Komposition als Disposition von Datentransformation und Sprachdesign

Folkwang-Texte

Eine Schriftenreihe der Folkwang-Hochschule Essen
herausgegeben von Prof. Dr. Josef Fellsches

Band 14

Markus Lepper

Komposition als Disposition von Datentransformation und Sprachdesign

(Originalausgabe "Verlag Die Blaue Eule", Essen 1999
ISBN 3-89206-948-4
Minimal korrigierter Neudruck 20210707)

Für

Neuhaus, Reith
& Zander.

Vorwort des Herausgebers

Die Folkwang-Texte stellen Ausschnitte aus dem theoretisch-praktischen Arbeitsanteil einer Kunsthochschule dar: der Folkwang-Hochschule für Musik, Theater, Tanz in Essen und Duisburg.

Sie nehmen an der Hochschule erbrachte künstlerische und wissenschaftliche Beiträge des gesamten Fächerspektrums zu Musik, Theater und Tanz sowie zu deren Grundlagen auf. Die wissenschaftliche Reflexion der künstlerisch-praktischen Tätigkeit und ihrer Grundlagen kann einem Bewußtheitsgrad dienen, der nicht nur von didaktischer Bedeutung ist, sondern auch dem Schöperischen zugute kommt.

Nicht alle Folkwang-Texte reflektieren künstlerische Arbeit in begrifflichem Denken hoher Abstraktion, und die Theorieanteile werden unterschiedlich sein. Reflektieren kann auch die Form der Dokumentation einer Projektplanung oder des resümierenden Rückblicks auf ein Projekt haben. Allgemeine Theoriearbeiten stehen auch für theoretische Grundlagen der künstlerisch-praktischen Arbeit gut, ohne daß hierauf in den einzelnen Schriften verwiesen werden muß.

Die Folkwang-Texte sollen nicht nur die Möglichkeit bieten, Erträge der Hochschule zur weiteren Arbeit zur Verfügung zu stellen, sie wünschen sich auch einen größeren Leserkreis, der über Studierende und Fachkollegenschaft hinaus aus Freunden und Interessenten der Folkwang-Hochschule bestehen könnte.

Die vorliebende Arbeit ist Ergebnis eines Forschungsprojektes am Institut für Computermusik und elektronische Medien (ICEM) der Folkwang-Hochschule in Verbindung mit der Firma micro-control, gefördert von Bund und Land NRW. Das Projekt zeichnet sich insbesondere durch seine Interdisziplinarität zwischen Informatik, Kompositionspraxis und Philosophie aus. Damit stellt diese Arbeit ein außergewöhnliches Forschungsdesign an einer Kunsthochschule dar.

Der Autor ermöglicht Tiefenblicke in den Vorgang des Komponierens: Welche neuen Aufschlüsse über die alte Tätigkeit des Komponierens gewährt die Verwendung des Digitalrechners? Und wie weit und bis wohin verändert sich musikalisches Denken durch den Einsatz des Rechners? Die das Buch tragende Philosophie Markus Leppers ist ihrerseits „rechnergestützt“; sie bietet ein Modell, mit dessen Hilfe Erfahrungen im Komponieren präzise erfaßt werden können.

Josef Fellsches

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Zur Übersicht.

Die folgende Schrift geht zurück auf einen gleichnamigen Vortrag, gehalten im Oktober 1994 im Rahmen der „Tape Sessions" genannten öffentlichen Vortragsreihe des Institutes für Computermusik und Elektronische Medien der Folkwang Hochschule Essen.

Gegenstand und Titel des Vortrages sind programmatisch: „Musikalische Komposition als Disposition von Datentransformation und Sprachdesign" bedeutet einen quasi ergonomischen, arbeitswissenschaftlichen Blick auf einen bestimmten Vorgang ästhetischer Produktion.

Diese zunächst auf das im weitesten Sinne Handwerkliche beschränkte Analyse sollte, ex negativo, spätere Schlüsse auf die über das Handwerkliche hinaus gehenden Komponenten des Kunstwerkes, – z.B. die sozialen, historischen, ritualischen, ethischen Implikationen etc. – erleichtern.

Auch, ja gerade das „rein Handwerkliche" ist nämlich im Falle der ästhetischen Produktion nicht außerhalb eines erkenntnistheoretischen Modellzusammenhanges analysierbar. Dieser ist explizit zu axiomatisieren. Die Wahl dieser Axiome ist durchaus problematisch und Gegenstand einer auch inhaltlichen Diskussion.

Zu Beginn werden wir also die begrifflichen Ausgangsannahmen aufstellen, die unserer Modellierung ästhetischen Handelns zugrundeliegen sollen. Da wir dazu etwas weiter ausholen müssen, hoffen wir, daß im folgenden auch Thesen allgemeinerer und kontroverser Natur aufgestellt werden.

Wegen der durchaus inhomogenen Zielgruppe, an die sich der ursprüngliche Vortrag wandte, bittet der Autor um Verständnis, daß im ersten Teil des Textes ein durchaus „populärwissenschaftlicher" Ton angeschlagen wird, während die Teile, die sich konkret auf den Akt des Komponierens beziehen, sich einer professionelleren Sprache bedienen.

Hilfreich wäre es, statt der einleitenden, etwas ermüdenden Explikation eines erkenntnistheoretischen Modelles ein vorhandenes Modell einfach zu zitieren, – leider ist dem Autor noch keines begegnet, welches hinreichend simpel und neutral und dennoch für unsere Überlegungen ausreichend wäre.

1.1 Zum Gegenstand.

Die Grundthesen, die im folgenden aufgestellt werden, besagen, daß ästhetische Produktion *immer* mit den Begriffen „Sprachdesign" und „Datentransformation" zusammenhängt.

„Sprachdesign" ist ein Begriff aus der praktischen Informatik, genauer: dem Übersetzerbau, und bezeichnet den Entwurf einer Programmiersprache, also die Bestimmung ihrer Objekte, deren Semantik und Verknüpfungsmöglichkeiten, und damit implizit einer „Topologie (und „Kostenfunktion") des Ausdrückbaren".

Es soll sich im folgenden zeigen, daß, wann immer ein Digitalrechner im Kontext ästhetischer Produktion eingesetzt wird, die „formbildenden Tendenzen des Materials" darin bestehen, daß schon die Wahl einer vorhandenen oder der Entwurf einer eigenen Sprache eine erste ästhetische Entscheidung ist, daß zwischen Beschaffenheit der angewandten Sprache und Tiefenstruktur des zu schaffenden Werkes sehr enge Zusammenhänge bestehen, – enger wahrscheinlich, als bei den üblichen technologischen Anwendungen.

Der zweite Begriff, „Datentransformation", wird hier so verwendet, daß er einen Übergang zwischen Aggregatzuständen von Daten bedeutet, welcher den eigentlichen, physikalisch meßbaren Informationsgehalt des Datenbestandes *nicht* verändert, wohl aber seine intellektuelle Verfügbarkeit und Vorstellbarkeit, somit die Handhabbarkeit der Information.

Beide Begriffe werden – so hoffen wir – weder als lediglich gemeinsame Oberflächenphänomene, noch gar als Metaphern aus der Informatik in die Kompositionslehre übertragen, – vielmehr behaupten wir Übereinstimmungen auf einer ziemlich tief liegenden Ebene der Grundstrukturen menschlichen Denkens.

Auch die erste Hälfte des Werkes, die ein allgemeines Modell kompositorischen Denkens *ohne* jeden Einsatz des Digitalrechners versucht aufzubauen, welches durchaus *als solches* ernst genommen werden will, ist dennoch – wie der kundige Leser unschwer erkennen wird – durch die Erfahrungen mitgeprägt, die der Verfasser in dem langjährigen Versuch eben der Übertragung des musikalischen Denkens in eine rechnergestützte Arbeitsumgebung machen durfte.

Weitenteils handelt es sich – konkret – um Rückübertragungen von strukturanalytischen Vorüberlegungen zu dem leider nicht zustande gekommenen Projekt „mkSinf", durch welches die noch fehlende Schale „`sh4`" in die AUDIAC-Architektur integriert werden sollte.

Obwohl wir diese Aspekte der Entstehungsgeschichte und die Existenz der „unterirdischen Verbindungskanäle" als meta-sachlich durchaus dazugehörig hier mitgeteilt haben, bitten wir den geneigten Leser, sie erst einmal wieder zu *vergessen* und die ersten Kapitel als das zu lesen, was sie tatsächlich auch sind : der Versuch einer möglichst un-persönlichen und allgemein fruchtbaren Psycho-(= Welt-)Analyse.

1.2 Gliederung.

Inhaltlich wird im folgenden versucht, die konkreten Erfahrungen von fünfzehn Jahren Arbeit auf dem Gebiet der rechnergestützten Komposition einzubetten in ein allgemeineres Modell von ästhetischer Produktion.

In der ersten Hälfte werden, wie erwähnt, die *vor* Einführung des Digitalrechners in den künstlerischen Schaffensprozeß wirksamen Mechanismen analysiert, welche, so behaupten wir, auch *nach* dieser ihre prinzipielle Gültigkeit behalten haben.

Der zweite Teil untersucht dann, wie die analytisch gewonnenen Ergebnisse innerhalb des Rechners zu entsprechender Synthese führen können.

Zu diesem Zweck wird im zweiten Kapitel das o.e. erkenntnistheoretische Minimalmodell vorgestellt, auf das die folgende Diskussion sich begrifflich abstützt.

Das dritte und vierte Kapitel entwickeln ein allgemeines Modell von kompositorischer Tätigkeit, wobei Kapitel drei versucht, die psycho-internen Vorgänge zu systematisieren, und Kapitel vier die physischen Materialisierungen von Musik (soweit für unser Thema nötig) analysiert.

Kapitel fünf benennt die Forderungen an kompositionsstützende Digitalsysteme, die sich als Konsequenzen dieser Analysen ergeben.

Kapitel sechs enthält eine größere Fallstudie. Detailliert dargestellt wird eines der sehr unterschiedlichen Projekte, die im Laufe der Jahre mit dem AUDIAC-System realisiert wurden. Das vorgestellte Projekt wurde aufgeführt im Sommer 1994 im Rahmen der Landesveranstaltung „Spektakel '94".

Kapitel sieben bringt für den technisch Interessierten Details der Implementierung, Kapitel acht enthält sich an das Projekt anschließende weitergehende Überlegungen allgemeiner Art.

Das letzte Kapitel ist der Versuch einer knappen Zusammenfassung unserer Erfahrungen sub specie aeternitatis.

Letztendliches Ziel all unserer Bemühungen sind einerseits grundlegende physiologische/erkenntnistheoretische Aussagen, andererseits konkrete Hinweise für die Planung der zweckmäßigen Weiterentwicklung der digitalen Werkzeuge.

1.3 Zu Form und Sprache.

Durch schwere Erkrankung beim Schreiben sehr stark eingeschränkt, ja lange Zeit ganz gehindert, hat der Verfasser („aus der Not eine Tugend machend") auf eine Reihe älterer Texte und Entwürfe zurückgegriffen und diese grundlegend überarbeitet, wobei vieles diktierenderweise geschehen mußte.

Dennoch meinen wir, daß ein schlüssiges Ganzes herausgekommen ist,– man beachte besonders die hübsche zentralsymmetrische Bogenform.

Auch hat sich die Arbeit krankheitshalber über Jahre hinziehen müssen, so daß der Verfasser zeitweise für die Flüssigkeit des Gesamtgedankens und die Konsistenz der Nomenklatur fürchtete.

Zur Nomenklatur sei besonders bemerkt, daß es sich beim vorliegenden Werk nicht um ein „wissenschaftliches", sondern um ein „literarisches" handelt[1].

In jenen nämlich ist es bekanntlich zwingend erforderlich, jeden Begriff, so er nicht aus der herrschenden Lehre importiert wird, vor jedem Gebrauch möglichst so umfasssend wie im folgenden verwendet zu „definieren", – laut (Kant, 1781) (Methodenlehre, Disziplin der reinen Vernunft im dogmatischen Gebrauche) besser : zu „explizieren".

Jeder derartige Begriff wird dann mit *genau einem* Wort verbunden, an welchem zum alleinigen Gebrauch festgehalten wird[2].

Der hier verwendete Stil ist hingegen ein quasi lyrischer, als die Substanz der verwendeten Begriffe durch manngifaltige Wortwahl, quasi-Synonyme, bis hin sogar zu Metaphern aus der täglichen Erfahrung versucht wird zu übermitteln. Diese Strategie, begrifflichen Gehalt auch über das (sprachliche) Gefühl bei der Anschauung (oder die Anschauung des Gefühles) zu kodieren, bedeutet aber keinesfalls, daß die *Begriffe selbst* unscharf sein müssen.

Der Leser möge bitte überprüfen, wie scharf unsere Begriffe jenseits ihrer (teilweise bunten) Versprachlichung sind.

1.4 Homines, sine quibus non.

Dieses Werk ist gewidmet Dr. Helmut Zander, Prof. Dirk Reith und Thomas Neuhaus, in Anerkennung ihrer großen Verdienste um sein Zustandekommen. Ohne ihr Interesse, ihre Mitarbeit und Hilfe gäbe es dieses und vieles andere nicht. Die etwas saloppe Kurzform auf vorstehender Widmungseite wurde gewählt, weil (1) ihr Rhythmus so schneidig klingt – gerade angemessen einem erfolgreichen engineering-team –, und (2) weil diese Erläuterung die Gelegenheit bietet, dortige alphabetische Reihenfolge hier durch die projektchronologische zu ergänzen.

Der Verfasser fühlt sich weiterhin all denen verbunden, ohne deren Mitarbeit, Interesse, Ermunterung, Unterstützung, Diskusssionsbereitschaft oder ohne deren eigenes Schaffen dieses Werk und die anderen Ergebnisse seiner Arbeit in den zurückliegenden Jahren nicht zustande gekommen wären.

[1] Der aufmerksame Leser wird schnell merken, an welchen Stellen z.B. Horkheimer/Adorno o.a. in einem wissenschaftlichen Werk zitiert und bibliographiert werden müssten, – hier werden sie schlicht vorausgesetzt.

Darüber hinaus ist ziemlich unwissenschaftlich die seit dem TEX-book erlaubte Praxis, just in der Gestaltung des *Index* seinem (Galgen-)Humor freien Lauf zu lassen.

[2] Ein solcherart verwendetes Wort ist eigentlich *immer* ein „Kunstwort", – sein Bezug zur tagtäglichen Sprache genau genommen ein rein *phonetischer* und vorhandenes „intuitives Vorverständnis" unter Umständen sogar störend !

In diesem Sinne seien erwähnt:

Herr Roland Borrmann, Wuppertal, Herr Dr. Heinz-Josef Florian[3], Gladbeck, Herr Xavier Garavaglia, Essen, Herr Dipl.-Inf.Wolfgang Grieskamp, Berlin, Herr Thomas Hansen, Dortmund, Herr Friedhelm Hartmann, Tel Aviv, Herr Dipl.-Ing.Gerhard Kümmel, Essen, Herr Dipl.-Ing.Roland Masslich, Essen, Herr Alex Rosen, Moskau, Herr Arie van Schuetterhoef[4], Amsterdam, und Herr Mathias Wittekopf, Bochum.

[3] „. . . sei bedankt, er hat es ihm abverlangt".

[4] . . . dito!

Kapitel 2

Kleines Erkenntnistheoretisches Ausgangsmodell.

2.1 Das „Kunstwerk" und das „Ding an sich".

Wir werden nun als erstes ein kleines, rudimentäres erkenntnistheoretisches Modell definieren.

Dieses soll vorerst nur so weit ausgebaut werden, wie es nötig ist, damit es als Grundlage für die folgende Diskussion dienen kann.

Der erste Schritt ist schnell vollzogen, da aus den unterschiedlichsten nachkantischen philosophischen Systemen bekannt und vielfach praktiziert :

Wir postulieren die Existenz eines jeden „Kunstwerkes" als „empirisch realistisch und transzendental idealistisch".

Mit diesen Eigenschaften ähnelt es dann dem, was bei KANT das sog. „Ding an sich" ist. Deshalb können wir in erster Näherung qua Analogieschluß einige der Kantschen Forschungsergebnisse unmittelbar auf dieses so geforderte „Kunstwerk" übertragen.

Im Zusammenhange unseres Minimalsystem bedeuten diese Eigenschaften lediglich zweierlei :

– Zum einen die außerzeitliche (quasi immerwährende) *Identität* eines jeden „Kunstwerkes" mit sich selbst, – zum anderen seine wesentliche (bei KANT: „transzendentale"), unüberbrückbare *Geschiedenheit* von unserer Vorstellung und Wahrnehmung.

2.2 Selbstidentität als Postulat.

Die erste Forderung besagt, daß es sinnvoll sei, von „Beethovens Fünfter Sinfonie" überhaupt zu reden. Wenn in zwei verschiedenen Sätzen jeweils die Wortfolge „Beethovens Fünfte Sinfonie" erscheint, dann meinen beide Sätze, dasselbe „Ding" zu meinen.

Wohlgemerkt, dies ist eine Forderung, eine Arbeitsdefinition, – keine Strukturaussage. Wir *postulieren* die empirisch realistische und transzendental idealistische Existenz eines ominösen Etwas, welches wir, ohne es genauer beschreiben zu wollen, lediglich als gemeinsamen Bezugspunkt für alle möglichen Referenzen betrachten.

Wir behaupten schlicht, daß ein „Kunstwerk" im selben Sinne dinglich existiert wie ein Bagger oder eine Brücke. Die Existenz jedoch von Bagger und Brücke in der Dinglichkeit ist lediglich eine intellektuelle *Hypothese*, die Dinge selbst sind uns nicht erfahrbar.

Diese „Dingliche Existenz" von z.B. Beethovens Fünfter Sinfonie zu behaupten scheint angemessen und einfach, selbstverständlich und unproblematisch. Dennoch ist diese Behauptung stets als Arbeitshypothese bewußt zu halten, die Konsequenzen ihrer Hypothesen-Natur sind teilweise dramatisch.

Man weiß z.B., daß in vielen Fällen dieses gleichsam „personalisierende" Vorgehen nicht unbedenklich ist. Immer dann, z.B., wenn von „Dingen" die Rede ist, die sich als Organisationsformen von Verhaltensweisen von Individuen konstituieren, — Vereine, Parteien, Schichten, Traditionen, Rituale –, muß streng darauf geachtet werden, die Ebene der abstrakten Betrachtung zugunsten der Analyse indvidueller Verhaltensweisen rechtzeitig zu verlassen, um sich nicht der *anti-ethischen Abstraktion* schuldig zu machen („Die Juden sind unser Unglück", „Der Russe kommt" etc.).

Schwerer noch wiegt, daß wir uns mit der Behauptung der immerwährenden Selbstidentität des Dinges „Kunstwerk", über das wir reden, in einen wesentlichen Widerspruch setzen zu unserer *Erfahrung im Umgang* mit jenem Ding, zu unserem Bild von jenem Ding.

Die Vorstellung von einem Kunstwerk und seine Wahrnehmung sind nämlich Sachen, für die Konstanz und Identität das letzte wären, was man ihnen zuspräche.

Vielmehr weiß ja jeder Rezipient, daß „Beethovens Fünfte Symphonie" halt nicht bei jedem Hören, Lesen oder Erinnern dieselbe ist. Die Vorstellung ist jedesmal eine andere; das Kunstwerk als *inneres Modell* ist im stetigen Wandel: das virginale Erleben, das allererste Hören ist offensichtlich unwiderruflich vorbei, die virginale Empfindung ist nur noch zufällig, sporadisch und meist spontan wieder-erlebbar. Aber auch das letzte Hören ist nicht wiederholbar; – ja, während des Hörens gar verändert sich mit der vergehenden Zeit in jedem Moment meine Vorstellung vom Werk.

Der gemeinsame Bezugspunkt aller Vorstellungen jedoch ist das imaginäre, von uns lediglich behauptete und nie nachweisbare „Kunstwerk an sich".

Die gegenteilige Axiomatik, – es gab nie ein Kunstwerk, und ein Ding namens „Beethovens Fünfte Sinfonie" existiert mitnichten, – wäre also ebenso legitim.

Denn selbst Bagger und Brücke, deren Existenzgrad der des Kunstwerkes oben gleichgesetzt wurde, und deren Dinghaftigkeit dem naiven Blick ohne Zweifel gegeben scheint, existieren als Objekte tatsächlich ausschließlich in der intra-psychischen Vorstellungswelt[1].

[1]Die Nomenklatur unseres Minimalmodelles macht keinen unterschiedlichen Gebrauch von den Wörtern „sein" und „existieren".

Die Mintarder Autobahnbrücke z.B. *ist* für ihre Betreiber ein wirtschaftliches und damit dia-chrones Arbeitsvorhaben, - für den fernreisenden Autofahrer einerseits nur ein Zentimeter einer farbigen Linie auf der Straßenkarte, welcher einen Teilabschnitt einer abzuleistenden Fahrtzeit repräsentiert, andererseits aber vielleicht Gefahrenquelle wegen plötzlich einsetzender Seitenwinde und Ablenkung durch die überwältigend schöne Aussicht, - für die dort arbeitenden Stricher *ist* sie Teil ihrer Heimat, - für das in ihren Röhren eingekerkerte Entführungsopfer *ist* sie der Ort äußerster Qual und Erniedrigung, - für den Wanderer in den Ruhrwiesen *ist* sie eine gewaltige Skulptur, welche das Erleben der zeitlosen Schönheit parabolischer Graphen ermöglicht, etc.

Nur die Praktikabilität der weiteren Diskussion erlaubt also überhaupt das Postulat von der selbstidentischen Existenz eines Kunstwerkes. Ein Kunstwerk mit Hilfe der analytischen Vernunft zu betrachten, kann sinnvoll sein, – oder auch nicht. Wenn wir uns aber dazu entschließen, dann müssen wir auch unsere Begrifflichkeit so aufbauen, daß sie der Struktur der Vernunft kompatibel ist.

Es sei erlaubt, zu wiederholen: diese Eigenschaften werden als *Arbeitsgrundlage axiomatisch festgelegt*, also willkürlich von uns gefordert und sind keine Strukturaussagen. Auf ihrer Grundlage jedoch werden wir durchaus diesem nebulösen Ding „Kunstwerk" näher rücken und durchaus zu genaueren möglichen Aussagen zu seinem Wesen gelangen. Der dazu erste Schritt besteht darin, einige der Eigenschaften des Kantschen „Dinges an sich" auf unseren Kunstwerkbegriff zu übertragen. Dies wird zeigen, daß unsere scheinbar einfache Forderung zu interessanten, quasi ontologischen, aber auch phänomenologischen Konsequenzen führt.

In der Tat wäre die letzte Konsequenz der Übertragung der Kantschen Erkenntnisse die, daß es tatsächlich keine Kunstwerke gibt. Da es im Bereich der Dinglichkeit, jenseits unserer Erfahrung, keine Erfahrung, also keine Kriterien oder Scheidemale gibt, gibt es auch nur *ein* Ding an sich, es gibt nur *ein* Ding, und es gibt nur *ein* Kunstwerk, welches alle Kunstwerke, so sie Modelle sind, hervorbringt.

2.3 Transzendentale Geschiedenheit.

Die zweite Eigenschaft des „Dinges an sich" besagt, daß wir, (abgesehen von jener ersten Eigenschaft der Selbstidentität, die wir gerade ja eben als Behauptung aufstellten), *gar nichts weiteres* von diesem „Ding" wissen *können*.

Warum dies?

Interessanterweise kann die Vernunft selbst erkennen, daß und wo sie nicht erkennen kann.

Die folgende Überlegung, die dafür den formalen Rahmen liefert, wurde in Jahrhunderten menschlichen Denkens entwickelt, bis sie dann bei Immanuel Kant ihre wohl zur Zeit klarste Formulierung fand.

Auf unsere Bedürfnisse reduziert erscheint sie unmittelbar einleuchtend, ja, als eine Binsenweisheit.

Ihre geistesgeschichtlichen Konsequenzen waren jedoch mitunter katastrophal.

Diese Überlegung verläuft ungefähr wie folgt :

Bei jeder Wahrnehmung eines (als existent postulierten) Objektes (also auch des von uns oben postulierten, quasi dinglich existierenden „Kunstwerkes") ist das Ergebnis dieser Wahrnehmung ein gemeinsames Hervorbringnis von diesem Objekt einerseits und den Eigenschaften des wahrnehmenden Subjektes andererseits.

Die physiologische Beschaffenheit der Sinnesorgane und des Nervensystems, die Denkformen, das Vorwissen, das Kurzeitgedächtnis und dessen momentaner Inhalt, die hormonelle Situation, der Zustand des vegetativen Nervensystems, sie alle fließen natürlich in das ein, was ich der Einfachheit halber schlicht als „meine Wahrnehmung von diesem Schwan auf diesem Teich zu dieser Stunde" gerne bezeichnen möchte.

Jede Wahrnehmung ist vom Wahrgenommenen *und* Wahrnehmenden determiniert. Jede Wahrnehmung aber ist – als existentielles Erlebnis des Wahrnehmenden – ein unteilbarer, momentaner Vorgang. Teilbar in Zeit und Parameter ist nur das im Nachhinein gefertigte *Modell* dieses Vorgangs, – teilbar ist nicht der Augenblick, nicht die Ergriffenheit und nicht der Erkenntnisschritt, so wie wir ihn dann und nur dann wahrhaft erleben.

So kann man weder qualitativ oder gar quantitativ auseinanderdividieren, was Wahrnehmender und was Wahrgenommenes zum Vorgang des Wahrnehmens beitragen. Da ein gemeinsames Modell fehlt, sind keine Kreuztests möglich. Wenn ein Akkord auf eine Testperson „grün" wirkt, dann weiß kein anderer Mensch, wie ihrerseits die Empfindung „grün" auf diese Testperson wirkt.

