilde{N}_{TPB}$.

Objektgrammatik	**Typgrammatik**	**Template-Grammatik**	**Erweiterbare Zeitreihengrammatik**
—	—	(W_7'') Die *Typbezeichnertemplates* $\widetilde{T}_{TP}$ werden aus den Nichtterminalsymboltemplates $\widetilde{N}_{TP}$ berechnet: $$\widetilde{T}_{TP} = \widetilde{\alpha}(\widetilde{N}_{TP})$$ Dabei ist $\widetilde{\alpha} : U\ \Sigma^* \longrightarrow \alpha(U)\ \Sigma^*$ eine *erweiterte Alphabetwechsel-Abbildung* mit $$\begin{aligned} \widetilde{\alpha}(NTS) &= \alpha(NTS) \\ \widetilde{\alpha}(NTS\langle TBT\rangle) &= \alpha(NTS)\langle TBT\rangle \\ \widetilde{\alpha}(NTS\langle TPar\rangle) &= \alpha(NTS)\langle TPar\rangle \\ \widetilde{\alpha}(NTS\langle TPar \preceq TB\rangle) &= \alpha(NTS)\langle TPar\rangle \\ \widetilde{\alpha}(M) &= \{\widetilde{\alpha}(x) \mid x \in M\} \end{aligned}$$ $NTS \in \widetilde{N}_{TP} \cap U$, $NTST \in \widetilde{N}_{TP}$, $TB \in \widetilde{T}_{TP} \cap \alpha(U)$, $TBT \in \widetilde{T}_{TP}$, $TPar \in \Sigma_{Par}$ und $M \subset \widetilde{N}_{TP}$. α ist die Alphabetwechsel-Abbildung aus Definition 4.4.	(W_7''') wie (W_7'')
—	—	(W_8'') Zu jedem Stammsymbol in $X \in U$ gibt es genau ein Nichtterminalsymboltemplate in $\widetilde{N}_{TP}$, das mit X beginnt.	(W_8''') Zu jedem Stammsymbol in $X \in U_B$ gibt es genau ein Nichtterminalsymboltemplate in $\widetilde{N}_{TPB}$, das mit X beginnt.
—	(W_9') Jedes Paar zusammengehöriger runder Klammern () umschließt einen *Gültigkeitsbereich*. Typbezeichner, Attribute und Bedingungen gehören zum Gültigkeitsbereich des sie direkt umschließenden Klammerpaars. In einem Gültigkeitsbereich darf ein Attributname nur einmal deklariert werden.	—	(W_9''') Jedes Paar zusammengehöriger runder Klammern () umschließt einen *Gültigkeitsbereich*. Typbezeichner(templates), Attribute und Bedingungen gehören zum Gültigkeitsbereich des sie direkt umschließenden Klammerpaars. In einem Gültigkeitsbereich darf ein Attributname nur einmal deklariert werden.
—	(W_{10}') Eine *Bedingung Bed* ist ein boolescher Ausdruck in den Attributnamen, die im Gültigkeitsbereich (W_9') der Bedingung deklariert sind (W_5'). Zusätzlich zu den deklarierten Attributnamen dürfen in Bedingungen Attribute auftreten, die durch Existenz- oder Allquantoren gebunden sind.	—	(W_{10}''') wie (W_{10}')

Objektgrammatik	**Typgrammatik**	**Template-Grammatik**	**Erweiterbare Zeitreihengrammatik**
—	—	(W_{11}'') Die *Expansionsfunktion* **subst** ersetzt Templates folgendermaßen durch eine Menge von Instanziierungen: $NTS \longrightarrow \{NTS\}$ $NTS\langle TBT\rangle \longrightarrow \{NTS\langle X\rangle \mid X \in \mathbf{subst}(TBT)\}$ $NTS\langle TPar \preceq TB\rangle \longrightarrow \{NTS\langle X\rangle \mid X \preceq TB \text{ in } \mathcal{D}_L\}$ $TB \longrightarrow \{TB\}$ $TBT \longrightarrow \widetilde{\alpha}(\mathbf{subst}(\widetilde{\alpha}^{-1}(TBT)))$ $M \longrightarrow \bigcup_{X \in M} \mathbf{subst}(X)$ Dabei ist $NTS \in U \cap \widetilde{N}_{TP}$, $NTST \in \widetilde{N}_{TP}$, $TB \in \alpha(U) \cap \widetilde{T}_{TP}$, $TBT \in \widetilde{T}_{TP}$ und $M \subset U \cup \alpha(U) \cup \widetilde{N}_{TP} \cup \widetilde{T}_{TP}$ eine Menge von Symbolen. $\widetilde{\alpha}$ ist die Alphabetwechsel-Abbildung aus (W_7'')	(W_{11}''') wie (W_{11}'')
—	—	(W_{12}'') Ein Alternativenregeltemplate (W_2'') wird expandiert zu $NTS \longrightarrow NTS_1 \mid NTS_2 \mid \ldots$ $NTS_1, NTS_2, \ldots \in \mathbf{subst}(\{NTST_1, \ldots, NTST_k\})$	(W_{12}''') wie (W_{12}'')
—	—	(W_{13}'') Das Definitionsregeltemplate $NTST \longrightarrow (\widetilde{\alpha}(NTST)\ [; AttListe])$ aus (W_3'') wird expandiert zu $NTS_1 \longrightarrow (\widetilde{\alpha}(NTS_1)\ [; AttListe(NTS_1)])$ $NTS_2 \longrightarrow (\widetilde{\alpha}(NTS_2)\ [; AttListe(NTS_2)])$ $\vdots$ mit $NTS_1, NTS_2, \ldots \in \mathbf{subst}(NTST)$. Bei der zweiten Form liegt kein Template vor, so dass das Definitionsregeltemplate direkt als Definitionsregel übernommen werden kann.	(W_{13}''') wie (W_{13}'')

Objektgrammatik	**Typgrammatik**	**Template-Grammatik**	**Erweiterbare Zeitreihengrammatik**
(W_{14}) Jedes Nichtterminalsymbol aus N tritt als linker Teil genau einer Produktion in P auf. Insbesondere gibt es eine Produktion, die das Startsymbol S ableitet.	(W_{14}') wie (W_{14})	(W_{14}'') Jedes Nichtterminalsymboltemplate aus $\widetilde{N}_{TP}$ tritt als linker Teil genau eines Produktionstemplates in $\widetilde{P}_{TP}$ auf. Insbesondere gibt es ein Produktionstemplate, das das Startsymbol S ableitet.	(W_{14}''') Jedes Nichtterminalsymboltemplate aus $\widetilde{N}_{TPB}$, insbesondere das Nichtterminalsymbol *BenutzerdefiniertesObjekt*, tritt als linker Teil genau eines Produktionstemplates in $\widetilde{P}_{TPB}$ auf.
(W_{15}) Jedes Nichtterminalsymbol aus N ist vom Startsymbol S *erreichbar* (d.h. $\forall X \in N$ existiert $S \stackrel{*}{\Rightarrow} X$), indem bei der Ableitung nur Alternativenregeln (W_2) angewendet werden.	(W_{15}') wie (W_{15})	(W_{15}'') Jedes Nichtterminalsymbol aus N_{TP} ist vom Startsymbol S aus erreichbar, indem bei der Ableitung nur (evtl. durch Expansion mit (W_{12}'') gewonnene) Alternativenregeln (W_2'') angewendet werden.	(W_{15}''') Jedes Nichtterminalsymbol, das aus einem Nichtterminalsymboltemplate in $\widetilde{N}_{TPB}$ gewonnen wurde, ist vom Nichtterminalsymbol *BenutzerdefiniertesObjekt* aus erreichbar, indem bei der Ableitung nur (evtl. durch Expansion mit (W_{12}''') gewonnene) Alternativenregeln (W_2''') angewendet werden.
—	(S_1') Alle booleschen Bedingungen in einem Wort werden durch die Attributwerte des zugehörigen Gültigkeitsbereichs erfüllt.	—	(S_1''') wie (S_1')
—	(S_2') Jede nicht-leere Teilsprache enthält mindestens ein *semantisch gültiges* Wort, d.h. $\forall\, NTS \in N: \quad \mathrm{Typ}(NTS) \neq \emptyset$	—	(S_2''') wie (S_2')

4.2.2 Objektgrammatik

Die Definition einer Objektgrammatik bildet die Grundlage für die Typgrammatik, die Templategrammatik und die Erweiterbare Zeitreihengrammatik (Abbildung 4.1).

Eine Objektgrammatik ist eine kontextfreie Grammatik, die gegenüber einer allgemeinen kontextfreien Grammatik zwei Besonderheiten aufweist: Zum einen erzeugt sie Wörter, die als hierarchisch aufgebaute *Objekte* interpretiert werden können. Zum anderen wird die Sprache, die von einer Objektgrammatik erzeugt wird, in Teilsprachen untergliedert. Die erzeugten Objekte enthalten Information darüber, zu welcher Teilsprache sie gehören.

Dadurch werden zwei Probleme gelöst, die bei der Repräsentation von musikalischem Wissen auftreten. Zum einen ermöglicht der hierarchische Aufbau der Objekte, diese zu größeren Strukturen zusammenzusetzen. Ein musikalisches Motiv und ein Akkord können z.B. durch eine Menge von Tönen charakterisiert werden. Wenn dem Motiv und dem Akkord dieselbe Tondefinition zugrundeliegt, können sie in Bezug zueinander gesetzt werden, um neues musikalisches Wissen zu gewinnen.

Zum anderen werden mehrdeutige, kontextabhängige Interpretationen des gleichen Symbols, wie sie in Abschnitt 4.2.1 diskutiert wurden, in einer Objektgrammatik aufgelöst. Wenn z.B. mit dem Symbol "D" sowohl die Tonhöhe D, die Tonart D-Dur, eine Dominante und eine Motivklasse gemeint sein können, werden in einer Objektgrammatik vier voneinander unterscheidbare Objekte erzeugt, die verschiedenen Teilsprachen angehören. Eine *Teilsprache* enthält alle Wörter, die von einem bestimmten Nichtterminalsymbol abgeleitet werden können (vgl. Definition 2.8). Wir werden sehen, dass die Teilsprachen einer Objektgrammatik eine Hierarchie bilden, die es ermöglicht, Algorithmen auf unterschiedlichen Abstraktionsebenen zu formulieren.

Betrachten wir zunächst eine Objektgrammatik, die Takte und ganze Zahlen erzeugt:

Beispiel 4.3 (Objektgrammatik)
Die Grammatik $G = (N, T, P, S)$ mit

$$
\begin{aligned}
N &= \{\mathit{Objekt},\ \mathit{Takt},\ \mathit{Int}\} \\
T &= \{\mathbf{Objekt},\ \mathbf{Takt},\ \mathbf{Int}\} \cup \{(\ )\ ;\ ,\} \cup \{\mathrm{A}-\mathrm{Z},\ \mathrm{a}-\mathrm{z},\ 0-9\}^* \cup \mathbb{Q} \\
P &= \left\{ \begin{array}{llll} (P_1) & \mathit{Objekt} & \longrightarrow & \mathit{Int} \mid \mathit{Takt} \\ (P_2) & \mathit{Takt} & \longrightarrow & (\mathbf{Takt};\ \mathit{Int}, \mathit{Int}) \\ (P_3) & \mathit{Int} & \longrightarrow & (\mathbf{Int};\ \mathbb{Z}) \end{array} \right\} \\
S &= \mathit{Objekt}
\end{aligned}
$$

ist eine Objektgrammatik.

* * *

Da auf der linken Seite der Produktionen ein einziges Nichtterminalsymbol steht, handelt es sich bei diesem Beispiel um eine kontextfreie Grammatik. Die Grammatik besteht aus Nichtterminalsymbolen N, Terminalsymbolen T, Produktionen P und einem Startsymbol

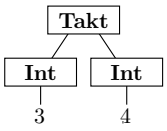


Abbildung 4.6: Graphische Darstellung des Objekts (**Takt**; (**Int**; 3), (**Int**; 4)). Die inneren Knoten werden durch die Typbezeichner, die Blattknoten durch Elemente von $\mathbb{Z}$ beschriftet.

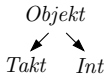


Abbildung 4.7: Teilsprachendiagramm für Beispiel 4.3

S. Neben Interpunktionszeichen, Zeichenketten und Zahlen enthält die Menge der Terminalsymbole auch fettgedruckte *Typbezeichner* **Objekt**, **Takt** und **Int**, die dazu dienen, die speziellste Teilsprache zu kennzeichnen, der ein Objekt angehört. Ein Typbezeichner ist als unteilbares Terminalsymbol aufzufassen, das nur wegen der besseren Lesbarkeit nicht als einzelner Buchstabe definiert wurde. Im Gegensatz zu den fettgedruckten Terminalsymbolen sind die Nichtterminalsymbole kursiv gesetzt. Auch Nichtterminalsymbole werden mit Hilfe unteilbarer Zeichenketten dargestellt, um ihre Bedeutung zu verdeutlichen.

Die Ableitung von Wörtern in einer Objektgrammatik unterscheidet sich nicht von der Ableitung in einer herkömmlichen Grammatik. Vom Startsymbol *Objekt* ausgehend wendet man die Produktionen der Grammatik an und erhält so eine geklammerte Zeichenkette, die als Baum interpretiert wird. Das Objekt (**Takt**; (**Int**; 3), (**Int**; 4)) stellt einen 3/4-Takt dar und entsteht durch die folgende Ableitung:

$$
\begin{array}{lll}
\mathit{Objekt} & \vdash \quad \mathit{Takt} & (P_1) \\
& \vdash \quad (\textbf{Takt};\ \mathit{Int}, \mathit{Int}) & (P_2) \\
& \vdash \quad (\textbf{Takt};\ (\textbf{Int}; 3), \mathit{Int}) & (P_3) \\
& \vdash \quad (\textbf{Takt};\ (\textbf{Int}; 3), (\textbf{Int}; 4)) & (P_3)
\end{array}
$$

In jeder Zeile ist rechts die angewendete Produktion angegeben. Die Ableitung beginnt mit dem Startsymbol $S = \mathit{Objekt}$, erzeugt dann mit der Produktion (P_2) das Gerüst für ein Taktobjekt und füllt dieses durch zweimaliges Anwenden von (P_3) mit den ganzen Zahlen 3 und 4. Abbildung 4.6 zeigt die graphische Darstellung des Takt-Objekts.

Jedes Nichtterminalsymbol des Beispiels induziert eine *Teilsprache*, die aus den Wörtern der Gesamtsprache besteht, die von diesem Nichtterminalsymbol abgeleitet werden können. Beispielsweise ist $L(\mathit{Int}) = \{\ldots(\textbf{Int}; -1),\ (\textbf{Int}; 0),\ (\textbf{Int}; 1)\ldots\}$ die Teilsprache, die vom Nichtterminalsymbol *Int* in G erzeugt wird. Die Menge aller Teilsprachen für die Nichtterminalsymbole der Beispielgrammatik ist $\mathcal{L}(G) = \{L(\mathit{Objekt}), L(\mathit{Takt}), L(\mathit{Int})\}$. Da man jedes Objekt, das von *Takt* abgeleitet werden kann, auch von *Objekt* ableiten kann, sind die zugehörigen Teilsprachen ineinander enthalten: $L(\mathit{Takt}) \subset L(\mathit{Objekt})$. Mit dem gleichen Argument gilt $L(\mathit{Int}) \subset L(\mathit{Objekt})$. Außerdem sind $L(\mathit{Takt})$ und $L(\mathit{Int})$ offensichtlich disjunkt, so dass sich die Inklusionen der Teilsprachen insgesamt mit dem *Teilsprachendiagramm* in

Abbildung 4.7 darstellen lassen. Der Typbezeichner **Takt** im obigen Takt-Objekt bezeichnet die kleinste Teilsprache $L(\mathit{Takt}) \in \mathcal{L}(G)$, zu der das Beispielobjekt gehört.

Es ist nur möglich, solche Aussagen über das Teilsprachendiagramm zu machen, weil die Produktionen einer Objektgrammatik eine stark eingeschränkte Gestalt haben. Jedes Nichtterminalsymbol wird durch genau eine Produktion in P abgeleitet. Bei den Produktionen handelt es sich entweder um *Alternativenregeln* wie (P_1), die ein Nichtterminalsymbol durch eine von mehreren Alternativen ersetzen[2], oder um *Definitionsregeln*, die ein Nichtterminalsymbol durch einen geklammerten Ausdruck ersetzen wie (P_2) und (P_3). Die erzeugte Klammerung ist immer balanciert, d.h. zu jeder öffnenden gibt es eine schließende Klammer, und jedes Präfix eines Worts hat mindestens so viele öffnende wie schließende Klammern. Dies garantiert, dass man das erzeugte Wort als Baum interpretieren kann. Die rechte Seite einer Definitionsregel enthält einen Typbezeichner und eine Liste von *Attributen*. In (P_2) besteht die Attributliste aus den Nichtterminalsymbolen: *Int*, *Int*. Ein Attribut kann aber auch in Form einer Menge angegeben werden, aus der bei der Ableitung ein Element auszuwählen ist wie in (P_3).

Wenn man in einer Kette von Ableitungen zum ersten Mal eine Definitionsregel anwendet, werden ein Klammerpaar und ein Typbezeichner erzeugt (vgl. zweite Zeile in der Ableitung des Taktobjekts). Da der Ausdruck nun Terminalsymbole enthält, kann er nicht mehr zu einem einzelnen Nichtterminalsymbol abgeleitet werden. Alle von diesem Ausdruck abgeleiteten Objekte gehören daher zu derselben Teilsprache der Grammatik. Teilsprachen, die von Nichtterminalsymbolen auf der linken Seite von Definitionsregeln erzeugt werden, sind also unteilbar und enthalten nur die leere Sprache. Die erzeugenden Nichtterminalsymbole werden als *atomar* bezeichnet.

Die Typbezeichner, die Nichtterminalsymbolen auf der linken Seite von Alternativenregeln entsprechen, treten hingegen nie in fertig abgeleiteten Objekten auf. Teilsprachen, die von diesen Nichtterminalsymbolen erzeugt werden, werden nicht durch einen einheitlichen Typbezeichner charakterisiert, sondern setzen sich aus mehreren atomaren Teilsprachen zusammen. Solche Teilsprachen und ihre erzeugenden Nichtterminalsymbole heißen *abstrakt*. Da jedes Nichtterminalsymbol auf der linken Seite genau einer Produktion vorkommt, zerfällt die Menge der Nichtterminalsymbole im Beispiel disjunkt in die abstrakten und die atomaren Nichtterminalsymbole. Im Beispiel ist *Objekt* abstrakt; *Takt* und *Int* sind hingegen atomare Nichtterminalsymbole.

Insgesamt sind in einer Objektgrammatik zwei Arten hierarchischer Strukturen zu unterscheiden: Zum einen lassen sich die erzeugten Wörter als Bäume interpretieren; zum anderen bilden die Teilsprachen ein durch die Teilmengenrelation induziertes hierarchisches Teildiagramm.

Diese Ideen werden nun durch die Definition von Objektgrammatiken präzisiert. Damit Objektgrammatiken nur baumförmige Objekte erzeugen wird ihre Gestalt durch Wohlgeformtheitsregeln eingeschränkt. Um den Zusammenhang zwischen Objektgrammatik und den später eingeführten Grammatiken (Typgrammatik, Templategrammatik und Erweiterbare Zeitreihengrammatik) zu verdeutlichen, wurden die Wohlgeformtheitsregeln nicht in die Definitionen der Grammatiken integriert, sondern in Tabelle 4.4 nebeneinandergestellt.

[2] Eine Alternativenregel wird hier als *eine* Produktion behandelt, auch wenn sie genau genommen eine abkürzende Schreibweise für eine Menge von Produktionen ist.

Definition 4.4 (Objektgrammatik)
Gegeben seien ein Alphabet von kursiven Zeichen $\Sigma_{NTS} = \{A-Z,\ a-z,\ 0-9\}$ zur Bildung von Nichtterminalsymbolen, ein Alphabet von fettgedruckten Zeichen $\Sigma_{TB} = \{\mathbf{A}-\mathbf{Z},\ \mathbf{a}-\mathbf{z},\ \mathbf{0}-\mathbf{9}\}$ zur Bildung von Typbezeichnern, ein Alphabet $\Sigma_{Str} = \{\mathrm{A}-\mathrm{Z},\ \mathrm{a}-\mathrm{z},\ 0-9\}$ zur Bildung Zeichenketten, eine Menge von Interpunktionszeichen $\Sigma_I = \{\ (\ )\ ;\ ,\ \}$ zur Markierung von Hierarchieebenen und Aufzählungen in Objekten, die *Wertemenge* $\Sigma_W = \Sigma^*_{Str} \cup \mathbb{Q}$ zur Bildung von Werten an den Blattknoten von Objekten und eine *Alphabetwechsel-Abbildung* $\alpha : \Sigma^*_{NTS} \longrightarrow \Sigma^*_{TB}$, die einer Zeichenkette aus Σ^*_{NTS} die entsprechende Zeichenkette in Fettschrift aus Σ^*_{TB} zuordnet.

Eine Grammatik $G = (N, T, P, S)$ mit Nichtterminalsymbolen N, Terminalsymbolen T, Produktionen P und einem Startsymbol $S \in N$ heißt *Objektgrammatik*, wenn T und N endliche Mengen mit $T_{TB} \subseteq T \subset T_{TB} \cup \Sigma_I \cup \Sigma_W$ und $N \subset \Sigma^+_{NTS}$ sind und die Grammatik die Wohlgeformtheitsregeln (W_1), (W_2), (W_3), (W_4) (W_{14}) und (W_{15}) in Tabelle 4.4 erfüllt.

Die Menge der *Typbezeichner* $T_{TB} = \alpha(N)$ ergibt sich dabei durch Anwendung der Alphabetwechsel-Abbildung direkt aus den Nichtterminalsymbolen N.

Ein Nichtterminalsymbol in einer Objektgrammatik heißt *abstrakt*, wenn es als linke Seite einer Alternativenregel (W_2) auftritt; es heißt *atomar*, wenn es als linke Seite einer Definitionsregel (W_3) verwendet wird. Eine Teilsprache einer Objektgrammatik heißt *abstrakt* bzw. *atomar*, wenn sie von einem abstrakten bzw. atomaren Nichtterminalsymbol erzeugt wird.

* * *

Grundmengen. Die Alphabete $\Sigma_{NTS} = \{A-Z,\ a-z,\ 0-9\}$ und $\Sigma_{TB} = \{\mathbf{A}-\mathbf{Z},\ \mathbf{a}-\mathbf{z},\ \mathbf{0}-\mathbf{9}\}$ geben an, aus welchen Zeichen sich Nichtterminalsymbole und Typbezeichner zusammensetzen dürfen. An den Blattknoten eines Objekts kommen die Zeichen aus der Wertemenge $\Sigma_W = \Sigma^*_{Str} \cup \mathbb{Q}$ vor, die sich aus Zeichenketten Σ^*_{Str} und rationalen Zahlen zusammensetzt. Nicht in der Wertemenge enthalten sind die Interpunktionszeichen $\Sigma_I = \{\ (\ )\ ;\ ,\ \}$, diese legen die Hierarchieebenen eines Objekts mit runden Klammern fest und trennen die Bestandteile einer Hierarchieebene durch Kommas. So befinden sich im Objekt $(\mathbf{Takt};\ (\mathbf{Int}; 3), (\mathbf{Int}; 4))$ die Typbezeichner **Takt** und **Int** direkt hinter der öffnenden Klammer der zugehörigen Hierarchieebene. Der Typbezeichner **Takt** ist ein unteilbares Terminalsymbol, auf das nach dem Semikolon eine Liste durch Kommas getrennter Attribute folgt.

Nichtterminalsymbole und Typbezeichner. In Beispiel 4.3 sind die Nichtterminalsymbole $N = \{\mathit{Objekt},\ \mathit{Takt},\ \mathit{Int}\}$ und die Typbezeichner $\{\mathbf{Objekt},\ \mathbf{Takt},\ \mathbf{Int}\}$ bis auf das zugrundeliegende Alphabet gleich. In der Definition werden die Typbezeichner $T_{TB} = \alpha(N)$ mit Hilfe der Alphabetwechsel-Abbildung α aus den Nichtterminalsymbolen N gewonnen. Auf diese Weise gibt es in jeder Objektgrammatik eine Entsprechung zwischen Nichtterminalsymbolen und Typbezeichnern. In den Wörtern, die von einer Objektgrammatik erzeugt werden, treten aber nur die Typbezeichner auf, die atomaren Nichtterminalsymbolen entsprechen. Die Alphabetwechsel-Abbildung generiert die Typbezeichner aus den Nichtterminalsymbolen (und nicht umgekehrt), weil die Teilsprachen für allgemeine Grammatiken definiert wurden und dort von den Nichtterminalsymbolen erzeugt werden. Typbezeichner sind lediglich ein Hilfsmittel, um in einer Objektgrammatik die Teilsprachen in den erzeugten Objekten kenntlich zu machen.

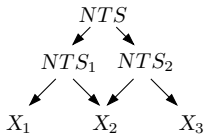


Abbildung 4.8: Mehrfachvererbung

Alternativenregeln. Die Wohlgeformtheitsregeln (W_1), (W_2), (W_3) und (W_4) legen die Gestalt der Produktionen in einer Objektgrammatik fest. Damit beeinflussen sie direkt die Teilsprachen, die von einer Objektgrammatik erzeugt werden. Eine *Alternativenregel* hat die Form

$$NTS \longrightarrow NTS_1 \mid ... \mid NTS_k$$

für paarweise verschiedene Nichtterminalsymbole $NTS, NTS_1, ..., NTS_k \in N$ und $k \geq 2$ (W_2). Sie ersetzt ein Nichtterminalsymbol durch ein anderes Nichtterminalsymbol. Für den Umfang der von einer Objektgrammatik erzeugten Sprache ist sie überflüssig, da man die rechten Seiten der Produktionen für $NTS_1, ..., NTS_k$ direkt einsetzen könnte. Da aber jedes Nichtterminalsymbol eine Teilsprache erzeugt, legen die Alternativenregeln die Teilsprachenhierarchie fest: Für $1 \leq i \leq k$ gilt $L(NTS_i) \subseteq L(NTS)$, also $(NTS_i, NTS) \in \mathcal{D}_L$. Die von einem abstrakten Nichtterminalsymbol erzeugte Teilsprache umfasst alle Teilsprachen, die von atomaren Nichtterminalsymbolen erzeugt werden:

$$L(NTS) = \bigcup_{i=1}^{k} L(NTS_i)$$

Auch wenn Nichtterminalsymbole auf der rechten Seite einer Alternativenregel paarweise verschieden sind, folgt daraus nicht, dass die von ihnen erzeugten Teilsprachen disjunkt sind (Abbildung 4.8). Der Grund liegt darin, dass abstrakte Nichtterminalsymbole NTS_1 und NTS_2 auf der rechten Regelseite stehen können, die später durch das gleiche atomare Nichtterminalsymbol X_2 ersetzt werden. Das Teilsprachendiagramm ist daher im allgemeinen kein Wurzelbaum, sondern ein azyklischer, gerichteter Graph. In der objektorientierten Programmierung würde man von Mehrfachvererbung sprechen.

Zerlegung in atomare Teilsprachen. Man kann jedoch jede abstrakte Teilsprache eindeutig in disjunkte atomare Teilsprachen zerlegen. Man erhält die Zerlegung, indem man in der Alternativenregel für das erzeugende abstrakte Nichtterminalsymbol solange Nichtterminalsymbole durch das Einsetzen rechter Regelseiten eliminiert, bis nur noch Alternativen von geklammerten Definitionen übrig sind. In Beispiel 4.3 zerfällt $L(\mathit{Objekt})$ in die Teilsprachen $L(\mathit{Takt})$ und $L(\mathit{Int})$, denn in der Alternativenregel (P_1) kann man *Int* und *Takt* eliminieren, indem man die rechten Seiten der Definitionsregeln (P_2) und (P_3) einsetzt. Die gewonnene Zerlegung ist (bis auf Wiederholungen) immer eindeutig, weil jedes Nichtterminalsymbol durch eine eindeutige Produktion abgeleitet wird (W_{14}).

Globale Eigenschaften einer Objektgrammatik. Während (W_1), (W_2), (W_3) und (W_4) die Struktur einer Objektgrammatik beschreiben, legen (W_{14}) und (W_{15}) globale Eigenschaften fest. Jedes Nichtterminalsymbol muss auf der linken Seite genau einer Produktion

auftreten (W_{14}), damit es eindeutig entweder als abstrakt oder atomar identifiziert werden kann. Außerdem ist so sichergestellt, dass jedes Nichtterminalsymbol weiter abgeleitet werden kann.

Die Tatsache, dass jedes Nichtterminalsymbol durch die Anwendung von Alternativenregeln erreichbar ist (W_{15}), stellt sicher, dass alle Objekte, die von einem Nichtterminalsymbol abgeleitet werden können, auch vom Startsymbol abgeleitet werden können und somit zu einer Teilsprache gehören. Jede Teilsprache einer Grammatik liegt definitionsgemäß in der Gesamtsprache, die von der Grammatik erzeugt wird, so dass das Teilsprachendiagramm einer Grammatik zusammenhängend ist.

Desweiteren gibt es in einem Teilsprachendiagramm keine nicht-trivialen Zyklen, an denen ungleiche Teilsprachen beteiligt sind, denn ein Zyklus $L(A) \subseteq L(B) \subseteq \ldots \subseteq L(C) \subseteq L(A)$ impliziert die Gleichheit aller beteiligten Teilsprachen. Daraus folgt, dass ein Teilsprachendiagramm ein azyklischer Digraph ist. Da alle Teilsprachen in der durch das Startsymbol erzeugten Sprache enthalten sind, hat das Teilsprachendiagramm eine eindeutige Quelle.

Chomsky-Typ einer Objektgrammatik. Eine Objektgrammatik ist per Definition kontextfrei, da auf der linken Seite jeder Produktion nur ein Nichtterminalsymbol steht (vgl. (W_2), (W_3)). Die von ihr erzeugte Sprache ist jedoch im allgemeinen nicht regulär, weil eine Objektgrammatik Produktionen der Form

$$A \longrightarrow (\mathbf{A}; A|a)$$

für ein Nichtterminalsymbol A und ein Terminalsymbol a erlaubt (W_3). Durch wiederholte Anwendung dieser Produktion können Wörter mit beliebig tief geschachtelten balancierten Klammerketten gebildet werden. Wie in Abschnitt 2.1.2 des Grundlagenkapitels gezeigt wurde, ist eine Sprache, die solche Klammerungen enthält, nicht regulär.

Die gerade diskutierten Eigenschaften einer Objektgrammatik werden in einem Lemma zusammengefasst.

Lemma 4.1 (Eigenschaften einer Objektgrammatik)

(a) Eine Objektgrammatik ist kontextfrei, aber im allgemeinen nicht einseitig linear.

(b) Das Teilsprachendiagramm einer Objektgrammatik ist ein zusammenhängender azyklischer Digraph mit einer Quelle.

(c) Wenn das Teilsprachendiagramm ein Baum ist, sind zwei Teilsprachen entweder ineinander enthalten oder disjunkt.

(d) Die Quelle des Teilsprachendiagramms hat nur auslaufende Kanten. Knoten ungleich der Quelle, die einer abstrakten Teilsprachen entsprechen, haben ein- und und auslaufende Kanten. Knoten, die atomare Teilsprachen darstellen, haben einlaufende Kanten. Ihre auslaufenden Kanten führen zur leeren Teilsprache, der Senke des Teilsprachendiagramms.

(e) Jede Teilsprache lässt sich eindeutig in eine disjunkte Vereinigung von atomaren Teilsprachen zerlegen.

4.2.3 Typgrammatik

Beim maschinellen Lernen hängt der Lernerfolg entscheidend davon ab, dass man die Lerndaten problemangepasst und kompakt codiert. Die Objektgrammatik ist ein erster Schritt, um die Wertebereiche unterschiedlicher Merkmale von Zeitreihen zu repräsentieren. Die Wertebereiche sind jedoch im allgemeinen zu groß, um eine erfolgversprechende Codierung von Lernmustern zu erlauben. Daher werden in einer *Typgrammatik* Bedingungen (*constraints*) eingeführt, mit denen Teilsprachen auf Werte eingeschränkt werden können, die für das jeweilige Lernproblem sinnvoll sind. Eine so gefilterte Teilsprache heißt *Typ*.

Die Filterbedingungen werden in die Definitionsregeln eingebaut, indem man einen booleschen Ausdruck angibt, der von den vorhandenen Attributen abhängt. In Beispiel 4.3 konnten auch Takte erzeugt werden, die üblicherweise in der Musik nicht vorkommen wie ein $-\frac{3}{4}$- oder ein $\frac{4}{23}$-Takt. Im folgenden Beispiel für eine Typgrammatik wird die Teilsprache $L(\mathit{Takt})$ auf Taktarten mit einem Zähler zwischen 1 und 7 und einer Zweierpotenz als Nenner eingeschränkt. Wenn ein musikalischer Corpus vorliegt, ist es sinnvoll, sogar nur die tatsächlich vorkommenden Taktarten zuzulassen.

Beispiel 4.5 (Typgrammatik)
Die Grammatik $G = (N, T, P, S)$ mit

$$
\begin{array}{lcl}
N & = & \{\mathit{Objekt},\ \mathit{Takt},\ \mathit{Int}\} \\
T & = & \{\mathbf{Objekt},\ \mathbf{Takt},\ \mathbf{Int}\} \\
P & = & \left\{ \begin{array}{llll}
(P_1) & \mathit{Objekt} & \longrightarrow & \mathit{Int} \mid \mathit{Takt} \\
(P_2) & \mathit{Takt} & \longrightarrow & (\mathbf{Takt};\ z : \mathit{Int},\ n : \mathit{Int};\ (1 \leq z \leq 7) \wedge (\exists k \in \mathbb{N} :\ n = 2^k)) \\
(P_3) & \mathit{Int} & \longrightarrow & (\mathbf{Int};\ \mathbb{Z})
\end{array} \right\} \\
S & = & \mathit{Objekt}
\end{array}
$$

ist eine Typgrammatik.

* * *

Im Vergleich zu einer Objektgrammatik ist hier neu, dass Definitionsregeln einen dritten, durch ein Semikolon abgetrennten Eintrag besitzen können, der eine boolesche Bedingung enthält. Die Bedingung hängt von Attributnamen ab, die ebenfalls hier eingeführt werden.

Ein Beispiel für ein von G erzeugtes Wort ist der $\frac{3}{4}$-Takt

$$(\mathbf{Takt};\ z : (\mathbf{Int}; 3),\ n : (\mathbf{Int}; 4);\ (1 \leq z \leq 7) \wedge (\exists k \in \mathbb{N} :\ n = 2^k)),$$

der in Abbildung 4.9 graphisch dargestellt wird. Im Unterschied zu Beispiel 4.3 sind die Attribute für Zähler und Nenner nun mit z und n benannt, um die die Taktarten durch eine boolsche Bedingung einschränken zu können. Die Bedingung ist in Form eines prädikatenlogischen Ausdrucks angegeben, der nur Variablen enthält, die in der zugehörigen Attributliste deklariert sind. Die Variablen in der Bedingung sind durch die Deklarationen gebunden.

Da Bedingungen kein Bestandteil herkömmlicher generativer Grammatiken sind, werfen sie die Frage auf, zu welchem Zeitpunkt sie ausgewertet werden. Um den Erzeugungsmechanismus einer Grammatik unverändert zu lassen, werden die Objekte einer Typgrammatik in

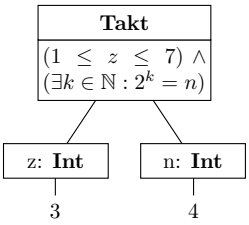


Abbildung 4.9: Graphische Darstellung des Objekts
(**Takt**; $z : (\mathbf{Int}; 3)$, $n : (\mathbf{Int}; 4)$; $(1 \leq z \leq 7) \wedge (\exists k \in \mathbb{N} : \ n = 2^k)$).
Bedingungen werden unter dem Typbezeichner des Gültigkeitsbereichs angegeben.

zwei Schritten erzeugt. Erst wird ein Wort durch die Anwendung von Produktionen generiert; Bedingungen werden dabei wie Terminalsymbole behandelt:

$$
\begin{aligned}
\mathit{Objekt} &\vdash \mathit{Takt} && (P_1)\\
&\vdash (\mathbf{Takt};\ z : \mathit{Int},\ n : \mathit{Int};\ (1 \leq z \leq 7) \wedge (\exists k \in \mathbb{N} : \ n = 2^k)) && (P_2)\\
&\vdash (\mathbf{Takt};\ z : (\mathbf{Int}; 3),\ n : \mathit{Int};\ (1 \leq z \leq 7) \wedge (\exists k \in \mathbb{N} : \ n = 2^k)) && (P_3)\\
&\vdash (\mathbf{Takt};\ z : (\mathbf{Int}; 3),\ n : (\mathbf{Int}; 4);\ (1 \leq z \leq 7) \wedge (\exists k \in \mathbb{N} : \ n = 2^k)) && (P_3)
\end{aligned}
$$

Falls das erzeugte Wort Bedingungen enthält, werden diese im zweiten Schritt als logische Ausdrücke interpretiert und ausgewertet. Sind alle Bedingungen erfüllt, so wird das Wort als *semantisch gültig* akzeptiert, andernfalls wird es verworfen. Im Beispiel handelt es sich um ein semantisch gültiges Objekt. Syntaktisch, aber nicht semantisch gültig ist das Objekt

$$(\mathbf{Takt};\ z : (\mathbf{Int}; -3), n : (\mathbf{Int}; 13);\ (1 \leq z \leq 7) \wedge (\exists k \in \mathbb{N} : \ n = 2^k))$$

Die Filterung der semantisch gültigen Objekte durch Bedingungen führt zu einer Verkleinerung der Teilsprachen auf *Typen*, für die wie für Teilsprachen Teilmengenrelationen gebildet werden. Im vorliegenden Beispiel ist $L(\mathit{Int}) = \mathrm{Typ}(\mathit{Int})$, weil die Definitionsregel zu *Int* keine Bedingung enthält, aber $\mathrm{Typ}(\mathit{Takt}) \subset L(\mathit{Takt})$, denn es gibt Takt-Objekte, die die boolesche Bedingung für den Takt nicht erfüllen. Insgesamt erhält man für Beispiel 4.5 die Teilmengenbeziehungen $L(\mathit{Int}) = \mathrm{Typ}(\mathit{Int}) \subset \mathrm{Typ}(\mathit{Objekt}) \subset L(\mathit{Objekt})$, $\mathrm{Typ}(\mathit{Takt}) \subset \mathrm{Typ}(\mathit{Objekt})$ und $L(\mathit{Takt}) \subset L(\mathit{Objekt})$. Typ- und Teilsprachendiagramm von Beispiel 4.5 stimmen mit dem Teilsprachendiagramm von Beispiel 4.3 überein (Abbildung 4.7).

Definition 4.6 (Typgrammatik)
Gegeben seien drei Alphabete $\Sigma_{NTS} = \{A - Z,\ a - z,\ 0 - 9\}$, $\Sigma_{TB} = \{\mathbf{A} - \mathbf{Z},\ \mathbf{a} - \mathbf{z},\ \mathbf{0} - \mathbf{9}\}$ und $\Sigma_{Str} = \{\mathrm{A} - \mathrm{Z},\ \mathrm{a} - \mathrm{z},\ 0 - 9\}$, eine Menge von *Interpunktionszeichen* $\Sigma_I = \{\ (\ ) : ;\ ,\ \}$, die *Wertemenge* $\Sigma_W = \Sigma_{Str}^* \cup \mathbb{Q}$, eine Menge von *Attributnamen* $\Sigma_{Att} = \{a - z\}$ zur Formulierung von Bedingungen, eine *Alphabetwechsel-Abbildung*

$\alpha : \Sigma^*_{NTS} \longrightarrow \Sigma^*_{TB}$, die einer Zeichenkette aus Σ^*_{NTS} die entsprechende Zeichenkette in Fettschrift aus Σ^*_{TB} zuordnet,

Eine Grammatik $G = (N, T, P, S)$ mit einer endlichen Menge von Nichtterminalsymbolen $N \subset \Sigma^+_{NTS} \backslash \Sigma_{Att}$, Terminalsymbolen $T = T_{TB} \cup \Sigma_I \cup \mathbb{Q} \cup \Sigma_{Str}$, Produktionen P und einem Startsymbol $S \in N$ heißt *Typgrammatik*, wobei die *Typbezeichner* als $T_{TB} = \alpha(N)$ definiert sind und die Grammatik die Wohlgeformtheitsregeln $(W_1{}')$, $(W_2{}')$, $(W_3{}')$, $(W_4{}')$, $(W_5{}')$, $(W_9{}')$, $(W_{10}{}')$, $(W_{14}{}')$ und $(W_{15}{}')$ in Tabelle 4.4 erfüllt.

Ein *Typ* Typ(X) von G ist definiert als die Menge der Wörter der Sprache $L(G)$, die vom Nichtterminalsymbol $X \in N$ abgeleitet werden können und die semantischen Regeln $(S_1{}')$ und $(S_2{}')$ aus Tabelle 4.4 erfüllen:

$$\text{Typ}(X) = \{w \in L(X) \mid w \text{ erfüllt } (S_1{}') \text{ und } (S_2{}') \}.$$

Dabei bezeichnet $L(X)$ die in Definition 4.4 eingeführte Teilsprache. Die Menge $\{\text{Typ}(X) \mid X \in N\}$ aller Typen von G wird mit $\mathcal{T}$ bezeichnet. Ist das Nichtterminalsymbol X atomar bzw. abstrakt, so wird auch der Typ Typ(X) *atomar* bzw. *abstrakt* genannt. Das *Typdiagramm* $\mathcal{D}_T$ von G ist definiert als die Relation

$$\mathcal{D}_T \subseteq N \times N \text{ mit } (X, Y) \in \mathcal{D}_T \iff \text{Typ}(X) \subseteq \text{Typ}(Y).$$

Wenn Typ(X) ein Untertyp von Typ(Y) ist, d.h. $(X, Y) \in \mathcal{D}_T$, so schreibt man abkürzend $X \preceq Y$.

* * *

Bei einer Typgrammatik besteht die wesentliche Veränderung im Vergleich zu einer Objektgrammatik in der Einführung von Bedingungen in Definitionsregeln $(W_3{}')$. Damit die Bedingungen sich auf die Attribute in den Definitionsregeln beziehen können, werden die Attribute mit Attribut-Namen Σ_{Att} benannt $(W_4{}')$, $(W_5{}')$. Die Elemente von $\{a - z\}$ können nun nicht mehr als Namen von Nichtterminalsymbolen dienen, um eine Verwechslung zwischen Nichtterminalsymbolen und Attributnamen auszuschließen. Daher wird die Menge des zulässigen Nichtterminalsymbole im Vergleich mit der Objektgrammatik auf $N \subset \Sigma^+_{NTS} \backslash \Sigma_{Att}$ verkleinert. Für die hier betrachteten Beispiele sind Attributnamen ausreichend, die aus einem einzelnen Kleinbuchstaben bestehen. Bei Bedarf wäre es einfach, diese und andere Grundmengen der Grammatikdefinitionen zu erweitern.

Der Gültigkeitsbereich, der von einem Paar runder Klammern definiert wird, umfasst den Typbezeichner, die Attributliste und die Bedingung auf der Ebene der Klammern$(W_9{}')$. Trivial ist, dass ein Attributname in seinem Gültigkeitsbereich nur einmal auftreten darf, da andernfalls nicht klar wäre, auf welches Attribut sich eine Bedingung bezieht. Attribute auf einer tieferen oder höheren Schachtelungsebene sind für die Bedingung eines Gültigkeitsbereichs unsichtbar. Beispielsweise enthält das folgende Objekt zwei mit i benannte Attribute, die unterschiedlichen Bedingungen genügen müssen. Aus der Klammerung ergibt sich, dass in Objekten vom Typ **Eins** das erste Attribut eine positive ganze Zahl sein muss. Objekte vom Typ **Zwei** enthalten eine negative ganze Zahl.

$$(\textbf{Eins};\ i : (\mathit{Int}; 5), (\textbf{Zwei};\ i : (\mathit{Int}; -5); i < 0);\ i \geq 0)$$

Genaugenommen müssten die Bedingungen $i < (\mathbf{Int}; 0)$ und $i \geq (\mathbf{Int}; 0)$ lauten, d.h. man müßte Vergleichsrelationen $<$ und $\geq$ auf der Teilsprache $L(Int)$ definieren. Da die Bedeutung der booleschen Bedingungen bei Basisdatentypen aber klar ist, wird der Übersichtlichkeit halber die einfachere Schreibweise verwendet.

Damit ein Objekt semantisch gültig ist, müssen die Attribute die eventuell vorhandenen Bedingungen erfüllen ($S_1{}'$). Diese Bedingungen sind boolesche Ausdrücke, die lediglich die vorhandenen Attribute auswerten und einen Wahrheitswert berechnen. Seiteneffekte auf die Attributwerte (wie z.B. das Inkrementieren mit impliziter Zuweisung "i++") sind bei booleschen Ausdrücken ausgeschlossen.

Das folgende Lemma zeigt, dass das Teilsprachendiagramm einer Typgrammatik mit seinem Typdiagramm identisch ist, wenn jeder Typ mindestens ein semantisch gültiges Objekt hat ($S_2{}'$). Diese Gleichheit wird später bei der Erweiterbaren Zeitreihengrammatik benötigt, um das Typdiagramm durch Bestimmung der Teilsprachen ermitteln zu können.

Lemma 4.2 (Teilsprachen- und Typdiagramme)
In einer Typgrammatik G sind Typdiagramm und Teilsprachendiagramm gleich: $\mathcal{D}_T = \mathcal{D}_L$

Beweis. Es sei $S \subseteq L(G)$ die Menge aller semantisch gültigen Wörter in $L(G)$.

Zeige $\mathcal{D}_L \subseteq \mathcal{D}_T$. Diese Richtung der Gleichheit ergibt sich aus den Teilmengenbeziehungen zwischen Teilsprachen und Typen. Es sei $(X, Y) \in \mathcal{D}_L$, d.h. $L(X) \subseteq L(Y)$. Dann gilt:

$$\mathrm{Typ}(X) = L(X) \cap S \subseteq L(Y) \cap S = \mathrm{Typ}(Y),$$

also $(X, Y) \in \mathcal{D}_T$ und damit insgesamt $\mathcal{D}_L \subseteq \mathcal{D}_T$.

Zeige $\mathcal{D}_L \supseteq \mathcal{D}_T$. Es sei $(X, Y) \in \mathcal{D}_T$, also $\mathrm{Typ}(X) \subseteq \mathrm{Typ}(Y)$. Angenommen, $L(Y) \subset L(X)$. Da jede Teilsprache nach Lemma 4.1(e) eine disjunkte Zerlegung in atomare Teilsprachen besitzt, gibt es ein atomares Nichtterminalsymbol $A \in N$ mit $L(A) \subseteq L(X) \backslash L(Y)$. In einer Typgrammatik besitzt wegen ($S_2{}'$) jede Teilsprache mindestens ein semantisch gültiges Element, so dass $\mathrm{Typ}(A)$ nicht leer ist. Daher gilt:

$$\begin{aligned}
\emptyset &\neq \mathrm{Typ}(A) \\
&= L(A) \cap S \\
&\subset (L(X) \backslash L(Y)) \cap S \\
&= (L(X) \cap S) \backslash (L(Y) \cap S) \\
&= \mathrm{Typ}(X) \backslash \mathrm{Typ}(Y) \\
&= \emptyset
\end{aligned}$$

Aus diesem Widerspruch folgt, dass die Annahme falsch ist. Es ist also $L(X) \subseteq L(Y)$, $(X, Y) \in \mathcal{D}_L$ und insgesamt $\mathcal{D}_T \subseteq \mathcal{D}_L$ $\square$

4.2.4 Templategrammatik

Nachdem Objektgrammatiken durch Bedingungen zu Typgrammatiken erweitert wurden, um Teilsprachen auf eine für das maschinelle Lernen geeignete Größe reduzieren zu können, befassen wir uns nun mit der Frage, wie sich abstrakte Datenstrukturen in die Repräsentation von Zeitreihen integrieren lassen.

Beim maschinellen Lernen in musikalischen Anwendungen besteht das Trainingsmaterial aus Kompositionen, die jeweils durch eine Menge von Ansichten repräsentiert werden. Um automatisch Lernmuster aus ihnen berechnen zu können, ist es sinnvoll, alle Kompositionen des Trainingsmaterials gleich aufzubauen. Dies erreicht man, indem man die Elemente einer Ansicht auf einen Typ einschränkt und die Kompositionen aus Ansichten zusammengesetzt, die jeweils mit der gleichen Folge von Typen parametrisiert sind. In einer Objekt- und einer Typgrammatik ist eine solche Festlegung von Ansichten auf einen Typ nur möglich, wenn zu jedem musikalischen Typ (z.B. *Takt*, *Tonhöhe*, *THMidi*, *Motiv*) je ein spezieller Ansicht-Typ (hier *TaktAnsicht*, *TonhoehenAnsicht*, *THMidiAnsicht*, *MotivAnsicht*) definiert wird. Dieser Ansatz hat zwei Nachteile. Zum einen wird eine Fülle ähnlicher Definitionen erzeugt, die die Repräsentation unübersichtlich machen. Zum anderen löst er nicht das Problem, Kompositionen durch die Typen ihrer Ansichten zu charakterisieren, da die Benennung einer Ansicht z.B. als *TonhoehenAnsicht* nicht garantiert, dass sie tatsächlich Objekte des suggerierten Typs, also Tonhoehen, enthält.

Eine elegantere Lösung besteht in der Einführung eines Template-Mechanismus, wie er aus Programmiersprachen wie C++ bekannt ist. Templates parametrisieren einen abstrakten Typ mit einem Typparameter. Für diesen werden später alle benötigten Typen eingesetzt und dadurch konkrete Typen erzeugt.

Wenn man ein Template $\mathit{Ansicht}\langle X\rangle$ definiert, das von einem Platzhalter oder *Typparameter* X abhängt, erzeugt man durch Einsetzen von **Takt**, **Tonhöhe**, **THMidi**, **Motiv** die Typen Typ($\mathit{Ansicht}\langle\mathbf{Takt}\rangle$), Typ($\mathit{Ansicht}\langle\mathbf{Tonhöhe}\rangle$), Typ($\mathit{Ansicht}\langle\mathbf{THMidi}\rangle$) und Typ($\mathit{Ansicht}\langle\mathbf{Motiv}\rangle$). Eine Komposition kann dann vom eingesetzten Typparameter abhängig gemacht werden. Damit hat man zwei Ziele erreicht: Man vermeidet die Wiederholung ähnlicher Definitionen und erhält eine typsichere Parametrisierung abstrakter Datenstrukturen.

Im folgenden wird eine *Templategrammatik* definiert, bei der es sich um eine Objektgrammatik handelt, die mit Hilfe einer *Grammatiktemplate* genannten Schablone generiert wird. Ein Grammatiktemplate besteht aus *Nichtterminalsymboltemplates*, *Typbezeichnertemplates* und *Produktionstemplates* , aus denen mit einem Expansionsmechanismus Nichtterminalsymbole, Typbezeichner und Produktionen einer Templategrammatik erzeugt werden. Die Produktionen dieser expandierten Templategrammatik (und nicht die Produktionstemplates des Grammatiktemplates) werden dann zur Erzeugung von Objekten herangezogen.

Vorerst werden die Bedingungen, die in der Typgrammatik eingeführt wurden, wieder weggelassen, um die Erweiterungen der Objektgrammatiken durch Bedingungen und durch Templates nicht zu vermischen. In der Erweiterbaren Zeitreihengrammatik (Abschnitt 4.2.5) werden Bedingungen und Grammatiktemplates dann zusammengeführt. Bevor die Templategrammatik genau definiert wird, wird die Templategrammatik an einem Beispiel erläutert.

Beispiel 4.7 (Templategrammatik)
Gegeben seien die die Nichtterminalsymbole

$$U = \{\mathit{Objekt},\ \mathit{Takt},\ \mathit{Int},\ \mathit{Vektor}\},$$

aus denen mit der Alphabetwechsel-Abbildung α die Typbezeichner

$$\alpha(U) = \{\mathbf{Objekt},\ \mathbf{Takt},\ \mathbf{Int},\ \mathbf{Vektor}\}$$

gewonnen werden. Die Mengen

$$\begin{aligned}
\widetilde{N}_{TP} &= \{\mathit{Objekt},\ \mathit{Takt},\ \mathit{Int},\ \mathit{Vektor}\langle X\rangle\} \\
\widetilde{T}_{TP} &= \{\mathbf{Objekt},\ \mathbf{Takt},\ \mathbf{Int},\ \mathbf{Vektor}\,\langle X\rangle\} \\
\widetilde{P}_{TP} &= \left\{\begin{array}{llll}
(P_1) & \mathit{Objekt} & \longrightarrow & \mathit{Takt} \mid \mathit{Int} \mid \mathit{Vektor}\langle X\rangle \\
(P_2) & \mathit{Takt} & \longrightarrow & (\mathbf{Takt};\ \mathit{Int},\ \mathit{Int}) \\
(P_3) & \mathit{Int} & \longrightarrow & (\mathbf{Int};\ \mathbb{Z}) \\
(P_4) & \mathit{Vektor}\langle X\rangle & \longrightarrow & (\mathbf{Vektor}\,\langle X\,\rangle;\ [\widetilde{\alpha}^{-1}(X)\,[,\ \widetilde{\alpha}^{-1}(X)]^*])
\end{array}\right\}
\end{aligned}$$

bilden ein Grammatiktemplate. Das Nichtterminalsymboltemplate $\mathit{Vektor}\langle X\rangle$ aus $\widetilde{N}_{TP}$ wird durch das Einsetzen von Nichtterminalsymbolen für den Parameter X zu einer Menge von Nichtterminalsymbolen expandiert:

$$\begin{aligned}
\mathbf{subst}(\mathit{Vektor}\langle X\rangle) = \{\ & \mathit{Vektor}\langle\mathbf{Objekt}\rangle, \mathit{Vektor}\langle\mathbf{Takt}\rangle, \mathit{Vektor}\langle\mathbf{Int}\rangle, \\
& \mathit{Vektor}\langle\mathbf{Vektor}\,\langle\mathbf{Objekt}\rangle\rangle, \mathit{Vektor}\langle\mathbf{Vektor}\,\langle\mathbf{Takt}\rangle\rangle, \ldots\}
\end{aligned}$$

Ebenfalls mit der Expansionsfunktion **subst** und auf die gleiche Weise werden Typbezeichner T_{TP} aus den Typbezeichnertemplates $\widetilde{T}_{TP}$ und Produktionen P_{TP} aus den Produktionstemplates $\widetilde{P}_{TP}$ gewonnen. Im Beispiel müssen die das Typbezeichnertemplate $\mathbf{Vektor}\,\langle X\rangle$ und die Produktionstemplates (P_1) und (P_4) expandiert werden:

$$\begin{aligned}
N_{TP} &= \mathbf{subst}(\widetilde{N}_{TP}) = \mathbf{subst}(\mathit{Vektor}\langle X\rangle) \cup \{\mathit{Objekt},\ \mathit{Takt},\ \mathit{Int}\} \\
T_{TP} &= \mathbf{subst}(\widetilde{T}_{TP}) = \mathbf{subst}(\mathbf{Vektor}\,\langle X\rangle) \cup \{\mathbf{Objekt},\ \mathbf{Takt},\ \mathbf{Int}\} \\
P_{TP} &= \mathbf{subst}(\widetilde{P}_{TP}) = \mathbf{subst}((P_1)) \cup \mathbf{subst}((P_4)) \cup \{(P_2), (P_3)\}
\end{aligned}$$

Die expandierten Mengen des Grammatiktemplates bilden die Templategrammatik $G = (N_{TP}, T, P_{TP}, \mathit{Objekt})$, wobei die Terminalsymbole $T \supset T_{TP}$ sich aus den expandierten Typbezeichnern T_{TP} und weiteren Symbolen wie z.B. Interpunktionszeichen zur Kennzeichnung der Hierarchieebenen eines Objekts zusammensetzen.

* * *

Wie bei Objektgrammatiken gibt es auch bei Templategrammatiken eine Entsprechung zwischen Nichtterminalsymbolen und Typbezeichnern, um den Typ eines Objekts explizit durch

einen Typbezeichner an seinem Wurzelknoten sichtbar zu machen. Um dieses Verhalten auch in einer Templategrammatik zu erhalten, wird die bisher verwendete Alphabetwechsel-Abbildung α zu einer Abbildung $\widetilde{\alpha}$ erweitert, die auch Template-Parameter berücksichtigt. Dabei sind die Alphabetwechsel-Abbildungen α und $\widetilde{\alpha}$ auf ihr Argument anzuwenden, bevor ein Ausdruck expandiert oder abgeleitet wird. Da im Produktionstemplate (P_4) des Beispiels für den Parameter X Typbezeichner eingesetzt werden, ist der Ausdruck $\widetilde{\alpha}^{-1}(X)$ auf der rechten Seite also ein Platzhalter für ein Nichtterminalsymbol, z.B. für $\widetilde{\alpha}^{-1}(\mathbf{Takt}) = \mathit{Takt}$.

Symbole. Ausgangspunkt für die Konstruktion der Templates sind die Stammsymbole U, die den Stamm der Namen von Nichtterminalsymboltemplates $\widetilde{N}_{TP}$ und der Typbezeichnertemplates $\widetilde{T}_{TP}$ bilden. Um ein Nichtterminalsymboltemplate zu bilden, hängt man an ein Stammsymbol einen Templateparameter an, der als Platzhalter für einen Typ dient. Zur Konstruktion eines Typbezeichnertemplates bildet man das Stammsymbol mit der Alphabetwechsel-Abbildung α in das Typbezeichner-Alphabet Σ_{TB} ab und hängt einen Templateparameter an. Beispielsweise ist $\mathit{Vektor}\langle X\rangle$ ein Nichtterminalsymboltemplate und $\widetilde{\alpha}(\mathit{Vektor}\langle X\rangle) = \alpha(\mathit{Vektor})\langle X\rangle = \mathbf{Vektor}\langle X\rangle$ ein Typbezeichnertemplate. Aus den Templates gewinnt man Nichtterminalsymbole und Typbezeichner, indem man für die Parameter X Typbezeichner einsetzt. Für $X = \mathbf{Takt}$ erhält man z.B. das Nichtterminalsymbol $\mathit{Vektor}\langle\mathbf{Takt}\rangle$ und den Typbezeichner $\mathbf{Vektor}\langle\mathbf{Takt}\rangle$.

Beispiel für ein Objekt. Wenn man den Parameter $X = \mathbf{Takt}$ in das Produktionstemplate (P_4) einsetzt und die Alphabetwechsel-Abbildung $\widetilde{\alpha}^{-1}(\mathbf{Takt}) = \mathit{Takt}$ auswertet, erhält man die Produktion

$$\mathit{Vektor}\langle\mathbf{Takt}\rangle \longrightarrow (\mathbf{Vektor}\langle\mathbf{Takt}\rangle\,;\ [\mathit{Takt}[,\ \mathit{Takt}]^*])$$

In (P_1) setzt man für X alle Typbezeichner ein:

$$\begin{aligned}\mathit{Objekt} \longrightarrow\ & \mathit{Takt} \mid \mathit{Int} \mid \mathit{Vektor}\langle\mathbf{Objekt}\rangle \mid \mathit{Vektor}\langle\mathbf{Takt}\rangle \mid \mathit{Vektor}\langle\mathbf{Int}\rangle \mid \\ & \mathit{Vektor}\langle\mathit{Vektor}\langle\mathbf{Objekt}\rangle\rangle \mid \ldots\end{aligned}$$

Aus dieser und den anderen, nicht parametrisierten Produktionen aus $\widetilde{P}_{TP}$ lässt sich z.B. der folgende Taktvektor ableiten, der aus einem 3/4- und einem 4/4-Takt besteht:

$$(\mathbf{Vektor}\langle\mathbf{Takt}\,\rangle\,;\ (\mathbf{Takt};\ (\mathbf{Int};3),\ (\mathbf{Int};4)), (\mathbf{Takt};\ (\mathbf{Int};4),\ (\mathbf{Int};4)))$$

Abbildung 4.10 zeigt die Baumdarstellung für dieses Objekt, in der der gewählte Parameter $X = \mathbf{Takt}$ an den Knoten für das Templatesymbol $\mathbf{Vektor}\langle X\rangle$ angefügt ist. Die zugehörige Ableitung sieht wie folgt aus:

Objekt	$\vdash$	$\mathit{Vektor}\langle\mathbf{Takt}\rangle$	(P_1) expandiert
	$\vdash$	$(\mathbf{Vektor}\langle\mathbf{Takt}\rangle\,;\ [\mathit{Takt}[,\ \mathit{Takt}]^*])$	(P_4) mit $X = \mathbf{Takt}$
	$(\vdash)$	$(\mathbf{Vektor}\langle\mathbf{Takt}\rangle\,;\ \mathit{Takt}, \mathit{Takt})$	Auswahl
	$\vdash$	$(\mathbf{Vektor}\langle\mathbf{Takt}\rangle\,;\ (\mathbf{Takt};\ \mathit{Int},\ \mathit{Int}),\ (\mathbf{Takt};\ \mathit{Int},\ \mathit{Int}))$	(P_2)
	$\vdash$	$(\mathbf{Vektor}\langle\mathbf{Takt}\rangle\,;\ (\mathbf{Takt};\ (\mathbf{Int};\ 3),\ (\mathbf{Int};\ 4)),\ (\mathbf{Takt};\ (\mathbf{Int};\ 4),\ (\mathbf{Int};\ 4)))$	(P_3)

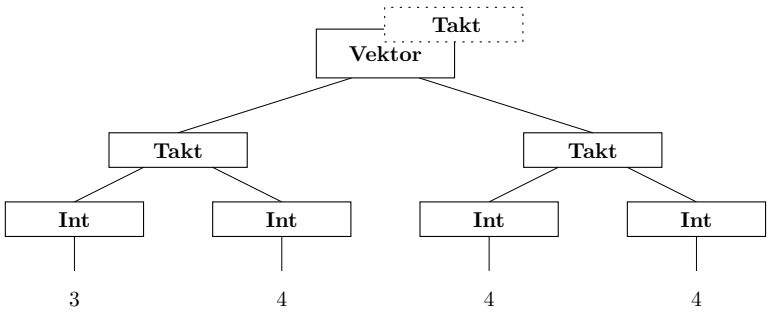


Abbildung 4.10: Ein mit **Takt** instantiierter Vektor

In der ersten Zeile wird das Startsymbol *Objekt* mit der expandierten Version der Alternativenregel (P_1) zu *Vektor*⟨**Takt**⟩ abgeleitet. Im zweiten Schritt wird die Definitionsregel (P_4) angewendet, wobei $X =$ **Takt** gewählt wurde. Nun wird aus der Menge, die durch den regulären Ausdruck [*Takt*[, *Takt*]*] beschrieben wird, das Element '*Takt*, *Takt*' ausgewählt. Da keine Produktion angewendet wird, handelt es sich hier nicht um einen Ableitungsschritt. Zuletzt werden die Nichtterminalsymbole *Takt* mit (P_2) und (P_3) in Takt-Objekte umgewandelt, die einen 3/4- und einen 4/4-Takt darstellen. Im fertig abgeleiteten Ausdruck zeigt nur noch der Typbezeichner **Vektor** ⟨**Takt**⟩ an, dass bei der Entstehung expandierte Produktionstemplates beteiligt waren.

Expansion von Produktionstemplates. Der oben angegebene Taktvektor wurde abgeleitet, indem Typbezeichner in Produktionstemplates eingesetzt wurden. Durch Anwendung der so gewonnenen Produktionen wurden auf die herkömmliche Weise Nichtterminalsymbole ersetzt, bis alle Nichtterminalsymbole eliminiert waren.

Analog zu Objektgrammatiken wird auch in Templategrammatiken zwischen Alternativenregeltemplates und Definitionsregeltemplates unterschieden. Ein Alternativenregeltemplate hat nur auf der rechten Seite Templateparameter. Es wird expandiert, indem in die Nichtterminalsymboltemplates auf der rechten Seite der Regel alle erlaubten Typen eingesetzt werden und die so entstandenen zusammengesetzten Nichtterminalsymbole zur Auswahl angeboten werden. Dadurch ergibt sich eine Alternativenregel, die unendlich lang sein kann. Im Beispiel wird (P_1) expandiert zu:

$$
\begin{aligned}
\mathit{Objekt} \longrightarrow\ & \mathit{Takt} \mid \mathit{Int} \mid \mathit{Vektor}\langle\mathbf{Objekt}\rangle \mid \mathit{Vektor}\langle\mathbf{Takt}\rangle \mid \mathit{Vektor}\langle\mathbf{Int}\rangle \\
& \mid \mathit{Vektor}\langle\mathbf{Vektor}\,\langle\mathbf{Objekt}\rangle\rangle \mid \mathit{Vektor}\langle\mathbf{Vektor}\,\langle\mathbf{Takt}\rangle\rangle \mid \mathit{Vektor}\langle\mathbf{Vektor}\,\langle\mathbf{Int}\rangle\rangle \\
& \mid \mathit{Vektor}\langle\mathbf{Vektor}\,\langle\mathbf{Vektor}\,\langle\mathbf{Objekt}\rangle\rangle\rangle \mid \ldots
\end{aligned}
$$

Ein Definitionsregeltemplate hat ein Nichtterminalsymboltemplate auf der linken Seite, dessen Parameter die Expansion der ganzen Regel steuert. Jeder Typ, der für den Parameter des Nichtterminalsymboltemplates gewählt werden darf, erzeugt eine neue Regel, auf deren rechter Seite ebenfalls der gewählte Typ eingesetzt wird. Während ein Alternativenregeltemplate durch eine verlängerte Alternativenregel ersetzt wird, entsteht aus einem Definitionsregeltemplate durch Expansion eine Menge von Definitionsregeln. Im Beispiel ergeben sich bei

der Expansion von (P_4) unendlich viele Produktionen :

$$
\begin{array}{lcl}
\mathit{Vektor}\,\langle\mathbf{Objekt}\rangle & \longrightarrow & (\mathbf{Vektor}\,\langle\mathbf{Objekt}\rangle\,;\,[\mathit{Objekt}[,\ \mathit{Objekt}]^*]) \\
\mathit{Vektor}\,\langle\mathbf{Takt}\rangle & \longrightarrow & (\mathbf{Vektor}\,\langle\mathbf{Takt}\rangle\,;\,[\mathit{Takt}[,\ \mathit{Takt}]^*]) \\
\mathit{Vektor}\,\langle\mathbf{Int}\rangle & \longrightarrow & (\mathbf{Vektor}\,\langle\mathbf{Int}\rangle\,;\,[\mathit{Int}[,\ \mathit{Int}]^*]) \\
\mathit{Vektor}\,\langle\mathbf{Vektor}\,\langle\mathbf{Objekt}\rangle\rangle & \longrightarrow & (\mathbf{Vektor}\,\langle\mathbf{Vektor}\,\langle\mathbf{Objekt}\rangle\rangle\,; \\
 & & [\mathit{Vektor}\,\langle\mathbf{Objekt}\rangle\,[,\ \mathit{Vektor}\,\langle\mathbf{Objekt}\rangle]^*]) \\
 & \vdots &
\end{array}
$$

Als Parameter von *Vektor* sind auch abstrakte Typen wie **Objekt** erlaubt. Ein solcher gemischter Vektor lässt alle Objekte als Elemente zu, die von *Objekt* abgeleitet sind. In diesem Beispiel sind das beliebige Objekte, da *Objekt* das Startsymbol der Grammatik ist. Auch restriktivere Vektoren, die z.B. verschiedene Arten von Tonhöhen enthalten, sind denkbar. Im Vergleich zu Objektgrammatiken und Typgrammatiken ist die Verwendung von abstrakten Typbezeichnern wie **Objekt** in Templateargumenten hier neu, während bisher Typbezeichner nur eingesetzt wurden, um an der ersten Position eines Objekts die speziellste Teilsprache anzugeben, der das Objekt angehört.

Die Produktionen (P_2) und (P_3) müssen nicht expandiert werden, da es sich nicht um Produktionstemplates handelt.

Unendliche Grundmengen der Templategrammatik. Da die so gewonnene Grammatik eine unendlich lange Alternativenregel und unendlich viele Definitionsregeln besitzt, kann sie nicht explizit angegeben werden. Die gültigen Alternativen- und Definitionsregeln lassen sich aber bei Bedarf herleiten, so dass man auch mit der implizit definierten Grammatik Objekte ableiten kann. Die Ableitung greift dabei ausschließlich auf die Produktionen der expandierten Grammatik zurück. In Abschnitt 4.2.4.2 wird ein Algorithmus angegeben, mit dem man eine expandierte Grammatik bis zur benötigten Schachtelungstiefe berechnen kann.

Teilsprachen. Wie für Objekt- und Typgrammatiken kann man auch für Templategrammatiken Teilsprachen bilden. Die Teilsprache $L(\mathit{Vektor}\,\langle\mathbf{Takt}\rangle)$ besteht z.B. aus allen Vektoren, die eine endliche Anzahl von **Takt**-Objekten enthalten. Die expandierte Alternativenregel (P_1) liefert die allgemeinste Teilsprache $L(\mathit{Objekt}) = L(\mathit{Takt}) \cup L(\mathit{Int}) \cup L(\mathit{Vektor}\,\langle\mathbf{Objekt}\rangle \cup L(\mathit{Vektor}\,\langle\mathbf{Takt}\rangle \cup L(\mathit{Vektor}\,\langle\mathbf{Int}\rangle) \cup \ldots$, d.h. die von der Grammatik erzeugte Sprache $L(G) = L(\mathit{Objekt})$.

Da man Templates beliebig tief schachteln kann, ist die Menge aller Teilsprachen $\mathcal{L}(G)$ wie die Menge der Nichtterminalsymbole und der Typbezeichnertemplate für das Beispiel unendlich groß:

$$
\begin{array}{rcl}
\mathcal{L}(G) & = & \{\ L(\mathit{Objekt}),\ L(\mathit{Takt}),\ L(\mathit{Int}), \\
 & & L(\mathit{Vektor}\,\langle\mathbf{Objekt}\rangle),\ L(\mathit{Vektor}\,\langle\mathbf{Takt}\rangle),\ L(\mathit{Vektor}\,\langle\mathbf{Int}\rangle), \\
 & & L(\mathit{Vektor}\,\langle\mathbf{Vektor}\,\langle\mathbf{Objekt}\rangle\rangle), \\
 & & L(\mathit{Vektor}\,\langle\mathbf{Vektor}\,\langle\mathbf{Takt}\rangle\rangle), \\
 & & L(\mathit{Vektor}\,\langle\mathbf{Vektor}\,\langle\mathbf{Int}\rangle\rangle), \\
 & & L(\mathit{Vektor}\,\langle\mathbf{Vektor}\,\langle\mathbf{Vektor}\,\langle\mathbf{Objekt}\rangle\rangle\rangle),\ \ldots\}
\end{array}
$$

Definition 4.8 (Templategrammatik)
Gegeben seien die Alphabete $\Sigma_{NTS} = \{A - Z,\ a - z,\ 0 - 9\}$, $\Sigma_{TB} = \{\mathbf{A - Z},\ \mathbf{a - z},\ \mathbf{0 - 9}\}$ und $\Sigma_{Str} = \{\mathrm{A - Z},\ \mathrm{a - z},\ 0 - 9\}$, eine Menge von *Interpunktionszeichen* $\Sigma_I = \{\ (\ )\ \langle\ \rangle\ :\ ;\ ,\ \preceq\ \}$, die *Wertemenge* $\Sigma_W = \Sigma^*_{Str} \cup \mathbb{Q}$, eine Menge von *Parameternamen* $\Sigma_{Par} = \{A - Z\}$ zur Referenzierung von Typparametern in Typbedingungen und das *Gesamtalphabet* $\Sigma = \Sigma_{NTS} \cup \Sigma_{TB} \cup \Sigma_I \cup \Sigma_W \cup \Sigma_{Att} \cup \Sigma_{Par}$ zur Definition der erweiterten Alphabetwechsel-Abbildung.

Von diesen Grundmengen ausgehend wählt man *Stammsymbole* $U \subset \Sigma^+_{NTS} \backslash \Sigma_{Par}$, um daraus *Nichtterminalsymboltemplates* $\widetilde{N}_{TP}$, *Typbezeichnertemplates* $\widetilde{T}_{TP}$ und *Produktionstemplates* $\widetilde{P}_{TP}$ zu bilden. Die genannten Mengen müssen endlich sein und die Wohlgeformtheitsregeln ($W_1{''}$)–($W_{15}{''}$) in Tabelle 4.4 erfüllen. Die *erweiterte Alphabetwechsel-Abbildung* $\widetilde{\alpha} : U\Sigma^* \longrightarrow \alpha(U)\Sigma^*$ überführt Nichtterminalsymboltemplates in Typbezeichnertemplates ($W_7{''}$). Das Tripel $(\widetilde{N}_{TP}, \widetilde{T}_{TP}, \widetilde{P}_{TP})$ heißt *Grammatiktemplate*.

Aus einem Grammatiktemplate erzeugt man eine *Templategrammatik* $G = (N_{TP},\ T,\ P_{TP},\ S)$ mit Nichtterminalsymbolen $N_{TP} = \mathbf{subst}(\widetilde{N}_{TP})$, Terminalsymbolen $T = T_{TP} \cup \Sigma_I \cup \Sigma_W \cup \Sigma_{Att} \cup \Sigma_{Par}$ mit $T_{TP} = \mathbf{subst}(\widetilde{T}_{TP})$, Produktionen $P_{TP} = \mathbf{subst}(\widetilde{P}_{TP})$ und einem Startsymbol $S \in N_{TP}$, indem man die Mengen $\widetilde{N}_{TP}$, $\widetilde{T}_{TP}$, $\widetilde{P}_{TP}$ des Grammatiktemplates mit Hilfe der in ($W_{11}{''}$) definierten *Expansionsfunktion* **subst** expandiert.

* * *

Templatemengen. Ausgangspunkt für die Definition der Templategrammatik ist die Objektgrammatik aus Definition 4.4. Neu ist, dass eine Templategrammatik nicht direkt durch ihre Grundmengen, sondern indirekt über Templatemengen eines Grammatiktemplates definiert wird. Analog zu den Nichtterminalsymbolen, den Typbezeichnern und den Produktionen gibt es drei Templatemengen: Nichtterminalsymboltemplates $\widetilde{N}_{TP}$ ($W_6{''}$), Typbezeichnertemplates $\widetilde{T}_{TP}$ ($W_7{''}$) und Produktionstemplates $\widetilde{P}_{TP}$ (($W_1{''}$),($W_2{''}$), ($W_3{''}$)). Aus diesen Mengen werden die Grundmengen der Templategrammatik generiert, indem man mit der Expansionsfunktion **subst** zu jedem Template die Menge aller Instanzen berechnet, durch die man das Template ersetzen darf ($W_{11}{''}$). Nichtterminalsymboltemplates werden durch Nichtterminalsymbole, Typbezeichnertemplates durch Typbezeichner und Produktionstemplates durch Produktionen ersetzt.