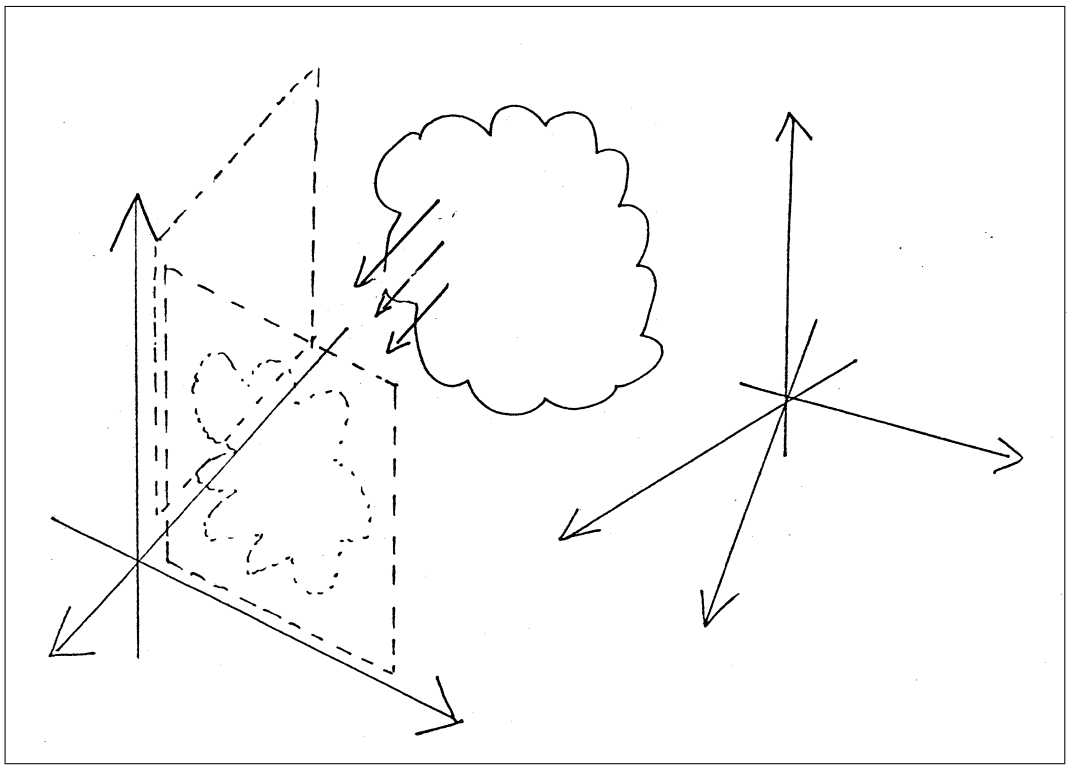


Abbildung 2.1: Die modellbestimmende Funktion der Wirklichkeitsachsen.

Vielmehr sind Wahrgenommenes und Wahrnehmender, Subjekt und Objekt, wesentlich getrennt, begrifflich inkommensurabel, – ja, der *Akt* der Wahrnehmung selbst ist die einzige Kommensur, der Akt der Wahrnehmung läßt Mensch und Ding als Subjekt und Objekt erst entstehen.

Zusammentreffend im Akt der Wahrnehmung generieren Wahrgenommenes und Wahrnehmender etwas Drittes, von beiden verschiedenes, qualitativ neues und einer ganz anderen, inkommensurablen Welt angehöriges, die Empfindung.

Wenn die Substanz aller unserer Vorstellungen und Urteile jedoch die Erfahrung ist, dann folgt daraus, daß alle unsere Vorstellungen und Urteile nur in der Funktion von *Modellen* ihre Gültigkeit haben.

„Transzendental idealistisch und empirisch realistisch" bedeutet, grob zusammengefaßt also, daß die „Welt" als „Ding" zwar unabhängig von uns als existent angenommen wird, jedoch *als solche* unserer Wahrnehmung nie und nimmer zugänglich ist.

Jede „Wirk-Lichkeit" ist vielmehr immer ein *Modell*.

2.4 Exkurs gegen den Behaviorismus.

Obige Formulierung, in der die das Erleben determinierenden inneren Parameter aufgezählt wurden, ist übrigens in so weit irreführend, als sie einen Trugschluß induzieren könnte, welcher konstitutiv ist für alle Schulen der positivistischen Anthropologie.

Die oben aufgezählten angeblichen Bestimmungsgrößen (wie „physiologische Beschaffenheit der Sinnesorgane", „Zustand des vegetativen Nervensystems" etc.) sind als Begriffe selbstverständlich *ebenfalls* nur *innerhalb bestimmter Modelle* relevant.

Den „hormonellen Zustand" gibt es genau so lange, wie ich mich in einem wohldefinierten Modell aus der klinischen Endokrinologie bewege. Außerhalb des Modells hat der Begriff keinerlei Bezüge.

Der Denkfehler von Gehirnforschern und Behavioristen besteht darin zu meinen, durch immer weitergehende Erforschung des menschlichen Denkens – betrachtet aus der Sicht ihrer Modelle – dem Erkenntnis des Wesens des Denkens näher zu kommen[2].

In Wahrheit aber ersetzten sie lediglich ein Modell durch ein anderes, wenn auch verfeinertes.

Dem Wesen des Erlebnisses als einem unteilbaren, unmittelbaren und konstitutiven Seelenvorgang ist damit nicht beizukommen.

Selbstverständlich ist „mein Wille" bis zum Äußersten unfrei; unfrei bis dahin, wo die Elementarteilchen vielleicht frei sind. Jedoch (erstes Argument zur Erlösungsfähigkeit des Menschen) ist keine vollständige Modellierung meiner selbst denkbar, die für mich relevant, d.h. handlungsnützlich wäre.

[2]Man lese die entzückend geschriebenen „Rezensionen" der Werke des Dr. Julian Freke in D. L. Sayers „Whose Body?" (Sayers, 1930).

2.5 Exkurs über Ursache und Wirkung.

Die Grundzüge dieser Überlegung sind bekannt und in vielen philosophischen Axiomatiken in Gebrauch. Frühe, unfertige Formulierungen finden wir im „Schleier der Maja", welcher von SCHOPENHAUER aufgegriffen wurde, im „Höhlengleichnis" etc., – die bis dato klarste Ausprägung in KANTs systematischer Vivisektion der „reinen Vernunft".

Diese Überlegungen klären z.B. das langezeit einzig als göttliches Schöpfungswunder erklärbare, ansonsten schier unbegreifliche Phänomen der sog. „Praestabilierten Harmonie":

Wie kann es sein, daß die *Welt*, die uns als ein Äußeres entgegentritt, in ihren Eigenschaften dennoch der Mathematik isomorph ist, welche doch eine Schöpfung einzig des *Geistes* ist?

Die Antwort ist nun klar: Die Welt, die uns entgegentritt, ist ebenfalls Schöpfung unseres Geistes und somit allen anderen Hervorbringnissen des Geistes durchaus kompatibel.

Neben dieser wohltätigen Wirkung, welche die menschliche Erkenntnis von schweren Zweifeln und nagenden Fragen entlastete, zeitigte dieser Weltbegriff andererseits jedoch grausame Konsequenzen, indem er den Menschen zwang, vom Aberglauben der Universalität bestimmter Vorstellungen Abschied zu nehmen, wie z.B. der von „Ursache und Wirkung".

Das Begriffspaar „Ursache und Wirkung" kann nämlich nur dann zur Anwendung gelangen, wenn ich als Urteilender die zugrundeliegende Modellbildung irgendwo *willkürlich abbreche*.

Wenn in einer Gymnasialklasse der Kaiserzeit ein Bleistift klirrend zu Boden fällt, dann reicht zum Zwecke der disziplinären Weiterbearbeitung des Vorfalles die Aussage :„ *Weil* Schüler Müller mit dem Zeigefinger vor den Bleistift schnippste, *deshalb* rollte dieser vom Pulte".

Geschieht dies im Physikunterricht, so müßte der Lehrer wohl einige Darlegungen zu Masse, Trägheit und Impuls folgen lassen.

Noch genauer müßte man, um zu beantworten, *warum* der Bleistift fiel, die materialkundlichen Eigenschaften von Fingernagel und Bleistiftlack heranziehen, Elastizität und Plastizität. Zu deren Erklärung müßte man sich wiederum auf interatomare Kräfte beziehen, sodann auf kernphysikalische Modelle, auf Elementarteilchentheorie, Elektromagnetismus etc.

Man sieht, die Frage nach dem *Warum?* induziert in der menschlichen Vernunft nicht nur den temporalen regressus ad infinitum, welcher zu der causa prima non causata führen soll, – nein, ein instantieller regressus ad infinitum ist durch jede einfach Frage der Gestalt *Warum?* unmittelbar gegeben.

Lediglich indem ich mein Modell nach unten abschneide, d.h. seine weitere Begründbarkeit axiomatisch behaupte, und seine Wahrheit ebenfalls, kann ich dann, im Rahmen dieses Modells, die Frage nach dem *Warum?* auf eine von vielen möglichen Arten beantworten.

Interessanterweise sind die Modelle skalar in *beide* quantitativen Richtungen

begrenzt : das Atomare Modell kann vielleicht erklären, warum sich ein Masseteil in Bewegung setzt, weiß aber nicht, daß diese Bewegung einer Zykloide folgen wird. Denn von dem millionenfach größeren Bleistift und seiner Zylindergestalt kann das Atomare Modell, schlicht wegen der zu umfangreichen Datenmenge, einfach keinen Begriff haben.

Gerade die Tatsache der *nach oben* begrenzten Modellbildung wird im folgenden, bei der Betrachtung der kompositorischen Praxis eine zentrale Rolle spielen!

Um in irgendeinem z.B. „naturwissenschaftlichem" Gebiet überhaupt Erkenntnisse gewinnen zu können, also Modelle aufstellen, verifizieren, vergleichen, kombinieren und vereinheitlichen etc., muß ich meine Modellbildung zumindest nach unten hin (willkürlich oder qua historischer Übereinkunft) abschneiden:

Atomphysik, Chemie, Biochemie, Physiologie, Zoologie und Ökologie bilden z.B. eine typische *Wissenschaftskette*: Die (vereinfachten) Ergebnisse des jeweils vorangehenden Erkenntnisbereiches werden als Axiome und nicht-hinterfragbare Wahrheiten in den folgenden Bereich übernommen und ermöglichen dort erst die Begriffsbildung und ein quasi autarkes symbolisches Operieren.

Die Aufgabe des Tierforschers, die Suche einer Population nach Nahrung zu erforschen, wäre schlechterdings unmöglich, müßte er zur Begründung seiner Ergebnisse bis hinunter zur Bindungsenergie des Kohlenstoffes ausholen.

Jede betrachtbare Wirk-Lichkeit ist also Projektion der (sinnvollerweise zu behauptenden) Ding-Lichkeit. Das erscheinende Bild jedoch ist mitbestimmt von den von den Projektionsflächen. Diese ergeben sich aus der jeweils autarken, höchstens qua Konvention vorgeschriebenen Wahl der *Wirklichkeitsachsen* (siehe Bild 2.1).

Die Basisvektoren der Wirklichkeit sind wählbar.

2.6 Das Utilitarische Wahrheitsindiz.

Wenn jedoch die Modellbildung prinzipiell auf beliebigen Prämissen aufbauen kann, dann ist der einzig mögliche Begriff von „Wahrheit" ein relativer.

Jedes Modell ist dann „wahr", d.h. dem Ding an sich adäquat, wenn es seiner Funktion, Anwendung und Benutzung adäquat ist.

Der Ingenieur, der eine Brücke berechnet, bedient sich des aus der jahrhundertelang gewonnenen Erfahrung abstrahierten Modelles, konstruiert innerhalb des Modells und überträgt das Resultat seines Operierens zurück in das „Modell Materie".

Wenn nun die nach den modellinternen Überlegungen getroffenen Entscheidungen ihrem Effekte nach in der Realität mit den Vorhersagen des Modelles (hinreichend genau) übereinstimmen, so gilt das Modell als „korrekt".

Die Grenzen dieses Korrektheitsbegriffes auf rein technologischer Ebene sind offensichtlich und die Geschichte zeigt ja auch eine regelmäßige Folge von Modell-Revisionen.

Grundlegend philosophisch ist zu urteilen, daß Erfahrungswerte (als dem Wesen nach unwiederholbare Einzelerlebnisse) immer transzendental geschieden sind von Vorhersagen, welche stets nur als statistischer Durchschnitt Gültigkeit haben: Das

„thermodynamische Wunder", daß ein Stahlträger trotz 10-facher Sicherheit doch einmal reißt, ist (schon rein begrifflich) niemals auszuschließen.

Einziger Hinweis auf die Adäquatheit einer Modellbildung kann *vor der Vernunft* nur deren Zweckmäßigkeit in der praktischen Anwendung sein. Dies nennen wir das *Utilitarische Wahrheitsindiz.*

2.7 Exkurs über Schuld und Sühne.

Selbstverständlich ist vorliegende Schrift in Inhalt und Methode ein rationaler Diskurs. Aber man kann nicht Erkenntnis- und Erlebnismöglichkeiten des Menschen, nicht Seelenzustände und Motivationen des Kunstschaffenden diskutieren, ohne an einigen Stellen einen kleinen Ausschnitt von herkömmlicherweise eher dem „Religiösen" zugerechneten Fragen zu streifen.

Außerdem kann es, um über ästhetische Positionen letztlich *wertend* urteilen zu können, notwendig sein, die Art und Richtung der Nachbarschaft zu ethischen Fragestellungen zumindest erwähnt zu haben.

Einschneidend nämlich, mit letztlich katastrophalen Folgen und fünfundfünfzig Millionen Toten, war der Sieg jenes erkenntnistheoretischen Modelles und die Etablierung des Utilitarischen Wahrheitsindizes auf dem Gebiet der Ethik:

Der Mensch, welcher erkennen mußte, daß die vom urzeitlichen Beginn der Vernunft an scheinbar gültigen Grundformalismen, wie z.B. „Ursache und Wirkung", zwar nicht obsolet, aber durchaus redefinierbar sind, – der Mensch, verunsichert dadurch, daß durchaus verschiedene Modelle, verschiedene Wirklichkeiten der einen, selben, an sich unerfahrbaren Dinglichkeit durchaus adäquat sein können, – der Mensch, welcher der gerade frisch errungenen Sieg der Vernunft über Aberglaube, der „praestabilisierten Harmonie" zwischen Welt, Vernunft und Güte verlustig wurde, – der Mensch reagierte panisch.

Als tiefste, also allen modernen Verunsicherungen zugrundeliegende Frage, welche ihrerseits von der Vernunft nicht beantwortbar, sondern nur vorbereitend bearbeitbar ist, kann angesehen werden die Frage:

Wie kann der individuelle Mensch, von dem ein behavioristisches Automatenbild ja ein adäquates Modell sein *kann*, für irgendeine seiner Taten jemals verantwortlich sein?

Die Anwort ergibt sich daraus, daß, wenn dies schon wie oben gezeigt für die jeweils zu wählenden Grundlagen rein rationaler Erkenntnis gilt, erst recht die Grundlagen der Ethik, also der Bewertung eigenen und fremden Handelns, in einem *axiomatischem Akt* der Selbstdefinition jeweils neu *gesetzt* werden.

Der Herrschaftsbereich der Vernunft ist beschränkt. Die Postulierung ethischer Normen erfolgt nämlich *nicht* auf der Ebene der Vernunft, sondern auf der des Sozialen. Ich kann mich allemal entscheiden, Buddhist oder Faschist zu werden. Ob ich dies dann auch durchhalten werde, nur *darauf* kann evtl. ein behavioristisches Modell Antwort geben.

Die Vernunft, gerade mal vor ein paar lächerlichen Jahrhunderten an die scheinbar vollständige Macht gelangt, sah sich ihrer Vorherrschaft beraubt.

Gerade nämlich waren die vernünftigen Modelle (wie Ursache und Wirkung, Verbrechen und Gesellschaft, Gerechtigkeit und Gesundheit) an die Stelle der alten jahwistischen angetreten zur Begründung von sittlichen Werturteilen, gerade begannen sich positivistische oder naturrechtliche Begründungen sittlichen Handelns durchzusetzen, da wurde mit einmal die Autarkie der Vernunft selbst aufgehoben.

Die Katastrophe der theoretischen Ethik war perfekt; sie blieb nicht ohne Folgen. Was als akademische Arbeit der idealistischen Autoren begann, war Sprengstoff zwischen Buchdeckeln, welcher zuletzt mit zum Untergang der Weltordnung führte.

Da das Kriterium für die Gültigkeit von Modellen in erster Linie ihre Utilität ist, folgte zunächst natürlicherweise die verschiedenen Utilitarismen.

Ihnen folgten die Nihilismen als Gegenbewegung und Überspitzung zugleich; sie leisteten auch ihren Beitrag zu zwei Weltkriegen.

Erst wir heute können die von KANT in der Kritik der reinen Vernunft angebotene Lösung des Konfliktes der vernunftgemäßen Begründung des Ethischen als solche überhaupt erst zur Kenntnis nehmen.

Die Vorkriegsgenerationen sahen diese gar nicht.

In Wahrheit nämlich ist und war die soziale Existenz des Menschen von der Krise der Vernunft und der Welt-Anschauung in keiner Weise betroffen!

Die Vernunft nämlich muß, das wissen wir heute, nur als *eine* Komponente der menschlichen Existenz angesehen werden, als eine unter a priori gleichberechtigten vielen.

Sowohl die Triebstruktur und die verborgenenen, schlafenden Programme unseres Reptilienhirns, als auch die unterbewußt, „prae-sapiential" ablaufenden symbolischen Operationen machen uns aus, sind Teil unseres Schicksals, stets problematischer Teil unseres Selbstbegriffes und Gegenstand von Selbsterforschung und -kontrolle.

Die Vernunft, als evolutorisch hervorgebrachtes Werkzeug, hat ihren Herrschaftsbereich; dieser ist groß, aber begrenzt. Außerhalb dieses hat sie zu *dienen*.

Die Lösung des Dilemmas ist also die : Wir *brauchen* keine vernunftgemäße Ethikbegründung, wir brauchen keinen rationalen Gottesbeweis. Diese historischen Sackgassen sind Folge der falschen Einschätzung von der Universalität der Anwendbarkeit unserer Vernunft, sind Gebrauch eines Werkzeugs am falschen Objekt.

Die Ethik, die Beziehungen der Menschen untereinander und zu sich selbst, werden zwar mit Hilfe der Vernunft modelliert, nicht aber von ihr determiniert. Wenn sie sich ehrlich umblickt, dann erkennt die Vernunft außerhalb ihrer die autarke Existenz von *Glaube*, *Liebe* und *Hoffnung*.

Heute erst könnten wir aus der Relativität der rationalen Modellbildung die sozialen Konsequenzen ziehen, könnten einen *Pluralismus* praktizieren, der nicht Indifferenz, sondern Toleranz bedeutet, nicht Beliebigkeit und Nachsicht gegen sich selbst, sondern Anerkenntnis des anderen.

Wie ein ensprechender Gottesbegriff vor unserer Vernunft und unserer Unvernunft nur aussehen *kann*, – zur Erforschung dieser (nicht unwichtigen) Frage kann die Vernunft dann durchaus dienen. Ob dieser Gott aber postuliert wird oder nicht, das entscheidet nicht die Vernunft, – das nimmt sie entgegen als ein Eingabedatum.

2.8 Die dem Kunstwerk zugrundeliegende Wirk-Lichkeit.

Eine Wirk-Lichkeit ist also immer ein Modell.

Die Basisvektoren jeder Wirklichkeit sind somit *wählbar*; die Projektionsflächen im Höhlengleichnis sind die von zwei Basisvektoren aufgespannten Ebenen.

Bei allen Versuchen von „Aussagen über die Dinge" ist der Autor also rechenschaftspflichtig über die Wahl seiner Basisvektoren[3]; nur innerhalb dieses Modelles können seine Aussagen diskutiert werden, – oder aber die Wahl des Modelles kann als inadäquat insgesamt abgelehnt werden.

Demhingegen gilt in der Kunst, daß gerade die Wahl des Modelles, die Definition der Achsen der Wirklichkeit, schon *Teil des ästhetischen Produktionsprozesses* sind; die Setzung des Modelles ist eine der ersten sog. „künstlerischen Entscheidungen", die Setzung des Modelles ist ein ästhetischer *Akt* und wie alle künstlerischen Entscheidungen (mehr oder weniger scheinbar) *autonom*.

2.9 Anwendung auf das bildnerische Kunstwerk.

Dieses Bewußtsein von der gleichzeitigen Gültigkeit von Modellen verschiedener Ebenen ist ein konstitutives Prinzip in einer Vielzahl von Werken und Stilrichtungen der modernen bildenden Kunst, – wobei der Ausdruck „modern" in Zeiten der Postmoderne und dank dieser statt der Verwaschenheit älteren Trivialverständnisses, welches alles jenseits des röhrenden Hirsches als „modern" bezeichnete, nun wieder positiv steht für alles, was nicht postmodern ist, also für Konzept, Emanzipation und Fortschritt.

Die in vielen Stilrichtungen moderner bildender Kunst anzutreffende Komponente von intellektual/kognitivem *Lustgewinn* beruht auf dem immer wieder neu einsetzenden überwältigenden Schwindelgefühl bei der unmittelbar physiologisch relevanten Demonstration solchen Schwankens zwischen verschiedenen Modellen, auf der *Körperlichkeit* des Erfahrens über die Struktur unseres Denkens, welche sonst, als lediglich theoretische, begriffliche Einschätzung, uns schwerlich so überwältigen würde, – trotz ihrer intellektualen Evidenz und existenziellen Bedeutung.

Robert Morris z.B. legte in einer ziemlich hinterhältigen Installationen[4] drei solide, vertrauenerweckende Metermaße, versehen mit Zentimeter- und Millimetereinteilung, nebeneinander in eine Vitrine. Ihr einziger Nachteil : Die drei „Meter" sind nicht gleich lang !

Der Anblick dieser Installation entzieht unserem Geiste schlagartig das trügerische Gefühl von Vertrautheit. Die gesamte Grundlage unserer Industriekultur wird in Frage gestellt, unserer vitale Abhängigkeit von fremdbestimmter Norm wird schwindelerregend plötzlich schmerzhaft bewußt, und die Tragweite unseres Vetrauens auf die Eichgenauigkeit Unbekannter.

[3] „Gott ist tot", „der Mensch ist elend", „... ergo sum" etc.

[4] Nach Angabe von Herrn David Gründer, Köln, zurückgehend auf die Installation „Trois Stopage Étalong" von Marcel Duchamp.

Der zeitgenössische Homo Sapiens kann nämlich nur dann überleben, kann nur noch dann die ihn schützende Hütte, das lebensnotwendige Dach bauen, wenn die „M-10"-Schraube der Firma Ixx auch zu der „M-10"-Mutter der Firma Ypsilon paßt.

Mit Binsen und Schindeln können wir uns nicht mehr schützen, da dafür Handwerk und Logistik wegbrachen; – in Elendshütten aus Aldi-Tüten wäre unser Immunsystem wohl bald überfordert. Die in DIN und ISO kristallisierte und über Jahrtausende organisch gewachsene Fertigkeit der Materialbearbeitung in zunehmender Präzision ist unabdingbare Voraussetzung, daß in unseren Großstädten lebensspendendes Trinkwasser wie selbstverständlich aus dem Hahn läuft.

Ein zweites treffendes Beispiel für den Wirkungsbereich „Aufklärung durch Verunsicherung" der zeitgenössischen Bildenden Kunst finden wir in Barnet Newmans Gemälde „Wer hat Angst vor Rot, Blau und Gelb".

Die schlichtest mögliche Begrifflichkeit, eine der elementarsten, quasi atomaren Vorstellungen, die Vorstellung von der „Geraden Strecke", welche zwei Flächen trennt, tritt in ihrer Einfachheit in unmittelbaren Gegensatz zu dem komplexen und brillianten Feuerwerk in Netzhaut, Ganglien und Hirnrinde, welches erblüht und verströmt, wenn wir dieses „Einfachste Ding", die „Gerade Linie", versuchen zu *sehen*. Dieses so einfache Ding unserer Vorstellung, – die „Gerade Linie"–, welches wir so genau kennen, welches uns von Kindheit an vertraut ist, und welches uns ein Bestandteil unserer *optischen* Vorstellung zu sein scheint, ist nämlich mitnichten „sichtbar".

Der vermeintliche Versuch, es betrachten zu wollen, führt zu im doppelten Sinne des Wortes „sensationellen" Erfahrungen und eröffnet uns eine Expedition durch die Schichtigkeit der Strukturen unseres inneren Erlebens.

Kapitel 3

Mögliches Modell des Kompositionsprozesses.

Konkretisieren wir nun das aufgestellte erkenntnistheoretische Modell, indem wir es auf das musikalische Kunstwerk beziehen. Dabei betrachten wir das Kunstwerk wiederum ergonomisch und diachron, anhand eines möglichen Modells seiner inneren Entstehungsgeschichte.

Zweck dieser kleinen Untersuchung ist es nicht nur, allgemeine Erkenntnisse zu gewinnen, sondern vordringlich, die Rahmenbedingungen für den *sinnvollerweise wünschenswerten* Einsatz des Digitalrechners in der musikalischen Komposition herauszuarbeiten.

Die gröbste Darstellung der Lebensgeschichte eines Werkes ist zweifellos eine dreischrittige: Aus der (intra-psychischen) Ausgangsvorstellung des Autors werden physische Objekte (Notentext, Luftschwingungen bei Aufführung und Plattenwiedergabe, etc.), diese ihrerseits induzieren die soziokulturelle Wirkungsgeschichte eines Werkes als Hülle für die (wiederum intra-psychischen) Vorstellungen der Rezipienten.

Eine genauere Betrachtung des Entstehungsprozesses einer Komposition bedarf also parallel einer Analyse ihrer unterschiedlichen Seinsformen (physikalische wie intra-psychische), da jener sich natürlicherweise darstellen läßt als Folge von *Übersetzungen* zwischen diesen.

Alle folgenden Aussagen haben zweifellos unterschiedlichen Grad an allgemeiner Gültigkeit: Das Spektrum geht von schlechthin wahren Sätzen, welche sich ergeben aus der Analyse der Materialeigenschaften selbst und unabhängig sind vom persönlichen Umgang damit, über Aussagen, die nur für bestimmte Typen oder Schulen von Komponisten gültig sind, bis hin zu sehr persönlichen und durchaus bestreitbaren Hypothesen. Die jeweilige Einordnung möge der Leser freundlicherweise selbst vollziehen.

Wie jede andere konkrete Begegnungsweise mit der *transzendenten Realität*, wie z.B. der Vollzug des Mordes oder des Geschlechtsaktes, entzieht sich der künstlerisches Schaffensprozeß nämlich nicht nur wegen der Höhe seiner Komplexität, sondern schlechthin *wesentlich* der vollständigen sprachlichen Darstellung.

3.1 Grundthese zur Entstehung eines kompositorischen Werkes.

Wir behaupten drei Ausgangsthesen über die Entstehung eines Kunstwerkes:

erstens Ein Kunstwerk entsteht als Folge von Modellen.

zweitens Ein Kunstwerk entsteht in der „Inneren Sprache" des Autors.

drittens Ein Kunstwerk entsteht als Netzwerk von Bestimmungen.

3.1.1 Zu Eins: Modell und Modelle.

Ein Kunstwerk entsteht ursprünglich zweifellos in der Form eines Modelles, d.h. einer Struktur der Vorstellung, weil „es selbst", das von uns oben angenommenen „Ding", jeder Wahrnehmung, also auch der Wahrnehmung des Autors, nicht erreichbar ist.

Dies ist trivial und vorerst nur eine Frage unserer gewählten Terminologie; zu dem von uns hypostatisierten „Kunstwerk als Ding an sich" gehört ja z.B. auch seine *zukünftige* Rezeptionsgeschichte. Diese zumindest ist auch dem Autor verschlossen.

Sofort aber wird es interessant, wenn wir genauer hinsehen. In der Tat kann der gesamte Entstehungsprozess als eine *Folge von verschiedenen* Modellen angesehen werden, wobei die einen aus den anderen durch diverse Transformationen hervorgehen.

Charakteristische Grundtypen dieser Modellstationen und Transformationen herauszuarbeiten ist Ziel der hiermit begonnenen Analyse.

3.1.2 Zu Zwei : Zum Begriff einer „Inneren Sprache".

Als weitere Arbeitshypothese fordern wir für jedes Individuum die Existenz einer „Inneren Sprache".

Über die Struktur der Inneren Sprache eines jeden Menschen können wir a priori keinerlei Aussagen machen, – nur die eine, *daß* sie eine Sprache ist.

Wenn „Denken" in diesem Zusammenhang das Operieren auf einem absoluten, also *autarken* Zeichenvorrat bedeuten soll, dann ist die Innere Sprache die Menge der dem Individuum möglichen Denkvorgänge.

Dieser Autarke Zeichenvorrat ist das terminale Alphabet der Inneren Sprache.

Dieser Autarke Zeichenvorrat wird gebildet von der Gesamtheit aller Vorstellungen des Individuums.

Die von uns geforderte Innere Sprache hat aber, wenigstens das wissen wir, Spracheigenschaften.

Das bedeutet zum einen, diese „autarken Zeichen" sind Entitäten, d.h. sie werden als selbstidentische und rekonstruierbare Einheiten postuliert.

Zum anderen heißt es, daß auf den autarken Zeichen operiert werden kann, es gibt Möglichkeiten der Verknüpfung, also Satzformen.

Das heißt weiterhin, es gibt für das Operieren auf den autarken Zeichen Regeln und Prinzipien, welche die Menge der „syntaktisch möglichen Aussagen" von jener der „unmöglichen Aussagen" unterscheidet.

NB sind diese internen Zeichen und Regeln keinesfalls unmittelbar darstellbar.

Erstens : Die Zeichen sind autark, d.h. sie bedeuten keinesfalls ein Objekt einer anderen Welt, sondern existieren nur im autarken inneren Symbolraum. Jede Beziehung auf ein Drittes, außerhalb des Denkens Liegendes sind Konstruktionen, die von der Pragmatik bewertet werden, die aber als neue, interpretierende Ebene hinzukommen und nicht in unseren Begriff fallen.

Zweitens : Das operierenden Individuum *selbst* braucht die Operationen nicht unbedingt *bewußt* zu erleben, ja – bestimmte Denkvorgänge können die Ebene des Unterbewußten nur unter erheblichem Aufwand oder gar nicht verlassen, sind aber dennoch exakt ablaufenden symbolische Operationen.

Künstlerisches Schaffen ist allemal autarkes symbolisches Operieren, sei es bewußt oder vorbewußt. Damit ist es im weiteren Sinne *immer durchaus „vernünftig"*, auch wenn es nicht reflexiv analysiert ist, und sein Material ist immer eine *Sprache* im Sinne der oben angegebenen zwei Eigenschaften.

Deshalb gehören auch die „Begriffe", welche das Ergebnis des *meta*sprachlichen Vorganges der *Abstraktion* sind, nicht notwendigerweise zu diesem autarken Zeichenvorrat der Inneren Sprache.

Der Satz „Es riecht heute wie an meinem siebenten Geburtstag, aber meine Nerven fühlen sich genau wie am letzten Freitag, nur noch etwas mieser" ist ein für uns selber durchaus verifizierbare und bewertbare Aussage.

Der „Geruch an meinem siebenten Geburtstag" und „das Gefühl vom letzten Freitag" nämlich gehören als Entitäten u.U. zu meiner inneren Modellwelt. Sie sind selbstidentisch und sie sind operabel, d.h. ich kann ihre Attribute vergleichen und bewerten, kann sie beliebig mit Namen versehen, kann mit ihnen Aussagesätze bilden und kann derartige Aussagen exakt oder quantifiziert verifizieren oder falsifizieren etc.

Darüber hinaus kann ich eine Meta-Ebene konstruieren, indem ich die Art, in der ich auf den inneren Objekten operiere, reflexiv beobachte und meine inneren Schlußregeln und Denkmethoden wiederum als innere Objekte betrachte[1].

[1]Die Existenz einer Inneren Sprache ist mit ein Grund sowohl für die Erlösungsbedürftigkeit als auch die Erlösungsfähigkeit des menschlichen Geistes. Der Mensch, aus Unfähigkeit verurteilt, andere zu verletzen (sog. „Erbsünde"), kann erlittene Verletzungen rekapitulieren und perpetuieren, und sich so der Verdammnis ausliefern.

Er kann aber auch im momentanen Erleiden oder Ausüben der Verletzung durch Wieder-Holen des Gewesenen die mögliche Handlungsalternative als wirklich empfinden.

3.1.3 Zu Drei: Entstehung als Netzwerk.

Jedes Kunstwerk entsteht in der Inneren Sprache des Autors. Dort entsteht es als *Netzwerk*. Die Knoten dieses Netzwerkes nennen wir *Bestimmungen*[2].

Diese Bestimmungen sind einzelne (materielle oder methodologische) Entscheidungen oder Entscheidungsalternativen, welche in ihrer Gesamtheit *ein* Modell des entstehenden Werkes ausmachen.

Dieses Modell nähert sich dem von uns behaupteten „existenten Werk" im Laufe der Arbeitszeit zunehmend an.

Dieses Netzwerk, so es einmal vollständig würde, wäre eine explizite Darstellung der Struktur des Werkes, von dem die fertige Partitur eine implizite ist. Beide Darstellungen wären von *identischem* Informationsgehalt und die Ausarbeitung der Partitur wäre ein simpler Vorgang von „Datentransformation".

Dieses Netzwerk ist aber auch nur *eine* explizite Darstellung des Werkes von vielen möglichen. Andere Explikationen sind z.B. das Ergebnis einer später durchgeführte Analyse, oder das Modell entstehend bei der bloßen Konsumption des Werkes im Kopfe des Hörers.

3.2 Versuch einer Strukturierung der Arbeitsphasen beim Komponieren.

Wir wollen nun versuchen, einige der Dimensionen dieser Netzwerke im Falle des musikalischen Werkes herauszuarbeiten.

Dies versuchen wir, indem informell eine kleine Kompositionstheorie entwickelt wird. Diese erhebt keinen Anspruch auf hundertprozentige Realisierung in irgend einer Praxis, sondern ist idealisiertes Grundmodell, welches einerseits zu kalkulieren erlaubt, andererseits in der täglichen Arbeit als als *eine* wichtige Schicht der Grundstruktur des Arbeitsprozesses (allerdings in unterschiedlichen Graden von Relevanz) stets nachweisbar ist.

Diese kleine Theorie drängt sich (als Beispiel) dem Autor auf, wenn er die Verfahren des akademisch abgesegneten Vorgehens, wie es z.B. in der Lehre Wolfgang Hufschmidts vermittelt wird, unter den Erfahrungen des Digitalrechnereinsatzes reimplementiert.

Wir unterscheiden idealisierter Weise vier(4) Arbeitsphasen oder -modi, zwischen denen in der Praxis u.U. ein ständiges, unmerkliches Hin- und Herspringen stattfindet, welche aber auch durchaus sauber unterscheidbar sein können, und die wir hier in eine künstliche Reihenfolge bringen (siehe Bild 3.1) :

[2]Der Autor dankt besonders Herrn Mathias Wittekopf, Bochum, für die fruchtbaren Diskussionen, welche zur genaueren Begriffsbildung in diesem Untersuchungsbereich wesentlich beitrugen.

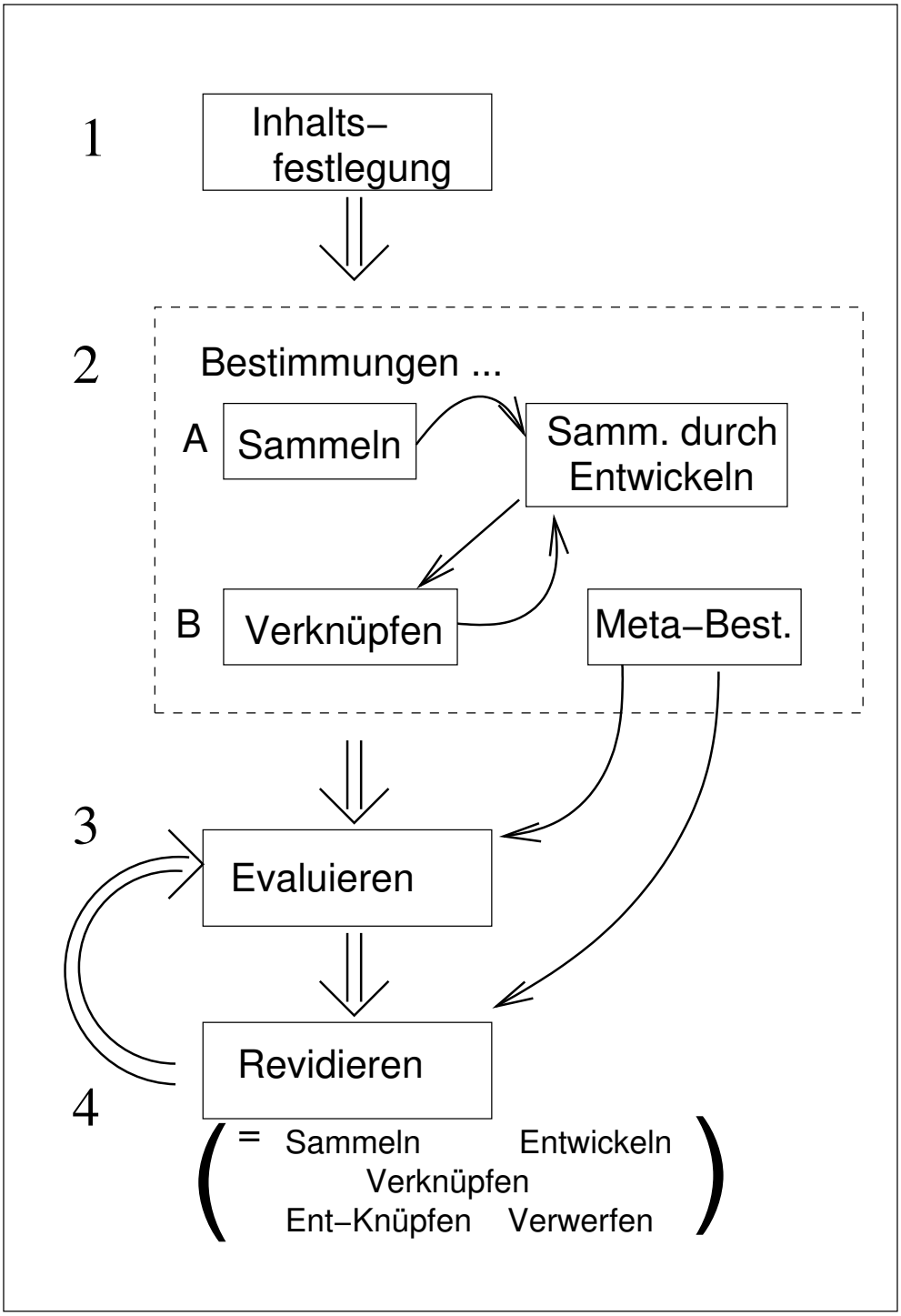


Abbildung 3.1: Arbeitsphasen bei der Erstellung einer Komposition.

1. Inhaltsfestlegung (Thema / Themen).
2.A (Einzel-)Bestimmungen Sammeln (das beinhaltet Suchen, Finden, auch Auseinander-Entwickeln, *nicht aber* Evaluieren von Vorschriften, sobald dieses „aufwendiger" ist !).
2.B Bestimmungen Verknüpfen (= Meta-Bestimmungen konstruieren und hinzufügen).
3 Evaluieren des Netzwerkes.
4 Netzwerk revidieren.

NB ist Phase vier inhaltlich identisch mit Phase zwei, erweitert um *destruktive* Maßnahmen wie das Löschen von nicht-bewährten Verknüpfungen, das Aussondern inkompatibler Ideen etc.

3.3 Vorder-, Mittel- und Hintergrund.

In den anschließenden Kapiteln werden wir auf die im folgenden ausgeführte detaillierte Klassifikation kompositorischer Akte und Entscheidungen aufbauen.

Dazu werden wir (zusammenfassend und vergröbernd) von *Hintergrund*, *Mittelgrund* und *Vordergrund* reden.

„Hintergrund" beinhaltet die (in Schritt *eins* obigen Ablaufplanes) anzutreffenden Entscheidungen und Voraussetzungen bezgl. inhaltlicher, philosophischer, historischer Randbedingungen etc.

„Vordergrund" beinhaltet das (qua praktisch-wirtschaftlichem „Ablieferungsvorgang" u.U. recht konkret definierbare) *Resultat* des Produktionsprozesses, – also das klingende Ereignis bewegter Luft im Aufführungsraum, das lesbare Notat, die „fertige" magnetische Schallaufzeichnung o.ä.

„Mittelgrund" ist alles andere, also besonders die in allen o.e. Arbeitsphasen (ab zwei-A) auftretenden Entscheidungen, ihre Objekte und deren Materialisierungen, – also Regeln, Motive, Algorithmen, Themen, Reihen, Vorschriften, Verknüpfungen nebst all deren Skizzen und Notizen etc.

Somit stellt sich die Entstehung eines Werkes vereinfacht dar als schrittweise Transformation von Hintergrund- über Mittelgrund- in Vordergrundinformation.

Vorgreifend, um Mißverständnisse zu vermeiden, drei Bemerkungen :

Erstens:

Die anscheinend problemlos dem Vordergund zurechenbaren Strukturen sind mitnichten endgültig oder fixierbar oder terminal etc. Vielmehr wird sich später zeigen (4.4), daß z.B. die *Notation von Musik* nur scheinbar ein reines Vordergrund-Phänomen ist. Im Gegenteil: der kommunikative Vorgang des Verstehens eines Notates ist nur möglich, als er sofort Mittelgrund-Information induziert /rekonstruiert.

Tatsächlich kann man durchaus zu dem Ergebnis kommen, daß Vordergrund ein rein virtueller Arbeitsbegriff (=Zielvorstellung) ist und daß streng genommmen keinerlei Materialisierungsformen existieren können, die diesem Begriff tatsächlich entsprechen.

Zweitens:

Die tatsächliche *kompositorische Tätigkeit* spielt sich ab (und ihre Gegenstände befinden sich) im *Mittelgrund*.

Kompositorisches Handeln ist Operieren auf „Objekten des Mittelgrundes", – besonders Transformation zwischen den verschiedenen Teil-Schichten des Mittelgrundes.

Dies wird besonders im Bezug auf die Anforderungen an den Einsatz des Digitalrechners im Kompositionsprozeß zentrale Bedeutung bekommen (siehe 5.4).

Aber auch der Vorgang des „*Begreifens*" einer Komposition kann, wie wir sehen werden, als kommunikativer Akt mit Objekten und Strukturen des Mittelgrundes aufgefaßt werden.

Jede *Notation* von Musik ist in unserer Terminologie zwar einerseits Teil des (physikalischen) Vordergrundes, andererseits aber (1) stellt sie *Mittel*grund-Informationen dar, (2) ist sie nur in einem Mittelgrund-Kontext überhaupt lesbar, und (3) wird sie vom Lesenden sofort unbewußt im Vorgang des Lesens wieder in Mittelgrundinformation zurückübersetzt.

Drittens :

Die Terminologie lehnt sich an das Werk Heinrich Schenkers an.

Wichtiger Unterschied ist jedoch, daß bei diesem Hinter-, Mittel- und Vordergrund allesamt *diachrone* Strukturen sind, welche alle denselben Zeitablauf, die Dauer des Werkes, zunehmend feiner strukturieren, – während bei uns im Mittel- und besonders im Hintergrund Out-Of-Time Informationen und Entscheidungen fast wichtiger sind.

Das Urlinie-Modell nach Schenker erscheint in unserem Zusammenhang als eine „eingefrorene" Darstellung des von uns dynamisch gedachten Entscheidungsprozesses und als Bevorzugung der großräumig-funktionalharmonischen Hintergrundsinformation unter Weglassung aller anderen.

Diese apodiktische Verkürzung ist aus dem historischen Zusammenhang durchaus verständlich und mindert die Bedeutung von Schenkers Ansatz auch für unsere Arbeiten keineswegs.

3.4 Inhaltsfestlegung, Hintergrund.

Schon lange Zeit vor dem Entschluß zu einem *bestimmtem* Werk werden häufig verstreute Skizzen von *Materialien* (Motive, Reihen, Zahlenfolgen, Formteilskizzen, Rhythmen, Algorithmen und dafür Parametersätze, Gedichtzeilen, Liedtexte, Formideen, weiterhin Meta-Notizen wie: Anzuwendende Kompositionsverfahren und dafür Parametersätze, staztechnischen Aufgabenstellungen, zu studierende/studierte Vorläufer, historische Beispiele und deren Analyse etc.) gesammelt, z.T. abhängig von einander, z.T. noch völlig vereinzelt.

Dennoch kann (in unserem abstrahierten Modell) der *Zeitpunkt des Beginnes der Existenz eines bestimmten Werkes* „o.b.d.A."[3] auf den Zeitpunkt des Vollzuges des Arbeitspunktes 1, „Inhaltsfestlegung", festgelegt werden.

[3]= „ohne Beschränkung der Allgemeingültigkeit."

Dieser Punkt beinhaltet die Festlegung von ...

1. Thema,
2. Zielkontext,
3. Ausgangskontext.

3.4.1 Zu: Thema.

Als *sogenanntes* „Thema" kommt (nach der reinen Lehre) ausschließlich *ein einziger* Gedanke in Frage.

Dieser jedoch kann aus einem fast beliebigem Bereich unserer Realität entnommen werden, – kann z.B. typischerweise sein ...

- ein existierendes, aus seinem Vordergrund zu zitierendes *Stück Musik*, – von kurzem Fragment bis zum ganzen Werk (cf. Luciano Berio „sinfonia", zitierend einen vollständigen Satz von Mahler).
- eine zitierte Mittelgrund-Struktur eines existierenden Werkes, z.B. die Reihe einer Webern-Komposition.
- ein existierendes Kunstwerk beliebiger Gattung (ein Gedicht, Schauspiel, Bild, Skulptur, Choreographie etc.), dessen Struktur herausanalysiert und in Musik „übersetzt" werden soll.
- ein philosophischer Satz.
- ein technologischer Gegenstand (simulierte Verkehrsstauung, Verarbeitungsstraße im Stahlwerk, der Grün-Rhythmus der Fußgängerampel am Postscheckamt etc.).
- ein politisches Ereignis, ein Stück Realzeit (z.B. wie in „November '52" von Earle Brown).
- ein satztechnisches Problem.

3.4.2 Zu: Zielkontext.

Hierhin gehört die Beantwortung so wichtiger, aber oft mißachteter Fragen wie ...

- *Wie* soll das Werk überhaupt *physisch in Existenz* treten.
- Genauer: Für welche *Besetzung* schreibt man ?
- Noch genauer: Was ist der Schwierigkeitsgrad? Was ist der technische / personelle / finanzielle Aufwand für eine Einstudierung / Aufführung ? Wie hoch qualifiziert müssen die Interpreten sein ? Sind Alternativbesetzungen möglich / vorgesehen / vorbereitet ?

- Ggfls: *Wieviel Zeit* steht für die Erstellung des Werkes / für die einzelnen Arbeitsphasen zur Verfügung ? In welchen Aggratzuständen (Parititur, Mitschnitt, Stimmen etc.) liegt das Werk wann und zu welchen Kosten vor ?
- Wie / in welcher Form kann das Werk nach der Uraufführung / Auftragsablieferung / Produktion *weiterleben* ?

3.4.3 Zu: Ausgangskontext.

Unsere Nomenklatur von „Zielkontext" und „Ausgangskontext" ist vielleicht nicht so glücklich, – beide bezeichnen wichtige Ausgangs-Fragen, die am Anfang jeder kompositorischen Tätigkeit zu beantworten sind, (wenn dies auch häufig vorbewußt und traditionsgesteuert und gewohnheits-gesteuert geschieht), – beide betreffen das Ziel der Tätigkeit, nämlich den kommunikativen Akt.

Beide berühren sich genau im Punkte dieses kommunikativen Aktes und sind insofern symmetrisch, als unser „Zielkontext" die Rahmenbedingungen der *Produktion* festlegt, während der „Ausgangskontext" diejenigen der *Rezeption* versucht zu umreißen (also zu vemuten oder vorauszusetzen).

Unter der Bezeichnung „Ausgangskontext" verstehen wir die Beantwortung der einzigen Frage ...

- An *welchen Rezipienten* richtet sich das Werk?

Mit der (rein theoretisch!) vollständigen Beantwortung dieser Frage und der Explizierung aller aus dieser Antwort folgenden Konsequenzen läge nämlich vor eine vollständige Beschreibung der *vorauszusetzenden Sprachkompetenz*, – also der Sprache, derer ich mich im Werke bedienen kann, eben das *voraussetzbare* Ausgangssprachverständnis.

Zwei Unterfälle mögen das verdeutlichen :

Erstens :

Die Verwendung von *Zitaten* ist ein plakativer Extremfall und verdeutlicht als solcher für subtilere Fälle lediglich das Prinzip :

Gesetzt sei also ein Zitat, was nicht nur *als ein solches, ein Zitierendes* erkannt, sondern wo *das Zitierte wiedererkannt* werden soll, mit all seiner Aura und Semantik (= „Identifizierbarkeit"gefordert) .

Dann muß dieses (stark vereinfacht gesprochen!) dem Hörer „bekannt sein"[4].

Die Verwendung des Eisler-Solidaritätsliedes (im Wiedererkennungs-Sinne) spricht also ein anderes Publikum an, als die Verwendung eines Luther-Chorales, wobei es u.a. auch eine Schnittmenge gibt, – und eine Vereinigungsmenge[5].

[4]Oder zumindest muß er in ersterem Falle (= „Zitat als solches erkannt und neugierig geworden!") problemlos an das Original gelangen können, – die Rezeptionsgeschichte des Werkes erschöpft sich ja u.U. nicht im einmaligen/erstmaligen Hören!

[5]„Wer vieles bringt, wird manchem etwas bringen."

Allerdings ist in diesem Zusammenhange zu bedenken, das die *anderen Funktionen* des Zitates (jenseits des vollen Wiedererkennens und Identifizierens) auf der Materialebene sich *selbstverständlich auch denjenigen* Hörern mitteilen, welche das Zitat *nicht bewußt* wiedererkennen :

- Die aus der Verwendung einer Choralmelodie folgenden satztechnischen/intervallischen Konsequenzen,
- und zumindest die aus diesen folgende „choralmäßige Aura".
- Der rhythmische Gestus/Impetus, den z.B. das Solidaritätslied dem Satz allemal mitgeben würde.
- etc.

Zweitens :

Auch die Entscheidung für die anwendbaren / „goutierbaren " / akzeptierbaren Klanglichkeiten ist hier, bei der übergeordneten Frage der Zielgruppenbestimmung, zu beantworten.

Damit ist (evtl. aber keinesfalls immer) verbunden die Entscheidung über die angemessenen / anwendbaren / zu meidenden *Satztechniken.*

Wer dem Altenheim eine Zwölftonreihe vorsetzt, erweckt normalerweise höchstens HITCHCOCK-Stimmung.