Konstruktion der Symbole. Die Nichtterminalsymboltemplates und Typbezeichnertemplates werden aus den Stammsymbolen in U gebildet, die genau einmal als Anfang eines Nichtterminalsymboltemplate und eines Typbezeichnertemplate auftreten müssen ($W_8{''}$). Für ein Stammsymbol $X \in U$ hat ein Nichtterminalsymboltemplate die Form

$$X\ [\,\langle\text{Templateargument}\rangle\,]$$

Ein Typbezeichnertemplate hat die Form

$$\alpha(X)\ [\,\langle\text{Templateargument}\rangle\,]$$

Ohne Templateargument erhält man ein Nichtterminalsymbol als Spezialfall eines Nichtterminalsymboltemplates bzw. einen Typbezeichner als Spezialfall eines Typbezeichnertemplates. Da jedes Stammsymbol genau einmal am Anfang eines Nichtterminalsymboltemplates

Symbolart	**Typparameter**
Nichtterminalsymboltemplate	*Vektor*$\langle X \preceq$ **Objekt**$\rangle$
zusammengesetztes Nichtterminalsymbol	*Vektor*$\langle$**Int**$\rangle$
einfaches Nichtterminalsymbol und Stammsymbol	*Int*
Typbezeichnertemplate	**Vektor** $\langle X \rangle$
zusammengesetzter Typbezeichner	**Vektor** $\langle$**Int**$\rangle$
einfacher Typbezeichner und α(Stammsymbol)	**Int**

Tabelle 4.5: Symbolarten in einer Templategrammatik

auftritt, kann es, wenn es in der Menge der Nichtterminalsymboltemplates $\widetilde{N}_{TP}$ bereits ein Nichtterminalsymbol *Int* gibt, daher nicht zusätzlich ein Nichtterminalsymboltemplate *Int*$\langle$Templateargument$\rangle$ in $\widetilde{N}_{TP}$ geben. Zur Vereinfachung des Modells darf in Templates nur ein Parameter verwendet werden.

Symbolarten. Da die Symbole in einer Templategrammatik aus Stammsymbolen konstruiert werden, unterscheidet man zwischen einfachen und zusammengesetzten expandierten Symbolen. Nichtterminalsymbole und Typbezeichner sind *zusammengesetzt*, wenn sie durch das Ersetzen eines Templateparameters entstanden sind, und *einfach*, wenn es sich gleichzeitig auch um ein Stammsymbol handelt. Tabelle 4.5 zeigt Beispiele für die verschiedenen Symbolarten. Zusätzlich werden die Symbole wie in der Objektgrammatik in atomare und abstrakte Symbole eingeteilt, je nachdem ob sie als linke Seite einer Definitions- oder einer Alternativenregel auftreten. Zusammengesetzte Symbole sind immer atomar, da auf der linken Seite eines Alternativenregeltemplates kein Nichtterminalsymboltemplate, sondern nur ein Nichtterminalsymbol zulässig ist ($W_2{''}$). Einfache Symbole können abstrakt oder atomar sein.

Erweiterte Alphabetwechsel-Abbildung. Wie in Objektgrammatiken gibt es auch Templategrammatiken eine Entsprechung zwischen den Nichtterminalsymboltemplates und den Typbezeichnertemplates. Sie wird durch eine erweiterte Alphabetwechsel-Abbildung $\widetilde{\alpha}$ hergestellt ($W_7{''}$), die im Vergleich zu α nicht nur die Stammsymbole U, sondern auch Temp-

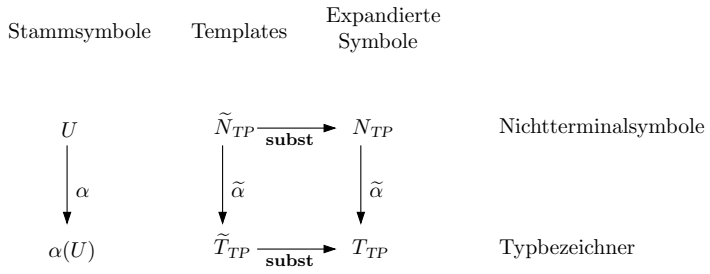


Abbildung 4.11: Grundmengen einer Templategrammatik

lateargumente berücksichtigt. $\widetilde{\alpha}$ ersetzt den Nichtterminalsymbol-Stamm eines Nichtterminalsymboltemplates durch den entsprechenden Typbezeichner-Stamm und übernimmt das eventuell vorhandene Templateargument unverändert, beispielsweise:

$$\widetilde{\alpha}(\mathit{Vektor}\,\langle\mathbf{Vektor}\,\langle\mathbf{Int}\rangle\rangle) = \alpha(\mathit{Vektor})\,\langle\mathbf{Vektor}\,\langle\mathbf{Int}\rangle\rangle = \mathbf{Vektor}\,\langle\mathbf{Vektor}\,\langle\mathbf{Int}\rangle\rangle$$

Abbildung 4.11 zeigt, wie die Templatemengen und die Symbole der erzeugten Grammatik zusammenhängen. Da jedes Stammsymbol genau einmal als Anfang eines Nichtterminalsymboltemplates und eines Typbezeichnertemplates dient (W_8'') und Templateargumente von Nichtterminalsymboltemplates und Typbezeichnertemplates auf die gleiche Weise expandiert werden, ist $\widetilde{\alpha}$ eine Bijektion sowohl zwischen den Nichtterminalsymboltemplates $\widetilde{N}_{TP}$ und den Typbezeichnertemplates $\widetilde{T}_{TP}$ als auch zwischen ihren Expansionen N_{TP} und T_{TP}. Daher konnte in Beispiel 4.7 im Produktionstemplate (P_4) ohne Bedenken die Umkehrung $\widetilde{\alpha}^{-1}$ angewendet werden, um ein Nichtterminalsymboltemplate zu einem gegebenen Typbezeichnertemplate zu ermitteln.

Wohlgeformtheitsregeln für Produktionstemplates. Analog zur Objektgrammatik sind Produktionstemplates in einer Templategrammatik entweder Alternativenregeltemplates oder Definitionsregeltemplates (W_1''). Das Symbol auf der linken Seite eines Alternativenregeltemplates ist immer ein echtes Nichtterminalsymbol und kein Nichtterminalsymboltemplate. Ein Alternativenregeltemplate wird zu einer Alternativenregel expandiert, die verschiedene Möglichkeiten anbietet, ein Nichtterminalsymbol zu ersetzen (W_2''). Ein Definitionsregeltemplate spezifiziert eine Menge von Definitionsregeln, die mit einem Nichtterminalsymboltemplate parametrisiert sind (W_3''). Auf der linken Seite des Definitionsregeltemplates befindet sich ein Nichtterminalsymboltemplate aus $\widetilde{N}_{TP}$. Die rechte Seite setzt sich aus einem Typbezeichnertemplate und einer Attributliste zusammen, die beide vom Parameter auf der linken Seite abhängen können. Das Typbezeichnertemplate zeigt den Typ eines Objekts an und muss zum Nichtterminalsymboltemplate auf der linken Seite der Regel passen, d.h. es muss $\widetilde{\alpha}(\mathit{NTST}) = \mathit{TBT}$ gelten. Wie in Definitionsregeln von Objektgrammatiken legt die Attributliste die Elemente eines Objekts fest (W_4'').

Globale Wohlgeformtheitsregeln. Wie bei Objekt- und Typgrammatiken bestimmen auch bei Templategrammatiken zwei globale Regeln den Zusammenhang zwischen den verschiedenen Mengen. Jedes Nichtterminalsymboltemplate aus $\widetilde{N}_{TP}$ muss genau einmal als linke Seite eines Produktionstemplates auftreten (W_{14}''). Jedes expandierte Nichtterminalsymbol aus N_{TP} muss durch Anwendung expandierter Alternativenregeln aus P_{TP} vom Startsymbol aus erreichbar sein (W_{15}'').

Aus dem eindeutigen Anfang der Nichtterminalsymboltemplates, Typbezeichnertemplates (W_{14}'') und Produktionstemplates (W_8'') und der Endlichkeit von U folgt, dass die drei Templatemengen eines Grammatiktemplates gleich groß und ebenfalls endlich sind: $|\widetilde{N}_{TP}| = |\widetilde{T}_{TP}| = |\widetilde{P}_{TP}| = |U|$.

Expansion von Templatemengen. Die Wohlgeformtheitsregeln (W_{11}''), (W_{12}'') und (W_{13}'') beschreiben, wie man Templates mit der Expansionsfunktion **subst** expandiert. Wie in Beispiel 4.7 beschrieben, werden Alternativenregeltemplates horizontal (W_{12}'') und Definitionsregeltemplates vertikal (W_{13}'') expandiert. (W_{11}'') spezifiziert, wie die Expansionsfunktion **subst** Templateparameter ersetzt: Nichtterminalsymbole und Typbezeichner

werden durch die Expansionsfunktion **subst** nicht verändert, da sie nicht parametrisiert sind. Nichtterminalsymboltemplates werden durch das Einsetzen von Typbezeichnern für den Parameter, Typbezeichnertemplates werden durch Transformation mit der Alphabetwechsel-Abbildung $\widetilde{\alpha}$, Expansion der erzeugten Nichtterminalsymboltemplates und Rücktransformation mit $\widetilde{\alpha}^{-1}$expandiert. Mengen werden expandiert, indem ihre Elemente expandiert und das Ergebnis zu einer Menge vereinigt wird.

4.2.4.1 Typbedingungen

Templategrammatiken stellen einen allgemeinen Mechanismus zur Definition von parametrisierten Typen zur Verfügung. Für einen Typparameter können alle Typbezeichner eingesetzt werden, insbesondere auch solche, die selbst Instanzen von Typbezeichnertemplates sind. Bei der Modellierung von Zeitreihenproblemen kann es jedoch sinnvoll sein, mit Hilfe sogenannter *Typbedingungen* nur bestimmte, problemangepasste Ersetzungen zuzulassen.

Beispiel 4.9 (Beispiel mit Typbedingung)
Die Mengen

$$
\begin{aligned}
U &= \{\mathit{Objekt},\ \mathit{Tonhöhe},\ \mathit{THMidi},\ \mathit{THPentatonischEs},\ \mathit{Tonhöhenvektor}\}\\
\alpha(U) &= \{\textbf{Objekt},\ \textbf{Tonhöhe},\ \textbf{THMidi},\ \textbf{THPentatonischEs},\ \textbf{Tonhöhenvektor}\}\\
\widetilde{N}_{TP} &= \{\mathit{Objekt},\ \mathit{Tonhöhe},\ \mathit{THMidi},\ \mathit{THPentatonischEs},\ \mathit{Tonhöhenvektor}\,\langle X \preceq \textbf{Tonhöhe}\rangle\}\\
\widetilde{T}_{TP} &= \{\textbf{Objekt},\ \textbf{Tonhöhe},\ \textbf{THMidi},\ \textbf{THPentatonischEs},\ \textbf{Tonhöhenvektor}\,\langle X\rangle\}\\
\widetilde{P}_{TP} &= \{\\
&\quad (P_1)\ \ \mathit{Objekt} \longrightarrow \mathit{Tonhöhe} \mid \mathit{Tonhöhenvektor}\,\langle X \preceq \textbf{Tonhöhe}\rangle\\
&\quad (P_2)\ \ \mathit{Tonhöhe} \longrightarrow \mathit{THMidi} \mid \mathit{THPentatonischEs}\\
&\quad (P_3)\ \ \mathit{THMidi} \longrightarrow (\textbf{THMidi};\ \{0,\ldots,127\})\\
&\quad (P_4)\ \ \mathit{THPentatonischEs} \longrightarrow (\textbf{THPentatonischEs};\ \{es,f,g,b,c\})\\
&\quad (P_5)\ \ \mathit{Tonhöhenvektor}\ \langle X \preceq \textbf{Tonhöhe}\rangle \longrightarrow (\textbf{Tonhöhenvektor}\,\langle X\rangle\,;\ [\widetilde{\alpha}^{-1}(X)\,[,\ \widetilde{\alpha}^{-1}(X)]^*])\\
&\}
\end{aligned}
$$

erzeugen eine Templategrammatik $G = (N_{TP}, T, P_{TP}, \mathit{Objekt})$ mit $T = \textbf{subst}(\widetilde{T}_{TP}) \cup \Sigma_I \cup \Sigma_W \cup \Sigma_{Att} \cup \Sigma_{Par}$.

* * *

Parametrisierte Tonhöhenvektoren. Einerseits ist es häufig nicht sinnvoll, in einer musikalischen Repräsentation verschiedene Arten von Tonhöhen wie die MIDI-Darstellung und eine pentatonische Skala zu vermischen, weil dadurch die Herstellung von Bezügen innerhalb einer Komposition erschwert wird. Andererseits ist es zweckmäßig, allgemeine Konzepte für Tonhöhen nicht für jeden besonderen Tonhöhentyp neu formulieren zu müssen.

Beispiel 4.9 zeigt, wie durch Templateparameter beides erreicht wird. Der Tonhöhenvektor (P_5) wird mit einem Typbezeichner instanziiert, der **Tonhöhe** spezialisiert: $X \preceq \textbf{Tonhöhe}$.

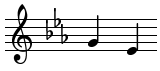
Auf diese Weise ist man sicher, dass ein Tonhöhenvektor immer Tonhöhen enthält, und diese zum gewählten Typ gehören. Wenn man z.B. weiß, dass es für Tonhöhen desselben Typs immer möglich ist, Intervalle zu berechnen, kann man einen Algorithmus zur Analyse einer Tonfolge schreiben, ohne den konkreten Typ der Tonhöhen kennen zu müssen.

Im Definitionsregeltemplate (P_5) hat der mit einem kursiv gedruckten Großbuchstaben bezeichnete Typparameter X zwei Aufgaben. Zum einen erlaubt er, den für den Parameter eingesetzten Typ auch auf der rechten Seite der Regel zu verwenden. Zum anderen ist X Teil der Typbedingung, die die zulässigen Typen einschränkt.

Tonhöhentypen. Im Beispiel sind zwei Arten von Tonhöhen definiert – *THMidi* (P_3) und *THPentatonischEs* (P_4) –, die von der Alternativenregel (P_2) erzeugt werden. Das Attribut der MIDI-Tonhöhe ist von vorneherein auf die ganzen Zahlen $\{0, \ldots, 127\}$ beschränkt, ohne das Nichtterminalsymbol *Int* zu verwenden. Dadurch ist die Darstellung knapper und man benötigt keine Typbedingung, um die *Int*-Objekte wieder einzuschränken. Allerdings kann man die Attribute nicht zu *Int*-Objekten in Beziehung setzen, weil man keine Typinformation über sie besitzt: Es könnte sich bei den Attributwerten ebensogut um Zeichenketten handeln. Die pentatonische Skala wird für einen festen Grundton *es* definiert. Jeder Ton der Skala wird oktavinvariant durch eine Zeichenfolge dargestellt.

Expansion. Das Definitionsregeltemplate (P_5) wird expandiert, indem alle Typbezeichner für X eingesetzt werden, die die Bedingung $X \preceq$ **Tonhöhe** erfüllen, also **Tonhöhe** selbst und die beiden Untertypen **THMidi** und **THPentatonischEs**. Man erhält die expandierten Produktionen:

Tonhöhenvektor ⟨**Tonhöhe**⟩	⟶	(**Tonhöhenvektor** ⟨**Tonhöhe**⟩ ; [*Tonhöhe* [, *Tonhöhe*]*])
Tonhöhenvektor ⟨**THMidi**⟩	⟶	(**Tonhöhenvektor** ⟨**THMidi**⟩ ; [*THMidi* [, *THMidi*]*])
Tonhöhenvektor ⟨**THPentatonischEs**⟩	⟶	(**Tonhöhenvektor** ⟨**THPentatonischEs**⟩ ; [*THPentatonischEs* [, *THPentatonischEs*]*])

Die erste Definitionsregel ist mit dem abstrakten Typbezeichner **Tonhöhe** instanziiert und erzeugt einen gemischten Tonhöhenvektor mit beliebigen Typen von Tonhöhen, d.h. eine Mischung von Objekten verschiedener Tonhöhentypen. Die Tonfolge  kann dann z.B. durch den gemischten Tonhöhenvektor

(**Tonhöhenvektor** ⟨**Tonhöhe**⟩ ; (**THMidi**; (**Int**; 67)),
(**THPentatonischEs**; *es*))

repräsentiert werden. Die zweite und dritte Definitionsregel sind mit einem atomaren Typbezeichner instanziiert und enthalten nur Objekte eines Typs, so dass sich für die obige Tonfolge die pentatonische Darstellung

(**Tonhöhenvektor** ⟨**THPentatonischEs**⟩ ; (**THPentatonischEs**; *g*),
(**THPentatonischEs**; *es*))

ergibt. Die zweite Repräsentation der Tonfolge ist leichter weiterzuverarbeiten, weil sie genauere Information über die Elemente des Vektors zur Verfügung stellt.

Bedingungen in Typ- und Templategrammatiken. Während Bedingungen in Definitionsregeln von Typgrammatiken dazu dienen, ein Objekt, nachdem es erzeugt wurde, auf seine semantische Gültigkeit zu überprüfen, werden Typbedingungen in Templategrammatiken schon bei der Expansion der Templates ausgewertet, also um die Grammatik zu generieren, aus der später Objekte erzeugt werden. In der expandierten Grammatik werden die Bedingungen nicht mehr benötigt und sind daher nicht mehr sichtbar.

4.2.4.2 Praktische Umsetzung der Expansion

Die Definition der Templategrammatik beschreibt, wie man Templates expandieren darf (W_{11}''), aber nicht, auf welche Weise man alle expandierten Symbole und Produktionen aufzählt.

Wenn man Templatemengen expandieren will, stößt man auf das Problem, dass Typbedingungen in einem Templateargument ausgewertet werden, indem man das Teilsprachendiagramm $\mathcal{D}_L$ (vgl. Definition 2.8) konsultiert. Dieses ist durch Inklusionen zwischen Teilsprachen definiert, die ihrerseits die Existenz von Objekten erfordern. Die Erzeugung von Objekten setzt jedoch die expandierte Templategrammatik voraus, die gerade erzeugt werden soll – ein Zirkelschluss.

Glücklicherweise kann man den Zirkelschluss umgehen, indem man die Templatemengen iterativ expandiert. Dabei macht man sich zunutze, dass die Produktionstemplates eine sehr eingeschränkte Form haben. Zunächst identifiziert man alle Anteile des Grammatiktemplates, die nicht expandiert werden müssen (z.B. einfache Nichtterminalsymbole und Typbezeichner). An den Alternativenregeltemplates lässt sich schon vor der Expansion ein Teil des Teilsprachendiagramms ablesen. Tritt auf der rechten Seite eines Alternativenregeltemplates ein Nichtterminalsymbol auf, so liegt die von ihm erzeugte Teilsprache unabhängig von der späteren Expansion in derjenigen Teilsprache, die von dem Nichtterminalsymbol auf der linken Seite des Definitionsregeltemplates stammt. Daraus erstellt man ein erstes *partielles Teilsprachendiagramm* $\mathcal{D}_0$. Die einfachen Nichtterminalsymbole und Typbezeichner in $\widetilde{N}_{TP}$ und $\widetilde{T}_{TP}$ (die also kein Templateargument besitzen) sind immer in den Mengen der expandierten Symbole N_{TP} und T_{TP} enthalten. Nichtparametrisierte Produktionstemplates P_0 müssen ebenfalls nicht expandiert werden. Insgesamt erhält man folgende nichtexpandierbaren Anteile einer Templategrammatik:

$$\begin{aligned} N_0 = U \cap \widetilde{N}_{TP} &\subset N_{TP} \\ T_0 = \alpha(U) \cap \widetilde{T}_{TP} &\subset T_{TP} \\ P_0 &\subset P_{TP} \end{aligned}$$

Nun werden die Symbole aus N_0 und T_0 dort, wo es möglich ist, in die Argumente der Templatesymbole eingesetzt und die Produktionstemplates mit dieser ersten Generation zusammengesetzter Symbole expandiert. Das Ergebnis dieser Ersetzungen bildet zusammen mit den nicht-zusammengesetzten Symbolen und Produktionen die nächste Generation von

Mengen N_1, T_1 und P_1. Das partielle Teilsprachendiagramm $\mathcal{D}_0$ wird zu $\mathcal{D}_1$ ergänzt, indem die neugewonnenen Produktionen P_1 ausgewertet werden. Dieses Verfahren wird fortgesetzt, bis sich die erzeugten Mengen nicht mehr ändern. Wenn unendlich tiefe Schachtelungen von Templates möglich sind wie in Beispiel 4.7, terminiert das Verfahren nicht. Aus diesem Grund bricht man bei der praktischen Umsetzung der Expansion das Verfahren bei einer hinreichend großen Schachtelungstiefe ab.

Hilfsdefinitionen. Um das gerade skizzierte Expansionsverfahren aufschreiben zu können, braucht man eine feinere Definition der Expansionsfunktion **subst** aus (W_{11}''), (W_{12}'') und (W_{13}''). Für eine Menge T von Templates und eine Menge von Symbolen M bezeichne $\mathbf{subst}(T, M)$ alle gültigen Ersetzungen von Parametern aus M in den Templates in T. Templates sind dabei Nichtterminalsymboltemplates, Typbezeichnertemplates oder Produktionstemplates. Symbole sind einfache und zusammengesetzte Nichtterminalsymbole und Typbezeichner. Für die Templatemengen in Beispiel 4.7 besteht

$$\mathbf{subst}(\{\, \mathit{Vektor}\langle X\rangle\}, \{\mathit{Int}, \mathbf{Int}\}) = \{\, \mathit{Vektor}\langle \mathbf{Int}\rangle\}$$

nur aus einem Element, weil für X nur ein Typbezeichner eingesetzt werden kann. Entscheidend ist, dass die Menge M nur Symbole ohne freie Templateparameter enthalten darf:

$$\mathbf{subst}(\{\, \mathit{Vektor}\langle X\rangle\}, \{\mathbf{Int}, \mathbf{Vektor}\,\langle \mathbf{Int}\rangle\}) = \{\, \mathit{Vektor}\langle \mathbf{Int}\rangle\,,\, \mathit{Vektor}\langle \mathbf{Vektor}\,\langle \mathbf{Int}\rangle\rangle\},$$

damit man bei wiederholter Anwendung von $\mathbf{subst}(T, \cdot)$ immer vollständig substituierte Symbole oder Produktionen erhält. Wenn man in einem Produktionstemplate einen Parameter ersetzt, wird das Produktionstemplate expandiert wie z.B. dieses Alternativenregeltemplate:

$$\begin{aligned}&\mathbf{subst}(\{\, \mathit{Objekt} \longrightarrow \mathit{Int} \mid \mathit{Tonhöhe} \mid \mathit{Tonhöhenvektor}\langle X \preceq \mathbf{Tonhöhe}\rangle\}, \{\mathbf{THMidi}\})\\ &= \{\, \mathit{Objekt} \longrightarrow \mathit{Int} \mid \mathit{Tonhöhe} \mid \mathit{Tonhöhenvektor}\langle \mathbf{THMidi}\rangle\}\end{aligned}$$

An einer expandierten Alternativenregel kann man Teilsprachenrelationen ablesen, hier etwa

$$\begin{aligned} L(\mathit{Int}) &\subset L(\mathit{Objekt})\\ L(\mathit{Tonhöhe}) &\subset L(\mathit{Objekt})\\ L(\mathit{Tonhöhenvektor}\langle \mathbf{THMidi}\rangle) &\subset L(\mathit{Objekt}) \end{aligned}$$

Bei der schrittweisen Expansion der Alternativenregeltemplates werden die Teilsprachenrelationen, die den neuen Alternativenregeln entsprechen, zum partiellen Teilsprachendiagramm $\mathcal{D}_i$ der i-ten Generation hinzugefügt.

Algorithmus zur Erzeugung einer Templategrammatik. Mit diesen Bezeichnungen kann man ein iteratives Verfahren zur Expansion der Templates in einer Templategrammatik angeben:

Gegeben seien folgende Mengen:

$N_0 = U \cap \widetilde{N}_{TP}$	alle einfachen Nichtterminalsymbole.
$T_0 = \alpha(U) \cap \widetilde{T}_{TP}$	alle einfachen Typbezeichner.
$P_0 \subset \widetilde{P}_{TP}$	alle nichtparametrisierten Produktionen (ohne Templateparameter links oder rechts).
$\mathcal{D}_0$	Alle Teilsprachenrelationen zwischen Elementen von N_0.

Für $i \in \mathbb{N}$ iteriere man unter Berücksichtigung des partiellen Teilsprachendiagramms $\mathcal{D}_{i-1}$:

$$\begin{aligned} N_i &= \mathbf{subst}(\widetilde{N}_{TP}, N_{i-1} \cup T_{i-1}) \\ T_i &= \mathbf{subst}(\widetilde{T}_{TP}, N_{i-1} \cup T_{i-1}) \\ P_i &= \mathbf{subst}(\widetilde{P}_{TP}, N_{i-1} \cup T_{i-1}) \\ \mathcal{D}_i &= \text{Teilsprachenrelationen in } P_i \end{aligned}$$

bis $N_i = N_{i-1}$, $T_i = T_{i-1}$, $P_i = P_{i-1}$.

Die expandierten Mengen sind dann:

$$\begin{aligned} N_{TP} &= \lim_{i \to \infty} N_i \\ T_{TP} &= \lim_{i \to \infty} T_i \\ P_{TP} &= \lim_{i \to \infty} P_i = \mathbf{subst}(\widetilde{P}_{TP}, N_{TP} \cup T_{TP}) \\ \mathcal{D}_L &= \lim_{i \to \infty} \mathcal{D}_i = \text{Teilsprachenrelationen in } P_{TP} \end{aligned}$$

Das Verfahren findet alle zulässigen Instanzen der Templates, weil Expansion per Definition darauf beruht, dass ein bereits expandiertes Symbol für einen Templateparameter eingesetzt wird. Der Algorithmus ist lediglich eine konstruktive Umsetzung dieses Prinzips.

Sobald unendlich häufig schachtelbare Templates wie z.B. *Vektor*$\langle X \rangle$ in Beispiel 4.7 vorhanden sind, terminiert das Verfahren nicht mehr. Man kann das Verfahren aber nach einer festen Anzahl von Iterationen $n \in \mathbb{N}$ abbrechen und erhält dann eine endliche Grammatik $G_n = (N_n, T_n, P_n, S)$ für ein Startsymbol S, das für jeden freien Parameter alle zulässigen Ersetzungen bis zur Schachtelungstiefe $\leq n$ enthält. Wenn ein Typparameter in einem Template bereits in einem geschachtelten Ausdruck wie *Vektor*$\langle$**Vektor** $\langle X \rangle\rangle$ enthalten ist, kann die Grammatik G_n auch tiefer geschachtelte Ausdrücke enthalten.

Da der Algorithmus in erster Linie demonstrieren soll, dass man die expandierten Mengen trotz der oben beschriebenen Abhängigkeiten generieren kann, wurde bei der Formulierung die Verständlichkeit und nicht die Effizienz in den Vordergrund gestellt. Wenn man den Algorithmus implementieren wollte, wäre es sinnvoll, unnötige Ersetzungsversuche von vorneherein auszuschließen. Beispielsweise enthält die Menge $\widetilde{N}_{TP}$ auch einfache Nichtterminalsymbole, die in einem Expansionsschritt niemals zu einem neuen Symbol führen. Außerdem gelten $N_{i-1} \subseteq \mathbf{subst}(\widetilde{N}_{TP}, N_{i-1} \cup T_{i-1})$ und $T_{i-1} \subseteq \mathbf{subst}(\widetilde{T}_{TP}, N_{i-1} \cup T_{i-1})$, d.h. bei jedem Iterationsschritt wird das Ergebnis des vorigen Schrittes neu konstruiert. Effizienter wäre, hier nur die neu hinzukommenden Symbole zu ergänzen.

4.2.5 Erweiterbare Zeitreihengrammatik

Eine *Erweiterbare Zeitreihengrammatik* ist eine für Zeitreihenprobleme maßgeschneiderte generative Grammatik. Sie kombiniert die Eigenschaften von Typ- und Templategrammatiken. Um einen allgemeinen, aber problemspezifisch erweiterbaren Rahmen für die Zeitreihenmodellierung vorzugeben, wird die Erweiterbare Zeitreihengrammatik in einen allgemeinen und einen benutzerdefinierten Teil aufgespalten. Der allgemeine Teil ist fest vorgegeben und legt zeitreihenspezische Aspekte der Grammatik fest. Im benutzerdefinierten Teil definiert der Benutzer abhängig von der Fragestellung problemspezifische Strukturen. In Abschnitt 4.2.6 wird ein musikalisches Beispiel für einen benutzerdefinierten Teil der Erweiterbaren Zeitreihengrammatik vorgestellt, das als Grundlage für die späteren Kapitel dient.

Definition 4.10 (Erweiterbare Zeitreihengrammatik)
Gegeben seien *Alphabete* Σ_{Att} wie in Definition 4.6 und Σ_{NTS}, Σ_{TB}, Σ_I, Σ_W, und Σ_{Par} wie in Definition 4.8, aus denen sich das *Gesamtalphabet* $\Sigma = \Sigma_{NTS} \cup \Sigma_{TB} \cup \Sigma_I \cup \Sigma_W \cup \Sigma_{Att} \cup \Sigma_{Par}$ zusammensetzt. Die Expansionsfunktion **subst** und die Abbildungen α und $\widetilde{\alpha}$ seien mit den unten angegebenen Stammsymbolen $U = U_A \cup U_B$ wie in Definition 4.8 definiert.

Eine *Erweiterbare Zeitreihengrammatik* ist ein Tupel

$$G = (N_{TP},\ T_{TP} \cup \Sigma_I \cup \Sigma_W \cup \Sigma_{Att} \cup \Sigma_{Par},\ P_{TP},\ \mathit{Objekt}),$$

das vom Grammatiktemplate

$$(\widetilde{N}_{TPA} \cup \widetilde{N}_{TPB},\ \widetilde{T}_{TPA} \cup \widetilde{T}_{TPB},\ \widetilde{P}_{TPA} \cup \widetilde{P}_{TPB})$$

generiert wird. Dabei müssen die Mengen U_B, $\widetilde{N}_{TPB}$, $\widetilde{T}_{TPB}$ und $\widetilde{P}_{TPB}$ die Wohlgeformtheitsregeln $(W_1{}''')$–$(W_{15}{}''')$ in Tabelle 4.4 erfüllen. Die Mengen U_A, $\widetilde{N}_{TPA}$, $\widetilde{T}_{TPA}$ und $\widetilde{P}_{TPA}$ enthalten zeitreihenspezifische Symbole und Produktionstemplates und sind fest vorgegeben:

- U_A enthält die *allgemeinen Stammsymbole*:

$$\begin{aligned} U_A \quad = \quad & \{\mathbf{Objekt}, \mathbf{Vektor}, \mathbf{Ansicht}, \mathbf{Segment}, \\ & \mathbf{Int}, \mathbf{Bool}, \mathbf{Rational}, \mathbf{String}, \mathbf{BenutzerdefiniertesObjekt}\} \end{aligned}$$

- Die *allgemeinen Nichtterminalsymboltemplates* $\widetilde{N}_{TPA}$ sind gegeben durch:

$$\widetilde{N}_{TPA} = \{\mathit{Objekt}, \mathit{Vektor}\langle X\rangle, \mathit{Ansicht}\langle X\rangle, \mathit{Segment}\langle X\rangle, \mathit{Int}, \mathit{Bool}, \mathit{Rational}, \mathit{String}, \mathit{BenutzerdefiniertesObjekt}\}$$

- Die *allgemeinen Typbezeichnertemplates* $\widetilde{T}_{TPA}$ ergeben sich aus den Nichtterminalsymboltemplates:

$$\begin{aligned}\widetilde{T}_{TPA} &= \widetilde{\alpha}(\widetilde{N}_{TPA}) \\ &= \{\mathbf{Objekt}, \mathbf{Vektor}\langle X\rangle, \mathbf{Ansicht}\langle X\rangle, \mathbf{Segment}\langle X\rangle, \\ &\quad \mathbf{Int}, \mathbf{Bool}, \mathbf{Rational}, \mathbf{String}, \mathbf{BenutzerdefiniertesObjekt}\}\end{aligned}$$

- Die *allgemeinen Produktionstemplates* $\widetilde{P}_{TPA}$ sind gegeben durch:

$\widetilde{P}_{TPA} = \{$

// Generischer Teil

(P_1) $\mathit{Objekt} \longrightarrow \mathit{Vektor}\langle X\rangle \mid \mathit{Ansicht}\langle X\rangle \mid \mathit{Segment}\langle X\rangle \mid \mathit{Int} \mid \mathit{Bool} \mid \mathit{Rational} \mid \mathit{String} \mid \mathit{BenutzerdefiniertesObjekt}$

(P_2) $\mathit{Vektor}\langle X\rangle \longrightarrow (\mathbf{Vektor}\langle X\rangle;\ [\widetilde{\alpha}^{-1}(X)[,\ \widetilde{\alpha}^{-1}(X)]^*])$

// Zeitreihenspezifischer Teil

(P_3) $\mathit{Ansicht}\langle X\rangle \longrightarrow (\mathbf{Ansicht};\ [\mathit{Segment}\langle X\rangle[,\ \mathit{Segment}\langle X\rangle]^*])$

(P_4) $\mathit{Segment}\langle X\rangle \longrightarrow (\mathbf{Segment}\langle X\rangle;\ t : \mathit{Int},\ d : \mathit{Int},\ \widetilde{\alpha}^{-1}(X);\ t \in \mathbb{T}, d \in \mathbb{T}^+)$

// Basisdatentypen

(P_5) $\mathit{Int} \longrightarrow (\mathbf{Int};\ \mathbb{Z})$

(P_6) $\mathit{Bool} \longrightarrow (\mathbf{Bool};\ \text{wahr} \mid \text{falsch})$

(P_7) $\mathit{Rational} \longrightarrow (\mathbf{Rational};\ \mathbb{Q})$

(P_8) $\mathit{String} \longrightarrow (\mathbf{String};\ \Sigma^*_{Str})$

$\}$

Die Mengen U_B, $\widetilde{N}_{TPB}$, $\widetilde{T}_{TPB}$ und $\widetilde{P}_{TPB}$ werden durch den Benutzer definiert und dienen dazu, problemspezifische Symbole und Produktionstemplates in eine Erweiterbare Zeitreihengrammatik zu integrieren:

- Die endliche Menge $U_B \subset \{w \in {\Sigma_{TB}}^* \mid |w| \geq 2\} \backslash U_A$ enthält *benutzerdefinierte Stammsymbole* der Länge mindestens 2, die nicht bereits als allgemeine Stammsymbole verwendet werden.
- Die endliche Menge $\widetilde{N}_{TPB}$ der *benutzerdefinierten Nichtterminalsymboltemplates* enthält Kombinationen von Stammsymbolen aus U_B und gegebenenfalls von Templateparametern, die auf der linken Seite der Produktionstemplates aus $\widetilde{P}_{TPB}$ auftreten. Die Menge $\widetilde{N}_{TPB}$ wird durch die Wohlgeformtheitsregel $(W_6{}''')$ charakterisiert.

- Die *benutzerdefinierten Typbezeichnertemplates* $\widetilde{T}_{TPB} = \widetilde{\alpha}(\widetilde{N}_{TPB})$ ergeben sich durch Anwendung der erweiterten Alphabetwechselabbildung auf die benutzerdefinierten Nichtterminalsymboltemplates $\widetilde{N}_{TPB}$ (W_7''').
- Die Gestalt der *benutzerdefinierten Produktionstemplates* $\widetilde{P}_{TPB}$ wird durch die Wohlgeformtheitsregeln (W_1''')–(W_5''') festgelegt.

Daraus werden die Mengen der Erweiterbaren Zeitreihengrammatik durch Expansion (wie in Definition 4.8) konstruiert:

- Nichtterminalsymbole: $N_{TP} = \mathbf{subst}(\widetilde{N}_{TPA} \cup \widetilde{N}_{TPB})$
- Terminalsymbole: $T_{TP} = \widetilde{\alpha}(N_{TP}) \ \cup \ \Sigma_I \cup \Sigma_W \cup \Sigma_{Att} \cup \Sigma_{Par}$
- Produktionen: $P_{TP} = \mathbf{subst}(\widetilde{P}_{TPA} \cup \widetilde{P}_{TPB})$
- Startsymbol: $\mathit{Objekt} \in \widetilde{N}_{TPA} \cap U_A \subset N_{TP}$

Wie bei einer Typgrammatik ist der *Typ* $\mathrm{Typ}(X)$ von G definiert als die Menge der Wörter der Sprache $L(G)$, die vom Nichtterminalsymbol $X \in N$ abgeleitet werden können und die semantischen Regeln (S_1''') und (S_2''') aus Tabelle 4.4 erfüllen:

$$\mathrm{Typ}(X) = \{w \in L(X) \mid w \text{ erfüllt } (S_1''') \text{ und } (S_2''') \ \}.$$

* * *

Allgemeine und benutzerdefinierte Grundmengen. Die Definition einer Erweiterbaren Zeitreihengrammatik baut auf den Definitionen von Typ- und Templategrammatiken auf und spaltet darüberhinaus die Mengen einer Grammatik in allgemeine und benutzerdefinierte Anteile. Aufgeteilt werden die Stammsymbole $U = U_A \cup U_B$, die Nichtterminalsymboltemplates $\widetilde{N}_{TPA}$ und $\widetilde{N}_{TPB}$ und die Produktionstemplates $\widetilde{P}_{TPA}$ und $\widetilde{P}_{TPB}$, die in Tabelle 4.3 im Vergleich mit den Grundmengen der anderen Grammatiken dargestellt sind. Der allgemeine Teil einer Erweiterbaren Zeitreihengrammatik legt Zeitinformation wie z.B. Segmente und allgemeine Objekte wie z.B. Vektoren fest und ist in der Definition vorgegeben. Im benutzerdefinierten Teil spezifiziert der Anwender problembezogene Mengen. Damit die benutzerdefinierten Mengen sich in den allgemeinen Teil der Grammatik integrieren, wird ihre Gestalt wie bisher durch Wohlgeformtheitsregeln vorgegeben. Der benutzerdefinierte Teil der Erweiterbaren Zeitreihengrammatik wird an den allgemeinen Teil durch ein spezielles Nichtterminalsymbol *BenutzerdefiniertesObjekt* angebunden, das als linke Seite einer benutzerdefinierten Produktion auftreten muss (W_{14}''').

Konstruktion einer Erweiterbaren Zeitreihengrammatik. Wie bei einer Templategrammatik wird auch eine Erweiterbare Zeitreihengrammatik nicht direkt durch Grundmengen, sondern indirekt durch ein Grammatiktemplate definiert, aus dem durch Expansion eine Grammatik generiert wird. Daraus werden die Typbezeichnertemplates durch Anwendung der Alphabetwechsel-Abbildung $\widetilde{\alpha}$ auf die Nichtterminalsymboltemplates gewonnen. Die expandierten Mengen erzeugt man mit Hilfe der Expansionsabbildung **subst** (Abbildung 4.12).

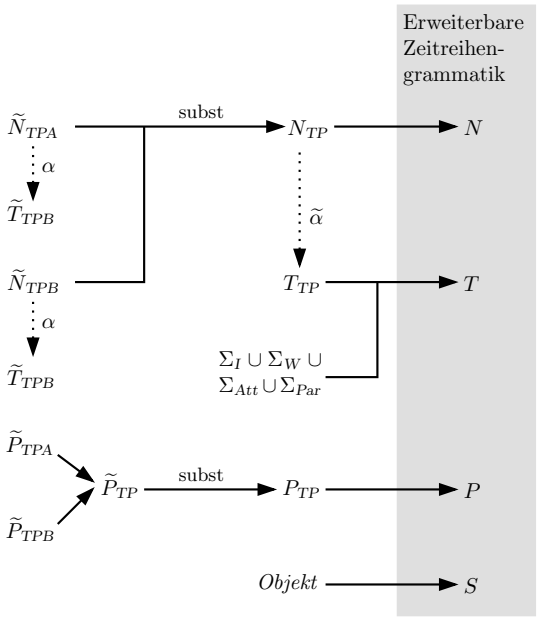


Abbildung 4.12: Konstruktion einer Erweiterbaren Zeitreihengrammatik

Die allgemeinen und benutzerdefinierten Templatemengen werden vor der Expansion vereinigt. Dies hat den Vorteil, dass eine Erweiterbare Zeitreihengrammatik auch Objekte wie Vektoren aus problemspezifischen Objekten wie z.B. (**Vektor** ⟨**Takt**⟩) erzeugen kann, die allgemeine und benutzerdefinierte Bestandteile kombinieren. Damit man im Nachhinein die Bedeutung der allgemeinen Nichtterminalsymbole nicht ändern kann, dürfen sich die allgemeinen und die benutzerdefinierten Nichtterminalsymboltemplates nicht überschneiden. Dies ist sichergestellt, wenn die Mengen der allgemeinen und der benutzerdefinierten Stammsymbole disjunkt sind ($U_A \cap U_B = \emptyset$), denn nach (W_6''') beginnt jedes Nichtterminalsymboltemplate mit einem Stammsymbol.

Erzeugung von Objekten mit der Erweiterbaren Zeitreihengrammatik. Wenn man Objekte mit einer Erweiterbaren Zeitreihengrammatik erzeugen will, geht man folgendermaßen vor: Als erstes wählt man die benutzerdefinierten Mengen und vereinigt diese mit den allgemeinen Mengen. Das entstandene Grammatiktemplate wird mit dem Algorithmus aus Abschnitt 4.2.4.2 expandiert, wobei man bei der praktischen Umsetzung nach einer endlichen Anzahl von Expansionsschritten abbricht. Durch Anwendung der expandierten Produktionen auf das Startsymbol *Objekt* werden Objekte erzeugt, deren semantische Gültigkeit schließlich wie in der Typgrammatik durch Prüfung eventuell vorhandener Bedingungen ermittelt wird.

Es gibt also zwei Arten von Bedingungen in einer Erweiterbaren Zeitreihengrammatik, die bei der Erzeugung eines Objekts zu unterschiedlichen Zeitpunkten überprüft werden. Erstens treten Typbedingungen in Templateparametern auf, die beim Expandieren der Templates

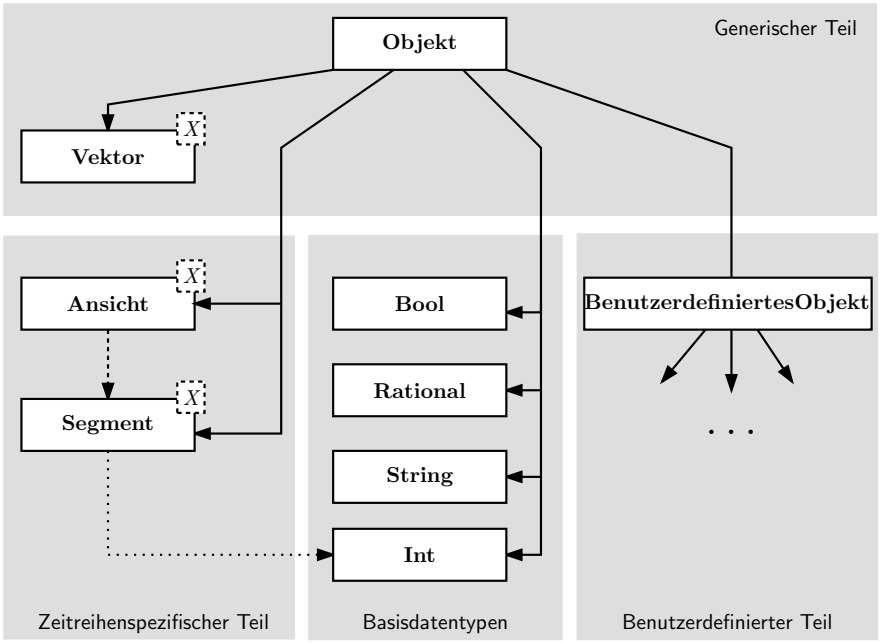


Abbildung 4.13: Typdiagramm für den allgemeinen Teil einer Erweiterbaren Zeitreihengrammatik. Ein durchgezogener Pfeil zwischen Nichtterminalsymbolen $X \longrightarrow Y$ heißt, dass die zugehörigen Typen ineinander enthalten sind: $\text{Typ}(Y) \subset \text{Typ}(X)$. Ein gestrichelter Pfeil von X nach Y bedeutet, dass die Definition von X das Nichtterminalsymbol Y verwendet. Ein Typ mit freiem Parameter X steht für alle Typen, die durch Expansion aus dem Typbezeichnertemplate gebildet werden können.

ausgewertet werden. Zweitens gibt es semantische Bedingungen in Definitionen, die erst nach dem Erzeugen eines Objekts ausgewertet werden.

Typdiagramm für den allgemeinen Teil der Erweiterbaren Zeitreihengrammatik. Der allgemeine Teil einer Erweiterbaren Zeitreihengrammatik besteht aus generischen, zeitreihenspezifischen und benutzerdefinierten Bestandteilen sowie Basisdatentypen. Das Typdiagramm in Abbildung 4.13, das die Abhängigkeiten zwischen den Nichtterminalsymboltemplates des allgemeinen Teils einer Erweiterbaren Zeitreihengrammatik zeigt, spiegelt diese Aufteilung wider. Ein durchgezogener Pfeil zwischen Nichtterminalsymbolen $X \longrightarrow Y$ heißt, dass die zugehörigen Typen ineinander enthalten sind, d.h. $\text{Typ}(Y) \subset \text{Typ}(X)$. Ein gestrichelter Pfeil von X nach Y bedeutet, dass die Definition von X das Nichtterminalsymbol Y verwendet.

Der generische Teil enthält das Startsymbol *Objekt* und das Nichtterminalsymboltemplate *Vektor*$\langle X \rangle$, das Objekte vom Typ X gruppiert. Wenn man Vektoren schachtelt, kann man Baumstrukturen erzeugen, wie sie z.B. bei Reduktionsanalysen nach Lerdahl und Jackendoff [51] benötigt werden.

Der zeitreihenspezifische Teil definiert die Nichtterminalsymboltemplates $\mathit{Segment}\langle X\rangle$ und $\mathit{Ansicht}\langle X\rangle$, so dass die Ansichten einer Komposition mit vorgegebenen Typen parametrisiert werden können. Ein Segment enthält neben Einsatzzeit und Dauer noch die eigentliche problemspezifische Information in Form eines Objekts vom Typ X. Die Dauer und die Einsatzzeit sind Elemente von $\mathbb{T} = \mathbb{T}_r$ und $\mathbb{T} = \mathbb{T}_r^+$, wobei $r \in \mathbb{Q}$ eine feste Auflösung ist, die nicht verändert und daher im folgenden weggelassen wird (vgl. Definition 4.1).

Wohlgeformtheitsregeln. Die Wohlgeformtheitsregeln für die Erweiterbare Zeitreihengrammatik fassen die entsprechenden Wohlgeformtheitsregeln der Typ- und der Templategrammatik zusammen, wobei ihre Gültigkeit nur für den benutzerdefinierten Teil der Erweiterbaren Zeitreihengrammatik überprüft werden muss. Für den allgemeinen, fest vorgegebenen Teil der Erweiterbaren Zeitreihengrammatik sind die Wohlgeformtheitsregeln per Definition erfüllt.

Die Regeln (W_4'''), (W_{10}'''), (S_1''') und (S_2'''), die die Verkleinerung von Teilsprachen mit Hilfe von Bedingungen betreffen, werden unverändert aus der Typgrammatik übernommen. Aus der Templategrammatik stammen die Wohlgeformtheitsregeln (W_6''') und (W_7''') über (Nicht-)Terminalsymboltemplates sowie (W_{11}'''), (W_{12}''') und (W_{13}''') über die Expansion von Produktionstemplates. Ebenfalls aus der Templategrammatik kommen die Wohlgeformtheitsregeln über die Struktur der Produktionstemplates (W_1'''), (W_2'''), (W_8'''), (W_{14}''') und (W_{15}'''). Die restlichen Regeln (W_3'''), (W_5''') und (W_9''') kombinieren Eigenschaften der entsprechenden Regeln aus Typ- und Templategrammatik.

4.2.6 Beispiel für eine Erweiterbare Zeitreihengrammatik

Nun folgt ein musikalisches Beispiel für einen benutzerdefinierten Teil einer Erweiterbaren Zeitreihengrammatik, mit dem sich auch die in den weiteren Kapiteln diskutierten Beispiele erzeugen lassen.

Beispiel 4.11 (Erweiterbare Zeitreihengrammatik)
Um eine Erweiterbare Zeitreihengrammatik anzugeben, muss man die benutzerdefinierten Mengen in Definition 4.10 festlegen. Zu definieren sind die Stammsymbole U_B, die benutzerdefinierten Nichtterminalsymboltemplates $\widetilde{N}_{TPB}$ und die benutzerdefinierten Produktionstemplates $\widetilde{P}_{TPB}$. Als Menge der benutzerdefinierten Stammsymbole U_B wird gewählt:

$$
\begin{aligned}
U_B \quad = \{ \quad & \mathit{Tonhöhe}, \mathit{THMidi}, \mathit{THChromOktaväquiv}, \mathit{THEnhÄquiv}, \mathit{THPentatonischEs}, \\
& \mathit{Intervall}, \mathit{IntervallNumerisch}, \mathit{IntervallSymbolisch}, \mathit{Kontur}, \mathit{Tonfolge}, \\
& \mathit{Motiv}, \mathit{Motivklasse}, \mathit{Phrase}, \mathit{Phrasenform}, \mathit{Position}, \mathit{Harmonie}, \\
& \mathit{HarmonischesSymbol}, \mathit{HarmonischeFunktion}, \mathit{HarmonischeUmkehrung}, \\
& \mathit{HarmonischeDissonanz}, \mathit{TonalesZentrum}, \mathit{Takt}, \mathit{Oktave}, \mathit{Ambitus}, \\
& \mathit{Häufigkeit}, \mathit{Bigramm} \}
\end{aligned}
$$

Hier ist wichtig, dass die neuen Stammsymbole sich nicht mit den bereits existierenden Stammsymbolen U_A aus Definition 4.10 überschneiden. Nun werden an einen Teil der

Stammsymbole Templateparameter angefügt. Jedes Stammsymbol tritt wegen (W_8''') genau einmal als linker Teil eines Nichtterminalsymboltemplates in $\widetilde{N}_{TPB}$ auf:

$$\begin{aligned}\widetilde{N}_{TPB} = \{\ & \textit{Tonhöhe}, \textit{THMidi}, \textit{THChromOktaväquiv}, \textit{THEnhÄquiv}, \textit{THPentatonischEs} \\ & \textit{Intervall}, \textit{IntervallNumerisch}, \textit{IntervallSymbolisch}, \textit{Kontur}, \textit{Tonfolge}, \\ & \textit{Motiv}, \textit{Motivklasse}, \textit{Phrase}, \textit{Phrasenform}, \textit{Position}, \textit{Harmonie}, \\ & \textit{HarmonischesSymbol}, \textit{HarmonischeFunktion}, \textit{HarmonischeUmkehrung}, \\ & \textit{Tonart}, \textit{Takt}, \textit{Tonhöhenvektor}\,\langle X \preceq \textbf{Tonhöhe}\rangle, \\ & \textit{HarmonischeDissonanz}, \textit{Oktave}, \textit{Ambitus}, \textit{Häufigkeit}\,\langle X\rangle, \textit{Bigramm}\,\langle X\rangle\ \}\end{aligned}$$

Es sind verschiedene Definitionen für Tonhöhen vorgesehen. *Tonhöhe* wird später als abstraktes Nichtterminalsymbol verwendet. *THMidi* soll Tonhöhen mit Hilfe von MIDI-Nummern darstellen. *THChromOktaväquiv* codiert Tonhöhen in 12 Stufen unabhängig von ihrer Oktavzugehörigkeit. *THEnhÄquiv* berücksichtigt die enharmonische Äquivalenz von Tönen, d.h. *fis* und *ges* werden nicht unterschieden. *THPentatonischEs* realisiert eine pentatonische Skala mit Grundton *es*. Das Nichtterminalsymboltemplate *Tonhöhenvektor* $\langle X \preceq \textbf{Tonhöhe}\rangle$ wird mit einer dieser Tonhöhen instanziiert, was durch die Bedingung $X \preceq \textbf{Tonhöhe}$ im Templateargument ausgedrückt wird. Der Parameter X soll also **Tonhöhe** spezialisieren. Das Nichtterminalsymboltemplate *Häufigkeit* $\langle X\rangle$ speichert einen Vektor von Objekten vom Typ X sowie deren Häufigkeiten. Ein *Bigramm* vom Typ X ist eine Folge von zwei Elementen vom Typ X. All diese Nichtterminalsymboltemplates treten wegen (W_{14}''') genau einmal als linker Teil eines Produktionstemplates in $\widetilde{P}_{TPB}$ auf:

$\widetilde{P}_{TPB} = \{$

// Verknüpfung mit allgemeiner Grammatik

(P_9) *BenutzerdefiniertesObjekt* $\longrightarrow$ *Tonhöhe* | *Intervall* | *Kontur* | *Tonfolge* | *Motiv* | *Motivklasse* | *Phrase* | *Phrasenform* | *Position* | *Harmonie* | *HarmonischesSymbol* | *HarmonischeFunktion* | *HarmonischeUmkehrung* | *HarmonischeDissonanz* *Tonart* | *Takt* | *Tonhöhenvektor* $\langle X \preceq \textbf{Tonhöhe}\rangle$ | *Oktave* | *Ambitus* | *Häufigkeit* $\langle X\rangle$ | *Bigramm* $\langle X\rangle$

// Hierarchische Definition von Tonhöhen und Intervallen

(P_{10}) *Tonhöhe* $\longrightarrow$ *THMidi* | *THChromOktaväquiv* | *THEnhÄquiv* | *THPentatonischEs*

(P_{11}) *THMidi* $\longrightarrow$ (**THMidi**; $i : \textit{Int}$; $0 \leq i \leq 127$)

(P_{12}) *THChromOktaväquiv* $\longrightarrow$ (**THChromOktaväquiv**; $s : \textit{String}$; $s \in$ {c,cis,des,d,dis,es, e,f,fis,ges,g,gis,as,a,ais,b,h})

(P_{13}) *THEnhÄquiv* $\longrightarrow$ (**THEnhÄquiv**; $s : \textit{String}, \textit{Oktave}$; $s \in$ {c,cis,d,es,e,f,fis,g,as,a,b,h}

(P_{14}) *THPentatonischEs* $\longrightarrow$ (**THPentatonischEs**; $s : \textit{String}$; $s \in$ {es,f,g,b,c})

(P_{15}) *Intervall* $\longrightarrow$ *IntervallNumerisch* | *IntervallSymbolisch*

(P_{16}) *IntervallNumerisch* $\longrightarrow$ (**IntervallNumerisch**; $i : Int;\ 0 \leq i \leq 11$)

(P_{17}) *IntervallSymbolisch* $\longrightarrow$ (**IntervallSymbolisch**; $s : String,\ i : Int;$
$(s \in \{\text{g,k}\} \wedge i \in \{2,3,6,7\}) \vee (s \in \{\text{r,v,ü}\} \wedge i \in \{1,4,5,8\})$)

// Definitionen für die Musikbeispiele

(P_{18}) *Kontur* $\longrightarrow$ (**Kontur**; $i : Int;\ -1 \leq i \leq 1$)

(P_{19}) *Tonfolge* $\longrightarrow$ (**Tonfolge**; $Vektor\langle$**THMidi**$\rangle$)

(P_{20}) *Motiv* $\longrightarrow$ (**Motiv**)

(P_{21}) *Motivklasse* $\longrightarrow$ (**Motivklasse**; $s : String;\ s \in \{\text{A}, \ldots, \text{Z}\}$)

(P_{22}) *Phrase* $\longrightarrow$ (**Phrase**)

(P_{23}) *Phrasenform* $\longrightarrow$ (**Phrasenform**; $s : String;\ s \in \{$steigend, fallend,

(P_{24}) *Position* $\longrightarrow$ (**Position**; $s : String;\ s \in \{$Anfang, Mitte, Ende$\}$)

(P_{25}) *Harmonie* $\longrightarrow$ (**Harmonie**; $s : String;$
$s \in \{\text{C,G,D,A,E,H,Fis,Des,As,Es,B,F}\}$)

(P_{26}) *HarmonischesSymbol* $\longrightarrow$ (**HarmonischesSymbol**; *HarmonischeFunktion*,
HarmonischeUmkehrung, *HarmonischeDissonanz*)

(P_{27}) *HarmonischeFunktion* $\longrightarrow$ (**HarmonischeFunktion**;
$s : String;\ s \in \{\text{T,D,S,DD,SS,Tp,VTp}, \ldots\}$)

(P_{28}) *HarmonischeUmkehrung* $\longrightarrow$ (**HarmonischeUmkehrung**; $s : String;$
$s \in \{$Grundton, Terz, Quinte, harmonieergänzend, harmoniefremd$\}$)

(P_{29}) *HarmonischeDissonanz* $\longrightarrow$ (**HarmonischeDissonanz**; $s : String;$
$s \in \{$Sekunde,Quarte,Sexte, Septime, keiner$\}$)

(P_{30}) *Tonart* $\longrightarrow$ (**Tonart**; $s : String;\ s \in \{\text{C,Cis,D,Dis,E}, \ldots\}$)

(P_{31}) *Oktave* $\longrightarrow$ (**Oktave**; $i : Int;\ -3 \leq i \leq 3$)

(P_{32}) *Ambitus* $\longrightarrow$ (**Ambitus**; *Tonhöhe*, *Tonhöhe*)

(P_{33}) *Takt* $\longrightarrow$ (**Takt**; $z : Int, n : Int;\ (1 \leq z \leq 7) \wedge (\exists k \in \mathbb{N} \text{ mit } 2^k = n)$)

// Benutzerdefinierte Templates

(P_{34}) *Häufigkeit* $\langle X \rangle$ $\longrightarrow$ (**Häufigkeit** $\langle X \rangle$;
$v : Vektor\langle X \rangle,\ w : Vektor\langle$**Rational**$\rangle$;
$|v| = |w| \wedge \forall 1 \leq i \leq |w| :\ 0 \leq w_i \leq 1 \wedge \sum_{i=1}^{|w|} w_i = 1$)

(P_{35}) *Tonhöhenvektor* $\langle X \preceq$ **Tonhöhe**$\rangle$ $\longrightarrow$ (**Tonhöhenvektor** $\langle X \rangle$; $[\widetilde{\alpha}^{-1}(X)[,\ \widetilde{\alpha}^{-1}(X)]^*]$)

(P_{36}) *Bigramm* $\langle X \rangle$ $\longrightarrow$ (**Bigramm** $\langle X \rangle$; $v : Vektor\langle X \rangle$; $|v| = 2$)

}

* * *

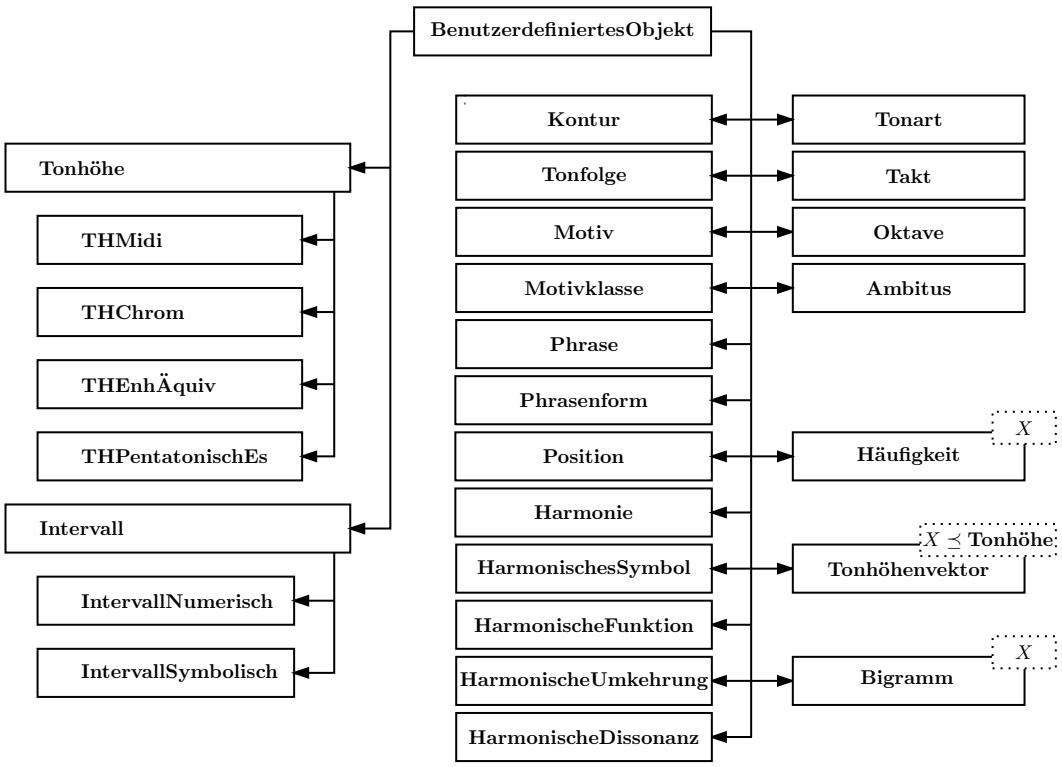


Abbildung 4.14: Untertypen für die benutzerdefinierten Typen in Beispiel 4.2.6. Die Pfeile zeigen an, wie die Typen durch Alternativenregeln zusammenhängen.

Produktionstemplates. Die Produktionstemplates definieren eine Reihe musikalischer Strukturen. Es gibt beispielsweise verschiedene Definitionen für Tonhöhen, die Alterationen und Oktavlage eines Tons in unterschiedlichem Umfang berücksichtigen ((*P*11), (*P*12), (*P*13)). Bisweilen kann es auch sinnvoll sein, eingeschränkte Skalen zu verwenden wie z.B. einen pentatonischen Tonvorrat bezogen auf einen Grundton (*P*14). Auch für Intervalle sind zwei Definitionen angegeben. Die erste Definition gibt die Größe eines Intervalls numerisch in Halbtonschritten an (*P*16), die zweite Definition folgt der Konvention, die Art eines Intervalls (z.B. "Terz") mit einem Modifikator (z.B. k=klein) zu kombinieren (*P*17).

Die Produktionstemplates setzen sich aus Alternativenregeltemplates, Alternativenregeln, Definitionsregeltemplates und Definitionsregeln zusammen. (*P*9) ist ein Alternativenregeltemplate, weil unter den Alternativen auf der rechten Seite Nichtterminalsymboltemplates auftreten, die noch expandiert werden müssen. (*P*10) und (*P*15) sind dagegen einfache Alternativenregeln, die man ohne Expansion zur Ableitung eines Objekts einsetzen kann. (*P*10) ersetzt das abstrakte Nichtterminalsymbol *Tonhöhe* durch eines der atomaren Nichtterminalsymbole *THMidi*, *THChromOktaväquiv*, *THEnhÄquiv* oder *THPentatonischEs*.

Auch bei den Definitionsregeltemplates und Definitionsregeln lassen sich verschiedene Formen unterscheiden. (P_{35}) ist ein Definitionsregeltemplate mit einer unbestimmten Anzahl von Elementen, das auch in einer Templategrammatik auftreten könnte. Bei den Definitionsregeltemplates (P_{34}) und (P_{36}) kommen hingegen auf der rechten Seite des Produktionstemplates semantische Bedingungen hinzu, die in einer Templategrammatik nicht vorgesehen sind. Ein Häufigkeitsobjekt (P_{34}) besteht aus einem Vektor v von Objekten, deren Typ durch den Templateparameter X festgelegt ist, und einem Vektor w von Häufigkeiten der Objekte. Da die Häufigkeiten den Objekten zugeordnet werden sollen, fordert die Bedingung $|v| = |w|$, dass die beiden Vektoren die gleiche Länge haben sollen. Die Häufigkeiten liegen zwischen 0 und 1 und sollen aufsummiert 1 ergeben.

Die Definitionsregeln enthalten unterschiedliche Elemente, etwa ein Nichtterminalsymboltemplate mit eingesetztem Templateparameter (P_{19}), keine Elemente ((P_{20}), (P_{22})), drei atomare Nichtterminalsymbole (P_{26}) und ein abstraktes Nichtterminalsymbol (P_{32}). Bei der Erzeugung eines Ambitusobjekts mit (P_{32}) können die beiden Nichtterminalsymbole *Tonhöhe* zu unterschiedlichen atomaren Tonhöhen abgeleitet werden.

Typdiagramm für benutzerdefinierten Teil. Die Alternativenregeltemplates in $\widetilde{P}_{TPB}$ legen die Hierarchie für die Typen fest, die von der Erweiterbaren Zeitreihengrammatik im Beispiel induziert werden. Abbildung 4.14 zeigt das Typdiagramm $\mathcal{D}$ für den benutzerdefinierten Teil des Beispiels. Da man durch die Expansion von Templates unendlich viele Typen für eine Erweiterbare Zeitreihengrammatik erhält, kann man das Typdiagramm nur teilweise visualisieren. Ersatzweise sind die freien Parameter hier in einem gestrichelten Kasten dargestellt. In Abbildung 4.14 sind die Tonhöhen und die Intervalle zu abstrakten Typen **Tonhöhe** bzw. **Intervall** zusammengefasst. Alle anderen Typen sind direkte Spezialisierungen des allgemeinsten benutzerdefinierten Typs *BenutzerdefiniertesObjekt*.

Ein musikalisches Objekt mit geschachtelten Templates. In der Computerlinguistik ist ein n-Gramm eine Folge n aufeinanderfolgender Zeichen in einem Text. Das Wort "Text" enthält z.B. die Trigramme "Tex" und "ext". C. Shannon hat n-Gramme verwendet, um die Redundanz der englischen Sprache zu berechnen [77]. Die Idee lässt sich auch auf die Musik übertragen. Allgemeiner besteht ein n-Gramm aus n Objekten desselben Typs. Beispielsweise ist

$$(1,-1),(-1,-1),(-1,-1),(-1,-1),(-1,1),(1,-1)$$

die Folge der Kontur-Bigramme für die erste Phrase des Vivaldi-Themas in Abbildung 4.5. Der erste Vektor $(1,-1)$ beschreibt die Auf-Ab-Bewegung des Melodieanfangs *es"-b"-as"*. Das musikalische Objekt in Abbildung 4.2.6, graphisch dargestellt in Abbildung 4.16, gibt die relativen Häufigkeiten der Kontur-Bigramme für die erste Phrase des Vivaldi-Themas wieder. Der geschachtelte Typbezeichner **Häufigkeit** $\langle$**Bigramm** $\langle$**Kontur**$\rangle\rangle$ gibt an, dass Häufigkeitsobjekte von Kontur-Bigrammen dargestellt werden. Der Vektor v besteht aus den drei verschiedenen Kontur-Bigrammen $(-1,1)$, $(-1,-1)$, $(1,-1)$, die in der Phrase auftreten. Der zweite Vektor w enthält die relativen Häufigkeiten der unterschiedlichen Kontur-Bigramme bezogen auf die Phrase. Die Bedingung $|v| = |w|$ stellt sicher, dass das Objekt ebensoviele Kontur-Bigramme wie Häufigkeiten enthält. Der zweite Teil der Bedingung besagt, dass die relativen Häufigkeiten Werte zwischen 0 und 1 annehmen und summiert 1 ergeben sollen.

Das Beispiel illustriert die Gültigkeitsbereiche von Attributbezeichnern (W_9'''): Jedes Bigramm enthält einen Vektor v, dessen Länge durch die Bedingung $|v| = 2$ festgelegt wird.

(**Häufigkeit** ⟨**Bigramm** ⟨**Kontur**⟩⟩ ;
  v : (**Vektor** ⟨**Bigramm** ⟨**Kontur**⟩⟩ ;
    (**Bigramm** ⟨**Kontur**⟩ ;
      v : (**Vektor** ⟨**Kontur**⟩
        (**Kontur**; (**Int**; 1)),
        (**Kontur**; (**Int**; −1))
      );
      $|v| = 2$
    ),
    (**Bigramm** ⟨**Kontur**⟩ ;
      v : (**Vektor** ⟨**Kontur**⟩
        (**Kontur**; (**Int**; −1)),
        (**Kontur**; (**Int**; −1))
      );
      $|v| = 2$
    ),
    (**Bigramm** ⟨**Kontur**⟩ ;
      v : (**Vektor** ⟨**Kontur**⟩
        (**Kontur**; (**Int**; −1)),
        (**Kontur**; (**Int**; 1))
      );
      $|v| = 2$
)
  ),
  w : (**Vektor** ⟨**Rational**⟩ ;
    (**Rational**; 0.33),
    (**Rational**; 0.5),
    (**Rational**; 0.17)
  );
  $|v| = |w| \wedge \forall 1 \leq i \leq |w| : \ 0 \leq w_i \leq 1 \wedge \sum_{i=1}^{|w|} w_i = 1$
)

Abbildung 4.15: Häufigkeits-Objekt für Kontur-Bigramme

Auf der nächsthöheren Ebene gibt es ebenfalls einen Vektor v, der aus Kontur-Bigrammen besteht. Auf diesen Attributbezeichner bezieht sich die Bedingung $|v| = |w| \wedge \forall 1 \leq i \leq |w| : 0 \leq w_i \leq 1 \wedge \sum_{i=1}^{|w|} w_i = 1$ auf der obersten Hierarchieebene des Objekts. Innerhalb der Kontur-Bigramme ist der äußere Attributbezeichner v nicht sichtbar.

Strukturierung von Problemwissen. Zum Schluss dieses Abschnitts kommen wir auf das Beispiel zur Strukturierung von musikalischem Problemwissen vom Anfang dieses Kapitels zurück (Abschnitt 4.2.1). Dort wurde thematisiert, dass sich die musikwissenschaftliche Praxis, gleiche Symbole zur Bezeichnung mehr oder weniger verwandter musikalischer Gegenstände zu verwenden, nicht auf die computergestützte Repräsentation musikalischer Strukturen übertragbar läßt. Die Erweiterbare Zeitreihengrammatik aus Abschnitt 4.2.6 realisiert die Symbolmengen, die einander zuvor teilweise überlappten, durch verschiede-

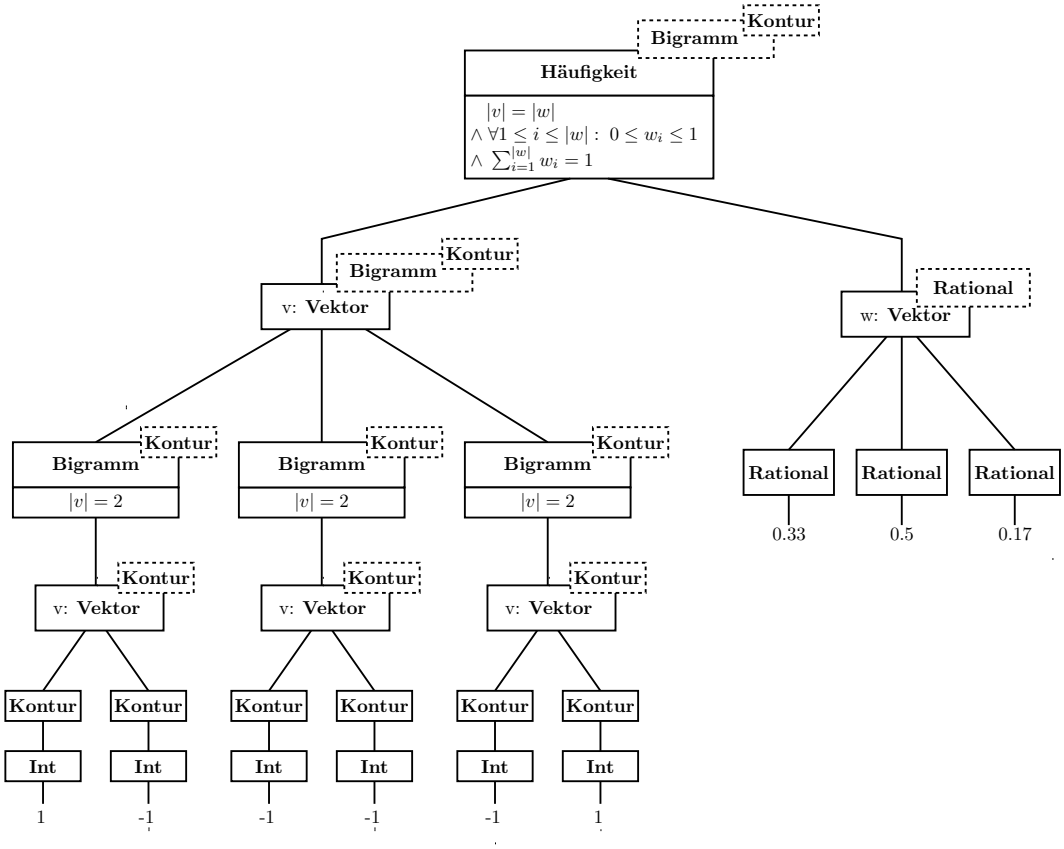


Abbildung 4.16: Ein Häufigkeitsobjekt, das einen Vektor von n-Grammen sowie Bewertungen enthält. Die Zahlen im rechten Vektor geben die Häufigkeiten der entsprechen Kontur-Bigramme im linken Vektor für die erste Phrase des Vivaldi-Themas in Abbildung 4.5 an.

nen Typen. Ein "D" steht z.B. sowohl für das Funktionssymbol der Dominante, als auch für die Tonart D-Dur und eine mit "D" bezeichnete Motivklasse. In der Erweiterbaren Zeitreihengrammatik wird der Buchstabe "D" zwar auch mehrfach verwendet, etwa in den musikalischen Objekten (**HarmonischeFunktion**; (**String**; D)), (**Tonart**; (**String**; D)) und (**Motivklasse**; (**String**; D)). Die jeweilige Bedeutung des "D" erschließt sich aber nun anhand der Typbezeichner **HarmonischeFunktion**, **Tonart** und **Motivklasse**.

In Abschnitt 4.2.1 wurde auch erwähnt, dass nicht jedes musikalische Objekt außer seiner zeitlichen Position weitere musikalische Information besitzen muss. Als Beispiele wurden Motive und Phrasen genannt. Sie werden auch im obigen Beispiel definiert und können nun, im Gegensatz zu Tabelle 4.2.1, voneinander unterschieden werden. Die Objekte (**Segment**; (**Int**; 0); (**Int**; 8); (**Motiv**)) und (**Segment**; (**Int**; 0); (**Int**; 8); (**Phrase**)) geben jeweils ein Zeitintervall bestehend aus Einsatzzeit und Dauer an. Ihr Typbezeichner weist sie als Motiv bzw. als Phrase aus.

4.3 Musikalische Strukturen

In Abschnitt 4.1 wurden Zeitgitter, Segmente und Segmentierungen definiert, um die Zeit darzustellen. Diese wurde in Abschnitt 4.2 in die Definition musikalischer Objekte durch eine Erweiterbare Zeitreihengrammatik integriert. Die musikalischen Objekte werden nun zu größeren Strukturen zusammengesetzt.

Bei der Definition einer Erweiterbaren Zeitreihengrammatik wurde eine parametrisierte Ansicht **Ansicht** $\langle X \rangle$ definiert, die Objekte vom Typ X mit Zeitinformation verknüpft und zu einem Vektor zusammenfasst. Eine Komposition wird durch Ansichten unterschiedlichen Typs dargestellt. Gleichartige Kompositionen bilden einen *Corpus*, der Trainingsbeispiele für das maschinelle Lernen enthält.

Definition 4.12 (Ansicht, Komposition, Corpus)
Gegeben sei eine Erweiterbare Zeitreihengrammatik mit Typbezeichnern T_{TP}. Jedes Element der Menge Typ(**Ansicht** $\langle t \rangle$), $t \in T_{TP}$, heißt *Ansicht vom Typ* t. Eine $(t_1, \ldots, t_n)$-*Komposition* K ist ein Vektor, der aus n Ansichten mit Typen $t_1, \ldots, t_n \in T_{TP}$ besteht. Eine endliche Folge von $(t_1, \ldots, t_n)$-Kompositionen heißt $(t_1, \ldots, t_n)$-*Corpus*.