Hier, in der (aus kompositorischer Sicht halt so genanntem) Ausgangskontext-Festlegung spielen sich evtl. (z.B. unter Umständen, welche Wahlentscheidungen erzwingen) die eher politischen oder marktorientierten, ja sogar die ethischen Überlegungen und Entscheidungen ab, welche auch einen zentralen Gegenstand der kompositorischen Tätigkeit und Lehre ausmachen, aber von unserem *formalen* Modell selbstverständlich nicht erfaßbar sind.

3.5 (Einzel-)Bestimmungen Sammeln.

Wir versuchen hier, durch die künstliche Trennung der Vorgänge zwei-A, „Bestimmungen Sammeln" und zwei-B, „Bestimmungen Vernetzen", etwas Ordnung zu schaffen.

Im konkreten Ablauf des Komponierens wird jedoch *zwischen den Materialgewinnungsschritten zwei bis vier* beständig hin und her gesprungen! (Dies wird genauer betrachtet unten in 3.9).

Die Art der Bestimmungen, welche gesammelt werden können, wurde ganz zu Beginn unserer Modellbeschreibung schon einmal aufgezählt (3.4), - in der Tat gehen ja unabhängige Einzelskizzen in der Praxis fast immer zeitlich dem Werkentschluß (evtl. unmerklich) voraus.

Den Begriff „Bestimmung" im Sinne des Vorganges „Sammeln" nennen wir auch „Einzelbestimmung", und begreifen darunter u.a. die Dinge, welche sonst vielleicht Materialien, Materialskizzen, Einfälle, Ideen, Vorhaben etc. genannt werden.

NB können Bestimmungen durchaus auch auf die Ebene des in Schritt eins ausdrücklich singulär gesetzten „Themas" fast gleichberechtigt hinzutreten. So gab es z.B. ein Gruppen-Kompositions-Projekte namens Todesfuge, bei dem das gleichnamige Gedicht von Celan zentrales Thema war und seine lineare Gestalt den Ablauf des ganzen Stückes bestimmt. Als gleichberechtigtes Gegenstück trat dann der Contrapunctus XVIII aus Bachs Kunst der Fuge hinzu, welcher dann (in Augmentation) die konkreten Dauernproportionen steuert, auf welche der Gedichttext projiziert wird.

NB kann es dabei aber auch passieren, daß im Laufe eines tatsächlichen Arbeitsabschnittes an einem Werk x festgestellt wird, daß die Themenfestlegung aus dem (in unserem abstrakten Modell stets zeitlich vorangehenden) Schritt eins *revidiert* werden muß.

In unserem formalen Modell würde dies dargestellt durch den Beginn eines *neuen* Projektes x_1 unter Übernahme der bereits gesammelten Bestimmungen.

NB spielt sich *hier*, bei der Diskussion und Entscheidung, *welche* Bestimmungen in das Netzwerk (in *welcher* Form) überhaupt aufzunehmen sind, – *wie* weitere nötige Bestimmungen möglichst nicht additiv hinzuzufügen, sondern vielmehr aus dem Kern des Vorhandenen zu *entwickeln* sind, – *hier* spielt sich der eigentlich spannende ästhetische (= kreative / kritische / pädagogische / didaktische / wahrnehmungstheoretische / historiologische) Prozeß ab.

Diesen aber zu behandeln ist ja keinesfalls Gegenstand dieses kurzen Abschnittes, sondern vielmehr, halbwegs saubere *formale Rahmenbedingungen* für mögliche inhaltliche Diskussionen zu bieten.

3.5.1 Bestimmungen Entwickeln.

Notwendige Bedingung nämlich für die ästhetische Konsistenz des Endproduktes ist es allemal, daß die Einzelbestimmungen des Ausgangsnetzwerkes so wenig wie möglich den Charakter einer rein additiven Anhäufung tragen.

Je weniger Ausgangsbestimmungen zu seiner Errichtung notwendig sind, um so konziser ist das resultierende Werk.

Vielmehr sind *so wenig willkürliche Hinzunahmen wie möglich* anzustreben, – benötigte neue Teilmaterialien sind möglichst aus Vorhandenem abzuleiten, zu entwickeln.

Das Wort „Entwickeln" deutet an, daß hier zwar synthetische Vorgänge stattfinden (Anwendung eines *Verfahrens* auf ein Material), das Ergebnis jedoch als ein *analytisches* gelten soll.

Steht im Zentrum des thematischen Bereiches (s.o.) z.B. eine Gedichtzeile, so können unter Hinzunahme diverser bekannter Text-Lese-Techniken aus diesem sprachlichen Objekt alle möglichen benötigten Materialien einfachst entwickelt werden: Mit Hilfe von Morsealphabet / Kabbala-Technik (Buchstaben als Zahlzei-

chen) / Herausfiltern von Tonnamen, Abkürzungen oder Vortragsbezeichnungen / Klanganalyse, Übertragung der Phonetik in eine falsche Sprache / Braille-Schrift / Flaggen-Alphabet etc. können aus wenigen Worten Text ganze Welten von Rhythmen, Melodien, Proportionsformeln, graphische Formpläne etc. abgeleitet werden.

((Obwohl diese Art der Materialgewinnung technisch gesehen durch „Verknüpfungen der applikativen Art" geschieht (= Anwendung eines Verfahrens auf ein Material), und deshalb streng genommen weiter unten behandelt werden müßte, behandeln wir sie hier unter dem Begriff „Sammeln", weil der Aspekt des zunächst unverknüpften brain-storming hier arbeitstechnisch eindeutig im Vordergrund steht.))

Die ganze Bandbreite analytischer Materialgewinnungsverfahren ist hier anzusiedeln :

Eine ganze Strukturschicht eines historisch gegebenen Corpus kann zum Material einer neuen Komposition werden (z.B. die Einsatzabstände des Themas in einer Bach-Fuge oder die Menge von Akkord-Formen eines Debussy-Preludes.)

Ebenfalls hierhin gehört eine stattfindende *Improvisation* zur Materialgewinnung: Das tatsächlich gefundene Material (Tonhöhenfolgen, Harmonieverläufe etc.) ensteht ja gerade durch nachträgliche *Analyse* des im improvisierenden Bewußtseinszustand geschaffenen Totals.

3.5.2 Zeitpläne als spezielle (Einzel-)Bestimmungen.

Innerhalb der (ganz oben 3.4) nur beispielhaft aufgezählten Mannigfaltigkeit an „Typen" von (Einzel-)Bestimmungen fordern wir nun, daß es eine ausgezeichnete Teilmenge gibt, deren Elemente wir *Zeitpläne* nennen.

Ein „Zeitplan" sollte eigentlich heißen „Dauern-Hierarchie-Generator-Algorithmus".

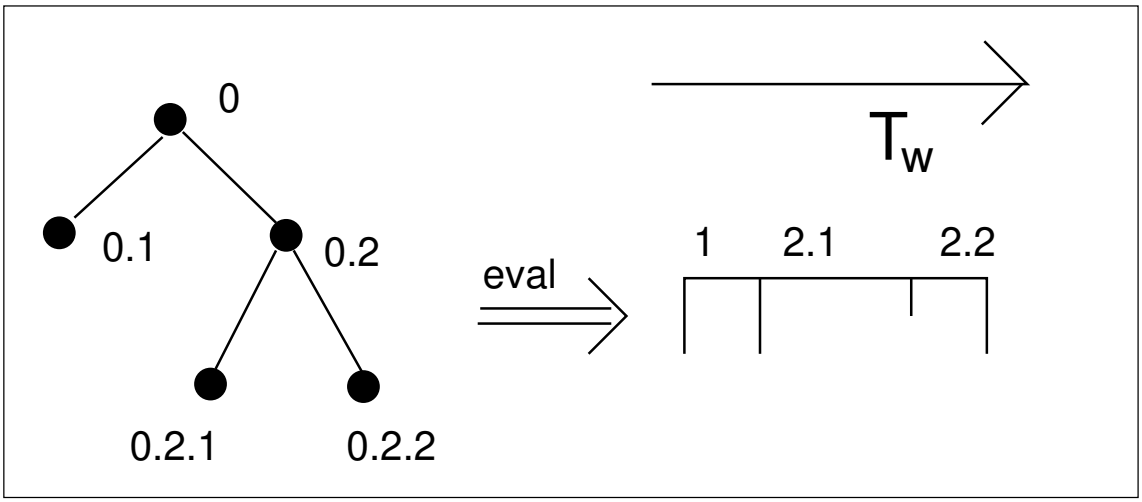


Abbildung 3.2: Dauernbäume als Ergebnis der Auswertung von Zeitplänen.

Ein Zeitplan ist ein (Objekt repräsentierend einen) Algorithmus, welcher, (sobald er hinreichend evaluiert ist, also spätestens, wenn alle seine Eingangsparameter mit konkreten Werten versehen sind) einen *Baum von Zeitdauern* liefert.

Jeder Knoten das Baumes repräsentiert die Dauer, welche der (lückenlosen und die Reihenfolge respektierenden) Aneinanderreihung der Dauern seiner Subknoten entspricht.

Ein Sonderfall sind dabei degenerierte Bäume mit nur einer Ebene, welche eine Folge von Dauern liefert.

Die Sub-Knoten eines jeden Knotens sind geordnet und werden von 1 bis n nummeriert, - die Wurzel eines Baumes heißt „0", Jeder Knoten wird durch die Folge der Indizes bezeichnet, die auf dem Pfad zu ihm von der Wurzel liegen.

Bild 3.2 zeigt unsere Symbolisierung der Auswertungsergebnisse von Zeitplänen.

Es ist natürlich, daß diejenigen Objekte, welche *ablaufende Zeit quantitativ strukturieren* im Falle der Komposition von Musik teilweise grundlegend *anders behandelt* werden müssen als die zeitinvarianten (= „out-of-time") Objekte (Tonmengen, Silbenfolgen, Proportionen, etc.).

Dennoch ist der Eintritt eines Objektes in die Klasse „Zeitplan" ein definitorischer Akt : Die reine Zahlenfolge

$$\langle\ 1\ ,\ 2\ ,\ 4\ \rangle$$

kann als zunächst ohne weiteres als (recht primitiver) statischer Materialgenerator verwendet werden.

Als Zeitplan *deklariert* ergeben sich sofort Einschränkungen und Randbedingungen, z.B. daß dieser nur parametrisiert mit einer Dauern-Einheit instantiierbar ist, daß vor einer Verwendung die Art der Zeitangabe entschieden werden muß etc.

3.5.3 Der Bestimmungsraum, Raumachsen als spezielle Meta-Bestimmungen.

Im folgenden Arbeitsschritt zwei-B wird eine der wichtigsten Formen der Verknüpfung von Bestimmungen die sog. „Plazierung" sein, welche die gegenseitige Positionierung der beteiligten (im nächsten Abschnitt eingeführten) „Werkfasern" in erster Linie bezogen auf deren Zeitpläne realisiert.

Die Entwicklungsarbeit unseres kleinen Modelles zeigte aber bald, daß die gegenseitigen Zeitverhältnisse sich nur in Verbindung mit inhaltlichen Relationen zweckmäßig darstellen lassen. Dies führt zur Forderung eines *werkweiten, aber werkspezifischen Bestimmungsraumes*, auf dessen Koordinaten sich alle auszuführenden „Plazierungen" beziehen.

Dieser Bestimmungsraum hat n Dimensionen, welche in natürlicherweise in drei Klassen zerfallen:

- die eine Achse „Realzeit" $= \hat{T}$. Diese kann als gültige Koordinatenwerte entweder die Gesamtdauer des Werkes tragen, oder einen beliebigen (abstrakten) Zeitabschnitt im Falle der Generierung noch nicht im Gesamtwerk plazierter „pre-takes".
- die mehreren *Stilabhängigen* Achsen, benutzerdefiniert qua Meta-Bestimmung.
- die eine Achse *geplante Arbeitsphasen*, benutzerdefiniert qua Meta-Meta-Bestimmung.

Im Falle der klassischen post-seriellen, poly-paradigmatischen, phänomenologisch-symbolischen Kompositionsweise (nach HUFSCHMIDT et al.) wären die Stilabhängigen Achsen wahrscheinlich „Parameter" und „(Material-)Schicht".

Erstere Achse enthielte z.B. die Koordinatenwerte Tonhöhe, Rhythmus, Lautstärke, Klangfarbe, – die andere z.B. die beiden Koordinatenwerte „CELAN-Gedicht-Text-Schicht" und „BACH-Zitat-Schicht".

Es entsteht in diesem Falle insgesamt ein vierdimensionaler Bestimmungsraum.

NB ist die *Festlegung* selbst der Koordinatenpunkte der „Parameter"-Achse bereits ein erster Schritt der Materialgewinnung; sie ist eine konstruktive ästhetische Entscheidung, also (Meta-)Bestimmung, insofern sie sowohl Stil-beeinflussend als die Gewinnung weiteren Materials vor-definierend ist:

Eine herkömmliche romantische Filmmusik würde hier eher die Koordinaten ⟨ melodisches Material, harmonisches Material, Satztechnik, Instrumentation ⟩ haben, – eine Zwölfton-Kompositon hätte vielleicht die Koordinaten ⟨ Reihenmodus, Reihen-Transposition, Reihen-Verwendung (= Satztechnik) ⟩.

Eine der Konsequenzen der analytischen Anwendung des „Denkens der Neuen Musik" ist in diesem Zusammenhang, das Eigenschaften eines Satzes, die herkömmlicherweise als *Meta*-Bestimmungen betrachtet wurden, nun durchaus als *skalierbar* aufzufassen sind und zu Koordinaten einer Achse werden können : Die relative Länge der Abspaltungen einer klassischen Durchführung, die prozentualen Anteile von An- und Abwesenheit thematischen Materials, – ja selbst die Disposition der zur Anwendung kommenden kontrapunktischen *Verfahren* kann analytisch behandelt werden wie die „eindimensionalen" Parameter Lautstärke und Tonhöhe.

Bild 3.3 zeigt als Beispiel einen Bestimmungsraum des eher romantischen Typs.

NB ist bemerkenswert, daß die Achse „Arbeitsphasen" völlig unterschiedlichen Stellenwert haben kann.

In der historischen Voltage-Control-und-Tonband-Technik der Siebziger und Achtziger wurden z.B. arbeitsaufwendige und nicht ohne weiteres rekonstruierbare Verarbeitungsprozesse (Evaluationsläufe) (teils schon zur reinen Primärmaterial-Gewinnung) gepipelined.

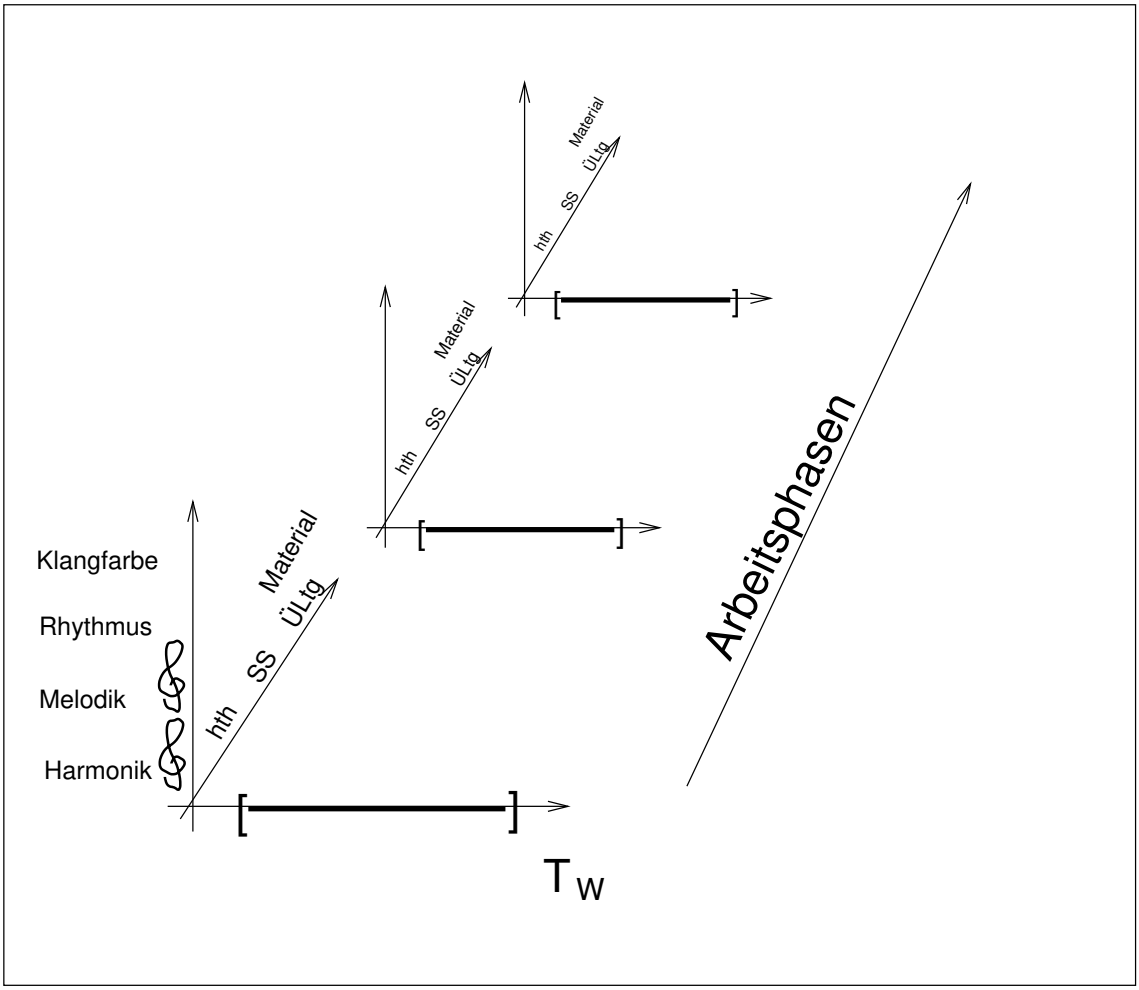


Abbildung 3.3: Beispiel für einen Bestimmungsraum.

Ein typisches Beispiel aus jener guten alten Zeit wäre eine Folge wie ...

$\Longrightarrow$ Material-(Ausgangs-)Band zerhacken und verwürfeln,
$\Longrightarrow$ dann qua in-time-session durch Tape-Transposition $\Longrightarrow$ `ADSR-VCA`-Zerhacker $\Longrightarrow$ Hall und Echo jagen, dabei in Realzeit *reaktiv* regeln und pegeln,
$\Longrightarrow$ Ergebnisband dieser Session seinerseits nach Schritt eins und zwei behandeln.
$\Longrightarrow$ Ebenso das Ergebnis dieser Session.
$\Longrightarrow$ Fünf derartiger Bänder zusammenmischen.
$\Longrightarrow$ Das Summenband auch zweimal nach Schritten eins und zwei behandeln.
$\Longrightarrow$ etc.

In diesem stilistisch/technologischen Umfeld reichte die Arbeitsschritt-Planung tief in die „formbildenden Tendenzen des Materials herein" und waren u.U. ihrerseits *selbst Gegenstand* kompositorischer Dispositionstechniken (z.B. Zufallsgenerator oder Zahlenfolgen-Permutation zur Bestimmung der Anzahl der Hall-Durchläufe, der Reglerstellungen o.ä.).

Auch in unserer Fallstudie wird sich zeigen, daß die Planung der praktischen Arbeitsschritte Auswirkungen bis in die Organisation der rechnerinternen Datenstrukturen hat, und damit auch die ästhetischen Entscheidungsmöglichkeiten des Komponisten durchaus mitbestimmt (s.u. 6.8).

Im U-Musik-Bereich wiederum könnten die Arbeitsschritte „erfinden", „aussetzen" und „arrangieren" sogar von *verschiedenen Beteiligten* durchgeführt werden, so daß dieselbe Achse hier ganz anderen Stellenwert erhält.

An Werken der „Übergangszeit", wie z.B. „Nested Loops" von Dirk Reith (Reith, 1980), kann man Formbildende Tendenzen der Arbeitsverfahren sehr schön nachweisen:

Dort wurde (1) eine aus dem Kompositionsprogramm Projekt Eins von Gottfried Michael König generierte Tonhöhenstrukturen (2) mit einem digitalen Synthesizer realisiert und dann (3) im Analogsynthesizer SynLab von Hofschneider grundlegend verarbeitet.

3.6 Bestimmungen verknüpfen.

Die aktuell vorliegenden Bestimmungen (zu *Beginn* also nur die gesammelten und – sofern nicht auseinander abgeleiteten – unzusammenhängenden *Einzel*bestimmungen) werden nun „vernetzt", – technisch gesprochen : verschiedene Relationen werden auf ihnen konstruiert. Dies ist in Bild 3.1 der Schritt zwei-B.

Das Erstellen einer Verknüpfung kann auch aufgefaßt werden als (und im Material heruntergefaltet werden auf) die Hinzufügung neuer Bestimmungen, die dann „Meta-Bestimmungen" zu nennen sind.

3.6.1 Werkfasern als Ergebnis der Verknüpfungsart „Plazieren".

Die zentralen Bausteine, in deren Verknüpfung, Anordnung, Entwicklung, Ordnung etc. das Handwerk des Komponierens sich abspielt, nennen wir Werkfasern.

Eine Werkfaser ist ein Drei-Tupel, welches in Beziehung setzt ...

- ein (Hyper-)Rechteck unseres oben definierten Bestimmungsraumes,
- einen Zeitplan
- und eine Menge von Bestimmungen (Kompositionsalgorithmen und ihre Parameter) nebst ihrer *Zuordnung* zu den Knoten des Zeitplanes.

Eine Werkfaser bedeutet, ...

- daß für die ausgewählten Koordinaten (Dauer = Realzeit-Abschnitt / Arbeitsphase / stilabhängige Material- und Parameter-Koordinaten)
- und folgend dem ausgewählten Zeitplan
- die angegebenen Bestimmungen „gelten sollen", sie also evaluiert werden sollen, um das klangliche Endresultat (oder ein Zwischenresultat) zu liefern oder zumindest mitzubestimmen..

Unter den angegebenen Bestimmungen können sich ganze Kompositionsprogramme befinden, wie es für das Projekt MW0 = mkSinf geplant gewesen ist, – entweder einschließlich ihrer vollständigen Parametersätze, oder diese werden (qua pipelining, s.u.) von parallelen Werkfasern als Ausgangsergebnis geliefert.

Die Zuordnung einer gegebenen Werkfaser zu den Koordinaten unseres Bestimmungsraumes nennen wir im folgenden seine „Plazierung". Die Plazierung bezg. der Zeitachse (Gesamt-Werkdauer) ist eine der wichtigsten Bestimmungen jedes Formteils, da sie die letztlich herauskommende zeitliche Erscheinung des Endproduktes allemal mitbestimmt.

Eine *Vollständige* Werkfaser ist eine solche, für die (1) alle o.e. Eigenschaften bereits bestimmt sind, und (2) in der alle referierten Algorithmen vollständig parametrisiert sind.

Vollständige Werkfasern können somit in zwei verschiedenen Lebensabschnitten eines Werkes ihre Bedeutung haben : Zum einen während seiner Entstehung als die wichtigsten „Bauklötze" des spielerisch-kreativen Prozesses im Mittelgrund des kompositorischen Denkens, – zum anderen bei der Rezeption des Werkes als konstitutive Bestandteile seiner Wirkungsweise, als die Fasern seiner Substanz.

Werkfasern sind Resultate der Verknüpfung von Bestimmungen, als solche ihrerseits (Meta-)Bestimmungen und so neuer Bestandteil unseres gesamten Bestimmungsnetzwerkes.

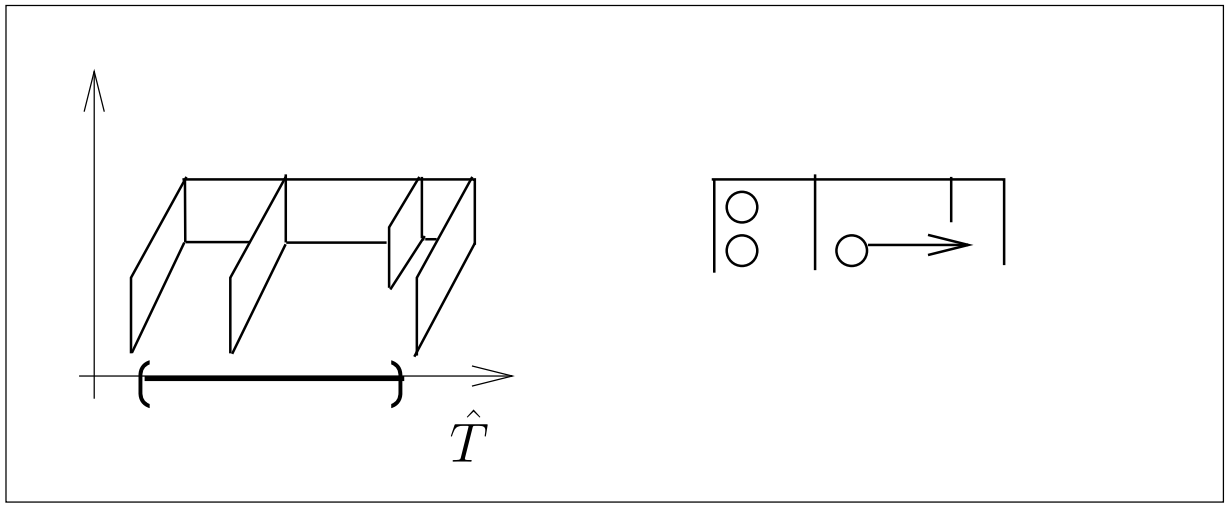


Abbildung 3.4: Zwei Ansichten auf eine Werkfaser.

3.6.2 Vollständige Werkfasern.

Eine Vollständige Werkfaser bezieht sich auf einen Teilabschnitt der Gesamtdauer des Werkes[6].

Fordern wir für unser Modell die Existenz eines ausgezeichneten Objektes der Klasse Werkfaser namens „Gesamtform", welches die gesamte Dauer des Werkes umfasst (und dem ein entsprechendes Zeitplan-Objekt zugeordnet ist).

Dann können wir formulieren, daß *jede* Vollständige Werkfaser (bis auf „Gesamtform") ihre Zeitachsen-Plazierung durch Referenz auf eine andere („übergeordnetete") Werkfaser bezieht[7], welche sie „ausfüllen" soll.

Dies wird kompliziert dadurch, daß eine plazierte Vollständige Werkfaser sich auch beziehen kann auf die *Summe* mehrerer, auch *innerer* Knoten des Zeitplanes der übergeordneten Werkfaser: „Während Exposition und Reprise schweigen die Trompeten!". Dies sollte auf der Ebene der *Zeitpläne* modelliert werden, indem diese auch nicht-disjunkte Mengen von nicht-benachbarten Zeitintervallen (formuliert qua expression) repräsentieren können.

Bild 3.4 zeigt Innen- und Außenansicht einer Werkfaser: Einerseits ist die Werkfaser plaziert im Bestimmungsraum, andererseits attributiert mit (Einzel-)Bestimmungen (Algorithmen, Daten, Regeln etc.)

3.6.3 Werkfaser-Polyphonie.

Spätestens seit Erfindung der seriellen und post-seriellen Kompositionsweisen ist Analytikern und Komponisten bewußt, daß im Bestimmungsraum von Zeit, Parametern, Material, Arbeitsschritt etc. eine komplexe *Polyphonie* unabhängiger Werkfasern durchaus sinnvoll sein kann.

Bild 3.5 zeigt einen bewußt kompliziert gewählten Fall:

Die Werkfasern sind dabei symbolisiert durch Serra-ähnliche, frei im Raum schwebende (= unsichtbar aufgehängte) Stahlprofile. Diese sind an ihrer Rückwand jeweils mit ihrem Namen bezeichnet.

Die stilabhängigen Raumachsen seien (im postseriellem Stil) *Material* und *Parameter*. Formteil A strukturiert dann die Gesamtdauer des Werkes für zwei Materialschichten und zwei Parameter. Die erste Teildauer (A_1) wird ihrerseits je Material unterschiedlich und für die Parameter gleichartig strukturiert durch Formteil B_1 und B_2, die zweite Teildauer wird genau gegensätzlich behandelt (unterschiedliche Strukturierung B_3 und B_4 je Parameter, aber gleich für beide Materialien). Werkfaser C hingegen strukturiert zwei Parameter eines Materials und zwei Materialien in einem Parameter gleichartig.

NB: Eine hinreichend vollständige *Algebra* aller sinnvollen Werkfaser-Plazierungen und eine diese repräsentierende Termsprache ist noch nicht entwickelt und sollte damals Gegenstand des leider nicht zustande gekommenen Projektes mkSinf sein. Sie wäre mit Sicherheit eine Herausforderung.

[6] ... oder auf eine Menge nicht-benachbarter Teilabschnitte!

[7] Genauer gesagt: Durch Referenz auf einen Knoten des ausgewerteten Zeitplanes der übergeordneten Werkfaser.

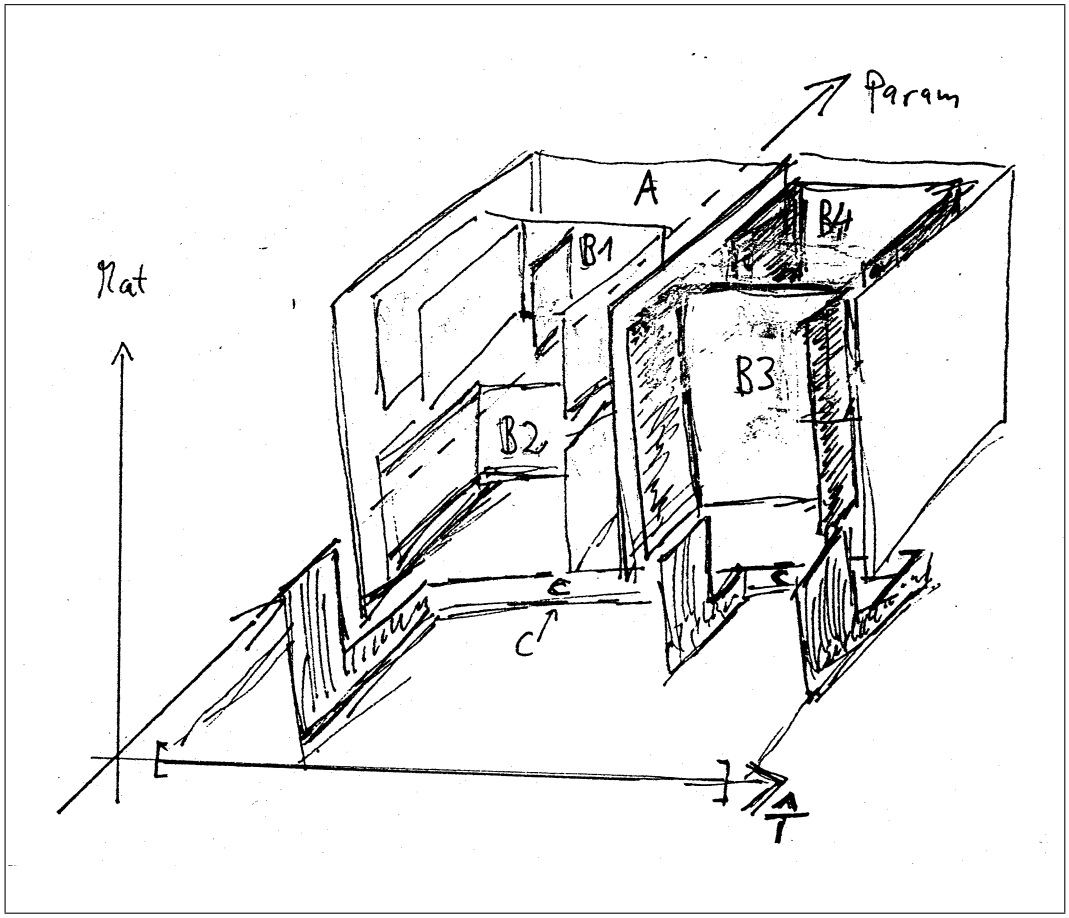


Abbildung 3.5: Komplexes Beispiel von Werkfaser-Polyphonie.

Ähnliches gilt auch schon für unseren (etwa weniger-dimensionalen) Begriff des Zeitplanes. Auch hier sollte dringend der Versuch einer Spezifikation möglicher Algebren versucht werden!

Bild 3.6 zeigt den Fall, daß Werkfasern auf der Achse „Arbeitsphase" unterschiedlich positioniert werden: Harmonik und Stimmführung sind in diesem Beispiel die im Arbeitsgang „Manuskript-Erstellung" einzigen Bestimmungen, im Arbeitsgang „Particell-Schreiben" kommen Oktavlage, Artikulation und Vortragsbezeichnung hinzu, – diese, nicht jene, bleiben auch im Arbeitsgang „Partiturreinschrift", in welchem Instrumentenwahl, Spielweisen, etc. noch hinzutreten.

Interessanterweise kann auch eine solche Werkfaser-Polyhonie, die in diesem konstruierten Beispiel vielleicht etwas künstlich wirkt, auch bei der *Analyse* klassischer Beispiele durchaus sinnvolle Ergebnisse, ja unerwartete Erkenntnisse liefern:

Es kann z.B. durchaus adäquat sein, einen MOZARTschen oder BRUCKNERschen Sinfoniesatz nicht nach dem herkömmlichen Formverständnis zu gliedern (also nach einem hierarchischen Formplan ⟨ Expo, Df, Rp, Cd ⟩ = ⟨ ⟨ HTh, SS, SchlGr ⟩ ⟨ Df1, DfHth, Rf ⟩ ⟨ HTh, SS', SchlGr' ⟩ ⟨ ⟩ ⟩ , welcher *alle* Achsen von Parameter, Material etc. synchron durchschneidet), sondern formale Proportionen in Instrumentation, Harmonik, melodischem Material etc. *unabhängig* zu untersuchen.

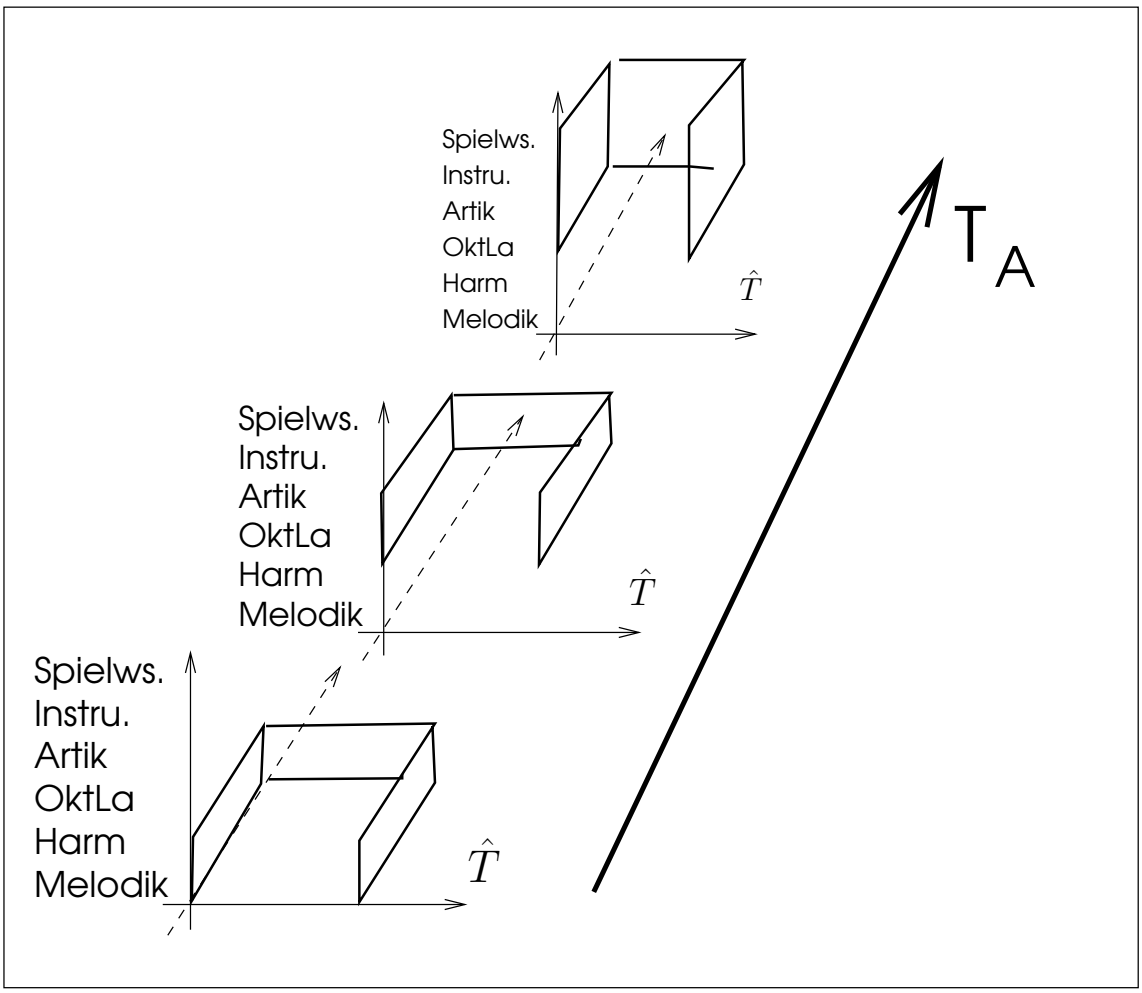


Abbildung 3.6: Werkfasern positioniert nach Arbeitsphasen.

3.6.4 Unvollständige Werkfasern.

Eine „Werkfaser" unserer Nomenklatur kann nun in den verschiedensten Ebenen unvollständig sein.

Trivialerweise können z.B. die Parameter eines anzuwendenen Kompositionsalgorithmus noch nicht alle vollständig bestimmt sein. Diese sind dabei aber meist als Einschränkungen oder Alternativen doch teilweise bestimmt. Die Offenen-Entscheidungs-Alternativen (s.u.) von internen Parametern einer Werkfaser können stets (rein formal) auf die Alternativen von Werkfasern „herausmultipliziert" werden.

Ebenso unergiebig ist die Unvollständigkeit der *Plazierung* von Werkfasern bezogen auf die Achse „Arbeitsschritt" oder die Stilabhängigen Achsen (z.B. Parameter oder Material etc.): Eine Werkfaser, welche auf Tonhöhe, Rhythmus und Klangfarbe gleichermaßen anwendbar wäre, ist halt nur in bestimmten (seriellen oder postseriellen) Stilen möglich und dortselbst hinreichend erforscht.

Spannender ist jede *Unvollständigkeit der Plazierung bezgl. der Zeitachse* : Je nach Art der Unvollständigkeit nämlich erscheinen die von uns allesamt als „Werkfasern" bezeichneten Objekte als herkömmlicherweise durchaus unterschiedlich benannte Vorstellungsgegenstände:

- Der Zeitplan einer Werkfaser ist noch gar nicht bestimmt (oder zumindest kaum instantiiert): In diesem Fall bedeutet die Werkfaser lediglich eine Verknüpfung von Materialien (=Bestimmungen) ihrer *Reihenfolge* nach. Diese Kategorie enthält damit so unterschiedliche Objekte wie reine Tonklasssenfolgen (z.B. Zwölfton-Reihen), historisch begründete sog. „Formen" („Exposition, Durchführung, Reprise"), grundsätzliche (grobe) Ablauf-Vorstellungen („Steigerung$\Longrightarrow$Zusammenbruch$\Longrightarrow$etwas kürzere Steigerung") oder Instrumentationspläne („Holz erst nach einem Drittel, Blech erst am Schluß!").

- Der Zeitplan der Werkfaser ist bis auf einen Multiplikationsfaktor bestimmt: Dies beinhaltet die klassischen Begriffe von „Thema, Motiv, Melodie", falls diese in verschiedenen Tempi oder Augmentationen auftreten sollen, - kann aber auch eine „rhythmische Formel" sein, welche im post-seriellen Sinne für alle Werk-Ebenen (mit entsprechend unterschiedlichem Mulitplikator) verwendet werden soll, — von der Generierung motivischen Materials bis zur Organisation der Großform.

- Eine Werkfaser ist in allen Bestandteilen bestimmt und auch ihr Zeitplan ist bereits fest an die Realzeit *als solche* angekoppelt, aber noch nicht innerhalb der Werkfaser des Gesamtwerkes (unser ausgezeichnetes Objekt „Gesamtform") zeitlich plaziert: Derartige Werkfasern sind also „ausgeführte Skizzen", deren letztliche Verwendung noch offen ist. Typischerweise existieren Meta-Bestimmungen, welche diese Verwendung zumindest einschränken („Diese Steigerung s_1 kann in Teil D oder am Übergang von G nach H auftreten; wenn aber s_2 verwendet wird, kann sie gar nicht benutzt werden.").

Daß Werkfasern i.A. während der Entwicklung einer Komposition größtenteils lange Zeit *unterdeterminiert* sind, bedeutet allerdings ein zentrales Problem für eine evtl. rechnergestützte Implementierung:

Besonders die *Zuordnung* von Bestimmungen (Generatoralgorithmen) zu Sub-Knoten eines Zeitplanes muß generell *generisch formuliert* sein, da ja z.B. die Struktur des (bei vollständiger Parametrisierung) letztlich entstehenden Zeit-Baumes nicht bekannt ist: z.B. kann die Anzahl der Sub-Knoten eines gegebenen Knotens eines gegeben Zeitplanes variabel sein oder sogar 0(null) werden.

3.6.5 Übervollständige Werkfasern.

Eine *Vereinfachung* für die (im Interesse der Freiheit der Kreativität) möglichst freie Verschiebbarkeit unserer „building blocks" ergibt sich aus der Erlaubnis einer gewissen *Überdefinition*.

Wir können so allgemein gehaltenen und vielseitig verwendbare Formvorstellungen definieren und diese in einem Achsenbereich (Kontext) verwenden, in welchem lediglich *Teile* der generierten Daten sinnvoll sind, - andere aber (ohne zu schaden) sinnlos werden.

Sei z.B. g_0 ein Generator, welcher gleichzeitig Tonhöhenfolgen, Akkordstrukturen, Rhythmen (=Einsatzzeitpunkt $\times$ Tondauer) und Instrumentationsformeln generiert.

Wird unser g_0 zur Generierung einer Pfte-Stimme verwendet, so werden alle generierten Ausgangsfunktionen verwendet, – bis auf die letzte, welche „automatisch vergessen" wird. Bei der Anwendung auf ein Ensemble perkussiver Geräuschinstrumente hingegen würden nur Einsatzzeitpunkt und Instrumentationsformel sich auswirken.

Dieser „implizite forget-funktor" wird, da selbst-dokumentierend, die Formulierungsarbeit des Komponisten auf der Materialebene u.U. erheblich entlasten, – bei einer späteren rechnergestützten Implementierung jedoch erkauft werden müssen durch eine gewisse Aufweichung des überprüfbaren Typsystems.

3.6.6 Einfachere Vernküpfungen.

Eine Werkfaser entsteht also durch Verküpfung eines (u.U. generischen) Zeitplanes mit diversen Bestimmungen. Dies stellt die komplexeste Verknüpfungsart in unserem Modell dar.

Glücklicherweise sind alle übrigen Verknüpfungsarten nicht so komplex und nun in aller Kürze vorgestellt.

Zu beachten ist allerdings, daß diese Verknüpfungen nur von ihrem Prinzip her einfach sind, – ihr struktureller Gehalt bleibt einfach, so lange sie sich auf die ganz zu Beginn gesammelten Einfachen Bestimmungen beziehen.

Sie können sich allerdings auch auf Meta-Bestimmungen sowie auf beliebig zusammengesetzte, d.h. ihrerseits durch Verknüpfung entstandene Bestimmungen beziehen, – also auch auf ganze Werkfasern, so daß die durch sie beschriebene Struktur beliebig komplex werden kann.

3.6.6.1 Die Anfüllung.

Eine der einfachsten Grundverknüpfungen zweier Werkfasern oder Zeitpläne F_{sup} und F_{sub} ist die der „Anfüllung".

Dies bedeutet zunächst und hauptsächlich, daß während der Dauer eines Knotens des Zeitplanes der übergeordneten Werkfaser F_{sup} die klingende Realzeit „erfüllt" wird von dem (klingenden) Evaluierungsergebnis der untergeordneten Werkfaser F_{sub}. Aus der Sicht der untergeordneten Werkfaser entspricht diese Verknüpfungsart der „relativen Plazierung bezgl. der Zeitachse", und könnte als „Einfügung" oder „Einbettung" bezeichnet werden.

Die konventionelle (und innerhalb unserer Modellbildung als etwas oberflächlich zu bezeichnende) Redeweise lautet z.B. „Die Exposition *besteht* aus dem Hauptthema, der Überleitung, dem Seitensatz, der Schlußgruppe und einem Epilog".

Tatsächlich aber ist nicht nur die Zeitachsen-Plazierung wirksam. Vielmehr ist die Anfüllung/Einbettung immer verbunden mit einer Parametrisierung/Instantiierung: Bestimmungen aus F_{sup} fließen ein in die konkrete Gestalt der untergeordneten Werkfaser F_{sub}.

(Konventionelles Beispiel: Die Tonart der Exposition des Seitensatzes ist Funktion der Tonart der Exposition ist Funktion der Tonart des Satzes ist Funktion der Tonart des Gesamtwerkes.)

Im Zusammenspiel zwischen Anfüllung/Einbettung und Parametrisierung/Applikation findet die eigentliche Bestimmungsarbeit an der Architektur jeden Werkes statt.

3.6.6.2 Die Offene-Entscheidungs-Alternative.

Eine ebenfalls einfache Grundverknüpfungen zweier (beliebig komplexer) Bestimmungen ist die der „Offenen Entscheidungs-Alternative"(= OEA). Diese wird allerdings in konventionellen Modellen ausschließlich einer Meta-Ebene zugerechnet und somit *nicht als Material* betrachtet, – welches einer rechnerunterstützen Realisierung

jedoch mitnichten angemessen ist.

Für viele Bestimmungen / Werkfaser-Entscheidungen findet der Autor während des Arbeitsvorganges mehrere mögliche Lösungen, welche zunächst nicht entschieden, sondern als Alternativen festgehalten werden. Diese werden weiter unten bei der Betrachtung des *Versions-Problem*s eine zentrale Rolle spielen.

3.6.6.3 Logische Verküpfung, z.B. Wenn-Dann-Vernküpfung.

Verknüpfungen des logischen Typs treten in durchaus unterschiedlicher Gestalt, unterschiedlichem Stellenwert und in verschiedenen (Meta-)Ebenen der psychointernen Architektur auf:

Eine Wenn-Dann-Verknüpfung der Gestalt „Wenn Material b_x zur Anwendung kommt, soll gleichzeitig Material b_y gelten", schafft eigentlich eine *neue* Bestimmung $(b_x \wedge b_y)$, und stellt gleichzeitig die Regel (Meta-Bestimmung) auf, daß nur diese neue Bestimmung anwendbar ist und keinesfalls mehr b_x alleine! (z.B. „Wenn das Werk eine ungradzahlige Anzahl von Sätzen hat, dann ist der Mittelsatz zentralsymmetrisch.")

Andererseits sind durchaus logische Verknüpfungen von Bestimmungen möglich, welche als eigenständige Materialbausteine anzusehen sind:

Sei ein Zeitplan dreiteilig, dann könnte die Bestimmung „Jede Instrumentengruppe spielt ausschließlich in einem der Teile, und zwar alleine" von ihrem rein logischen Gehalt her durchaus als z.B. Disjunktion von Konjunktion notiert werden, oder als Konjunktionen von Implikationen etc.

Im *Denken* des Komponisten jedoch sind derartige *Verteilungs-Vorschriften* hingegen ganzheitliche, atomare Makro-Begriffe sind.

Viele Verknüpfungen des logischen Typs gehören zu den Meta-Bestimmungen (z.B. „Wenn ich Thema *a* verwende, kann ich leider Thema *b* nicht benutzen, weil sie zu ähnlich sind").

Andere sollen global für ein gesamtes Werk gelten, d.h. sind bzgl. der Zeitachse der Werkfaser Gesamtform zugeordnet („Immer, wenn Trompeten blasen, ist die Pauke auch nicht still.").

Die *Art der zur Anwendung kommenden Logik* ist dabei in keiner Weise eingeschränkt, – gerade solange wir psycho-interne „Implementierungen" kompositorischen Denkens betrachten, – wo ja manche Entscheidung „aus dem Bauch" getroffen wird –, können durchaus ternäre oder gewichtete Logiken eine adäquae Modellierung des vor-bewußten Kalküls darstellen.

Die Wahl der verwendeten Logik-Systems beeinflußt jedoch die Architektur unseres Rahmenwerkes glücklicherweise mitnichten

3.6.6.4 Obwohl-Verknüpfung.

Die „Obwohl-Verknüpfung" ist eine ausnahmsweise auftretende Vernküpfungsart und eng mit dem unten beschriebenen „licenca-Problem" verwandt (s.u. 5.7.1).

Sie zeigt deutlich die Grenzen (des dem Bewußtein zugänglichen Teiles) unseres Modelles.

Grundsätzlich beschreibt sie *Ausnahmen*, z.B. Zeitpunkte im Gesamtablauf, zu denen, entgegen der im umgebenden Formteil generell geltenden Regeln, *ausnahmsweise* ein bestimmtes *abweichendes* Verhalten stattfinden soll.

Derartige *Singularitäten* werden häufig begründet durch ein übergeordnetes (außenstehendes) semantisches Konzept, welches sowohl nicht immer verbalisierbar ist, und darüber hinaus häufig nur mit unwirtschaftlich hohem Aufwand formalisierbar oder auch nur formulierbar wäre.

Zum Beispiel : „Das Orchester BRUCKNERs enthält grundsätzlich kein Schlagzeug, – jedoch im Adagio der Siebenten Sinfonie erfolgt ein Beckenschlag"[8].

Oder aber:

- Der Summenrhythmus von Contrapunctus IX besteht ab Takt 13 aus durchlaufenden Achteln.
- Ausnahme: Das angebundene Viertel im Kontrasubjekt wird niemals von Achteln begleitet, ...
- ... bis auf (Ausnahme der Ausnahme) die Einsätze Takt 45, 73 und 89, wo doch Achtel hinzutreten.
- Die wird durch die Ausnahme kompensiert, daß die Einsätze in Takt 35 und 45 ein anderes Viertel des Kontrasubjektes (das verzierte Vorhaltsviertel) zusätzlich von Achtelanschlägen freihalten!

[8]Der Sage nach hatte BRUCKNER gerade diese Stelle der Partiturreinschrift erreicht, als ihn die Nachricht vom Tode WAGNERs erreichte.

- Ergebnis: Löcher im Summenrhythmus in Takten 23, 36, (39!), 49, 80, 100, 120.