* * *

Die Parametrisierung von Komposition und Corpus mit Typbezeichnern $(t_1, \ldots, t_n)$ stellt sicher, dass man Merkmale und Lernmuster auf allen Kompositionen eines Corpus in gleicher Weise berechnen kann. Die Begriffe Komposition und Corpus sind durch die Untersuchung musikalischer Strukturen motiviert, deren Analyse den Schwerpunkt dieser Arbeit darstellt. Die Definitionen von Komposition und Corpus sind jedoch so allgemein, dass sie auch auf nicht-musikalische Zeitreihen übertragen werden können.

Beispiel 4.5 zeigt mehrere Ansichten des Largo aus dem Winter von Vivaldis "Vier Jahreszeiten", die jeweils einen bestimmten Aspekt der Melodie herausheben. Beispiele für Corpora sind "Alle Kinderlieder im 2/4-Takt in der EsAC-Melodiesammlung" oder eine "Zufällige Auswahl von 100 Altstimmen aus den Bachchorälen". Das erste Beispiel stellt einen Corpus anhand von beschreibenden und musikalischen Attributen zusammen: Die Klassifikation einer Melodie als Kinderlied ist keine Eigenschaft, die aus der Melodie zu erkennen ist, sondern eine externe Zuschreibung. Die Taktart kann hingegen anhand des Notentextes der Melodie bestimmt werden. Eine solche Filterung von Musikstücken anhand verschiedener Kriterien kann hilfreich sein, wenn man homogene musikalische Trainingsmengen sucht.

Der Ansichtsbegriff ist insofern zentral, als dass er bei der Zeitreihen-Analyse von der Betrachtung einzelner Ereignisse wegführt hin zu einer abstrakteren Modellierung von Zeitreihen aus verschiedenen Perspektiven. Er stellt die Voraussetzung dar für ein flexibles Konzept zur Analyse und Vorverarbeitung von Merkmalen für das maschinelle Lernen von Musikstilen.

Die Erweiterbare Zeitreihengrammatik schafft nun ein Fundament für die formale Definition des Ansichtsbegriffs, indem sie typisierte Datenstrukturen zur Repräsentation zeitabhängiger Merkmale generiert, die die Trennung zeitreihenspezifischer und problemspezifischer Aspekte ermöglicht.

Was nun noch fehlt, ist ein Mechanismus zur automatisierten Berechnung von Ansichten aus vorgegebenen Zeitreihen. Diese Operationalisierung leisten Operatorgraphen, die im nächsten Kapitel eingeführt werden.

Die Kombination von Zeitreihengrammatik und Ansichten mit Operatorgraphen zeichnet sie gegenüber anderen perspektivischen Musikstrukturmodellen (vgl. Abschnitt 3.2.2) dadurch aus, dass sie explizit die Repräsentation und Transformation zeitabhängiger Daten trennt, die für die Austauschbarkeit von Datenmengen bei der Generierung von Lernmustern für das maschinelle Lernen wichtig ist.

Kapitel 5

Transformation musikalischer Strukturen

In Kapitel 4 wurde eine Repräsentation entwickelt, die musikalische Kompositionen und ihre Analysen mit Hilfe von *Ansichten* aus verschiedenen Blickwinkeln beschreibt. Die Repräsentation hält den Zustand der betrachteten musikalischen Strukturen statisch fest; sie gibt also keinen Aufschluss darüber, wie eine Ansicht berechnet wurde.

In diesem Kapitel wenden wir uns der Frage zu, wie man neue Ansichten eines Musikstücks erzeugen kann. Typischerweise liegen die Musikbeispiele, mit denen man ein lernbasiertes Modell trainieren will, in einer elementaren Beschreibung vor, die die für das Lernen relevanten musikalischen Zusammenhänge nur in unzureichendem Maße ausdückt. Daher benötigt man alternative Beschreibungen der Trainingsbeispiele, die einem lernbasierten Modell Wissen über musikalische Zusammenhänge zugänglich machen. In diesem Kapitel werden *Operatorgraphen* entwickelt, mit denen man problemspezifische Merkmale beschreiben und neue Ansichten von Zeitreihen berechnen kann.

Anforderungen an die Merkmalsmodellierung. Wenn man musikalische Stile mit lernbasierten Modellen charakterisieren will, benötigt man einerseits viele Trainingsbeispiele, um eine angemessene Lernleistung zu erreichen. Andererseits stehen viele potentiell relevante musikalische Merkmale zur Verfügung. Die Merkmale unterscheiden sich teilweise nur durch ihren zeitlichen Kontext (z.B. die Form eines Motivs und die Form einer Phrase). Andere Motive haben den gleichen zeitlichen Kontext, drücken aber eine unterschiedliche musikalische Information aus (z.B. das Intervall und Kontur zwischen dem ersten und letzten Ton der aktuellen Phrase). Daraus ergeben sich folgende Anforderungen an die Modellierung musikalischer Merkmale:

- Die Merkmale müssen automatisch aus der Basisdarstellung der Trainingsbeispiele zu berechnen sein. Eine manuelle Berechnung musikalischer Merkmale ist wegen der großen Trainingsmengen nicht praktikabel.

- Man benötigt eine Beschreibung der Merkmale, die es einem Merkmalsselektionsverfahren erlaubt, aus einer Menge von Kandidaten durch Optimierung eine geeignete Teilmenge von Merkmalen auszuwählen, die sich zur Lösung einer vorgegebenen Lernaufgabe eignen.

- Die Berechnung der Merkmale sollte modularisiert werden, damit wiederkehrende Teile der Merkmalsberechnung ohne Programmieraufwand wiederverwendet werden können.

Ideen des Ansatzes. Die Berechnung von Merkmalen wird in Bausteine zerlegt, die zu komplexeren Funktionen zusammengesetzt werden können. Die Bausteine heißen *Operatoren*, weil sie auf musikalischen Kompositionen (oder anderen Zeitreihen) „operieren". Operatoren werden zu *Operatorgraphen* verkettet, die beschreiben, wie aus einer musikalischen Kompositionen ein neues Merkmal oder Lernmuster berechnet werden soll. Ein Operatorgraph ist in ein allgemeineres Rahmenverfahren eingebettet, das die Auswertung des Operatorgraphen anstößt und das berechnete Ergebnis weiterverwertet. Beispiele für Rahmenverfahren sind die Ansichten- und Lernmusterberechnung, die Merkmalsselektion und die evolutionäre Komposition. Das Rahmenverfahren koordiniert, wie Operatorgraphen mit den Kompositionen interagieren: Beim Berechnen von Merkmalen und bei der Erzeugung neuer Musikstücke wird das Ergebnis eines Operatorgraphen beispielsweise in eine geeignete Ansicht eingefügt. Werden Lernmuster generiert, so schreibt das Rahmenverfahren die erzeugten Muster in eine Datei.

Ein Operatorgraph ist also eine Vorschrift für die Berechnung eines Merkmals. Das Merkmal selbst erhält man, wenn man von der Implementierung der Berechnung abstrahiert und nur die Abbildung zwischen Ein- und Ausgabe eines Operatorgraphen betrachtet. Die Menge aller Operatorgraphen, die sich aus einem Vorrat an Operatoren bilden lassen, beschreibt damit implizit einen „Merkmalsraum". Der hier entwickelte Ansatz bietet dem Anwender die Möglichkeit, aus einem Vorrat an Operatoren immer neue Operatorgraphen zu entwerfen, ohne selbst Operatoren implementieren zu müssen.

Neben der Modularisierung der Merkmalsberechnung mit Operatorgraphen berücksichtigt der Ansatz als zweite zentrale Idee die Eigenarten von Zeitreihen. Die Behandlung zeitlicher Aspekte ist für Zeitreihen unterschiedlicher Anwendungsfelder immer ähnlich, wohingegen das Hintergrundwissen, welches in die Merkmale integriert werden soll, für verschiedene Fachgebiete und sogar einzelne Anwendungen variiert. Wenn man die Merkmals- und Lernmusterberechnung existierender neuronaler Systeme zur Musikmodellierung (HARMONET, MELONET, MeloGeNet [36]) analysiert, stellt man fest, dass gewisse atomare Transformationen musikalischer Strukturen immer wieder auftreten. Dazu gehören die Fortbewegung entlang einer Zeitachse, das Auffinden gleichzeitiger Ereignisse und die Zusammenfassung musikalischer Ereignisse zu einer Einheit (z.B. Motive und Harmonien).

Der hier entwickelte Ansatz berücksichtigt dies, indem die Operatoren an ihrer Funktionalität orientiert in Gruppen eingeteilt werden. Es hat sich als zweckmäßig erwiesen, zwischen Operatoren zur zeitlichen Fortbewegung, zur Modellierung von Gleichzeitigkeit, zur Berechnung von Anwendungswissen und zur Strukturmanipulation zu unterscheiden. Insbesondere die Zusammenfassung der Wissensoperatoren zu einer Gruppe ermöglicht es, den Ansatz leicht an neue musikalische Inhalte oder nicht-musikalische Zeitreihen anzupassen.

Neuerungen. Gegenüber früheren Ansätzen bietet die systematische Beschreibung der Berechnung von Merkmalen mit Operatorgraphen mehrere Vorteile.

- **Wiederverwendbarkeit von Operatoren.** Das Konzept der Operatorgraphen erlaubt es, Operatoren auf viele Arten miteinander zu kombinieren. Ist ein Vorrat an Operatoren vorhanden, so kann man neue Merkmale spezifizieren ohne zu programmieren. Die Implementierung einer Funktion zur Merkmalsberechnung erfordert vom Benutzer hingegen Programmierkenntnisse sowie die Auseinandersetzung mit den Datenstrukturen der Rahmenalgorithmen, die die neuen Merkmale verwenden sollen. Die Wiederverwendbarkeit von Operatoren garantiert außerdem, dass atomare Teilaufgaben bei der Merkmalsberechnung immer auf dieselbe Weise durchgeführt werden.

- **Austauschbarkeit von Merkmalen.** Aus der Sicht eines Rahmenverfahrens sind alle Operatorgraphen austauschbar, deren Ein- und Ausgabespezifikation den Erwartungen des Rahmenverfahrens entspricht. Der Aufbau eines Operatorgraphen und seine Auswertung sind für ein Rahmenverfahren eine Blackbox. Operatorgraphen ermöglichen es daher, Algorithmen zu schreiben, deren Merkmale beim Entwurf noch nicht bekannt sein müssen und frei ausgetauscht werden können, solange sie eine vorgegebene Ein- und Ausgabespezifikation erfüllen.

- **Visualisierung.** Operatorgraphen sind graphisch darstellbar. Sie abstrahieren von der genauen Implementierung der Merkmalsberechnung, indem sie atomare Aufgaben wie den Sprung zu einem anderen Ereignis in Operatoren kapseln. Die Operatoren werden zu einem graphisch darstellbaren Datenflussdiagramm verknüpft, das für Anwender besser nachvollziehbar ist als eine monolithische Implementierung derselben Funktionalität. Insbesondere Anwendern, die eher in musikalischen als in informatischen Begrifflichkeiten denken, soll diese Abstraktion entgegenkommen.

Abkürzungen. In Kapitel 4 wurde die Erweiterte Zeitreihengrammatik eingeführt, aus deren Elementen sich Ansichten sowie die Werte- und Definitionsbereiche der Operatoren und Operatorgraphen zusammensetzen. Um die Definitionen und Beispiele dieses Kapitels kompakter zu notieren, führen wir zwei Abkürzungen ein.

Für ein Nichtterminalsymbol $X \in N_{TP}$ wird der Typ $\mathrm{Typ}(X) \in \mathcal{T}$ von nun an abkürzend durch den zugehörigen Typbezeichner $\widetilde{\alpha}(X) \in T_{TP}$ bezeichnet. Wenn z.B. vom Typ **Int** die Rede ist, ist eigentlich der Typ $\mathrm{Typ}(\mathit{Int})$ gemeint. Die Basistypen **Bool**, **Int**, **Rational** und **String** werden durch die entsprechenden Zahlen und Zeichenketten ersetzt. Wenn die genaue Beschreibung eines musikalischen Objekts unerheblich ist, wird sie durch die Verwendung musikalischer Symbole abgekürzt. Beispielsweise wird ein Segment, das zur Einsatzzeit 16 beginnt, 12 Zeiteinheiten dauert und als Wert einen $\frac{3}{2}$-Takt enthält

$$(\mathbf{Segment}\,\langle\mathbf{Takt}\rangle\,;\,(\mathbf{Int};\,16),(\mathbf{Int};\,12),(\mathbf{Takt};\,(\mathbf{Int};\,3),(\mathbf{Int};\,2))),$$

kürzer notiert als

$$(\mathbf{Segment}\,\langle\mathbf{Takt}\rangle\,;\,16,12,(\mathbf{Takt};\,\frac{3}{2})).$$

Die Menge aller Segmente einer Erweiterbaren Zeitreihengrammatik wird mit

$$\mathbf{Segment} = \bigcup_{X \in T_{TP}} \mathbf{Segment}\,\langle X \rangle$$

bezeichnet.

5.1 Operator

Ausgangslage. Wenn man Lernmuster oder musikalische Merkmale berechnet, dient eine Menge von Kompositionen als Ausgangsmaterial. Eine *Komposition* besteht aus typisierten *Ansichten*, die Objekte vom Typ **Segment** $\langle t \rangle$ enthalten, wobei $t \in T_{TP}$ ein expandiertes Typbezeichnertemplate der Grammatik ist (Definition 4.12). Die Segmente stellen musikalische Ereignisse mit der Zeitinformation (Einsatzzeit, Dauer) dar und sind innerhalb einer Ansicht zeitlich geordnet.

Die gewünschte Transformation der Komposition wird durch einen *Operatorgraphen* beschrieben, der sich aus *Operatoren* zusammensetzt (Definitionen 5.1 und 5.8). Operatoren sind Funktionen, die Objekte abbilden, wobei sie die Position der Objekte in einer Komposition berücksichtigen. Die Definitions- und Wertebereiche von Operatoren und Operatorgraphen setzen sich aus den Typen einer Erweiterbaren Zeitreihengrammatik zusammen, damit sie auf den gleichen Daten wie die Kompositionen definiert sind. Die Typen, die mit dem kartesischen Produkt zum Definitions- oder Wertebereich eines Operators verknüpft werden, werden im folgenden als *Eingabetypen* bzw. *Ausgabetypen* bezeichnet.

Anwendung eines Operatorgraphen. Um den Operatorgraphen auf eine Komposition anzuwenden, muss man festlegen, welche Ansicht als Eingabeansicht des Operatorgraphen dienen soll. Als Eingabeansichten kommen alle die Ansichten in Frage, deren Typ spezieller oder gleich dem Eingabetyp des Operatorgraphen ist. Wenn der Operatorgraph Objekte einer Ansicht berechnet, muss auch die Ausgabeansicht in der Komposition ausgewählt und gegebenenfalls neu erzeugt werden. Bei der Berechnung von Lernmustern ist dies nicht nötig.

Kontext. Es wäre ausreichend, die Ein- und Ausgabeansicht eines Operatorgraphen festzulegen, wenn man das Verhalten von Operatoren unabhängig von einer Komposition definieren könnte. Bei der Charakterisierung von Musikstücken spielt aber im allgemeinen der Kontext, in dem ein musikalisches Ereignis auftritt, eine wichtige Rolle. Es ist daher nicht sinnvoll, Operatoren als feste Abbildungen zwischen Objekten einer Erweiterbaren Zeitreihengrammatik zu definieren. Eine Abbildung musikalischer Ereignisse muss vielmehr in der Lage sein, den kompositorischen Kontext seiner Eingabeobjekte zu berücksichtigen. Sie muss beispielsweise die gleichzeitig mit einer vorgegebenen Note auftretenden musikalischen Ereignisse bestimmen können, die sich für jeden Zeitpunkt einer Komposition und auch zwischen verschiedenen Kompositionen unterscheiden. Andererseits kann man aber bei der Definition von Operatoren nicht die genaue Struktur jeder Komposition in Betracht ziehen, weil man musikalische Transformationen entwickeln möchte, die möglichst allgemein einsetzbar sind.

Dieser Zielkonflikt wird hier die Einführung sogenannter *Kontexttypen* gelöst, die ausdrücken, welche Typen von Objekten ein Operator in seinem musikalischen Kontext erwartet. Vor

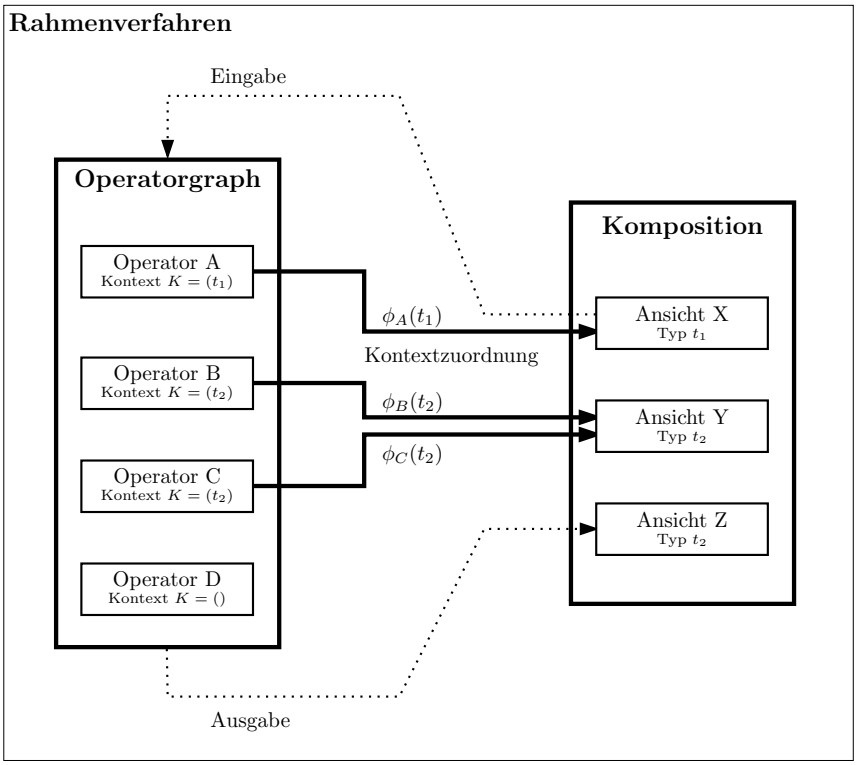


Abbildung 5.1: Verknüpfung eines Operatorgraphen mit einer Komposition. Der Fluss der Ein- und Ausgabeobjekte ist durch die gestrichelten Pfeile markiert. Die Kontexttypen der Operatoren A, B und C sind den Ansichten X und Y zugeordnet. Operator D benötigt keinen Kontext.

der Anwendung eines Operators auf eine Komposition werden den Kontexttypen des Operators Ansichten eines geeigneten Typs zugeordnet, d.h. es wird festgelegt, mit Hilfe welcher Ansichten eine Komposition die Erwartungen des Operators an seinen Kontext erfüllen soll. Die Zuordnung von Kontexttypen zu Ansichten ist nicht eindeutig. Beispielsweise ist eine zeitliche Navigation in jeder Ansicht möglich. Vielmehr stellt die Kontextzuordnung eine Entscheidung dar, auf welche Weise ein Operator in einem bestimmten Fall auf eine Komposition angewendet werden soll. Abbildung 5.1 zeigt die Kontextzuordnung zwischen Operatorgraph und Komposition sowie den Fluss der Ein- und Ausgabeobjekte. Die Trennung von Kontexttypen und Kontextzuordnung erlaubt es, einen Operator zu definieren, ohne andere Details der bearbeiteten Kompositionen zu kennen als die Typen einiger Ansichten.

Definition 5.1 (Operator)
Gegeben sei eine Erweiterbare Zeitreihengrammatik mit Typen $\mathcal{T}$.

(a) Ein *Operator* ist eine partielle Funktion

$$\mathbf{op}_K : t_1 \times \ldots \times t_r \longrightarrow t$$

deren Definitions- und Wertebereich sich aus Typen $t_1, \ldots, t_r, t \in \mathcal{T}$ der Erweiterbaren Zeitreihengrammatik zusammensetzt ($r \in \mathbb{N}$). Der Parameter $K = (k_1, \ldots, k_m) \in \mathcal{T}^*$ besteht aus einer endlichen Folge von Typen, den *Kontexttypen* des Operators. Ein nicht-definiertes Ergebnis eines Operators wird durch das Zeichen $\perp$ dargestellt. Wenn alle Eingaben eines Operators undefiniert sind, ist auch sein Ergebnis undefiniert.

(b) Gegeben seien ein Operator $\mathbf{op}_K$ mit Kontexttypen $K = (k_1, \ldots, k_m) \in \mathcal{T}^*$ und eine $(s_1, \ldots, s_n)$-Komposition $(A_1, \ldots, A_n)$ über der Erweiterbaren Zeitreihengrammatik, d.h. für alle $1 \leq i \leq n$ ist A_i eine Ansicht vom Typ **Ansicht** $\langle s_i \rangle$.

Eine *Kontextzuordnung* ist eine Folge von Ansichten $(\tilde{A}_1, \ldots, \tilde{A}_m) \in \{A_1, \ldots, A_n\}^m$ der vorgegebenen Komposition, deren Typen spezieller oder gleich den entsprechenden Kontexttypen aus K sind: Ist $\tilde{A}_i = A_j$ für ein $j \in \{1, \ldots, n\}$, so muss $s_j \preceq k_i$ gelten.

Die Zuordnung von Ansichten zu den Kontexttypen $(k_1, \ldots, k_m)$ der Kontextzuordnung $(\tilde{A}_1, \ldots, \tilde{A}_m)$ wird elementweise ausgedrückt durch $\phi_{\mathbf{op}_K}(k_i) = \tilde{A}_i$ bzw. $\phi(k_i) = \tilde{A}_i$, wenn der Operator aus dem Zusammenhang ersichtlich ist. Ist einem Operator $\mathbf{op}_K$ der Kontext $(\tilde{A}_1, \ldots, \tilde{A}_m)$ zugeordnet, so wird dies durch die Schreibweise $\mathbf{op}_K[\tilde{A}_1, \ldots, \tilde{A}_m]$ notiert.

(c) Ein Operator $\mathbf{op}_K$ ist auf eine Komposition *anwendbar*, wenn eine Kontextzuordnung zwischen den Kontexttypen des Operators und der Komposition existiert. Die Komposition, auf die der Operator angewendet wird, heißt dann *Kontextkomposition* des Operators.

(d) Ein *abstrakter Operator* ist ein unvollständig spezifizierter Operator, der aus einer Deklaration und Kontexttypen besteht. Sein Verhalten kann durch Bedingungen eingeschränkt werden, die die Abbildung oder die Zugehörigkeit der Ein- oder Ausgabeobjekte zu Kontextansichten beschreiben. Ein *(konkreter) Operator* ist ein Operator, dessen Abbildung vollständig spezifiziert ist. Ein Operator ist eine *Spezialisierung* eines abstrakten Operators, wenn seine Deklaration sich als Spezialfall der Deklaration des abstrakten Operators auffassen lässt und wenn er alle Bedingungen des abstrakten Operators erfüllt.

* * *

Beispiel. Gegeben sei die Erweiterbare Zeitreihengrammatik mit dem benutzerdefinierten Teil aus Abschnitt 4.2.6. Ein Beispiel für einen Operator ist der *Iterator*

$$\mathbf{it}_{+1} : \textbf{ Segment} \longrightarrow \textbf{Segment} \text{ mit } K = (\textbf{Objekt}),$$

der in einer Kontextansicht eines Typs $\preceq$ **Objekt**, d.h. in einer beliebigen Ansicht, den Nachfolger des Eingabesegments sucht. Definitions- und Wertebereich des Iterators ist die

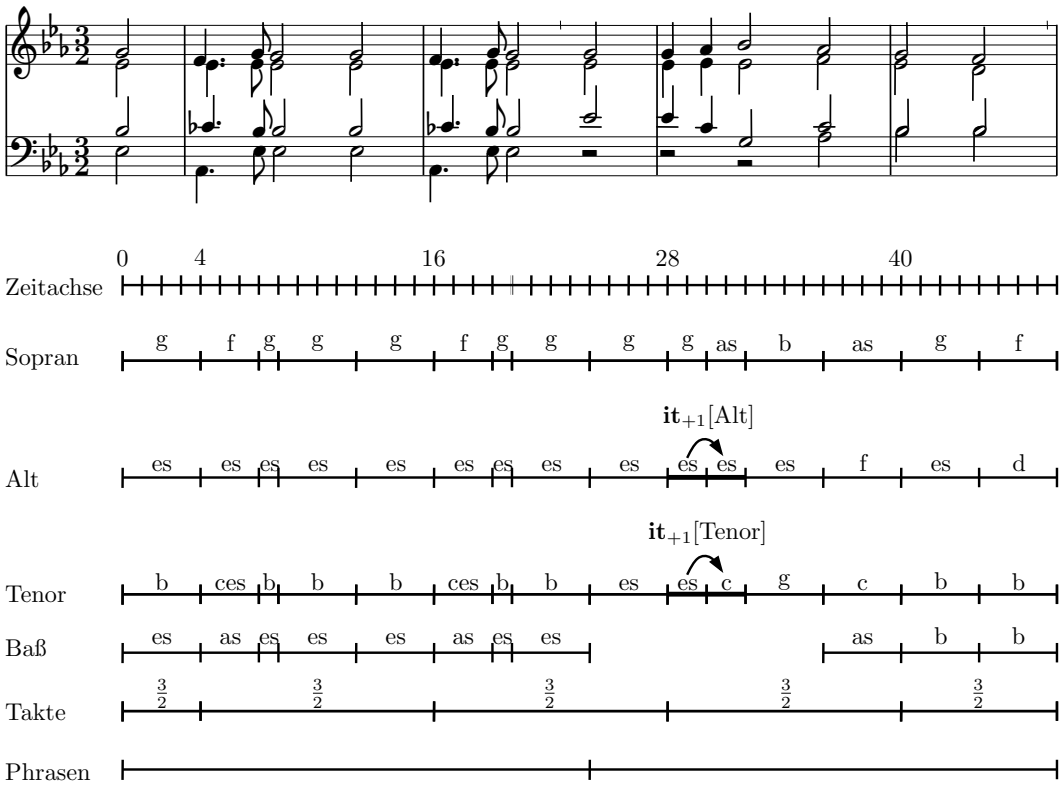


Abbildung 5.2: Anfang des Chorsatzes: „In stiller Nacht“ von Johannes Brahms

Menge **Segment**, die alle Segmenttypen der zugrundeliegenden Erweiterbaren Zeitreihengrammatik vereinigt.

Um den Iterator auf eine Komposition anzuwenden, muss man dem Kontexttyp **Objekt** zunächst eine Ansicht zuordnen. In Abbildung 5.2 ist der Anfang des Chorsatzes „In stiller Nacht“ von Johannes Brahms mit vier Ansichten vom Typ **Tonhöhe** sowie je einer Ansicht der Takte und der Phrasen dargestellt. Der Iterator $\mathbf{it}_{+1}$ kann in jeder dieser Ansichten iterieren. Will man etwa in der Altstimme navigieren, so ordnet man dem Kontexttyp **Objekt** die Ansicht „Alt“ zu, also $\phi(\mathbf{Objekt}) = \text{Alt}$. Dies ist möglich, weil der Typ der Alt-Ansicht, **Tonhöhe**, den Kontexttyp **Objekt** des Iterators spezialisiert. Das Musikbeispiel dient nun als Kontextkomposition für den Iterator $\mathbf{it}_{+1}$. Sie bildet den musikalischen Kontext, in dem der Iterator operiert.

Die Existenz einer Kontextzuordnung zeigt, dass der Iterator auf das Musikbeispiel anwendbar ist. Im Falle des Iterators $\mathbf{it}_{+1}$ ist die Anwendbarkeit trivial, da der Kontexttyp **Objekt**, der allgemeinste Typ einer Erweiterbaren Zeitreihengrammatik, mit Ansichten beliebigen Typs verknüpft werden kann. Der Iterator $\mathbf{it}_{+1}$ kann auf beliebigen Ansichten iterieren.

Der Iterator $\mathbf{it}_{+1}$[Alt], dessen Kontextzuordnung in eckigen Klammern angezeigt wird, liefert

für das Eingabesegment

$$s = (\textbf{Segment}\ \langle \textbf{Tonhöhe} \rangle\ ;\ 28, 2, (\textbf{Tonhöhe};\ es))$$

den Nachfolger

$$s = (\textbf{Segment}\ \langle \textbf{Tonhöhe} \rangle\ ;\ 30, 2, (\textbf{Tonhöhe};\ es)).$$

Wählt man als Kontext die Tenor-Ansicht, so findet der Iterator $\mathbf{it}_{+1}$[Tenor] für dieselbe Eingabe das Segment

$$s = (\textbf{Segment}\ \langle \textbf{Tonhöhe} \rangle\ ;\ 30, 2, (\textbf{Tonhöhe};\ c)),$$

das eine andere Tonhöhe besitzt. Da das Eingabesegment sowohl in der Alt- als auch in der Tenorstimme auftritt, ist die Kontextzuordnung ist hier nötig, um einen eindeutigen Nachfolger des Eingabesegments zu bestimmen.

Der Iterator im Beispiel kann auf beliebigen Ansichten iterieren, weil sein Kontexttyp der allgemeinste Typ **Objekt** ist. Wir werden später auch Iteratoren kennenlernen, die speziellere Kontexttypen besitzen, weil sie Objekte bestimmter Typen zueinander in Beziehung setzen. Wenn ein Iterator etwa das nächste Motiv sucht, das den gleichen harmonischen Kontext wie die Eingabe hat, ist er auf Kontextansichten der Typen **Motiv** und **Harmonie** angewiesen. Die Unterscheidung von Kontexttypen und Kontextansichten ermöglicht es, den Typ der Kontextansichten einzuschränken, ohne bestimmte Ansichten einer Komposition in der Definition eines Operators angeben zu müssen. Auf diese Weise lassen sich die Anforderungen eines Operators an die Kontextkomposition spezifizieren, ohne den Operator vom Gesamtaufbau der Kontextkomposition oder der Benennung ihrer Ansichten abhängig zu machen.

Anwendung von Operatoren. Bei der Definition von Operatoren wurde zwischen zwei Arten von „Eingaben“ unterschieden. Zum einen haben Operatoren einen Definitionsbereich, der sich aus Typen einer Erweiterbaren Zeitreihengrammatik zusammensetzt. Daneben können Kontexttypen definiert werden, die vor der Auswertung eines Operators geeigneten Kontextansichten einer Komposition zugeordnet werden müssen. Eingabetypen und Kontexttypen spielen bei der Auswertung der Operatoren unterschiedliche Rollen. Wenn man einen Operator mehrfach in der gleichen Weise auf eine Melodie oder einen Corpus anzuwenden möchte, wählt man die Kontextzuordnung eines Operators nur einmal pro Komposition (bzw. Corpus). Mit dieser Zuordnung wird der Operator – typischerweise als Teil eines Operatorgraphen – an unterschiedlichen Stellen der Komposition wiederholt ausgewertet. Die Eingabeobjekte des Operators aus dem Definitionsbereich variieren dabei, nicht aber die Parametrisierung durch die Kontextansichten.

Gruppierung von Operatoren. Um Operatoren in Gruppen mit gleicher Funktionalität zusammenzufassen, werden abstrakte Operatoren eingeführt. Die Gruppierung von Operatoren ermöglicht es, später bei der Generierung von Operatorgraphen Mengen austauschbarer Operatoren zu beschreiben.

Ein *abstrakter Operator* charakterisiert ähnlich wie eine abstrakte Klasse in C++ eine Menge von (konkreten) Operatoren, die seine Eigenschaften erben. Abstrakte Operatoren bestehen aus einer Deklaration und Kontexttypen, die sich aus Typen der Erweiterbaren Zeitreihengrammatik und Typvariablen zusammensetzen. Das Verhalten abstrakter Operatoren wird

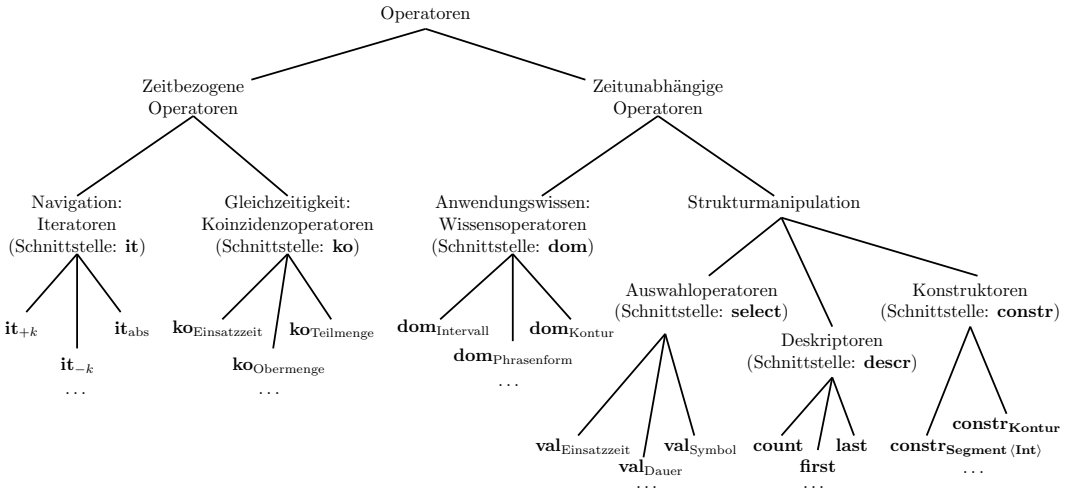


Abbildung 5.3: Operatorhierarchie

durch Bedingungen eingeschränkt, jedoch nicht vollständig spezifiziert. Durch Einsetzen von Typen in die Typvariablen erhält man Deklarationen *konkreter Operatoren*. Das Verhalten eines konkreten Operators muss die Bedingungen aller übergeordneten abstrakten Operators erfüllen. Damit wird sichergestellt, dass alle Operatoren einer Gruppe sich so verhalten, wie es durch die Bedingungen der abstrakten Operatoren der Gruppe festgelegt wird. Im Gegensatz zu einem abstrakten Operator ist das Verhalten eines (konkreten) Operators vollständig bekannt.

Durch die Spezialisierung abstrakter Operatoren erhält man eine Hierarchie von Operatorgruppen. Abbildung 5.3 zeigt die Operatorhierarchie für die Operatoren dieses Kapitels. Die inneren Knoten des Baums sind abstrakte Operatoren, die alle unter ihnen liegenden Operatoren zusammenfassen. Die Blattknoten bestehen aus konkreten Operatoren, die – da sie vollständig bekannt sind – nicht weiter spezialisiert werden können.

Beispiel. In Tabelle 5.6 findet man einige Beispiele für abstrakte Operatoren. Die allgemeine Deklaration eines Iterators hat die Form

$$\mathbf{it}: \ \mathbf{Segment} \times t_1 \times \ldots \times t_r \longrightarrow \mathbf{Segment}$$

und schränkt die Deklaration aller spezielleren Iteratoren ein. Die Eingabe eines Iterators besteht immer aus einem Segment und kann durch weitere Objekte beliebigen Typs ergänzt werden. Weiterhin besitzt ein Iterator mindestens einen Kontexttyp k_1. Das Verhalten eines Iterator wird durch die Bedingung $\mathbf{it}(x) \in \phi(k_1)$ eingeschränkt, die besagt, dass das Ergebnis des Iterators in der Ansicht $\phi(k_1)$ liegen muss, die dem Kontexttyp k_1 zugeordnet ist. Beispiele für konkrete Iteratoren sind der oben diskutierte Iterator $\mathbf{it}_{+1}$, der neben dem Eingabesegment die konstante Schrittweite 1 als Eingabe erhält, sowie die Iteratoren in Tabelle 5.7. Die Zugehörigkeit der Operatoren zu den verschiedenen Gruppen wird durch ihre Benennung angedeutet; die Namen der Operatoren haben aber keine gruppierende Funktion.

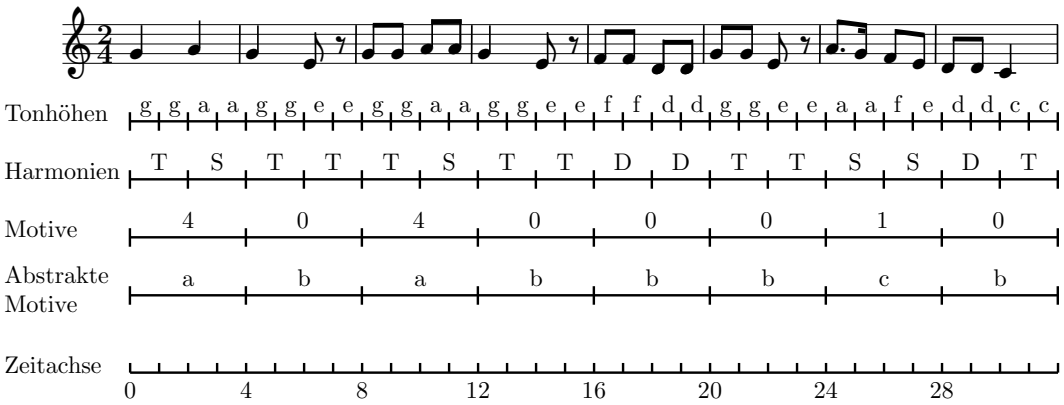


Abbildung 5.4: MeloGeNet: Rasterdarstellung des Kinderlieds „Kreis, Kreis, Kessel“

5.2 Operatoren für Zeitreihen

Die Berechnung musikalischer Merkmale lässt sich in einfache Teilaufgaben zerlegen, die durch Operatoren ausgeführt werden. In diesem Abschnitt werden Gruppen von Operatoren mit ähnlicher Funktionalität vorgestellt, die durch abstrakte Operatoren definiert werden.

Um die zeitspezifischen Teilaufgaben bei der Merkmalsberechnung für Zeitreihen unabhängig vom betrachteten Problemgebiet zu lösen, wird hier zwischen *zeitbezogenen* und *zeitunabhängigen* Operatoren unterschieden. Die Gruppe der zeitbezogenen Operatoren zerfällt ihrerseits in die *Iteratoren*, die sich entlang der Zeitachse fortbewegen und die *Koinzidenzoperatoren*, die gleichzeitig auftretende Ereignisse zueinander in Beziehung setzen. Bei den zeitunabhängigen Operatoren hat es sich als zweckmäßig erwiesen, die Modellierung von Hintergrundwissen in der Gruppe der *Wissensoperatoren* zusammenzufassen. Um den Ansatz an ein anderes Anwendungsgebiet von Zeitreihen anzupassen, müssen die Wissensoperatoren angepasst werden; die anderen Gruppen von Operatoren können weiterverwendet werden. Schließlich benötigt man *Strukturmanipulationsoperatoren*, die die allgemeine Struktur der Objekte verändern ohne ihre Bedeutung zu beachten.

Abbildung 5.3 gibt eine Übersicht über die hierarchische Gruppierung der Operatoren. An den Blättern des Baums befinden sich konkrete Operatoren, deren Verhalten vollständig definiert ist. Sie erben die Eigenschaften der über ihnen liegenden abstrakten Operatoren, die in den folgenden Abschnitten vorgestellt werden. Eine Übersicht über alle Operatoren, die in den Beispielen dieses Kapitels vorkommen, findet man in den Tabellen 5.6, 5.7 und 5.8.

5.2.1 Navigation

Wenn man Musik anhört, spielt, analysiert oder komponiert, begegnet man vielen Varianten zeitlicher Beziehungen und zeitlicher Fortbewegung. Man durchläuft ein Musikstück, wenn man es anhört oder spielt. Erinnert man sich an bereits gehörte musikalische Motive, so

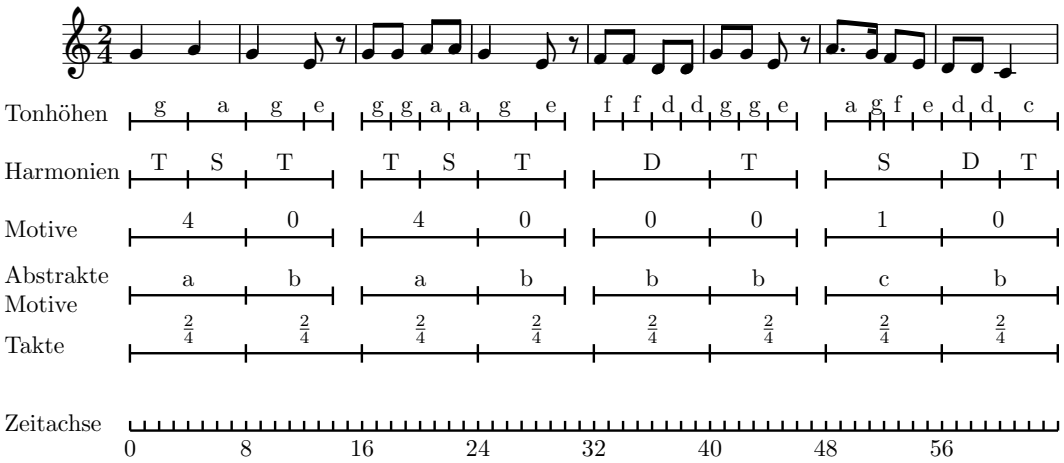


Abbildung 5.5: Ereignisorientierte Darstellung des Kinderlieds „Kreis, Kreis, Kessel“

stellt man einen relativen Bezug zu einem vergangenen Zeitpunkt her. Wenn man die Gesamtstruktur eines Musikstücks analysiert, kann man zeitliche Beziehungen abhängig oder unabhängig von ihrem zeitlichen Auftreten untersuchen.

Da man beim Entwurf neuer Merkmale nach musikalisch plausiblen Zusammenhängen sucht, orientiert man sich an solchen „natürlichen“ Situationen. Ein Beispiel macht deutlich, dass sich die hier verwendeten Segmente variabler Länge bereits ein einfaches Volkslied genauer darstellen als eine gerasterte Darstellung musikalischer Ereignisse.

Beispiel. Die Abbildungen 5.4 und 5.5 zeigen das Kinderlied „Kreis, Kreis, Kessel“ in der Rasterdarstellung des Melodiegenerierungssystems MeloGeNet und in der ereignisorientierten Repräsentation mit Segmenten variabler Länge. Die Repräsentation besteht aus vier Ansichten: Die Tonhöhenansicht enthält Tonhöhen ohne Oktavinformation, die Harmonieansicht enthält harmonische Funktionen (T=Tonika, D=Dominante, S=Subdominante). Die Motivansicht zeigt die Klassennummern einer fiktiven Motivanalyse. Motive mit gleichen Klassennummern werden als ähnlich betrachtet. Die Ansicht der Abstrakten Motive abstrahiert von den Nummern der Motivklassen. Die Buchstaben dieser Ansicht zeigen nur an, ob ein Motiv neu auftritt oder bereits in der Vergangenheit verwendet wurde.

Die Zeitbehandlung ist in beiden Repräsentationen der Melodie unterschiedlich. In der Rasterdarstellung werden die Tonhöhen in Achtelauflösung, die Harmonien in Viertelauflösung und die Motive in einer Auflösung von halben Noten repräsentiert. Es gibt keine Pausen, und Sechzehntelnoten gehen durch Glättung der Melodie verloren. Wenn ein Ereignis in mehrere aufeinanderfolgende gleiche Ereignisse zerlegt wurde, zeigt ein sogenanntes „Tenuto-Bit“ an (hier nicht notiert), dass es sich um ein übergebundenes Ereignis handelt.

In der ereignisorientierten Darstellung wird jedes musikalische Ereignis mit seiner tatsächlichen Länge modelliert. Pausen werden in diesem Beispiel durch das Nichtvorhandensein eines musikalischen Ereignisses angezeigt. Alternativ könnte man z.B. die Tonhöhen um ein

stummes Element erweitern. Auch überlappende Tonhöhen in einer Ansicht können hier dargestellt werden, treten aber im Beispiel nicht auf.

Nachbarschaft. Ein wesentlicher Unterschied zwischen den beiden Repräsentationen zeigt sich beim Durchlaufen der Melodie. Wenn man von einem Segment zu seinem Nachfolger springt, orientiert man sich bei der Rasterdarstellung an einem Zeitgitter, das von der betrachteten Melodie unabhängig ist. Da ein langes Ereignis in mehrere kürzere Ereignisse zerlegt wird, benötigt man mehrere Schritte, um es zu durchlaufen. Bei der ereignisorientierten Repräsentation springt man hingegen von einem musikalischen Ereignis der Originalmelodie zum nächsten. Die zeitliche Fortbewegung nimmt also Bezug auf den musikalischen Inhalt, nicht auf ein Zeitgitter. Die Nachbarschaft von Segmenten hat in den beiden Repräsentationen eine unterschiedliche musikalische Bedeutung. Während bei der Rasterrepräsentation zwei benachbarte Segmente sowohl zu demselben als auch zu benachbarten Ereignissen der Originalmelodie gehören können, sind zwei Segmente in der neuen Repräsentationen genau dann benachbart, wenn die zugehörigen Ereignisse in der Originalmelodie ebenfalls benachbart sind. Eine ereignisorientierte Darstellung ist daher zur Untersuchung von Beziehungen zwischen entfernten musikalischen Ereignissen geeigneter als eine Rasterdarstellung, weil sich die Entfernung zwischen den Ereignissen auf musikalisch plausiblere Weise ausdrücken lässt. Neben der absoluten zeitlichen Entfernung zwischen Ereignissen kann man eine Entfernung nun auch durch die Anzahl der zu überbrückenden musikalischen Ereignisse beschreiben.

Aufwand. Man könnte natürlich die Iteration in einer Repräsentation mit Segmenten variabler Länge mit Hilfe der Verbindung von Segmenten konstanter Länge simulieren, doch hängt der Aufwand für das Durchlaufen einer Melodie dann immer noch von der Auflösung des Zeitgitters und nicht von der konstanten Anzahl der Ereignisse der Melodie ab. Je feiner man die Auflösung wählt und je größer die verarbeitete Melodiemenge ist, umso stärker macht sich dieser Unterschied praktisch bemerkbar (vgl. Abbildung 4.3).

Mustergenerierung. Auch bei der Generierung von Lernmustern ist die Rasterdarstellung problematisch. Wenn man z.B. Übergänge zwischen benachbarten Ereignissen lernen will, führt die Zerteilung langer Ereignisse dazu, dass der Anteil der konstanten Übergänge zwischen gleichen Ereignissen in den erzeugten Mustern gegenüber der musikalischen Wahrnehmung überrepräsentiert ist. Um diesen Effekt abzuschwächen, wurden die Tonhöhen in MeloGeNet in Achteldauern aufgelöst, obwohl eine Auflösung in Sechzehnteldauern nötig wäre, um die Melodie korrekt darzustellen. Bei der ereignisorientierten Repräsentation tritt dieser Effekt nicht auf, da die repräsentierten mit den wahrgenommenen Ereignissen übereinstimmen.

5.2.1.1 Iterator

Definition 5.2 (Iterator)
Es seien $t_1, \ldots, t_r \in \mathcal{T}$, $r \geq 0$, und $k_1, \ldots, k_m \in \mathcal{T}$, $m > 0$, Typen einer Erweiterbare Zeitreihengrammatik. Ein *Iterator* ist ein abstrakter Operator mit einer Deklaration der Form

$$\mathbf{it}: \ \mathbf{Segment} \times t_1 \times \ldots \times t_r \longrightarrow \mathbf{Segment}, \quad K = (k_1, \ldots, k_m).$$

Wenn der Iterator für ein Element $(s, x_1, \ldots, x_r) \in \mathbf{Segment} \times t_1 \times \ldots \times t_r$ ein definiertes Ergebnis $\mathbf{it}(s, x_1, \ldots, x_r) \neq \perp$ liefert, liegt dieses in der Kontextansicht $\phi(k_1)$, was durch die

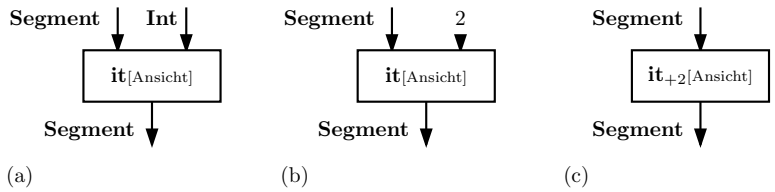


Abbildung 5.6: Graphische Darstellung von Iteratoren mit Kontexttyp t
und Kontextansicht $\phi(t) =$ „Ansicht“
(a) Iterator mit einem variablen Parameter vom Typ **Int**
(b) Iterator mit festem Parameter
(c) Iterator mit festem, in den Namen integrierten Parameter

Bedingung

$$\mathbf{it}(s, x_1, \ldots, x_r) \in \phi(k_1)$$

ausgedrückt wird. Das Ergebnis des Iterators darf nicht mit dem Eingabesegment s identisch sein, d.h. es muss

$$\mathbf{it}(s, x_1, \ldots, x_r) \neq s.$$

erfüllt sein.

* * *

Ein Iterator sucht zu einem vorgegebenen Segment ein benachbartes Segment, das in der Kontextansicht $\phi(k_1)$ liegt. Was unter der Nachbarschaft von Segmenten genau zu verstehen sein soll, legen die konkreten Iteratoren fest, die die abstrakte Definition eines Iterators spezialisieren. Als Minimalbedingung an das Verhalten eines Iterators verlangt die zweite Bedingung der Definition, dass ein Iterator sich überhaupt bewegt. Das Eingabesegment darf mit der Ausgabe des Iterators nicht identisch sein, d.h. $\mathbf{it}(s, x_1, \ldots, x_r) \neq s$.

Die Eingabe eines Iterators besteht immer aus einem Segment und kann durch Objekte beliebigen Typs ergänzt werden. Die erste Eingabe ist das Segment, das als Ausgangspunkt für die Suche des Iterators dient. Die fakultativen Eingaben $t_1, \ldots, t_r$ sind Parameter, mit denen sich das Verhalten eines Iterators variieren lässt. Ein Beispiel für einen Parameter ist die Anzahl der Schritte, die der Iterator bei einer Auswertung zurücklegt. Die Parameter eines Iterators können fest gewählt oder bei jeder Abfrage neu eingestellt werden. Abbildung 5.6 zeigt die graphische Darstellung eines Iterators mit einem festen und variablen Parameter.

Ein Iterator kann mehrere Kontexttypen $K = (k_1, \ldots, k_m)$ besitzen. Der erste Kontexttyp k_1 wird der Zielansicht des Iterators zugeordnet, d.h. der Ansicht, in der der Iterator nach einem Segment sucht. Die Tatsache, dass das Ergebnis eines Iterators zu einer bestimmten Ansicht gehört, wird durch die Bedingung $\mathbf{it}(s, x_1, \ldots, x_r) \in \phi(k_1)$ ausgedrückt (wenn es nicht undefiniert ist). Der erste Kontexttyp wird immer benötigt, daher ist $m \geq 1$. Die weiteren Kontexttypen werden Ansichten zugeordnet, die bei der Suche zusätzliche Information liefern.

Die Zugehörigkeit des Eingabesegments ist beim Iterator nicht festgelegt, damit auch Iteratoren definiert werden können, die unabhängig von der Eingabe an eine bestimmte Position einer Zielansicht springen. Typischerweise bewegt sich ein Iterator aber innerhalb einer Ansicht, d.h. das Eingabesegment gehört zur Zielansicht und das Ergebnis befindet sich einige Schritten vor oder hinter dem Eingabesegment.

5.2.1.2 Beispiele

Wie bereits diskutiert wurde, kann sich die Fortbewegung eines Iterators am musikalischen Inhalt oder am Zeitgitter orientieren. Dies wird nun am Beispiel einiger konkreter Operatoren illustriert.

Ereignisbasierte Navigation. Ein *relativer Iterator*

$$\mathbf{it}_{\text{rel}} : \mathbf{Segment} \times \mathbf{Int} \longrightarrow \mathbf{Segment}$$

bewegt sich in der Zielansicht um die Anzahl von Segmenten von seinem Eingabesegment fort, die durch das zweite Argument gegeben ist. Ist dieses Argument positiv, so springt der relative Iterator nach rechts, andernfalls nach links. Die Schrittweite $k \in \mathbb{N}$ eines relativen Iterators wird in den Beispielen mit $\mathbf{it}_{-k}$ und $\mathbf{it}_{+k}$ abgekürzt.

Um beim Durchlaufen einer Ansicht an einer bestimmten Position zu beginnen, benötigt man den *absoluten Iterator*

$$\mathbf{it}_{\text{abs}} : \mathbf{Segment} \times \mathbf{Int} \longrightarrow \mathbf{Segment}$$

der vom Anfang bzw. vom Ende einer Ansicht ausgehend soviele Segmente in die Ansicht hineinspringt, wie das zweite Argument vorgibt.

Dauernbasierte Navigation. Der absolute und der relative Iterator drücken den gesuchten Abstand durch die Anzahl von Segmenten aus, die zwischen dem Eingabesegment bzw. dem Anfang oder Ende des Musikstücks und dem gesuchten Segment liegen. Doch wie sucht man ein Ereignis, dessen Entfernung vom aktuellen Zeitpunkt durch ein anderes musikalisches Ereignis definiert wird? Als Beispiel wird nun gezeigt, wie man einen Ton zwei Takte nach einem aktuellen Ton findet. Die Volksliedmelodie in Abbildung 5.5 hat beispielsweise einen zweitaktigen Aufbau; die Takte 1-2 und 3-4 dieser Melodie gleichen einander.

Der *Gitteriterator*

$$\mathbf{it}_{\text{Dauer}} : \mathbf{Segment} \times \mathbf{Int} \times \mathbf{Int} \longrightarrow \mathbf{Segment}, \quad K = (\mathbf{Objekt})$$

sucht für eine Eingabe $(s, d, n) \in \mathbf{Segment} \times \mathbf{Int} \times \mathbf{Int}$ in der Kontextansicht ein Segment an der Stelle der um n-mal um die Dauer d verschobenen Eingabe s.

Der Operatorgraph in Abbildung 5.7 zeigt, wie man mit dem Gitteriterator einen Ton findet, der sich zwei Takte nach einem aktuellen Ton befindet. Der Operatorgraph soll auf die die ereignisorientierte Repräsentation des Kinderlieds „Kreis, Kreis, Kessel“ aus Abbildung 5.5 angewendet werden. Die Kontextansichten der Operatoren sind in eckigen Klammern in Abbildung 5.7 angegeben. Eingabe des Operatorgraphen ist das Segment $(4, 4; a)$

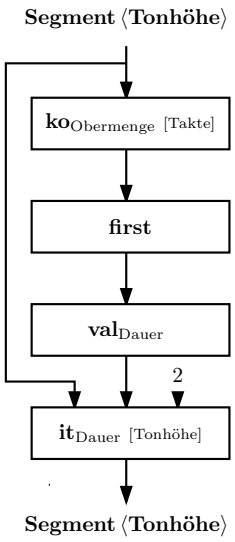


Abbildung 5.7: Iteration zur gleichen Taktposition zwei Takte später.

der Tonhöhenansicht. Zu diesem Segment werden zunächst mit einem Koinzidenzoperator[1] gleichzeitig auftretende Takte der Taktansicht gesucht. Da ein Koinzidenzoperator mehrere gleichzeitige Segmente zu einer Eingabe finden kann, wählt man mit dem Operator **first** das erste (und hier einzige) Ergebnis $(0, 8; (\mathbf{Takt}; \frac{2}{4}))$ des Koinzidenzoperators aus. Die Dauer 8 dieses Segments wird mit dem Wertextraktor[2] $\mathbf{val}_{\text{Dauer}}$ ermittelt und dient als Referenzlänge für den Gitteriterator $\mathbf{it}_{\text{Dauer}}$. Als erste Eingabe für den Gitteriterator wird das aktuelle Segment $(4, 4; a)$ der Eingabeansicht verwendet. Der Faktor n des Gitteriterators ist in diesem Beispiel fest auf 2 eingestellt. Ergebnis des Gitteritterators und des Operatorgraphen ist das Segment $(20, 2; a)$. Insgesamt wurde bei der Auswertung des Operatorgraphen nur zeitbezogene Information herangezogen.

Im Beispiel misst der Gitteritterator $\mathbf{it}_{\text{Dauer}}$ die zurückgelegte Entfernung mit Hilfe der Dauer eines anderen Ereignisses ab. Wenn man sowohl die Dauer d als auch den Faktor n des Gitteriterators fest einstellt, tastet er die Eingabeansicht in gleichmäßigen Abständen ab. Dadurch erhält man dieselbe Information wie beim Durchlaufen einer Rasterdarstellung in MeloGeNet.

In der Definition des Gitteriterators ist offen geblieben, auf welche Weise man das Zielsegment auswählt, das sich an der Stelle des verschobenen Eingabesegments befindet. Im Beispiel besteht die Strategie darin, das erste Tonhöhensegment auszuwählen, dessen Einsatzzeit im um zwei Takte verschobenen Eingabesegment liegt. Im Prinzip kann man aber hier jede Strategie zur Auswahl eines gleichzeitigen Segments anwenden. Unterschiedliche Formen der Gleichzeitigkeit werden im Abschnitt 5.2.2 über Koinzidenzoperatoren diskutiert.

[1] vgl. Abschnitt 5.2.2
[2] vgl. Abschnitt 5.2.4

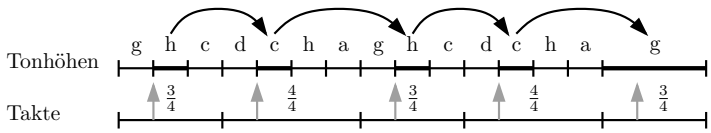


Abbildung 5.8: Iterieren mit einer Hilfsansicht

Navigation mit einer Hilfsansicht. Beim oben diskutierten Beispiel wurde die gewünschte Entfernung des Zieltons anhand der Dauer des aktuellen Takts abgemessen, die mit einem Faktor multipliziert wurde. Diese Herangehensweise liefert das erwartete Ergebnis, solange keine Taktwechsel auftreten. Die Multiplikation einer Taktdauer mit einem Faktor setzte implizit voraus, dass es sich bei der Taktansicht um ein Gitter mit einer festen Segmentlänge handelt. Wenn eine Komposition Taktwechsel enthält, ist diese Annahme nicht mehr richtig. Will man sich bei der Navigation in der Tonhöhenansicht allgemeiner auf die Taktansicht beziehen, so kann man sie als Kontextansicht in einen Iterator integrieren.

Die obige Aufgabenstellung lässt sich folgendermaßen für Kompositionen mit Taktwechseln präzisieren: Zu einem gegebenen Ton ist ein Zielton gesucht, der sich relativ zum aktuellen Takt an derselben Position befindet. Abbildung 5.8 skizziert diese Art der Fortbewegung. Für den Iterator

$$\mathbf{it}_{\text{relParallel}} : \mathbf{Segment} \longrightarrow \mathbf{Segment}, \quad K = (k_1, k_2) = (\mathbf{Objekt}, \mathbf{Objekt})$$

ordnet man die Tonhöhenansicht dem ersten und die Taktansicht dem zweiten Kontexttyp zu, d.h. $\phi(k_1) =$ „Tonhöhe“ und $\phi(k_2) =$ „Takte“. Der Iterator berechnet den Abstand zwischen der Einsatzzeit des Eingabesegments und der Einsatzzeit des zugehörigen Takts und wählt als Nachfolger das Segment, das sich relativ zum Taktanfang des nächsten Takts an der gleichen Position befindet. Der Übergang vom vierten zum fünften Takt in Abbildung 5.8 zeigt, dass auch dieser Iterator eine Strategie zur Auswahl eines gleichzeitigen Segments verwendet. In den ersten drei Iterationsschritten findet der Iterator jeweils genau ein Segment vor, das die Position und Dauer des verschobenen Eingabesegments hat. Im vierten Schritt hingegen wird deutlich, dass für den Iterator $\mathbf{it}_{\text{relParallel}}$ auch solche Segmente als gültige Sprungziele gelten, die das verschobene Eingabesegment nur enthalten. Im Gegensatz zum Gitteriterator ist $\mathbf{it}_{\text{relParallel}}$ unabhängig von der Dauer der einzelnen Takte.

Spezielle Kontexttypen. Mit der Einteilung der Operatoren in Gruppen strebt man die Trennung von zeitabhängigen und wissensabhängigen Operatoren an. Auch wenn diese Idee zur Übersichtlichkeit des Ansatzes beiträgt, kann sie nicht immer verwirklicht werden. Ein Grenzfall sind Iteratoren, die eine spezielle Kontextansicht bei der Navigation voraussetzen. Wenn man beispielsweise das metrische Gewicht von dissonanten Tönen berechnen will, benötigt man einen Iterator, der von einem dissonanten Ton zum nächsten springt. Ein solcher Iterator benötigt neben einer Tonhöhenansicht, in der der Zielton liegen soll, noch eine Harmonisierung der Melodie. Die Harmonisierung erlaubt dem Iterator zu entscheiden, ob ein Ton zur aktuellen Harmonie gehört oder dissonant ist. Daraus ergibt sich ein Iterator mit speziellen Kontexttypen:

$$\mathbf{it}_{\text{HarmonischeDissonanz}} : \mathbf{Segment} \longrightarrow \mathbf{Segment}, \quad K = (\mathbf{Tonhöhe}, \mathbf{Harmonie})$$

Der Iterator bewegt sich so lange von Ton zu Ton weiter, bis er einen Ton gefunden hat, der bezüglich der gleichzeitig auftretenden Harmonie dissonant ist. Anders in den bisher vorgestellten Beispielen kann dieser Iterator nicht auf Ansichten beliebigen Typs iterieren. Da er Tonhöhen und Harmonien in Verbindung bringt, benötigt er als Zielansicht $\phi(k_1)$ eine Ansicht vom Typ **Tonhöhe**. Die Art der zeitlichen Fortbewegung hängt bei diesem Iterator von einer musikalischen Eigenschaft der Komposition ab, vermischt also Problemwissen und zeitliche Navigation.

5.2.2 Gleichzeitigkeit

Die zweite Gruppe von Operatoren konkretisiert den musikalischen Gleichzeitigkeitsbegriff. Es gibt eine Vielzahl von Möglichkeiten, musikalische Gleichzeitigkeit zu definieren. Die engste Definition lässt zwei Ereignisse dann als gleichzeitig gelten, wenn Einsatzzeit und Dauer übereinstimmen. Wenn man musikalische Strukturen von unterschiedlichem Abstraktionsgrad miteinander in Verbindung bringen will, ist diese Definition jedoch zu restriktiv wie zwei Beispiele zeigen: Töne, aus denen sich ein musikalisches Motiv zusammensetzt, treten gleichzeitig auf wie das Motiv als übergeordnetes Ereignis ohne in Einsatzzeit und Dauer übereinzustimmen. Wenn man die Harmonie bestimmen will, die zu Beginn eines Motivs erklingt, sind nur Ereignisse zum Einsatzzeitpunkt des Motivs von Interesse; auch hier ist ein exakter Gleichzeitigkeitsbegriff nicht hilfreich. Es ist daher notwendig, verschiedene Interpretationen der musikalischen Gleichzeitigkeit zu definieren, damit man bei der Merkmalsberechnung genau ausdrücken kann, welche gleichzeitigen Ereignisse man in Beziehung zueinander setzen will.

Die engste Definition der Gleichzeitigkeit – exakte Gleichheit von Einsatzzeit und Dauer – wurde bereits genannt. Die allgemeinste Form der Gleichzeitigkeit besteht darin, alle Ereignisse als gleichzeitig zu definieren, die mindestens einen gemeinsamen Punkt auf dem Zeitgitter teilen. Zwischen diesen Polen bewegen sich die *Koinzidenzoperatoren*, die in diesem Abschnitt vorgestellt werden.

5.2.2.1 Koinzidenzoperator

Definition 5.3 (Koinzidenzoperator)
Es seien $k_1, \ldots, k_m \in \mathcal{T}$, $m > 0$, Typen einer Erweiterbaren Zeitreihengrammatik. Ein *Koinzidenzoperator* ist ein abstrakter Operator mit einer Deklaration der Form

$$\mathbf{ko}: \ \mathbf{Segment} \longrightarrow \mathbf{Vektor}\,\langle \mathbf{Segment} \rangle\,, \quad K = (k_1, \ldots, k_m).$$

Wenn der Koinzidenzoperator für ein Segment $x \in \mathbf{Segment}$ ein definiertes Ergebnis $\mathbf{ko}(x) \neq \perp$ liefert, liegen die Elemente des Ergebnisvektors in der Kontextansicht $\phi(k_1)$:

$$\forall y \in \mathbf{ko}(x): \ y \in \phi(k_1)$$

und überschneiden sich mit dem Eingabesegment s:

$$\forall\, y \in \mathbf{ko}(x): x \cap y \neq \emptyset.$$

* * *

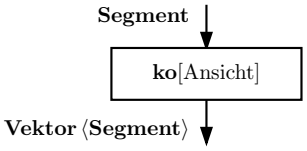


Abbildung 5.9: Graphische Darstellung eines Koinzidenzoperators mit Kontexttyp t und Kontextansicht $\phi(t) =$ „Ansicht“.

Da mehrere Segmente einer Ansicht ein Gleichzeitigkeitskriterium erfüllen können, beispielsweise weil sie sich mit dem Eingabesegment überschneiden, wurde der Wertebereich eines Koinzidenzoperators als **Vektor** ⟨**Segment**⟩ statt **Segment** gewählt. Der erste Kontexttyp k_1 wird wie beim Iterator der Zielansicht zugeordnet; die Segmente des Ergebnisvektors liegen also in dieser Ansicht, d.h. $\forall y \in \mathbf{ko}(x) : y \in \phi(k_1)$. Die zweite Bedingung $\forall\, y \in \mathbf{ko}(x) : x \cap y \neq \emptyset$ drückt die Eigenschaft aus, die alle Koinzidenzoperatoren charakterisiert und die allgemeinste Definition von Gleichzeitigkeit wiedergibt: Jedes Segment des Ergebnisvektors hat einen nicht-leeren Schnitt mit dem Eingabesegment. Abbildung 5.9 zeigt die graphische Darstellung eines Koinzidenzoperators mit Kontext $K = (t)$ und zugeordneter Kontextansicht $\phi(t) =$ „Ansicht“.

5.2.2.2 Beispiele

Bevor Beispiele für Koinzidenzoperatoren gegeben werden, stellt sich die Frage, welche Gleichzeitigkeitsbeziehungen bei der Analyse von Musikstücken auftreten. In Abbildung 5.4 wurde das betrachtete Kinderlied mit hierarchisch ineinandergeschachtelten Rastern dargestellt. Hier gibt es drei mögliche Gleichzeitigkeitsbeziehungen: Zwei Segmente sind entweder gleich, disjunkt oder ineinander enthalten.

Bei motivisch durchgestalteten Musikstücken erweitert sich das Spektrum der möglichen Gleichzeitigkeitsbeziehungen, weil Segmente einander überlappen können. Abbildung 5.10 zeigt eine Repräsentation mit überlappenden Segmenten: das Thema des Largo aus dem „Winter“ der Vier Jahreszeiten von A.Vivaldi. Die Melodie wurde auf vier verschiedene Arten (von Hand) analysiert.

Die Ansicht der rhythmischen Motive zeigt Klassen rhythmischer Muster der Melodie, wobei die Klassennummern nach absteigender Klassengröße vergeben wurden. Die Klasse 0 enthält das am häufigsten auftretende rhythmische Muster  und seine Verwandten. Während bei der rhythmischen Analyse die Kontur der Melodie außer Acht gelassen wurde, gibt die Analyse der Bogenmotive die bogenförmigen Teilstücke der Melodie wieder. Es werden aufsteigende und wieder abfallende ($\cap$) sowie abfallende und wieder aufsteigende ($\cup$) Melodieteilstücke unterschieden. Im Unterschied zu den anderen Ansichten überlappen die Segmente dieser Ansicht einander an vielen Positionen. Desweiteren wurde die grobe und feinere Phrasenstruktur der Melodie analysiert (Ansichten „Kurze Phrase“, „Lange Phrasen“).

Die Ansichten der Tonhöhen, der rhythmischen Motive sowie der kurzen und langen Phrasen sind zwar hierarchisch ineinander geschachtelt, doch ist dies nur der Fall, weil bei der

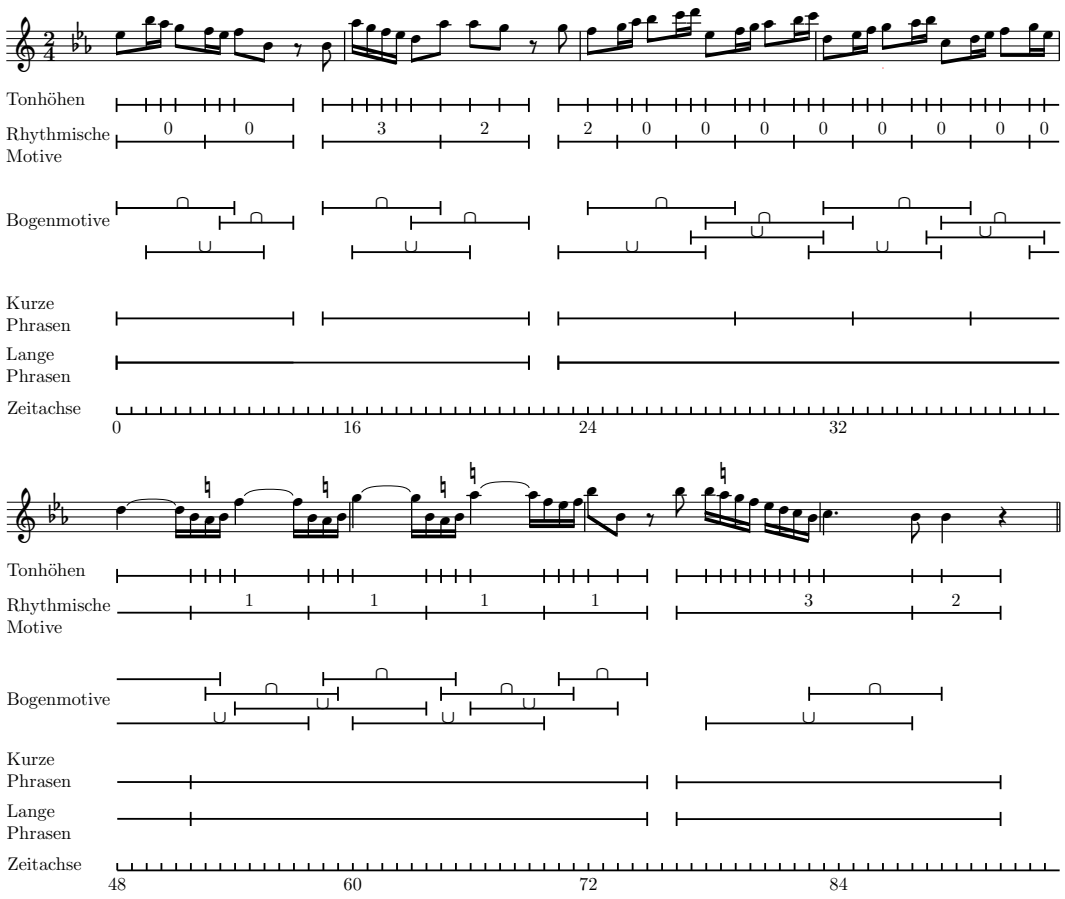


Abbildung 5.10: Vier Analysen des Largo von A. Vivaldi (op. 8, Nr.4)
Rhythmische Motivanalyse mit vier Motivklassen.
Bogenmotive: $\cap$ = aufsteigend/abfallend, $\cup$ = abfallend/aufsteigend
Lange und kurze Phrasen ohne Klassifikation

Analyse der rhythmischen Motive die Phrasengrenzen beachtet wurden. Nimmt man die Ansicht der Bogenmotive hinzu, so enthält die Repräsentation sowohl ansichtenübergreifende Überlappungen als auch Überlappungen innerhalb einer Ansicht. Insbesondere die rhythmischen Motive und die Bogenmotive zeigen konkurrierende Aspekte der Melodie, die bei der lernbasierten Melodiemodellierung von Interesse sein könnten.

Zu den drei bereits genannten Gleichzeitigkeitsbeziehungen „gleich“, „disjunkt“ und „ineinander enthalten“ bei hierarchischen Rasterdarstellungen kommt bei der Berücksichtigung beliebiger Segmentlängen die partielle Überlappung von Segmenten hinzu. Es ergeben sich viele Möglichkeiten, zu einem vorgegebenen Segment Selektionskriterien für teilweise überlappende Segmente zu definieren. Abbildung 5.11 zeigt schematisch die Funktionsweise eini-

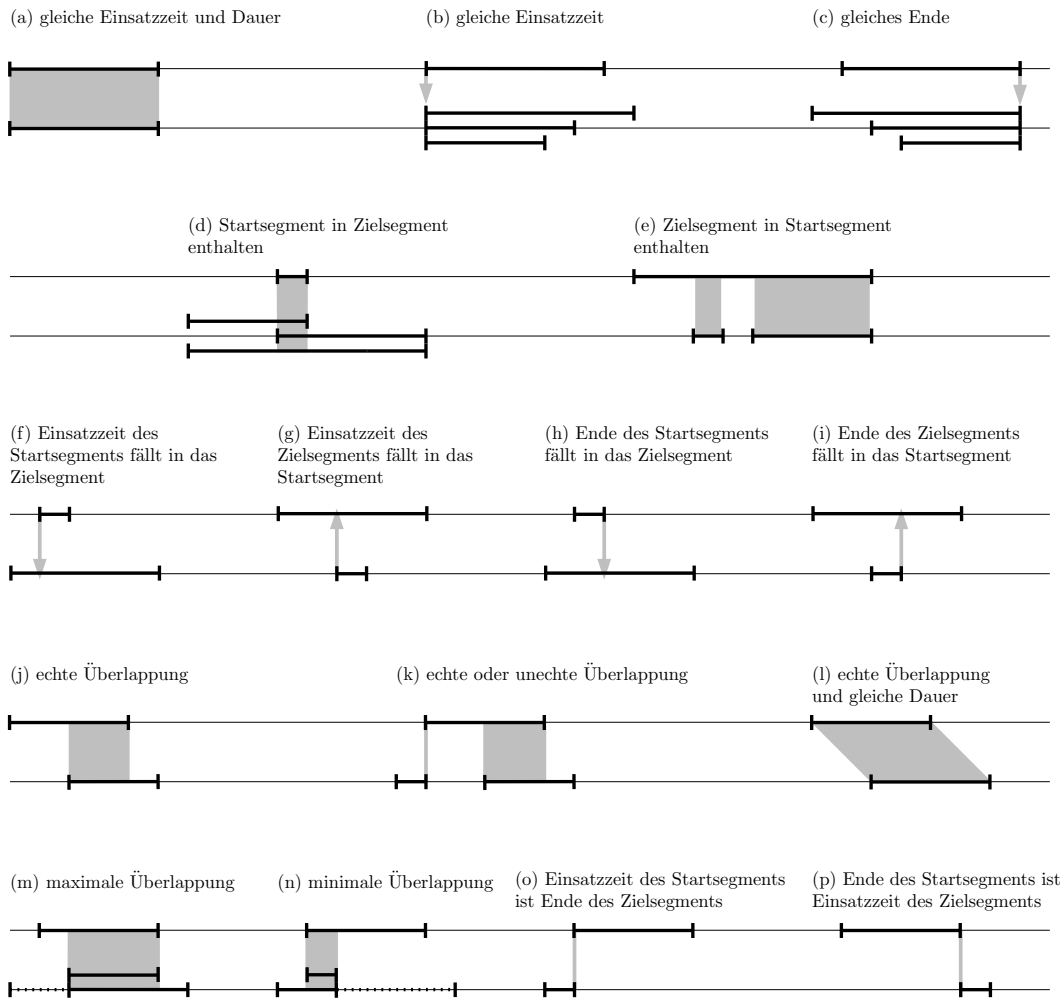


Abbildung 5.11: Beispiele für Koinzidenzoperatoren. Die obere Ansicht ist die Startansicht, die untere die Zielansicht. Gepunktete Segmente stellen nicht-ausgewählte Segmente dar.

ger Koinzidenzoperatoren, die verschiedene Selektionskriterien umsetzen. Die obere Ansicht enthält ein Startsegment, für das der jeweilige Koinzidenzoperator ein oder mehrere Zielsegmente in der unteren Ansicht findet. Einige nicht gefundene Segmente sind gepunktet eingezeichnet, um die Funktionsweise eines Koinzidenzoperators zu verdeutlichen. Zur Definition der Gleichzeitigkeit von Segmenten kommen die verschiedenen Bestandteile eines Segments in Frage: seine Einsatzzeit, die Dauer bzw. die inneren Punkte sowie sein Ende. Der einfachste Koinzidenzoperator sucht ein Segment, das mit der Eingabe identisch ist (a). Wenn man die vorletzte Note *d* der Sopranstimme im Anfang des Chorsatzes „Es ist genug“ (Abb. 5.12) vorgibt, finden die Koinzidenzoperatoren (e), (g), (i), (j) in der Altstimme ge-

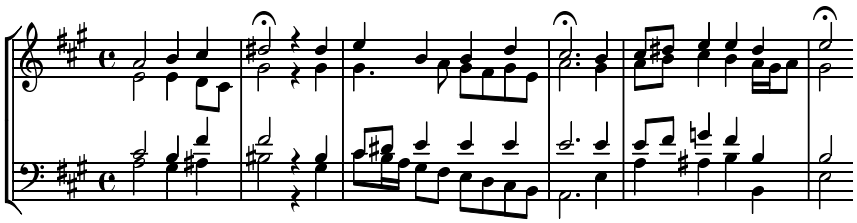


Abbildung 5.12: J.S.Bach: Choral „Es ist genug“

nau die Wechselnoten „*a gis a*“. Wählt man das zweite Viertel *h* des Soprans in Takt 3 als Eingabe, so findet nur der Koinzidenzoperator (f) genau die punktierte Viertel *gis* im Alt desselben Takts. Die Koinzidenzoperatoren (i),(j),(m) und (n) finden sowohl die punktierte Viertel *gis* als auch die darauffolgende Achtel *a*.