3.6.6.5 Pipelining-Verknüpfung, Parametrisierung und Applikation.

Pipelining, Parametrisierung und Applikation sind verschiedene *Sichtweisen* auf die selbe Art der Vernküpfung :

Die Ergebnisse der Auswertung der einen Bestimmung werden zu Eingangsdaten der zweiten, – oder : Die zweite Bestimmung wird (als Algorithmus) angewendet auf die Ausgangsdaten der ersten, – oder : die zweite Bestimmung wird parametrisiert instantiiert, wobei die Parameter die Ausgangsdaten der ersten Bestimmung sind.

Je nach „Gewicht" und Art der verknüpften Bestimmungen scheint die eine oder andere Redeweise natürlicher.

Rein technisch läßt sich diese Verknüpfung allemal abbilden auf ein *Datenflußmodell.Datenflussmodell=Datenflußmodell.*

Kompliziert wird dieses Modell dadurch, daß nicht nur „eins zu eins"-Datenflüsse, sondern auch „eins zu n", „m zu eins"- und „m zu n"-Verknüpfungen in fast beliebiger Kombination möglich sind.

Darüber hinaus sind *zyklische* Datenflüsse durchaus erlaubt. Meistens liegt auf einem der „rückwärtsweisenden" Pfeile des Datenflußgraphen als „Zwischenspeicher" eine diachrone Darstellung der Daten (s.u. „Synchrone Simulation" in Abschnitt 4.2), also ein Resultatobjekt eines Evaluationslaufes, was dieses Problem einerseits handhabbar macht, andererseits die inhärente Problematik der „Versionsverwaltung" (s.u.) zusätzlich verschärft.

3.6.7 Attributeinschränkungen als unvollständige Attributierung.

Ein sehr häufiger Fall von unvollständiger Attributierung ist die *Attributeinschränkung*.

Aussagen wie „Dieses Motiv soll irgendwo in der Reprise auftreten" oder „Diese harmonische Folge kann evtl. in E-Dur oder einer höheren Tonart auftreten" sind Meta-Bestimmungen, als sie die Menge der möglichen Belegungen (im ersten Beispiel: die Zeiträume der möglichen Plazierung) definitiv einengen, ohne schon endgültige Parameterwerte festzulegen.

Diese „fuzzy"-Festlegungen sind in gewissen Kompositionstechniken fast der Regelfall!

Technisch gesehen kann man diese Meta-Bestimmungen auf die Ebene der einfachen Bestimmungen hinabfalten, indem qua constraint-Formulierung (qua algebraischer Formulierung) automatisch Sub-Datentypen generiert werden.

3.7 Auswerten des Netzwerkes.

Ein Auswertungsvorgang ist dem Begriffe nach eine reine Datentransformation: Implizit durch die im Bestimmungsnetzwerk gegebenen Regeln definierte Inhalte werden expliziert.

Häufig wird eine out-of-time-Darstellung in eine in-time-Darstellung umgewandelt, aber auch alle anderen drei Kombinationsfälle sind möglich[9].

Hauptzweck von Evaluierungsläufen ist i.A. die bewußt gesetzten Bestimmungen zusammenzubringen mit den durch die *Eigengesetze des Materials implizit* gegebenen Regeln.

Ein *tiefer* Ton hat z.B. auf dem Pianoforte bedeutende Einschränkungen bzgl. der *Mindest*dauer, ein *hoher* Ton hingegen bezgl. der möglichen *Längst*dauer. Sehr hübsche post-serielle Materialkonzepte können an solch schnöden Bedingungen der Physik bei dem naiven Versuch der unmittelbaren Umsetzung in einem ersten Schritt durchaus scheitern.

3.7.1 Auswertung Offener Alternativen, Versionen.

Trifft ein Auswertungsprozeß auf eine noch „Offene Entscheidungs-Alternative", so sind prinzipiell zwei Verhaltensweisen praktikabel:

- Die Entscheidung kann nun getroffen werden, die nicht-gewählten Alternativbestimmungen werden aus dem Netzwerk eliminiert.
- Die Entscheidung wird nicht getroffen, vielmehr wird für jede Alternative ein Evaluierungslauf durchgeführt. Es entstehen so mehrere *Varianten* der Expliziten Darstellung, deren *Rezeption* dem Autor evtl. als Entscheidungsgrundlage zwischen den Offenen Alternativen dienen kann.

So entstehen die „Versionen diachroner Darstellungen", welche eine eigene Behandlungsproblematik innerhalb des Rechners mit sich bringen.

3.8 Revidieren des Netzwerkes.

Die Geschichte der Entstehung eines Kunstwerkes ist in unserem Modell die inkrementelle *Veränderung* des Bestimmungsnetzwerkes im Verlaufe der Arbeitszeit. Dadurch wird (zunächst unabhängig von „Zeit" auf der Materialebene) eine weitere diachrone Komponente dem Gesamtmodell hinzugefügt.

Normalerweise als *Reaktion auf die Rezeption* des Ergebnisses eines Auswertungslaufes verändert der Autor sein Netzwerk. Die Arbeit an einem Kunstwerk ist ja auch immer ein Erkenntnisprozeß (Wittekopf).

Das individuelle Schicksal einzelner Bestimmungen kann dabei sehr unterschiedlich sein. Wir versuchen hier, einige Grundtypen aufzulisten.

[9] wobei die Konvertierung von diachronen Darstellungen in out-of-time-Darstellungen ein genuin analytischer Schritt ist und so verhältnismäßig selten.

Ein Idealfall besteht dann, wenn eine „noch offene Entscheidungsalternative" quasi organisch sich selbst erledigt, – wenn also der (dank des Auswertungslaufes nun deutlicher sichtbare Kontext) von den möglichen nun Alternativen *eine* eindeutig bevorzugt.

Der Gegenfall ist der, daß eine Entscheidung nun zum Forsetzten der Auswertungen notwendig wird (also für die nächsten Arbeitsschritte zwingende Voraussetzung ist) und der Autor durch nicht-ästhetische, externe Motivation zum Fällen der Entscheidung gezwungen ist, – das Werk soll ja fertig werden!

Hier kann ein Großteil der beim künstlerischen Schaffen auftretenden *psychischen Belastung* lokalisiert werden: Jede Entscheidung für die eine Variante ist ja Wegwerfen der anderen, – ist Verlust an Möglichkeit und Negierung eines Arbeitsergebnisses.

Nicht nur entsprechend determinierte Persönlichkeitskomponenten leisten hier Widerstand (Festhalten-Wollen, Veränderungsfurcht, Verlustangst), vielmehr erkennen wir transzendentalanalytisch einen tatsächlich gegebenen Konflikt: Die ganz zu Beginn von uns behauptete *zunehmende Annäherung* des (zunehmend stark materialisierten Netzwerk-)Modelles an das Modell der ursprüglichen (nicht materialisierbaren, rein psycho-internen) Ausgangsvorstellung ist nämlich nur die eine Richtung der Entwicklung.

In Wirklichkeit *verändert* sich nämlich unsere Grundvorstellung des zu errichtenden Werkes immerfort, – die Ausgangsvorstellung wird einerseits deutlicher ausgeführt, andererseits *als solche*, nämlich als ganzheitlicher Begriff eines imaginären, aber vollständigen Werkes, *entfernt* sie sich zunehmend vom Bewußtsein und vom Körpergefühl und wird – Eindruck flüchtigst möglicher Natur der sie ist – zunehmend von den Eindrücken des momentan konkret bearbeiteten Teilproblems verdeckt und geht letztlich *als solche* verloren.

Ein ehemals unerfüllter Wunsch, der nun erfüllt ist, kann nie wieder unerfüllter Wunsch werden.

Aus dieser transzendental-formalen Notwendigkeit folgt ein großer Teil von künstlerischer Unsicherheit, da der Autor niemals wirklich sicher sein kann, daß die Entscheidung, die er nun treffen muß, ihn in Richtung auf seine (Ausgangs- oder) Grundvorstellung führt und nicht weg davon, – und kein Mensch auf der Welt kann ihn dabei unterstützen, da diese Grundvorstellung ja eine psycho-interne ist und durch die ewige transzendentale Grenze von aller Mitteilbarkeit ausgeschlossen, so lange sie nicht zum Werk geronnen ist, wofür aber (circulus vituosus) jene Entscheidung zu fällen ja gerade notwendig ist.

3.8.1 Kreativer Konflikt.

Ein „Materialgenerierender Konflikt" ist glücklicherweise auch nicht so selten: Zwei Bestimmungen, welche sich gegenseitig ausschließen, aber aus zwingenden Gründen dennoch beide gelten sollen, zwingen zu kreativen Lösungen. Wir behaupten, daß derartige Konflikte normalerweise einen sehr großen Teil zur *innersten Substanz* eines Werkes beitragen. Sie sind in doppeltem Sinne des Wortes „Kreative Konflikte".

Nicht so sehr durch Modifikation der konflizierenden Einzelbestimmungen, sondern vielmehr durch deren dialektische Aufhebung werden so Lösungen gefunden, die zwingend logisch und gleichzeitig überraschend und innovativ sein können. Dies gilt sowohl für konkrete Einzelprobleme als auch für übergeordnete historische Tendenzen:

- Die Notwendigkeit, mit einem Bläserquintett einen Sechs-Klang darzustellen, fordert weitergehende analytische Aufarbeitung der tatsächlich gemeinten Harmonik.

- Drei melodische Materialien gleichzeitig in zwei melodischen Linien darzustellen, führt zu melodischen Neubildungen durch Anwendung der Technik der internen Zweistimmigkeit.

- Der Wunsch (die „künstlerische Notwendigkeit"), auf einem einstimmigen Instrument Harmonik auszudrücken, führte zur Entwicklung des ganzen Reichtums der *Partitentechnik*.

- Der Wunsch (die „künstlerische Notwendigkeit"), auf dem Pianoforte dem neuentdeckten interpretierendem Subjekt den ganzen Klangreichtum des großen Orchesters (anscheinend) zur Verfügung zu stellen, führte zu neuen Spiel- und Satztechniken (Sprünge, Passagen etc.) und zu fortschreitender Abstraktionstendenz im Material (Allusionstechniken).

3.8.2 Ersatzlose Streichung und Ersetzung von Bestimmungen.

Die ersatzlose Streichung einer Bestimmung (Idee) aus dem Zusammenhang des entstehenden Werkes ist paradoxerweise ein seltener Idealfall.

Dasjenige Kunstwerk ist das wertvollste, welches die wenigsten Teilbestimmungen beinhaltet: Das Finale z.B. des cis-moll-Quartettes von Beethoven, über die ersten beiden Akkorde, die Themenbildung, Durchführungsmuster bis hin zur Großform, kann vollständig begriffen werden als die einfache, organische, zwangsläufige Auskomponierung des allerersten Impulses.

Deshalb anerziehe sich jeder den Mut zum Wegwerfen.

Tröstlicherweise geschieht es nicht selten, daß ein ausgesondertes Material zur Keimzelle eines ganz neuen Werkes wird, – ein netter kleiner Kontrapunkt, der leider keine Verwendung fand, kann ganze Sinfonien hervor-rufen.

3.8.3 Ersetzung durch Konkretisierung, Parameterzuwachs.

Der Normalfall einer Netzwerkrevision besteht in der weiteren Einschränkung der noch offenen oder noch gar nicht betrachteten Bestimmungen.

Häufig aufgrund der Rezeption der Ergebnisse von Evaluierungsläufen mit verschiedenen Parametersätzen („Versionen", siehe oben) kommt der Komponist zu einer engeren oder endgültigen Festlegung von Parametersätzen oder Algorithmen.

Dies entspricht einem Übergang vom reinen „Plan" zu einer konkreteren Stufe des Zwischenproduktes hin zum endgültigen Werk.

3.8.4 Weitergehende Abstraktion.

Jedoch auch der gegenteilige Fall ist durchaus nicht selten:

Ein zunächst definitiv gemeintes Stück Material, welches sich „wörtlich" doch nicht verwenden läßt, wird weiter abstrahiert zu einer Regel und kann als solche dann doch verwendet werden.

Die Akkordfolge ⟨ C-Dur, g-Moll ⟩ kann z.B. vom Autor irgendwann als Idee notiert worden sein, und wird nun in die abstrakte Folge ⟨ T, d ⟩ übersetzt, – deren letztliche Verwendung normalerweise als „*transponierte* Verwendung der Ausgangsidee" bezeichnet würde.

Oder aber ein seinerzeit als konkret notierter Rhythmus wird als ein Spezialfall eines rhythmischen Prinzips erkannt (z.B. als Überlagerung von sich verschiebenden Patterns zwecks zunehmender Verdichtung), und dieses isolierte Prinzip wird, anders parametrisiert und resultierend in ganz andere Oberflächengestalt, anstelle der (nicht-abstrahierten) Ausgangsidee weiter verwendet.

3.9 Entscheidungsgraph und -protokoll.

Die *oberste* Strukturebene eines kompositorischen Arbeitsprozesses ist also die Folge der getroffenen Entscheidungen.

Diese kann im nachhinein auf zweierlei Weise strukturiert werden:

Einmal empirisch-historisch : Wann hat der Komponist welche Entscheidung getroffen, - ergänzt durch: wann hat er welchen Evaluationslauf gestartet und rezipiert ?

Diese beiden Informationen ergeben eine diachrone Darstellung des Kompositionsprozesses.

Zum anderen ist eine rein logische Gliederung möglich: Welche Entscheidungen haben welche anderen zur Voraussetzung ?

Bedauerlich für Musikwissenschaft und Kompositionslehre ist, daß ohne Benutzung einer Integrierenden Digitalen Arbeitsumgebung in der Praxis derartige Untersuchungen und Protokollierungen immer nur annäherungsweise durchführbar sind.

3.10 Musikalische Analyse.

Nach dieser ganzen Vorarbeit können wir – als Abfallprodukt – auch endlich mal eine griffige und knappe Begriffsbestimmung von „musikalischer Analyse" geben[10], indem wir alle bisher aufgelisteten Gestaltungsmöglichkeiten (wie angekündigt) als „Mittelgrund-Operation" zusammenfassen und das „vorliegende" Werk (stark vereinfachend) dem Vordergrund zurechnen :

Musikalische Analyse ist die Umkehrung der Auswertungsläufe im Mittelgrund des Schaffensprozesses.

Genauer:

Musikalische Analyse ist der *Rückschluß* von der vorliegenden Vordergrundstruktur auf einen (möglichen, besser noch: wahrscheinlichen) Mittelgrund, der diese Struktur hervorgebracht hat, – nebst einer Beschreibung von dessen Entstehen aus dem (bekannten oder vermuteten) Hintergrund.

[10] Welche mit Sicherheit keine allgemeine Verwendbarkeit und schlechthinnige Gültigkeit fordern kann.

Kapitel 4

Praedigitale Materialisationsformen kompositorischer Denkinhalte.

Im vorigen Kapitel wurde ein Modell der psycho-internen Modellbildung beim Vorgang des Komponierens aufgestellt.

Diese inneren Denkvorgänge bedienen sich, wie jedes menschliche Denken, in unterschiedlichem Maße *physischer Repräsentationen* der internen Objekte, als Materialisierungen der Einzel-Bestimmungen, der Werkfasern und ab und zu sogar der Meta-Bestimmungen.

Auch das *Endergebnis* des künstlerischens Schaffensprozesses bedarf stets einer Materiellen Gestalt, um überhaupt mitteilbar zu sein. Diese kann allerdings so unterschiedlicher Natur sein wie die Luftschwingungen, die bei einer spontanen Improvisation des Ohr des Hörers erreichen, einerseits und eine ausgefeilte Orchesterpartitur (Schrift auf Papier) andererseits.

Die inneren Eigengesetzmäßigkeiten dieser Medien, die strukturellen als auch die historischen, haben nun entscheidenden Einfluß auf Kommunizierbarkeit, ästhetisches Material, Arbeitsweise und Gehalt.

Ihre Analyse ist Gegenstand dieses Kapitels, – diese Eigengesetze sollen als benennbare und kalkulierbare Bestandteile in unsere Modellbildung einfließen, haben sie doch in der Praxis (natürlicherweise) allemal die Tendenz, unbemerkt das künstlerische Schaffen und das ästhetische Produkt zu beeinflussen.

4.1 Das Memo.

Die Organisation des Arbeitsprozesses im Materiellen folgt zunächst einmal der Struktur des inneren Denkens.

Die Entstehung des Entscheidungsnetzwerkes bedeutet im Materiellen (bei vielen Komponisten) die Entstehung von unzähligen Einzelnotizen, welche die o.e. „Bestimmungen" auf dem Papier fixieren.

Dabei wird schon ein entscheidender transzendentaler Schritt vollzogen. Während das eigentliche Vorstellungsnetzwerk ausschließlich in der Inneren Sprache existiert, bedeutet der Versuch der Fixierung auf dem Papier schon eine *Übersetzung* in eine andere Sprache.

Diese Sprache jedoch ist immer noch eine persönliche, d.h. die Übersetzungsregeln sind i.A. weder sinnvoll formalisierbar noch notwendigerweise überpersönlich-historisch begründet.

Vom „mittleren BEETHOVEN" wissen wir z.B., daß die ersten Skizzen zu einigen Werken nur aus wilden Tintenstrichen bestehen, welche einen Strukturverlauf als solchen grob andeuten, ohne sich überhaupt der Sprache der Tonhöhen oder Rhythmen zu bedienen.

Wir als Außenstehende können aus diesen Skizzen lediglich Allgemeinstes entnehmen, wie: „Hier geht die Klangschicht A aufwärts und wird dabei immer schneller, während Klangschicht B ganz tief nach der Hälfte einsetzt".

Für den Autor solcher Notizen jedoch ist der Informationsgehalt wesentlich höher.

Diachrone Notizen (= solche, die eine Zeitverlauf darstellen sollen) sind nämlich (im Falle der Musik bestimmt, aber nicht nur dort) zumeist Hilfsmittel, einen bestimmten *dynamischen* Denkvorgang als solchen *wiederholen* zu können. Sie dienen als Trigger, um das, was ich gestern so überzeugend mir habe vorstellen können, noch einmal und mehrfach wieder vor meiner Inneren Vorstellung ablaufen lassen zu können, – sie sind Auslöser für das bewußte Wieder-Erleben des einmal Erlebten.

Insofern ist die in ihnen waltende Sprache der Inneren Sprache sehr nahe, ist determiniert durch den momentanen Gesamtzustand meines ästhetischen Denkens, und dieser definiert den Vorgang der Rückübersetzung: Vom statischen Bild der Tinte auf dem Papier in den dynamischen inneren Vorstellungsablauf.

4.2 Evaluierung des Netzwerkes und rückwirkende Modifikation.

Die ästhetische Produktion, welche dieses Netzwerk bis zu einem bestimmten Punkt aufgebaut hat, kann dann in ihre zweite Phase übergehen, die *Auswertung* dieses Netzwerkes.

Diese Auswertung des Netzwerkes fällt unter unseren Begriff der *Datentransformation*. Eine implizit gegebene Strukturinformation wird durch ihre Auswertung zunächst (theoretisch!) in keiner Weise vermehrt, sondern „nur" in einen anderen Aggregatzustand („Seinsform") überführt.

NB ist auch der umgekehrte Vorgang nicht selten : Eine ganzheitlich erfundene Struktur (etwas implizit gegebenes) wird a posteriori analysiert und in eine explizite Darstellung der Tiefenstruktur übersetzt. So etwas ist der Fall z.B. bei den romantischen Themen, d.h. bei den quasi melodiös empfundenen Sätzen, die dann in ihrer Durchführung einer schrittweisen De-Komposition und Analyse unterworfen werden.

Vermehrt wird das Netzwerk, die innere Vorstellung selbst, allerdings in dem Maße, wie die Auswertung zur Folge hat, daß der Autor bis dato offene Entscheidungen jetzt doch fällt oder fällen *muß*.

Diese Darstellung ist selbstverständlich idealisiert. Der tatsächliche Schaffensprozeß wird kompliziert dadurch, daß ...

1. während der Auswertungsphase sich die Definition des Netzwerkes ändern kann,

2. zum selben Arbeitszeitpunkt verschiedene Teile oder Schichten des Werkes sich in verschiedenen Phasen befinden,

3. und somit die Arbeitstechnik zwischen Auswerten und Definieren des Bestimmungsnetzwerkes pendeln kann.

In der Tat ist der Fall, daß das eigene Bestimmungsnetzwerk vom Komponisten „sklavisch" umgesetzt wird, ohne es zu modifizieren, eher die Ausnahme, - eingrenzbar entweder auf Substrukturen, für die ein Algorithmus streng angewandt werden soll, oder auf Testläufe zur Überprüfung des Netzwerkes, oder auf bestimmte Kompositions- oder Werkstile oder rein pragmatisch arbeitsökonomisch begründet.

Meistens jedoch ist die Auswertung eines Bestimmungsnetzwerkes ein *Erkenntnisprozeß* (Wittekopf), was ja die Arbeit des Komponierens erst so spannend macht.

Die so zusätzliche entstehenden oder modifizierten oder getilgten Bestimmungen brauchen dabei nicht unbedingt dem Ausführenden bewußt zu sein, und werden auch nicht unbedingt auf die materielle Ebene der Skizzen und Memos zurückübertragen (so daß das wirksame Bestimmungsnetzwerk sich von seiner Materialisierung leider zunehmend entfernt).

„Erkenntnisprozeß" im engeren Sinne des Wortes allerdings kann ein Auswertungslauf nur sein, wenn zunehmend Bestimmungen vom Vorbewußten in das Bewußte übergehen, so daß mit jeder Komposition die Fähigkeit zur Material- und Selbstbeherrschung zunimmt.

Die Bestimmungen des ursprünglichen Netzwerkes (der Zustand vor Beginn des Auswertungslaufes) können dabei unterschieden werden in (1) einerseits die, welche bereits vorher vollständig entschieden sind, d.h. die im entstehenden Werk fraglos gültig sein sollen und berücksichtigt werden müssen, (2) diejenigen, welche durch Modifikation des Netzwerkes als obsolet gestrichen werden, und (3) in die, welche eine noch offene Entscheidung repräsentieren.

Ein Evaluationslauf ist nun der Zeitpunkt, der die letztere Gruppe von Bestimmungen wiederum differenziert nach denen, die (3a) *nun* entschieden werden müssen, und denen, die (3b) weiterhin offen bleiben können.

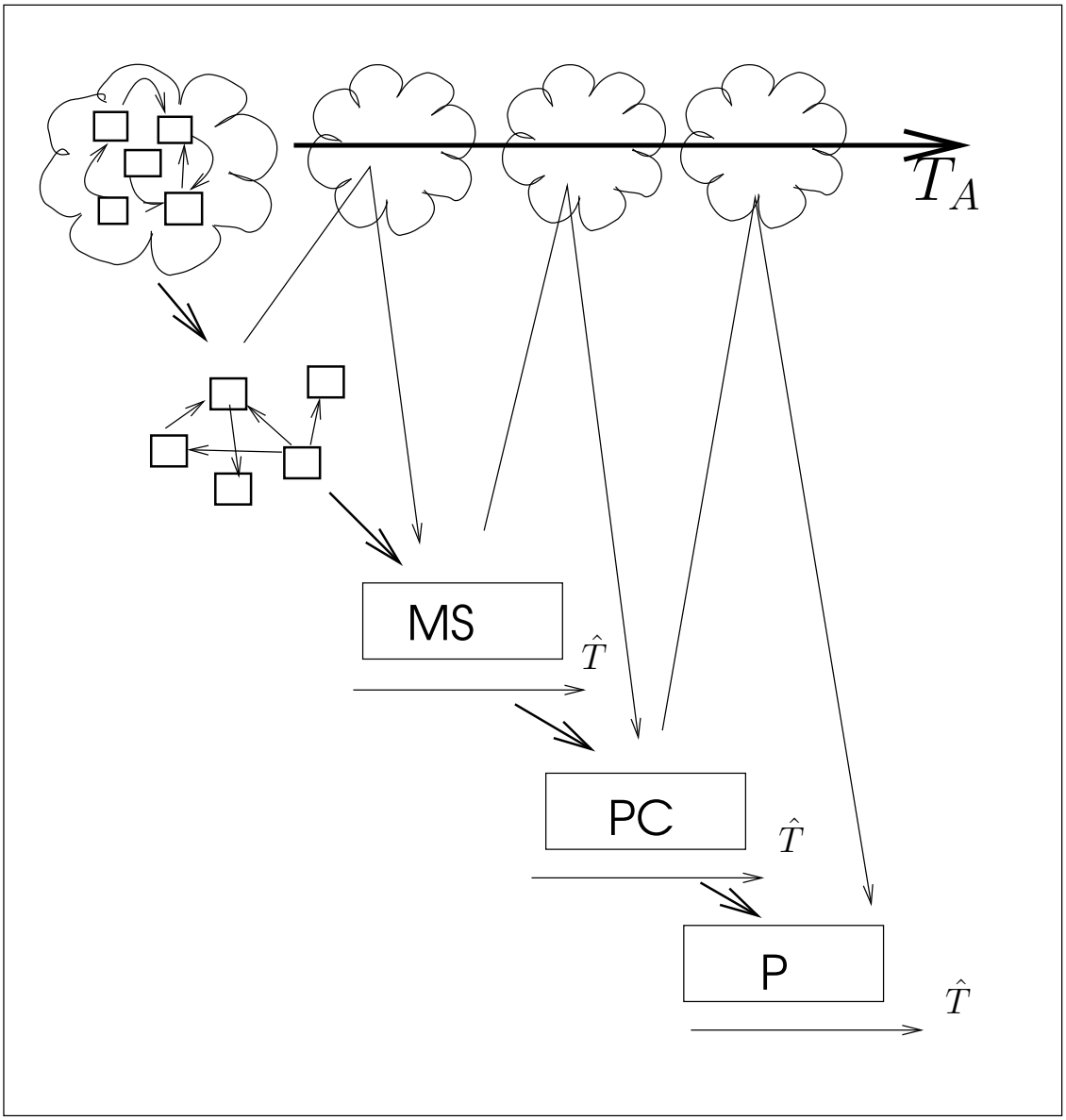


Abbildung 4.1: Idealisierte Folge von Niederschriften als Arbeitsphasen.