5.2.3 Anwendungswissen

Ein Wissensoperator modelliert Zusammenhänge innerhalb eines Anwendungsgebiets ohne sich direkt auf eine Komposition zu beziehen. Ein Beispiel für einen Wissensoperator ist ein Operator, der aus zwei Tonhöhen ein Intervall berechnet. Für die Berechnung des Intervalls ist es unwichtig, an welcher Stelle einer Komposition die Tonhöhen auftreten oder ob sie überhaupt Teil einer Komposition sind.

Wissensoperatoren sind die einzigen Operatoren, die explizit musikalisches Wissen verwenden. Die Motivation für die Einführung von Wissensoperatoren besteht darin, dass man das Anwendungswissen in einem klar definierten Bereich des Modells kapseln möchte, um es bei veränderter Aufgabenstellung erweitern oder durch das Wissen eines anderen Problembereichs austauschen zu können. Auch wenn hier der Anschaulichkeit halber immer von Kompositionen und musikalischem Wissen die Rede ist, ist der hier entwickelte Ansatz nicht auf musikalische Anwendungen beschränkt. Allerdings ist mir kein Anwendungsgebiet bekannt, das eine so aufwendige Behandlung der zeitlichen Darstellung erfordert wie die Musik.

Wie bei den Iteratoren und den Koinzidenzoperatoren legt auch hier ein Prototyp die allgemeine Form eines Wissensoperators fest. Neben Wissensoperatoren, die musikalisches Wissen zu einem abstrakteren oder komplexeren Objekt umformen, fallen auch die numerische Codierung von Objekten und die Rücktransformation von Zahlenmustern in Objekte (Decodierung) in die Gruppe der Wissensoperatoren, denn sie setzen Information über den Aufbau der Objekte voraus. Ein spezieller Wissensoperator ist der Neuro-Operator, der es ermöglicht, ein neuronales Netz innerhalb eines Operatorgraphen auszuwerten. Das neuronale Netz stellt dabei ein Modell für Anwendungswissen dar, das mit Hilfe eines Lernverfahrens aus musikalischen Beispielen gewonnen wurde.

5.2.3.1 Wissensoperator

Definition 5.4 (Wissensoperator)
Es seien $t_1, \ldots, t_r, t \in \mathcal{T}$, $r > 0$, Typen einer Erweiterbaren Zeitreihengrammatik. Ein *Wissensoperator* ist ein abstrakter Operator mit einer Deklaration der Form

$$\mathbf{dom}: \; t_1 \times \ldots \times t_r \longrightarrow t, \;\; K = ().$$

* * *

Ein Wissensoperator benötigt keinen Kontext, da sein Verhalten nicht von der Anordnung der Elemente einer Komposition abhängt. Der Name **dom** leitet sich vom englischen Begriff *domain* = Anwendungsgebiet ab.

5.2.3.2 Beispiele für allgemeine Wissensoperatoren

Beispiele. Einfache Beispiele für einen Wissensoperator sind der Intervalloperator

$$\mathbf{dom}_{\text{Intervall}}: \; \mathbf{Tonhöhe} \times \mathbf{Tonhöhe} \longrightarrow \mathbf{Intervall},$$

der das Intervall zwischen zwei Tonhöhen ermittelt, der Konturoperator

$$\mathbf{dom}_{\text{Kontur}}: \; \mathbf{Tonhöhe} \times \mathbf{Tonhöhe} \longrightarrow \mathbf{Kontur},$$

der die Richtung zwischen zwei Tonhöhen bestimmt, und der Beschleunigungsoperator

$$\mathbf{dom}_{\text{Dauernverhältnis}}: \; \mathbf{Int} \times \mathbf{Int} \longrightarrow \mathbf{Rational},$$

der das Verhältnis zweier Dauern berechnen. Die drei Operatoren setzen jeweils zwei Ereignisse zueinander in Beziehung, deren zeitliche Position in der Komposition für das Ergebnis des Operators irrelevant ist.

Metrisches Gewicht. Mit jeder Taktart wird in der Musiktheorie ein bestimmtes Betonungsschema verbunden. Das *metrische Gewicht* ist eine Zahl, die die relative Bedeutung eines Zeitpunkts in diesem Schema angibt. In einem $\frac{3}{4} - Takt$ ist z.B. die erste Zählzeit stärker betont als die zweite und dritte. Um das metrische Gewicht eines Ereignisses zu berechnen, benötigt man daher die Einsatzzeit des Ereignisses sowie die Lage und die Art des Takts, der als Referenz dienen soll:

$$\mathbf{dom}_{\text{MetrischesGewicht}}: \; \mathbf{Int} \times \mathbf{Segment}\,\langle \mathbf{Takt} \rangle \longrightarrow \mathbf{Int}$$

Als Wissensoperator kennt $\mathbf{dom}_{\text{MetrischesGewicht}}$ die Betonungsschemata, die den einzelnen Taktarten zugeordnet sind und setzt die Einsatzzeit eines Ereignisses zum passenden Schema in Relation.

Beide Eingaben des Operators $\mathbf{dom}_{\text{MetrischesGewicht}}$ stammen im allgemeinen aus einer musikalischen Komposition. Dennoch ist der Operator nicht von der Existenz der Komposition oder einer bestimmten Ansicht abhängig, weil er sein Ergebnis unabhängig von der

tatsächlichen Herkunft der Eingabeobjekte berechnen kann. Zeitabhängige Operatoren wie Iteratoren oder Koinzidenzoperatoren übernehmen die Aufgabe, geeignete Eingaben für den Operator $\mathbf{dom}_{\text{MetrischesGewicht}}$ aus der Komposition zu extrahieren. In Abschnitt 5.3 wird ein musikalisches Beispiel für die Berechnung des metrischen Gewichts diskutiert.

Phrasenform. Der Operator zur Bestimmung der Phrasenform ist ein Beispiel für einen Klassifikator:

$$\mathbf{dom}_{\text{Phrasenform}} : \mathbf{Vektor}\,\langle\mathbf{Tonhöhe}\rangle \longrightarrow \mathbf{Phrasenform}$$

Er ordnet einer Folge von Tonhöhen eine von mehreren vorgegebenen Charakterisierungen zu (z.B. aufsteigend, absteigend, gleichbleibend,...), die durch die Definition des Typs **Phrasenform** im benutzerdefinierten Teil der Erweiterbaren Zeitreihengrammatik festgelegt sind.

Eine Besonderheit des Operators $\mathbf{dom}_{\text{Phrasenform}}$ besteht darin, dass er eine variable Anzahl von Tonhöhen als Eingabe erhält. Die Eingabe des Operators ist daher ein Objekt vom Typ $\mathbf{Vektor}\,\langle\mathbf{Tonhöhe}\rangle$.

Die Phrasenform ist ein Beispiel für ein Merkmal, das auf viele Weisen definiert werden kann. Während hier nur eine Abfolge von Tonhöhen als Eingabe in den Operator eingeht, könnte man z.B. auch die Dauern und die metrischen Positionen der Töne berücksichtigen.

5.2.3.3 Codierungs- und Decodierungsoperatoren

Eine Anwendung von Operatorgraphen ist die Berechnung von Lernmustern für das Training neuronaler Netze. Da Lernmuster, die ja die Ein- und Ausgabe eines neuronalen Netzes bilden, numerische Vektoren sind, müssen die symbolischen Objekte der Repräsentation bei der Lernmusterberechnung in eine numerische Form gebracht werden. Dies ist die Aufgabe der Codierungsoperatoren. Die Codierungsoperatoren berücksichtigen die Struktur der codierten Objekte und gehören daher zu den Wissensoperatoren.

Wenn neuronale Netze zur Komposition von Musikstücken eingesetzt werden, benötigt man neben der Codierung der Netzeingabe auch eine Decodierung der numerischen Netzausgabe. Ein Decodierungsoperator konstruiert aus einem numerischen Vektor, beispielsweise der Ausgabe eines neuronalen Netzes, ein symbolisches Objekt, das in die symbolische Repräsentation eines Musikstücks eingefügt werden kann. Er muss den Typ des zu erzeugenden Objekts kennen und Information darüber besitzen, welche Werte des Typs in der numerischen Darstellung unterscheidbar sind. Für die Attribute des Typs, die bei der Codierung als irrelevant weggelassen wurden, setzt ein Decodierungsoperator Standardwerte.

Für natürliche Zahlen $n, m, k \in \mathbb{N}$ und Typen $t_1, \ldots, t_r, t \in \mathcal{T}$ werden der *Codierungsoperator* und der *Decodierungsoperator* folgendermaßen deklariert:

$$\mathbf{code} : t_1 \times \ldots \times t_r \longrightarrow \mathbf{Rational}^n$$

$$\mathbf{decode} : \mathbf{Rational}^m \longrightarrow t$$

Der Codierungsoperator berechnet also ein numerisches Muster der Breite n; der Decodierungsoperator erwartet m rationale Zahlen als Eingabe. Jeder vollständig definierte Codierungs- und Decodierungsoperator lässt sich auf diese Deklarationen zurückführen.

Codierung musikalischer Objekte für das maschinelle Lernen. Bei der Mustererzeugung werden Ansichten codiert, indem ihre Segmente auf numerische Vektoren abgebildet werden. Es hängt von der Aufgabenstellung ab, welche Bestandteile eines Objekts man in eine Codierung einfließen lässt. Beispielsweise kann man Einsatzzeit, Dauer oder Wert eines Segments in das erzeugte Muster aufnehmen. Allein für die Codierung der Tonhöhe kommen z.B. die 1-aus-n-Codierung, die harmonische Toncodierung, die Codierung mit Obertönen und die Codierung als Frequenz in Frage.

Wenn man Muster für das maschinelle Lernen entwirft, sucht man nach einer kompakten Codierung, die die musikalischen Objekte in einen möglichst niedrigdimensionalen Raum abbildet, aber dem Lernmodell gleichzeitig die für die Lösung der Lernaufgabe wesentlichen Informationen zur Verfügung stellt. Eine Codierung mit einem Vektor kleiner Dimension ist wünschenswert, weil sie der Gefahr des Overfittings entgegenwirkt und daher zu einer verbesserten Leistung des trainierten Modells führen kann. Auch wenn man nicht im vorhinein entscheiden kann, welche Kombination von Merkmal und Codierung zum Lernerfolg führt, so kann man doch einige allgemeine Überlegungen anstellen, welche Codierungen sich zum maschinellen Lernen eignen.

Bei der hier verwendeten Repräsentation ist der zu codierende Wertebereich im allgemeinen endlich und diskret. Jeder Wert ist ein Objekt, das eine hierarchische Struktur besitzt und zu anderen Werten in Relation stehen kann (z.B. durch eine Ordnungsrelation). Wenn man einen solchen strukturierten Wertebereich in einen Vektorraum einbettet, sind aus Sicht eines neuronalen Netzes nur noch die Abstände zwischen den codierten Werten sichtbar. Man muss daher die potentiell relevante Struktur des Wertebereichs dadurch ausdrücken, dass man die Werte in geeigneten Abständen codiert.

Ein Merkmal stellt zusammen mit seiner Codierung eine Hypothese darüber dar, welcher Aspekt eines Musikstils zur Lösung einer Lernaufgabe relevant sein könnte. Wenn zwei Merkmalswerte in der Codierung unterscheidbar sind, drückt man damit die Vermutung aus, dass dieser Unterschied dem trainierten Modell hilft, zu einem Eingabevektor eine korrekte Antwort zu erzeugen. Teilinformationen, die man als irrelevant einstuft, werden nicht codiert. Wenn man einen Musikstil anhand des Notentexts modellieren will, wird man etwa benutzerdefinierte Tempoänderungen in MIDI-Dateien als ungeeignetes Merkmal verwerfen, weil die Tempoänderungen nicht vom Notentext gestützt werden. Ein anderer Grund, Teile eines Merkmals nicht zu codieren, ist der Wunsch nach einem kleineren Modell, um den „Fluch der Dimensionalität“ abzumildern. Will man z.B. Tonhöhen codieren, so muss man abwägen, ob die Oktave des Tons und seine Lautstärke bei der Codierung berücksichtigt werden sollten, oder ob die Codierung der Tonhöhenklasse möglicherweise ausreicht.

Eine konstante Codierung, bei der alle Werte eines Wertebereichs gleich codiert werden, ist für das Lernmodell nutzlos und kann weggelassen werden. Auch wenn die Codierung zwar nicht konstant ist, aber das Merkmal für alle verfügbaren Daten mit demselben Punkt codiert wird, trägt es nicht zur Lösung der Lernaufgabe bei.

Filtern von Wertebereichen. Häufig besteht ein Typ, selbst wenn er wie z.B. der Typ **Takt** im benutzerdefinierten Teil der Erweiterbaren Zeitreihengrammatik in Kapitel 4 durch Bedingungen auf musikalisch sinnvolle Werte eingeschränkt wurde, aus vielen Elementen, von denen nur ein Teil in den Lernbeispielen vorkommt. Da unterschiedliche Taktarten

Harmonische Funktion	Töne (C-Dur)	Codierung
T	c e g	1 0 0 0 0 0 0 0 0 0 0 0
D	g h d f	0 1 0 0 0 0 0 0 0 0 0 0
S	f a c	0 0 1 0 0 0 0 0 0 0 0 0
Tp	a c e	0 0 0 1 0 0 0 0 0 0 0 0
Sp	d f a	0 0 0 0 1 0 0 0 0 0 0 0
Dp	e g h	0 0 0 0 0 1 0 0 0 0 0 0
DD	d fis a c	0 0 0 0 0 0 1 0 0 0 0 0
DP	e gis h d	0 0 0 0 0 0 0 1 0 0 0 0
TP	a cis e g	0 0 0 0 0 0 0 0 1 0 0 0
d	g b d	0 0 0 0 0 0 0 0 0 1 0 0
VTp	gis h d f	0 0 0 0 0 0 0 0 0 0 1 0
SS	b d f	0 0 0 0 0 0 0 0 0 0 0 1

Tabelle 5.1: Codierung harmonischer Funktionen als nominales Merkmal

nicht in einem inneren Zusammenhang stehen, sondern lediglich für ein Betonungsschema stehen, ist es hier sinnvoll, nur die tatsächlich vorkommenden Taktarten zu codieren. Vor der eigentlichen Codierung werden nicht benötigte Werte herausgefiltert.

Es kann vorkommen, dass einige Werte einer zusammengehörigen Menge wie z.B. Töne einer Skala in einer Menge von Lernbeispielen nicht vorkommen. Eine mögliche Erklärung besteht darin, dass man für den Stil, der modelliert werden soll, die falsche Skala zugrundegelegt hat. Dieser Fall könnte etwa bei einem Stil auftreten, der wie die indonesische Gamelanmusik oder die traditionelle Musik aus China eine pentatonische Skala verwendet. Hier muss die Skala zunächst an den untersuchten Stil angepasst werden, wobei eine einfache Filterung nicht auftretender Werte in den genannten Beispielen nicht ausreicht: Die pentatonische *slendro*-Skala in der Gamelanmusik teilt eine Oktave in fünf Intervalle annähernd gleicher Größe; das Fehlen von Tönen der chromatischen Skala hat hier also eine systematische Ursache. Chinesische Volksmusik besitzt zwar eine pentatonische Basis, ist jedoch nicht vollständig pentatonisch. In diesen Beispielen ist es ratsam, die musikalischen Tonhöhentypen für die ganze Modellierung zu überarbeiten.

Vergleichbarkeit von Codierungen. Wenn lediglich voneinander unabhängige Werte wie Taktarten ausgefiltert werden müssen, muss man abwägen, ob die Verkleinerung der Codierungen oder die Vergleichbarkeit der verwendeten Codierungen eine höhere Priorität hat. Lässt man in einer Codierung einen Wert weg, der in einer Trainingsmenge eines anderen Modells vorkommt, so lassen sich die Merkmale beider Modelle nicht miteinander vergleichen.

Nun werden Beispiele für Codierungen musikalischer Objekte vorgestellt, die die Eigenschaften der codierten Wertebereiche berücksichtigen.

Nominale Wertebereiche. Ein Merkmal mit einem *nominalen* Wertebereich beschreibt eine symbolische Eigenschaft (rot, grün, blau) anhand von Werten, die nicht miteinander

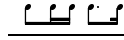
Frequenz	**Codierung als Zahl**
a'	440
a''	880
Rhythmisches Muster	**Gittercodierung in einem Zeitfenster fester Breite [54]**
	1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 1 1 1 0 0 0
Rhythmischer Wert	**Additive Codierung [40]**
	0 0 1 0 0 0 0 0
	0 0 0 1 0 0 0 0
	0 0 0 0 1 0 0 0
	0 0 0 1 1 0 0 0
	0 0 0 1 1 1 0 0
	0 0 1 0 1 0 0 0

Tabelle 5.2: Beispiele für die Codierung ordinaler Merkmale

verknüpft sind, sondern nur eine Zuordnung zu einer Klasse darstellen. Die Form einer Phrase (aufsteigend, abfallend, konkav, konvex, ...) ist ein Beispiel für ein nominales Merkmal. Harmonische Funktionen (Tonika, Dominante, Subdominante, ...) können ebenfalls als nominales Merkmals aufgefasst werden. Bei nominalen Merkmalswerten kann man nur feststellen, ob zwei Merkmalswerte gleich sind. Es ist daher sinnvoll, alle Merkmalswerte äquidistant zu codieren. Hier bietet sich die 1-aus-n Codierung an, den i-ten von $n \in \mathbb{N}$ Werten durch einen binären Vektor codiert, der eine 1 an i-ter Position enthält, während alle anderen Einträge 0 sind (Tabelle 5.1). Die 1-aus-n-Codierung wird häufig zur Codierung einer Klassenzugehörigkeit verwendet.

Ordinale Wertebereiche. Ein Merkmal mit einem *ordinalen* Merkmal hat angeordnete Merkmalswerte, die diskret oder stetig sein können. Ein stetiges, ordinales Merkmal ist z.B. die Frequenz eines Tons. Ein diskretes, ordinales Merkmal ist der rhythmische Wert einer Note (Viertel, Achtel, ...). Wenn die Anordnung eines nominalen Merkmals für das Modell sichtbar sein soll, bietet sich die Codierung als Zahl an. Der Wertebereich des Merkmals muss dabei nicht beschränkt werden, so dass man dann z.B. jede hörbare Frequenz darstellen kann (Tabelle 5.2, oben). Verknüpft man mehrere Merkmale zu einem Eingabemuster, sollte man aber darauf achten, dass alle Teilmuster ähnlich skaliert werden. Im Prinzip besitzt ein neuronales Netz zwar die Fähigkeit, z.B. Größenangaben unterschiedlicher Größenordnungen intern zu skalieren, doch ist zu erwarten, dass ein Netz, das zur Verwendung extremer Gewichte gezwungen wird, sich nicht robust verhält. Eine andere Möglichkeit zur Codierung ordinaler Merkmale ist eine Gittercodierung, wie Linster [54] sie zur Analyse rhythmischer Muster mit neuronalen Netzen einsetzt. Ein rhythmisches Muster einer festen Dauer wird dabei durch einen binären Vektor fester Breite codiert, der den Beginn eines rhythmischen Werts mit einer Eins und seine Fortsetzung mit Nullen beschreibt (Tabelle 5.2, Mitte). Eine additive Codierung rhythmischer Werte schlägt Huron [40] vor (Tabelle 5.2, unten).

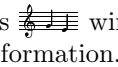
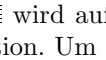
Zirkuläre Wertebereiche. Ein Beispiel für zirkulär angeordnete Elemente sind die Tonarten des Quintenzirkels. Wählt man eine ordinale Codierung der Tonarten, so wird kein

Tonart	**Rotierende Codierung [61]**
C	-1 -1 -1 1 1 1 1 1 1 -1 -1 -1
G	-1 -1 -1 -1 1 1 1 1 1 1 -1 -1
D	-1 -1 -1 -1 -1 1 1 1 1 1 1 -1
A	-1 -1 -1 -1 -1 -1 1 1 1 1 1 1
E	1 -1 -1 -1 -1 -1 -1 1 1 1 1 1
H	1 1 -1 -1 -1 -1 -1 -1 1 1 1 1
Fis/Ges	1 1 1 -1 -1 -1 -1 -1 -1 1 1 1
Des	1 1 1 1 -1 -1 -1 -1 -1 -1 1 1
As	1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1
Es	1 1 1 1 1 1 -1 -1 -1 -1 -1 -1
B	-1 1 1 1 1 1 1 -1 -1 -1 -1 -1
F	-1 -1 1 1 1 1 1 1 -1 -1 -1 -1

Tabelle 5.3: Rotierende Codierung des Quintenzirkels

Zusammenhang zwischen dem minimalen und maximalen Wert der Codierung hergestellt. Um die Zirkularität der Tonarten auszudrücken, kann man eine rotierende Codierung einsetzen, bei der ein Zahlenfeld einen Vektor entlangwandert (Tabelle 5.3 unten, nach Mozer [61]). Auf diese Weise schließt sich das erste Element eines geordneten Wertebereichs wieder an das letzte an. Der euklidische Abstand zwischen je zwei der Vektoren widerspiegelt den Abstand der Tonarten im Quintenzirkel. Eine Alternative zum rotierenden Zahlenfeld besteht darin, die zirkulär angeordneten Elemente als äquidistante Punkte auf dem Rand eines zweidimensionalen Kreises zu codieren. Diese Codierung spiegelt ebenfalls die Nachbarschaftsverhältnisse des Quintenzirkels wider. Mozer verwirft die zweite Codierung jedoch mit der Begründung, dass die sigmoiden Aktivierungsfunktionen, die üblicherweise in neuronalen Netzen verwendet werden, die Abstände zwischen den zweidimensionalen Punkten verfälschen. Da die sigmoide Aktivierungsfunktion in der Nähe von Null nahezu linear ist, lässt sich die von Mozer angesprochene Verzerrung der Abstände durch eine geeignete Skalierung der Werte gering halten, so dass die Frage bestehen bleibt, ob eine niedrigerdimensionale Codierung mit ungefähr gleichen Abstandseigenschaften wie das rotierende Zahlenfeld nicht besser geeignet sein könnte, Overfitting zu vermeiden.

Umkehrbare Wertebereiche. Alterationen sind ein Beispiel für einen symmetrischen und daher umkehrbaren Wertebereich. Die Vorzeichen $\{$𝄫, ♭, ♮, ♯, 𝄪$\}$ können als $\{-2, -1, 0, 1, 2\}$ um den „Nullpunkt“ ♮ herum codiert werden.

Komplementäre Intervallcodierung. Eine Codierung, die auf eine andere Weise die Umkehrbarkeit eines Wertebereichs berücksichtigt, ist die komplementäre Intervallcodierung im neuronalen Melodieumspielungssystem MELONET [36]. Zwei Intervalle, die sich zu einer Oktave ergänzen, können ineinander überführt werden, indem ein Ton des ersten Intervalls um eine Oktave transponiert wird: Aus  wird auf diese Weise  . Beide Intervalle enthalten eine ähnliche harmonische Information. Um melodische Motive ähnlich zu codieren, die sich harmonisch ähneln, codiert das System MELONET Intervalle ähnlich, die zu

		Komplementäre Intervallcodierung [36]		
Intervall		**Richtung**	**Oktave**	**Intervallgröße**
None	↘	1 0 0	1	0 0 0 0 0 0 1
Oktave	↘	1 0 0	1	1 0 0 0 0 0 0
Septime	↘	1 0 0	0	0 1 0 0 0 0 0
Sexte	↘	1 0 0	0	0 0 1 0 0 0 0
Quinte	↘	1 0 0	0	0 0 0 1 0 0 0
Quarte	↘	1 0 0	0	0 0 0 0 1 0 0
Terz	↘	1 0 0	0	0 0 0 0 0 1 0
Sekunde	↘	1 0 0	0	0 0 0 0 0 0 1
Prim	→	0 1 0	0	1 0 0 0 0 0 0
Sekunde	↗	0 0 1	0	0 1 0 0 0 0 0
Terz	↗	0 0 1	0	0 0 1 0 0 0 0
Quarte	↗	0 0 1	0	0 0 0 1 0 0 0
Quinte	↗	0 0 1	0	0 0 0 0 1 0 0
Sexte	↗	0 0 1	0	0 0 0 0 0 1 0
Septime	↗	0 0 1	0	0 0 0 0 0 0 1
Oktave	↗	0 0 1	1	1 0 0 0 0 0 0
None	↗	0 0 1	1	0 1 0 0 0 0 0

Tabelle 5.4: Komplementäre Intervallcodierung

einem vorgegebenen gleichen Stammton führen. Eine Terz aufwärts und eine Sexte abwärts werden beide mit 0010000 dargestellt; die Codierung der Intervallrichtung ist jedoch für die beiden Intervalle unterschiedlich (Tabelle 5.4).

Beziehungscodierung. Eine weitere Möglichkeit besteht darin, zwei Merkmale so zu codieren, dass sie aufeinander Bezug nehmen. Wenn zwischen den (endlichen) Wertebereichen beider Merkmale eine Relation besteht, codiert man das erste Merkmal mit einer 1-aus-n-Codierung. Die Werte des zweiten Merkmals werden durch einen binären Vektor codiert, der für jeden Wert des ersten Merkmals angibt, ob die Relation erfüllt ist.

Die harmonische Toncodierung im neuronalen Choralharmonisierungssystem HARMONET ist ein Beispiel für eine Beziehungscodierung. Als Referenzmerkmal dient die harmonische Funktion, die mit einer 1-aus-n-Codierung dargestellt wird (Tabelle 5.1). Ihr Wertebereich umfasst die zwölf harmonischen Funktionen, die in den Choralharmonisierungen von J.S.Bach am häufigsten auftreten.

Ist die Grundtonart bekannt, dann stehen Tonhöhen und harmonische Funktionen in einer Elementrelation: In der Grundtonart C-Dur kommt der Ton c beispielsweise in der Tonika (C-Dur), der Subdominante (F-Dur), der Tonikaparallele (a-moll) und der Doppeldominante mit Septime (D^7) vor. Der Ton c wird in der harmonischen Toncodierung durch den Vektor

101100100000

Tonhöhe	**Harmonische Toncodierung [37]**
c	1 0 1 1 0 0 1 0 0 0 0 0
cis/des	0 0 0 0 0 0 0 0 1 0 0 0
d	0 1 0 0 1 0 1 1 0 1 1 1
dis/es	0 0 0 0 0 0 0 0 0 0 0 0
e	1 0 0 1 0 1 0 1 1 0 0 0
f	0 1 1 0 1 0 0 0 0 0 1 1
fis/ges	0 0 0 0 0 0 1 0 0 0 0 0
g	1 1 0 0 0 1 0 0 1 1 0 0
gis/as	0 0 0 0 0 0 0 1 0 0 1 0
a	0 0 1 1 1 0 1 0 1 0 0 0
b	0 0 0 0 0 0 0 0 0 1 0 1
h	0 1 0 0 0 1 0 1 0 0 1 0

Tabelle 5.5: Harmonische Tonhöhencodierung: Beziehungscodierung der Tonhöhe mit den harmonischen Funktionen aus Tabelle 5.1 als Referenzmerkmal (bzgl. C-Dur).

dargestellt, dessen Einträge anzeigen, ob der zu codierende Ton in der entsprechenden harmonischen Funktion aus Tabelle 5.1 enthalten ist. Tabelle 5.5 gibt einen Überblick über die harmonische Toncodierung. Mit Hilfe der harmonischen Toncodierung lernt ein neuronales Netz, welche Töne und harmonischen Funktionen gleichzeitig auftreten, ohne Information über den Aufbau der harmonischen Funktionen zu benötigen.

Gewichtende Codierung. Eine gewichtende Codierung kombiniert mehrere Merkmalswerte, indem die Codierungen der einzelnen Merkmalswerte gewichtet summiert werden. Das Echtzeit-Harmonisierungssystem HarmoTheater [93] entscheidet zum Beispiel, ob es in eine andere Tonart modulieren soll, indem es die vom Benutzer gespielten Töne gewichtet summiert. Dabei wird ein Ton umso weniger gewichtet, je weiter er in der Vergangenheit liegt. Das System moduliert in die Tonart, die den so codierten tonalen Kontext am besten erklärt. In [27] wird ebenfalls ein sich abschwächender Kontext eingesetzt, um typische musikalische Muster in frühen Klavierkompositionen von W.A.Mozart zu suchen.

Eine andere Form einer gewichteten Codierung simuliert Aufmerksamkeitsebenen beim Hören von Musikstücken, indem musikalische Ereignisse proportional zu ihrer metrischen Bedeutung gewichtet werden [27]. Eine Dissonanz an einer betonten metrischen Position hat in dieser Codierung ein größeres Gewicht als wenn sie in Form einer Durchgangsdissonanz auf einer leichten Zählzeit auftritt.

Decodierung der Netzausgabe. Auch wenn ein neuronales Netz mit binären Mustern trainiert wurde, sind die Aktivierungen seiner Ausgabeneuronen im allgemeinen keine ganzen Zahlen. Die Ausgabe eines neuronalen Netzes muss daher einem der zulässigen Ausgabemuster zugeordnet werden. Die am weitesten verbreitete Strategie besteht darin, den Ausgabevektor deterministisch zu interpretieren. Dabei wird das musikalische Objekt ausgewählt, dessen Codierung den kleinsten Abstand zum Ausgabevektor besitzt. Im Fall von 1-aus-n-codierten Klassen ist diese Strategie gleichbedeutend damit, das Ausgabeneuron mit

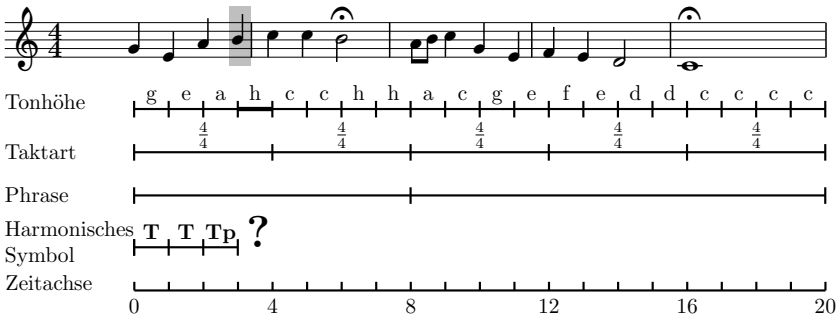


Abbildung 5.13: Choralharmonisierung mit HARMONET

maximaler Aktivierung zu suchen und die Ausgabe als Entscheidung für die entsprechende Klasse zu interpretieren. Die Decodierung kann als „Quantisierung“ der Netzausgabe bezüglich der zulässigen Ausgabemuster verstanden werden.

Eine weitere Möglichkeit besteht darin, die Ausgabe probabilistisch zu decodieren. Im neuronalen Echtzeit-Harmonisierungssystem HarmoTheater [93] wird die Ausgabe der Netze zur Bestimmung der harmonischen Funktion folgendermaßen interpretiert: Da man davon ausgeht, dass die drei Neuronen mit den höchsten Aktivierungen alle eine „sinnvolle“ Harmonisierung anzeigen, wird eine harmonische Funktion mit einer Wahrscheinlichkeit ausgewählt, die proportional zur Aktivierung der drei Ausgabeneuronen ist. Auf diese Weise können bei gleichem musikalischen Kontext unterschiedliche harmonische Fortsetzungen generiert werden. Ist die Aktivierung des zweiten und dritten Neurons sehr viel schwächer, so ist dies ein Hinweis, dass es in der gegebenen Situation nur eine plausible Harmonisierung aus den Lernbeispielen gibt. In einer solchen Situation wird mit großer Wahrscheinlichkeit die harmonische Funktion gewählt, die dem Neuron mit der höchsten Aktivierung entspricht; nur selten entscheidet sich das System für die beiden anderen fehlerhaften Harmonien.

5.2.3.4 Neurooperatoren

In den bisherigen Abschnitten war die Motivation für die Definition unterschiedlicher Operatoren, die Berechnung von musikalischen Merkmalen und von Lernmustern aus modularen Funktionseinheiten zusammenzusetzen. Wenn man nun mit Hilfe von Merkmalsselektionsverfahren neuronale Netze mit optimierten Eingabemerkmalen entwickelt hat, ergibt sich daraus die Frage, wie man diese Netze zur Komposition neuer Musikstücke einsetzen kann.

Angenommen wir haben ein Rahmenverfahren, das eine Strategie zur Komposition von Musikstücken mit Hilfe neuronaler Netze implementiert (Abschnitt 6.2 wird darauf näher eingehen). Die erzeugten Stücke werden wie bisher mit Hilfe von Ansichten dargestellt. Konkret stehen wir nun vor der Aufgabe, ein teilweise erzeugtes Musikstück mit Hilfe der optimierten Netze zu erweitern. Dazu müssen die erforderlichen Informationen aus der Repräsentation des unvollständigen Stücks extrahiert und als geeignete Muster codiert werden. Mit den

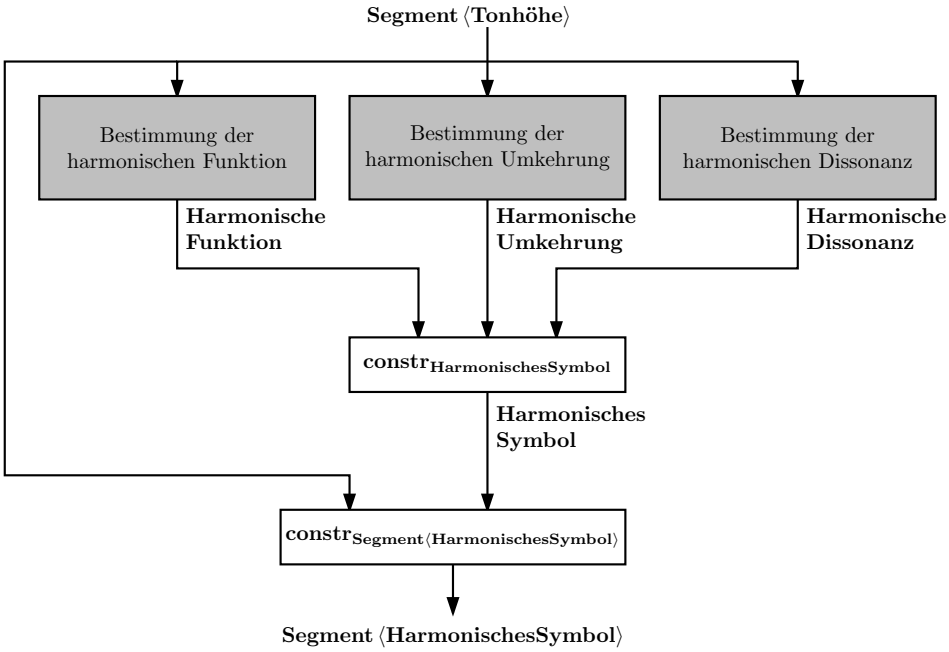


Abbildung 5.14: Aufruf der verschiedenen neuronalen Netze in HARMONET. Die Bestimmung der harmonischen Funktion ist in Abbildung 5.15 dargestellt.

Mustern werden die optimierten neuronalen Netze ausgewertet. Schließlich werden die Netzausgaben in musikalische Symbole zurückverwandelt und in das neue Stück eingefügt.

Die Codierung der musikalischen Symbole mit Codierungsoperatoren und die Rücktransformation von Vektoren in musikalische Objekte mit Decodierungsoperatoren wurde im letzten Abschnitt bereits diskutiert. Nun fehlt noch ein Operator, der ein neuronales Netz auswertet. Dies ist die Aufgabe eines *Neurooperators*, der folgendermaßen deklariert wird:

$$\textbf{neuro}: \textbf{Rational}^{n_1} \times \ldots \times \textbf{Rational}^{n_k} \longrightarrow \textbf{Rational}^{m}$$

für $n_1, \ldots, n_k, k, m \in \mathbb{N}$.

Beispiele. Wenn man die Systeme HARMONET und MELONET mit Operatorgraphen simuliert, erhält man Beispiele für den Aufruf von Neuro-Operatoren. Die Berechnung eines harmonischen Symbols in HARMONET ist schematisch in Abbildung 5.14 dargestellt. Das Modul „Bestimmung der harmonischen Funktion“ ist in Abbildung 5.15 genauer dargestellt und enthält den Aufruf eines neuronalen Netzes mit einem Neuro-Operator. In der graphischen Darstellung des Neuro-Operators ist – im Unterschied zu den anderen Operatoren – in den eckigen Klammern nicht der Name der Kontextansicht, sondern der Name des aufgerufenen Netzes angegeben. Für das Melodieumspielungssystem MELONET zeigt Abbildung 5.16 die Repräsentation einer Melodie während des Umspielens; Abbildung 5.17

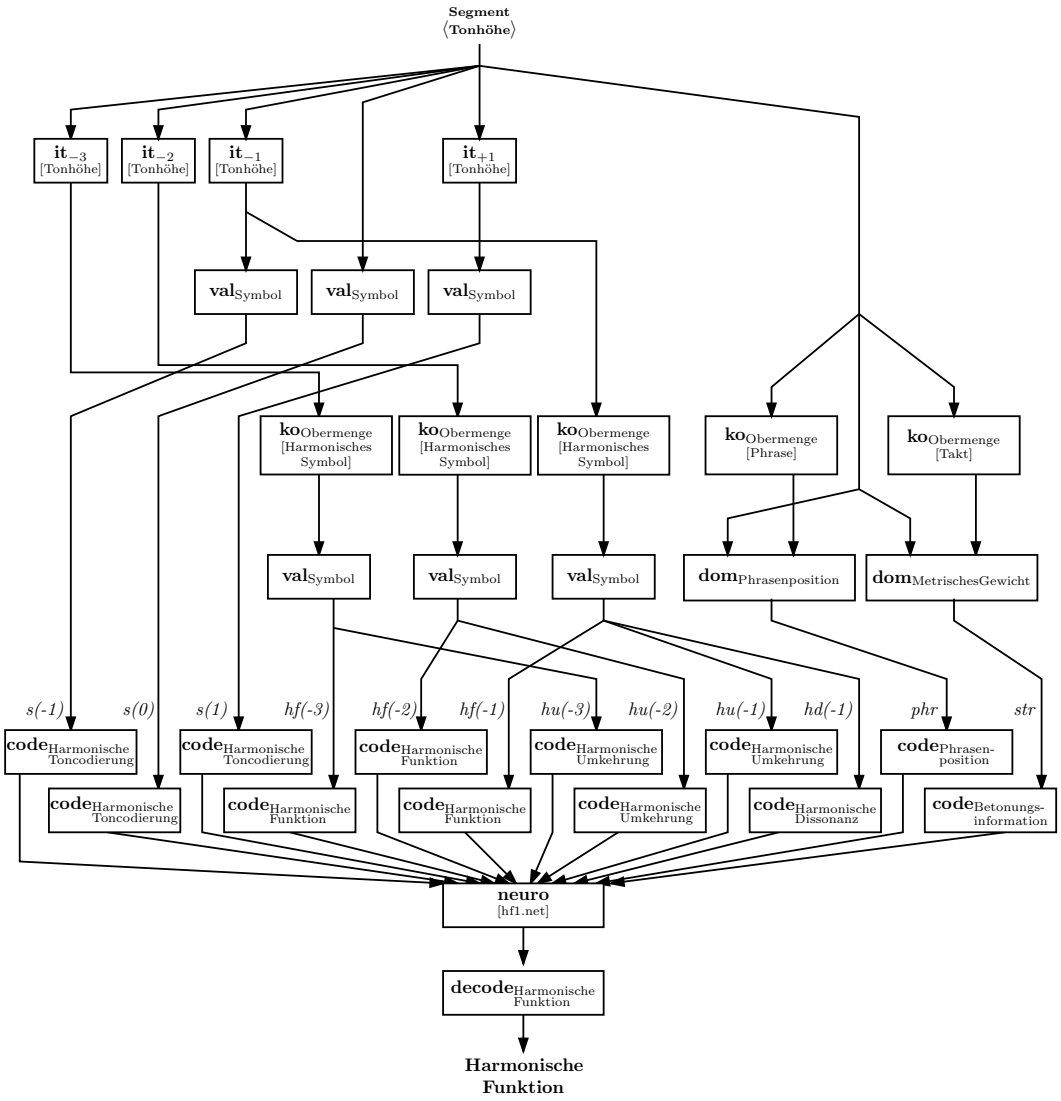


Abbildung 5.15: Aufruf eines neuronalen Netzes in einem Operatorgraphen. Die kursiven Beschriftungen geben die HARMONET-Bezeichnungen der Merkmale an.

skizziert das Rahmenverfahren; die lernbasierten Komponenten „ Supernetz“, „Referenztonnetz“ und „Subnetz“ finden sich in den Abbildungen 5.19, 5.20 und 5.21. Auf die genaue Funktionsweise der gezeigten Operatorgraphen werden wir in Abschnitt 5.3 zurückkommen.

Der Neurooperator erhält seine Eingaben von Codierungsoperatoren, die die benötigten Teilmuster für die Eingabe des Netzes des Neurooperators berechnen. Die Breite der Ausgabe eines Codierungsoperators muss mit Breite der entsprechenden Eingabe des nachfolgenden

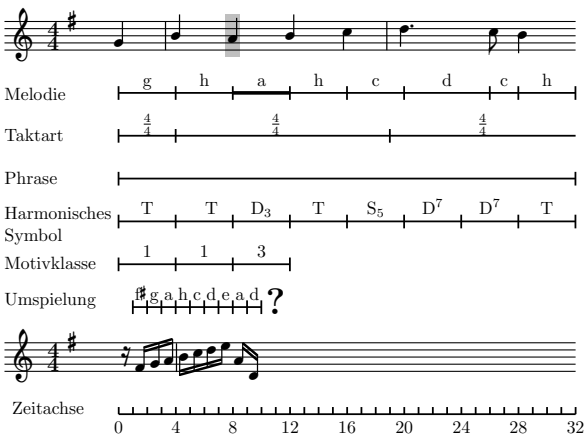


Abbildung 5.16: Umspielung einer Choralmelodie. Der Eingabeton ist in der Choralmelodie und in der Tonhöhenansicht markiert. Das Fragezeichen zeigt die Position des gesuchten Umspielungstons an.

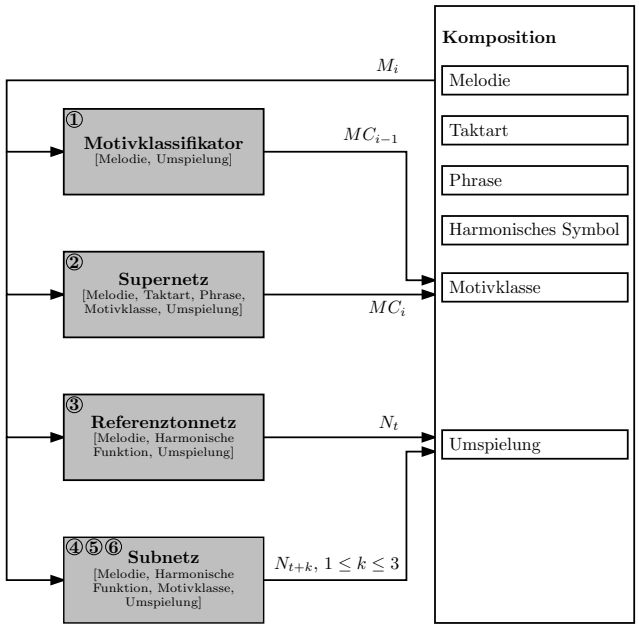


Abbildung 5.17: Rahmenverfahren für MELONET. Die Systemkomponenten (Motivklassifikator: Abbildung 5.18, Supernetz: Abbildung 5.19, Referenztonnetz: Abbildung 5.20, Subnetz: Abbildung 5.21) werden in der angegebenen Reihenfolge ausgewertet. Die Pfeile geben den Informationsfluss zwischen der Repräsentation und den Systemkomponenten an.

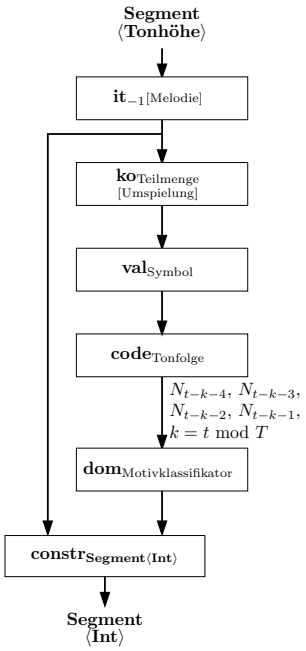


Abbildung 5.18: Aufruf des Motivklassifikators

Neurooperators übereinstimmen. Aus den Teilmustern seiner Eingabe setzt der Neurooperator ein Gesamtmuster zusammen, das er durch sein neuronales Netz schickt. Die Ausgabe ist wieder ein Muster, das durch einen Decodierungsoperator interpretiert werden muss. Auch hier muss die Breite der Ausgabe des Neurooperators mit der Breite der Eingabe des folgenden Decodierers übereinstimmen. Erst der Decodierungsoperator liefert das musikalische Objekt (das Symbol). Das Decodieren eines Netzes kann variiert werden, indem der Decodierungsoperator ausgetauscht wird. Wir haben bereits die deterministische und probabilistische Decodierung kennengelernt.

Genau genommen fällt das kartesische Produkt $\mathbf{Rational}^{n_1} \times \ldots \times \mathbf{Rational}^{n_k}$ in der Deklaration des Neurooperators zu $\mathbf{Rational}^{\sum_{i=1}^{k} n_i}$ zusammen. Die Aufteilung der Eingabe soll aber andeuten, dass der Neurooperator mit den Breiten der Eingabe-Teilmuster und des Ausgabemusters und einem neuronalen Netz parametrisiert ist, ohne dies weiter zu formalisieren. Es gibt also nur einen (parametrisierten) Neurooperator, der zur Auswertung eines beliebigen neuronalen Netzes eingesetzt werden kann.

Das Operatorentripel **code**, **neuro**, **decode** realisiert den Übergang vom symbolischen Raum der problemspezifischen Objekte in einen numerischen Modellraum und zurück. Konzeptionell bedeutet dies, dass man Musikstücke in einem diskreten, symbolischen Raum darstellt und analysiert, während man gleichzeitig über eine definierte Schnittstelle zu numerischen Modellen wie neuronalen Netzen verfügt. Bei der Hintereinanderschaltung von Codierungs- Neuro- und Decodierungsoperatoren müssen die Definitions- und Wertebereiche

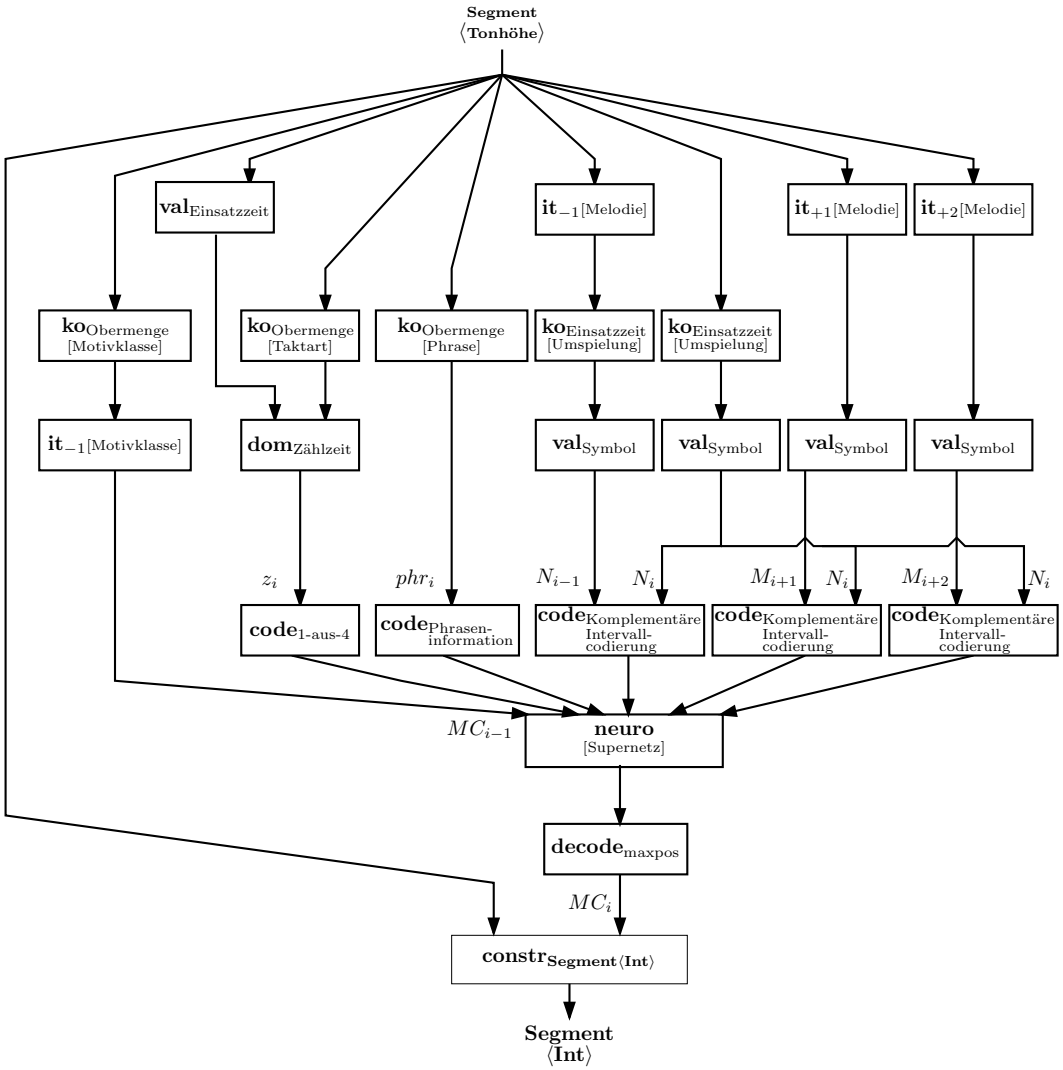


Abbildung 5.19: Aufruf des Supernetzes

so dimensioniert werden, dass der Neuro-Operator ein Eingabemuster einer für das neuronale Netz geeigneten Breite erhält und dass der Decodierungsoperator das Ausgabemuster des neuronalen Netzes verarbeiten kann.

Alternativ hätte man die Codierung und Decodierung der Merkmale auch in den Neurooperator integrieren können. Diese Lösung hätte den Vorteil gehabt, dass man syntaktisch korrekte Eingaben der Neurooperatoren hätte garantieren können. Demgegenüber hätte man jedoch in Kauf nehmen müssen, dass die Lernmuster zum Training von neuronalen Netzen von anderen Codierungsoperatoren berechnet werden als die Muster bei der Auswertung von

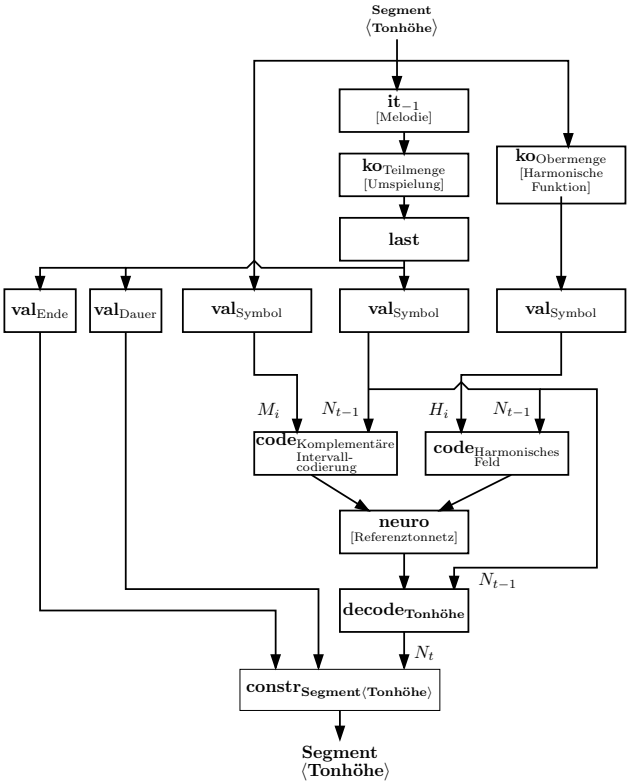


Abbildung 5.20: Aufruf des Referenztonnetzes

Netzen. Dies stellt eine potenzielle Fehlerquelle dar und führt darüberhinaus zur Vervielfachung von gleichem Programmcode. Auch die Benutzerfreundlichkeit hätte unter dieser Lösung gelitten, da ein Benutzer für jedes neuronale Netz einen angepassten Neurooperator hätte implementieren müssen. Richtschnur beim Entwurf der Operatorgraphen ist jedoch das Baukastenprinzip, das vorbereitete Operatoren so sicher wie möglich kombinierbar machen will, ohne dass Programmierkenntnisse erforderlich sind.

Zwei weitere Argumente sprechen für den hier gewählten Entwurf der Neurooperatoren. Zum einen entsteht durch die Trennung von Codierungs-, Decodierungs- und Neurooperatoren keine syntaktische Typunsicherheit. Das bedeutet, dass ein Operatorgraph, dessen Neurooperator eine korrekt codierte Eingabe aus „falschen“ Merkmalen besitzt, dennoch problemlos ausgewertet werden kann. Es handelt sich nur um „semantische“ Typunsicherheit, die bei anderen Operatoren auch nicht erkannt würde. Zweitens ist die Typsicherheit eines Operatorgraphen vor allem bei der Merkmalssuche von Bedeutung, also bei der automatischen Suche nach Operatorgraphen die Merkmale oder Lernmuster berechnen. Diese Graphen enthalten aber keine Neurooperatoren, da letztere zum Einsatz in neuronalen Kompositionsverfahren gedacht sind. Das geschilderte Problem tritt daher bei der Merkmalssuche

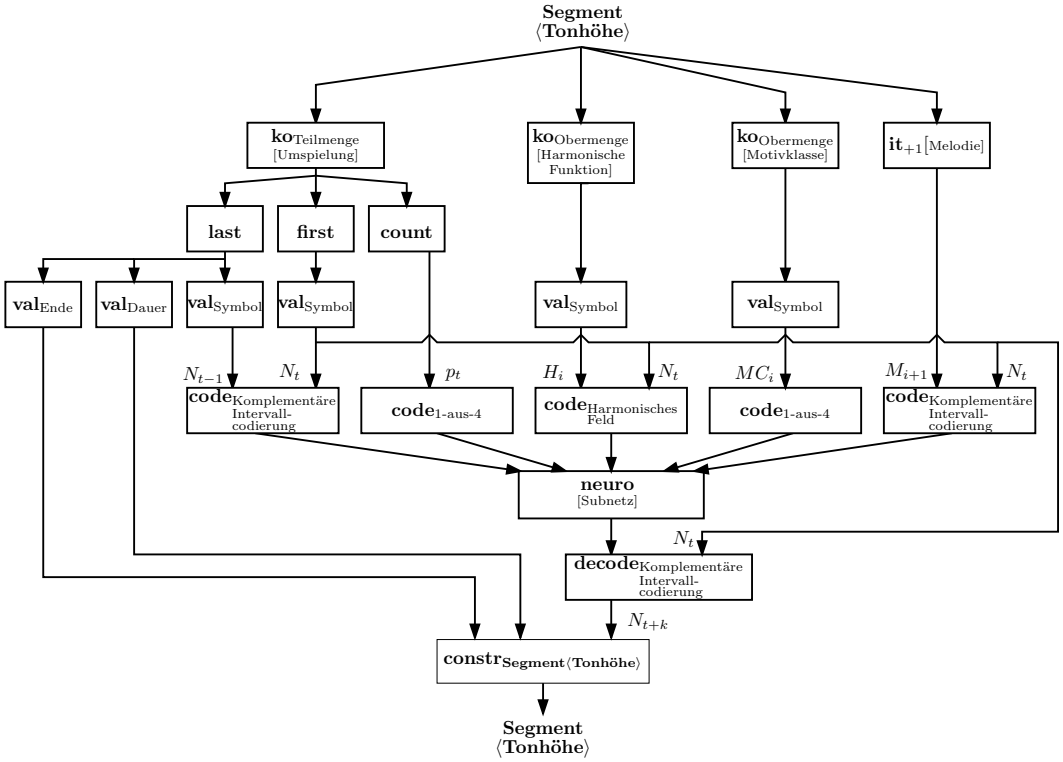


Abbildung 5.21: Aufruf des Subnetzes

nicht auf.

5.2.4 Strukturmanipulation

Die vierte Gruppe von Operatoren in Tabelle 5.6 besteht aus Operatoren zur *Strukturmanipulation*. Bisher haben wir Operatoren kennengelernt, mit denen man Objekte suchen (Iteratoren, Koinzidenzoperatoren) oder neu erzeugen kann (Wissensoperatoren). Es fehlt noch ein Werkzeug, um Teile dieser Objekte zu extrahieren oder Objekte in eine bestehende Repräsentation eines Musikstücks einzufügen. Allgemeiner ausgedrückt benötigt man Operatoren, die die Struktur eines Objekts unabhängig von seiner Bedeutung manipulieren. Strukturmanipulationsoperatoren sind problem- und zeitunabhängig, d.h. sie unterscheiden sich von Iteratoren und Koinzidenzoperatoren dadurch, dass sie wie die Wissensoperatoren nicht auf den musikalischen Kontext zugreifen. Im Gegensatz zu den Wissensoperatoren werten sie aber die Werte der verarbeiteten Objekte nicht aus, sondern verändern lediglich die Baumstruktur von Objekten. Zwei Situationen – die Konstruktion und die Zerlegung von Objekten – sind hier von besonderer Bedeutung.

5.2.4.1 Zerlegung von Objekten

Extraktion von Werten aus Segmenten. Alle musikalischen Objekte in der Repräsentation eines Musikstücks sind in einem Segment gekapselt, das die zeitliche Zuordnung des Objekts speichert. Wenn man mit einem Iterator oder einem Koinzidenzoperator ein Segment gefunden hat, will man auf seinen Inhalt oder auf die zeitliche Information des Segments zugreifen. Diese Aufgabe löst der Wertextraktor. Da ein Segment selbst ein Objekt und damit ein Baum ist, handelt es sich dabei um einen Spezialfall der Zerlegung von Bäumen.

Nun wird die Zerlegung von Objekten allgemeiner definiert. Ein *Teilbaum* eines Objekts besteht aus allen Knoten und Kanten, die sich unterhalb seines *Wurzelknotens* befinden. Jeder Teilbaum eines Objekts mit Ausnahme der Blattknoten ist selbst ein Objekt der Erweiterbaren Zeitreihengrammatik, da diese nur Objekte erzeugt, deren innere Knoten mit einem Typbezeichner beschriftet und damit selbst Wurzelknoten von Objekten sind.

Definition 5.5 (Auswahlfunktion)
Die *Auswahlfunktion*

$$\textbf{subtree}: \begin{cases} \textbf{Objekt} \times \mathbb{N}^* & \longrightarrow & \{\bot\} \cup \textbf{Objekt} \cup \Sigma_W \\ (x, v) & \longmapsto & \begin{cases} y & \text{falls } y \in \textbf{Objekt} \cup \Sigma_W \text{ der Teilbaum von } x \text{ mit der Adresse } v \text{ ist.} \\ \bot & \text{falls es in } x \text{ keinen Teilbaum mit der Adresse } v \text{ gibt.} \end{cases} \end{cases}$$

gibt an, welcher Teilbaum sich in einem Objekt x an der Adresse v befindet. Die *Adresse* $v \in \mathbb{N}^*$ beschreibt den Pfad von der Wurzel von x bis zur Wurzel des Teilbaums y, indem für jeden durchlaufenen Knoten die Nummer des Nachfolgers notiert wird.

* * *

Beispiel. Im Segment

$$s = (\textbf{Segment}\,\langle\textbf{Takt}\rangle\,;\,(\textbf{Int};\,0), (\textbf{Int};\,4), (\textbf{Takt};\;(\textbf{Int};3), (\textbf{Int};4)))$$

hat der Zähler des Takts z.B. die Adresse $(3, 1)$. Der Typbezeichner „$\textbf{Segment}\,\langle\textbf{Takt}\rangle$“ des Objekts wird beim Aufsuchen einer Adresse nicht mitgezählt, weil er in der Baumdarstellung eines Objekts als Beschriftung des Knotens und nicht als Nachfolger des aktuellen Knotens eingesetzt wird. Damit ist

$$\begin{array}{lcll} \textbf{subtree}(s, (3,1)) & = & (\textbf{Int}; 3) & \in \textbf{Int} \\ \textbf{subtree}(s, (3,1,1)) & = & 3 & \in \Sigma_W = \Sigma^*_{Str} \cup \mathbb{Q} \\ \textbf{subtree}(s, (3,1,1,1)) & = & \bot & \end{array}$$

Die Auswahlfunktion **subtree** ist nur eine Hilfsfunktion, um einen allgemeineren Auswahloperator zu definieren.

Definition 5.6 (Auswahloperator)
Es seien $t_1, t_2 \in \mathcal{T}$ Typen einer Erweiterbaren Zeitreihengrammatik. Ein *Auswahloperator* ist ein abstrakter Operator mit einer Deklaration der Form

$$\textbf{select}: \; t_1 \longrightarrow t_2, \;\; K = (),$$

der die Bedingung

$$\exists\, v \in \textbf{Vektor}\,\langle \textbf{Int} \rangle \;\text{ mit }\; \textbf{select}(x) = \textbf{subtree}(x, v)$$

erfüllt.

* * *

Ein Auswahloperator darf also nur Teilbäume seines Eingabeobjekts auswählen. Sein Ergebnis ist immer ein Objekt, da als Wertebereich von **select** ein Typ der Erweiterbaren Zeitreihengrammatik verwendet wird.

Das wichtigste Beispiel für einen Auswahloperator ist der *Wertextraktor*, der es ermöglicht, beim Durchlaufen einer Repräsentation von der zeitlichen auf die musikalisch-inhaltliche Ebene zu wechseln. Der Wertextraktor extrahiert aus einem Segment den gespeicherten Wert, die Einsatzzeit oder die Dauer des Segments. Wir definieren dazu je einen Wertextraktor:

$$\begin{aligned} \textbf{val}_{\text{Symbol}}: &\;\; \textbf{Segment} \longrightarrow t \\ \textbf{val}_{\text{Einsatzzeit}}: &\;\; \textbf{Segment} \longrightarrow \textbf{Int} \\ \textbf{val}_{\text{Dauer}}: &\;\; \textbf{Segment} \longrightarrow \textbf{Int} \end{aligned}$$

Mit der Auswahlfunktion **subtree** kann das Verhalten der Wertextraktoren genau angegeben werden:

$$\begin{aligned} \textbf{val}_{\text{Symbol}}(x) &= \textbf{subtree}(x, (3)) \\ \textbf{val}_{\text{Einsatzzeit}}(x) &= \textbf{subtree}(x, (1)) \\ \textbf{val}_{\text{Dauer}}(x) &= \textbf{subtree}(x, (2)) \end{aligned}$$

Weitere Beispiele für Auswahloperatoren sind Operatoren, die einen bestimmten Nachfolger eines Knotens oder allgemeiner einen Teilbaum an einer vorgegebenen Adresse des Objekts extrahieren. Der Wertextraktor ist ein Spezialfall dieser Operatoren, der explizit definiert wurde, da er häufig benötigt wird.

5.2.4.2 Konstruktion von Objekten

Verkettung von Vektoren. Wissensoperatoren erlauben die Erzeugung neuer Objekte für das Anwendungsgebiet, das durch den benutzerdefinierten Teil der Erweiterbaren Zeitreihengrammatik beschrieben wird. Es gibt aber auch Situationen, die eine allgemeinere Konstruktion von Objekten erfordern. Wenn beispielsweise mehrere Koinzidenzoperatoren Mengen (Vektoren) von Segmenten gefunden haben, möchte man diese Segmente vielleicht gemeinsam weiterverarbeiten. Beispielsweise könnte man die Tonhöhen mehrerer aufeinanderfolgender Phrasen gemeinsam analysieren. Wenn die Zahl der Phrasen für die Analyse

unwichtig ist, erwartet der Wissenoperator für die Analyse einen Tonhöhenvektor, der die Tonhöhen aller untersuchten Phrasen enthält. Man benötigt daher einen Verkettungsoperator für Vektoren, der mehrere Vektoren mit Objekten eines vorgegebenen Typs aneinander hängt. Die Beschaffenheit des Typs ist dabei unwichtig, so dass der Verkettungsoperator nicht den Wissensoperatoren, sondern den Strukturmanipulationsoperatoren zuzurechnen ist. Die Verkettung von Vektoren ist ein Spezialfall der Konstruktion von Objekten aus mehreren Bestandteilen.

Die Zerlegung von Objekten mit **select**, die im vorhergehenden Abschnitt definiert wurde, liefert immer ein gültiges Objekt eines Typs der Erweiterbaren Zeitreihengrammatik. Bei der Konstruktion neuer Objekte muss man hingegen sicherstellen, dass die erzeugten Objekte tatsächlich Elemente der zugrundeliegenden Erweiterbaren Zeitreihengrammatik sind.

Definition 5.7 (Konstruktor)
Es seien $t_1, \ldots, t_r \in \mathcal{T}$, $r \geq 0$, Typen einer Erweiterbaren Zeitreihengrammatik. Ein *Konstruktor* ist ein abstrakter Operator mit Deklaration

$$\mathbf{constr}: \; t_1 \times \ldots \times t_r \longrightarrow t, \;\; K = (),$$

der seine Eingaben zu einem gültigen Objekt vom Typ t zusammensetzt. Erfüllen die Eingaben eventuell vorhandene Randbedingungen nicht, so ist die Ausgabe des Konstruktors undefiniert.

* * *

Beispiel. Im benutzerdefinierten Teil der Erweiterbaren Zeitreihengrammatik in Abschnitt 4.2.6 sind die Werte des Typs **Kontur** auf -1, 0 und 1 für fallende, gleichbleibende und steigende Intervalle eingeschränkt. Ein Kontur-Konstruktor

$$\mathbf{constr_{Kontur}}: \; \mathbf{Int} \longrightarrow \mathbf{Kontur}$$

erzeugt daher für die zulässige Eingabe (**Int**; 0) ein Konturobjekt

$$\mathbf{constr_{Kontur}}((\mathbf{Int};\, 0)) = (\mathbf{Kontur};\, (\mathbf{Int};\, 0))$$

Ungültige Eingaben werden verworfen:

$$\mathbf{constr_{Kontur}}((\mathbf{Int};\, 42)) = \perp$$

Der am häufigsten verwendete Konstruktor ist der Segment-Konstruktor, der an der Ausgabe eines Operatorgraphen ein Objekt in das Segment einfügt, das die Zeitinformation des Objekts enthält (Abbildung 5.23).

Eine weitere Situation, in der man einen Konstruktor benötigt, tritt auf, wenn man mehrere neuronale Netze in einem Operatorgraphen auswertet und die erzeugten Objekte zu einem zusammenfügen will. In Abbildung 5.14 werden die harmonische Funktion, die Umkehrung und die harmonische Dissonanz mit einem Konstruktor zu einer Harmonie verknüpft.

Ein trivialer Auswahloperator ist die Identitität, die ein Objekt unverändert weitergibt ($\mathbf{id}(x) = x$). Die Identität wird nur benötigt, wenn man einen Operatorgraphen der Übersichtlichkeit halber in Schichten anordnen will und einen neutralen Operator als Platzhalter braucht, um Schichten zu überbrücken (vgl. Abb. 6.2).

5.2.4.3 Weitere Beispiele

Neben den schon beschriebenen Operatoren gibt es noch zahlreiche andere Beispiele für Strukturmanipulationen, die von dem Anwendungsbereich weitgehend unabhängig sind.

Man kann etwa Objekte eines geordneten Typs (z.B. Segmente) sortieren, minimale, maximale, erste, letzte oder mehrfach auftretende Elemente aus einem Vektor herausfiltern. Das Sortieren und das Herausfiltern maximaler und minimaler Elemente setzt eine Ordnungsrelation auf dem bearbeiteten Typ voraus. Das Herausfiltern von Mehrfachelementen ist immer möglich, da man alle Objekte einer Erweiterbaren Zeitreihengrammatik auf Gleichheit testen kann.

Ein weiteres Einsatzgebiet von Operatoren zur Strukturmanipulation ist die Strukturreduktion, die man benötigt, wenn man die Ergebnisse mehrerer Suchoperationen kombinieren will. Beispielsweise liefert die Suche mit einem Koinzidenzoperator einen Segmentvektor; die Suche mit einem Iterator liefert ein Segment. Beim Kombinieren mehrerer Suchoperatoren erhält man unter Umständen mehrfach verschachtelte Vektoren. Wenn nur die Menge der gefundenen, aber nicht die Struktur der Verschachtelung für den weiteren Verlauf der Berechnung wichtig sind, muss der verschachtelte Vektor erst einmal linearisiert werden. Die Strukturreduktion besteht in diesem Fall dann im Weglassen der Verschachtelung.

Diese Beispiele zeigen, dass die sinnvollen Strukturmanipulationen mit den genannten Beispielen nicht erschöpfend dargestellt wurden. Eine weitere Systematisierung würde jedoch den Rahmen der vorliegenden Arbeit sprengen.