4.3 Synchrone Simulation.

Im Bereich der Musik ist ein Evaluationslauf normalerweise, aber keinesfalls immer, ein Arbeitsvorgang, der als Ausgabemedium eine *diachrone* Darstellung des Gemeinten hat.

Aus den vereinzelten Papierskizzen und (nicht unbedingt bewußten) Bestimmungen im Kopf des Autors entsteht z.B. im Falle der klassischen Sinfonie-Komposition als Ergebnis des ersten Auswertungslaufes ein Particell, also eine die *Zeit* scheinbar direkt repräsentierende Darstellung.

Mehrere Auswertungsläufe dieser Art werden dann kaskadiert, wobei das Ergebnis des einen Auswertungslaufes Ausgangsmaterial des folgenden wird, - jede der

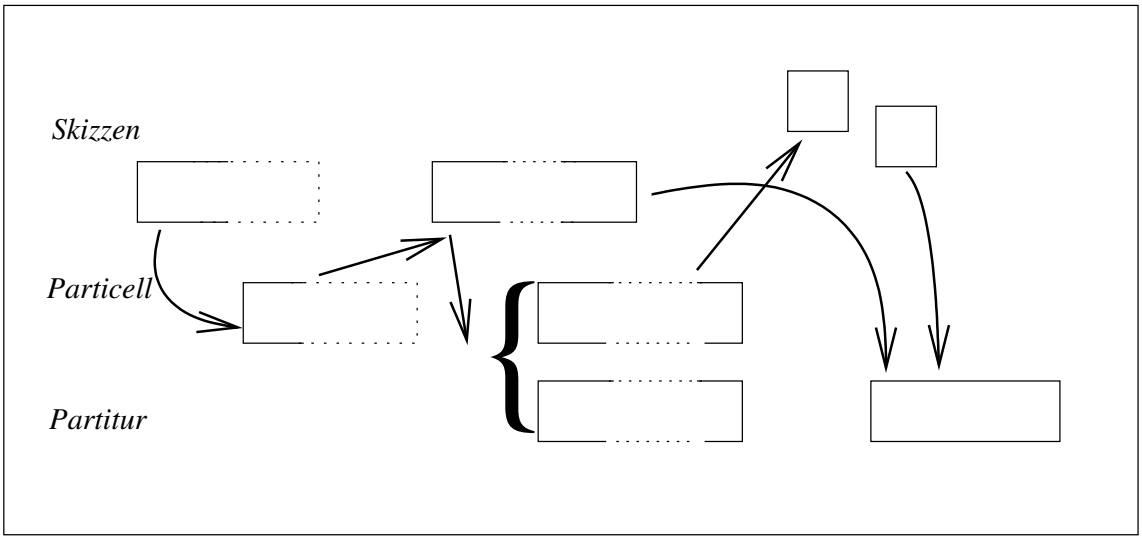


Abbildung 4.2: Tatsächliche Geschichte der Niederschriften.

entstehenden diachronen Darstellungen wird (als neue Materialisierung des modifizierten Netzwerkes) Bestandteil des intra-psychischen Gesamtnetzes.

Solche Art der Auswertung nennen wir *Synchrone Simulation*: Der Autor nimmt sich einen Forteilabschnitt des Gesamtwerkes vor und schreibt von dessen Anfang bis zu seinem Ende dem Zeitablauf folgend eine zeitgerechte Darstellung.

Eine wichtige Wirkungsweise dieser Methode besteht darin, daß durch das synchrone Aufschreiben das Unterbewußte des Autors die dem Hörer später zukommende Information in der richtigen Reihenfolge gespeichert wird, so daß notwendige Entscheidungen auf dem Hintergrund des Wissensstandes oder Stimmungszustandes ergeben, der später auch für den Rezipienten zutrifft („Ach, die zweite Trompete trägt ja noch einen Dämpfer, – na, dann gibt es hier keinen Grund, das zu ändern . . . ").

Natürlich bedeutet dieser Arbeitsvorgang weder, daß man sich währenddessen hemmungslos dem Rauschzustand des inneren Hörens überlassen darf, was eine recht niedrige Produktivität zur Folge hätte, noch daß ein Werk in allen Einzelheiten als Ablauf vom Komponisten gehört und einfach so aufgeschrieben wird. Man sieht z.B. ohne Schwierigkeit, daß die Reprise des Seitensatzes des Finales der ersten Beethoven-Sonate *vor* seiner Exposition erfunden wurde.

Der Arbeitsvorgang der Synchronen Simulation ist vielmehr ein auf die verschiedensten Weisen kombinierbarer Teilvorgang des kompositiorischen Schaffens: Horizontal montierbar, arbeitstechnisch kaskadierbar und Alternativen erprobend experimentell.

Z.B. kann man die Entstehung der Sinfonien Gustav Mahlers relativ exakt anhand der von der Musikwissenschaft „Manuskript" (= MS), „Partiturentwurf" (= PE) und „Partitur" (= P) genannten erhaltenene Niederschriften nachvollziehen. Den idealisierten Prozess zeigt Bild 4.1.

Der Verfasser hat allerdings erfahren, daß, wenn eigentlich mehrere diachrone Darstellungen im geplanten Arbeitsprozeß eine nach der anderen evaluiert werden sollten, das *tatsächliche* Verhalten des Komponierenden freier und komplexer ist (siehe Bild 4.2) : Zeitweise werden „Particell" und „Partitur" synchron vorangetrieben, – zwischendurch auf die Eben der „Skizzen" zurückgegangen, – endlich die Pflege des „Particells" ganz aufgegeben etc.

Normalerweise folgen in jedem Fall im Laufe der Arbeitszeit verschiedene (*Teil*-)Evaluationsläufe aufeinander.

Jeder Auswertungslauf verringert die Anzahl der offenen Bestimmungen und auch die Anzahl der weiteren Alternativen.

Somit sind die Evaluationsläufe (konkret, das Erstellen eines Particells oder der endgültigen Reinschrift) die Orte der künstlerischen *Entscheidungen* und somit wesentliche Gliederungs- und Ordnungsmarken der gesamten Arbeit.

Besonders bezogen auf *Teilabschnitte* des gesamten Zeitverlaufes (konkret: einzelne Formteile) sind die Evaluationsläufe auch deshalb eine Scheidemarke, weil hier *verschiedene Alternativen* gewählt und durchgespielt werden können.

Aus den so explizierten, vorher nur implizit gegebenen Varianten, kann dann eine Auswahl getroffen werden.

Hier ist eine wichtige mögliche sinnvolle Anwendung des Digitalrechners, der, qua Umschlag von Quantität in Qualität, dem ästhetischen Arbeiten wirklich neue Dimensionen öffnet, da quantitativ wesentlich schneller ausgeführte Evaluationsläufe die Quantität der ökonomisch sinnvoll produzierbaren Varianten merklich erhöhen und so neue Entscheidungsmöglichkeiten öffnen.

4.4 Notation und Mittelgrundinformation.

Jede *Notation* von Musik ist in unserer oben eingeführten Terminologie (3.3) durchaus zunächst zurechenbar dem (physikalischen) Vordergrund.

Genaueres Hinschauen zeigt jedoch bald, daß sie (1) tatsächlich vielmehr *Mittel*grund-Informationen darstellt.

Unmittelbar einleuchtend ist sofort, daß jede Notation (2) in einem Mittelgrund-Kontext überhaupt nur lesbar ist, und daß sie (3) von jedem Lesenden sofort unbewußt (beim Vorgang des Lesens) wieder in Mittelgrundinformation zurückübersetzt wird.

4.4.1 Lesen von Notation als Übersetzung in den Mittelgrund.

Ein Stück Notation wie in Bild 4.3 kann nur deshalb von einem Pianisten in beliebigem Tempo sofort vom Blatt gespielt werden, weil er ja nicht Note für Note liest, sondern sofort in einem einzigen Augenblick ein *Superzeichen* sieht.

Dieses heißt in unserem Fall „triolische Dreiklangsbrechung Passage abwärts" und ist wie alle musikalischen Superzeichen *generisch*, d.h. mit verschiedenen Werten

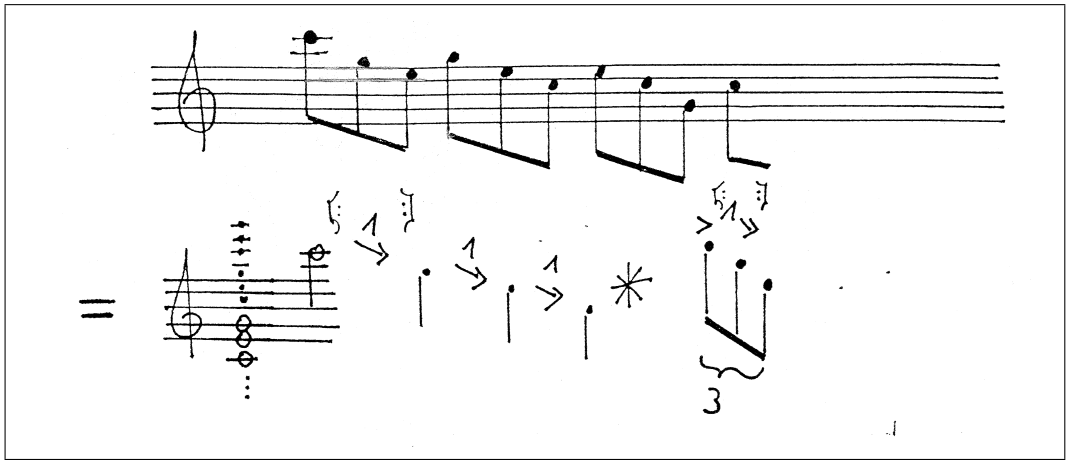


Abbildung 4.3: Notation und Superzeichen.

für die einzelnen Parameter instantiierbar.

Innerhalb weniger Mikrosekunden, für unsere Wahrnehmung „zeitlos", übersetzt das neuronale Netzwerk des Pianisten den Notentext aus Bild 4.3 in das instantiierte Superzeichen (= den parametrisierten Algorithmus) ...

Sequenzierte Dreiklangsbrechung :

- Benutzter Tastenvorrat: ausschließlich C-Dur-Töne.
- Beginn mit C'''.
- Drei Töne enge Lage je Viertel brechend abwärts.
- bei jeder Wiederholung: eine Lage tiefer.

Die gerahmt gesetzten Begriffe sind die in diesem Fall gültigen *Parametereinsetzungen* in das generische Superzeichen (= „template").

Derselbe Übersetzungsvorgang findet auch beim Notenkopieren statt: Ich blicke auf das Original, erkenne ein Superzeichen und seine Instantiierungswerte und schreibe eine neue Instantiierung in die Kopie.

Diese Wirkungsweise des Gehirns ist grundlegend und so wirkungsvoll, daß der Verfasser an den Stellen seiner Werke, an denen eine derartige Dreiklangsbrechung o.ä. mittendrin eine Abweichung aufweist ausdrücklich ein „NB" hinzuzusetzen pflegt, da einem Interpreten keinesfalls ein Vorwurf zu machen ist, wenn sie / er eine derartige, unerwartete Differenzierung überliest (cf Lepper, Weiss, Pg 17, Akk 2, Tkt 2, 2. Achtel, l.Hd).

4.4.2 Exkurs: Körpergefühl als eine von mehreren Inneren Sprachen.

Der Fall des Pianisten, welcher obiges Notenbeispiel auf der Tastatur realisiert, zeigt auch sehr schön, daß es (1) innerhalb der Psyche verschiedenen Ebenen von Inneren Sprachen gibt, zwischen denen interne Übersetzungsvorgänge ablaufen, von denen

(2) einige zweifellos vorsprachlich sind, ohne daß ihre Begrifflichkeit weniger exakt definiert wäre:

Die Vorstellung „nur Töne und Tasten des C-Dur-Dreiklanges" ist allemal ein Begriff der intellektuell-symbolischen Ebene, und zwar ein *atomarer*.

Hingegen ist die *Menge* aller physiologisch möglichen Fingerhaltungen von C-Dur-Anschlägen ein konkretes Symbol der Sprache „Körpergefühl".

Dank des jahrelangen Übungspensums mit Czerny, Hanon, Cortot et al. sind die stellungsrückmeldenden Nervenimpulse der physiologish vollkommen unterschiedlichen Hand- und Fingerstellungen, welche zu den verschiedenen Lagen eines Drei- oder Mehrklanges gehören, zu *jeweils einem* Begriff oder Superzeichen der Körpergefühlssprache zusammengefaßt worden, also zu einem einheitlichen, atomaren Objekt einer Inneren Sprache.

Der psychointerne Übersetzungsprozeß ist bemerkenswert: Während auf einer mittleren Ebene des Bewußtseins der Pianist einfach einen „einschrittigen Lagenwechsel abwärts" steuert, führen Finger und Handgelenke komplizierte und je nach Ausgangslage *unterschiedliche* Positionswechsel durch (man denke sich das Beispiel in D-Dur!), welche von der übergeordneten Bewußtseinsebene a tempo gar nicht zu realisieren wären.

Umgekehrt kann es auch sein, daß ein und demselben Objekt (Symbol) jener mittleren Ebene (C-Dur-Brechung abwärts) auf der Ebene der Körpersprache je nach hinzutretender Kontextbestimmung (z.B. ppp, leggiero vs. ff, grave) ganz unterschiedliche Anschlagstechniken und der Einsatz anderer Muskelgruppen entsprechen (leichter Fingeanschlag mit fixiertem Handgelenk vs. Gewichtsanschlag mit fixierten Fingern o.ä.)

Diese Symbolbildung auf der Ebene der Körperbefindlichkeit ist ein treffendes Beispiel für eine vorsprachliche „Innere Sprache" im Sinne unserer obigen Anforderung (vgl 3.1.2).

4.4.3 Notation als Darstellung des Mittelgrundes.

Die Komplexität der Superzeichenerkennung wird verdeutlicht durch das Beispiel in Bild 4.4, in welchem zu dem vorherigen Beispiel lediglich eine Artikulationsbezeichung hinzugefügt wurde.

Obwohl das Maß der hinzugefügten Information „physikalisch" gesehen recht gering ist, hat diese eine *völlig andersartige* Superzeichenbildung zur Folge, – die jetzt explizit geforderte *auftaktige* Gestaltung führt sofort zum Erkennen eines ganz anderen Superzeichens bei nur leicht modifizierter Notation:

Auftaktig verzierte Dreiklangsbrechung :

- Benutzter Tastenvorrat: ausschließlich C-Dur-Töne.
- Beginn mit C'''.
- je Viertel : eine Lage (= Dreiklangston) tiefer.
- jede Hauptnote (bis auf die erste) eingeleitet durch Mordent-ähnliche Umspielung (= abwärts/aufwärts-Pendelbewegung).

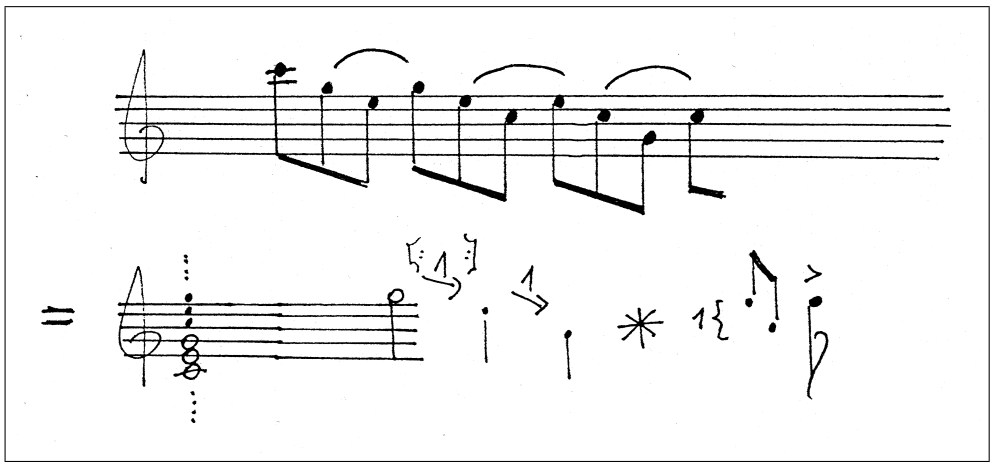


Abbildung 4.4: Leicht modifizierte Notation ergibt ganz anderes Superzeichen.

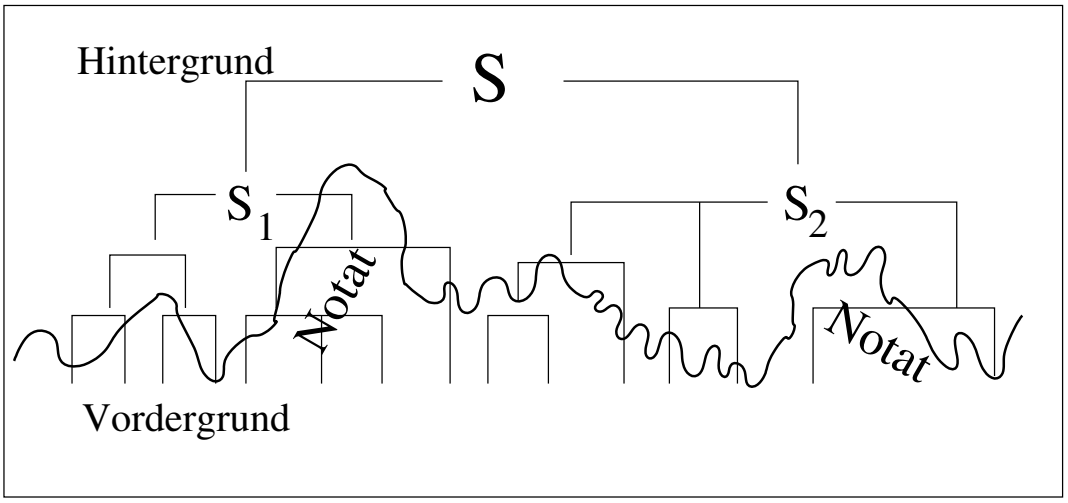


Abbildung 4.5: Notation bewegt sich im Mittelgrund.

Diese Beobachtung legt nahe, daß die Notation einen Schnitt durch den *Mittelgrund* darstellt, also im Rahmen der dynamischen „Entstehung" eines Werkes auch nur einen Übergangszustand.

Bild 4.5 zeigt einen symbolischen „Ableitungsbaum", oben die ursprünglichen Hintergrundsideen, dann die zunehmenden Konkretisierungsschritte bis hin zum Vordergrund der Blätter des Baumes, und (symbolisch) die Notation als sich durch das Geäst schlängelnder Schnitt.

Daß die Notation allemal Mittelgrundinformation enthält, wird an vielen schlagenden Beispielen deutlich: In Bachs Kunst der Fuge verdeutlicht die identische *graphische* Position der Notenköpfe bei verschiedenen Themeneinsätzen in Verbindung mit den unterschiedlichen Schlüsseln die Logik der Disposition von primaren, sekundären, tertiären, affinen und peregrinen Einsätzen ((Zacher, 1993)); - die dank den alten Schlüsseln selten notwendige Verwendung von Hilfslinien markiert auffällig satztechnische Ausnahmesituationen.

In vielen Werken findet man eingetragene *Zusatztexte*, welche nicht Bestandteil der Aufführung sind, sondern dem Interpreten Zusatzinformationen (Mittelgrund) zur Gestaltung liefern sollen (z.B. Liszt, „Wasserspiele": Zitat aus dem Johannesevangelium), oder gar die „unhörbare Melodie" bei Schumann (Humoreske op.20).

Ebenfalls Mittelgrund-Information bieten die im Jazz und in der sog. U-Musik verbreiteten Harmoniebezeichnungen: Sie gehören zum Vordergrund, soweit sie die zu greifenden und erklingenden Gitarrenakkorde etc. bezeichnen, zum Mittelgrund, als sie den gleichzeitig mit ihnen auskomponierten Notentext kommentieren, indem sie harmonieeigene von -fremden Tonhöhen differenzieren und so *gemeinte* Hierarchieen induzieren.

Immer jedoch wird Mittelgrundinformation vordergründig sichtbar bei *enharmonischer* Notation :

In der gesamten funktionalharmonischen Musik bis zur Spätromantik ist ja die zwölftönig-gleichschwebend temperierte Stimmung selbst nur Vordergrund-Medium zu einer funktionalen, d.h. eben *nicht* gleichberechtigten harmonischen Strukturierung im Mittelgrund.

Kernideen der Durchführungsgestaltung des Finales der Sturmsonate z.B. springen den Betrachter beim ersten, unscharfen Blick auf das Notenbild allein durch die Wahl der Vorzeichen förmlich an: Ein flüchtiger Blick auf die Henle-Ausgabe (II. Band, Seite 42/43) zeigt in den letzten Takten nach all den Bee's jenes häßliche, unverständliche gis, dessen Versuch zu begreifen mitten in die letzten Geheimnisse dieser Durchführung leitet.

Bei Franz Schubert in der Klaviersonate a-moll D 845 wird die enharmonische Schreibweise vom folgenden Kontext bestimmt, also von der intendierte Richtung des Klanges und der Zukunft des Vordergrundes (1. Satz Takt 21 es vs. Takt 32 dis, ebenso Takt 36 as vs. gis). Den Höhepunkt dieser Technik ist die gleichzeitige verminderte Sekunde, also ein funktional sinnvolles des und cis gleichzeitig (als Vorbereitung dazu die doppelt übermäßige Prime in Contrapunctus IV, Takt 61 ff dis vs. des, 66 as vs. gis und 78,97 b vs. ais, woraufhin das as im Übergang 84 nach 85 nicht mehr bestimmbar ist!).

Auch das als Scheinfunktion (c-dis-g) notierte c-moll in der Götterdämmerung ist Mittelgrundinformation.

Die ganze Art und Weise, in welcher Schenker schon in seinen frühen Werken *Artikulation* analysiert, zeigt u. E., daß er diese durchweg als relevante Mittelgrunds-Information (in unserer Definition) auffasst.

Die explizite Notation von Mittelgrund-Information hat durchaus den konkreten praktischen Zweck, die *Spielhaltung* des Interpreten zu beeinflussen, und damit die beim Hörer ankommende Gestalt zu modulieren. Der Verfasser geht an einer Stelle so weit, dem Pianisten vorzuschreiben, welche harmonische Funktionsfortschreitung er *innerhalb eines Vierklanges* hören/denken solle (siehe Abbildung 4.6).

Gemeint ist eine genaue Beschreibung des Anschlages, der so zu wählen ist, daß durch das Abklingverhalten der Pianoforte-Saiten zunächst die eine Funktion, dann die andere im Vordergrund erscheine.

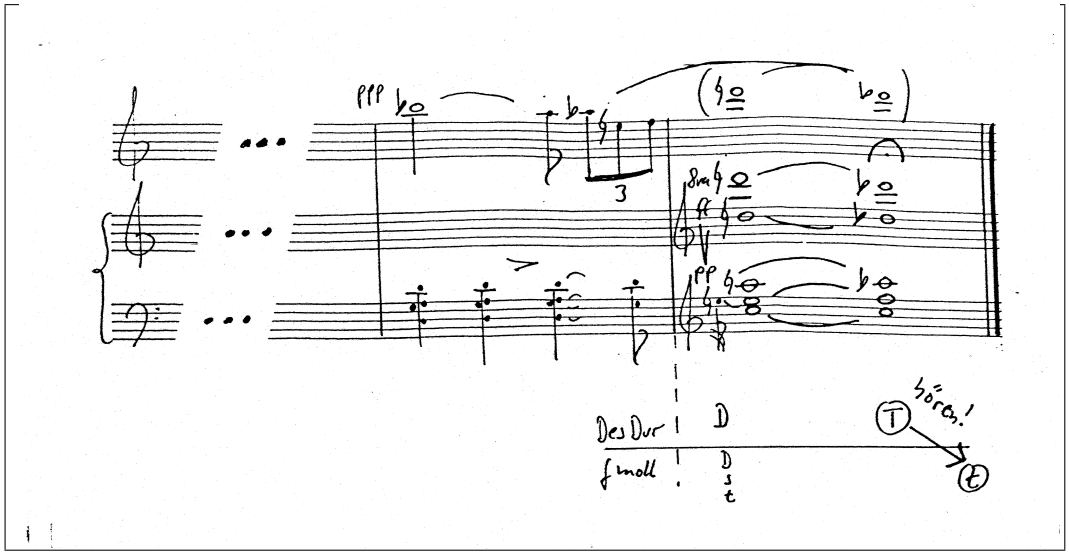


Abbildung 4.6: Expliziter Mittelgrund als Vortragsanweisung in Lepper: „Nachrufe", Vorspiel.

4.4.4 Informationsverlust beim Übergang vom Mittel- in den Vordergrund.

Natürlicherweise tritt bei der Übersetzung von Mittelgrundstrukturen in eine Vordergrundgestalt i.A. ein Informationsverlust auf, als unterschiedlichste Mittelgrundstrukturen die gleiche Vordergrundgestalt generieren können, – was als „impliziter forget-Funktor" bezeichnet werden könnte.

Ein reichlich *absurd konstruiertes*, aber dafür anschauliches Beispiel, – welches keine tatsächliche musikalische Relevanz beanspruchen will und kann ! – möge das verdeutlichen (Bild 4.7) :

In Bachs Weihnachtsoratorium stehen die Teile 1, 3 und 6 in D-Dur. Ebenso verwendet Bach in Teilen 1, 3 und 6 „Pauken und Trompeten".