Tabelle 5.6: Abstrakte Operatoren

Name	**Deklaration** $(n, m \in \mathbb{N}_0,\ t, t_i \in \mathcal{T})$	**Kontext** $(k_i \in \mathcal{T})$	**Bedingungen** (x, x_i aus Definitionsbereich)
Allgemein			
Operator	$\mathbf{op}_K : t_1 \times \ldots \times t_r \longrightarrow t$	$K = (k_1, \ldots, k_m)$	$x_1 = \ldots = x_n = \perp \Rightarrow \mathbf{op}(x_1, \ldots, x_n) = \perp$
Navigation			*Ergebnis gehört zu einer Ansicht*
Iterator	$\mathbf{it} : \mathbf{Segment} \times t_1 \times \ldots \times t_r \longrightarrow \mathbf{Segment}$	$K = (k_1, \ldots, k_m)$ $m \geq 1$	$\mathbf{it}(x) \in \phi(k_1)$ für $x \in \mathbf{Segment}$ $\mathbf{it}(x) \neq x$ für $x \in \mathbf{Segment}$
Gleichzeitigkeit			*Ergebnisse gehören zu einer Ansicht*
Koinzidenzoperator	$\mathbf{ko} : \mathbf{Segment} \longrightarrow \mathbf{Vektor}\langle\mathbf{Segment}\rangle$	$K = (k_1, \ldots, k_m)$	$\forall y \in \mathbf{ko}(x) : y \in \phi(k_1)$ $\forall\, y \in \mathbf{ko}(x) : x \cap y \neq \emptyset$
Anwendungswissen			*Ergebnis: beliebiges Objekt*
Wissensoperator	$\mathbf{dom} : t_1 \times \ldots \times t_r \longrightarrow t$	$K = ()$	
Codierungsoperator	$\mathbf{code} : t_1 \times \ldots \times t_r \longrightarrow \mathbf{Rational}^n$	$K = ()$	Codiert ein Objekt als numerischen Vektor
Neuro-Operator	$\mathbf{neuro} : \mathbf{Rational}^n \longrightarrow \mathbf{Rational}^m$	$K = ()$	Wertet ein neuronales Netz aus
Decodierungsoperator	$\mathbf{decode} : \mathbf{Rational}^m \longrightarrow t$	$K = ()$	Wandelt einen numerischen Vektor in ein Objekt um.
Strukturmanipulation			*Ergebnis ist Teil eines Objekts*
Auswahloperator	$\mathbf{select} : t_1 \longrightarrow t_2$	$K = ()$	$\exists\, v \in \mathbf{Vektor}\langle\mathbf{Int}\rangle$ mit $\mathbf{select}(x) = \mathbf{subtree}(x, v)$
Deskriptor	$\mathbf{descr} : t_1 \longrightarrow t_2$	$K = ()$	Strukturelle Eigenschaft eines Objekt vom Typ t_1
Konstruktor	$\mathbf{constr} : t_1 \times \ldots \times t_r \longrightarrow t$	$K = ()$	konstruiert ein Objekt vom Typ t

Tabelle 5.7: Beispiele für zeitabhängige Operatoren

Deklaration	**Kontext**	**Verhalten**
Iteratoren		
$\mathbf{it}_{+j}$: **Segment** $\longrightarrow$ **Segment**, $j \in \mathbb{N}$	$K = (\mathbf{Objekt})$	Springe j Segmente in die Zukunft.
$\mathbf{it}_{-j}$: **Segment** $\longrightarrow$ **Segment**, $j \in \mathbb{N}$	$K = (\mathbf{Objekt})$	Springe j Segmente in die Vergangenheit.
$\mathbf{it}_{\text{rel}}$: **Segment** $\times$ **Int** $\longrightarrow$ **Segment**	$K = (\mathbf{Objekt})$	Springe x Segmente in Vergangenheit oder Zukunft, wobei x der Wert der zweiten Eingabe ist.
$\mathbf{it}_{\text{abs}}$: **Segment** $\times$ **Int** $\longrightarrow$ **Segment**	$K = (\mathbf{Objekt})$	Gehe an die Position x, wobei x der Wert der zweiten Eingabe ist.
$\mathbf{it}_{\text{Dauer}}$: **Segment** $\times$ **Int** $\times$ **Int** $\longrightarrow$ **Segment**	$K = (\mathbf{Objekt})$	Gehe zum Segment, das sich x Zeiteinheiten weiter rechts (bzw. links) befindet, wobei x der Wert der zweiten Eingabe ist.
$\mathbf{it}_{\text{relParallel}}$: **Segment** $\longrightarrow$ **Segment**	$K = (\mathbf{Objekt}, \mathbf{Objekt})$	Gehe zum Segment der ersten Kontextansicht, das bezogen den Nachfolger des gleichzeitigen Objekts der zweiten Kontextansicht dieselbe Position hat.
$\mathbf{it}_{\text{MaximaleTonhöhe}}$: **Segment** $\longrightarrow$ **Segment**	$K = (\mathbf{Tonhöhe})$	Gehe zum nächsten Maximum der melodischen Kontur
$\mathbf{it}_{\text{Motiv}}$: **Segment** $\times$ **Harmonie** $\longrightarrow$ **Segment**	$K = (\mathbf{Motiv}, \mathbf{Harmonie})$	Gehe zum nächsten Motiv, das den gleichen harmonischen Kontext hat wie die zweite Eingabe
$\mathbf{it}_{\text{HarmonischeDissonanz}}$: **Segment** $\longrightarrow$ **Segment**	$K = (\mathbf{Tonhöhe}, \mathbf{Harmonie})$	Gehe zum nächsten Ton, der in seinem harmonischen Kontext dissonant ist.
Koinzidenzoperatoren		
$\mathbf{ko}_{\text{Einsatzzeit}}$: **Segment** $\longrightarrow$ **Vektor** $\langle$**Segment**$\rangle$	$K = (\mathbf{Objekt})$	Segmente mit gleicher Einsatzzeit wie Eingabe
$\mathbf{ko}_{\text{Teilmenge}}$: **Segment** $\longrightarrow$ **Vektor** $\langle$**Segment**$\rangle$	$K = (\mathbf{Objekt})$	Segmente, die im Eingabesegment enthalten sind
$\mathbf{ko}_{\text{Obermenge}}$: **Segment** $\longrightarrow$ **Vektor** $\langle$**Segment**$\rangle$	$K = (\mathbf{Objekt})$	Segmente, die das Eingabesegment enthalten
$\mathbf{ko}_{\text{Überlappung}}$: **Segment** $\longrightarrow$ **Vektor** $\langle$**Segment**$\rangle$	$K = (\mathbf{Objekt})$	Segmente, die einen mit der Eingabe einen nicht-leeren Schnitt haben

Tabelle 5.8: Beispiele für strukturmanipulierende Operatoren

Deklaration	**Laufzeittyp** (Eingabetyp $\rightsquigarrow$ Ausgabetyp)	**Verhalten**
Auswahloperatoren		
$\mathbf{val}_{\text{Symbol}} : \mathbf{Segment} \longrightarrow \mathbf{Objekt}$	$\mathbf{Segment}\,\langle X\rangle \rightsquigarrow X$	*Wertextraktor*: Wert eines Segments $\mathbf{val}_{\text{Symbol}}(x) = \mathbf{subtree}(x,(3))$
$\mathbf{val}_{\text{Einsatzzeit}} : \mathbf{Segment} \longrightarrow \mathbf{Int}$		Einsatzzeit eines Segments: $\mathbf{val}_{\text{Einsatzzeit}}(x) = \mathbf{subtree}(x,(1))$
$\mathbf{val}_{\text{Dauer}} : \mathbf{Segment} \longrightarrow \mathbf{Int}$		Dauer eines Segments: $\mathbf{val}_{\text{Dauer}}(x) = \mathbf{subtree}(x,(2))$
$\mathbf{id} : \mathbf{Objekt} \longrightarrow \mathbf{Objekt}$	$X \rightsquigarrow X$	Die *Identität* gibt die Eingabe unverändert aus: $\mathbf{id}(x) = x$
Verklebungsoperatoren		
$\mathbf{append} : \mathbf{Segment} \times \mathbf{Segment} \longrightarrow \mathbf{Segment}$	$\mathbf{Segment}\,\langle X\rangle \times \mathbf{Segment} \rightsquigarrow \mathbf{Segment}\,\langle X\rangle$	Hülle der beiden Segmente mit dem Wert des ersten Segments.
$\mathbf{union} : \mathbf{Vektor} \times \mathbf{Vektor} \longrightarrow \mathbf{Vektor}$	$\mathbf{Vektor}\,\langle X\rangle \times \mathbf{Vektor}\,\langle X\rangle \rightsquigarrow \mathbf{Vektor}\,\langle \mathrm{kgV}(X,Y)\rangle$	Vereinigung zweier Vektoren
Deskriptoren		
$\mathbf{count} : \mathbf{Vektor} \longrightarrow \mathbf{Int}$		Anzahl der Elemente eines Vektors
$\mathbf{first} : \mathbf{Vektor} \longrightarrow \mathbf{Objekt}$	$\mathbf{Vektor}\,\langle X\rangle \rightsquigarrow X$	Erstes Element eines Vektors
$\mathbf{last} : \mathbf{Vektor} \longrightarrow \mathbf{Objekt}$	$\mathbf{Vektor}\,\langle X\rangle \rightsquigarrow X$	Letztes Element eines Vektors
$\mathbf{length} : \mathbf{Vektor}\,\langle\mathbf{Segment}\rangle \longrightarrow \mathbf{Int}$	$X \rightsquigarrow \mathbf{Int}$	Gesamtdauer einer Segmentmenge
Konstruktoren		
$\mathbf{constr}_{\mathbf{Kontur}} : \mathbf{Int} \longrightarrow \mathbf{Kontur}$		Konstruktion eines Kontur-Objekts aus einer ganzen Zahl
$\mathbf{constr}_{\mathbf{Segment}} : \mathbf{Int} \times \mathbf{Int} \times \mathbf{Objekt} \longrightarrow \mathbf{Segment}$	$\mathbf{Int} \times \mathbf{Int} \times X \rightsquigarrow \mathbf{Segment}\,\langle X\rangle$	Konstruktion eines Segments-Objekts aus Einsatzzeit, Dauer und einem Objekt
$\mathbf{constr}_{\mathbf{Segment}} : \mathbf{Segment} \times \mathbf{Objekt} \longrightarrow \mathbf{Segment}$	$\mathbf{Segment} \times X \rightsquigarrow \mathbf{Segment}\,\langle X\rangle$	Konstruktion eines Segments-Objekts aus der Zeitinformation eines Segments und einem Objekt

Tabelle 5.9: Beispiele für Wissensoperatoren (I)

Deklaration	**Verhalten**
Symbolische Wissenstransformation	
$\mathbf{dom}_{\text{Intervall}}$: **Tonhöhe** × **Tonhöhe** ⟶ **Intervall**	Intervall zwischen zwei Tonhöhen
$\mathbf{dom}_{\text{Kontur}}$: **Tonhöhe** × **Tonhöhe** ⟶ **Kontur**	Richtung zwischen zwei Tonhöhen
$\mathbf{dom}_{\text{Dauernverhältnis}}$: **Int** × **Int** ⟶ **Rational**	Verhältnis zweier Dauern
$\mathbf{dom}_{\text{MetrischesGewicht}}$: **Int** × **Segment** ⟨**Takt**⟩ ⟶ **Int**	Metrisches Gewicht eines Ereignisses
$\mathbf{dom}_{\text{Zählzeit}}$: **Int** × **Segment** ⟨**Takt**⟩ ⟶ **Int**	Zählzeit einer Zeitposition in einem Taktsegment
$\mathbf{dom}_{\text{Phrasenform}}$: **Vektor** ⟨**Tonhöhe**⟩ ⟶ **Phrasenform**	Phrasenform einer Folge von Tönen
$\mathbf{dom}_{\text{Phrasenposition}}$: **Segment** × **Segment** ⟶ **Position**	Position des ersten Segments im zweiten.
$\mathbf{dom}_{\text{Motivklassifikator}}$: **Rational**^{12} ⟶ **Int**	Klassifikation einer codierten Tonfolge
$\mathbf{dom}_{\text{Ambitus}}$: **Vektor** ⟨**Segment** ⟨**Tonhöhe**⟩⟩ ⟶ **Ambitus**	Ambitus einer Tonfolge
$\mathbf{dom}_{\text{Ambitusaktualisierer mit Gedächtnis}}$: **Tonhöhe** ⟶ **Ambitus**	Aktualisierung des im Operator gespeicherten Ambitus
$\mathbf{dom}_{\text{Ambitusaktualisierer}}$: **Ambitus** × **Tonhöhe** ⟶ **Ambitus**	Aktualisierung eines Ambitus-Objekts durch eine Tonhöhe
Mutierer	
$\mathbf{rand}_{Tonhoehe}$: ∅ ⟶ **Tonhöhe**	zufällige Tonhöhe
$\mathbf{rand}_{Nachbarton}$: **Tonhöhe** ⟶ **Tonhöhe**	zufälliger ausgewählter Nachbarton der Eingabe
$\mathbf{rand}_{Schiebemutierer}$: **Tonhöhe** × **Tonhöhe** × **Tonhöhe** ⟶ **Tonhöhe**	Zufällige Verschiebung des zweiten Tons, so dass Kontur bzgl. des ersten und dritten Tons erhalten bleibt.
$\mathbf{rand}_{Gittermutierer}$: **Tonhöhe** × **Harmonie** ⟶ **Tonhöhe**	Zufällige Verschiebung einer Tonhöhe auf einem harmonischen Gitter
$\mathbf{rand}_{Rhythmusmutierer}$:	Veränderung des Rhythmus unter Beachtung der Phrasengrenzen
Vektor ⟨**Segment** ⟨**Tonhöhe**⟩⟩ × **Vektor** ⟨**Segment** ⟨**Phrase**⟩⟩ ⟶ **Vektor** ⟨**Segment** ⟨**Tonhöhe**⟩⟩	

Tabelle 5.10: Beispiele für Wissensoperatoren (II)

Deklaration	**Verhalten**
Codierung	
$\mathbf{code}_{\text{HarmonischeToncodierung}}$: **Tonhöhe** $\longrightarrow$ **Rational**^{12}	Harmonische Toncodierung einer Tonhöhe
$\mathbf{code}_{\text{HarmonischeFunktion}}$: **HarmonischesSymbol** $\longrightarrow$ **Rational**^{12}	Codierung der harmonischen Funktion eines harmonischen Symbols
$\mathbf{code}_{\text{HarmonischeUmkehrung}}$: **HarmonischesSymbol** $\longrightarrow$ **Rational**$^{?}$	Codierung der Umkehrung eines harmonischen Symbols
$\mathbf{code}_{\text{HarmonischeDissonanz}}$: **HarmonischesSymbol** $\longrightarrow$ **Rational**$^{?}$	Codierung der Dissonanz eines harmonischen Symbols
$\mathbf{code}_{\text{Phrasenposition}}$: **Position** $\longrightarrow$ **Rational**$^{?}$	Codierung der Phrasenposition
$\mathbf{code}_{\text{Phraseninformation}}$: ? $\longrightarrow$ **Rational**$^{?}$	Codierung von Phraseninformationen
$\mathbf{code}_{\text{Betonungsinformation}}$: **Int** $\longrightarrow$ **Rational**	Codierung von Betonungsinformation
$\mathbf{code}_{\text{Tonfolge}}$: **Vektor** $\langle$**Tonhöhe**$\rangle$ $\longrightarrow$ **Rational**$^{?}$	Codierung einer Tonfolge
$\mathbf{code}_{\text{1-aus-4}}$: **Int** $\longrightarrow$ **Rational**^{4}	1-aus-4-Codierung einer Zahl
$\mathbf{code}_{\text{KomplementäreIntervallcodierung}}$: **Tonhöhe** $\times$ **Tonhöhe** $\longrightarrow$ **Rational**$^{?}$	Komplementäre Intervallcodierung
$\mathbf{code}_{\text{HarmonischesFeld}}$: **HarmonischesSymbol** $\times$ **Tonhöhe** $\longrightarrow$ **Rational**^{7}	Codierung einer Harmonie als Harmonisches Feld
Decodierung	
$\mathbf{decode}_{\text{HarmonischeFunktion}}$: **Rational**^{12} $\longrightarrow$ **HarmonischeFunktion**	Decodierung einer harmonischen Funktion
$\mathbf{decode}_{\text{maxpos}}$: **Vektor** $\langle$**Rational**$\rangle$ $\longrightarrow$ **Int**	Index der (ersten) Klasse mit der stärksten Aktivierung

5.3 Operatorgraph

In Abschnitt 5.2 wurden Gruppen von Operatoren mit unterschiedlichen Aufgaben vorgestellt. Nun werden die Operatoren zu Operatorgraphen kombiniert, um Funktionen zur Berechnung musikalischer Objekte darzustellen.

Operatorgraph. Ein Operatorgraph ist ein gerichteter azyklischer Graph, dessen Knoten Operatoren enthalten. Die Kanten verbinden die Operatoren und verlaufen jeweils von der Ausgabe eines Operators zu einer Eingabe eines nachfolgenden Operators. Ein Operatorgraph wird ausgewertet, indem man als Eingabe ein Objekt wählt und bei den Eingabeknoten, d.h. bei Knoten ohne einlaufende Kanten beginnend sukzessive die Ausgabe aller Operatoren berechnet. Die Ausgabe eines Operators dient dabei als Eingabe für nachfolgenden Knoten. Die Ergebnisse der Ausgabeknoten (Knoten ohne auslaufende Kanten) sind gleichzeitig das Ergebnis des Operatorgraphen. Alle Operatoren eines Operatorgraphen beziehen sich auf eine gemeinsame Kontextkomposition (vgl. Abbildung 5.1). Ein Operatorgraph kann also nur dann ausgewertet werden, wenn alle Operatoren auf diese Kontextkomposition anwendbar sind.

Wohldefiniertheit. Ein Operatorgraph ist eine Verkettung von Operatoren. Da Operatoren unterschiedliche Definitions- und Wertebereiche haben, können sie nicht auf beliebige Weise miteinander verknüpft werden; vielmehr müssen die Definitions- und Wertebereiche aufeinanderfolgender Operatoren zusammenpassen. Ein Operator mit $k \in \mathbb{N}$ Eingaben muss daher Nachfolger von k Operatoren mit geeignetem Ausgabetyp sein.

Erweiterbarkeit und Abstraktion. Wie bei der Erweiterbaren Zeitreihengrammatik lässt sich neues Anwendungswissen auch in Operatorgraphen integrieren. Um einen Operatorgraphen auszuwerten, durchläuft man seine Knoten in der Richtung der Kanten. Die Durchlaufstrategie hängt nur von der Verknüpfung der Knoten, nicht aber von der Funktionalität der Operatoren in den Knoten ab. Man kann daher Operatoren mit neuer Funktionalität definieren, ohne dass sich die Auswertung von Operatorgraphen verändert. Rahmenverfahren verwenden Operatorgraphen wie Variablen, weil sie aufgrund der einheitlichen Auswertungsstrategie vom Aufbau der Operatorgraphen abstrahieren. Ein Rahmenverfahren, das Operatorgraphen einsetzt, ist somit durch Funktionsmodule parametrisiert.

In Abschnitt 5.4 werden wir drei Anwendungen von Operatorgraphen kennenlernen: die Berechnung von Merkmalen, die Berechnung von Lernmustern zum Training neuronaler Netze und Auswertung von neuronalen Netzen innerhalb eines Kompositionsverfahrens. Dies ist die Grundlage für die Entwicklung von Verfahren zur Merkmalssuche in Kapitel 6.

Definition 5.8 (Operatorgraph)

(a) Gegeben sei eine erweiterbare Zeitreihengrammatik mit Typen $\mathcal{T}$. Ein *Operatorgraph* $OG = (V, E)$ ist ein gerichteter Graph, der Operatoren über der vorgegebenen Erweiterbaren Zeitreihengrammatik als Knoten besitzt. Die Knoten sind in Schichten $V = V_1 \dot\cup \ldots \dot\cup V_m$ partitioniert, $m \in \mathbb{N}$. Die Kanten E verlaufen von einer Schicht V_i zur einer nachfolgenden Schicht V_j mit $i < j \leq m$ für $1 \leq i \leq m-1$. Ein Operator ist ein *Nachfolger* eines anderen Operators, wenn die zugehörigen Knoten durch eine gerichtete Kante vom zweiten zum ersten Knoten verbunden sind.

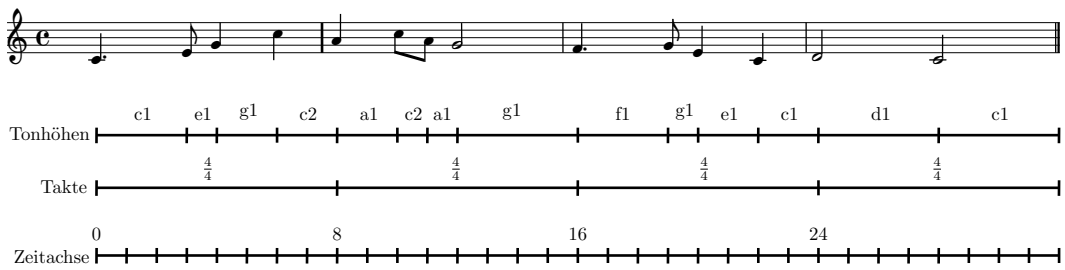


Abbildung 5.22: Repräsentation des Melodieanfangs von „Alle Vögel sind schon da“ mit zwei Ansichten.

Hat ein Operator eines Knotens die Signatur

$$t_1 \times \ldots \times t_r \longrightarrow t$$

für $t_1 \ldots, t_r, t \in \mathcal{T}, r \in \mathbb{N}$, so besitzt er r einlaufende Kanten, die den Eingaben des Operators zugeordnet sind. Jeder Operator hat mindestens eine auslaufende Kante. Die Operatoren der Eingabeschicht V_1 haben genau eine Eingabe, zu der je eine Kante von einer unsichtbaren Quelle hinführt. Die Ausgaben der letzten Schicht V_r führen in eine unsichtbare Senke.

(b) Gegeben sei eine $(s_1, \ldots, s_n)$-Komposition $(A_1, \ldots, A_n)$, $n \in \mathbb{N}$, mit einer *Eingabeansicht* $A_i \in \{A_1, \ldots, A_n\}$. Ein Operatorgraph ist auf die Komposition $(A_1, \ldots, A_n)$ *anwendbar*, wenn für jeden Operator des Operatorgraphen eine Kontextzuordnung existiert, so dass

- jeder Operator des Operatorgraphen auf die Komposition $(A_1, \ldots, A_n)$ anwendbar ist,
- der Typ **Segment** $\langle s_i \rangle$ der Objekte in der Eingabeansicht A_i spezieller oder gleich den Eingabetypen der Operatoren in der Eingabeschicht V_1 ist und
- der *Laufzeittyp*, d.h. der allgemeinstmögliche Ausgabetyp eines Operators spezieller oder gleich den deklarierten Eingabetypen nachfolgender Operatoren ist.

* * *

Beispiel. Die Definition des Operatorgraphen wird nun an einem Beispiel veranschaulicht. Gegeben ist der Anfang der Melodie „Alle Vögel sind schon da“ in einer Darstellung mit zwei Ansichten für die Tonhöhen und die Taktart jedes Takts (Abbildung 5.22). Es wurde die Erweiterbare Zeitreihengrammatik aus Kapitel 4 zugrundegelegt. Mit Hilfe eines Operatorgraphen soll nun aus den vorhandenen Ansichten eine dritte Ansicht berechnet werden, die für jeden Ton der Melodie das metrische Gewicht der Einsatzzeit angibt.

Mit jeder Taktart wird in der Musiktheorie ein bestimmtes Betonungsschema verbunden. Das *metrische Gewicht* ist eine Zahl, die die relative Bedeutung eines Zeitpunkts in diesem Schema angibt (vgl. Abschnitt 5.2.3.2). Der erste Schlag eines Takts erhält immer

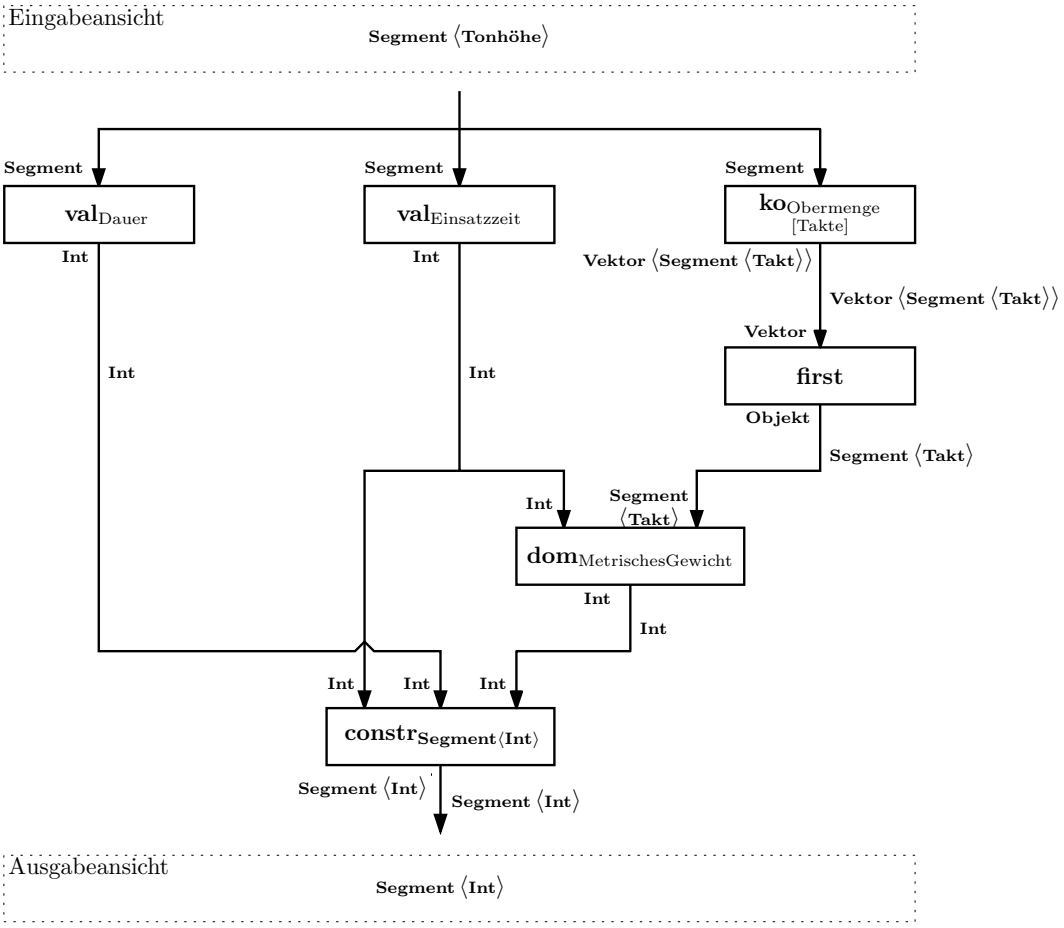


Abbildung 5.23: Ein Operatorgraph zur Berechnung des metrischen Gewichts. Kantenbeschriftung links: deklarierter Ein- bzw. Ausgabetyp eines Operators. Kantenbeschriftung rechts: Laufzeittyp der Ausgabe für den Typ der Eingabeansicht.

ein besonderes Gewicht, bei geraden Taktarten (z.B. $\frac{2}{4}$-, $\frac{4}{4}$-, $\frac{6}{8}$-Takt) existiert ein zweiter, schwächerer Schwerpunkt in der Taktmitte. Zählzeiten, die durch weitere Unterteilungen gewonnen werden, werden noch weniger betont. Als Betonungsbewertung für einen $\frac{4}{4}$-Takt mit Achtelauflösung wählen wir hier $(4, 1, 2, 1, 3, 1, 2, 1)$. Ein Ereignis am Taktanfang wird also mit 4 bewertet, ein Ereignis, das in der Taktmitte einsetzt, mit 3.

Aufbau des Operatorgraphen. Abbildung 5.23 zeigt einen Operatorgraphen, der das metrische Gewicht eines Ereignisses berechnet. Eingabe ist ein Segment der Ansicht „Tonhöhe“. Die Ausgabe des Operatorgraphen soll in der Ansicht „Metrisches Gewicht“ gespeichert werden. Die Ansichten werden in der Abbildung durch die gestrichelten Kästen dargestellt. Auf der linken Seite der Kanten sind die deklarierten Ein- und Ausgabetypen der Operatoren

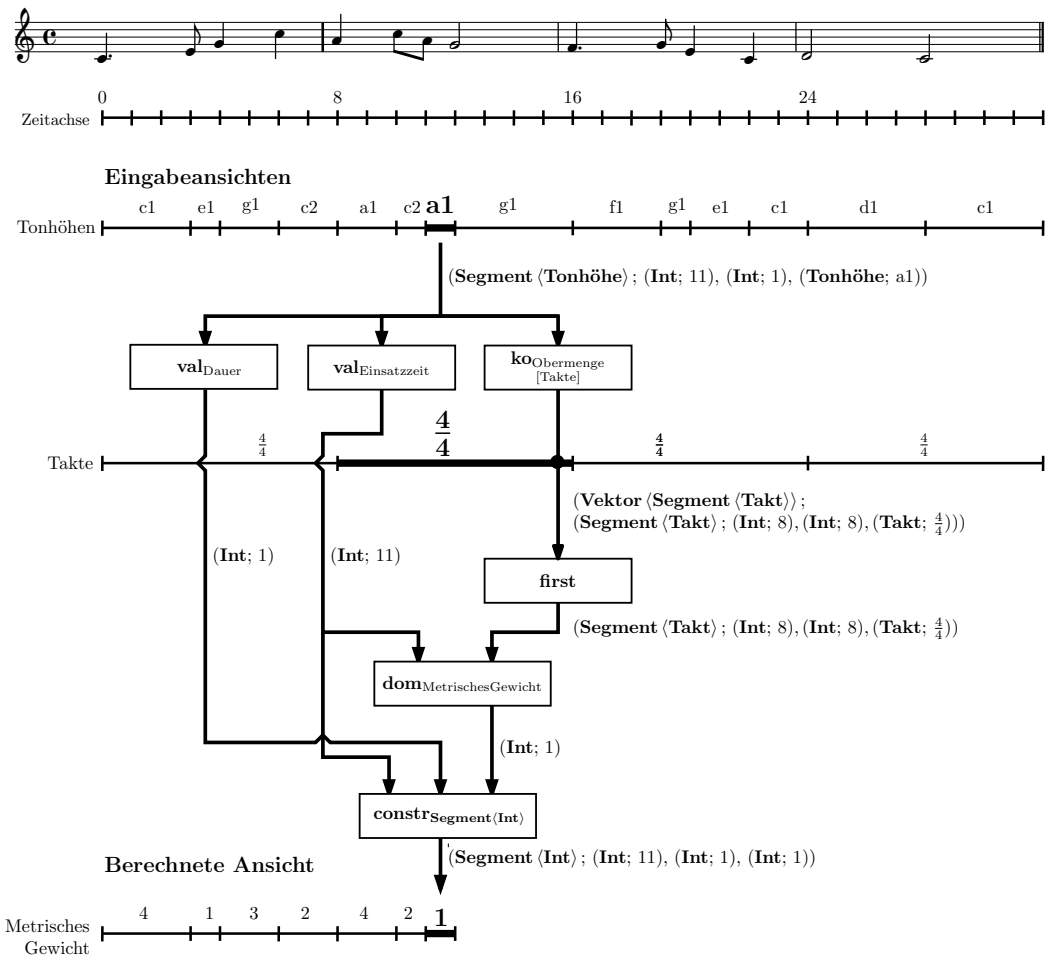


Abbildung 5.24: Auswertung eines Operatorgraphen aus Abbildung 5.23

angegeben. Der Laufzeittyp auf der rechten Seite gibt den allgemeinsten Typ an, der als Ausgabe des Operators auftreten kann, wenn man die Laufzeittypen seiner Vorgänger berücksichtigt. Beispielsweise verarbeitet der Operator **first** zwar beliebige Vektoren, doch wenn er wie im Beispiel einen Vektor von Taktsegmenten als Eingabe erhält, ist der Laufzeittyp seiner Ausgabe **Segment** ⟨**Takt**⟩

Die Anzahl der einlaufenden Kanten eines Operators stimmt mit der Anzahl der Eingaben in der Deklaration überein. Im Beispiel hat der Wissensoperator $\mathbf{dom}_{\text{MetrischesGewicht}}$ zwei einlaufende Kanten, die den beiden Eingaben vom Typ **Int** und **Segment** ⟨**Takt**⟩ zugeordnet sind. Ein Knoten kann mehrere auslaufende Kanten besitzen, die die Ausgabe des Operators an nachfolgende Operatoren weiterleiten. Der Wertextraktor $\mathbf{val}_{\text{Einsatzzeit}}$ bestimmt z.B. die Einsatzzeit der Eingabe, die von zwei Operatoren verwendet wird. Ein Knoten in

einem Operatorgraphen hat mindestens eine auslaufende Kante, da die Berechnung seines Operators andernfalls überflüssig für das Gesamtergebnis wäre. Es existiert also von jedem Knoten ein gerichteter Pfad zur Senke.

Die Auswertung eines Operatorgraphen terminiert immer, weil ein Operatorgraph ein azyklischer Graph ist: Wenn jede Knotenschicht nur Kanten in eine Schicht mit größerem Index führen, kann der Graph keine Zyklen enthalten. Umgekehrt kann jeder azyklische Graph in Knotenschichten partitioniert werden, so dass nur Kanten in Schichten mit größerem Index führen [9, S.173, 10.1.3]. Die Definition eines Operatorgraphen erlaubt es also, alle azyklischen Graphen auszudrücken. Die Darstellung mit Knotenschichten wurde nur der Anschaulichkeit halber gewählt.

Auswertung des Operatorgraphen. Betrachten wir einen Durchlauf durch den Operatorgraphen (Abbildung 5.24). Als Kontextkomposition dient die abgebildete Melodie, die mit Hilfe der Ansichten „Tonhöhe“ und „Takte“ repräsentiert wurde. Als Eingabeansicht wurde die Ansicht „Tonhöhe“ gewählt, d.h. die Segmente dieser Ansicht werden als Eingabe des Operators verwendet. Bei der Berechnung einer neuen Ansicht werden die Segmente der Eingabeansicht von links nach rechts, also in ihrer zeitlichen Reihenfolge abgearbeitet. Im Beispiel wurden bereits metrische Gewichte für die ersten sechs Töne bestimmt. Das aktuelle Eingabesegment ist nun:

$$(\mathbf{Segment}\,\langle\mathbf{Tonhöhe}\rangle\,;\,(\mathbf{Int};\,11),(\mathbf{Int};\,1),(\mathbf{Tonhöhe};\,\mathrm{a1}))$$

Die Funktionsweise eines Operatorgraphen lässt sich am besten verstehen, wenn man von dem gesuchten Objekt ausgehend überlegt, welche Informationen benötigt werden, um das Objekt zu erstellen. Hier soll ein neues Segment konstruiert werden, das dieselbe Zeitinformation wie der Eingabeton hat, aber statt einer Tonhöhe das metrische Gewicht der Eingabe enthält. Am Ende der Berechnung steht daher der Konstruktor $\mathbf{constr}_{\mathbf{Segment}\,\langle\mathbf{Int}\rangle}$, der aus einer Einsatzzeit, einer Dauer und einem metrischen Gewicht ein neues Segment konstruiert. Die Zeitinformation wird mit den Wertextraktoren $\mathbf{val}_{\mathrm{Einsatzzeit}}$ und $\mathbf{val}_{\mathrm{Dauer}}$ aus dem Eingabesegment ermittelt. Einsatzzeit (**Int**; 11) und Dauer (**Int**; 1) dienen als erste und zweite Eingabe des Konstruktors in der letzten Schicht des Operatorgraphen. Bleibt die Berechnung des metrischen Gewichts, der dritten Eingabe des Konstruktors.

Um das metrische Gewicht eines Ereignisses zu berechnen, benötigt man die Einsatzzeit des untersuchten Ereignisses, die Einsatzzeit des Takts, der das Ereignis enthält und seine Taktart. Die Einsatzzeit der Eingabe wurde bereits vom Wertextraktor $\mathbf{val}_{\mathrm{Einsatzzeit}}$ ermittelt, da sie auch als Einsatzzeit des neuen Segments benötigt wird. Der Koinzidenzoperator $\mathbf{ko}_{\mathrm{Obermenge}}[\mathrm{Takte}]$ sucht die Segmente der Taktansicht „Takte“, die die Eingabe zeitlich umschließen. Als Kontexttyp des Koinzidenzoperators wurde **Takt** gewählt. Der Kontexttyp wurde der Ansicht „Takte“ zugeordnet. Da ein Koinzidenzoperator zu einer Eingabe mehrere gleichzeitige Segmente finden kann, ist sein Ergebnis ein Segmentvektor:

$$(\mathbf{Vektor}\,\langle\mathbf{Segment}\,\langle\mathbf{Takt}\rangle\rangle\,;\,(\mathbf{Segment}\,\langle\mathbf{Takt}\rangle\,;\,(\mathbf{Int};\,8),(\mathbf{Int};\,8),(\mathbf{Takt};\,\tfrac{4}{4})))$$

Aus diesem wählt der Auswahloperator **first** das erste Element

$$(\mathbf{Segment}\,\langle\mathbf{Takt}\rangle\,;\,(\mathbf{Int};\,8),(\mathbf{Int};\,8),(\mathbf{Takt};\,\tfrac{4}{4}))$$

aus, dessen Position im Vektor durch die Adresse (1) festgelegt ist. Das ausgewählte Segment enthält sowohl die Einsatzzeit 8 des Takts als auch seine Taktart $\frac{4}{4}$. Zusammen mit der Einsatzzeit des Eingabeereignisses ermöglicht es dem Wissensoperator $\mathbf{dom}_{\text{MetrischesGewicht}}$, das gesuchte metrische Gewicht (**Int**; 1) zu ermitteln.

Nun sind alle Angaben vorhanden, die der Konstruktor $\mathbf{constr}_{\mathbf{Segment}\,\langle\mathbf{Int}\rangle}$ in der letzten Schicht braucht, um das Ausgabeobjekt

$$(\mathbf{Segment}\,\langle\mathbf{Int}\rangle\,;\,(\mathbf{Int};\,11),(\mathbf{Int};\,1),(\mathbf{Int};\,1))$$

des Operatorgraphen zu konstruieren. Das Ergebnis wird nun zur Zielansicht „Metrisches Gewicht“ hinzugefügt.

Verknüpfung von Operatorgraph und Kontextkomposition. Im Beispiel wurde das metrische Gewicht für die Einsatzzeiten der Melodietöne berechnet, d.h. als Eingabeansicht wurde die Ansicht „Tonhöhe“ gewählt. Betrachtet man die Eingabetypen der Operatoren in der ersten Schicht (Abbildung 5.23), so stellt man fest, dass sie ein beliebiges Segment erwarten, nicht notwendigerweise ein Segment, das eine Tonhöhe enthält. Man hätte als Eingabeansicht also auch jede andere Ansicht der Kontextkomposition wählen können. Dies ist auch musikalisch sinnvoll, da das metrische Gewicht die Einsatzzeit eines beliebigen Ereignisses bewertet. Der Operatorgraph ist also durch die Wahl der Eingabeansicht „Tonhöhe“ mit der bearbeiteten Komposition verknüpft.

Die zweite Art der Verknüpfung zwischen Operatorgraph und Komposition entsteht dadurch, dass manchen Operatoren ein Kontext in der Komposition zugeordnet werden muss. Der Koinzidenzoperator hat hier den Kontexttyp **Takt**, der der Ansicht „Takte“ zugeordnet wurde. Im Beispiel ist die Wahl des Kontexttyps und der zugeordneten Ansicht erzwungen, da der Wissensoperator $\mathbf{dom}_{\text{MetrischesGewicht}}$ ein Takt-Objekt als Eingabe benötigt. Da die Kontextkomposition nur eine Ansicht vom Typ **Takt** enthält, kann der Kontexttyp des Koinzidenzoperators nur dieser Ansicht zugeordnet werden.

Die Verknüpfungen zwischen der Repräsentation einer Komposition und einem Operatorgraphen sind schematisch in Abbildung 5.1 dargestellt. Wenn man einen Operatorgraphen zur Merkmalsberechnung einsetzt, kommt zu der Auswahl einer Eingabeansicht und der Operatorkontexte im Prinzip noch die Auswahl der Zielansicht, in die die generierten Objekte eingefügt werden. Für die Auswertung eines Operatorgraphen muss die Zielansicht aber nicht bekannt sein; es ist vielmehr die Aufgabe des Rahmenverfahrens, das die Auswertung eines Operatorgraphen anstößt, die berechnete Ausgabe weiterzuverwerten.

Die explizite Definition der Verknüpfungen von Operatorgraph und Kontextkomposition ermöglicht es, einen Operatorgraph auf alle Kontextkompositionen anzuwenden, für die eine passende Kontextzuordnung definiert werden kann. Wenn man die Berechnung musikalischer Merkmale monolithisch implementiert, werden häufig Annahmen über die Repräsentation der Kompositionen fest in die Implementierung hineincodiert. Solche Annahmen sind im allgemeinen schwer auffindbar und änderbar, wenn der betrachtete Musikstil sich ändert. Für die Massengenerierung von Lernmustern ist der hier entwickelte Ansatz geeigneter, der Repräsentation und Merkmalsberechnung einerseits von der Implementierung eines Rahmenverfahrens andererseits trennt.

5.4 Anwendungen von Operatorgraphen

In Abschnitt 5.3 wurde beschrieben, wie man einen Operatorgraphen für eine bestimmte Eingabe auswertet. Nun werden anhand einiger Beispiele Einsatzmöglichkeiten von Operatorgraphen vorgestellt.

5.4.1 Durchlaufstrategie

Operatorgraphen werden in verschiedenen Rahmenverfahren eingesetzt: zur Berechnung von Ansichten, von Lernmustern oder zur Auswertung von neuronalen Klassifikatoren. Die Auswertung eines Operatorgraphen für eine Menge von Stücken wird vom Rahmenverfahren angestoßen und folgt immer demselben Schema:

- Operatorgraph definieren (auswählen oder erzeugen)
- Corpus wählen. Alle Kompositionen bestehen aus der gleichen Typen von Ansichten.
- Eingabeansicht wählen.
- Kontext der Operatoren festlegen.
- Auswertbarkeit des Operatorgraphen prüfen
- Durchlaufstrategie (Startposition, Iterator, Abbruchbedingung) für eine Ansicht festlegen.
- Für jede Komposition im Corpus
  - Anhand der Durchlaufstrategie durch Ansicht iterieren. Für jedes aktuelle Segment der Durchlaufstrategie:
    - Operatorgraph auswerten
    - Ergebnis des Operatorgraphen verwerten (z.B. in Zielansicht oder Patterndatei einfügen)

Zunächst legt man fest, welcher Operatorgraph ausgewertet werden soll. Ein Operatorgraph kann vom Benutzer vorgegeben oder durch einen Algorithmus aus einer Menge von Kandidaten gewählt werden.

Nun werden die Daten ausgewählt, auf die der Operatorgraph angewendet werden soll. Bei einer musikalischen Anwendung sind die Daten als Corpus verwandter Musikstücke gegeben. Es ist sinnvoll, alle Stücke durch die gleichen Typen von Ansichten zu repräsentieren, damit sie einheitlich behandelt werden können.

Die Zuordnungen zwischen dem Operatorgraphen und Daten werden wie in Abbildung 5.1 skizziert angegeben, indem eine Eingabeansicht ausgewählt wird und den Operatoren passende Kontextansichten zugeordnet werden. Für alle Stücke des Corpus sollte man die gleiche Zuordnung wählen, damit alle später erzeugten Objekte miteinander vergleichbar sind.

Im nächsten Schritt prüft man die Auswertbarkeit des Operatorgraphen für die gewählten Festlegungen. Wenn für alle Stücke die gleichen Zuordnungen gewählt wurden, muss die Auswertbarkeit nur einmal untersucht werden. Ist die Prüfung positiv, so wird der Operatorgraph auf alle Daten angewendet.

Es wird eine Strategie festgelegt, mit der jede Komposition durchlaufen werden soll. Sie setzt sich zusammen aus einer Startposition in der Eingabeansicht, einem Steueriterator, der alle zu bearbeitenden Segmente durchläuft, und einer Abbruchbedingung. Die Durchlaufstrategie wird durch das Tripel $(\langle \mathit{Startposition} \rangle ; \langle \mathit{Iterator} \rangle ; \langle \mathit{Abbruchkriterium} \rangle)$ beschrieben. Bei der Berechnung neuer Ansichten oder der Generierung von Lernmustern durchläuft man alle Segmente der Eingabeansicht in ihrer zeitlichen Reihenfolge. Diese Strategie wird mit $(\mathbf{it}_1; \mathbf{it}_{+1}; \perp)$ abgekürzt. Der erste Iterator springt unabhängig von seiner Eingabe zum ersten Segment der Ansicht, der zweite Iterator bewegt sich von seiner Eingabe ausgehend ein Segment nach rechts. Wenn der Iterator das Ergebnis $\perp$ liefert, bricht der Durchlauf ab. Bei dieser trivialen Strategie durchläuft man alle Segmente in ihrer zeitliche Reihenfolge. Bei der evolutionären Melodiegenerierung in Kapitel 6 werden andere Durchlaufstrategien mit Zufallskomponente eingesetzt.

Gemäß der Durchlaufstrategie wird der Operatorgraph nun auf alle Eingabeansichten aller Kompositionen des Corpus angewendet. Die vom Operatorgraph erzeugten Objekte werden vom Rahmenverfahren geeignet weiterverarbeitet. Soll eine neue Ansicht erzeugt oder eine vorhandene Ansicht verändert werden, werden die Ergebnisse in die Zielansicht eingefügt. Sollen Lernmuster berechnet werden, wird das erzeugte Muster zu einer Lernmusterdatei hinzugefügt.

5.4.2 Berechnung musikalischer Merkmale

Die Hauptmotivation bei der Entwicklung von Operatorgraphen war der Wunsch nach einem Werkzeug zur Berechnung neuer Merkmale, das auf die Eigenheiten musikalischer Zeitreihen zugeschnitten ist. Nachdem nun alle Definitionen unseres Ansatzes bekannt sind, rekapitulieren wir den Zusammenhang zwischen dem hier entwickelten Instrumentarium und den Grundbegriffen der Datenanalyse (Statistik).

Präzisierung des Merkmalsbegriffs. Die Grundgesamtheit der untersuchten Objekte[3] wird hier von einem Corpus stilistisch zusammengehöriger Melodien gebildet. Ein Merkmal bezeichnet eine untersuchungsrelevante Eigenschaft eines Objekts. In diesem Ansatz entspricht ein Merkmal einer Abbildung, die durch einen Operatorgraphen definiert wird. Dabei ist es unwichtig, auf welche Weise der Operatorgraph sein Ergebnis berechnet. Im allgemeinen gibt es mehrere Operatorgraphen, die dasselbe Merkmal berechnen. Ein Merkmal

[3] Hier sind nicht die musikalischen Objekte einer Erweiterbaren Zeitreihengrammatik, sondern reale Objekte gemeint.

kann Merkmalswerte annehmen, die zum Wertebereich des Merkmals gehören. Die Typen einer Erweiterbaren Zeitreihengrammatik dienen hier als Wertebereiche der Merkmale und Operatorgraphen. Die Elemente (Objekte) des Typs sind die Merkmalswerte eines Merkmals mit diesem Wertebereich. Ein Merkmal einer Grundgesamtheit wird erfasst, indem die zugehörige Eigenschaft ihrer Objekte gemessen wird. Ein Operatorgraph ist ein Messinstrument, mit dem ein Merkmal einer Melodie erfasst werden kann.

Neben den Merkmalen, die mit Hilfe von Operatorgraphen aus einer bereits vorhandenen Repräsentation einer Melodie berechnet werden, gibt es Eingabemerkmale, die eine Melodie auf elementare Weise (z.B. durch Tonhöhen und Dauern) beschreiben. Die Repräsentation der Melodien mit Elementarmerkmalen ist das Ausgangsmaterial, von dem alle weiteren Merkmalsberechnungen ausgehen. Die Repräsentation einer Melodie mit elementaren Merkmalen ist bereits eine Interpretation der „realen“ Melodie, da bereits hier bestimmte Aspekte einer Melodie durch die Ausgestaltung der Elementarmerkmale betont werden.

Ein Merkmal wird nicht nur durch seinen Wertebereich, sondern auch durch die Art seiner Messung charakterisiert. Zwei Merkmale, die den gleichen Wertebereich haben, sich aber auf Informationen mit unterschiedlicher zeitlicher Zuordnung stützen, sind unterschiedliche Merkmale. Beim Intervall zwischen dem aktuellen und dem vorhergehenden Ton einerseits und andererseits dem aktuellen Ton und dem nächsten Ton einer Melodie handelt es sich beispielsweise um Merkmale, die sich nur durch den Zeitbezug ihres Kontexts unterscheiden. Während man zur Berechnung des ersten Merkmals einen Schritt in die Vergangenheit iteriert, iteriert man zur Berechnung des zweiten Merkmals einen Schritt in die Zukunft. Aus der Perspektive eines vorgegebenen Tons hat ein Intervall, das zu diesem Ton hinführt, eine andere musikalische Funktion hat als ein Intervall, das die Melodie fortsetzt. Es ist daher auch musikalisch plausibel, hier von zwei verschiedenen Merkmalen zu sprechen.

Musikalischer Kontext eines Merkmals. Ein allgemeineres Unterscheidungskriterium für Merkmale ist der musikalische Kontext, der in die Berechnung eines Merkmalswerts eingehen. Ein *lokales Merkmal* charakterisiert ein einzelnes Eingabesegment und seinen zeitnahen Kontext. Ein *melodieglobales Merkmal* charakterisiert ein ganzes Musikstück. Ein *corpusglobales Merkmal* oder *Corpusmerkmal* beschreibt eine Eigenschaft eines ganzen Corpus.

Die meisten bisher beschriebenen Merkmale sind lokale Merkmale: Intervalle, Konturen, Harmonien, Motive. Melodieglobale Merkmale sind Eigenschaften wie die Taktart oder die Tonart einer Melodie. Die Unterscheidung zwischen lokalen und melodieglobalen Merkmalen ist aber nicht eindeutig möglich. Wenn eine Melodie eine konstante Taktart hat, erscheint diese als melodieglobales Merkmal. Enthält eine Melodie Taktwechsel, so ist die Taktart ein lokales Merkmal der Melodie. In einem Corpus, der Melodien mit Taktwechsel enthält, kann man die Taktart einer Melodie ohne Taktwechsel als ein lokales Merkmal ansehen, das sich über den ganzen Verlauf der Melodie erstreckt. Von einem praktischen Standpunkt aus betrachtet ist es zweckmäßig, melodieglobale Merkmale in einer eigenen Ansicht zu speichern, die sich über den gesamten Verlauf des Stücks erstreckt. So sind auch melodieglobale Merkmale für die Operatorgraphen in derselben Weise erreichbar wie lokale Merkmale. Ein Beispiel für ein corpusglobales Merkmal ist die größte rhythmische Einheit, mit der sich alle Musikstücke darstellen lassen. Sie wird benötigt, um die Auflösung der Zeitachse zu wählen, die bei der Repräsentation von Musikstücken als zeitliche Referenz dient.

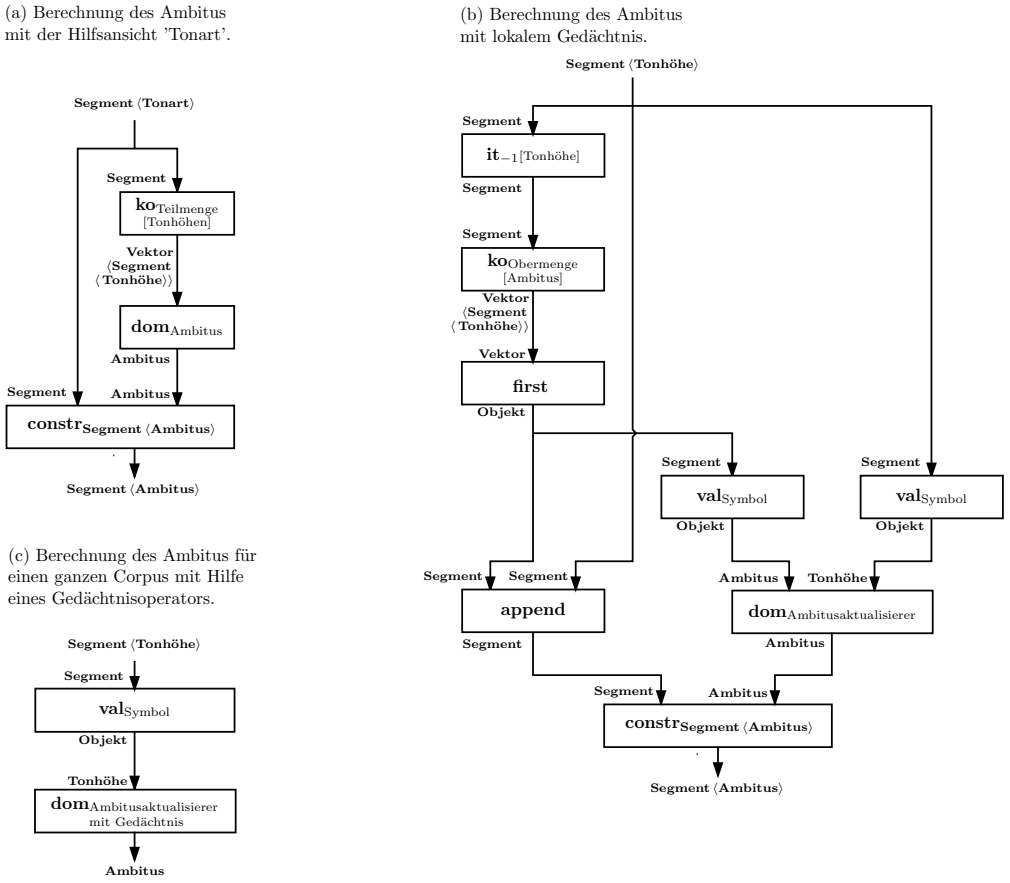


Abbildung 5.25: Operatorgraph zur Berechnung des Ambitus für unterschiedliche Daten

Interaktion zwischen Rahmenverfahren und Operatorgraphen. In Abschnitt 5.4.1 wurde die prinzipielle Herangehensweise bei der Auswertung von Operatorgraphen beschrieben. Das Rahmenverfahren, das die Auswertung eines Operatorgraphen initiiert, ist dafür verantwortlich, das Ergebnis in die Repräsentation einzufügen, auf der der Operatorgraph operiert. Die folgenden Beispiele illustrieren verschiedene Möglichkeiten, die Ergebnisse eines Operatorgraphen weiterzuverwenden.

Beispiel: Lokales Merkmal. Die Berechnung eines lokalen Merkmals wurde bereits weiter oben am Beispiel des metrischen Gewichts diskutiert (Abbildungen 5.23 und 5.24). Das Rahmenverfahren fügt das Ergebnis des Operatorgraphen nach jeder Auswertung in die Zielansicht des Musikstücks ein, auf der Operatorgraph gerade operiert.

Beispiel: Melodieglobales Merkmal. Abbildung 5.25(a) zeigt, wie man ausgehend von einer melodieglobalen Hilfsansicht den Ambitus für eine ganze Melodie berechnet. Der Operatorgraph extrahiert ausgehend von der Ansicht „Tonart“, die sich hier über den ganzen Ver-

lauf des Stücks erstreckt, mit dem Koinzidenzoperator $\mathbf{ko}_{\text{Teilmenge}}$[Tonhöhe] alle Tonhöhensegmente aus der Ansicht „Tonhöhe“. Der Wissensoperator $\mathbf{dom}_{\text{Ambitus}}$ berechnet daraus ein **Ambitus**-Objekt, das der Konstruktor $\mathbf{constr}_{\mathbf{Segment}\,\langle\mathbf{Ambitus}\rangle}$ mit der Zeitinformation des Tonhöhensegments zu einem Segment zusammensetzt.

Als Durchlaufstrategie wählt man $(1; \mathbf{it}_{+1}, \perp)$. Man beginnt beim ersten Segment, iteriert nach jeder Berechnung um ein Segment nach rechts und stoppt, wenn der Iterator ein undefiniertes Ergebnis liefert. Im vorliegenden Fall enthält die Eingabeansicht „Tonart“ nur ein Segment, so dass der Durchlauf nach einer Auswertung des Operatorgraphen beendet ist.

Während in Beispiel (a) alle Tonhöhensegmente gleichzeitig ausgewertet werden, zeigt Abbildung 5.25(b), wie man schrittweise Information akkumulieren kann, indem man eine Ansicht als „lokales Gedächtnis“ verwendet. Die Idee ist, den Ambitus der bisher durchlaufenen Tonhöhensegmente in einem Segment zu speichern, das sich vom Anfang des Stücks bis zum aktuellen Zeitpunkt erstreckt. Wenn der Operatorgraph den nächsten Ton bearbeitet, extrahiert er mit den Operatoren $\mathbf{it}_{-1}$, $\mathbf{ko}_{\text{Obermenge}}$[Ambitus] und **first** das Ambitussegment der früheren Töne aus der Zielansicht „Ambitus“. Mit dem Operator **append** wird die Zeitinformation des Eingabesegments wird an das Ambitussegment angehängt, das die Dauer aller bisher betrachteten Töne umfasst. Der Ambitusaktualisierer $\mathbf{dom}_{\text{Ambitusaktualisierer}}$ bildet aus dem bisherigen Ambitus und der Tonhöhe des aktuellen Eingabesegments den neuen Ambitus, der bei der Konstruktion eines neuen Segments mit dem Konstruktor $\mathbf{constr}_{\mathbf{Segment}\,\langle\mathbf{Ambitus}\rangle}$ als Wert in das verlängerte Segment eingesetzt wird. Das Rahmenverfahren ersetzt das Segment in der Ambitusansicht durch das Ergebnis des Operatorgraphen, der dann mit der Auswertung des nächsten Segments fortfährt.

Die Operatorgraphen (a) und (b) unterscheiden sich in der Weiterverarbeitung des berechneten Ergebnisses. Während das Rahmenverfahren bei Operatorgraph (a) ein neues Segment berechnet und in der Zielansicht speichert, wird bei Operatorgraph (b) das Ambitussegment der Zielansicht bei jeder Auswertung ersetzt. Bei (b) greift der Operatorgraph also auf die Zielansicht zu, noch während diese berechnet wird.

Abbildung 5.25(c) zeigt einen dritten, sehr einfachen Operatorgraphen, der ebenfalls den Ambitus einer Melodie berechnet. Der Operatorgraph durchläuft die Tonhöhenansicht und speichert in einem Wissensoperator mit Gedächtnis den Ambitus der bisher gesehenen Töne. Bei diesem Operatorgraphen muss der Wissensoperator mit Gedächtnis vor Beginn der Auswertung einer Melodie zurückgesetzt werden. Das Ergebnis des Operatorgraphen hängt vom inneren Zustand eines Operators ab, so dass eine Abhängigkeit zwischen aufeinanderfolgenden Auswertungen eines Operatorgraphen entsteht. Da das Verhalten von Operatorgraphen durch die Verwendung von inneren Zuständen für einen Anwender schnell unübersichtlich wird, wurden Operatoren mit Gedächtnis in dieser Arbeit nicht weiterverfolgt.

Beispiel: Corpusglobales Merkmal. Bevor man Merkmale entwerfen kann, die sich zum Lernen melodischer Stile eignen, ist es sinnvoll, globale Eigenschaften des untersuchten Melodiecorpus zu analysieren. Beispielsweise kann man die Häufigkeit der Werte eines Wertebereichs in einem Corpus untersuchen, um bei der Generierung von Lernmustern einen kompakten, an den untersuchten Stil angepassten Wertebereich zugrundezulegen. Wenn man zwischen dem Wunsch nach leistungsfähigen Netzen und nach einer genauen Modellierung der Ausgangsdaten abwägen muss, hilft eine Häufigkeitsanalyse eines Wertebereichs, neben

nicht benötigten auch selten verwendete Werte aus einem angepassten Wertebereich auszuschließen. Die neuronalen Netze des Harmonisierungssystems HARMONET werden z.B. für die Harmonisierungstile von J.S. Bach und M. Reger mit unterschiedlichen harmonischen Funktionen trainiert.

Um ein Corpusmerkmal zu berechnen, durchläuft man die Eingabeansichten aller Stücke in einem Corpus, wobei ein Operatorgraph für jedes Eingabesegment ausgewertet wird. Während bisher die berechnete Information wieder lokal im betrachteten Musikstück abgelegt wurde, steht man nun vor dem Problem, Information über mehrere Stücke übergreifend zu akkumulieren. Der bereits diskutierte Operatorgraph mit Gedächtnis aus Abbildung 5.25(c) bietet hier eine elegante Lösung, da er in der Lage ist, Information über den Ambitus über den ganzen Corpus hinweg akkumuliert.

Segmentierung von Melodien. Ebenfalls unter die Berechnung von Merkmalen fällt die Segmentierung von Melodien (vgl. Abschnitt 3.4.2). In einer Diplomarbeit [79] wurde gezeigt, wie man den *Grouper*-Algorithmus von D. Temperley [80] durch Auswertung einer Menge von Operatorgraphen simulieren kann. Die Segmentierungskriterien, die in Form von Präferenzregeln vorliegen, werden dadurch austauschbar, so dass auch andere als die von Temperley vorgesehenen Merkmale betrachtet werden können.

5.4.3 Berechnung von Lernmustern

Die Berechnung von Lernmustern unterscheidet sich von der Berechnung neuer Ansichten nur durch den Ausgabetyp des Operatorgraphen. Während bei der Berechnung einer Ansicht vom Typ **X** ein Objekt vom Typ **Segment** $\langle$**X**$\rangle$ als Ausgabe des Operatorgraphen erwartet wird, muss ein Lernmuster numerisch sein. Als Ausgabetypen eines Operatorgraphen kommen daher Vektoren mit Elementen von **Int**, **Rational** und **Bool** in Frage. Typischerweise befinden sich in der Ausgabeschicht eines Operatorgraphen zur Lernmusterberechnung Codierungsoperatoren. Das Rahmenverfahren zur Lernmusterberechnung setzt alle ausgegebenen numerischen Vektoren zu einem Muster zusammen.

Wie bei der Berechnung von Ansichten durchläuft auch hier das Rahmenverfahren alle Stücke des Corpus und iteriert durch alle Eingabeansichten. Die Ausgabe eines Operatorgraphen wird nicht in die Repräsentation, sondern in eine Lernmusterdatei geschrieben.

Beispiel. Wenn man in Abbildung 5.15 den Neuro- und den Decodierungsoperator entfernt, erhält man den Operatorgraphen, der die Lernmuster für das `hf1`-Netz in HARMONET generiert. Aufgabe des `hf1`-Netzes ist die Vorhersage einer harmonischen Funktion bei der Melodieharmonisierung.

Als Eingabeansicht wird die Tonhöhenansicht der Melodie verwendet, die harmonisiert werden soll. Der Operatorgraph stellt Lernmuster mit Codierungen der folgenden Merkmale zusammen: vorhergehender, aktueller und nächster Melodieton ($s(-1)$, $s(0)$, $s(1)$); die harmonischen Funktionen vor der aktuellen Position ($hf(-3)$, $hf(-2)$, $hf(-1)$); die harmonischen Umkehrungen vor der aktuellen Position ($hu(-3)$, $hu(-2)$, $hu(-1)$); die harmonische Dissonanz vor der aktuellen Position ($hd(-1)$); die Phrasenposition und die Betonungsinformation für die aktuelle Position (*phr* und *str*). Wenn man statt der Rasterdarstellung in

Abbildung 5.13 die ereignisorientierte Repräsentation für die analysierten Choräle verwendet, erhält man mit demselben Operatorgraphen Lernmuster, deren zeitlicher Kontext ein anderer ist als in HARMONET. Durch eine andere Repräsentation der Eingabe wird hier der Kontext variiert, der bei der Berechnung von Lernmustern berücksichtigt wird.

5.4.4 Auswertung von Klassifikatoren

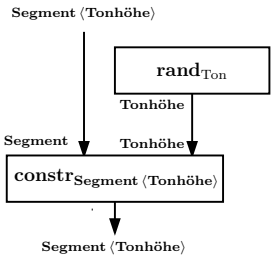
Eine weitere Anwendung von Operatorgraphen ist die Auswertung von Klassifikatoren. Während das Training neuronaler Netze nur technische Kriterien wie die Klassifikationsgüte optimiert, ist man bei musikalischen Anwendungen daran interessiert, auch die musikalische Leistungsfähigkeit einer Kombination neuronaler Netze auszuwerten. Zu diesem Zweck setzt man neuronale Netze zum Komponieren neuer Musikstücke ein. Die Stiltreue der generierten Melodien ist dann ein musikalisches Bewertungskriterium für das Gesamtmodell.

Operatorgraphen sind für diese Aufgabe geeignet, weil sie den Aufruf von Klassifikatoren mit Neurooperatoren (Abschnitt 5.2.3.4) mit regelbasierten Anteilen in ein Modell integrieren können. Die Simulation der improvisatorischen Kompositionssysteme HARMONET und MELONET wird als Beispiel herangezogen.

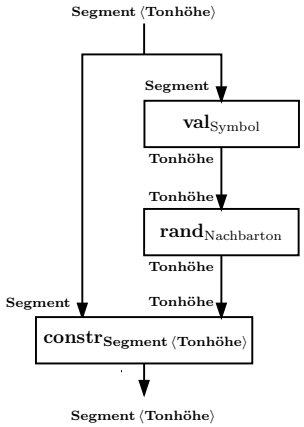
HARMONET. Aufgabe des lernbasierten Systems HARMONET [32] ist es, Choralmelodien mit Hilfe neuronaler Netze vierstimmig zu harmonisieren. Die neuronalen Netze wurden in einem existierenden Choralstil trainiert. Abbildung 5.12 zeigt den Anfang eines vierstimmigen Choral von J.S. Bach. Die Aufgabenstellung wurde in verschiedene Schritte zerlegt, von denen hier nur die Bestimmung der harmonischen Symbole näher beleuchtet wird. In Abbildung 5.13 sieht man eine Choralmelodie, die harmonisiert werden soll, die Repräsentation der Melodie durch die Ansichten „Tonhöhe“, „Taktart“ und „Phrase“ sowie eine Ansicht der harmonischen Symbole, die bereits harmonische Symbole für die ersten drei Töne enthält. Der aktuelle Ton h ist in den Noten grau unterlegt. Für das zugehörige Segment wird mit dem (schematisch dargestellten) Operatorgraphen in Abbildung 5.14 ein harmonisches Symbol berechnet. Die Auswertung der neuronalen Klassifikatoren verbirgt sich in den grau unterlegten Kästen der Abbildung, von denen die „Bestimmung der harmonischen Funktion“ in Abbildung 5.15 detailliert wird. In diesem Operatorgraphen wird zunächst musikalische Information aus dem Kontext der Eingabe zusammengestellt und als Muster codiert. Das Muster dient als Eingabe für den Neuro-Operator, dessen Ausgabe vom nachfolgenden Decodierer wieder in ein musikalisches Symbol zurückgewandelt wird. Dieses Symbol wird mit der Zeitinformation des Eingabesegments (im Bild weggelassen) in die Zielansicht der harmonischen Symbole eingefügt.

MELONET. Ähnlich erfolgt der Aufruf von Klassifikatoren beim Melodieumspielungssystem MELONET [37]. Gegeben ist eine Choralmelodie in Viertelnoten, die im Stil barocker Choralvariationen mit Sechzehntelnoten umspielt werden soll. Abbildung 5.16 zeigt den Anfang einer Choralmelodie und eine Umspielung. Wie HARMONET ist auch MELONET modularisiert (Abbildung 5.17). Bei der Melodieumspielung einer Viertelnote legt zunächst ein sogenanntes „Supernetz“ die Motivklasse fest, die als Schablone für das zu erzeugende Sechzehntelmotiv verwendet werden soll. Das „Referenztonnetz“ und das „Subnetz“ erzeugen dann aus der zuvor ermittelten Motivklasse eine Sechzehntelumspielung. Mit dem „Motivklassifikator“ wird die tatsächliche Motivklasse der generierten Umspielungstöne analysiert,

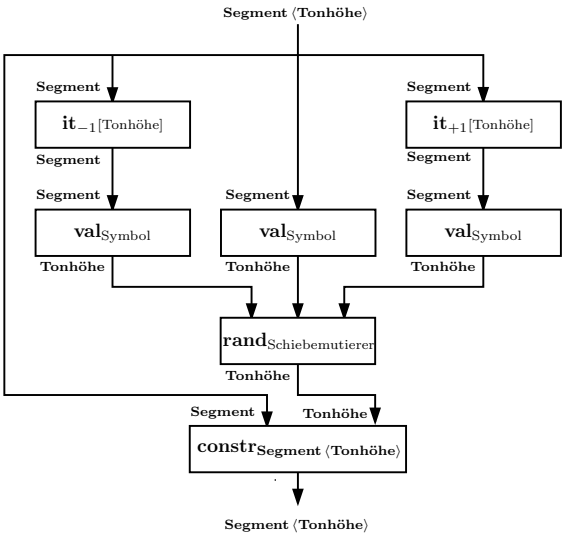
(a) zufälliger Ton



(b) benachbarter Ton



(c) zufälliger benachbarter Ton zwischen vorhergehendem und nachfolgendem Ton



(d) zufälliger benachbarter Ton auf harmonischem Gitter der aktuellen Harmonie

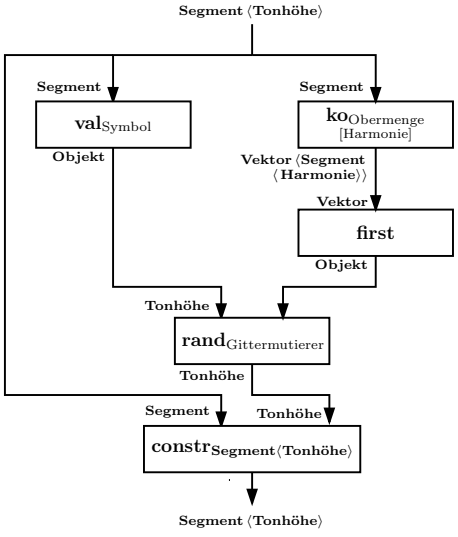
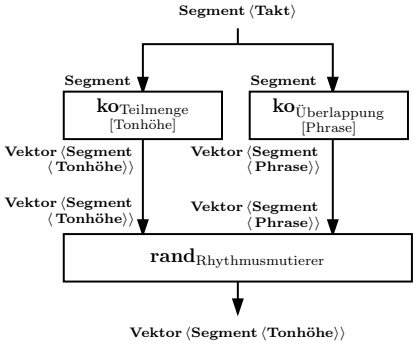


Abbildung 5.26: Operatorgraphen zur Mutation von Melodien

damit die neuronalen Netze im nächsten Umspielungsschritt auf korrekte Motivklasseninformation zurückgreifen können. Die Systemkomponenten von MELONET werden von den Operatorgraphen in den Abbildungen 5.18, 5.19, 5.20 und 5.21 simuliert, von denen die letzten drei neuronale Klassifikatoren auswerten.

(e) Der Rhythmus der Töne eines Takts wird zufällig mutiert, wobei alle Phrasengrenzen unverändert bleiben.



(f) Die Grenze zwischen zwei aufeinanderfolgenden Tönen wird zufällig verschoben, wobei alle Phrasen unverändert bleiben.

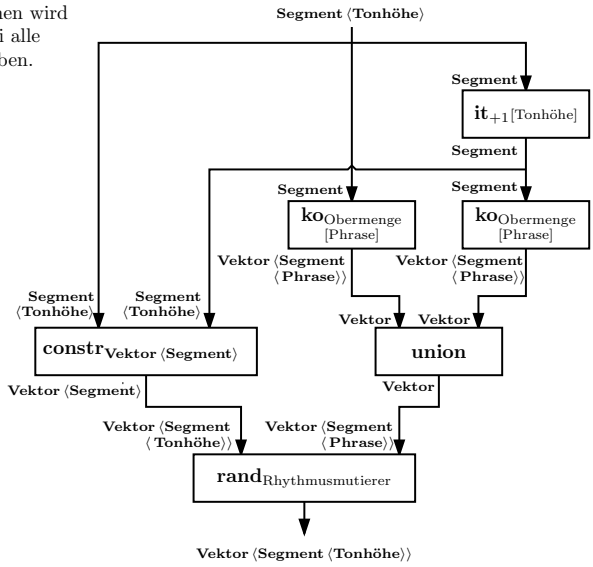


Abbildung 5.27: Operatorgraphen zur Mutation von Melodien

5.4.5 Evolutionäre Generierung von Melodien

Die Grundidee bei der Generierung von Melodien mit evolutionärer Suche besteht darin, aus einer Population von Melodien durch Mutation und Rekombination neue Melodien zu erzeugen. Die Melodien werden mit neuronalen Netzen bewertet, die im vorgegebenen Melodiestil trainiert wurden. Die Melodien mit einer guten Bewertung werden ausgewählt und bilden die Population der nächsten Generation, die wiederum verändert und bewertet wird.

Wenn man die evolutionäre Generierung von Melodien mit Operatorgraphen realisiert, treten Operatorgraphen an zwei Stellen des Verfahrens auf. Einerseits werden sie benötigt, um die (mit Ansichten repräsentierten) Melodien der Population zu bewerten. Dies erfolgt nach dem Schema, das in Abschnitt 5.4.4 für den Aufruf von neuronalen Netzen mit Operatorgraphen beschrieben wurde. Die interessantere Anwendung von Operatorgraphen tritt bei

der Mutation von Melodien auf, da hier eine Zufallskomponente ins Spiel kommt.

Mutation mit Operatorgraphen. Um eine Melodie zu mutieren, braucht man einen Operatorgraphen, der von der aktuellen Repräsentation ausgehend ein oder mehrere Segmente verändert. Da bei der Evolution Mitglieder der Population nach und nach verändert werden sollen, ist es wenig sinnvoll, eine ganze Ansicht einer Melodie bei der Mutation mit einem Operatorgraphen zu transformieren. Man wählt stattdessen einen Iterator mit größerer Schrittweite als 1 oder eine zufallsgesteuerte Durchlaufstrategie. Bei der zufallsgesteuerten Durchlaufstrategie entscheidet der Iterator sich bei jeder Auswertung zufällig für eine Schrittweise. Lässt man sowohl negative als auch positive Schrittweiten zu, darf der Iterator sich sowohl nach links als auch nach rechts bewegen. In diesem Fall muss in der Durchlaufstrategie eine Abbruchbedingung festlegen, die den Durchlauf terminieren lässt. Beispielsweise kann man die Mutation nach einer festen Anzahl von Schritten des Iterators beenden. Die gewählte Durchlaufstrategie bestimmt, wieviel Information des vorgegebenen Elements bei einer Mutation verändert wird.

Die Abbildungen 5.26 und 5.27 zeigen verschiedene Operatorgraphen, die eine Melodie mutieren. Die Operatorgraphen in Abbildung 5.26 mutieren Ansichten vom Typ „Tonhöhe “. In Operatorgraph (a) wird ein Tonhöhensegment durch ein Segment mit der gleichen Zeitinformation und einer zufällig gewählten anderen Tonhöhe ersetzt. Während dieser Operatorgraph den Kontext der Eingabe nicht berücksichtigt, wird in Operatorgraph (b) eine Tonhöhe mit größerer Wahrscheinlichkeit ausgewählt, je näher sie beim Eingabeton liegt. Auf diese Weise verändert man die Intervalle der Eingabemelodie nur wenig. Auch Operatorgraph (c) berücksichtigt den Eingabeton und seine Nachbarn. Mit dem sogenannten „Schiebemutierer“ verändert er den Eingabeton so, dass die Kontur der benachbarten Intervalle erhalten bleibt. In Operatorgraph (d) wird neben dem Ton, der mutiert werden soll, der aktuelle harmonische Kontext berücksichtigt. Als mutierter Ton wird ein Ton des harmonischen Kontexts gewählt. Eine solche Mutation könnte bei der Modellierung von harmonieorientierten Melodiestilen wie z.B. Jodlern in Betracht kommen.

Wenn man die Zeitinformation einer Melodie nicht verändert, ist es also einfach, die Werte in den Segmenten einer Ansicht unter Berücksichtigung des Kontexts zu mutieren. Doch wie kann man den Rhythmus der musikalischen Ereignisse ändern, ohne dass Pausen oder unerwünschte Überlappungen aufeinanderfolgender Ereignisse entstehen?

Eine Möglichkeit besteht darin, mehrere aufeinanderfolgende Ereignisse gleichzeitig zu betrachten. Operatorgraph (e) in Abbildung 5.27 verändert die Länge der Noten in einem Takt. Ausgehend von der Taktansicht extrahiert der Operatorgraph alle Tonhöhensegmente des aktuellen Takts sowie die Phrasen, die einen nicht-leeren Schnitt mit dem aktuellen Takt haben. Mit dieser Information kann der Rhythmusmutierer die Grenzen der Töne innerhalb eines Takts so verschieben, dass jeder Ton nach der Mutation noch immer in einer Phrase enthalten ist. Man nimmt dabei an, dass ein Ton nicht Teil mehrerer nicht-überlappender Phrasen sein kann. Wenn die Taktansicht nicht zur Verfügung steht, kann man wie in Operatorgraph (f) auch die Grenze zwischen aufeinanderfolgenden Tönen ändern. Auch hier werden die Phrasengrenzen bei der Mutation berücksichtigt.

Die Berücksichtigung der Phrasengrenzen bei der Mutation ist dadurch motiviert, dass man bei der Generierung von Melodien musikalische Vorgaben machen möchte, die im Verlauf

der Evolution nicht verändert werden dürfen. Eine solche Vorgabe könnte die Phrasierung der Melodien sein. Im System MeloGeNet wird der Melodieanfang fest vorgegeben, der im weiteren Verlauf der Melodie als motivisches Material dient.

Wenn man eine Ansicht einer Melodie mutiert, können dadurch Folgeänderungen nötig werden. Werden Tonhöhen einer Melodie verändert, so müssen alle Ansichten, die bei ihrer Erstellung Tonhöhen verwenden (z.B. eine Intervallansicht), neu berechnet werden. Die Behandlung solcher Abhängigkeiten und die Rekombination von Melodien wird in Kapitel 6 diskutiert.

5.5 Operatorgraph und Zeitreihengrammatik

In Kapitel 4 wurde eine Repräsentation für diskrete Zeitreihen entwickelt, die es einem Anwender erlaubt, gezielt problemspezifisches Wissen zu integrieren, um die Repräsentation an das untersuchte Problem anzupassen. Darauf aufbauend wurde in diesem Kapitel mit Operatorgraphen ein neuer Ansatz zur Transformation der repräsentierten Daten vorgestellt.

Im Vergleich zur monolithischen Implementierung musikalischer Transformationen bietet die Verwendung von Operatorgraphen den methodisch wichtigen Vorteil, dass Operatorgraphen unabhängig von ihrer Struktur immer auf dieselbe Weise ausgewertet werden. Es ist daher möglich, Algorithmen mit frei austauschbaren, von Operatorgraphen dargestellten Merkmalen zu implementieren, ohne dass auf die tatsächliche Beschaffenheit der Merkmale Bezug genommen werden muss. Operatorgraphen bilden damit eine einheitliche Schnittstelle zwischen einem generischen Algorithmus einerseits und einer problembezogenen Repräsentation musikalischer Daten andererseits.

In der Erweiterbaren Zeitreihengrammatik und bei den Operatorgraphen wird die Schnittstelle zwischen dem generischem Rahmenverfahren und den problembezogenen Merkmalen dadurch realisiert, dass alle Datenstrukturen gleichzeitig auf zwei Abstraktionsebenen modelliert werden: Jedes von einer Erweiterbaren Zeitreihengrammatik erzeugte musikalische Objekt gehört aufgrund der Typhierarchie einer Erweiterbaren Zeitreihengrammatik einerseits zum abstrakten Typ **Objekt** und andererseits zu einem Typ mit bekannter Struktur wie etwa **Takt**. In einem Operatorgraphen kann jeder Operator zugleich als allgemeiner Operator **op** und als Operator mit spezifischer Funktionalität (wie z.B. ein Iterator) betrachtet werden. Während das Rahmenverfahren nur eine abstrakte Sicht auf die gekapselten Datenstrukturen hat, stehen bei der Auswertung eines Operators alle Daten der manipulierten Objekte zur Verfügung.

Wir haben gesehen, dass es verschiedene Anwendungen von Operatorgraphen gibt: die Berechnung von musikalischen Merkmalen, von Lernmustern und die Analyse von Musikstücken. Damit ließen sich komplexe Systeme wie HARMONET und MELONET als Beispiele des neuen Modellierungsansatzes identifizieren und anschaulich visualisieren. Im nächsten Kapitel wird an realen Daten gezeigt, wie man Operatorgraphen bei der Selektion stilrelevanter Merkmale und bei der evolutionären Melodiegenerierung einsetzen kann.

Kapitel 6

Musikalische Stilmodellierung

Die Repräsentation diskreter Zeitreihen aus Kapitel 4 und die Operatorgraphen aus Kapitel 5 werden nun auf Problemstellungen der musikalischen Stilmodellierung angewandt.

Merkmalsselektion. In Abschnitt 6.1 steht die musikalische Stilanalyse im Mittelpunkt. Als grundlegende Aufgabenstellung werden zunächst mit Hilfe von Merkmalsselektionsverfahren die Eingabemerkmale lernbasierter Systeme im Hinblick auf ihre Relevanz für ein Ausgabemerkmal optimiert (Abschnitt 6.1.1). Darauf aufbauend wird die Unterscheidung zweier Melodiestile (Abschnitt 6.1.2) und die Zuordnung einer Melodie zu einem von mehreren vorgegebenen Stilen untersucht (Abschnitt 6.1.3).

Melodiegenerierung. Die zweite in diesem Kapitel behandelte Problemstellung ist die Melodiegenerierung. Mit Hilfe eines evolutionären Suchverfahrens wird untersucht, ob sich mit merkmalsoptimierten Klassifikatoren Melodien generieren lassen, die menschlichen Experten als stiltreu erscheinen (Abschnitt 6.2). Dabei werden Operatorgraphen für die Transformation und die Bewertung der Melodien eingesetzt.

Merkmalssuche. Abschließend wird an einem Beispiel gezeigt, dass die Manipulation von Operatorgraphen es erlaubt, Verfahren zur Generierung stilrelevanter Merkmale anzugeben. Im Gegensatz zu den Merkmalsselektionsverfahren werden hier nicht bereits definierte Merkmale geeignet kombiniert, sondern die Berechnung der Merkmale selbst wird bei der Suche verändert (Abschnitt 6.3).

Operatorgraphen als Variable. Eine Gemeinsamkeit zwischen allen hier entwickelten Verfahren besteht darin, dass die verwendeten musikalischen Merkmale austauschbar sind, da sie immer mit Operatorgraphen berechnet werden. Im Vergleich zu Stilmodellierungsverfahren, die die Verwendung bestimmter musikalischer Merkmale vorschreiben, handelt es sich bei den hier behandelten Verfahren um Familien von Verfahren, die mit Merkmalen parametrisiert sind.

Automatische Optimierung. Die hier vorgestellten Optimierungsverfahren sehen keine Interaktion mit einem Anwender vor, sondern ziehen objektive Gütekriterien wie z.B. die

Klassifikationsrate neuronaler Modelle heran, denn in dieser Arbeit steht die Frage im Mittelpunkt, welche musikalischen Merkmale *gestützt auf Originalbeispiele* für einen Musikstil als charakteristisch anzusehen sind. Damit unterscheidet sich dieser Ansatz von der interaktiven Generierung von Musikstücken. In [1] werden z.B. Harmonisierungen mit einem interaktiven evolutionären Verfahren generiert, bei dem menschliche Experten während der Evolution wiederholt die Qualität der Melodien bewerten müssen. Um zu vermeiden, das individuelle Wissen der Experten an Stelle der musikalischen Lernbeispiele zu optimieren, wurden interaktive Ansätze hier nicht betrachtet.

Die vorgestellten Verfahren werden anhand von Melodien aus der EsAC-Volkslied-Sammlung [19] experimentell evaluiert.

6.1 Merkmalsselektion

In diesem Abschnitt wird gezeigt, wie man die in dieser Arbeit entwickelte Repräsentation musikalischer Merkmale mit Merkmalsselektionsverfahren kombinieren kann, um Musikstile zu analysieren. Dabei steht die Frage im Vordergrund, ob ein maschinelles Lernverfahren unterschiedliche musikalische Eingabemerkmale benötigt, um die gleiche Lernaufgabe für verschiedene Melodiestile zu lösen.

6.1.1 Modellierung eines einzelnen Stils

Aufgabenstellung. Bei jeder computergestützten Modellierung musikalischer Stile muss man unabhängig von der Komplexität des entwickelten Systems aus einer Vielzahl möglicher Merkmale diejenigen auswählen, die als Eingabe für die Modellierung relevant sind. In diesem Abschnitt untersuchen wir zunächst die Relevanz von Eingabemerkmalen für eine minimale Aufgabenstellung. Gegeben ist ein musikalisches Merkmal, das mit einem lernbasierten Klassifikator vorhergesagt werden soll. Aus einer Menge von Eingabemerkmalen ist eine Teilmenge minimaler Größe auszuwählen, die dem Klassifikator erlaubt, das vorgegebene Merkmal am besten zu prognostizieren. Das Optimierungsverfahren variiert die Eingabemerkmale so, dass ein Gütekriterium – hier die Klassifikationsgüte – maximiert wird. Dabei macht man sich die Tatsache zunutze, dass die Leistungsfähigkeit eines Modells davon abhängt, ob seine Eingabemerkmale für die Lösung einer Lernaufgabe relevant sind.

Eine Konfiguration von Merkmalen wird durch einen binären Aktivierungsvektor repräsentiert, der für jedes potentielle Merkmal angibt, ob es gerade aktiv (1) oder inaktiv (0) ist. Für jede Konfiguration wird ein Klassifikator mit derselben Menge von Trainingsbeispielen trainiert. Als Optimierungskriterium wird die Klassifikationsgüte des Klassifikators auf einer nicht zum Trainieren verwendeten Testmenge herangezogen. Verschiedene Stategien zur Merkmalsselektion wurden in Abschnitt 2.1.5 des Grundlagenkapitels beschrieben.

Die Optimierung der Eingabemerkmale eines Klassifikators löst bereits ein kleines Stilmodellierungsproblem: Sie stellt einen statistischen Zusammenhang zwischen einem einzelnen Merkmal und einer möglichst kleinen Menge anderer Merkmale her, die sich als Stellvertreter des Einzelmerkmals eignen würden. Daraus leitet man die Vermutung ab, dass auch

Corpus	Anzahl der Melodien	Anzahl der Lernmuster
Kinder (Dur)	212	7485
Irland (Dur)	49	3603
Shanxi	358	16686

Tabelle 6.1: Drei Melodiemengen aus der EsAC-Volksliedsammlung.

ein musikalischer Zusammenhang zwischen dem Einzelmerkmal und der Merkmalsmenge besteht. Später bildet die Optimierung der Eingabemerkmale eines Klassifikators die Basis für die Stilunterscheidung und die Stilerkennung und kann darüberhinaus zur Verbesserung der Bewerter bei der evolutionären Suche eingesetzt werden.

Merkmalsselektion für Melodien. Nachdem in einer Diplomarbeit [28] gezeigt werden konnte, dass Merkmalsselektion im von Hand optimierten System HARMONET zu einer Reduktion der benötigten Merkmale führt, wurde die Merkmalsselektion auch auf andere musikalische Modelle untersucht. In diesem Experiment zur Stilmodellierung geht es darum, Eingabemerkmale zu finden, die es einem Klassifikator erlauben, ein vorgegebenes Ausgabemerkmal möglichst gut vorherzusagen. Die optimale Menge von Eingabemerkmalen soll aus einer vorgegebenen Menge von Merkmalen ermittelt werden, mit denen in der musikwissenschaftlichen Literatur Melodien beschrieben werden. Ausgabemerkmal des Klassifikators ist die Tonhöhe; der Klassifikator soll jede Tonhöhe einer Melodie getrennt vorhersagen.

Als Datenbasis für die Merkmalsselektion wurden drei stilistisch sehr unterschiedliche Melodiemengen aus der EsAC-Volksliedsammlung [19] ausgesucht: Kinderlieder, irische Volkslieder und Volkslieder aus der chinesischen Provinz Shanxi (Tabelle 6.1).

Vorgehensweise. Abbildung 6.1 zeigt die Vorgehensweise des Experiments. Aus dem Melodiecorpus, der als Trainingsmenge für einen Melodiestil dienen soll, wird zunächst eine Basisrepräsentation erzeugt, die die Merkmale des EsAC-Formats umfasst (Tonhöhen, Dauern, Phrasen, Taktart, Tonart, evtl. Taktwechsel). Mit Hilfe von Operatorgraphen wird eine angereicherte Darstellung berechnet, die neben den Ansichten der Basismerkmale weitere abgeleitete Ansichten enthält. Nun werden die Melodien mit einem weiteren Operatorgraphen (dem sogenannten Patterngraphen) in Lernmuster zerlegt, wobei weitere einfache Merkmale wie Intervalle oder Konturen erst bei der Erzeugung der Lernmuster berechnet werden. Die

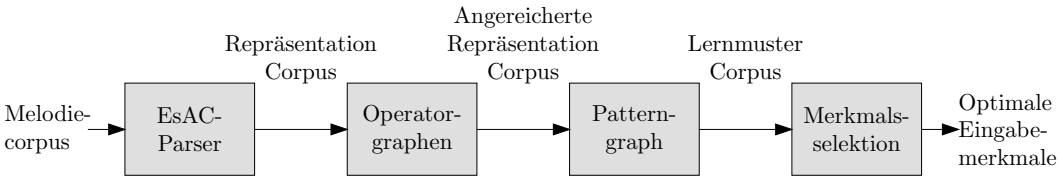


Abbildung 6.1: Transformation musikalischer Information mit Operatorgraphen bei der Optimierung der Eingabemerkmale eines lernbasierten Klassifikators durch Merkmalsselektion.

Lernmuster dienen schließlich als Eingabe für das Merkmalsselektionsverfahren.

Merkmale. Das Zielmerkmal für dieses Experiment ist die Tonhöhe. Die Eingabemerkmale sind in Tabelle 6.2 angegeben. Sie lassen sich in drei Gruppen einteilen: Die erste Gruppe enthält lokale Merkmale, deren zeitliche Position direkt von der Position der vorhergesagten Tonhöhe abhängt (Stufe der Tonhöhe – Harmonisches Gewicht des aktuellen Takts). Die zweite Gruppe von Merkmalen beschreibt die aktuelle Phrase (Ambitus der Phrase – Phrasenrichtung). Die dritte Gruppe besteht aus globalen Merkmalen, die die ganze Melodie charakterisieren (Letzter Ton – Anzahl der Phrasen).

Einige Eingabemerkmale sind durch die Besonderheiten der chinesischen Volkslieder motiviert. So werden zwei Codierungen der Tonhöhen als Kandidaten für stilrelevante Merkmale vorgeschlagen. Die erste Codierung stellt die Tonhöhe diatonisch mit einer 1-aus-7 Codierung dar, aus der sich zusammen mit der Codierung einer Alteration und der Oktavinformation der Melodieton rekonstruieren lässt. Die zweite Codierungen der Tonhöhen codiert die Stufen I, II, III, V, VI des pentatonischen Gerüsts der zugrundeliegenden Skala und fasst alle anderen Stufen als „nicht-pentatonisch“ zusammen, so dass jede Tonhöhe mit einer 1-aus-6 Codierung dargestellt wird. Hintergrund für diese Codierung ist die Unterscheidung zwischen einem pentatonischen Gerüst und Verzierungstönen in der chinesischen Musik. Das Merkmal „letzter Ton der Melodie“ ist dadurch motiviert, dass der Modus einer chinesischen Melodie durch ihren letzten Ton bestimmt wird [22]. Die Merkmale „Existenz einer Synkope“ und „Ambitus der Phrase“ wurden hinzugenommen, weil im Shanxi-Corpus viele Synkopen und ein für europäische Hörgewohnheiten großer Tonumfang der Melodien festgestellt wurde. Die anderen Merkmale sind durch die musikwissenschaftliche Literatur zur Melodieanalyse motiviert (z.B. [20]).

Mustererzeugung. Die Lernmuster für diese Merkmale werden mit dem Patterngraphen generiert, der ausschnittsweise in Abbildung 6.2 gezeigt wird. Die Operatoren des Patterngraphen sind entsprechend ihrer Funktionalität den Schichten des Operatorgraphen zugeordnet. Die ersten beiden Schichten enthalten zeitabhängige Operatoren. Die dritte Schicht wird nur im nicht gezeigten Teil des Patterngraphen verwendet. Die vierte Schicht enthält Wertextraktoren, die fünfte Wissensoperatoren und die sechste Codierungsoperatoren. Die Identitäten realisieren Verbindungen von Operatoren über mehr als eine Schicht hinweg.

Die Melodiemengen in den drei untersuchten Stilen werden zunächst aus dem EsAC-Format in eine Basisdarstellung gebracht, die neben einer Tonart-, Taktart- und Phrasenansicht eine Tonhöhenansicht enthält. Vorbereitend werden aus der Basisdarstellung weitere Ansichten für die Merkmale Bewegtheit (defininiert als mittlere Intervallgröße einer Phrase), Richtung einer Phrase, Existenz einer Synkope, Existenz einer Punktierung, Existenz eines Auftakt, mittlere Phrasendauer, Rhythmusklasse des aktuellen Takts, implizite Harmonische Funktion des aktuellen Takts, Ambitus, Konturklasse der Phrase und Metrisches Gewicht des Tons abgeleitet. Dadurch wird der Gesamtaufwand für die Musterberechnung reduziert, da der Patterngraph für verschiedene Eingaben mehrmals auf das gleiche Element einer Ansicht zugreift (z.B. als linker und rechter Nachbar eines gegebenen Ereignisses).

Bei der Auswertung des Patterngraphen dient die Tonhöhenansicht als Eingabeansicht. Die Koinzidenzoperatoren greifen auf die Basisansichten der Phrase und der Tonart sowie auf die zuvor berechneten abgeleitete Ansichten zu. Jeder Codierungsoperator in der letzten Schicht

des Patterngraphen erzeugt die Codierung für ein Merkmal. In Bezug auf die Merkmalsselektion ist die gewählte Codierung (z.B. Stufencodierung oder pentatonische Codierung einer Tonhöhe) eines symbolischen Merkmals (z.B. Tonhöhe) als Bestandteil eines Merkmals zu verstehen, da für ein Merkmalsselektionsverfahren lediglich die codierte Darstellung eines Merkmals sichtbar ist. Die Merkmalsselektion bewertet also nur die Relevanz eines Merkmals in einer bestimmten Codierung, nicht das symbolische Merkmal selbst.

Klassifikator. Zur Bewertung einer Menge von Eingabemerkmalen wird ein neuronaler Klassifikator verwendet. Dieser besteht aus einem Komitee von drei linearen, neuronalen Netzen, die mit dem Lernverfahren RPROP und dreifacher Kreuzvalidierung 50 Epochen trainiert werden. Die Verwendung linearer anstatt mehrschichtiger Netze ist nötig, um die Laufzeit des Selektionsverfahrens in einem akzeptablen Rahmen zu halten. Für die Selektion von Merkmalen ist die Verwendung mehrschichtiger Netze nicht so wesentlich, weil nur die relative Klassifikationsgüte der Netze beim Vergleich von Merkmalsaktivierungen bedeutsam ist. Dadurch können auf Kosten der Qualität des Klassifikators eine größere Anzahl von Suchpunkten ausgewertet werden. Der Zielkonflikt zwischen dem Aufwand, der in die Bewertung der Suchpunkte investiert wird, und einer ausgiebigen Erkundung des Suchraums ist in der Literatur als *Exploration versus Exploitation*-Dilemma bekannt [45].

Suchstrategie. Als Merkmalsselektionsstrategien werden die in Abschnitt 2.1.5 beschriebenen Verfahren der Vorwärts- und Rückwärtsselektion eingesetzt. Bei der Vorwärtsselektion trainiert man zunächst ein Modell, das keine Eingabemerkmale verwendet und fügt dann iterativ dasjenige Eingabemerkmal ein, das die Klassifikationsgüte am meisten verbessert. Entsprechend werden bei der Rückwärtsselektion aus einem Klassifikator mit allen Eingabemerkmalen iterativ die am wenigsten benötigten Eingabemerkmale entfernt, bis alle Merkmale deaktiviert sind. Als Selektionskriterium dient hier die mittlere Klassifikationsgüte der drei kombinierten neuronalen Netze auf einer Testmenge, die nicht zum Trainieren verwendet wurde. Das Ergebnis ist die Merkmalskombination mit der höchsten Klassifikationsgüte unter allen betrachteten Merkmalskombinationen. Aufschlussreich sind auch die Merkmale einer minimalen Teilmenge von Eingabemerkmalen, deren Bewertung gegenüber der optimalen Bewertung nicht allzusehr abfällt, da diese Merkmalsmenge keine redundanten Merkmale enthält.

Ergebnisse. Für die drei betrachteten Melodiestile wurde je ein Suchlauf mit der Vorwärts- und der Rückwärtsselektion durchgeführt. Tabelle 6.2 zeigt in der linken Spalte die Menge der untersuchten Eingabemerkmale, anhand derer die Stufe der Tonhöhe an einer aktuellen Position t vorhergesagt werden soll.

In den weiteren Spalten sind die Merkmalskombinationen (oder *Zustände*) angegeben, die unter allen betrachteten Merkmalskombinationen die höchste Klassifikationsgüte besitzen. Sie sind bezüglich der ausgewerteten Zustände optimal, nicht aber in Bezug auf den sehr viel größeren gesamten Zustandsraum. Die Bezeichnung „optimal“ in der Tabelle ist mit dieser Einschränkung zu verstehen.

Vergleich von Klassifikatoren. Die Klassifikationsgüten der optimalen Zustände am Fuß der Tabelle zeigen, dass die Optimierung für alle Stile und beide Selektionsverfahren Klassifikatoren findet, die besser sind als ein Klassifikator mit allen Eingabemerkmalen. Durch das Weglassen von Merkmalen gelingt es also, die Generalisierungsleistung der Klassifikatoren

zu steigern. Der Grad der Verbesserung hängt dabei stärker von der Anzahl der Lernmuster als vom Selektionsverfahren ab. Während bei den Kinderliedern und den Shanxi-Melodien die Verbesserung nur etwa 1-3% beträgt, ist sie bei den irischen Melodien mit 8% deutlich größer. Dies ist vermutlich darauf zurückzuführen, dass für den irischen Stil nur eine vergleichsweise geringe Anzahl von Trainingsmustern zur Verfügung stand.

Um die Lernleistung der trainierten Klassifikatoren in Abhängigkeit von der Beschaffenheit der Datenmengen einschätzen zu können, wurde die Klassifikationsgüte eines konstanten Klassifikators berechnet, der immer die häufigste Tonhöhe als Zielwert vorhersagt. Seine Klassifikationsgüte wurde als die a-priori-Wahrscheinlichkeit der häufigsten Tonhöhe auf allen Daten berechnet. Die häufigste Tonhöhe ist bei den Kinderliedern und den chinesischen Volksliedern die fünfte Stufe (29.23% bzw. 23.45%), bei den irischen Volksliedern der Grundton der Skala (25.46%). Wenn man die Klassifikationsgüte eines solchen minimal informierten Klassifikators von der Klassifikationsgüte eines anderen Klassifikators subtrahiert, erhält man einen Eindruck der nicht-trivialen Lernleistung des zweiten Klassifikators. Im Beispiel beträgt diese Differenz für alle Stile und beide Selektionsverfahren etwa 30-32%. Die nicht-triviale Lernleistung der optimierten Klassifikatoren ist also für alle drei Stile und beide Selektionsverfahren vergleichbar.

Die Entwicklung der Klassifikationsgüte während der Merkmalsselektionsläufe ist in Abbildung 6.3 angegeben.

Vorwärtsselektion. Bei der Vorwärtsselektion steigt die Klassifikationsgüte bei allen Stilen zunächst steil an. Für die Kinderlieder stabilisiert sie sich dann bei etwa 59%, findet dann um den 20. Selektionsschritt herum aber noch eine bessere Lösung. Das Optimum für den Kinderliedstil wird in Selektionsschritt 42 angenommen. Das Diagramm zeigt aber, dass es viele niedrigerdimensionale Zustände mit vergleichbar guter Bewertung gibt, beispielsweise den in Selektionsschritt 22 gefundenen Zustand. Die zwischen diesen Schritten eingefügten Merkmale verbessern den Klassifikator nicht wesentlich, verschlechtern ihn aber auch nicht. Die zwischen Schritt 22 und 42 eingefügten Merkmale sind für die Lösung der Lernaufgabe redundant. Für jeden Selektionslauf wurde daher anhand der Diagramm der kleinste Zustand ausgewählt, dessen Bewertung nicht wesentlich gegenüber dem Optimum abfällt, um eine Auskunft über die tatsächlich zur Lösung der Lernaufgabe benötigten Merkmale zu erhalten. Die aktiven Merkmale dieser „minimalen guten Zustände“, die nach Definition der Vorwärtsselektion auch im optimalen Zustand aktiv sind, sind in Tabelle 6.2 kursiv wiedergegeben. Gegen Ende der Vorwärtsselektion fällt die Klassifikationgüte leicht ab – ein Hinweis darauf, dass der Klassifikator in dieser Phase der Suche bereits so viele Eingabemerkmale besitzt, dass die Trainingsdaten nicht ausreichen, um eine gute Generalisierungsleistung zu erzielen.

Der Verlauf der Klassifikationsgüte für die irischen Volkslieder nimmt nach dem anfänglichen Leistungsanstieg bis zum Maximum in Selektionsschritt 11, das fast die Klassifikationsgüte der Kinderlieder erreicht, wieder deutlich ab, je mehr Merkmale eingefügt werden. Der Overfitting-Effekt, der durch das Einfügen von Merkmalen bei konstanter, zu geringer Anzahl von Trainingsdaten entsteht, übersteigt hier den Nutzen, der sich durch das Einfügen musikalisch hilfreicher Merkmale ergeben könnte.

Für den Shanxi-Stil ist die Klassifikationsgüte über den ganzen Verlauf der Vorwärtsselektion niedriger als die Klassifikationsgüte der anderen beiden Stile. Auch bewirkt das Einfügen

neuer Merkmale nicht viel, nachdem in Selektionsschritt 11 etwa das Niveau des Optimums (Schritt 17) erreicht wurde. Hier entsteht der Eindruck, dass die wesentlichen Merkmale bis Schritt 11 eingefügt wurden, ein Overfitting-Effekt durch das Einfügen weiterer Merkmale aber nicht eintritt, da die Shanxi-Trainingsmenge sehr groß ist.

Rückwärtsselektion. Bei der Rückwärtsselektion zeigen die Verläufe der Klassifikationsgüten ähnliche Charakteristiken wie bei der Vorwärtsselektion: Beim Entfernen der ersten Merkmale verbessert sich zunächst die Generalisierungsleistung, wobei diese Phase bei den irischen Melodien sehr viel länger anhält als bei den anderen beiden Stilen. Es folgt eine Reihe von Zuständen auf etwa optimalem Niveau und am Ende ein starker Abfall der Klassifikationsgüte, wenn die wirklich wesentlichen Merkmale aus der Eingabe des Klassifikators entfernt werden. Wie bei der Vorwärtsselektion wurden auch hier hinsichtlich ihrer Dimensionalität minimale Zustände von Hand ausgewählt, deren Klassifikationsgüte mit der des optimalen Zustands vergleichbar ist, um Aufschluss über die unverzichtbaren Merkmale zu erhalten.

Selektierte Merkmale. Wie bereits erläutert, sind die aktiven Merkmale der minimalen guten Zustände für die musikalische Interpretation der Ergebnisse interessanter als die aktiven Merkmale der optimalen Zustände. In den Tabellen 6.3 und 6.4 sind die Merkmale in der Reihenfolge angegeben, in der sie bei der Vorwärtsselektion eingefügt bzw. bei der Rückwärtsselektion entfernt wurden. Bei der Vorwärtsselektion ist ein Merkmal umso wichtiger, je früher es eingefügt wurde. Bei der Rückwärtsselektion ist ein Merkmal umso wichtiger, je später es entfernt wurde. Die fett gedruckten Merkmale gehören zum minimalen guten Zustand für die jeweilige Kombination von Stil und Selektionsverfahren.

Es gibt Merkmale, die bei allen Stilen und beiden Merkmalsselektionsverfahren eine Rolle spielen. Die direkten Nachbarn des Zielwerts (Tonhöhenstufe$[t-1]$, Tonhöhenstufe$[t+1]$), das Intervall zwischen diesen Tönen (Intervall$[t-1, t+1]$) und die implizite Harmonie des aktuellen Takts treten in allen minimalen gut bewerteten Zuständen auf. Hier überrascht, dass die implizite Harmonie eines Takts auch bei den Shanxi-Melodien, die nicht dur-molltonal komponiert sind, nützlich bei der Vorhersage eines Melodietons ist.

Einige Merkmale treten nur im minimalen guten Zustand eines Stils, aber nicht in dem der anderen Stile auf. Bei den Kinderliedern sind dies z.B. die Alteration der Tonhöhe an der Stelle $t-1$, die eingrenzt, wie der gesuchte Ton die Melodie fortsetzen darf, und die Oktavlage der Tonhöhe an der Stelle $t+1$, die einen Hinweis darauf enthält, auf welche Fortsetzung der vorherzusagende Ton hinzielen muss. Ein Merkmal, das nur in den minimalen gut bewerteten Zuständen für den irischen Stil auftritt, ist die Bewegtheit einer Phrase, die die mittlere Intervallgröße in einer Phrase beschreibt. Ein solches abstraktes Merkmal in der Menge der unentbehrlichen Merkmale zu finden, ist überraschend und würde bei einer musikwissenschaftlichen Anwendung dieses Ansatzes Anlass zu einer genaueren Untersuchung der Phrasentypen in irischen Volksliedern geben. Bei den minimalen Shanxi-Merkmalen findet sich keines, das ausschließlich für diesen Stil unentbehrlich ist. Man kann diesen Befund als einen Hinweis darauf sehen, dass die Shanxi-Klassifikatoren vor allem Gemeinsamkeiten mit den anderen Stilen gelernt haben.

Robustheit der Optima. Auch wenn man neuronale Klassifikatoren mit mehrfacher Kreuzvalidierung trainiert, stellt sich die Frage, in welchem Maße die als optimal identifizierten Eingabemerkmale eines Klassifikators von der Zusammenstellung der Trainings-

und Testmengen und von der verfügbaren Anzahl an Trainingsdaten abhängen. Diese Frage wird nun an einem kleinen Beispiel untersucht. Um die Zusammenstellung der Datenmengen unabhängig von den Eigenschaften eines bestimmten suboptimalen Merkmalsselektionsverfahrens beurteilen zu können, wählen wir einen kleinen Merkmalsraum, in dem das globale Optimum für das Selektionsproblem noch mit erschöpfender Suche ermittelt werden kann. Für die bereits bekannten Melodiemengen aus Tabelle 6.1 werden Muster der Form $(TH(t-2), TH(t-1), TH(t+1), TH(t+2); TH(t))$ erzeugt, die die Tonhöhen an der vorletzten, der letzten, der nächsten und der übernächsten Position auf die aktuelle Tonhöhe – das Zielmerkmal – abbilden.

Im folgenden wird die Selektion von Eingabemerkmalen für Trainingsmengen unterschiedlicher Größe durchgeführt. In der ersten Konfiguration werden 70% der verfügbaren Daten zufällig als Trainingsdaten ausgewählt, in drei Kreuzvalidierungsmengen gleicher Größe unterteilt, die zur Selektion der Eingabemerkmale eingesetzt werden sollen. Alle Muster, die aus einer Melodie erzeugt werden, werden dabei gemeinsam einer Menge zugeteilt, da beim Lernen auch ein impliziter Zusammenhang zwischen den Mustern einer Melodie berücksichtigt werden soll. In der zweiten Konfiguration werden etwa 900 Datenpunkte als Trainingsmenge ausgewählt und ebenfalls in drei Kreuzvalidierungsmengen gleicher Größe unterteilt.

Um stabile Ergebnisse zu erhalten, werden pro Konfiguration für jeden Melodiecorpus aus Tabelle 6.1 je 20 Merkmalsselektionsläufe mit erschöpfender Suche durchgeführt, bei denen die Trainingsmenge jeweils zufällig ausgewählt wird. Als Klassifikator wird ein Komitee aus drei linearen neuronalen Netzen verwendet, das 50 Epochen lang mit dem Lernverfahren RPROP trainiert wurde. Die mittlere Klassifikationgüte der drei neuronalen Netze auf einer unabhängigen Testmenge dient der Merkmalsselektion als Gütekriterium.

Als Ergebnis erhält man für jeden Melodiestil 20 global optimale Mengen von Eingabemerkmalen, die durch einen binären Aktivierungsvektor der Länge 4 repräsentiert werden. Tabelle 6.5 zeigt die über 20 Läufe gemittelte Aktivierung jedes Eingabemerkmals für die drei betrachteten Melodiestile in beiden Konfigurationen. Die Aktivierung 0.95 besagt zum Beispiel, dass das zugehörige Merkmal in 19 von 20 berechneten Optima aktiv ist. Ein Wert nahe bei 1 oder 0 drückt aus, dass das zugehörige Merkmal so relevant bzw. irrelevant für die Lösung der Lernaufgabe im vorgegebenen Melodiestil ist, dass seine Zugehörigkeit zu einem Optimum nicht wesentlich von der Zusammenstellung von Trainings- und Testmengen abhängt.

Die Tonhöhen an der letzten und nächsten Position $TH(t-1)$ und $TH(t+1)$ sind in allen Melodiestilen robust gegenüber der Datenauswahl. Tritt ein Merkmal ungefähr in der Hälfte der Läufe in einem Optimum auf wie z.B. die vorletzte Tonhöhe $TH(t-2)$ im Shanxi-Stil, so muss die Bedeutung des Merkmals für die Lernaufgabe mit anderen Mitteln analysiert werden.

Bei der Konfiguration mit umfangreichen Trainingsmengen wird z.B. für die Shanxi-Daten das Merkmal $TH(t-2)$ in 55 Prozent der Läufe selektiert, was darauf hindeutet, dass es sich um ein irrelevantes Merkmal handelt. Tatsächlich beträgt die absolute Differenz der Testfehler für die Zustände 0111 und 1111 gemittelt über alle Läufe nur 0.23 Prozent (nicht in der Tabelle dargestellt). Die Berücksichtigung des Merkmals $TH(t-2)$ führt also für den Shanxi-Stil zu keiner wesentlichen Veränderung der Klassifikationgüte. Im Vergleich

dazu wird im Kinderliedstil, in dem das Merkmal $TH(t-2)$ in jedem Optimum auftritt, der Zustand 1111 im Mittel um 2.5 Prozent besser bewertet als der Zustand 0111. Auch wenn die mittlere Klassifikationsgüte der optimalen Merkmale variiert (zweite Spalte der Tabelle), so spiegelt die Bewertung der Zustände 0111 und 1111 dennoch die Bedeutung der zugehörigen Merkmalsmengen für die Lösung der Lernaufgabe wider.

Die untere Hälfte von Tabelle 6.5 zeigt die mittleren Aktivierungen der globalen Optima für die Konfiguration mit kleinen Trainingsmengen. Da die Trainingsmengen kleiner sind als in der ersten Konfiguration, enthalten die optimalen Zustände erwartungsgemäß weniger Merkmale. Die vorherige und die nächste Tonhöhe $TH(t-1)$ und $TH(t+1)$ sind auch hier in jedem ermittelten Optimum enthalten; diese Merkmale sind offenbar so relevant, dass sie trotz des „Rauschens“ durch die Datenauswahl immer benötigt werden. Die Verkleinerung der Trainingsmengen wird dadurch kompensiert, dass die weniger relevanten Merkmale $TH(t-2)$ und $TH(t+2)$ in den optimalen Zuständen seltener auftreten als in den Ergebnissen für die größeren Datenmengen. Doch ist auch hier die Tendenz zu erkennen, dass beim Kinderliedstil alle Merkmale zur Lösung der Lernaufgabe benötigt werden, während bei den beiden anderen Stilen in erster Linie die Merkmale $TH(t-1)$ und $TH(t+1)$ von Bedeutung sind.

Fazit. Insgesamt legen die Ergebnisse nahe, bei Experimenten, die aufgrund ihrer langen Laufzeit nicht wiederholt werden können, neben der maximal bewerteten Menge von Eingabemerkmalen auch die Merkmalsmenge minimaler Größe zu beachten, deren Bewertung nicht allzusehr gegenüber der maximalen Bewertung abfällt. Bei der Rückwärtsselektion kann man schließen, dass alle Merkmale dieser zweiten Menge für die Lernaufgabe relevant sind, da man kein Merkmal ohne eine wesentliche Verschlechterung der Bewertung weglassen kann. Die Ergebnisse der Merkmalsselektionsexperimente zeigen, dass für die verschiedenen Stile unterschiedliche Merkmale als relevant selektiert werden.

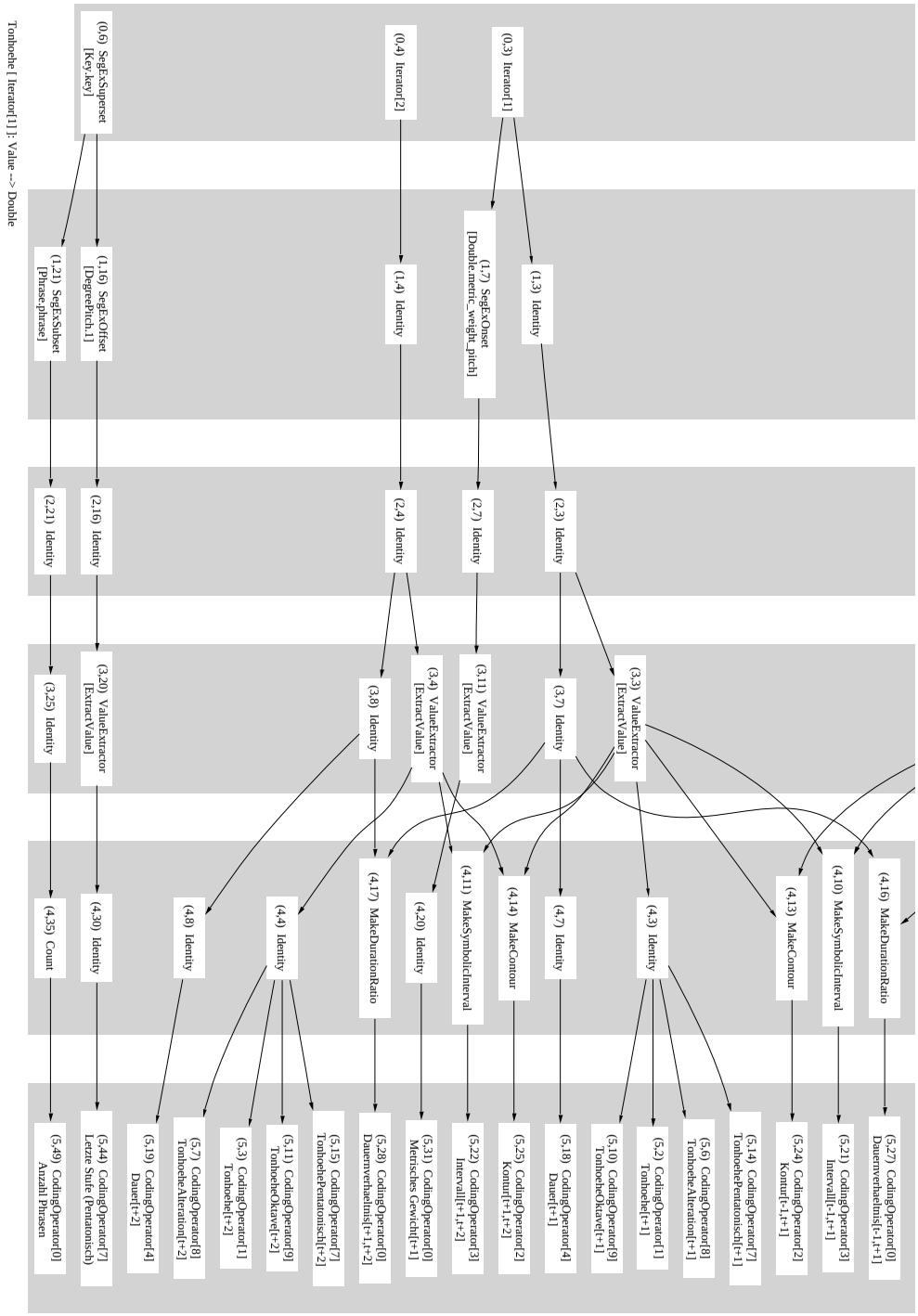


Abbildung 6.2: Ausschnitt aus dem Patterngraphen zur Erzeugung der Lernmuster. Zielmerkmal ist die Tonhöhe[t].

Eingabemerkmal	Vorwärts			Rückwärts		
	Kinder	Irland	Shanxi	Kinder	Irland	Shanxi
Tonhöhenstufe$[t-2]$	1	0	*1*	*1*	*1*	1
Tonhöhenstufe$[t-1]$	*1*	*1*	*1*	*1*	*1*	*1*
Tonhöhenstufe$[t+1]$	*1*	*1*	*1*	*1*	*1*	*1*
Tonhöhenstufe$[t+2]$	*1*	0	*1*	*1*	1	*1*
Alteration der Tonhöhe$[t-2]$	1	0	0	1	0	0
Alteration der Tonhöhe$[t-1]$	*1*	0	0	*1*	0	0
Alteration der Tonhöhe$[t+1]$	0	1	0	1	0	0
Alteration der Tonhöhe$[t+2]$	1	0	0	0	0	0
Oktave der Tonhöhe$[t-2]$	1	0	0	1	0	0
Oktave der Tonhöhe$[t-1]$	*1*	0	0	1	0	0
Oktave der Tonhöhe$[t+1]$	*1*	0	0	*1*	0	1
Oktave der Tonhöhe$[t+2]$	1	0	0	0	0	1
Tonhöhenstufe$[t-2]$ (Pentatonisch)	*1*	*1*	0	0	0	*1*
Tonhöhenstufe$[t-1]$ (Pentatonisch)	1	0	1	0	0	0
Tonhöhenstufe$[t+1]$ (Pentatonisch)	*1*	0	0	1	0	0
Tonhöhenstufe$[t+2]$ (Pentatonisch)	*1*	0	*1*	0	0	0
Dauer$[t-2]$	0	0	1	*1*	0	0
Dauer$[t-1]$	*1*	0	0	1	0	0
Dauer$[t+1]$	*1*	0	0	1	1	0
Dauer$[t+2]$	1	0	0	1	0	1
Intervall$[t-2,t-1]$	*1*	0	0	1	0	0
Intervall$[t-1,t+1]$	*1*	*1*	*1*	*1*	*1*	*1*
Intervall$[t+1,t+2]$	1	0	0	1	0	0
Kontur$[t-2,t-1]$	1	*1*	1	0	0	0
Kontur$[t-1,t+1]$	1	0	0	*1*	1	0
Kontur$[t+1,t+2]$	0	*1*	*1*	1	0	1
Dauernverhältnis$[t-2,t-1]$	*1*	0	0	1	0	0
Dauernverhältnis$[t-1,t+1]$	1	0	1	0	0	0
Dauernverhältnis$[t+1,t+2]$	1	0	0	1	0	0
Metrisches Gewicht$[t-1]$	*1*	0	*1*	1	0	1
Metrisches Gewicht$[t]$	1	1	*1*	1	1	0
Metrisches Gewicht$[t+1]$	1	0	1	*1*	0	1
Rhythmusklasse des Takts	0	0	0	0	0	0
Implizite Harmonie des Takts	*1*	*1*	*1*	*1*	*1*	*1*
Harmonisches Gewicht des Takts	*1*	1	*1*	*1*	1	1
Ambitus der Phrase	1	0	0	*1*	0	0
Erster Phrasenton	1	0	*1*	0	0	1
Letzter Phrasenton	1	0	1	*1*	0	0
Erster Phrasenton (Pentatonisch)	*1*	0	0	*1*	0	0
Letzter Phrasenton (Pentatonisch)	0	0	0	0	0	0
Intervall der Phrasengrenzen	1	0	0	1	0	0
Phrasenform	1	0	0	1	0	0
Bewegtheit	1	*1*	0	1	*1*	0
Phrasenrichtung	*1*	0	0	1	0	0
Letzter Ton (Pentatonisch)	0	1	0	0	0	0
Synkope?	1	0	0	1	0	1
Punktierung?	*1*	0	0	1	0	0
Auftakt?	*1*	0	0	*1*	0	1
Mittlere Phrasendauer	*1*	0	0	1	1	0
Anzahl der Phrasen	0	0	0	0	0	0
Anzahl aktiver Zustände						
optimierter Zustand	43	12	18	38	12	17
minimaler guter Zustand	22	8	12	16	6	6
alle Merkmale	50	50	50	50	50	50
Klassifikationsgüte						
optimierter Zustand	61.52	57.69	53.23	62.05	57.01	53.07
minimaler guter Zustand	61.10	57.36	53.10	61.13	56.60	51.77
alle Merkmale	60.28	49.23	50.78	–	–	–
konstanter Klassifikator	29.23	25.46	23.45	–	–	–

Tabelle 6.2: Optimale Zustände bei der Vorwärts- und Rückwärtsselektion der Eingabemerkmale(1=aktiv, 0=deaktiviert). Die kursiv gesetzten Einsen sind auch in den minimalen gut bewerteten Zuständen aktiv.

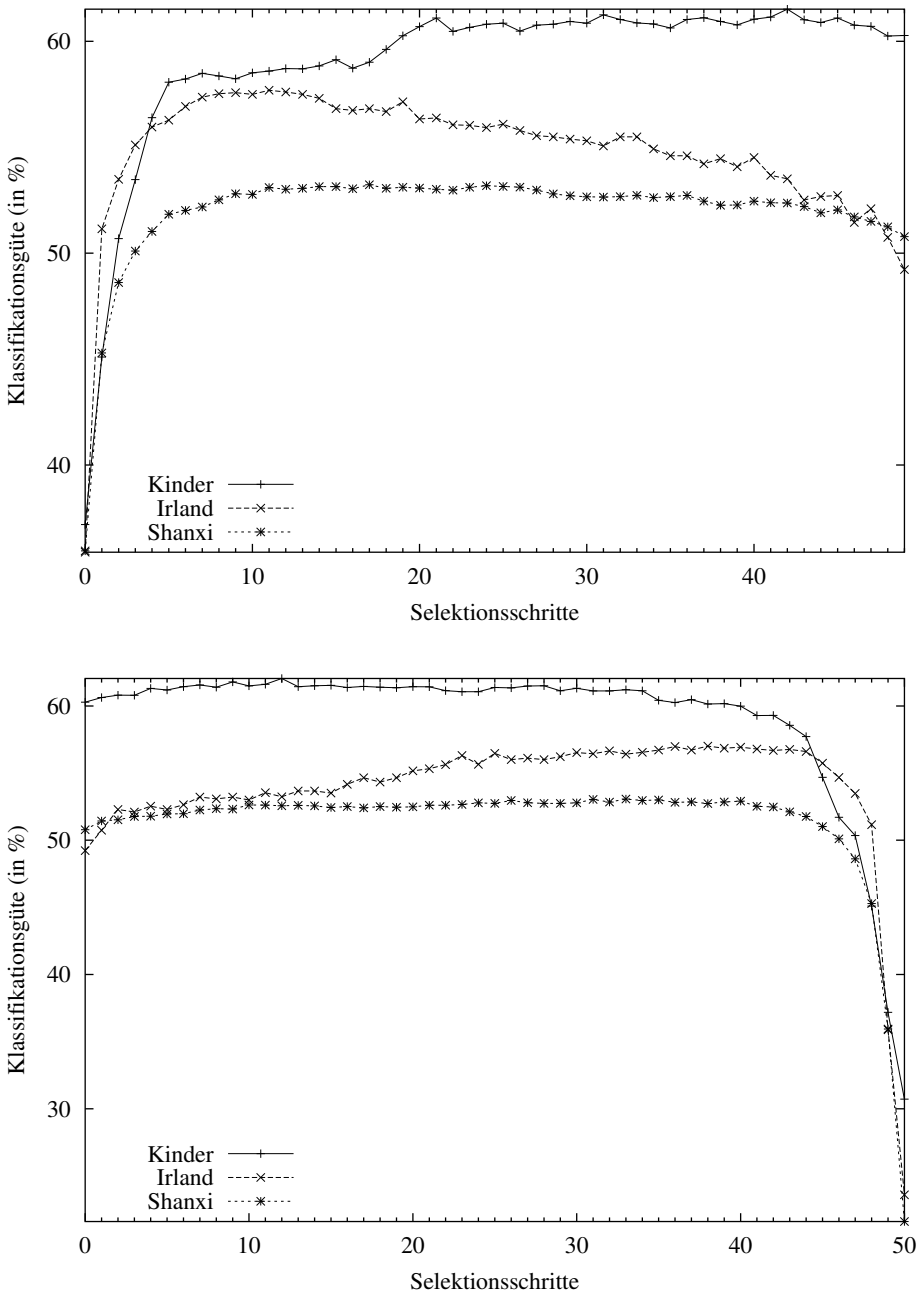


Abbildung 6.3: Verlauf der Klassifikationsgüte bei der Merkmalsselektion. Oben: Vorwärtsselektion. Unten: Rückwärtsselektion

Schritt	Kinder	Irland	Shanxi
0	**Implizite Harmonie/Takt**	**Tonhöhenstufe[t-1]**	**Tonhöhenstufe[t+1]**
1	**Tonhöhenstufe[t-1]**	**Tonhöhenstufe[t+1]**	**Tonhöhenstufe[t-1]**
2	**Tonhöhenstufe[t+2]**	**Implizite Harmonie/Takt**	**Implizite Harmonie/Takt**
3	**Harmonisches Gewicht/Takt**	**Tonhöhenstufe[t-2] (pent.)**	**Tonhöhenstufe[t+2]**
4	**Tonhöhenstufe[t+1]**	**Kontur[t+1,t+2]**	**Intervall[t-1,t+1]**
5	**Erster Phrasenton (pent.)**	**Bewegtheit**	**Tonhöhenstufe[t-2]**
6	**Auftakt?**	**Intervall[t-1,t+1]**	**Metrisches Gewicht[t]**
7	**Alteration der Tonhöhe[t-1]**	**Kontur[t-2,t-1]**	**Harmonisches Gewicht/Takt**
8	**Mittlere Phrasendauer**	Metrisches Gewicht[t]	**Kontur[t+1,t+2]**
9	**Metrisches Gewicht[t-1]**	Letzter Ton (pent.)	**Erster Phrasenton**
10	**Oktave der Tonhöhe[t+1]**	Alteration der Tonhöhe[t+1]	**Tonhöhenstufe[t+2] (pent.)**
11	**Dauernverhältnis[t-2,t-1]**	Harmonisches Gewicht/Takt	**Metrisches Gewicht[t-1]**
12	**Tonhöhenstufe[t+1] (pent.)**	Oktave der Tonhöhe[t-2]	Metrisches Gewicht[t+1]
13	**Oktave der Tonhöhe[t-1]**	Auftakt?	Letzter Phrasenton
14	**Phrasenrichtung**	Alteration der Tonhöhe[t-2]	Dauernverhältnis[t-1,t+1]
15	**Dauer[t-1]**	Dauer[t+1]	Dauer[t-2]
16	**Tonhöhenstufe[t+2] (pent.)**	Oktave der Tonhöhe[t-1]	Tonhöhenstufe[t-1] (pent.)
17	**Tonhöhenstufe[t-2] (pent.)**	Dauer[t+2]	Kontur[t-2,t-1]
18	**Intervall[t-1,t+1]**	Punktierung?	Alteration der Tonhöhe[t+1]
19	**Intervall[t-2,t-1]**	Tonhöhenstufe[t+1] (pent.)	Tonhöhenstufe[t-2] (pent.)
20	**Punktierung?**	Phrasenrichtung	Tonhöhenstufe[t+1] (pent.)
21	**Dauer[t+1]**	Metrisches Gewicht[t+1]	Phrasenrichtung
⋮			

Tabelle 6.3: Sukzessives Einfügen von Merkmalen bei der Vorwärtsselektion. Je früher ein Merkmale eingefügt wird, umso wichtiger ist sein Beitrag für die Lösung der Lernaufgabe.

Schritt	Kinder	Irland	Shanxi
⋮			
34	**Auftakt?**	Dauer[t-1]	Dauer[t+2]
35	**Letzter Phrasenton**	Tonhöhenstufe[t+1] (pent.)	Metrisches Gewicht[t+1]
36	**Alteration der Tonhöhe[t-1]**	Metrisches Gewicht[t+1]	Tonhöhenstufe[t-2]
37	**Metrisches Gewicht[t+1]**	Oktave der Tonhöhe[t-1]	Oktave der Tonhöhe[t+1]
38	**Dauer[t-2]**	Kontur[t-1,t+1]	Oktave der Tonhöhe[t+2]
39	**Oktave der Tonhöhe[t+1]**	Metrisches Gewicht[t]	Metrisches Gewicht[t-1]
40	**Ambitus der Phrase**	Tonhöhenstufe[t+2]	Synkope?
41	**Kontur[t-1,t+1]**	Mittlere Phrasendauer	Erster Phrasenton
42	**Erster Phrasenton (pent.)**	Dauer[t+1]	Kontur[t+1,t+2]
43	**Tonhöhenstufe[t+2]**	Harmonisches Gewicht/Takt	Harmonisches Gewicht/Takt
44	**Harmonisches Gewicht/Takt**	**Intervall[t-1,t+1]**	**Tonhöhenstufe[t-2] (pent.)**
45	**Tonhöhenstufe[t-2]**	**Bewegtheit**	**Intervall[t-1,t+1]**
46	**Intervall[t-1,t+1]**	**Tonhöhenstufe[t-2]**	**Tonhöhenstufe[t+2]**
47	**Tonhöhenstufe[t+1]**	**Implizite Harmonie/Takt**	**Implizite Harmonie/Takt**
48	**Tonhöhenstufe[t-1]**	**Tonhöhenstufe[t+1]**	**Tonhöhenstufe[t-1]**
49	**Implizite Harmonie/Takt**	**Tonhöhenstufe[t-1]**	**Tonhöhenstufe[t+1]**

Tabelle 6.4: Sukzessives Entfernen von Merkmalen bei der Rückwärtsselektion. Je später ein Merkmale entfernt wird, umso wichtiger ist sein Beitrag für die Lösung der Lernaufgabe.

Corpus	KG (σ)	Mittlere Aktivierung der Merkmale				Trainingsmuster
		TH(t-2)	TH(t-1)	TH(t+1)	TH(t+2)	
Kinder (Dur)	48.854 (1.508)	1	1	1	1	1672
Irland (Dur)	54.063 (2.097)	0.95	1	1	0.65	812
Shanxi	45.792 (1.163)	0.55	1	1	1	3786
Kinder(Dur)	43.954 (1.024)	0.95	1	1	0.95	305
Irland(Dur)	49.598 (0.819)	0.4	1	1	0.15	311
Shanxi	41.105 (1.306)	0.2	1	1	0.1	294

Tabelle 6.5: Gemittelte optimale Zustände, mittlere Klassifikationsgüte und Standardabweichung für jeden Stil. Oben: 70% der Muster als Trainingsmenge. Unten: Etwa 900 Muster pro Stil als Trainingsmenge (300 pro Kreuzvalidierungsmenge). In der rechten Spalte ist die mittlere Anzahl der Trainingsmuster in einer Kreuzvalidierungsmenge angegeben.

6.1.2 Stilunterscheidung

Bei der Modellierung eines einzelnen Stils im letzten Abschnitt wurden musikalische Merkmale selektiert, mit denen ein Zielmerkmal für einen einzelnen Melodiestil vorhergesagt werden sollte. Für die drei untersuchten Melodiestile wurden unterschiedliche beste Merkmalskombinationen identifiziert. Nun sollen die Fragen untersucht werden, ob diese Merkmale sich auch zur Unterscheidung der gelernten Musikstile eignen und ob sich die Ähnlichkeit unterschiedlicher Stile mit dem hier entwickelten Instrumentarium ermitteln lässt.

Aufgabenstellung. In diesem Abschnitt wird die *Stilunterscheidung* mit Hilfe neuronaler Netze untersucht. Dabei sucht man mit Hilfe eines Merkmalsselektionsverfahrens Merkmale, die zur Unterscheidung zweier Musikstile besonders geeignet sind. Wie die Modellierung eines Einzelstils und im Gegensatz zur kompositorischen Verwendung der Netze, bietet die Stilunterscheidung den Vorteil eines objektiven Gütekriteriums: Lernmodelle und ihre Eingabemerkmale können anhand der Stilunterscheidungsgüte miteinander verglichen werden.

Stilunterscheidungsgüte. Man erhält die *Stilunterscheidungsgüte*, wenn man zwei Klassifikatoren, die mit Lernmustern unterschiedlicher Stile trainiert wurden, jeweils mit Testmelodien beider Stile testet. Eine Testmelodie gilt als von demjenigen Klassifikator erkannt, der auf den Mustern der Testmelodie die höhere Klassifikationsrate besitzt. Die Stilunterscheidungsgüte ist die Differenz zwischen der Erkennungsrate des einen und der Fehlerrate des anderen Klassifikators. Sie leitet sich vom ROC-Kriterium[1] zur Auswahl bester Entscheidungsmodelle her [21] und bevorzugt Stilunterscheidungsmodelle, die den Fehler beider Klassifikatoren gleichmäßig minimieren.

Die Stilunterscheidungsgüte wird wie folgt berechnet. Gegeben sind zwei Klassifikatoren K_1 und K_2, die mit Lernmustern in unterschiedlichen Musikstilen trainiert wurden, und

[1] ROC steht für „receiver operating characteristics“

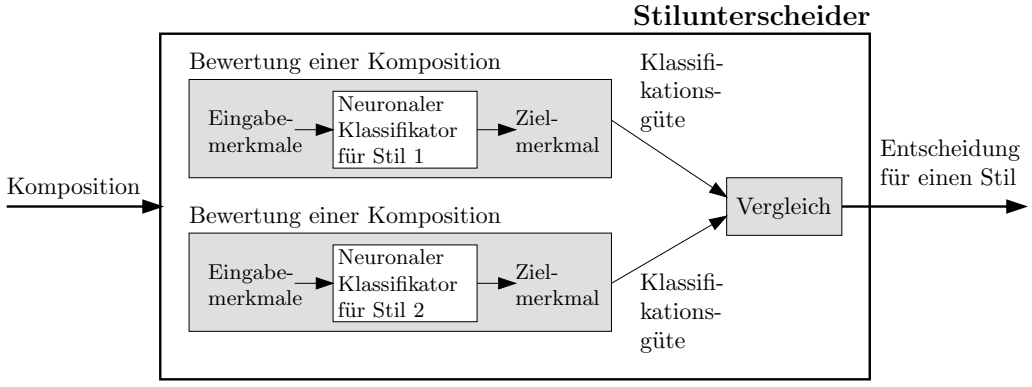


Abbildung 6.4: Stilunterscheidung mit einem zweistufigen Ansatz. Der Stilunterscheider ist ein Klassifikator mit zwei Stilen als Zielklassen. Er enthält neuronale Klassifikatoren, die je nach gelerntem Zielmerkmal eine beliebige Anzahl von Zielklassen besitzen.

musikalische Testbeispiele M_1 und M_2 in beiden Stilen, die nicht zum Trainieren verwendet wurden. Die Klassifikationsgüte $\mathrm{kg}(m, K)$ eines Klassifikators K auf einer Melodie m ist definiert als der Anteil der Melodiemuster, die der Klassifikator richtig klassifiziert. Die Funktion ist_treffer drückt aus, für welchen der beiden gelernten Stile sich die Klassifikatoren K_1 und K_2 entscheiden oder ob sie unentschieden bleiben:

$$\text{ist\_treffer}(m, K_1, K_2) = \begin{cases} 1 & : \text{ falls } \mathrm{kg}(m, K_1) > \mathrm{kg}(m, K_2) \\ 0.5 & : \text{ falls } \mathrm{kg}(m, K_1) = \mathrm{kg}(m, K_2) \\ 0 & : \text{ falls } \mathrm{kg}(m, K_1) < \mathrm{kg}(m, K_2) \end{cases}$$

Die Erkennungs- oder *Trefferrate* $\mathrm{treffer}(M_1)$ und die *Fehlerrate* $\mathrm{fehler}(M_1)$ für die Beispielmenge M_1 im Stil des Klassifikators K_1 sind folgendermaßen definiert:

$$\begin{aligned} \mathrm{treffer}(M_1) &= \frac{1}{|M_1|} \sum_{m \in M_1} \text{ist\_treffer}(m, K_1, K_2) \\ \mathrm{fehler}(M_1) &= \frac{1}{|M_1|} \sum_{m \in M_1} (1 - \text{ist\_treffer}(m, K_1, K_2)) \end{aligned}$$

Entsprechend sind die Treffer- und die Fehlerrate für die Beispielmenge M_2 definiert:

$$\begin{aligned} \mathrm{treffer}(M_2) &= \frac{1}{|M_2|} \sum_{m \in M_2} \text{ist\_treffer}(m, K_2, K_1) \\ \mathrm{fehler}(M_2) &= \frac{1}{|M_2|} \sum_{m \in M_2} (1 - \text{ist\_treffer}(m, K_2, K_1)) \end{aligned}$$

Da die Melodien, die von beiden Klassifikatoren gleich bewertet werden, je zur Hälfte als Treffer und als Fehler verbucht werden, ist die Summe der Treffer- und der Fehlerrate auf einer Melodiemenge immer 1:

$$\begin{aligned} \mathrm{treffer}(M_1) + \mathrm{fehler}(M_1) &= 1 \\ \mathrm{treffer}(M_2) + \mathrm{fehler}(M_2) &= 1 \end{aligned}$$

Um die Stilunterscheidungsgüte eines Klassifikatorpaars (K_1, K_2) zu bewerten, ist es also ausreichend, für beide Beispielmengen nur je eine der Kennzahlen zu berücksichtigen. Die *Stilunterscheidungsgüte* wird definiert als

$$\begin{aligned} \mathrm{SUG} &= \mathrm{treffer}(M_1) - \mathrm{fehler}(M_2) \\ &= \mathrm{treffer}(M_2) - \mathrm{fehler}(M_1) \end{aligned}$$

Je besser das Klassifikatorpaar (K_1, K_2) die Melodien der beiden Stile trennt, desto höher ist die Stilunterscheidungsgüte. Sie nimmt das Minimum -1 für $\mathrm{treffer}(M_1) = 0$ und $\mathrm{fehler}(M_2) = 1$ an, wenn also alle Melodien fehlklassifiziert werden. Das Maximum 1 nimmt die Stilunterscheidungsgüte für $\mathrm{treffer}(M_1) = 1$ und $\mathrm{fehler}(M_2) = 0$ an, d.h. wenn alle Melodien korrekt klassifiziert wurden. Sind die Klassifikatoren K_1 und K_2 gleich, so werden

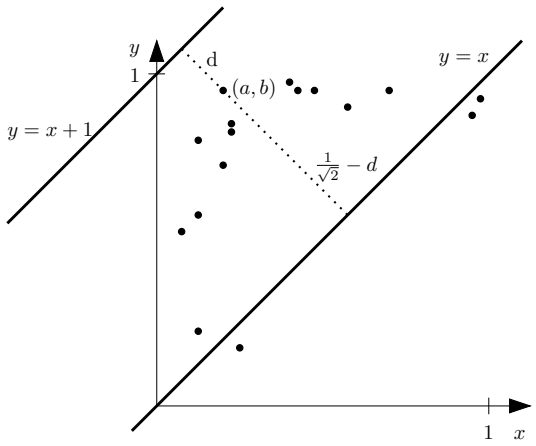


Abbildung 6.5: ROC-Diagramm für die Stilunterscheidung (fiktive Daten). Jeder Punkt (x, y) entspricht einem Klassifikator mit Fehlerrate $x = \text{fehler}(M_2)$ und Trefferrate $y = \text{treffer}(M_1)$ für Testmengen M_1 und M_2 in den beiden betrachteten Stilen.

alle Testmelodien von beiden Klassifikatoren gleich bewertet werden. Dann ist unabhängig davon, ob die Testmengen M_1 und M_2 gleich sind, $\text{treffer}(M_1) = \text{fehler}(M_1) = \text{treffer}(M_2) = \text{fehler}(M_2) = 0.5$. In diesem Fall ist die Stilunterscheidungsgüte 0. Eine positive Stilunterscheidungsgüte bedeutet, dass die Summe der Anteile der korrekt klassifizierten Melodien in beiden Testmengen 0.5 übersteigt. Umgekehrt drückt eine negative Stilunterscheidungsgüte aus, dass die Summe der Anteile der fehlklassifizierten Melodien mehr als 0.5 beträgt.

Die Stilunterscheidungsgüte hat auch eine entscheidungstheoretische Interpretation. Gegeben sei eine Menge von Klassifikatoren, die das Auftreten eines Ereignisses erkennen. Die Klassifikatoren haben also die beiden Zielwerte „Ereignis erkannt“ und „Ereignis zurückgewiesen“. Wenn man unter diesen Klassifikatoren einen auswählen will, der das Verlustrisiko durch eine Fehlentscheidung minimiert, unterscheidet man vier Fälle: Ein eingetretenes Ereignis kann (1) korrekterweise erkannt oder (2) fälschlicherweise zurückgewiesen werden (Fehler 1. Art). Ein nicht-eingetretenes Ereignis kann (3) korrekterweise zurückgewiesen oder (4) fälschlicherweise erkannt werden (Fehler 2. Art). Um den Klassifikator mit dem geringsten Verlustrisiko zu identifizieren, trägt man für jeden getesteten Klassifikator im sogenannten ROC-Diagramm den Anteil der Fehler 2. Art gegen den Anteil der korrekt erkannten Ereignisse auf (Abbildung 6.5). Wenn beide Fehlerarten mit dem gleichen Verlustrisiko behaftet sind, minimiert der Klassifikator, der am nächsten an der Diagonalen $y = x + 1$ liegt, das Verlustrisiko [21].

Auf die Stilunterscheidung übertragen entspricht die Klassifikation eines Ereignisses der Zuordnung einer Melodie aus der Melodiemenge $M_1 \cup M_2$ zum korrekten Melodiestil mit einem Paar (K_1, K_2) von Klassifikatoren, die in beiden Stilen trainiert wurden. Das Testen einer Melodie aus M_1 wird mit dem Eintreten eines Ereignisses, das Testen einer Melodie aus M_2 mit dem Nicht-Eintreten eines Ereignisses identifiziert. Damit minimiert ein Klassifikatorpaar (K_1, K_2) die Fehlzuordnung von Melodien, wenn die zugehörige ROC-Koordinate

Verdoppelter Aktivierungsvektor		**Direkte Nachfolger**		**Kombinierte Nachfolger**	
Stil 1	Stil 2	Stil 1	Stil 2	Stil 1	Stil 2
1 1 0	0 1 1	**0** 1 0	0 1 1	**0** 1 0	0 **0** 1
		1 **0** 0	0 1 1	**0** 1 0	0 1 **0**
		1 1 0	0 **0** 1	1 **0** 0	0 **0** 1
		1 1 0	0 1 **0**	1 **0** 0	0 1 **0**

Tabelle 6.6: Nachfolgerkandidaten des Aktivierungsvektors 110001 bei der doppelten Merkmalsselektion. Bei den Nachfolgerkandidaten sind die im aktuellen Schritt deaktivierten Zustände fett gedruckt.

$(\text{fehler}(M_2), \text{treffer}(M_1))$ unter allen ROC-Koordinaten getesteter Klassifikatoren minimalen Abstand zur Geraden $y = x + 1$ hat. Der minimale Abstand $d = \frac{1}{\sqrt{2}} |1 + a - b|$ zwischen einem Punkt (a, b) und der Geraden $y = x + 1$ wird von einem Klassifikatorpaar (K_1, K_2) mit maximaler Stilunterscheidungsgüte $1 + \text{fehler}(M_2) - \text{treffer}(M_1)$ angenommen. Ein Klassifikatorpaar mit maximaler Stilunterscheidungsgüte minimiert also die Fehlzuordnung von Melodien gleichmäßig für beide betrachteten Stile.

Vorüberlegungen. In einem vorbereitenden Experiment wurde die Hypothese untersucht, dass die bei der Stilmodellierung optimierten Klassifikatoren auch zu einer verbesserten Stilunterscheidungsgüte im Vergleich zu den Klassifikatoren mit allen Eingabemerkmalen führen könnten. Wenn man durch die Optimierung der Eingabemerkmale einen Klassifikator an die Besonderheiten eines bestimmten Melodiestils anpasst, könnte man erwarten, dass der Zuwachs der Klassifikationsgüte beim Vergleich eines Klassifikators mit allen Eingabemerkmalen und eines optimierten Klassifikators für den optimierten Stil deutlicher ist als für Testbeispiele anderer Stile. Es zeigte sich, dass diese Erwartung für die hier betrachteten Daten nicht erfüllt wird: Durch die Optimierung der Eingabemerkmale erhöhte sich die Klassifikationsgüte auch für Melodien nicht-trainierter Stile. Dies deutet darauf hin, dass nicht nur die Unterschiede, sondern auch die Gemeinsamkeiten zwischen den Stilen durch die selektierten Eingabemerkmale besser modelliert werden.

Doppelte Merkmalsselektion. Um die Eingabemerkmale zweier Klassifikatoren gezielt zu optimieren, wurde ein angepasstes Merkmalsselektionsverfahren entwickelt, das Klassifikatoren in zwei Stilen trainiert und anhand der Stilunterscheidungsgüte optimiert. Es handelt sich dabei um eine Abwandlung der sequentiellen Suche (Abschnitt 2.1.5).

Die Anzahl der Merkmale wird verdoppelt, um die Merkmale für beide Stile getrennt darzustellen. Um eine Aussage darüber zu erhalten, ob für beide Stile dieselben Merkmale unterscheidungsrelevant sind, wird die Merkmalsselektion auf dem verdoppelten Aktivierungsvektor ausgeführt, der die Eingabemerkmale für die Klassifikatoren beider Stile repräsentiert.

Gegeben sind Trainings- und Testmuster, die aus Melodien in zwei Musikstilen generiert wurden. Für $n \in \mathbb{N}$ potentielle Merkmale werden Aktivierungsvektoren $\{0, 1\}^{2n}$ gebildet, bei denen die ersten n Einträge die Aktivierung der Eingabemerkmale eines Klassifikators für den ersten Stil und die Einträge $n + 1 \ldots, 2n$ die die Aktivierung der Eingabemerkmale

eines Klassifikators für den zweiten Stil bezeichnen.

Optimierungskriterium. Als Optimierungskriterium dient die Stilunterscheidungsgüte. Ein Aktivierungsvektor wird bewertet, indem Klassifikatoren in den beiden untersuchten Melodiestilen mit den aktiven Eingabemerkmalen trainiert werden und ihre Stilunterscheidungsgüte auf einer nicht zum Trainieren benutzten Testmenge berechnet wird.

Suchstrategie. Die Suche beginnt mit dem Einszustand, für den alle Merkmale beider Netze aktiv sind. Für diesen werden Klassifikatoren in den beiden vorgegebenen Melodiestilen trainiert, deren Stilunterscheidungsgüte als Bewertung des Einszustands dient.

Die Kandidaten für Nachfolger eines beliebigen Aktivierungsvektors werden berechnet, indem man wie bei der Standardrückwärtssuche alle Aktivierungsvektoren bildet, bei denen ein aktiver Zustand deaktiviert wurde. Für alle Vektorhälften der Kandidaten werden nun Klassifikatoren in den zugehörigen Stilen trainiert. Da sich die Klassifikatoren beider Stile beliebig kombinieren lassen, können mit diesen Klassifikatoren nicht nur die Stilunterscheidungsgüten für die Kandidaten berechnet werden, sondern auch die Stilunterscheidungsgüten für alle Kombinationen von Vektorhälften.

Tabelle 6.6 veranschaulicht die Bildung von Nachfolgekandidaten für den Zustand 110011. Seine direkten Nachfolger, die sich durch Deaktivierung eines Merkmals ergeben, sind in der mittleren Spalte angegeben. In der rechten Spalte befinden sich die Kandidaten, die zusätzlich durch Kombination der rechten und linken Hälfte der Kandidaten der mittleren Spalte entstehen (außer dem aktuellen Zustand 110011).

Aus allen so berechneten Aktivierungsvektoren (im Beispiel: mittlere und rechte Spalte) wird ein Nachfolger mit der höchsten Stilunterscheidungsgüte ausgewählt. Als Ergebnis werden aus allen berechneten Aktivierungsvektoren diejenigen mit der höchsten Stilunterscheidungsgüte ausgewählt.

Bei der doppelten Rückwärtsselektion werden die Merkmale nicht abwechselnd aus der Eingabe beider Klassifikatoren entfernt, sondern an der vielversprechendsten Position des Aktivierungsvektors. Es können also auch mehrere Merkmale hintereinander aus *einem* Klassifikator entfernt werden. Die Eingabemerkmale des anderen Klassifikators bleiben dann über mehrere Suchschritte hinweg gleich.

Da man durch die zusätzlichen Kombinationen Zustände bewertet, die bei der Standard-Rückwärtssuche erst später auftreten würden, werden die bisher bewerteten Aktivierungsvektoren zwischengespeichert, um Mehrfachbewertungen von Aktivierungsvektoren zu vermeiden. Die Suche wird abgebrochen, wenn für einen der Klassifikatoren keine Eingabemerkmale mehr vorhanden sind.

Experiment. Wie bei der Einzelstilmodellierung werden auch hier drei Melodiecorpora aus der EsAC-Volksliedsammlung – Kinderlieder, irische Volkslieder und Lieder aus Shanxi — ausgewählt (Tabelle 6.1). Um einen Vergleich mit der Einzelstilmodellierung zu ermöglichen, werden dieselben Merkmale wie zuvor als potentiell relevante Merkmale gewählt (Abbildung 6.2) und auch alle Klassifikatoren wie bisher mit dreifacher Kreuzvalidierung und dem Lernverfahren RPROP trainiert.

Für jedes Paar von Melodiestilen werden mit der doppelten Rückwärtsselektion Paare optimierter Klassifikatoren mit maximaler Stilunterscheidungsgüte gesucht.

Ergebnisse. Tabelle 6.7 zeigt, dass sich durch die gemeinsame Optimierung der Eingabemerkmale für beide Stile die Stilunterscheidungsgüte gegenüber der Stilunterscheidung mit einzeln optimierten Netzen verbessert. Zum Vergleich sind die niedrigeren Stilunterscheidungsgüten für die Stilmodellierungklassifikatoren und die nicht optimierten Klassifikatoren angegeben, die alle Eingabemerkmale verwenden. Für die hier untersuchten Daten findet die doppelte Merkmalsselektion also ein besseres Modell zur Stilunterscheidung als die zweifache Anwendung der einfachen Merkmalsselektion aus Abschnitt 6.1.1. Die Stilunterscheidungsgüte verbessert sich durch die doppelte Merkmalsselektion, obwohl sich die Klassifikationsgüten gegenüber den einzeln optimierten Klassifikatoren deutlich verschlechtert haben (Tabelle 6.8). Dies lässt sich so interpretieren, dass die Klassifikatoren sich lediglich auf die Unterschiede zwischen den Stilen konzentriert haben ohne die Gemeinsamkeiten zu lernen.

Abbildung 6.6 zeigt den Verlauf der Stilunterscheidungsgüten für die sechs Selektionsläufe (3 Stile mit Vorwärts- bzw. Rückwärtsselektion). Im oberen Bild (Vorwärtsselektion) werden mit jedem Selektionsschritt ein oder zwei Merkmale eingefügt. Die früheste maximale Stilunterscheidungsgüte zeigt den gesuchten optimalen Zustand an. Im unteren Bild (Rückwärtsselektion) ist der späteste Zustand mit maximaler Bewertung der gesuchte. Wie bei der Einzelstilselektion bietet es sich beim Vergleich der Kinderlieder mit den irischen Volksliedern an, neben den optimalen Zuständen, die sehr viele aktive Merkmale enthalten, auch gut bewertete Zustände mit einer geringeren Anzahl von Merkmalen zu betrachten, um Aufschluss über die besonders relevanten Merkmale zu erhalten. Bei den anderen Stilkombinationen war die Anzahl der Merkmale für die optimalen Zuständen bereits hinreichend klein.

Die Merkmalsselektion für die Kinder- und Shanxi-Lieder durch Vorwärtsselektion erreicht bereits nach einem Schritt einen Zustand, in dem sich alle Testmelodien perfekt klassifizieren lassen. Nach einigen weiteren Schritten führt das Hinzufügen neuer Merkmale sogar wieder zu einer deutlichen Verschlechterung der Stiltrennung.

Bei den nach einem Selektionsschritt ausgewählten Merkmalen handelt es sich um die Alteration des vorletzten Tons für den Kinderliedstil und die Anzahl der Phrasen für den Shanxi-Stil. Dass diese Merkmale zwar zur Stilunterscheidung, aber nicht zur Prognose der Tonhöhen der Einzelstile geeignet sind, erkennt man an den geringen Klassifikationsgüten der zugehörigen Klassifikatoren (30.72% bzw. 21.61%) in Tabelle 6.8, die sich nicht wesentlich von den Klassifikationsgüten der konstanten Klassifikatoren in Tabelle 6.2 unterscheiden. Bei der Rückwärtsselektion wird für die gleiche Stilkombinationen ein optimaler Zustand mit einer deutlich größeren Anzahl von Merkmalen gewählt. Wie bei der Vorwärtsselektion ist die Stilunterscheidungsgüte hier 1. Die Klassifikationsgüten der zugehörigen Klassifikatoren sind jedoch deutlich höher (55.77% bzw. 51.34%).

Aufgrund der Ergebnisse ist die Rückwärtsselektion gegenüber der Vorwärtsselektion zu bevorzugen, weil die optimalen Klassifikatoren bei der Rückwärtsselektion bei vergleichbarer Stilunterscheidungsgüte mehr musikalische Zusammenhänge gelernt haben. Diese Tendenz ist auch bei den anderen Stilkombinationen erkennbar.

Das Ziel der Merkmalsselektion, eine gemeinsame Verteilung aller Merkmale durch ein robusteres Modell mit weniger Merkmalen zu approximieren, wird also besser durch die Rückwärtsselektion erreicht, bei der die Differenz zwischen idealer und bewerteter Verteilung

Merkmal	Vorwärts						Rückwärts					
	Kinder/ Irland		Kinder/ Shanxi		Irland/ Shanxi		Kinder/ Irland		Kinder/ Shanxi		Irland/ Shanxi	
Tonhöhenstufe$[t-2]$	0	1	0	0	*1*	0	0	0	0	*1*	0	0
Tonhöhenstufe$[t-1]$	0	0	0	0	*1*	0	0	0	*1*	*1*	*1*	0
Tonhöhenstufe$[t+1]$	0	0	0	0	0	*1*	1	*1*	0	0	*1*	*1*
Tonhöhenstufe$[t+2]$	1	0	0	0	*1*	*1*	0	0	0	0	*1*	0
Alteration der Tonhöhe$[t-2]$	*1*	*1*	*1*	0	0	*1*	0	*1*	0	0	0	0
⋮												
Letzter Ton (pent.)	*1*	1	0	0	0	0	*1*	0	0	0	*1*	0
Synkope?	*1*	*1*	0	0	0	0	*1*	0	*1*	0	0	0
Punktierung?	1	0	0	0	0	0	0	0	0	0	0	0
Auftakt?	1	1	0	0	0	0	1	0	0	*1*	0	0
Mittlere Phrasendauer	1	1	0	0	0	*1*	1	0	0	0	0	0
Anzahl der Phrasen	1	1	0	*1*	*1*	0	1	0	0	0	0	0
Stilunterscheidungsgüte												
Stilunterscheidung	0.984		1.000		1.000		0.984		1.000		0.991	
Einzelstilmodellierung	0.802		0.892		0.919		0.752		0.942		0.885	
alle Merkmale aktiv	0.898		0.917		0.816		–		–		–	

Tabelle 6.7: Optimale und minimale gute Zustände bei der Stilunterscheidung (Ausschnitt). Der Index $t+k$ bezeichnet die Position des $t+k$-ten Melodietons, $k \in \{0,1,2\}$. Die Stilunterscheidungsgüten wurden für die Stilunterscheidungs- und Stilmodellierungsklassifikatoren sowie für die nicht-optimierten Klassifikatoren berechnet.

durch Weglassen möglichst irrelevanter Merkmale minimiert werden soll. Bei der Vorwärtsselektion hingegen maximiert man durch das Hinzufügen möglichst relevanter Merkmale den Abstand zum vorhergehenden Modell, wodurch zwar das Optimierungskriterium verbessert wird, jedoch nicht sichergestellt ist, dass man sich damit in Richtung der idealen Verteilung bewegt.

Alternativen. Man könnte zur Stilunterscheidung auch andere Selektionsverfahren als die Vorwärts- und die Rückwärtsselektion einsetzen. Der verdoppelte Merkmalsraum macht sich bei Verfahren mit höherer Zeitkomplexität jedoch so stark bemerkbar, dass auf ihre Untersuchung an dieser Stelle verzichtet wurde.

Eine andere Variante der Stilunterscheidung besteht darin, die zu unterscheidenden Stile direkt zu lernen. In diesem Fall ist die Ausgabe des Netzes kein zu prognostizierendes Merkmal – also z.B. die Tonhöhe zum Zeitpunkt t – sondern der Stil des Melodiestils der zugehörigen Melodie selbst. Der Vorteil dieser Variante besteht darin, dass für die Merkmalsselektion keine Verdopplung der Merkmale erforderlich ist. Ein Nachteil ist, dass mit einem solchen Modell eine nachträgliche Analyse stiltypischer Übergänge nicht möglich ist. Lernt ein Modell musikalische Zielwerte, so kann man es auch zur Bewertung der Stiltreue einer Komposition hinsichtlich des gelernten Merkmals einsetzen.

		vs. Kinder	vs. Irland	vs. Shanxi	einzeln optimiert
Vorwärts	Kinder	–	48.72	30.72	61.52
	Irland	31.95	–	40.77	57.69
	Shanxi	21.61	37.87	–	53.23
Rückwärts	Kinder	–	56.17	55.77	62.05
	Irland	48.22	–	49.96	57.01
	Shanxi	51.34	48.04	–	53.07

Tabelle 6.8: Klassifikationsgüten (in %) für die optimalen Stilunterscheidungsmerkmale aus Tabelle 6.7. Die Zeilen entsprechen dem zum Trainieren verwendeten Stil, die Spalten geben den Vergleichsstil bei der Optimierung an. Die Klassifikationsgüten der Stilunterscheidungsklassifikatoren sind niedriger als diejenigen für die besten Stilmodellierungsklassifikatoren in der rechten Spalte.

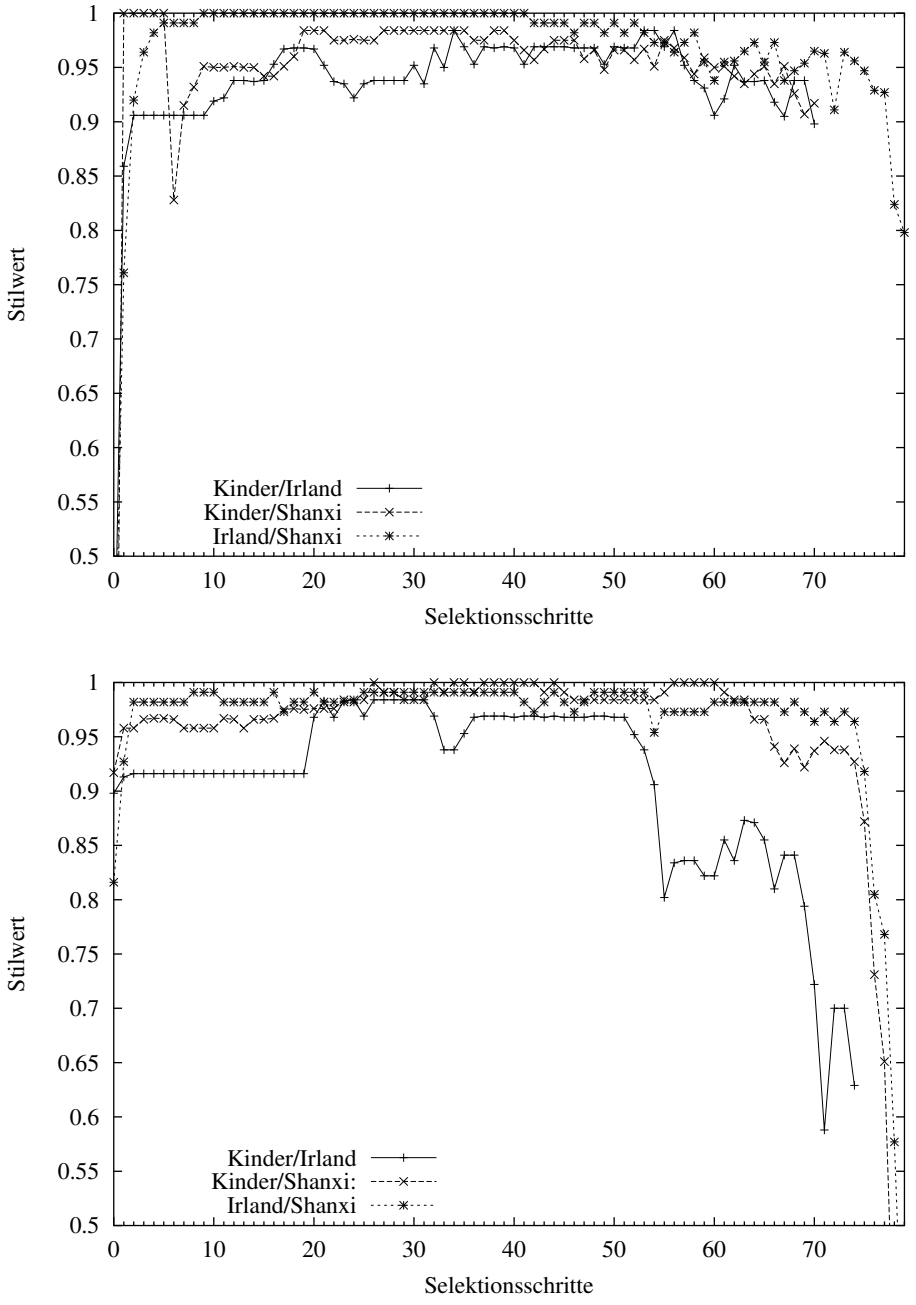


Abbildung 6.6: Verlauf der Stilwerte bei der Optimierung der Stilunterscheidung. Oben: Vorwärtsselektion. Unten: Rückwärtsselektion.

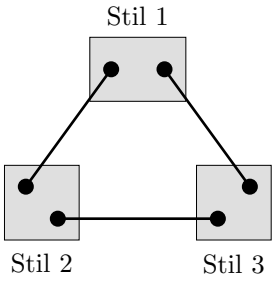


Abbildung 6.7: Der Stilerkenner für $n = 3$ Stile setzt sich aus $\frac{1}{2}n(n-1) = 3$ Paaren von neuronalen Klassifikatoren zusammen. Jeder Punkt entspricht einem Klassifikator, jede Kante steht für einen Stilunterscheider wie in Abbildung 6.4.

6.1.3 Stilerkennung

Bei der Stilunterscheidung im letzten Abschnitt ging es darum, zwei Stile mit speziell dafür optimierten neuronalen Klassifikatoren zu unterscheiden. Das zuvor vorgestellte Verfahren lieferte einen Klassifikator mit zwei Zielklassen, den beiden untersuchten Stilen. Die Aufgabe der Stilunterscheidung wird in diesem Abschnitt zur *Stilerkennung* erweitert. Gesucht ist ein n-Klassen-Klassifikator, der in der Lage ist, eine Melodie, die zu einem von vorgegebenen n Melodiestilen gehört, dem richtigen Stil zuzuordnen. In Verbindung mit der Merkmalsselektion handelt es sich hierbei um eine Verallgemeinerung bestehender Ansätze zur Stilunterscheidung.

Eine Möglichkeit wäre, die im letzten Abschnitt diskutierte doppelte Merkmalsselektion direkt auf die Optimierung von n Klassifikatoren zu übertragen, indem der verdoppelte Aktivierungsvektor durch einen n-fachen Aktivierungsvektor ersetzt wird, oder einen Stil von den anderen abzugrenzen, indem man die Lernmuster der anderen Stile zu einer gemeinsamen Menge zusammenfasst. Da die Größe des Suchraums exponentiell mit der Anzahl der Dimensionen steigt, wird es mit zunehmender Anzahl der betrachteten Stile unwahrscheinlicher, dass die Rückwärtssuche ein aussagekräftiges (lokales) Optimum für den n-fachen Aktivierungsvektor findet. Zur Konstruktion eines Stilerkenners wird daher ein anderer Ansatz gewählt, bei dem die Größe des Suchraums bei der Merkmalsselektion von der Anzahl der betrachteten Musikstile unabhängig ist.

Der Stilerkenner wird aus Stilunterscheidern wie in Abbildung 6.4 zusammengesetzt. Um zu erkennen, ob eine Melodie zu einem von mehreren vorgegebenen Stilen gehört, werden für alle Paare von Stilen Stilunterscheidungsklassifikatoren optimiert, die den charakteristischen Unterschied zwischen zwei Stile lernen und sie so voneinander abgrenzen. Bei n Stilen ergeben sich so $\binom{n}{2} = \frac{1}{2}n(n-1)$ Klassifikatoren, da für je zwei Stile ein Paar von Klassifikatoren erzeugt wird. Abbildung 6.7 zeigt die Struktur eines Stilerkenners für drei Stile.

Stilerkennungsrate. Um beurteilen zu können, wie gut das so gebildete Team von „Stilexperten“ von Klassifikatoren neue Melodien in einem der gelernten Stile dem richtigen Stil zuordnen kann, wird die *Stilerkennungsrate* definiert.

	Kinder	Irland	Shanxi	Mittlere Stilerkennungsrate
Stilunterscheidung	98.44	97.37	98.67	98.16
Einzelstilmodellierung	82.03	84.21	92.92	86.39
Alle Merkmale aktiv	92.19	76.32	95.13	87.88

Tabelle 6.9: Stilerkennungsraten (in %) der Klassifikatoren für die Erkennung von Melodien des eigenen Stils sowie die über alle Stile gemittelte Stilerkennungsrate. Als Merkmalsselektionsverfahren wurde die Rückwärtsselektion verwendet.

Die Zuordnung einer Melodie m zu einem der n Stile geschieht wie folgt. Für eine Melodie m und ein Paar von Klassifikatoren K_i und K_j drückt die Funktion $\text{ist\_treffer}(m, K_i, K_j)$ aus, ob sich die Klassifikatoren entweder für einen der gelernten Stile entscheiden (0, 1) oder unentschieden bleiben (0.5) (vgl. Abschnitt 6.1.2).

Diese Trefferwerte werden für alle Klassifikatoren eines Stils und ihre Entsprechungen in den anderen Stilen zu einem Vektor von Treffersummen summiert:

$$\text{treffersumme}(m) = \left(\sum_{\substack{1 \le i \le n \\ i \neq s}} \text{ist\_treffer}(m, K_s, K_i) \right)_{s=1}^{n}$$

Die Entscheidung fällt dann für diejenigen Stile, für die die Summe ihrer Trefferwerte maximal ist (Trefferstile):

$$\text{ist\_trefferstil}(m) = \left(\begin{cases} 1 & : \text{ falls } \text{treffersumme}^{(s)}(m) = \max\left(\text{treffersumme}(m)\right) \\ 0 & : \text{ sonst} \end{cases} \right)_{s=1}^{n}$$

wobei der hochgestellte Index (s) den s-ten Eintrag des Treffersummenvektors bezeichnet. Für eine Testmenge M mit Melodien im Stil s wird die Stilerkennungsrate SER für das Stilexperten-Team folgendermaßen berechnet:

$$\text{SER}(M, s) = \frac{1}{|M|} \sum_{m \in M} \left[\frac{\text{ist\_trefferstil}^{(s)}(m)}{\sum_i \text{ist\_trefferstil}^{(i)}(m)} \right]$$

Wenn eine Melodie eindeutig dem Stil s zugeordnet wurde, gibt es nur einen Treffer und die Melodie trägt 1 zur Stilerkennungsrate bei. War die Entscheidung zwischen mehreren Stilen nicht eindeutig, so trägt die Melodie nur einen Anteil zur Stilerkennungsrate bei, der umgekehrt proportional zur Anzahl der Trefferstile ist. Damit geht auch die Einigkeit des Stilexperten-Teams in die Stilerkennungsrate ein.

Die mittlere Stilerkennungsrate zur Gesamtbewertung der Klassifikatoren ergibt sich als Mittelwert der Stilerkennungsraten für Testmengen in allen Einzelstilen.

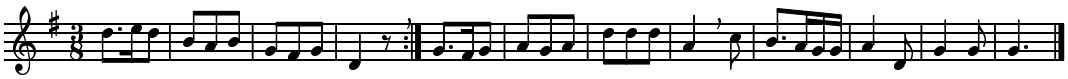


Abbildung 6.8: Fälschlicherweise als irisches Volkslied erkanntes Kinderlied „Eia popeia was rasselt im Stroh“

Experiment. Die Stilerkennung wird mit denselben Melodiemengen aus der EsAC-Volkslied-Sammlung getestet, die bereits in den vorangegangenen Experimenten verwendet wurden (Tabelle 6.1).

Ergebnisse und Interpretation. In Tabelle 6.9 sind die Stilerkennungsraten für die verschiedenen Stile für Testmengen angegeben, die nicht zum Trainieren der Klassifikatoren verwendet wurden: Bei Klassifikatoren, die zur Stilunterscheidung optimiert wurden, werden zwischen 97% und 99% der Testmelodien ihrem Stil korrekt zugeordnet, ein bemerkenswertes Ergebnis, wenn man bedenkt, dass die Klassifikationsraten der verwendeten neuronalen Klassifikatoren alle unter 57% liegen (Tabelle 6.8). Für die einzeln optimierten Klassifikatoren und die mit allen Merkmalen trainierten Klassifikatoren fallen die Erkennungsraten geringer aus.

Abbildung 6.8 zeigt ein Kinderlied, das mit großer Sicherheit als irisches Volkslied klassifiziert wurde. Die harmonieorientierte Melodik, die den tänzerischen 3/8-Takt unterstreichende Motivik und die für ein Kinderlied großen Intervalle erinnern in der Tat eher an irische Folklore als an ein Wiegenlied, um das es sich laut Annotation der EsAC-Sammlung eigentlich handelt. Die Analyse der fehlklassifizierten Melodien kann also Aufschluss auf Merkmale des vermeintlich erkannten Stils geben.

Das vorgeschlagene Stilerkennungsverfahren eignet sich daher nicht nur, um Melodien aufgrund ihrer stilistischen Ähnlichkeit mit einem Suchmuster aus einer Melodiesammlung (z.B. einer Datenbank oder dem Internet) zu extrahieren, sondern auch als computergestütztes Werkzeug, um stilistisch interessante Ausreißer in einem Melodiecorpus aufzuspüren.

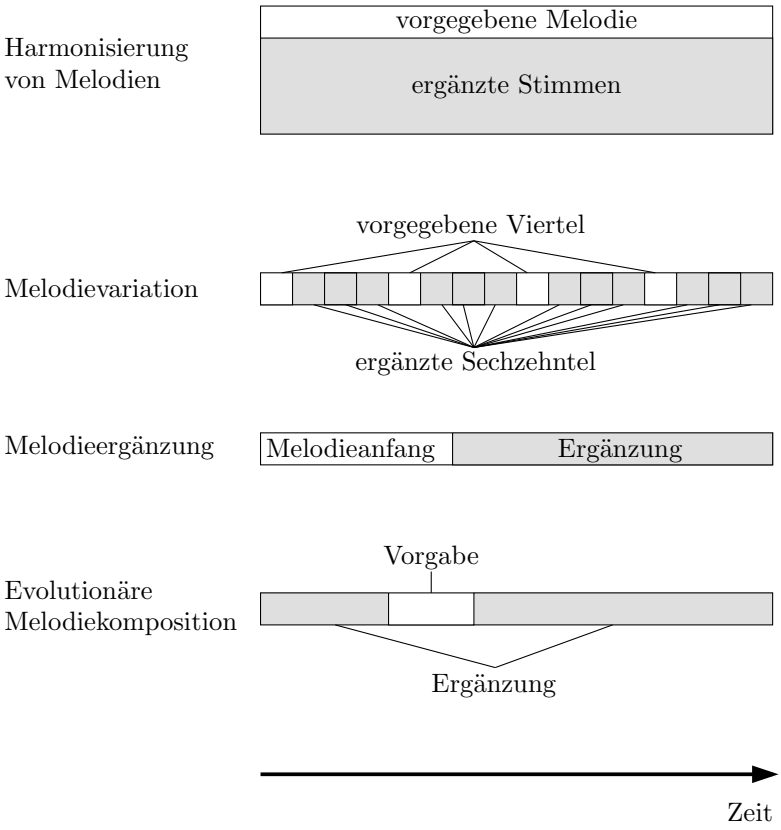


Abbildung 6.9: Vier musikalische Lernaufgaben mit abnehmender zeitlicher Vorgabe

6.2 Evolutionäre Komposition

Evaluierung durch Melodiegenerierung. Wenn man ein einzelnes Merkmal vorhersagen, Musikstile unterscheiden oder erkennen will, ist die Qualität eines Modells objektiv durch die Klassifikationsleistung, die Stilunterscheidungs- und die Stilerkennungsgüte messbar. Diese Gütekriterien sagen jedoch nur indirekt etwas darüber aus, ob ein Modell die wesentlichen Eigenheiten eines Musikstils erfasst hat. Dies zeigt sich erst, wenn man mit Hilfe des Modells überzeugende neue Melodien im untersuchten Stil generieren kann. Da es für die Stiltreue der generierten Melodien kein objektives Gütekriterium gibt, ist man darauf angewiesen, menschliche Experten beurteilen zu lassen, ob die erzeugten Melodien in ihrer Gesamtgestalt adäquate Repräsentanten des modellierten Stils sind.

Die Generierung von Melodien ist in dreifacher Hinsicht schwieriger als bisher betrachtete Musikmodellierungsaufgaben. Zum einen ist es im allgemeinen einfacher, die Grenze zwischen Klassen von Objekten zu bestimmen – wie z.B. bei der Stilunterscheidung – als eine Klasse selbst zu beschreiben, diese also gegenüber Objekten abzugrenzen, die nicht zu dieser

Erzeuge eine zufällige Population.
Solange Abbruchbedingung nicht erfüllt ist:
• Erzeuge Nachkommen durch Mutation.
• Erzeuge Nachkommen durch Rekombination.
• Aktualisiere Nachkommen.
• Bewerte Nachkommen.
• Selektiere neue Population aus alter Population und Nachkommen.

Tabelle 6.10: Generierung von Melodien mit evolutionärer Suche

Klasse gehören. Zweitens haben wir bisher nur Klassifikatoren mit dem gleichen Zielmerkmal kombiniert; bei der Melodiegenerierung müssen nun Klassifikatoren kombiniert werden, die unterschiedliche Merkmale vorhersagen. Dabei treten Wechselwirkungen zwischen den Ausgabemerkmalen auf, die bei der Generierung berücksichtigt werden müssen und eine Evaluierung der Rolle einzelner Komponenten erschweren.

Die dritte Schwierigkeit ergibt sich daraus, dass im Vergleich zu früher untersuchten Fragestellungen die Vorgabe für das erzeugte Musikstück bei der Melodiegenerierung schrumpft. Abbildung 6.9 zeigt schematisch vier musikalische Aufgabenstellungen mit abnehmender zeitlicher Vorgabe. Bei der Harmonisierung einer Melodie mit HARMONET und der Melodievariation mit MELONET ist eine Melodie als Vorlage gegeben, die sich über den gesamten zeitlichen Verlauf des generierten Musikstücks erstreckt und den globalen Verlauf prägt. Dies erklärt, warum beide Systeme imstande sind, ihre Aufgabe allein mit lokaler Kontextinformation zu lösen. Bei der Melodieergänzung mit dem System MeloGeNet ist nur noch ein Melodieanfang vorgegeben; bei der Melodiegenerierung mit dem im Rahmen dieser Arbeit entwickelten System MeloLab kann eine musikalische Vorgabe an beliebiger Stelle gegeben oder ganz weggelassen werden.

Verallgemeinerung von MeloGeNet**.** Bereits das System MeloGeNet setzte evolutionäre Suchverfahren zur Komposition von Melodien ein [89]. Dort waren die Repräsentation, die Veränderungen der Melodien und die Bewertung aber fest vorgegeben. Der Ansatz aus MeloGeNet wird nun mit Hilfe von Operatorgraphen verallgemeinert. Neu im Vergleich zu MeloGeNet ist die Repräsentation der Melodien mit Hilfe von Ansichten, die mit ihrer variableren Zeitrepräsentation die Voraussetzung schafft, um komplexeren musikalischen Stilen als Kinderliedern gerecht zu werden. Die Veränderung und Bewertung von Melodien wird mit Operatorgraphen realisiert, so dass die musikalischen Merkmale als Variable der evolutionären Melodiegenerierung nach Bedarf zusammengestellt werden können.

6.2.1 Vorgehensweise

In diesem Abschnitt wird beschrieben, wie das allgemeine Schema der evolutionären Suche zur Generierung von Melodien konkretisiert und an die Erfordernisse der Aufgabenstellung angepasst wird. Durch das Austauschen der Operatorgraphen können die verwendeten mu-

Operatorgraph	**Eingabe- und Ausgabe**	**Durchlaufsteuerung**
Randomisierer	Ein- und Ausgabeansicht identisch, mutierbar	(Standarditerator $\mathbf{it}_{+1}$)
Mutierer	Ein- und Ausgabeansicht identisch, mutierbar	Zufallsiterator
Rekombinierer	Schnittansicht als Vorlage	(probabilistische Strategie)
Bewerter	beliebige Eingabeansicht, numerische Ausgabe	(Standarditerator $\mathbf{it}_{+1}$)
Aktualisierer	beliebige Eingabeansicht, abgeleitete Ausgabeansicht	beliebiger Iterator
Population	Größe der Population	
	Anzahl der Nachkommen durch Mutation	
	Anzahl der Nachkommen durch Rekombination	
Bewertung	Gewichtung der Bewerter	
Abbruch-	Maximale Anzahl von Epochen	
bedingung	oder Maximale Fitness	

Tabelle 6.11: Variable Komponenten und Parameter bei der evolutionären Melodiesuche

sikalischen Merkmale variiert werden. Tabelle 6.11 gibt einen Überblick über die variablen Parameter der evolutionären Melodiesuche mit MeloLab.

Suchraum. Als Suchpunkte bei der evolutionären Melodiegenerierung dienen Melodien. Anfangs wird eine Population von Zufallsmelodien erzeugt, aus denen durch Mutation und Rekombination Nachkommen erzeugt werden. Eine mutierte Melodie liegt in der Nähe ihres Elternelements. Durch Rekombination gewonnene Melodien unterscheiden sich stärker von ihren Elternelementen, sind also im Suchraum „weiter“ von diesen entfernt. Nachdem die Population und die Nachkommen bewertet wurden, wird aufgrund eines Selektionskriteriums eine Population für die nächste Generation zusammengestellt und mit der Erzeugung von Nachkommen fortgefahren, bis ein Abbruchkriterium erfüllt ist (Tabelle 6.10).

Da die evolutionäre Suche mit Zufallsmelodien beginnt, die erst nach und nach eine musikalisch sinnvolle Gestalt annehmen, wurde der Begriff einer Melodie in Abschnitt 2.2.1 sehr allgemein gefasst. Auch eine zufällige Folge von Tönen gilt hier als Melodie, selbst wenn sie uns nicht musikalisch sinnvoll erscheint.

Repräsentation. Bei der evolutionären Suche werden Melodien wie bisher mit Hilfe von Ansichten dargestellt, die mit Hilfe von Operatorgraphen erzeugt oder verändert werden. Zwischen verschiedenen Ansichten einer Melodie bestehen im allgemeinen vielfältige Wechselbeziehungen: Intervalle stellen den Abstand zwischen zwei Tönen dar, Motive werden ebenfalls aus Tönen berechnet, ein einzelner Ton gehört zu genau einer Phrase, Tongruppen induzieren eine implizite Harmonik. Um die bei der Melodiegenerierung relevanten Abhängigkeiten beschreiben zu können, werden drei Attribute für Ansichten eingeführt, die die Rolle der Ansichten während der evolutionären Suche beschreiben. Eine *mutierbare* Ansicht ist eine Ansicht, deren Inhalt durch Mutation verändert werden darf. Eine *abgeleitete* Ansicht ist eine Ansicht, deren Inhalt als Folge einer Veränderung der Melodie aktualisiert werden muss. Eine Ansicht kann gleichzeitig mutierbar und abgeleitet sein. Eine *konstante* Ansicht ist

bereits in der Mustermelodie vorgegeben und wird während der evolutionären Suche nicht verändert. Sie kann eine gewünschte Eigenschaft einer Melodie wie z.B. eine bestimmte Harmonisierung beschreiben, auf die bei der Bewertung der Melodien Bezug genommen wird. Eine konstante Ansicht ist weder mutierbar noch abgeleitet.

Aktualisierung. Wenn man nun bei der evolutionären Suche beliebige Ansichten zufällig verändert, bringt man die Repräsentation der Melodie leicht in einen inkonsistenten Zustand. Beispielsweise ändert sich eine Intervallansicht, wenn die Tonhöhenansicht verändert wird, aus der die Intervallansicht berechnet wurde. Daher wird bei der evolutionären Generierung von Melodien zusätzlich zu den Schritten der allgemeinen evolutionären Suche (Abschnitt 2.1.5) die *Aktualisierung* einer Melodie als neue Operation eingeführt. Sie stellt sicher, dass durch zufällige Änderungen entstandene Inkonsistenzen in der Repräsentation einer Melodie vor ihrer Bewertung eliminiert werden. Ein *Aktualisierer* ist ein Operatorgraph, der eine abgeleitete Ansicht berechnet. Um einen Aktualisierer bei der Melodiegenerierung einzusetzen, muss man zusätzlich zum Operatorgraphen einen Steueriterator sowie eine Ein- und eine Ausgabeansicht angeben, deren Typen mit dem Ein- bzw. Ausgabetyp des Operatorgraphen verträglich sind. Falls die Operatoren des Aktualisierers Kontexttypen besitzen, müssen diese geeigneten Ansichten zugeordnet werden. Aktualisierung bedeutet, dass jede Melodie der Population durch eine ähnliche, in sich konsistente Melodie ersetzt wird.

Die improvisatorische Lokaloptimierung im neuronalen Melodieergänzungssystem MeloGeNet kann als ein Spezialfall der Aktualisierung aufgefasst werden. Bei der Lokaloptimierung wird nach der Veränderung einer Melodie durch Mutation und Rekombination jeder Ton der generierten Melodie mit Hilfe eines neuronalen Netzes aus seinem lokalen Kontext neu berechnet. Dadurch fließt das gelernte Wissen des Netzes direkt in die Generierung der Melodien ein und die Bewertung der Melodien verbessert sich. Die schnelle Verbesserung der Bewertung ist aber gleichzeitig auch ein Nachteil der Lokaloptimierung, denn wenn Änderungen einer Melodie sofort durch ein neuronales Netz geglättet werden, wird der Weg der evolutionären Suche durch weniger gut bewertete Bereiche des Melodiesuchraums in besser bewertete Regionen möglicherweise von vorneherein unterbunden. In der Tat gerät die Melodiegenerierung beim System MeloGeNet schnell in lokale Optima.

Die anderen Schritte der evolutionären Melodiegenerierung orientieren sich am allgemeinen Ablauf der evolutionären Suche.

Initialisierung. Die Anfangspopulation für die evolutionäre Suche wird erzeugt, indem eine Mustermelodie zufällig verändert wird. Ein Element der Anfangspopulation wird erzeugt, indem eine oder mehrere Ansichten der Mustermelodie durchlaufen und jedes Segment mit einem Operatorgraph zufällig verändert wird. Die Operatorgraphen, die zum Initialisieren einer Melodie dienen, werden im folgenden als *Randomisierer* bezeichnet. Sie operieren auf mutierbaren Ansichten. Da die Ansichten voneinander abhängen können, werden nach der Initialisierung die weiter unten beschriebenen Aktualisierer auf die Elemente der Anfangspopulation angewandt.

Nicht alle Ansichten einer Melodie werden durch die Initialisierung verändert. Wenn man z.B. die Harmonisierung der Mustermelodie für die generierten Melodien vorgeben will, wählt man die Zielansichten für Randomisierer und die Aktualisierer so, dass die Harmonieansicht nicht verändert wird. Bei der Bewertung einer Melodie wird dann die tatsächliche Har-

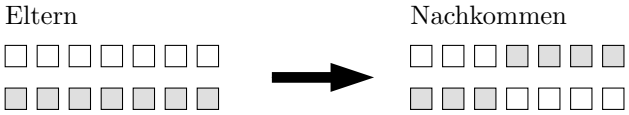


Abbildung 6.10: Rekombination mit gleichmäßig aufgebauten Elementen

monisierung der generierten Melodie berechnet und mit der gewünschten Harmonisierung verglichen.

Maske. Wenn man musikalisches Material wie z.B. einen Melodieanfang vorgeben will, das bei der Melodiegenerierung wiederaufgegriffen werden soll, markiert man einen Teil der Melodie als unveränderlich. Zu diesem Zweck definiert man eine sogenannte Maskenansicht, deren Segmente Bereiche der Melodie kennzeichnen, die durch Mutation und Rekombination nicht verändert werden dürfen. Bei der Initialisierung einer Melodie und den Veränderungen im Verlauf der evolutionären Suche werden Segmente, die sich mit der Maskenansicht überschneiden, nicht verändert.

Mutation. Bei der Mutation wird ein Element der Population zufällig ausgewählt und mit Mutierern verändert. Ein *Mutierer* ist ein Operatorgraph, der einzelne Werte einer Ansicht verändert. Er durchläuft die Ansicht mit einem *Zufallsiterator*, der seine Schrittweite in einem vorgegebenen Intervall zufällig wählt. Dadurch unterscheiden sich bei Nachkommen, die aus dem gleichen Element durch Mutation erzeugt wurden, nicht nur die mutierten Werte, sondern auch die Positionen der Mutationen. Die Mutation ist eine lokale Verschiebung eines Suchpunkts in einer Population, die über den Suchraum verstreut ist.

Rekombination. Durch die Rekombination (engl. *cross-over*) zweier Elemente der Population erzeugt man Nachkommen, die Eigenschaften beider vorgegebenen Elemente kombinieren. Rekombinierte Nachkommen werden gebildet, indem man aus der aktuellen Population zufällig zwei Elemente auswählt und diese an einem zufällig gewählten Schnittpunkt teilt. Der Anfang der ersten und das Ende des zweiten Elements werden zum ersten Nachkommen zusammengesetzt. Entsprechend besteht der zweite Nachkomme aus dem Anfang des zweiten und dem Ende des ersten Elements.

Wenn alle Elemente einer Population aus der gleichen Anzahl von Bausteinen gleicher Größe bestehen, ist die Rekombination eine einfache Operation (Abb. 6.10). Bei der hier verwendeten Repräsentation von Melodien mit Ansichten können jedoch sowohl Segmente einer Ansicht als auch Segmente unterschiedlicher Ansichten einander überlappen. Daher wählen wir einen zweistufigen Ansatz zur Rekombination von Melodien. Es wird eine *Schnittansicht* vorgegeben, deren Segmentgrenzen Kandidaten für musikalisch sinnvolle Schnitte einer Melodie angeben. Man könnte etwa die Phrasen oder Motive einer Melodie verwenden, um vielversprechende Schnittpunkte zu erhalten. Anhand der Schnittansichten beider Elternelemente wird zufällig ein globaler Schnittpunkt ausgewählt, der in beiden Schnittansichten keine Segmente zerschneidet. Für alle weiteren Paare von Ansichten sucht man dann einen lokalen Schnittpunkt, der möglichst nah am globalen Schnittpunkt liegt und in den betrachteten Ansichten keine Segmente zerteilt. Abbildung 6.11 skizziert diese Herangehensweise.

Konstante Ansichten, die nur Referenzinformation wie z.B. eine gewünschte Harmonisierung

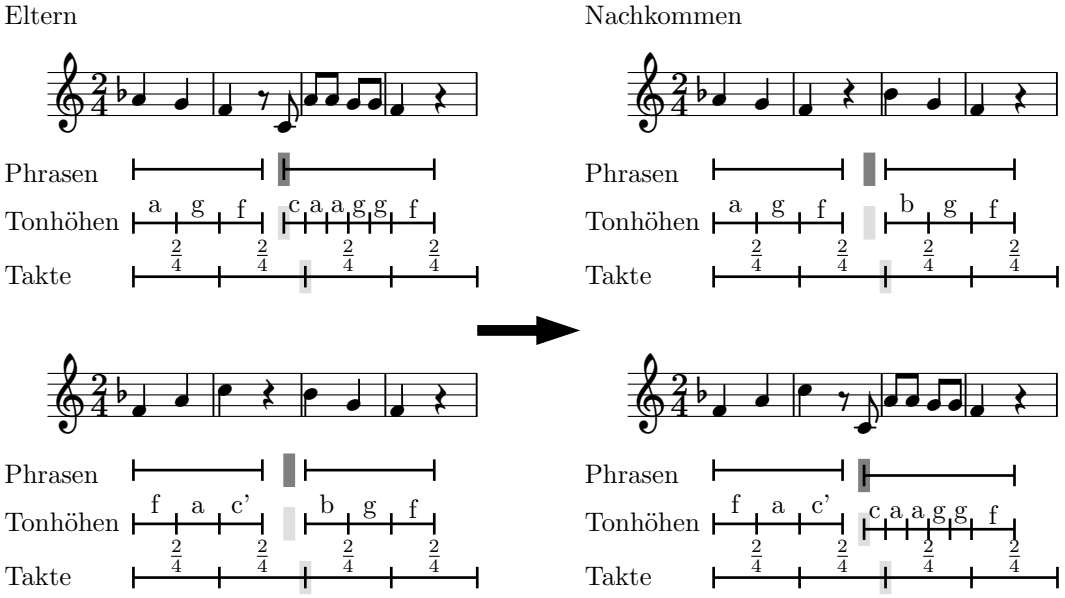


Abbildung 6.11: Rekombination von Melodien. Die Phrasenansicht dient hier als Schnittvorlage und enthält den dunkelgrau markierten globalen Schnittpunkt. Die daraus abgeleiteten lokalen Schnittpunkte in der Tonhöhen- und der Taktansicht sind hellgrau gekennzeichnet.

enthalten, brauchen bei der Rekombination nicht berücksichtigt zu werden. Auch abgeleitete Ansichten werden nicht rekombiniert, sondern nach der Kombination der anderen Ansichten neu berechnet. Auf diese Weise werden insbesondere musikalische Strukturen wie Motivklassen oder Phrasenformen, die sich auf Segmente auf beiden Seiten der Schnittpunkte beziehen können, im Nachkommen konsistent mit den kombinierten Ansichten gemacht.

Während die Mutation einer Melodie eine lokale Verschiebung eines Suchpunkts darstellt, kann man die Rekombination zweier Suchpunkte als gezielte Erzeugung eines Suchpunkts „zwischen“ den Elternsuchpunkten begreifen, der im allgemeinen besser bewertet ist als ein zufällig erzeugter neuer Suchpunkt.

Bewertung. Auch zur Bewertung von Melodien werden Operatorgraphen, sogenannte *Bewerter* eingesetzt. Die Bewerter enthalten das Wissen über den modellierten Stil in Form von neuronalen Netzen oder Regeln. Beispielsweise kann ein Bewerter die Angemessenheit einzelner musikalischer Elemente bezogen auf übergeordnete Strukturen bewerten, etwa ob sich ein einzelner Ton in den melodischen Bogen einer Phrase einfügt.

Bewerter steuern die Suchrichtung der evolutionären Melodiesuche, indem sie Melodien umso höher bewerten, je mehr sie ihren stilistischen Erwartungen entsprechen. Um Bewertungen für mehrere Eingaben mitteln zu können, müssen Bewerter eine numerische Ausgabe liefern. Die Bewertung $f(M, B)$ einer Melodie M mit einem Bewerter B ist definiert als mittlere

Ausgabe des Bewerters über alle Segmente der Eingabeansicht A des Bewerters:

$$f(M, B) = \frac{1}{|A|} \sum_{s \in A} B(s).$$

Daraus ergibt sich für eine Menge von Bewertern $B_1, \ldots, B_k$ und Gewichten $\alpha_1, \ldots, \alpha_k$ mit $\alpha_i \geq 0$ und $\sum_{j=1}^{k} \alpha_j = 1$ die Bewertung $f(M)$ einer Melodie als

$$f(M) = \sum_{j=1}^{k} \alpha_j f(M, B_j)$$

sowie die mittlere Bewertung $\bar{f}$ und die maximale Bewertung $f_{\max}$ einer Population von Melodien $M_1, \ldots, M_n$ als

$$\bar{f} = \frac{1}{n} \sum_{i=1}^{n} f(M_i)$$

bzw.

$$f_{\max} = \max_{1 \leq i \leq n} f(M_i).$$

Die Hyperparameter $\alpha_1, \ldots, \alpha_k$ drücken die relative Bedeutung der Bewertung im Ensemble aller Bewerter aus. In den Experimenten der vorliegenden Arbeit werden die Hyperparameter experimentell ermittelt. Zur algorithmischen Ermittlung der Hyperparameter könnte man auch das in [36] vorgeschlagene entropiebasierte Maß zur Berechnung der Hyperparameter für die bewertenden Netze verwenden.

Selektion. Bei der Selektion werden aus der aktuellen Population und den durch Mutation und Rekombination gewonnenen Nachkommen die Elemente mit der höchsten Bewertung ausgewählt. Sie bilden die Population für die nächste Generation.

Ein Problem bei dieser Selektionsstrategie ergibt sich aus der Tatsache, dass mehrfach auftretende Elemente mit hoher Bewertung bevorzugt ausgewählt werden, so dass sich im Lauf der Evolution ein einziges gut bewertetes Element durchsetzt, alle Elemente in der Population also gleich sind. Bei den nachfolgenden Suchschritten geht man damit nur noch von einem Punkt des Suchraums aus. Die Rekombination wird wirkungslos, da das Kreuzen gleicher Elemente diese Elemente selbst erzeugt, die wegen ihrer hohen Bewertung zudem bevorzugt in die nachfolgende Generation übernommen werden. Die Vorzüge eines parallelen Suchverfahrens verschwinden also, wenn man zulässt, dass die Population im Laufe der Suche kollabiert.

Diversifizierung. In der Literatur werden verschiedene Strategien angegeben, um die Diversität einer Population zu erhöhen. Anstatt die besten Elemente der letzten Population und der Nachkommen auszuwählen, kann man die Elemente der neuen Population mit einer Wahrscheinlichkeit selektieren, die proportional zur Bewertung der Elemente ist. Dadurch werden auch schlechter bewertete Elemente in die nächste Population übernommen, die sich im nächsten Suchschritt dann verbessern können. Andere Selektionstrategien wählen ausschließlich Elemente aus den mutierten und rekombinierten Nachkommen aus oder teilen die

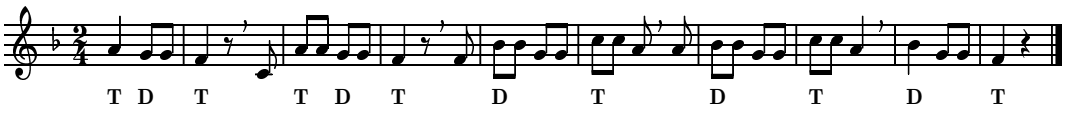
Kandidaten aufgrund ihrer Ähnlichkeit in Gruppen ein, aus denen jeweils Elemente für die nächste Generation ausgewählt werden [70].

In den hier durchgeführten Experimenten hat es sich für den Sucherfolg als entscheidend erwiesen, eine harte Strategie zur Reduktion von mehrfach auftretenden Melodien zu wählen. In den unten beschriebenen Experimenten wurden bei der Selektion zunächst mehrfach auftretende Elemente getrennt aus Elternpopulation und Nachkommenschaft entfernt. Dann wurden die am höchsten bewerteten Elemente aus beiden Mengen ausgewählt. Eine Melodie kann in einer so selektierten Population also höchstens zweimal vorkommen, wenn sie eine hohe Bewertung hat und sowohl in der vorhergehenden Elternpopulation als auch unter den Nachkommen aufgetreten ist. Bei den darauffolgenden Mutations- und Rekombinationsoperationen erhöht sich dadurch die Wahrscheinlichkeit, dass Nachkommen einer hoch bewerteten Melodie erzeugt werden, während gleichzeitig die Diversität der Population sichergestellt wird. Ist die Anzahl der so gewonnenen Elternelemente und Nachkommen kleiner als die Größe der Population, so wird die neue Population mit zufällig erzeugen Elementen aufgefüllt.

Andere, weichere Strategien wie etwa die Elimination aufeinanderfolgender gleicher Melodien haben in einer Voruntersuchung nicht zur gewünschten Diversität der Population geführt und wurden verworfen.

Abbruchbedingung. Die Suche wird abgebrochen, wenn eine vorgegebene Anzahl von Epochen durchlaufen wurde oder das beste Element der Population einen vorgegebenen Fitnesswert erreicht. Da die absoluten Fitnesswerte für eine gewichtete Summe unterschiedlich skalierter Bewerter wenig aussagekräftig sind, wird im folgenden der maximal erreichbare Fitnesswert von 100 als Abbruchkriterium gewählt. Wird er nicht erreicht, so bricht die Suche nach einer festen Anzahl von Epochen ab.

Evaluierung der evolutionären Suche. Nach der evolutionären Suche müssen die generierten Melodien musikalisch evaluiert werden. Die Bewerter, die bei der Suche das Optimierungskriterium bilden, sind ein technisches Gütemaß, das nur für die jeweils aktuelle Konfiguration der Suche vergleichbare Werte liefert. Das eigentliche Ziel der Melodiegenerierung besteht aber darin, von der stilistischen Stimmigkeit der generierten Melodien ausgehend eine überzeugende Konfiguration der evolutionären Suche zu finden, die wiederum Aufschluss über die Relevanz der untersuchten Merkmale gibt.




```
KINDER
CUT[SCHLAF KINDLEIN SCHLAF]
REG[Europa, Mitteleuropa, Deutschland, Hessen ?]
KEY[K0001  08  F 2/4]
MEL[  3_22  1_0
    -5  3322  1_0
    1  4422  553
    3  4422  553_
    4_22  1_0_ //] >>
FCT[Kinder -, Wiegen - Lied]
```

Abbildung 6.12: Beispiel 1: Mustermelodie „Schlaf, Kindlein, schlaf“ (K0001) in Notendarstellung und im EsAC-Format

6.2.2 Ein einfaches Beispiel

Zunächst wird ein einfaches Beispiel für eine evolutionäre Melodiesuche gegeben, das den Aufbau eines Generierungsexperiments beschreibt und die Rolle der im vorangegangenen Abschnitt vorgestellten Komponenten verdeutlicht.

Ziel des Experiments ist es, neue Melodien im Kinderliedstil zu erzeugen. Als Beispiele für den untersuchten Stil stehen 212 Kinderlieder in Dur aus der EsAC-Volksliedsammlung zur Verfügung.

Es werden zwei Merkmale zur Bewertung einer generierten Melodie während der evolutionären Suche herangezogen: die Abfolge der Intervalle und die Übereinstimmung der impliziten Harmonik der generierten Melodie mit einer vorgegebenen Harmonisierung. Das erste Merkmal wird mit einem neuronalen Netz aus den Musikbeispielen gelernt. Das zweite Merkmal ist regelbasiert und zeigt, dass man beliebiges Hintergrundwissen in die Melodieevolution einfließen lassen kann, sofern man es als numerische Bewertung von Merkmalskombinationen darstellen kann.

Repräsentation der Melodien. Die Beispielmelodien stehen im EsAC-Format zur Verfügung, das den Melodieverlauf in Form von Tonhöhen und Rhythmus, die Phrasierung, die Tonart und die Taktarten einer Melodie codiert (Abbildung 6.12). Aus dem EsAC-Format werden drei Ansichten gewonnen (Tabelle 6.12): Die Ansicht „DegreePitch.1“ enthält die Tonhöhen und ihre Dauern. Die Ansicht „Phrase.phrase“ gibt die Phrasierung der Melodie wieder, die im EsAC-Format durch die Zeilenumbrüche in der Melodie gekennzeichnet sind. Die Ansicht „Meter.measure“ speichert zu jedem Takt des Stücks die Taktart und gibt Aufschluss über Auftakte und Taktwechsel. Um die Übersichtlichkeit zu erhöhen, werden die Namen der Ansichten hier aus dem Typ der Ansicht[2] und einem Hinweis auf ihre Bedeutung

[2]Da die Typen der musikalischen Objekte im System MeloLab englisch benannt sind, werden bei der Beschreibung der Experimente die englischen Typbezeichner verwendet. Dadurch erschließt sich die Bedeutung der von MeloLab gezeichneten Operatorgraphen beim Lesen leichter. Beim „SegmentExtractor“ in

Ansicht	Eigenschaft	Bedeutung
DegreePitch.1	mutierbar	Tonhöhen in Stufen bezogen auf den Grundton der Tonart (EsAC)
Phrase.phrase	konstant	Phraseneinteilung (EsAC)
Meter.measure	konstant	Takt mit Taktart (EsAC)
Double.metric_weight_pitch	konstant	Metrisches Gewicht der Tonanfänge
Meter.divided_measure	konstant	Halbe Takte mit Taktart
HarmonicFunction.target_harmony	konstant	Implizite Harmonie der halben Takte der Mustermelodie
HarmonicFunction.implicit_harmony	abgeleitet	Implizite Harmonie der halben Takte einer generierten Melodie

Tabelle 6.12: Beispiel 1: Repräsentation einer Melodie mit den während der Suche gesetzten Eigenschaften. Die mit (EsAC) gekennzeichneten Ansichten stammen bei der Mustermelodie aus der Eingabe im EsAC-Format. Der Name jeder Ansicht ist aus dem Typ der Ansicht und ihrer Bedeutung zusammengesetzt.

zusammengesetzt. „DegreePitch.1“ ist die erste (und hier einzige Ansicht) mit Tonhöhen; „Phrase.phrase“ enthält Phrasen vom Typ **Phrase**; „Meter.measure“ speichert zu jedem Takt des Stücks die Taktart.

Die Tonart wird nicht explizit als Ansicht repräsentiert, sondern ist in der Stufendarstellung der Tonhöhen enthalten. Da die Stufendarstellung vom Grundton der Tonart abstrahiert, entfällt anders als bei der absoluten Tonhöhendarstellung die Notwendigkeit, alle Melodien in eine gemeinsame Tonart zu transponieren, um verschiedene Melodien beim maschinellen Lernen zueinander in Beziehung setzen zu können.

Die anderen in der Tabelle angegebenen Ansichten werden zur Erzeugung von Lernmustern und bei der Bewertung von generierten Melodien benötigt und weiter unten erklärt.

Mustermelodie, Population und Suchparameter. Als Mustermelodie für die evolutionäre Suche wird die Melodie „Schlaf, Kindlein, schlaf“ (K0001) aus dem Corpus der Kinderlieder der EsAC-Sammlung gewählt (Abbildung 6.12). Die rhythmische Struktur der Melodie wird in diesem Experiment fixiert, so dass sie sich im Lauf der Evolution nicht ändert. Die Tonhöhen werden vor Beginn der evolutionären Suche mit Hilfe des Randomisierers zufällig initialisiert. Die implizite Harmonik der Mustermelodie dient als Referenzharmonisierung für die Bewertung der generierten Melodien. Es wird eine Population mit 50 Elementen, 10 Nachkommen durch Mutation, 10 Nachkommen durch Rekombination, d.h. 5 Paaren von rekombinierten Melodien verwendet. Die Suche wird nach 1000 Epochen oder beim Erreichen der maximal möglichen Bewertung 100 abgebrochen.

Operationen auf Melodien. In diesem ersten Experiment werden nur die Tonhöhen einer Melodie mutiert, so dass während der evolutionären Suche nur die Ansicht „DegreePitch.1“ mutierbar ist. Die anderen Ansichten sind entweder konstant oder abgeleitet.

den von MeloLab generierten Bildern handelt es sich um den Koinzidenzoperator aus Kapitel 5.

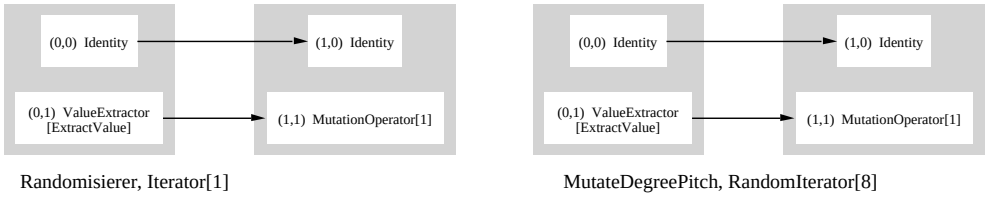


Abbildung 6.13: Experiment 1: Randomisierer „Randomisierer“(links) und Mutierer „MutateDegreePitch“ (rechts). Die Operatorgraphen unterscheiden sich nur durch die Iteratoren, der ihre Anwendung auf eine Ansicht steuern.

Die Initialisierung und die Mutation von Melodien werden beide durch einen Mutationsoperator realisiert, der eine Tonhöhe diatonisch durch eine zufällig gewählte Tonhöhe ersetzt. Die mutierte Tonhöhe ist höchstens drei Stufen vom Ausgangston entfernt. Dadurch ändert sich die Kontur einer Melodie während der Suche nur allmählich. Abbildung 6.13 zeigt den hier verwendeten Randomisierer und Mutierer. Die beiden Operatorgraphen unterscheiden sich nur durch den Steueriterator: Während der Randomisierer mit dem Standarditerator jeden Ton der Tonhöhenansicht „DegreePitch.1“ verändert, verwendet der Mutierer „MutateDegreePitch“ einen Zufalliterator „RandomIterator“, der für den Parameter $n = 8$ Schritte einer zufälligen Größe aus dem Intervall $[0.5 * n, 1.5 * n]$ zurücklegt. Im Gegensatz zum Randomisierer verändert der Iterator nur wenige Töne einer Melodie an wechselnden Positionen. Ein- und Ausgabeansicht beider Operatorgraphen ist die Ansicht „DegreePitch.1“, ein Beispiel für Operatorgraphen die nicht wie in den Beispielen aus Kapitel 5 aus einer Eingabeansicht eine Ausgabeansicht mit anderer, z.B. abstrakterer Information ableiten, sondern eine Ansicht manipulieren. Tabelle 6.13 zeigt die Ein- und Ausgabeansichten aller hier verwendeten Operatorgraphen im Überblick.

Bei der Rekombination wird die Phrasenansicht als Schnittvorlage verwendet. Da die Phrasenansicht aus der Mustermelodie übernommen wird und während der Suche konstant ist, kann eine generierte Melodie nur an den Positionen zerlegt werden, die in Abbildung 6.12 durch Atemzeichen markiert sind.

Bewertung der Melodien. Zur Bewertung der generierten Melodien werden ein neuronaler und ein regelbasierter Bewerter eingesetzt. Der neuronale Bewerter „ClassifyInterval“ in Abbildung 6.14 durchläuft die Tonhöhenansicht „DegreePitch.1“ mit einem Standarditerator und berechnet in den ersten vier Schichten ein Eingabemuster für den Neuro-Operator in der fünften Schicht. Der Operator „MaxPosEqual“ ermittelt, ob die maximale Position der Netzausgabe mit der maximalen Position des codierten Sollwerts übereinstimmt. Er vergleicht das Intervall zwischen dem aktuellen und vorhergehenden Ton mit der Ausgabe eines neuronalen Netzes, das mit Melodien im Kinderliedstil trainiert wurde und das gleiche Merkmal anhand von Kontextinformationen vorhersagt. Stimmt das Intervall an der aktuellen Position der Eingabeansicht mit dem vom neuronalen Netz vorhergesagten Intervall überein, so erhöht sich die Bewertung der Melodie.

Das neuronale Netz des Neuro-Operators wurde mit 212 Melodien in Dur aus der EsAC-Volksliedsammlung trainiert und getestet. Dabei wurden alle Muster, die aus einer Melodie

Operatorgraph: Eingabeansicht ⟶ Ausgabeansicht oder Typ
Vorbereitung
ComputeDividedMeasure: Meter.measure ⟶ Meter.divided_measure
ComputeMetricWeight: DegreePitch.1 ⟶ Double.metric_weight_pitch
ComputeImplicitHarmony: Meter.divided_measure ⟶ HarmonicFunction.target_harmony
Aktualisierer
ComputeImplicitHarmony: Meter.divided_measure ⟶ HarmonicFunction.implicit_harmony
Randomisierer
Randomisierer: DegreePitch.1 ⟶ DegreePitch.1
Mutierer
MutateDegreePitch: DegreePitch.1 ⟶ DegreePitch.1
Bewerter
ClassifyInterval: DegreePitch.1 ⟶ Double
MatchTargetWeighted: HarmonicFunction.implicit_harmony ⟶ Double

Tabelle 6.13: Beispiel 1: Auswertung von Operatorgraphen. Da die Ausgabe eines Bewerters numerisch ist, wurde statt einer Ausgabeansicht der Ausgabetyp des Bewerters angegeben.

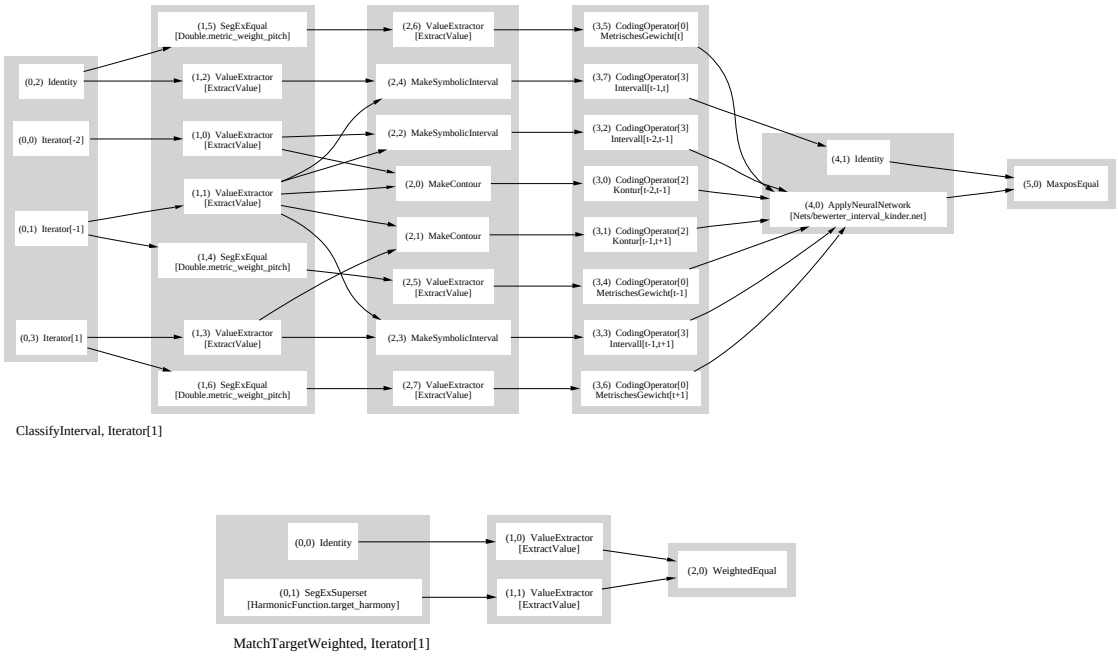


Abbildung 6.14: Experiment 1: Bewerter „ClassifyInterval“ (oben) und „MatchTargetWeighted“ (unten)

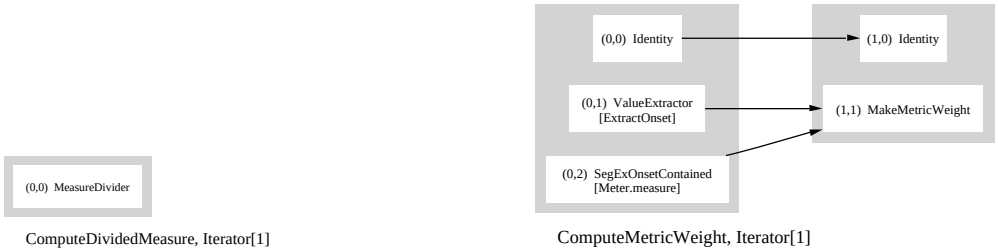


Abbildung 6.15: Experiment 1: Berechnung konstanter Ansichten mit den Operatorgraphen „ComputeDividedMeasure“ (links) und „ComputeMetricWeight“ (rechts)

stammen, gemeinsam der Trainings- oder der Testmenge zugeordnet. Die Mustermelodie K0001 gehört zur Testmenge. Als Eingabemerkmale des neuronalen Netzes dienen die Kontur und die Intervallgrößen an den Positionen $[t-2, t-1]$ und $[t-1, t+1]$ sowie die metrische Betonung an den Positionen $t-1$, t und $t+1$. Dabei bezeichnet t das aktuelle Segment der Eingabeansicht, $t-1$ das vorhergehende und $t+1$ das nächste Segment der Eingabeansicht.

Der neuronale Bewerter „ClassifyInterval“ stützt sich auf die Merkmale Intervall, Kontur und metrisches Gewicht. Mit den Tonhöhen einer Melodie ändern sich während der Melodiesuche auch ihre Intervalle und Konturen. Daher werden diese Merkmale im Bewerter „ClassifyInterval“ durch die Wissensoperatoren „MakeSymbolicInterval“ und „MakeContour“ für jede Melodie neu berechnet. Sie werden also nicht durch Ansichten repräsentiert. Das metrische Gewicht aller Einsatzzeiten von Melodietönen bleibt während der Suche gleich, da der Rhythmus in diesem Experiment nicht verändert wird. Mit dem Operatorgraphen „ComputeMetricWeight“ (Abbildung 6.15) wird vor Beginn der Suche eine Ansicht „Double.metric_weight_pitch“ für die Mustermelodie berechnet, die während der Suche konstant ist (Tabelle 6.12).

Der zweite, regelbasierte Bewerter „MatchTargetWeighted“ vergleicht die implizite Harmonik (vgl. Kapitel 2.2.3) einer generierten Melodie mit einer vorgegebenen Harmonisierung. Als Vorstufe zu einer aus Melodiebeispielen gelernten Harmonisierung wird zunächst eine feste Harmonisierung gewählt, um die Anpassung der Melodie an diese Harmonisierung evaluieren zu können. In diesem Experiment wird die implizite Harmonik für die halben Takte

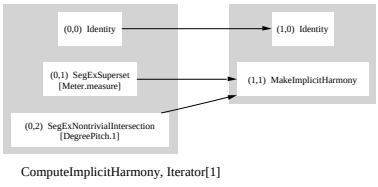


Abbildung 6.16: Experiment 1: Aktualisierer „ComputeImplicitHarmony“

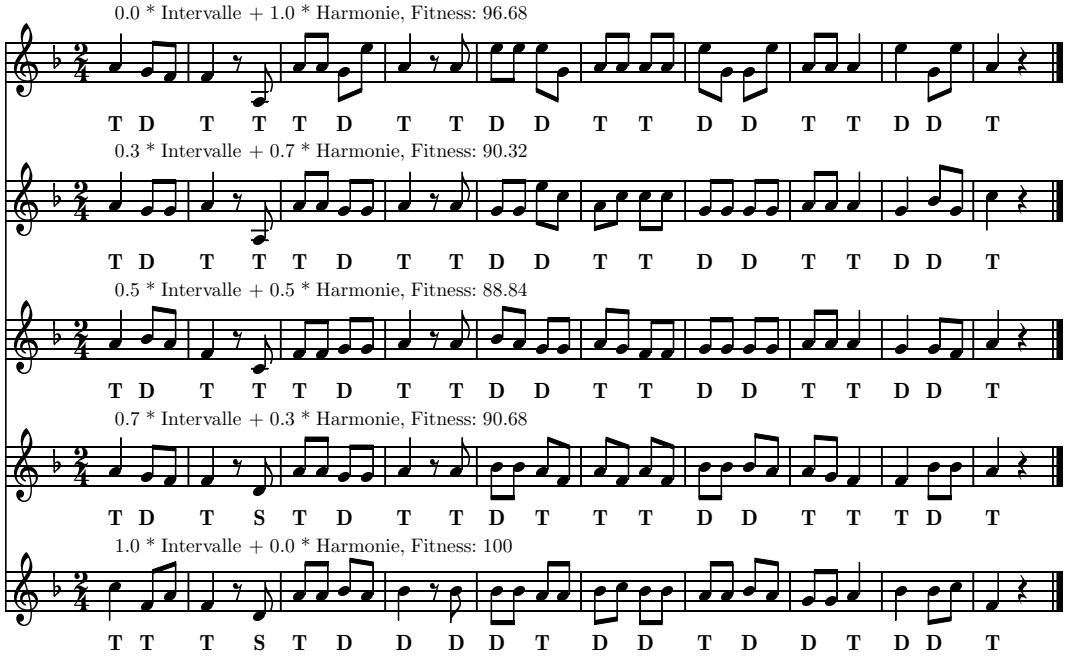


Abbildung 6.17: Am besten bewertete Melodien M1-M5 bei der evolutionären Suche. Bewertet wurde die Intervallgröße und die Übereinstimmung der impliziten mit einer vorgegebenen Harmonisierung. Die implizite Harmonisierung jeder Melodie ist hier, die vorgegebene Harmonisierung in Abbildung 6.12 angegeben.

der Mustermelodie als feste Harmonisierung gewählt. Vor Beginn der Suche werden die Takte zunächst mit dem Operatorgraphen „ComputeDividedMeasure“ geteilt und als Ansicht „Meter.divided_measure“ gespeichert. Mit dem Operatorgraphen „ComputeImplicitHarmony“ wird dann die implizite Harmonik jedes halben Takts berechnet und in der Ansicht „HarmonicFunction.target_harmony“ abgelegt. Wie die Ansicht der metrischen Gewichte bleiben auch die Ansicht der halbierten Takte „Meter.divided_measure“ und die Ansicht der Zielharmonien „HarmonicFunction.target_harmony“ während der Melodiesuche konstant. Abbildung 6.12 zeigt die impliziten Harmonien für die Mustermelodie.

Aktualisierer. Mit den Tonhöhen einer generierten Melodie ändert sich auch ihre implizite Harmonik, so dass zwischen der Mutation der Tonhöhen und der Bewertung der Melodie die von den Tonhöhen abhängige Ansicht „HarmonicFunction.implicit_harmony“ aktualisiert werden muss. Dies ist die Aufgabe des Aktualisierers „ComputeImplicitHarmony“, der schon zur Berechnung der Referenzharmonisierung „HarmonicFunction.target_harmony“ eingesetzt wurde. Für jede Melodie der Population erzeugt er bei der Aktualisierung die Ansicht „HarmonicFunction.implicit_harmony“.

Ergebnis. Um zu zeigen, wie sich das Wissen der Bewerter in den generierten Melodien niederschlägt, werden Melodien mit fünf Gewichtungen $(0, 1)$, $(0.3, 0.7)$, $(0.5, 0.5)$, $(0.7, 0.3)$

und $(1, 0)$ der beiden Bewerter erzeugt. Abbildung 6.17 zeigt die Melodien aus jedem Lauf.

Die generierten Melodien zeigen folgende Auffälligkeiten:

- Die Harmonisierung der Melodien M1, M2 und M3, in denen der harmonische Bewerter mindestens mit 0.5 gewichtet war, stimmt mit der Vorlage überein.
- Die Fitness der Melodie M1 ist nicht 100, weil das Gewicht des Treffers in die Bewertung eingeht und nicht die binäre Information, ob ein Treffer vorliegt. Durch das geringere Gewicht harmoniefremder Töne wird die Fitness reduziert.
- Die Subdominante (B-Dur) kommt nur in den Melodien M4 und M5 vor.
- Bei der Melodie M5 entsprechen alle Intervalle den Erwartungen des neuronalen Netzes; daher ist die Fitness 100.
- Die Sequenz in den Takten 5 und 6 in Melodie M3 klingt überzeugend, ist aber zufällig entstanden. Keines der verwendeten Merkmale unterstützt Sequenzierung.
- Wenn der neuronale Bewerter „ClassifyInterval“ ein geringes Gewicht hat, treten größere Intervalle auf: Im Mittel haben die Melodien (2.4, 1.5, 0.8, 1.1, 1.1) diatonische Schritte pro Intervall. In M1 ist jedes Intervall also im Mittel eine Terz (2 Schritte) bis Quarte (3 Schritte). In den Melodien M3, M4, M5 herrschen Sekundschritte vor.
- Eine höhere Gewichtung des Intervallnetzes bedingt kleinere Intervalle.
- Die Septime tritt als einziges Intervall in keiner der Melodien auf.

Insgesamt wirken besonders die drei mittleren Melodien M2, M3 und M4 musikalisch überzeugend und stilecht. Die erste Melodie M1 weist zu viele große Intervallsprünge auf, da die Intervallstruktur nicht in die Bewertung eingeht. Die letzte Melodie M5 wirkt harmonisch inkohärent, z.B. im Mittelteil durch die häufige Wiederholung des Tons B auf betonter Taktzeit, der sich nur durch die Subdominante harmonisieren lässt. Am überzeugendsten erscheint die Melodie M3, in der Intervallstruktur und implizite Harmonik in einem ausgewogenen Verhältnis zueinander stehen.

Verlauf der Suche. Abbildung 6.18 zeigt den Verlauf der evolutionären Melodiesuche für drei der fünf untersuchten Gewichtungen der Bewerter. In den linken Diagrammen ist der Verlauf der Bewertung für das beste Element einer Population und die mittlere Bewertung der Population aufgetragen. Die Bewertung des besten Elements steigt wegen der Selektionsstrategie monoton an. Die mittlere Bewertung einer Population steigt im wesentlichen an, lokal kann sie sich auch verschlechtern, wenn mehrfach vorhandene Melodien durch schlechter bewertete aber von der bisherigen Population verschiedenen Melodien ersetzt werden. Dieser Effekt tritt z.B. ungefähr in der 500. Epoche für die Gewichtung $(0.5, 0.5)$ der Bewerter auf. Eine erhebliche Verschlechterung der mittleren Populationsbewertung wurde aber in keinem der durchgeführten Experimente beobachtet. Typisch für den Verlauf der Bewertungskennzahlen ist ein schneller Anstieg am Anfang der Suche, dem eine Sättigungsphase folgt, in der keine großen Verbesserungen mehr gefunden werden. Im Detail zeigen die drei Bewertungskurven aber unterschiedliche Charakteristiken: Nachdem für die Gewichtung $(0, 1)$ die

Sättigungsphase erreicht ist, folgen mehrere kleine Verbesserungen des besten Elements der Population in kleinen Abständen. Wenige Änderungen einer Melodie scheinen hier auszureichen, um eine verbesserte beste Melodie zu finden. Die Bewertung für die Gewichtung $(0.5, 0.5)$ scheint zunächst bei etwa 80% Klassifikationsgüte gesättigt zu sein. Dann tritt jedoch etwa bei Epoche 200 eine weitere deutliche Verbesserung ein. Für die Gewichtung $(1, 0)$ wird die Suche schon nach 338 Generationen abgebrochen, da die beste Melodie eine Bewertung von 100 erreicht hat. Diese verbesserte Melodie ist durch Mutation entstanden, wie auch am nebenstehenden Diagramm zu erkennen ist. Das Diagramm zeigt neben der Diversität (*), d.h. der Anzahl unterschiedlicher Melodien in einer Population die Anzahl der mutierten (+) und rekombinierten (x) Melodien, die bei der Selektion in die nächste Population übernommen wurden. Nachdem die Population bei Epoche 145 eine Diversität von 50 erreicht hat und alle Melodien die gleiche Bewertung von 96.429% haben, liefern Mutation und Rekombination lange kein bessere Melodie. Die in Epoche 347 durch Mutation erzeugte Melodie mit Bewertung 100% ist im Diversitäts-Diagramm rechts unten zu sehen. Die Diversitäts-Diagramme für die Gewichtungen $(0, 1)$ und $(0.5, 0.5)$ zeigen anfänglich eine leichte Verringerung der Diversität der Population, die sich jedoch auf hohem Niveau bei etwa 43 Melodien einpendelt. Durchgängig werden ein größer Teil mutierten und eine geringerer Teil der rekombinierten Melodien in die Population übernommen, was wie das Bewertungsdiagramm zeigt, zu einer allmählichen Verbesserung der mittleren Bewertung der Population führt.

Insgesamt ist der logarithmische Verlauf der Bewertungsdiagramme auffällig, der die Vermutung nahelegt, dass die hier verwendete Mutations- bzw. Rekombinationsoperation ab einer gewissen Qualität der Population ihr Potential einbüßt, wesentliche positive Veränderungen an einer Melodie zu bewirken. Eine Erklärung könnte sein, dass die hier verwendete Mutation ohne Rücksicht auf den musikalischen Kontext einzelne Tonhöhen verändert, die Bewerter den Kontext jedoch berücksichtigen. Für die weiteren Experimente ergibt sich daraus die Frage, ob man durch die stärkere Berücksichtigung der melodischen Struktur die Sättigungsphase der Suche hinauszögern kann.

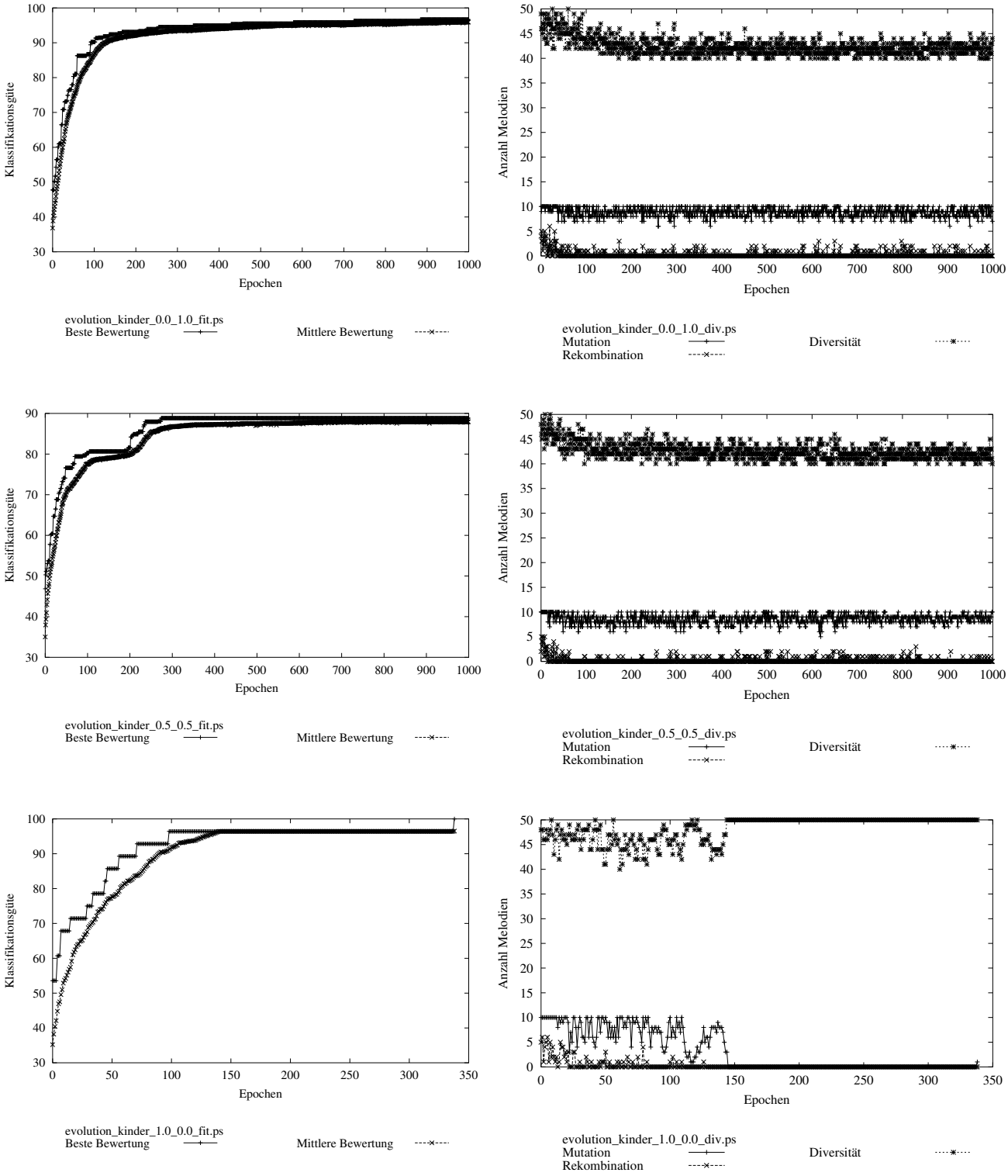


Abbildung 6.18: Entwicklung der Bewertung und der Diversität bei der Generierung der Melodien in Abbildung 6.17 für die Gewichtungen $(0, 1)$, $(0.5, 0.5)$ und $(1, 0)$ der Bewerter.

Tonhöhe[t]					
Kinder	TH[t-1]	TH[t+1]	TH[t+2] (pentatonisch)	implizite Harmonie[Takt]	harmonisches Gewicht[Takt]
Irland	TH[t-1]	TH[t+1]	TH[t-2] (pentatonisch)	implizite Harmonie[Takt]	Bewegtheit[P]
Shanxi	TH[t-1]	TH[t+1]	TH[t+2] (pentatonisch)	Intervall [t-1, t+1]	implizite Harmonie[Takt]
Intervall[t-1,t]					
Kinder	Alteration[t-2]	Alteration[t-1]	TH[t-2] (pentatonisch)	Dauer[t-1]	mittlere Phrasendauer
Irland	Intervall [t-2, t-1]	Intervall [t-1, t+1]	Letzter Ton (pentatonisch)	Synkope?	Punktierung?
Shanxi	Dauer[t-2]	Kontur [t+1, t+2]	Harmonisches Gewicht[Takt]	Metrisches Gewicht[t-2]	Phrasenform [P+1]
Phrasenform der aktuellen Phrase					
Kinder	OktaveTH[t+2]	Dauer[t+1]	Intervall Phrasengrenzen[P]	Phrasenrichtung [P]	Phrasenrichtung [P+1]
Irland	Alteration[t-2]	Phrasenrichtung [P]	Auftakt?	Phrasenform [P+1]	Phrasenrichtung [P+1]
Shanxi	Dauer[t-2]	Kontur [t-2 t-1]	Dauernverhältnis [t-2,t-1]	Punktierung?	Anzahl der Phrasen
Ambitus der aktuellen Phrase					
Kinder	Intervall [t-1, t+1]	Intervall [t+1, t+2]	Intervall Phrasengrenzen[P]	Phrasenform[P]	Bewegtheit[P]
Irland	1. Phrasenton[P] (pentatonisch)	Phrasenform[P]	Letzter Ton (pentatonisch)	Mittlere Phrasendauer	Ambitus[P+1]
Shanxi	Dauer[t+2]	Intervall[t-1, t+1]	Letzter Phrasenton[P] (pentatonisch)	Intervall Phrasengrenzen[P]	Bewegtheit[P]

Tabelle 6.14: Die fünf relevantesten Eingabemerkmale zum Lernen des Zielmerkmals (fettgedruckt) für die drei betrachteten Melodiestile.

6.2.3 Ein komplexeres Beispiel

Nachdem die evolutionäre Melodiegenerierung im vorigen Abschnitt an einem einfachen Beispiel vorgestellt wurde, werden nun Melodien im Shanxi-Stil unter Verwendung einer größeren Auswahl von Merkmalen generiert. Wie im ersten Beispiel wird auch hier die evolutionäre Melodiegenerierung zur Erzeugung der Melodien eingesetzt. Dabei steuern lernbasierte Bewerter den Verlauf der evolutionären Suche. Es soll untersucht werden, ob man neuronale Bewerter mit Beispielen eines Stils trainieren kann, so dass die mit ihrer Hilfe generierten Melodien den gelernten Stil widerspiegeln. Im Gegensatz zum Experiment im vorigen Abschnitt gibt es hier keine Vorgaben (Rhythmus, Harmonie) mehr.

Optimierung von Bewertern. Während im Beispiel des letzten Abschnitts zwei von Hand definierte Bewerter zur Generierung von Kinderliedern mit der evolutionären Melodiegene-

Aktualisierer	Metrisches Gewicht	Phrasenform
	Rhythmusklasse	Implizite Harmonie
	Bewegtheit	Existenz eines Auftakts
	Ambitus einer Phrase	
Randomisierer	Tonhöhen	
Mutierer	Tonhöhe	Rhythmus
Bewerter	Tonhöhe	Intervall
	Rhythmusklasse einer Phrase	Ambitus einer Phrase
	Übereinstimmung eines rhythmischen Musters mit dem Klassenprototypen	

Tabelle 6.15: Komponenten zur Generierung der Melodien in Abbildung 6.19

rierung verwendet wurden, wird nun das Merkmalsselektionsverfahren aus Abschnitt 6.1.1 zum Entwurf von Bewertern herangezogen.

Ausgangsmaterial für das Melodiemodell sind die Merkmale aus den Selektionsexperimenten, die in Tabelle 6.2 aufgeführt sind. Sie umfassen zeitlich lokale, auf die Ebenen der Takte und der Phrasen bezogene sowie hinsichtlich einer Melodie globale Merkmale. Während in den Selektionsexperimenten immer die Tonhöhe als Zielmerkmal verwendet wurde, um die Vergleichbarkeit der experimentellen Ergebnisse zu gewährleisten, sollen nun auch andere Merkmale als die Tonhöhe gelernt werden. Die hier betrachteten Zielmerkmale sind die Tonhöhe an einer aktuellen Stelle t, das metrische Gewicht$[t]$, das Intervall$[t-1,t]$, die Kontur$[t-1,t]$, das Dauernverhältnis$[t-1,t]$, die Rhythmusklasse und implizite Harmonie des aktuellen Takts, die Phrasenform, die Bewegtheit, die Richtung und der Ambitus der aktuellen Phrase. Ein Index $t-1$ bezeichnet dabei den linken Nachbarn des Ereignisses mit Index t. Da nun auch Merkmale auf übergeordneten Zeitebenen wie der Takt- und der Phrasenebene gelernt werden, wird die Menge der potentiell relevanten Eingabemerkmale um die direkten Nachbarn der Merkmale auf Takt- und Phrasenebene ergänzt, so dass 63 Merkmale als Eingabe für die Merkmalsselektion zur Verfügung stehen.

Mit Hilfe der Vorwärtsselektion werden die Eingabemerkmale der neuronalen Klassifikatoren für die Zielmerkmale optimiert, wobei jeder der drei Melodiestile – Kinderlieder, irische Volkslieder und Shanxi-Lieder – getrennt betrachtet wird. Bei der Optimierung jedes Klassifikators wird das zu prognostizierende Merkmal aus der Menge der Eingabemerkmale ausgeschlossen, damit die Suche keinen trivialen Zusammenhang als Optimum findet.

Tabelle 6.2.3 zeigt die Eingabe und Zielmerkmale der vier Bewerter für die Merkmale Tonhöhe$[t]$, Intervall$[t-1,t]$, Phrasenform und Ambitus einer Phrase, die im folgenden bei der evolutionären Melodiegenerierung eingesetzt werden. Die zugehörigen Klassifikatoren hatten unter den optimierten Modellen die beste Klassifikationsgüte, ohne dabei einen trivialen Zusammenhang zwischen Merkmalen herzustellen. Da sich bei der Einzelstilmodellierung musikalischer Stile in Abschnitt 6.1.1 gezeigt hat, dass die von der Vor- und Rückwärtssuche gefundenen am besten bewerteten Merkmalsmengen häufig viele irrelevante Merkmale enthalten, wurden hier nur fünf Eingabemerkmale pro Bewerter selektiert. Diese Bewerter sind auf Teilbereiche des Merkmalsraums spezialisiert und werden bei der Melodiegenerierung kombiniert.

Die selektierten Eingabemerkmale in Tabelle 6.2.3 stellen eine breite Auswahl aus der vorgegebenen Menge dar. Wie sich bereits gezeigt hat, spielt die implizite Harmonik eines Takts bei der Prognose eines Melodietons in allen betrachten Stilen eine Rolle. Die hier gefundene (statistische) Bedeutung der impliziten Harmonik für den Shanxi-Stil ist überraschend, da die implizite Harmonik hier basierend auf den harmonischen Funktionen Tonika, Dominante, Subdominante definiert ist, die mit der pentatonischen Grundskala des Shanxi-Stils nicht in einem offensichtlichen Zusammenhang stehen. Auch wenn die implizite Harmonik als Eingabemerkmal zur Prognose der Tonhöhe im Shanxi-Stil beitragen mag, scheint die Prognose dreiklangsbasierter Harmonien für pentatonische Melodien eher fragwürdig.

Bemerkenswert ist auch, dass bei der Prognose des Intervalls „Intervall[t-1,t]“ für die irischen Volkslieder neben den zu erwartetenden benachbarten Intervallen auch globale Merkmale wie der letzte Ton der Melodie sowie die Existenz einer Synkope und eines punktierten Rhythmus von den 63 potentiellen unter die wichtigsten fünf Merkmale aufgenommen wurde. Eine solche Konstellation legt die Hypothese nahe, dass die globalen Merkmale als eine Art Schalter wirken könnten, die zwischen unterschiedlichen Intervallmodi wechseln. Interessant ist auch die Tatsache, dass pentatonisch codierte Tonhöhen, anders als man erwarten würde, nicht vorwiegend im chinesischen Melodiestil, sondern in allen Stilen mit einem gewissen Abstand zum aktuellen Ton t auftreten. Man erhält den Eindruck, dass zur Beschreibung musikalischer Strukturen, die sich nicht in unmittelbarer Nähe des gesuchten Merkmals befinden, eine abstraktere Darstellung für den Klassifikator angemessen sein könnte als zur Beschreibung unmittelbar benachbarter Merkmale.

Konfiguration der Melodiegenerierung. Neben den durch Merkmalsselektion optimierten Bewertern müssen vor Beginn der evolutionären Melodiegenerierung die weiteren Komponenten des Verfahrens festgelegt werden (vgl. Tabelle 6.11).

Im Vergleich zum ersten Melodiegenerierungsexperiment, bei dem nur die Tonhöhe variiert wurde, werden nun alle Bestandteile der Melodie bei der Optimierung verändert. Neben dem Tonhöhenmutierer wird daher auch ein Rhythmusmutierer verwendet. Dieser erhält ein zufällig ausgewähltes Paar aufeinanderfolgender Segmente der Tonhöhenansicht und spaltet oder verschmilzt sie unter Berücksichtigung des metrischen Rasters. Dabei werden Dauern nicht über Taktgrenzen hinweg verschmolzen und ungewöhnliche Kombinationen von Dauer und metrischer Position vermieden, die durch die Trainingsdaten nicht gestützt werden. Der Rhythmusmutierer ist ein Beispiel für einen Wissensoperator, der allgemeines Wissen über den betrachteten Stil modelliert. Dadurch wird eine allgemeine Filterung der erzeugten Rhythmen vorgenommen. Die Bewertung der Rhythmen erfolgt aber durch den neuronalen Rhythmusklassenbewerter.

Der neuronale Rhythmusklassenbewerter lernt den Zusammenhang zwischen der Rhythmusklasse eines Takts und den zwei vorherigen und nachfolgenden Rhythmusklassen. Da bei diesem Bewerter nicht die korrekte Vorhersage einer einzelnen Rhythmusklasse aus beliebigen Merkmalen, sondern die Modellierung der Klassenabfolge in der Melodie im Vordergrund stand, wurden die Eingabemerkmale des Rhythmusklassenbewerters nicht optimiert. Die Abfolge der Rhythmusklassen sagt noch nichts darüber aus, wie typisch die Tondauern eines Takts für die zugehörige Rhythmusklasse sind. Daher wurde der Rhythmusklassenbewerter durch einen regelbasierten Rhythmusbewerter ergänzt, der die Übereinstimmung einer Tonfolge mit dem Prototypen der zugehörigen Rhythmusklasse misst. Dieser Bewer-

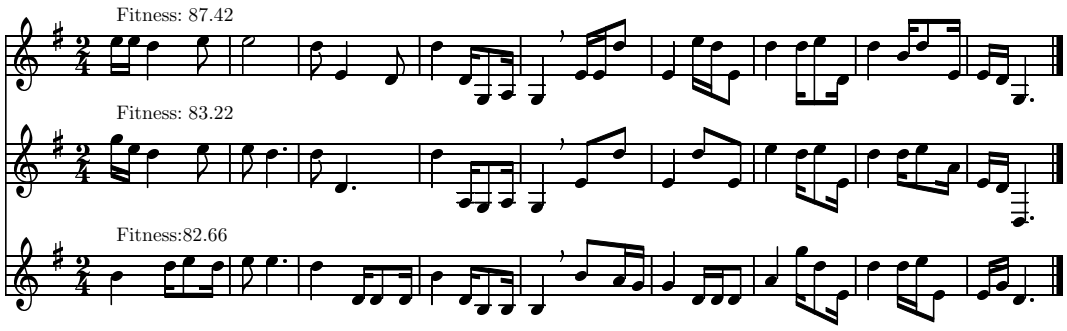


Abbildung 6.19: Drei durch evolutionäre Melodiegenerierung mit Bewertern im Shanxi-Stil erzeugte Melodien.

ter benötigt die Rhythmusklasse jedes Takts, die in der Aktualisierungsphase der Suche durch den Rhythmus-Aktualisierer berechnet wird. Der Rhythmusklassen-Aktualisierer ist ein Operatorgraph, der einen Wissensoperator zur Klassifikation von Rhythmen enthält. Der Wissensoperator kennt eine stiltypische Liste von rhythmischen Prototypen und ermittelt zu einer neuen Folge von Dauern mit einem Pattern-Matching-Ansatz den Prototypen mit den Fehlzuordnungen minimaler Gesamtentfernung.

Um ein Merkmal wie den Rhythmus zu optimieren, arbeiten bei der evolutionären Melodiegenerierung also verschiedene Komponenten zusammen: Wenn eine Melodie (z.B. mit dem Rhythmus-Mutierer) verändert wurde, berechnet der Rhythmusklassenaktualisierer die Ansicht der Rhythmusklassen neu. Diese dient als Eingabe für den Rhythmusklassenbewerter und den Bewerter für die Übereinstimmung von Dauernfolgen und Rhythmusklassen-Prototypen.

Die Rhythmuskomponenten werden ergänzt durch einen Tonhöhen- und einen Intervallbewerter sowie einen Ambitusbewerter, der kontrolliert, ob eine Phrase einen stiltypischen Ambitus besitzt.

In diesem Generierungsexperiment werden der Randomisierer und der Mutierer zur zufälligen Veränderung von Tonhöhen aus dem vorigen Abschnitt übernommen. Als Schnittansicht für die Rekombination von Melodien wird wie zuvor die aus den Ausgangsdaten ablesbare Phrasenansicht verwendet.

Ergebnis. Abbildung 6.19 zeigt drei Melodien aus der Endpopulation der Melodiegenerierung für den Shanxi-Stil nach 500 Epochen. Die Population bestand aus 50 Melodien, aus denen in jeder Epoche je 10 Nachkommen durch Mutation und Rekombination gebildet wurden. Zur Diversifizierung wurde das in Abschnitt 2.1.5 beschriebene Auswahlverfahren eingesetzt. Die Bewerter waren folgendermaßen gewichtet: Der Tonhöhenbewerter war mit 0.3 am stärksten gewichtet, gefolgt vom Intervallbewerter mit einem Gewicht von 0.2. Rhythmusklassen- und Rhythmusbewerter waren mit je 0.2 gewichtet, während der Ambitusbewerter mit einem Gewicht von 0.1 den schwächsten Einfluss hatte. Als Vorlage für die Randomisierung diente die Melodie J0054 aus dem Shanxi-Corpus der Datenbank. Ihre

Phrasenstruktur wurde übernommen und während der Evolution beibehalten. Der Rhythmus wurde bei der Randomisierung zunächst übernommen, aber durch Mutation sukzessive verändert. Die Tonhöhen wurden bereits bei der Randomisierung verändert.

Die abgebildeten Melodien zeigen stiltypische Synkopen in unterschiedlicher melodischer Gestalt. Die erste Phrase ist in allen Melodien in 2+1.5+1 Takte gegliedert. Das tiefe synkopische Motiv in der zweiten Hälfte des vierten Takts wirkt in allen Melodien wie eine Bestätigung des Melodiebeginns. Charakteristisch ist das fallende Terz-Sekund-Motiv am Anfang der zweiten Melodie, das man in der Instrumentalmusik häufig auch als Arpeggio findet. Der Beginn der dritten Melodie weist einen Melodiebogen auf, der an der pentatonischen Skala orientiert ist. In der zweiten Phrase findet man in den drei Melodien Septimsprünge, die durch die Trainingsbeispiele gestützt werden, wenn auch nicht in dieser Häufigkeit. Zum Teil lassen sich die Septimsprünge als Oktavfehler interpretieren. Große Sprünge sind aber in den Trainingsbeispielen nicht selten, da Teile der Melodie in verschiedene Stimmregister gelegt werden. Der Anfang der zweiten Phrase der dritten Melodie weist eine melodische Linie auf, zu der die am Anfang der Melodie bereits aufgetretenen Synkopen einen Kontrast bilden.

Die gezeigten Melodien weisen also eine Reihe von musikalischen Merkmalen auf, die sich auch in den Trainingsbeispielen wiederfinden. Im Vergleich zu einem Melodieergänzungssystem wie MeloGeNet mit einer Rasterdarstellung der Zeit konnten hier Rhythmen flexibel mutiert werden. Der Rhythmusklassen- und der Rhythmusbewerter stellten dann das Auftreten der stiltypischen rhythmischen Motive sicher.

6.3 Merkmalssuche

Bei der stilistischen Einordnung und Bewertung von Melodien durch Personen mit vergleichbarem kulturellen Hintergrund lassen sich häufig Übereinstimmungen feststellen. Dennoch fehlen im Gegensatz zu Harmonielehre und Kontrapunkt objektive Kriterien zur Beschreibung und Unterscheidung melodischer Stile, was daran zu erkennen ist, dass nur eine geringe Anzahl von systematischen Melodielehren existiert (vgl. [20]). In diesem Kapitel wurde gezeigt, wie man mit Hilfe von Merkmalsselektionverfahren aus einer Menge vorgegebener melodischer Merkmale die für eine Lernaufgabe relevanten ermitteln kann. Noch interessanter wäre ein Verfahren, bei dem die untersuchten Merkmale nicht vorgegeben werden müssen, sondern selbständig gefunden werden. In diesem Abschnitt wird an einem Beispiel angedeutet, wie ein solches Suchverfahren aussehen könnte.

Aus der Perspektive übergeordneter Rahmenverfahren wie der Merkmalsselektion wurden Operatorgraphen bisher wie Variablen behandelt. Sie wurden als abgeschlossene Einheiten angesehen. Gleichzeitig dienten sie der Modularisierung bei der Berechnung von Merkmalen. Dort war das Augenmerk darauf gerichtet, dass sich Operatorgraphen aus atomaren Operationen zusammensetzen.

Die Merkmalssuche kombiniert beide Sichtweisen. Ziel der Merkmalssuche ist es, automatisch neue musikalische Merkmale zu finden, die eine gewünschte Eigenschaft wie z.B. Relevanz für die Lösung einer Lernaufgabe besitzen.

Die Implementierung eines Merkmalssuchverfahrens hätte den Rahmen dieser Arbeit gesprengt. Im folgenden wird jedoch an einem Beispiel eine mögliche Vorgehensweise bei der Merkmalssuche skizziert, um das diesbezügliche Potential der Operatorgraphen zu veranschaulichen.

Im ersten Schritt wird ein Evaluierungkriterium zur Bewertung der Operatorgraphen ausgewählt. Beim maschinellen Lernen könnte dies z.B. der Einfluss eines Merkmals (dargestellt durch einen Operatorgraphen) auf die Lösung der Lernaufgabe sein.

Nun wird eine Menge von Operatorgraphen zusammengestellt, die bereits bekannte oder bewährte Merkmale berechnen. Da die Operatorgraphen empirisch ausgewertet werden sollen, benötigt man zusätzlich eine Menge von Kontext-Kompositionen, auf die die Operatorgraphen anwendbar sind (vgl. Definition 5.8). Abbildung 6.20 zeigt einfache Operatorgraphen, die aus dem Patterngraphen in Abbildung 6.2 gewonnen wurden, indem zu einem Codierungsoperator der Teilgraph des Patterngraphen extrahiert wurde, der zur Berechnung des Codierungsoperators notwendig ist.

Im nächsten Schritt wählt man eine Menge von Transformationen aus, mit denen Operatorgraphen verändert werden können. In Abbildung 6.21 sind Transformationen angegeben, die die Operatorgraphen aus Abbildung 6.20 ineinander überführen. Beispiele für Transformationen sind das Einfügen oder Entfernen von Operatoren aus einem Operatorgraphen oder die Änderung von Parametern eines Operators. Bei Anwendung der Operationen muss sichergestellt werden, dass der resultierende Operatorgraph ebenfalls auf die vorgegebenen Kontext-Kompositionen anwendbar ist. Die formale Definition der Ein- und Ausgabetypen von Operatoren in Kapitel 5 war notwendig, um die Anwendbarkeit von Operatorgraphen automatisch prüfen zu können.

Im Verlauf der Merkmalssuche werden dann wie bei der evolutionären Suche mit Hilfe der Transformationen neue Operatoren erzeugt und entsprechend ihrer Bewertung für den nächsten Suchschritt selektiert oder verworfen. Die Suche wird solange fortgeführt, bis ein Abbruchkriterium, beispielsweise eine bestimmte Anzahl von Suchschritten, erreicht ist.

Wenn man bei der Merkmalssuche Operatorgraphen mit beliebig kombinierten Operatoren zulässt, bewegt man sich in einem sehr großen Suchraum, in dem das Auffinden sinnvoller Operatorgraphen schwierig erscheint. Schränkt man jedoch den Aufbau der Operatorgraphen ein, beispielsweise indem man den Operatorgraphen in Schichten unterteilt, in denen nur bestimmte Klassen von Operatoren zugelassen sind, so verkleinert sich der Suchraum und wird systematischer exploriert. Im Beispiel in Abbildung 6.20 sind die Operatoren schichtweise angeordnet. Die ersten beiden Schichten enthalten zeitspezifische Operatoren (Iteratoren und Koinzidenzoperatoren), in der dritten Schicht befinden sich Wertextraktoren, in der vierten Wissensoperatoren und in der fünften ein Codierungsoperator.

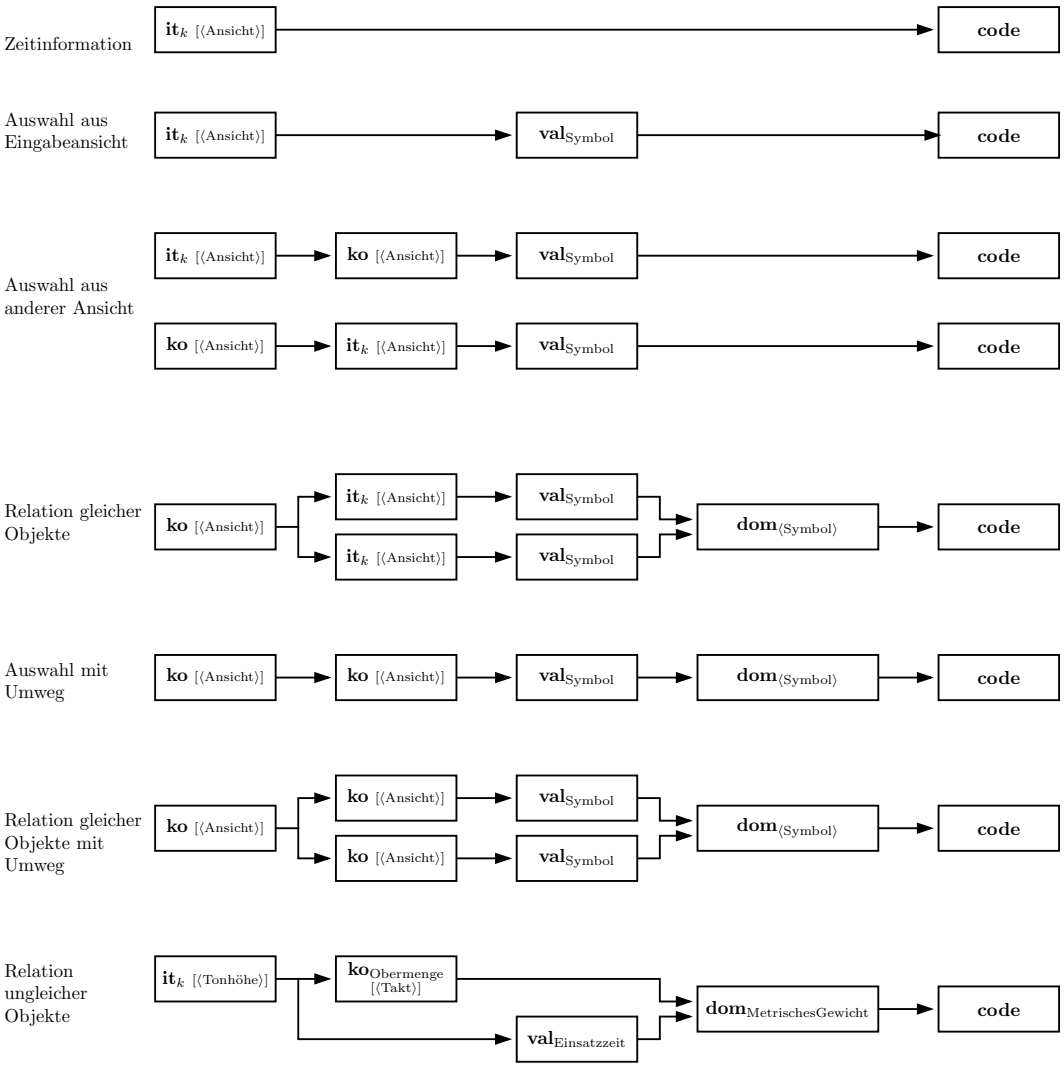


Abbildung 6.20: Klassifikation einfacher Operatorgraphen

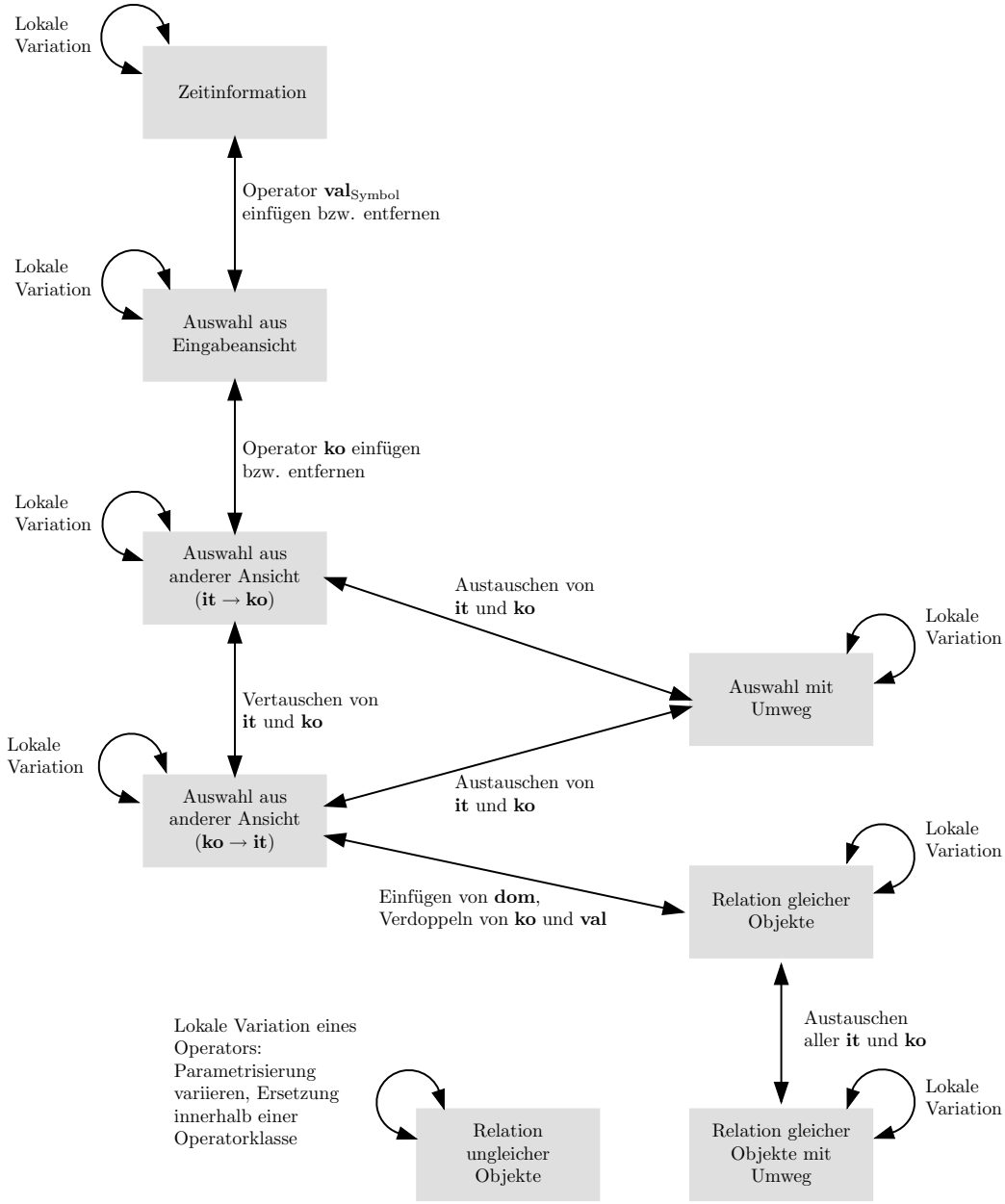


Abbildung 6.21: Transformation der Operatorgraphen aus Abbildung 6.20

Zusammenfassung und Ausblick

Gegenstand der Arbeit war die Analyse von Melodien mit Methoden des maschinellen Lernens. Ziel war es, Methoden zum Auffinden stilprägender Eigenschaften von Melodien zu entwickeln. Mit ihnen sollten Melodien gleichen Stils erkannt und neue Melodien in einem vorgegebenen Stil erzeugt werden. Zur Extraktion stiltypischen Wissens aus Beispielen sollten lernbasierte Methoden wie z.B. neuronale Netze eingesetzt werden.

Aus Sicht der Informatik handelt es sich bei Melodien um diskrete Zeitreihen mit einer komplexen inneren Struktur. Wenn diese innere Struktur nur implizit gegeben ist (wie z.B. bei der Repräsentation von Melodien als Tonfolgen), wird sie beim maschinellen Lernen stilistischer Strukturzusammenhänge von Melodien nicht ausreichend berücksichtigt. Bereitet man die Ausgangsdaten jedoch so auf, dass die implizite innere Struktur explizit sichtbar wird, so lassen sich stilprägende Eigenschaften von Melodien mit Hilfe von maschinellem Lernen aufspüren. Als Vorverarbeitungsverfahren sind Standardmethoden wie Whitening oder Differenzenbildung nicht ausreichend, da sie typische Strukturzusammenhänge des Anwendungsgebiets wie z.B. die latente Harmonik einer Melodie außer Acht lassen. Vielmehr benötigt man ein Verfahren, das sich bei der Vorverarbeitung auf problemspezifisches Wissen stützt. In dieser Arbeit wurde ein neuer Ansatz zur Aufbereitung der inneren Strukturzusammenhänge diskreter Zeitreihen entwickelt, der Wissen über das Anwendungsgebiet explizit integriert.

Operatorgraphen. Die Vorverarbeitung von Zeitreihen für das maschinelle Lernen zielt darauf ab, diese in eine Darstellung zu bringen, die ein erfolgreicheres Lernen ermöglicht. Um eine problemspezifische Transformation einer Zeitreihe zu beschreiben, wurde mit *Operatorgraphen* in diesem Ansatz ein neues Beschreibungsmittel für Merkmale diskreter Zeitreihen eingeführt.

Mit Hilfe eines Operatorgraphen lässt sich eine Zeitreihe in eine andere Darstellung überführen, die die Perspektive desjenigen Merkmals in den Vordergrund rückt, das der Operatorgraph beschreibt. Eine solche transformierte Darstellung einer Zeitreihe wird in Anlehnung an die unterschiedlichen Datensichten in Softwaresystemen als *Ansicht* (engl. *view*) der Zeitreihe bezeichnet.

Erweiterbare Zeitreihengrammatik. Als Grundlage für die Definition von Ansichten und Operatorgraphen wurde eine *Erweiterbare Zeitreihengrammatik* entwickelt, die sich aus einem vorgegebenen zeitspezifischen- und einem austauschbaren problemspezifischen Anteil zusammensetzt. Mit dieser Grammatik lassen sich die zeitreihen- und anwendungsbezogene

Symboltypen spezifizieren und generieren, auf denen die weiteren Modelle dieser Arbeit aufbauen. Insbesondere ist es dadurch möglich, in Operatorgraphen zeitreihenspezifische von problemspezifischen Aspekten zu trennen, so dass die Integration von neuem Problemwissen in den Ansatz erleichtert wird. Im Vergleich zur festen Implementierung musikalischer Transformationen bietet die Verwendung von Operatorgraphen den methodisch wichtigen Vorteil, dass Operatorgraphen unabhängig von ihrer Struktur immer auf dieselbe Weise ausgewertet werden. Es ist daher möglich, Algorithmen mit frei austauschbaren, von Operatorgraphen dargestellten Merkmalen zu implementieren, ohne dass auf die tatsächliche Beschaffenheit der Merkmale Bezug genommen werden muss. Operatorgraphen bilden damit eine einheitliche Schnittstelle zwischen einem generischen Algorithmus einerseits und einer problembezogenen Repräsentation musikalischer Daten andererseits.

Anwendungen. Operatorgraphen wurden in dieser Arbeit bei der Lösung verschiedener musikalischer Problemstellungen eingesetzt. Bei der *Stilerkennung* bestand die Aufgabe darin, Merkmale zu finden, die einem lernbasierten System die Zuordnung von Musikbeispielen zu vorgegebenen Stilen erlauben. Mit Hilfe von Merkmalsselektionsverfahren gelang es, stiltypische Merkmale in Volksliedmelodien zu finden, anhand derer einzelne Melodiestile charakterisiert bzw. gegeneinander abgegrenzt werden konnten. Diese Merkmale wurden dann bei der *Generierung neuer Melodien* in den untersuchten Stilen mit evolutionären Algorithmen verwendet. Dabei spielten wiederum Operatorgraphen eine wichtige Rolle, indem sie während der evolutionären Optimierung eingesetzt wurden, um zufällige Melodien zu erzeugen, Melodien durch Mutation zu verändern und sie zu bewerten. Die Bewertung von Melodien mit Operatorgraphen setzte *Neurooperatoren* ein, also Operatoren, die ein neuronales Netz auswerten. Operatorgraphen erweisen sich damit als ein Werkzeug, mit dem die Interaktion lernbasierter Klassifikatoren untereinander und mit zusätzlichem Problemwissen spezifiziert werden kann. Auch eine hybride Bewertung von Melodien, die Regelwissen und lernbasierte Elemente verknüpft, wird auf diese Weise ermöglicht.

Ausblick. Das in dieser Arbeit geschaffene theoretische Fundament zur Repräsentation von Zeitreihen bildet den Ausgangspunkt für eine Vielzahl weiterer Anwendungen. So wäre es interessant, anstelle der in dieser Arbeit untersuchten Volksliedmelodien komplexere Melodien wie beispielsweise Kunstlieder zu modellieren. Insbesondere der durch Koinzidenzoperatoren realisierte Synchronisationsmechanismus zwischen Ansichten schafft eine wesentliche Grundlage für die Untersuchung mehrstimmiger Musikstücke. Darüberhinaus wurde bei der Konzeption stets darauf Wert gelegt, den Ansatz so allgemein zu halten, dass er auf auch nichtmusikalische Zeitreihen wie beispielsweise Börsenkurse anwendbar ist.
Um einen musikwissenschaftlichen Anwenderkreis zu erreichen, ist es erforderlich, die im System MeloLab umgesetzte Funktionalität auf konkrete musikwissenschaftliche Probleme anzuwenden. Denkbar sind hier die Rekonstruktion musikalischer Fragmente und die Behandlung von Fragestellungen der Authentizitätsanalyse. Die stilbasierte Suche von Musikstücken in umfangreichen Musikcorpora stellt eine weitere Anwendung der Stilmodellierung im Bereich des *Music Information Retrieval* dar.
Als theoretisch wichtigste Fortführung der vorliegenden Arbeit erscheint mir die Konkretisierung und Umsetzung der in Abschnitt 6.3 an einem Beispiel skizzierten Methode zur Merkmalssuche. Dies würde einerseits zu einer Theorie der Merkmalsextraktion beitragen und könnte andererseits ein computergestütztes Werkzeug zur Entwicklung einer systematischen Melodielehre schaffen.

Anhang A

Das System MeloLab

Dieser Anhang stellt die Experimentierumgebung MeloLab vor, die den hier entwickelten Ansatz implementiert. Das System umfasst einen Interpreter für Operatorgraphen sowie Operatoren und Algorithmen für die Experimente in Kapitel 6. Als musikalische Datenstrukturen werden von einer Zeitreihengrammatik erzeugte Typen implementiert, deren Anwendungsteil sich eng an das Beispiel aus Abschnitt 4.2.6 anlehnt.

Architektur von MeloLab**.** Abbildung A.1 zeigt die wichtigsten Komponenten des Systems MeloLab. Jeder Kasten entspricht einer Komponente des Systems. Einfache Pfeile stellen eine "benutzt"-Beziehung zwischen den Komponenten dar; Doppelpfeile stehen für die Kommunikation von MeloLab mit externen Datenspeichern (dargestellt durch Zylinder). Basisfunktionalität wie Befehle zum Speichern und Anzeigen von Datenstrukturen wurden in Abbildung A.1 der Übersichtlichkeit halber weggelassen.

Die graphische Benutzeroberfläche kommuniziert über eine Zwischenschicht zur Befehlssteuerung mit dem Anwendungskern. Damit ist eine Entkopplung des Anwendungskerns von der Benutzeroberfläche möglich, deren Aufgaben im Batch-Betrieb von einer XML-Steuerdatei übernommen werden können. Die für den Benutzer sichtbare Funktionalität des Anwendungskerns (fett umrahmte Kästen) wird von der Befehlssteuerung angesprochen. Sie ist in allgemeine Aufgaben wie das Generieren von Ansichten und Lernmustern und das Trainieren von Klassifikatoren sowie musikspezifische Aufgaben wie die Melodiegenerierung, die Stilmodellierung, die Stilunterscheidung und die Stilerkennung unterteilt. Diese Komponenten benutzen interne Subkomponenten wie den Interpreter für Operatorgraphen und den Klassifikator. Die Repräsentation der Musikdaten, Lernmuster und Klassifikatoren sowie das Ein- und Auslesen von Daten übernehmen die Subkomponenten zur Daten-, Patternmengen- und Neuronale-Netze-Verwaltung. Die Trennung in einen Klassifikator, der eine Schnittstelle für übergeordnete Algorithmen bildet und Verwaltungskomponenten für Lernmuster und neuronale Netze, die die Funktionalität implementieren, wurde vorgenommen, um den Ansatz auch auf andere Klassifikatoren erweitern zu können, ohne aufrufende Algorithmen anpassen zu müssen.

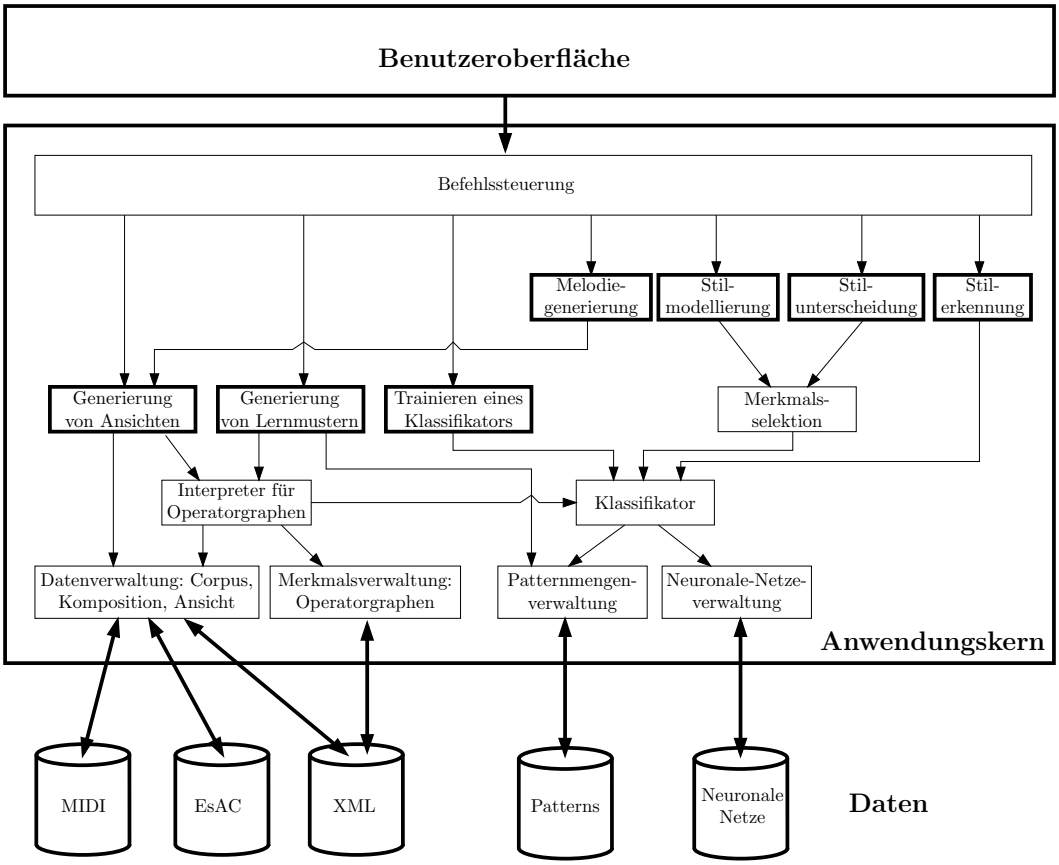


Abbildung A.1: Architektur des Systems MeloLab

Allgemeine Aufgaben. Die Generierung einer Ansicht dient der Analyse von Musikstücken mit Operatorgraphen. Sie wird wie alle für den Benutzer sichtbaren Aufgaben von der Befehlssteuerung angestoßen und ruft den Interpreter für Operatorgraphen auf, um einen Operatorgraphen für einen Corpus oder ein einzelnes Musikstück auszuwerten. Das Ergebnis des Interpreters wird in einer neuen Ansicht des jeweils als Eingabe verwendeten Musikstücks der Datenverwaltung gespeichert. Da der Interpreter Operatorgraphen auf Ansichten anwendet, greift er sowohl auf die Datenverwaltung also auch auf die Merkmalsverwaltung zu, die Operatorgraphen liest, speichert und für andere Komponenten bereithält.

Bei der Generierung von Lernmustern wird ebenfalls ein Operatorgraph (Patterngraph) ausgewertet, der seine Eingabe aus der Datenverwaltung bezieht. Die erzeugten Lernmuster werden von der Patternmengenverwaltung in eine Patterndatei geschrieben.

Das Trainieren eines einzelnen Klassifikators ist für den Anwender von Interesse, wenn aus früheren Merkmalsoptimierungsläufen optimierte Konfigurationen von Merkmalen bekannt

sind, für die kein Klassifikator (mehr) vorhanden ist, oder für die ein Klassifikator mit neuen Daten trainiert werden soll. Das Trainieren eines Klassifikators verwendet die allgemeine Klassifikatorkomponente, die ihrerseits auf die Patternmengen- und die Neuronale-Netze-Verwaltung zugreift. Letztere implementieren einen Klassifikator als Komitee neuronaler Netze und stützen sich auf die N++-Bibliothek [72].

Musikspezifische Aufgaben. Die musikspezifischen Aufgaben umfassen die in Kapitel 6 verwendeten Algorithmen (obere Zeile fett umrahmter Kästen in Abbildung A.1).

Die Melodiegenerierung ist als evolutionäre Suche implementiert, die eine Population von Melodien schrittweise verbessert. Die Melodien werden durch Ansichten repräsentiert, die mit Operatorgraphen initialisiert, mutiert und bewertet werden. Die Melodiegenerierung benötigt daher die Komponente zur Generierung von Ansichten.

Stilmodellierung und Stilunterscheidung sind Optimierungsaufgaben, die beide ein Merkmalsselektionsverfahren verwenden. Dieses bewertet Mengen von Merkmalen durch das Trainieren und Testen eines Klassifikators und greift daher auf die Klassifikatorkomponente zu.

Die Stilerkennung stützt sich auf Klassifikatoren, die mit Hilfe der Stilunterscheidung optimiert wurden. Sie verwendet daher lediglich die Klassifikatorkomponente, selektiert aber selbst keine Merkmale.

Steuerung von MeloLab**.** Das System MeloLab kann mit einer graphischen Benutzeroberfläche oder mit Hilfe einer Steuerdatei im XML-Format angesprochen werden. Abbildung A.2 zeigt einen typischen Screenshot der graphischen Benutzeroberfläche von MeloLab.

Alle Datenstrukturen, die zum Lösen einer Aufgabe benötigt werden, werden in MeloLab zu einer logischen Einheit – dem *Experiment* – zusammengefasst. Ein Experiment besteht aus musikalischen Daten (*Corpus*), Klassifikatoren (*Classifiers*), Operatorgraphen zur Berechnung von Ansichten und Lernmustern (*Features, PatternGraphs*). Der Screenshot stellt den Arbeitsbereich (*Workspace*) für das Experiment "Merkmalsselektion" dar, das zwei Corpora der EsAC-Sammlung (Chinesische Volkslieder und Kinderlieder) als musikalische Daten enthält. Der Operatorgraph *ComputeAmbitus*, der den Ambitus einer Phrase berechnet, wird im unteren Fenster visualisiert. Diese Darstellung wird von MeloLab mit Hilfe der Graphvisualisierungssoftware `GraphViz` [25] automatisch aus der internen Repräsentation eines Operatorgraphen generiert. Mit dem Dialog auf der rechten Seite des Screenshots kann ein Merkmalsselektionsverfahren konfiguriert und gestartet werden, wie es z.B. zur Modellierung musikalischer Einzelstile in Kapitel 6 verwendet wurde.

Im Rahmen dieser Arbeit wurde nur die Funktionalität der graphischen Benutzeroberfläche implementiert, die der unmittelbaren Unterstützung der hier durchgeführten Experimente diente. Dies sind die Visualisierung und Manipulation der Konfiguration eines Experiments mit dem im Screenshot abgebildeten Strukturbaum, die automatisierte Visualisierung von Operatorgraphen, die sich als Hilfsmittel bei der Zusammenstellung von Operatorgraphen als sehr hilfreich erwiesen hat, und Dialoge zum Aufruf der oben skizzierten Aufgaben des Systems.

Um MeloLab für einen an lernbasierter Musikanalyse interessierten Personenkreis einsetzbar zu machen, wurden beim Entwurf des Systems ein Editor für Ansichten und für

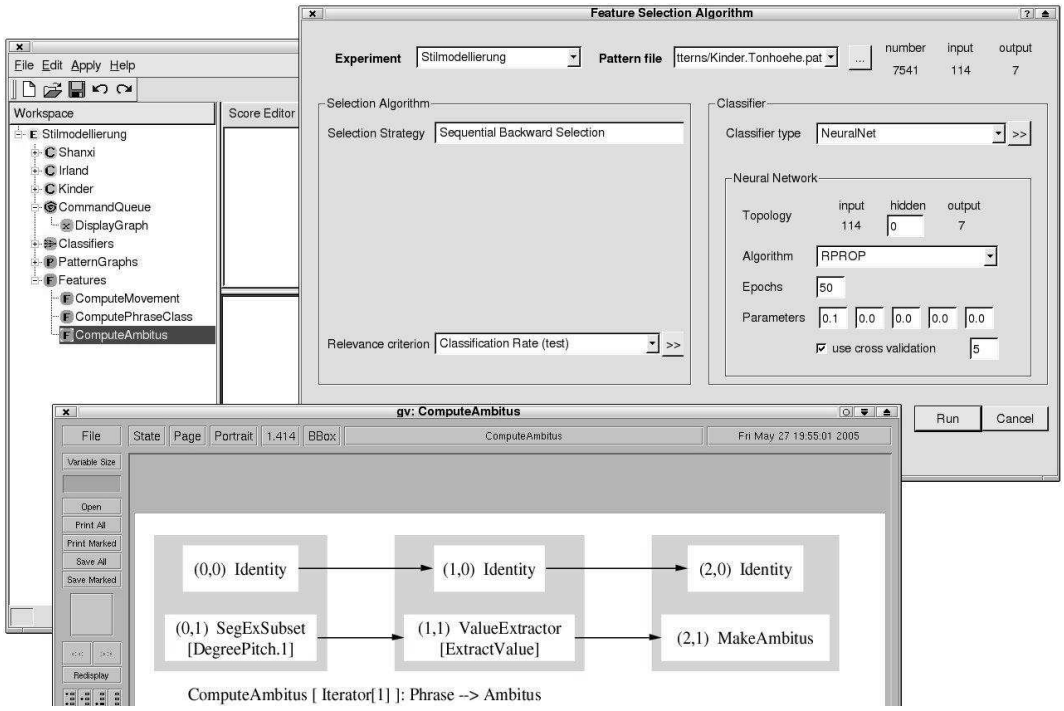


Abbildung A.2: Screenshot des Systems MeloLab

Operatorgraphen vorgesehen, dessen Implementierung für diese Arbeit jedoch keinen inhaltlichen Gewinn gebracht hätte. Stattdessen wurden die Experimente in Kapitel 6 mit XML-Steuerdateien konfiguriert.

Datenspeicherung und Befehlssteuerung durch XML-Datei. Die Konfiguration eines Experiments wird im Format XML [11] gespeichert, wobei das genaue Format der XML-Datei von einer *Schema*-Vorlage [23] festgelegt wird. Die Schema-Vorlage definiert XML-Tags für die Datenstrukturen von MeloLab und gibt vor, wie diese geschachtelt werden dürfen. Das XML-Format wurde gewählt, weil es als Textformat im Gegensatz zu binären Formaten von Hand editierbar ist, weil auf einen XML-Parser mit Schema-Validierung (Xerces [87]) zurückgegriffen werden konnte, der das Einlesen von Konfigurationsdateien übernimmt, und weil XML wegen seiner Redundanz gut komprimierbar ist.

Abbildung A.3 zeigt einen Ausschnitt der XML-Konfigurationsdatei für MeloLab, mit der der Screenshot in Abbildung A.2 erzeugt wurde. Im Tag `<melolab>` ist die Schema-Datei `melolab.xsd` angegeben, die Struktur der Konfigurationsdatei bestimmt. Weiter unten findet man die XML-Darstellung des Merkmals `ComputeAmbitus`, das aus einem Standard-Iterator mit Inkrement 1 und einem Operatorgraphen besteht. Der Operatorgraph besteht aus drei Knotenschichten mit je zwei Operatoren. Der Wissensoperator `MakeAmbitus` erwar-

tet als Eingabetyp Objekte vom Typ **Pitch** (im Beispiel für die Erweiterbare Zeitreihengrammatik in Abschnitt 4.2.6 **Tonhöhe**) und berechnet ein Objekt vom Typ **Ambitus**, das in der gegenwärtigen Implementierung zwei Objekte vom Typ **MidiPitch** enthält. Die anderen fünf Knoten des Operatorgraphen sind hier durch Faltung des XML-Tags ausgeblendet, was durch die Ellipsen ... kenntlich gemacht ist.

Die XML-Konfigurationsdatei kann auch dazu verwendet werden, MeloLab ohne graphische Benutzeroberfläche zu steuern. Teil jedes MeloLab-Experiments ist eine Kommando-Warteschlange, die beim Aufruf von MeloLab ohne Oberfläche sofort abgearbeitet wird. Dieser Mechanismus erlaubt es, MeloLab automatisch in einem Netzwerk auf anderen Computern zu starten. Dies ist z.B. sinnvoll, wenn man mehrere Suchverfahren mit langer Laufzeit auf verschiedene Rechner verteilen will. In Abbildung A.3 enthält die Kommando-Warteschlange `CommandQueue` den Befehl `DisplayGraph`, dessen Zieladresse `target` auf den Wert `Features|ComputeAmbitus` gesetzt ist. Wenn der Befehl `DisplayGraph` ausgeführt wird, sucht MeloLab im Experiment, zu dem die Kommando-Warteschlange gehört, zunächst den Merkmalsvektor `Features` und in diesem das Merkmal `ComputeAmbitus`. Wird es gefunden, so wird es wie im Screenshot in Abbildung A.2 angezeigt.

Erweiterbarkeit in MeloLab**.** Beim Entwurf des hier verfolgten Ansatzes spielte die Erweiterbarkeit von Problemwissen eine große Rolle. Dies wirft die Frage auf, wie sich diese Zielsetzung im System MeloLab wiederfindet.

Die einfachste Art der Erweiterbarkeit besteht in der Definition neuer Operatorgraphen mit Hilfe existierender Operatoren. Da die Operatorgraphen interpretiert werden, ist dazu keine Programmierung notwendig.

Wenn die vorhandenen Symboltypen und Operatoren für das betrachtete Anwendungsfeld nicht ausreichen, muss der Anwendungsteil der zugrundeliegenden Zeitreihengrammatik erweitert oder ersetzt werden. In der praktischen Umsetzung bedeutet dies, dass man neue Wert- und Operatorklassen implementieren muss, die von einer vorgegebenen Basisklasse abgeleitet werden und deren Schnittstelle nach außen durch die Methoden der zugehörigen Basisklasse vorgegeben ist. Der Rest des Systems wird durch diese Ergänzung nicht berührt, da er lediglich die Schnittstelle der Basisklasse verwendet.

```
<?xml version = '1.0' encoding = 'iso-8859-1' standalone = 'no'?>
<melolab xsi:noNamespaceSchemaLocation="melolab.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
    <Experiment name="Stilmodellierung" >
        <Features>
            <Feature name="ComputeAmbitus">
                <Comment>
                    Berechnet den Ambitus einer Phrase
                </Comment>
                <Value name="input_type" type="Phrase"/>
                <Value name="output_type" type="Ambitus"/>
                <Operator type="Iterator">
                    <ValueVector name="input" type="Segment"/>
                    <ValueVector name="result" type="Segment"/>
                    <ValueVector name="parameters">
                        <Int name="increment">1</Int>
                        <Bool name="randomize">0</Bool>
                        <String name="default_result">Empty</String>
                    </ValueVector>
                </Operator>
                <OperatorGraph>
                    <Layer id="0">
                        <Node id="0"> ...
                        <Node id="1"> ...
                    </Layer>
                    <Layer id="1">
                        <Node id="0"> ...
                        <Node id="1"> ...
                    </Layer>
                    <Layer id="2">
                        <Node id="0"> ...
                        <Node id="1">
                            <Operator type="MakeAmbitus">
                                <ValueVector name="input" type="Pitch"/>
                                <ValueVector name="result" type="Ambitus">
                                    <Value type="Ambitus">
                                        <Value name="min_pitch" type="MidiPitch"> ...
                                        <Value name="max_pitch" type="MidiPitch"> ...
                                    </Value>
                                </ValueVector>
                            </Operator>
                            <Int name="predecessor">1</Int>
                        </Node>
                    </Layer>
                </OperatorGraph>
            </Feature>
            <Feature name="ComputePhraseClass"> ...
            <Feature name="ComputeMovement"> ...
        </Features>
        <PatternGraphs> ...
        <Classifiers> ...
        <Corpus name="Kinder" load_mode="Load" save_mode="NoSave"> ...
        <Corpus name="Irland" load_mode="Load" save_mode="NoSave"> ...
        <Corpus name="Shanxi" load_mode="Load" save_mode="NoSave"> ...
        <CommandQueue>
            <Command name="DisplayGraph" active="1">
                <String name="target">Features|ComputeAmbitus</String>
            </Command>
        <CommandQueue>
    </Experiment>
</melolab>
```

Abbildung A.3: Ausschnitt der XML-Konfigurationsdatei für MeloLab, mit der der Screenshot in Abbildung A.2 erzeugt wurde.

Literaturverzeichnis

[1] S. Phon Amnuaisuk and Geraint Wiggins. The four-part harmonisation problem: A comparison between genetic algorithms and a rule-based system. In *Proceedings of the AISB-99 Symposium on Musical Creativity, Edinburgh*, 1999.

[2] C. Anagnostopoulou and G. Westermann. Classification in music: A computational model for paradigmatic analysis. In *Proceedings of the 1997 International Computer Music Conference, Thessaloniki, Greece*, pages 125–128. International Computer Music Association, 1997.

[3] Mario Baroni and Carlo Jacoboni. Analysis and generation of Bach's chorale melodies. In G. Stefani, editor, *Proceedings of the First International Congress on the Semiotics of Music*, Pesaro, Italy, 1975. Centro di Iniziativa Culturale.

[4] Matthew I. Bellgard and Chi Ping Tsang. On the use of an effective Boltzmann machine for musical style recognition and harmonization. In *Proceedings of the 1996 International Computer Music Conference, Hong Kong*, pages 461–464, San Francisco, CA:, 1996. International Computer Music Association.

[5] Jonathan Berger and Dan Gang. A neural network model of metric, perception and cognition in the audition of functional tonal music. In *Proceedings of the 1997 International Computer Music Conference*, Thessaloniki, 1996.

[6] Jamshed J. Bharucha and Peter M. Todd. Modeling the perception of tonal structure with neural nets. In Peter M. Todd and D. Gareth Loy, editors, *Music and Connectionism*, pages 128–137. MIT Press, Cambridge, MA, 1991.

[7] J.A. Biles. GenJam: A genetic algorithm for generating jazz solos. In *Proceedings of The International Computer Music Conference 1994, Århus, Denmark*. International Computer Music Association, 1994.

[8] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.

[9] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. North-Holland, New York, 1976.

[10] Heinz Braun and Martin Riedmiller. RPROP: A fast adaptive learning algorithm. In E. Gelenbe, U. Halici, and N. Yalabik, editors, *Proceedings of the International Symposium on Computer and Information Sciences VII 1992*, pages 279–286, Paris, 1992. EHEI Press.

[11] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, F. Yergeau, and J. Cowan. Extensible Markup Language (XML) 1.1, XML 1.1, W3C Recommendation, 2004.

[12] Emilios Cambouropoulos. The Local Boundary Detection Model (LBDM) and its application in the study of expressive timing. In *Proceedings of the International Computer Music Conference (ICMC01), Havana, Cuba*, 2001.

[13] Emilios Cambouropoulos and Alan Smaill. Similarity and categorisation inextricably bound together: The unscramble machine learning algorithm. In *Proceedings of the Interdisciplinary Workshop on Similarity and Categorisation, University of Edinburgh, Edinburgh*, 1997.

[14] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a Library for Support Vector Machines. Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm, 2002.

[15] Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137–167, 1959.

[16] Darrell Conklin and Ian H. Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, March 1995.

[17] David Cope. *Computers and Musical Style*. Oxford University Press, Oxford, 1991.

[18] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, New York, NY, USA, 1991.

[19] Ewa Dahlig. EsAC Database: Essen Associative Code and Folksong Database, 1994.

[20] D. de la Motte. *Melodie: Ein Lese- und Arbeitsbuch*. dtv/Bärenreiter, 1993.

[21] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. Wiley Interscience, New York, 2001.

[22] Christopher Evans. Chinese traditional music – pitches, scales and modes. http://www.cechinatrans.demon.co.uk/ctm-psm.html.

[23] D.C. Fallside and P. Walmsley. XML Schema Part 0: Primer Second Edition, W3C Recommendation, 2004.

[24] Allen Forte and Steven E. Gilbert. *Introduction to Schenkerian analysis*. Norton, New York, 1982.

[25] E.R. Gansner and S.C. North. *An open graph visualization system and its applications to software engineering*. AT&T, http://www.research.att.com/sw/tools/graphviz/GN99.pdf, 2000.

[26] P. M. Gibson and J. A. Byrne. NEUROGEN, music composition using genetic algorithms and cooperating neural networks. In *Proceedings of the Second International Conference on Artificial Neural Networks*, Conf. Publ. No. 349, pages 309–313, Bournemouth, UK, 1991. IEE, London.

[27] Robert O. Gjerdingen. Using connectionist models to explore complex musical patterns, addendum. In Peter M. Todd and D. Gareth Loy, editors, *Music and Connectionism*, pages 138–149. MIT Press, 1991.

[28] Jean-Baptiste Goyeau. Ein Wrapper-Verfahren zur Merkmalsselektion mit Mutual Information bei musikalischen Klassifikationsproblemen. Diplomarbeit, Fakultät für Informatik, Universität Karlsruhe, 2002.

[29] G. Grosche, V. Ziegler, D. Ziegler, and E. Zeidler, editors. *Bronstein-Semendjajew - Taschenbuch der Mathematik - Neubearbeitung*. B.G.Teubner Verlagsgesellschaft, Leipzig, 1995.

[30] Günter Haußwald. *Musikalische Stilkunde*. Heinrichshofen's Verlag, Wilhelmshaven, 1973.

[31] W. B. Hewlett and Eleanor Selfridge-Field. Humdrum: Music Tools for UNIX Systems. *Computing in Musicology*, 7:66–67, 1991.

[32] Hermann Hild, Johannes Feulner, and Wolfram Menzel. HARMONET: A neural net for harmonizing chorales in the style of J.S. Bach. In J. E. Moody, S. J. Hanson, and Richard P. Lippmann, editors, *Advances in Neural Information Processing Systems 4 (NIPS-4)*, pages 267–74. Morgan Kaufmann, 1992.

[33] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie*. Pearson Studium, München, 2. Auflage, 2002.

[34] Dominik Hörnel. SYSTHEMA - Analysis and automatic synthesis of classical themes. In *Proceedings of the 1993 International Computer Music Conference, Tokyo, Japan*, 1993.

[35] Dominik Hörnel. MELONET I: Neural nets for inventing baroque-style chorale variations. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

[36] Dominik Hörnel. *Lernen musikalischer Strukturen und Stile mit neuronalen Netzen*. PhD thesis, Fakultät für Informatik, Universität Karlsruhe, 2000. Shaker Verlag.

[37] Dominik Hörnel and Wolfram Menzel. Learning musical structure and style with neural networks. *Computer Music Journal*, 22(4):44–62, Winter 1998.

[38] Dominik Hörnel and Frank Olbrich. Comparative style analysis with neural networks. In *Proceedings of the 1999 International Computer Music Conference, Beijing*, pages 433–436, 1999.

[39] Dominik Hörnel and Thomas Ragg. Learning Musical Structure and Style by Recognition, Prediction and Evolution. In D. Rossiter, editor, *Proceedings of the 1996 International Computer Music Conference, Hong Kong*, pages 59–62. International Computer Music Association, 1996.

[40] David Huron. Design principles in computer based music representation. In Alan Marsden and Anthony Pople, editors, *Computer Representations and Models in Music*, pages 5–39. Academic Press, London, 1992.

[41] International MIDI Association. *Standard MIDI Files 1.0.* International MIDI Association, Los Angeles, 1988.

[42] T. Järvinen, Petri Toiviainen, and Jukka Louhivuori. Classification and categorization of musical styles with statistical analysis and self-organizing maps. In *Proceedings of the AISB-99 Symposium on Musical Creativity, Edinburgh*, pages 54–57, 1999.

[43] Brad Johanson and Riccardo Poli. GP-music: An interactive genetic programming system for music generation with automated fitness raters. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 181–186, University of Wisconsin, Madison, Wisconsin, USA, 1998. Morgan Kaufmann, San Francisco, CA, USA.

[44] Margaret Johnson. Neural networks and style analysis: A neural network that recognizes bach chorale style. In *Atti del X Colloquio di Informatica Musicale*, pages 109–116, 1993.

[45] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[46] Teuvo Kohonen. *Self-Organization and Associative Memory.* Berlin, Springer, 1984.

[47] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 284–292. Morgan Kaufmann, 1996.

[48] Mineichi Kudo and Jack Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33(1):25–41, 2000.

[49] Joachim Langnickel. Entwurf und Implementierung eines Musikkompositionssystems zur Generierung von Musikstücken im Stile der Klavierwalzer Frederic Chopins. Master's thesis, Institut für Betriebs- und Dialogsysteme, Fakultät für Informatik, Universität Karlsruhe, 1991.

[50] Philippe Leray and Patrick Gallinari. Feature Selection with Neural Networks. *Behaviormetrika*, 26(1), 1998.

[51] Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music.* MIT Press, Cambridge, 1983. 368p.

[52] D. Lidov and J. Gabura. A melody writing algorithm using a formal language model. *Computer Studies in the Humanities*, 4(3-4):138–148, 1973.

[53] B. Lindblom and Johan Sundberg. Towards a generative theory of melody. *Swedish Journal of Musicology*, 52:77–88, 1970.

[54] Christiane Linster. *'GET RHYTHM' - a Musical Application for Neural Networks*. PhD thesis, Gesellschaft für Mathematik und Datenverarbeitung mbH, Sankt Augustin, 1989.

[55] Guerino Mazzola. *Geometrie der Töne, Elemente der Mathematischen Musiktheorie*. Birkhäuser, Basel, 1990.

[56] Guerino Mazzola. *Topos of Music*. Birkhäuser, 2001.

[57] Wolfram Menzel. Problem solving with neural networks. In U. Ratsch, M.M. Richter, and I.-O. Stamatescu, editors, *Intelligence and Artificial Intelligence - An Interdisciplinary Debate*, pages 162–177. Springer, Heidelberg, 1998.

[58] Ulrich Michels. *dtv-Atlas zur Musik, Band 1*. dtv und Bärenreiter, 1981.

[59] Ulrich Michels. *dtv-Atlas zur Musik, Band 2*. dtv und Bärenreiter, 1981.

[60] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[61] Michael C. Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multiscale processing. *Connection Science*, 6(2+3):247–80, 1994.

[62] Michael C. Mozer and Todd Soukup. Concert: A connectionist composer of erudite tunes. In *Advances in Neural Information Processing Systems 3 (NIPS 3), San Mateo, California*. Morgan Kaufmann, 1991.

[63] P. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, 26(9):917–922, 1977.

[64] E. Narmour. *The Analysis and Cognition of Basic Melodic Structures*. Chicago, University of Chicago Press, 1990.

[65] E. Narmour. *The Analysis and Cognition of Melodic Complexity*. Chicago, University of Chicago Press, 1992.

[66] Jean-Jacques Nattiez. *Music and Discourse: Toward a Semiology of Music. Translated by Carolyn Abbate*. Princeton University Press, Princeton, 1990.

[67] Christian Pampus. Support-Vektor-Maschinen in der Anwendung auf musikalische Klassifikationsprobleme. Master's thesis, Institut für Logik, Komplexität und Deduktionssysteme, Fakultät für Informatik, Universität Karlsruhe, 2002.

[68] Pavel Pudil, F. J. Ferri, Jana Novovičová, and Josef Kittler. Floating search methods for feature selection with nonmonotonic criterion functions. In *Proceedings of the 12th ICPR*, volume 2, pages 279–283, 1994.

[69] Ernst Pöppel. *Grenzen des Bewußtseins: Über Wirklichkeit und Welterfahrung*. Deutsche Verlags-Anstalt, 1988.

[70] Thomas Ragg. *Problemlösung durch Komitees neuronaler Netze*. PhD thesis, Fakultät für Informatik, Universität Karlsruhe, 2000.

[71] Christoph Reuter. Musikalische Würfelspiele (Musical Dice Games) (deutsch/englisch). CD-ROM, 2001.

[72] Martin Riedmiller and Heinz Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The Rprop Algorithm. In *Proceedings of the ICNN*, pages 586–591, 1993.

[73] Curtis Roads and John Strawn. Grammars as representations for music. In *Foundations of Computer Music*. MIT Press, Cambridge USA, 1987.

[74] R. Rojas. *Theorie der neuronalen Netze: Eine systematische Einführung*. Springer, Berlin, 1996.

[75] Stanley Sadie, editor. *The New Grove Concise Dictionary of Music*. Macmillan, 1994.

[76] Eleanor Selfridge-Field, editor. *Beyond MIDI: the handbook of musical codes*. MIT Press, Cambridge, MA, USA, 1997.

[77] C. Shannon. Prediction and entropy of printed english. *Bell Systems Technical Journal*, 30:50–64, 1951.

[78] Petr Somol, Pavel Pudil, Jana Novovičová, and Pavel Paclík. Adaptive Floating Search Methods in Feature Selection. *Pattern Recognition Letters*, 20(11-13):1157–1163, 1999.

[79] Mathieu Steelandt. Erweiterung des Grouper-Algorithmus von Temperley auf allgemeine musikalische Merkmale. Diplomarbeit, Fakultät für Informatik, Universität Karlsruhe, 2002.

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

[81] William Forde Thompson. Eugene Narmour: A Review and Empirical Assessment. *Journal of the American Musicological Society*, 49, 1996.

[82] Peter M. Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4), 1989.

[83] Petri Toiviainen. Modeling the target-note technique of bebop-style jazz improvisation: an artificial neural network approach. *Music Perception,*, 12(4):399–413, 1995.

[84] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, N.Y., 1995.

[85] Lutz Volkmann. *Graphen und Digraphen. Eine Einführung in die Graphentheorie.* Springer, Wien, New York, 1991.

[86] Dorothea Wagner. Informatik III, Skript zur Vorlesung, 2004. WS 04/05.

[87] XERXES C++ Parser for XML 1.0. http://xml.apache.org/xerces-c.

[88] W. Zorn. Musik und Informatik – ein Brückenschlag. Manuskript, Institut für Betriebs- und Dialogsysteme, Univerität Karlsruhe, 1988.

Veröffentlichungsverzeichnis

[89] D. Hörnel and K. Höthker. A Learn-Based Environment for Melody Completion. In *Proceedings of the XII Colloquium on Musical Informatics, Gorizia*, 1998.

[90] K. Höthker, D. Hörnel, and C. Anagnostopoulou. Investigating the Influence of Representations and Algorithms in Music Classification. *Computers and the Humanities*, 35(1):65–79, 2001.

[91] Karin Höthker. Modelling the motivic process of melodies with markov chains. In *Proceedings of the 1999 International Computer Music Conference*, Beijing, 1999.

[92] Karin Höthker. Exponat "HarmoTheater". Ausstellung "Das Netz. Sinn und Sinnlichkeit vernetzter Systeme" im Museum für Kommunikation in Frankfurt, Hamburg, Nürnberg und Berlin, 2002-2003.

[93] Karin Höthker and Dominik Hörnel. Harmonizing in Real-Time with Neural Networks. In *Proceedings of the International Computer Music Conference*, pages 527–530, Berlin, 2000.

[94] Karin Höthker, Belinda Thom, and Christian Spevak. Melodic Segmentation: Evaluating the Performance of Algorithms and Musicians. Technical Report 2002-3, Faculty of Computer Science, University of Karlsruhe, 2002.

[95] Thomas Noll, Jörg Garbers, Karin Höthker, Christian Spevak, and Tillman Weyde. Opuscope – Towards a Corpus-Based Music Repository. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, 2002.

[96] Christian Spevak, Belinda Thom, and Karin Höthker. Evaluating Melodic Segmentation. In Christina Anagnostopoulou, Miguel Ferrand, and Alan Smaill, editors, *Music and Artificial Intelligence. Proceedings of the Second International Conference, ICMAI 2002*, number 2445 in LNAI, pages 168–182. Springer, 2002.

[97] Belinda Thom, Christian Spevak, and Karin Höthker. Melodic Segmentation: Evaluating the Performance of Algorithms and Musical Experts. In *Proceedings of the International Computer Music Conference (ICMC)*, Göteborg, 2002.

[98] Karin Höthker und Boris-Alexander Bolles. Komponieren wie Bach – künstlich oder künstlerisch? Interaktives Exponat in der Ausstellung Computer.Gehirn im Heinz Nixdorf MuseumsForum in Paderborn, 24.10.2001 - 28.04.2002.

[99] Karin Höthker und Boris-Alexander Bolles. Komponieren wie Bach – künstlich oder künstlerisch? Interaktives Exponat in der Dauerausstellung des Heinz Nixdorf MuseumsForums in Paderborn, seit 15.1.2004.

[100] Karin Höthker und Boris-Alexander Bolles. Komponieren wie Bach – künstlich oder künstlerisch? Interaktives Exponat in der Ausstellung "Elementa 1" im Landesmuseum für Technik und Arbeit in Mannheim, seit 16.5.2004.

[101] D. Hörnel und K. Höthker. Künstl(er)i(s)che Neuronale Netze lernen Musikstile. In Antje Erben, Clemens Gresser, and Arne Stollberg, editors, *Grenzgänge - Übergänge: Musikwissenschaft im Dialog. Bericht über das 13. Internationale Symposium des DVSM [Dachverband der Studierenden der Musikwissenschaft]*. Hamburg (von Bockel Verlag), Frankfurt am Main, Oktober 1998, 2000.

Symbolverzeichnis

Index

Lebenslauf

Persönliche Daten

Name:	Karin Höthker
Geburtsdatum:	30. Juni 1969
Geburtsort:	Marburg
Anschrift:	Körnerstr. 34, 76135 Karlsruhe

Schulbildung

1975-1979	Gemeinschaftsgrundschule Nord in Jülich
1979-1988	Gymnasium Zitadelle in Jülich Abitur mit Notendurchschnitt 1,2

Universität

8/1988-10/1988	RWTH Aachen:	Studienvorkurs Mathematik
10/1988-6/1996	RWTH Aachen:	Mathematik-Studium
	Abschluss:	Diplom-Mathematikerin
	Abschlussnote:	sehr gut
	Diplomarbeit:	Cycles of length 2 modulo 4 in 3-connected graphs (Note: sehr gut)

Studienbegleitende Beschäftigung

1991-1993 Studentische Hilfskraft am Hochschuldidaktischen Zentrum der RWTH Aachen

- Übungsbetreuung im Rahmen der Vorlesungen “Informatik im Maschinenbau” und “Kybernetische Verfahren der Ingenieurwissenschaften”
- Projektmitgestaltung sowie C- und C++-Programmierung in einem Forschungsprojekt zur Videoidentifizierung von Containern in einem Umschlagbahnhof

10/1996-8/1997 Leitung des Projekts “Schnupperstudium für Schülerinnen an der RWTH Aachen” als wissenschaftliche Hilfskraft im Frauenbüro der RWTH Aachen

Beschäftigung am ILKD der Universität Karlsruhe (TH)

Projekte:	9/1997-8/1999	"Informationsstrukturen in der Musik"
	9/1999-4/2001	"Integrierte Entwicklung von Komitees Neuronaler Netze"
	5/2001-11/2003	"Modellierung melodischer Strukturen mit lernbasierten Verfahren"
Praktika:	WS 99/00, 00/01	"Probleme lösen mit Neuronalen Netzen"
Seminare:	WS 97/98, 98/99, 99/00, 00/01, 01/02, 02/03	"Ausgewählte Kapitel der Musikinformatik"
Proseminar:	SS 2002	"Ausgewählte Kapitel der Musikinformatik: Modellierung musikalischer Strukturen"
Vorlesung:	SS 2001	Betreuung einer großen Übung zur Vorlesung Informatik II, Thema: Informationstheorie

Exponate

12/1999-5/2000	Interaktives Exponat "HarmoTheater" zur Echtzeit-Harmonisierung von Melodien mit neuronalen Netzen in der Ausstellung "Jahrhundertwenden 1000-2000. Rückblicke in die Zukunft" im Badischen Landesmuseum Karlsruhe
10/2001-4/2002	Interaktives Exponat "Komponieren wie Bach - Künstlich oder künstlerisch?" in der Sonderausstellung "Computer.Gehirn" im Heinz Nixdorf MuseumsForum in Paderborn
2/2002-2004	Interaktives Exponat "HarmoTheater" in der Ausstellung "Das Netz. Sinn und Sinnlichkeit vernetzter Systeme" des Museums für Kommunikation (Frankfurt, Berlin, Hamburg, Nürnberg)
seit 1/2004	Interaktives Exponat "Komponieren wie Bach - Künstlich oder künstlerisch?" in der Dauerausstellung des Heinz Nixdorf MuseumsForums in Paderborn
seit 5/2004	Interaktives Exponat "Komponieren wie Bach - Künstlich oder künstlerisch?" in der Ausstellung "Elementa 1" im Landesmuseum für Technik und Arbeit in Mannheim