Dieser einfache Befund des Vordergrundes kann nun den unterschiedlichsten Mittelgrund-Regeln folgend kompositorische zustande gekommen sein:

Setzen wir z.B. im Mittelgrund die Tonart-Disposition als (zeitlich und logisch) *vor* der Instrumentation feststehend voraus, so könnte der Einsatz der Trompeten (1) durch die Regel „Trpt genau dann, wenn D-Dur" zustande kommen. Diese „Bestimmung" wiederum könnte auf zwei verschiedene Weisen begründet sein – rein praktisch (1-a), denn „die F-Trompeten sind im Pfandhaus und von der A-Trompete sind zwei kaputt", so daß nur D-Trpt zur Verfügung stehen, – oder kompositionstheoretisch (1-b), weil „die Instrumentation als untergeordnet zur *Verdeutlichung* der tonalen Großarchitektur zu dienen" habe, – beide Gründe sind *Meta-Bestimmungen.*

In diesen Fällen wird eine „fertig ausgerechnete" *Vordergrund-Struktur*, nämlich die evaluierte, konkrete Tonartfolge, der eine (hier unbekannte) Mittelgrundregel x zugrunde liegt, in den Mittelgrund *zuückgefaltet*, um dort als Input-Material die Generierung der Instrumentation zu steuern.

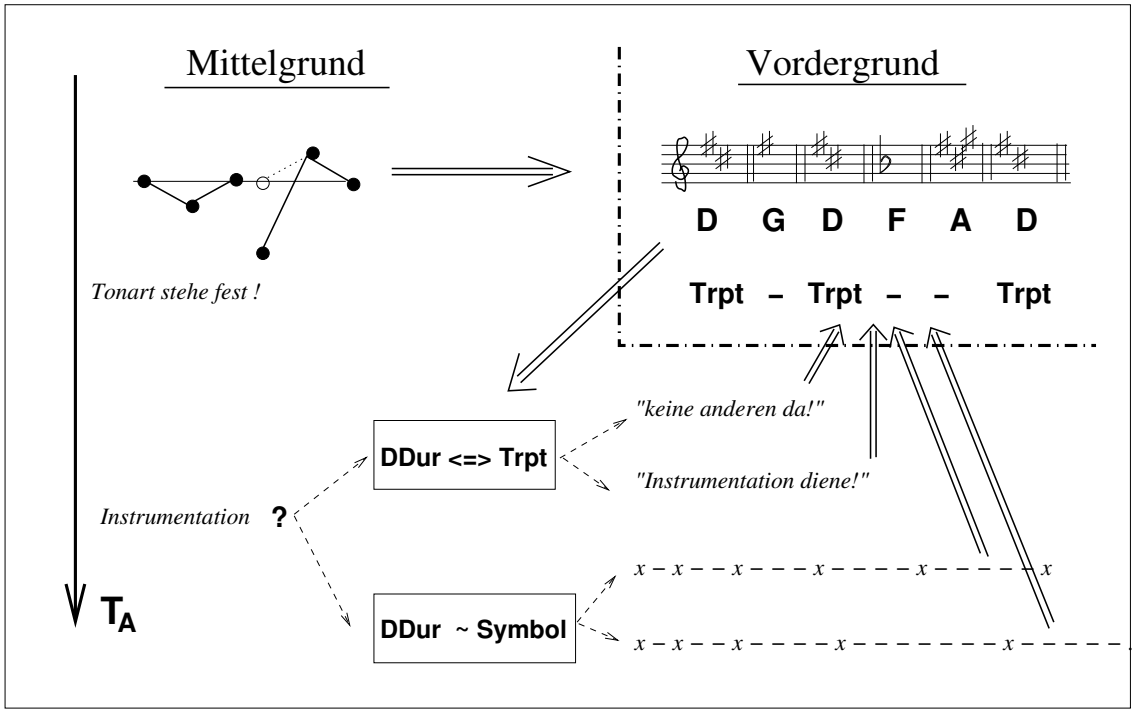


Abbildung 4.7: Informationsverlust beim Übergang von Mittel- nach Vordergrundstrukturen.

Andererseits (2) könnte der Trompeten-Einsatz aber auch (zugegeben: rein hypothetisch und keinesfalls wahrscheinlich) seinen Rhythmus ganz *unabhängig* von (und nur „zufälligerweise" identisch mit) dem Tonart-Muster erhalten haben. Die steigende *Seltenheit* der Trompetenauftritte könnte die Semantik haben, daß ausgehend von dem in Teil I schon stattfindenden zentralen Ereignis der *Geburt* Christi die „Heilsbotschaft" sich als konzentrisch vergrößernde Welle in alle Welt und Zeit ausbreitet.

Diese Ausbreitungsverhalten könnte entweder (2-a) nach der Folge der Dreieckszahlen ($\langle$ 1, 1+2, 1+2+3, 1+2+3+4, ... $\rangle$) reguliert sein, oder (2-b) nach der der Fibonaccizahlen ($\langle$ 1, 1+2, 2+3, 3+5, 5+8, ... $\rangle$).

Auch diese beiden, im Mittelgrund durchaus unterschiedlichen Fälle (2-a) und (2-b) wären im Vordergrund nicht mehr unterscheidbar, – Entscheidungs-Information ist verlorengegangen.

Aber auch der gegenteilige Effekt ist denkbar :

Für den Verfasser persönlich gehört es mit zu den befriedigendsten Momenten beim Komponieren, wenn zwei unabhängige und ganz unterschiedliche Mittelgrund-Überlegungen an derselben Werkstelle in dieselbe Vordergrundstruktur resultieren, – deshalb unangenehme Entscheidungspflicht vermeiden und darüber hinaus die Stimmigkeit des Mittelgrundes als Ganzem (das „Aufgehen des Konzeptes") zu beweisen scheinen.

4.4.5 Notation als historisch vermittelte Sprache.

In den wenigsten Fällen ist eine endgültige Partitur eine rein mathematische, kontextfreie Struktur-Beschreibung. Vielleicht einziges Beispiel könnten sein bei Nancarrows Walzen für automatische Klaviere.

Meist jedoch impliziert eine Notation ein *kulturell vermitteltes Vorwissen* von der Semantik der scheinbar objektiv notierten akustischen oder mechanischen Ereignisse, – selbst Stockhausens Studie Eins wurde mit der „Schere interpretiert".

Das Eröffnungsintervall von op. 111 z.B. würde von keinem Pianisten jemals mit beiden Händen gespielt. Sowohl die sichtbare Gestik, als auch die Klanglichkeit, als auch die psychische Befindlichkeit des Interpreten, welche sich ja dem Hörer mitteilen soll, erfordert als adäquate Realisierung das Spielen mit der linken Hand[1].

Diese zweifellos geforderte „Pranke des Löwen" steht nicht in den Noten, sie kennt man vielmehr dank der Tradition, und mit ihr die inhaltlichen Implikationen von Emanzipation des Individuums, Erstarken des Bürgertums und Rationalisierungstendenzen des ethischen Diskurses etc.

Ebenso, daß ein Beethovensches sforzato halt nicht nur eine Lautstärkenmodifikation bedeutet, sondern auch ein bestimmtes Tempoverhalten und einen Einfluß auf die anwendbaren Anschlagtechniken, eine psychische Grundhaltung und ein bestimmtes Körpergefühl sowie evtl. eine wichtige Markierung auf der eher intellektuellen Ebene der formalen Gliederung.

„Alberti-Bässe" bedeuten eben nicht vier gleichberechtigte Noten, sondern eine virtuelle Dreistimmigkeit, zumindest aber ein Absetzen der Unterstimme, – ein Drei-Viertel-Takt bedeutet im Zusammenhange eines Walzers eine erhebliche Ungleichverteilung der physikalisch meßbaren Zeit, während im Zusammenhang der Transkription mittelalterlicher Vokalmusik ein „tempus perfectum" gemeint sein kann, mit ganz anderer zeitlicher Realisierung.

Bild 4.8 zeigt einen Komponisten, der für eine Sängerin eine Partitur erstellt und als inhaltliche Zielvorstellung hat entweder (α) die Lösung eines (durch historische Abstrahierung entstandenen) satztechnischen Problems, oder (β) die Hervorbringung eines konkreten Schallereignisses, oder (γ) die Vorstellung eines Rezipienten, welcher (z.B.) einen „Pol mit Vorzeichenwechsel" empfinden soll, – also einen „Umschlag von Quantität in Qualität"[2], z.B. als eine eine Steigerung, welche auf ihrem Höhepunkt an der Grenze (gegeben durch die inneren Widersprüche des Materials) sich bricht und in einen Tiefpunkt umschlägt.

[1] Das Hinzutreten der kleinen Sept ges zum Schlußdreiklang des vorhergehenden „Satzes" der Hyper-Sonate und dessen enharmonische Umdeutung zur Doppeldominaten von c schließt die zwischen den ersten beiden Sätzen (op. 109 E-Dur und op. 110 As-Dur) schmerzhaft klaffende (sic!) Lücke.

[2] Dieser ist als musikalische Grundgestalt ein durchaus exakt bestimmbares Phänomen innerhalb der Rezeption von Musik. In unserem Bild 4.8 gelingt die Umsetzung dieses Wunsches übrigens nicht ganz, – der eingezeichnete Hörer empfindet etwas, das der intendierten Spannungskurve lediglich ähnelt.

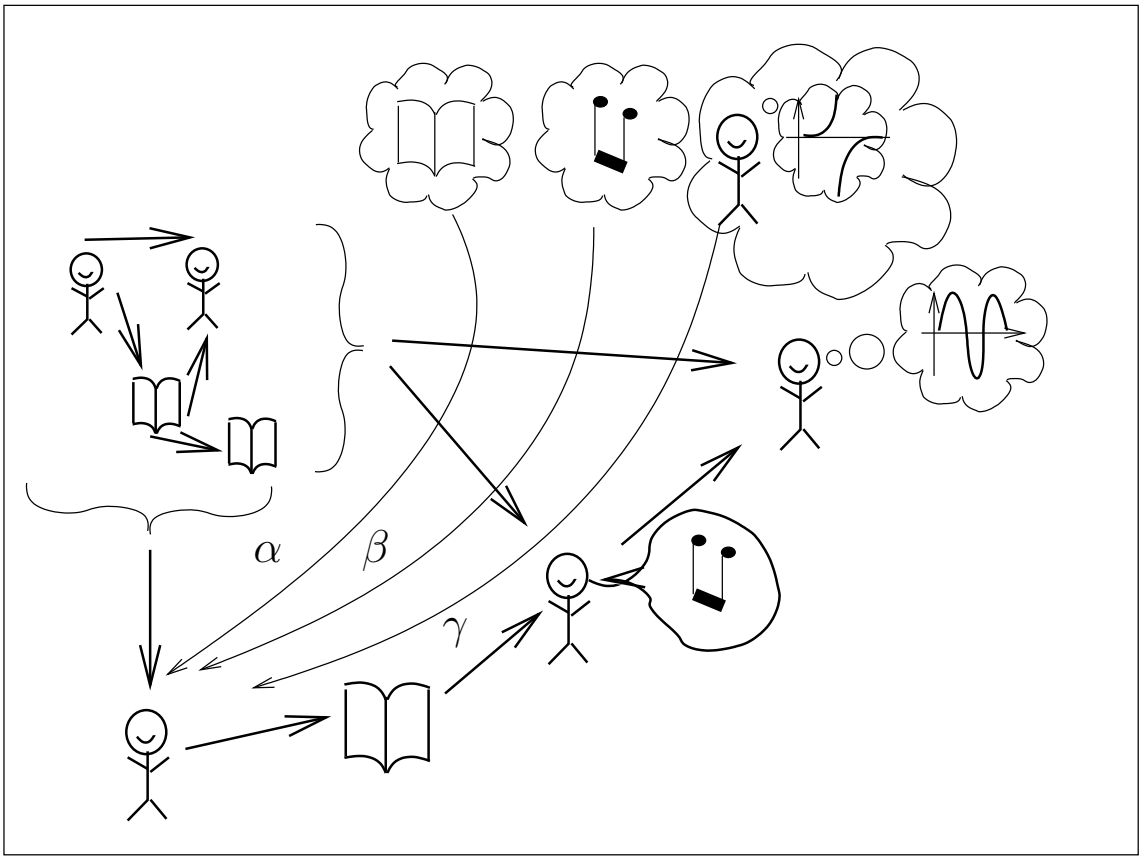


Abbildung 4.8: Historischer Kontext der an Musik Beteiligten.

Neben den durch Pfeile dargestellten Informationsflüssen, die u.a. den Interpreten mit der Tradition verknüpfen und somit die Kommunikation qua Partitur erst ermöglichen, fließt in die gestalterische Arbeit des Komponisten auch das Meta-Wissen um die Kommunikationsflüsse ein, z.B. die vorauszusetzende Hörerfahrung des Rezipienten, welches das Funktionieren der doppelt indirekten Kommunikation erst handhabbar machen kann.

4.5 Physischer vs. Ergonomischer Informationsgehalt.

Auch ein „fertiges" Modell des Kunstwerkes, die letztlich veröffentlichte Partitur z.B., ist also verschiedener tiefengestaffelter Transformation bedürftig.

Bei deren Betrachtung ist wichtig, nicht den mathematisch/physikalischen Begriff des „Informationsgehaltes" zu verwechseln mit dem psychologisch/wirtschaftlichen Begriff der „Informationsverfügbarkeit".

Eine Orchesterpartitur und ein ausführliches Particell können, abgesehen von Heterophonien und Vortragsbezeichnungen der Instrumentationsebene, durchaus fast identischen Informationsgehalt haben.

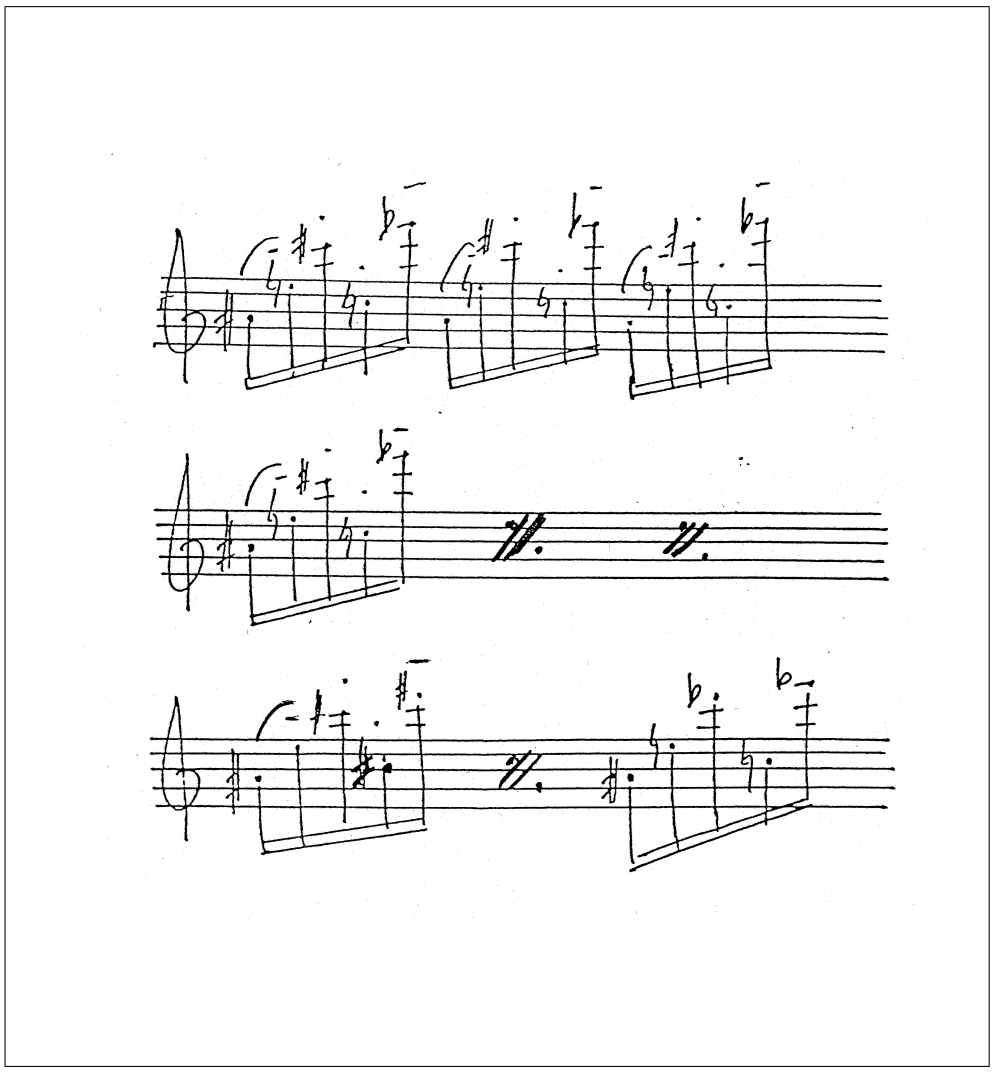


Abbildung 4.9: Unterschiedlicher Ergonomischer Informationsgehalt.

Um jedoch die Häufigkeit der Holzbläserverwendung zu überblicken oder den ersten Trompeteneinsatz zu finden, ist die Partitur „billiger", d.h. der Aufwand zur Gewinnung der Information („retrieval costs") ist geringer[3].

Für die Beurteilung von Satztechnik und Stimmführung ist hingegen das Particell zweckdienlicher. Die Arbeit der Informationsaufbereitung, in diesem Falle das Erkennen der wenigen Satzstimmen aus den vielen Orchesterstimmen, ist da nämlich schon geleistet.

[3]Auch auf der Meta-Ebene der *Informations-Sicherheit* („safety") ist die Bedeutung von Particell und ausinstrumentierter Partitur durchaus unterschiedlich: Eine im jeweiligen Kontext ungewöhnliche harmonische Wendung z.B. ist weniger wahrscheinlich ein Schreibfehler, wenn sie in mehreren Instrumentalstimmen ausgeschrieben steht.

So wurde der Verfasser manchmal von Zweifeln geplagt, ob jenes einsame, geniale, göttliche gis (klingend cis) im vierten Horn auf der Drei des letzten Taktes vor $\mathcal{R}$ im Adagio von Bruckners Neunter nicht doch bedauerlicherweise ein Kopierfehler des Notenstechers ist.

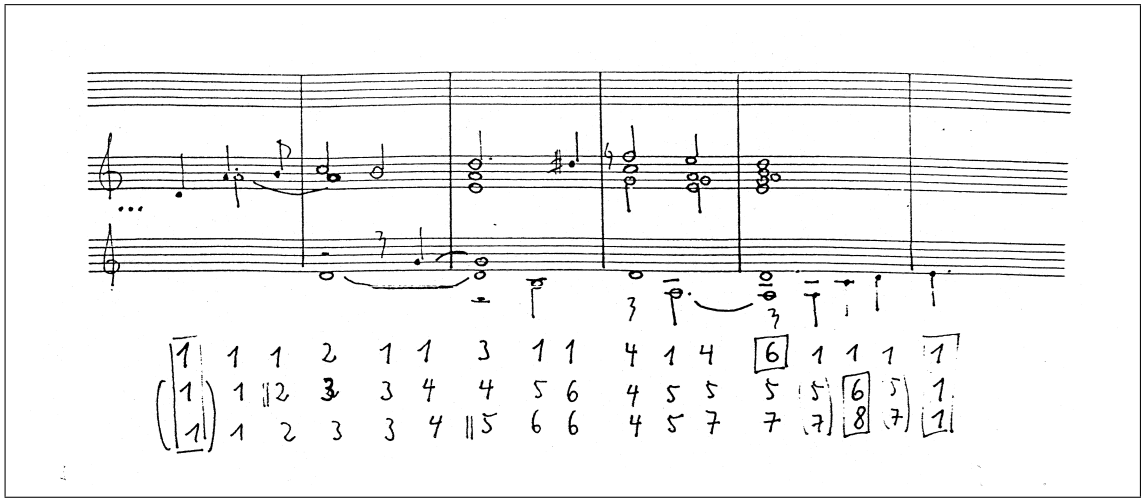


Abbildung 4.10: Erste Schritte musikalischer Analyse als schlichte Datentransformation.

Die abkürzende Bezeichung *Ergonomischer Informationgehalt* steht im folgenden für die (in der Tat recht komplizierte) Kosten-Funktion von Aggregatzustand einer Information „kreuz" gewünschter Teilinformation (= konkretem Anfrage-Kontext) in ein *Maß* des Informationsgewinnungs*aufwandes*[4].

Verschiedene Darstellungsformen (Aggregatzustände) der „physikalisch" identischen Information haben unterschiedlichen Ergonomischen Informationsgehalt, da sie verschiedenen Fragestellungen und verschiedenen Ebenen des menschlichen Denkens unterschiedlich stark adäquat sind.

Z.B. die Verwendung von sog. „*Faulenzern*", also Wiederholungszeichen, ändert den physikalischen Informationsgehalt keineswegs, gibt aber die zusätzliche (und ergonomisch höchst nützliche) analytische Information „hier ändert sich erstmal nichts".

Bild 4.9 zeigt verschiedene Schreibweisen der rechten Hand eines Klaviersatzes. Die zweite Zeile ist mit Sicherheit leichter zu lernen, wegen eben dieser Zusatzinformation, daß es sich um wörtliche Wiederholungen handelt, was aus der ersten Zeile nur durch relativ umständliches Vergleichen zu entnehmen ist.

Die dritte Zeile bringt eine Variante, in welcher sich (im Falle des Klavieres) physikalisch ebenfalls nichts ändert, der Autor jedoch Wert darauf legt, durch enharmonisch geänderte Notation eine Mittelgrundinformation (harmonisch geänderter Kontext) mitzuteilen, vgl. oben.

Ein *Gegenbeispiel* sind die Faulenzer im Vorspiel zu Bizets Carmen, welche, da nur *zufällig identische*Tonfolgen mit metrisch unterschiedlichster Bedeutung *unifiziert* werden, den Notentext häßlich und die Struktur schwer verständlich machen.

[4]Um alle Perikopen zu finden, welche ein bestimmtes Stichwort enthalten, ist das Erste Testament auf CD-ROM äußerst zweckmäßig, eignet sich in dieser Form allerdings kaum zu erbaulicher Lektüre vor dem Einschlafen.

Auch viele erste Schritte von musiktheoretischer *Analyse* sind lediglich Transformationen der Darstellungsform und damit Änderungen nur des Ergonomischen Informationsgehaltes, auf den dann ein Erkenntnisprozeß nach dem Motto des „Schau hin und erkenne" selbsttätig folgen soll.

Bild 4.10 zeigt einen Ausschnitt aus einem Klaviersatz des Verfassers (Lied „Von guten Mächten").

Die Ziffern der ersten Zeile bedeuten die stattfindenden gleichzeitigen Anschläge, die der zweiten Zeile die der noch klingenden Tonklassen (modulo Oktave), die dritte Zeile die klingenden Tonhöhen. Die in diesen Zahlenfolgen enthaltene Information ist implizit im Klaviernotat *vollständig enthalten*. Dennoch ist die dem Prozeß zugrundeliegende Dialektik aus den Ziffernreihen prima vista ablesbar, im Klaviersatz allenfalls erahnbar.

Ein Grenzfall ist die Bezeichnung „simile". Sie ist nicht nur gleichbedeutend mit einer fortgesetzten Wiederholung von Artikulation, spieltechnischen Angaben etc., sondern kann (ganz im Gegenteil !) auch bedeuten „spiele, daß es genau so *klingt* wie gerade".

Kapitel 5

Die Integrierende Digitale Arbeitsumgebung.

Die Hauptmotivation für den Einsatz von Digitalrechnern und ihre Hauptarbeitsgebiet sind (innerhalb unserer Modellbildung) zunächst strukturell äußert einfach, ja primitiv:

Der Rechner vollführt *aufwendige Daten-Transformationen* zwischen verschiedenen Datenformaten (nach Maßgabe des Benutzers) „automatisch".

Diese Transformationen geschehen zumeist von einer out-of-time Darstellung in eine diachrone (z.B. Reihengenerierung oder Partiturgenerierung durch Evaluierung von Generator-Algorithmen), – sind also Umformungen einer implizit gegebenen Funktion in eine explizite Darstellung.

Auch der umgekehrte Fall („automatische Analyse") ist in den letzten Jahren ein zunehmend wichtiges Anwendungsgebiet.

Die produktions- und rezeptionsästhetischen Konsequenzen allerdings jenes so einfältigen Grundprinzips sind überaus vielfältig und (sich gegenseitig modulierend) komplex vernetzt.

Zu deren Untersuchung wählen wir hier einen historisch vermittelten inkrementellen Ansatz:

Es soll in den folgenden Kapiteln untersucht werden, wie das bis hierhin entwickelte Modell von psycho-internen und sozial-kommunikativen Wirkungsmechanismen sich auswirkt und modifziert werden muß, wenn nun der *Digitalrechner als Werkzeug bei Kompositions- und Klangsynthese-Arbeit* hinzutritt.

Dies erscheint uns allemal sinnvoll, da sowohl die meisten Musiker, als auch fast alle Hörer ja mit den Werken der seit Hunderten von Jahren schrittweise entwickelten prae-digitalen Klangsprache aufgewachsen sind, – die zugrund liegenden Mechanismen also allemal im Hinterkopf weiterhin zunächst wirkmächtig bleiben.

Selbstverständlich ohne ausschließen zu wollen, daß völlig neuartige Material-, Organisations- und Denkformen sich wegen des Einsatzes des Digitalrechners herausbilden, steht im Zentrum dieser Schrift die Untersuchung der Fragen, (1) in welcher Form und mit welchen Konsequenzen eine *Übertragung* der prae-digitalen Arbeitsweisen in den Rechner sinnvollerweise stattfinden kann, aber auch (2), wie evtl.

entdeckte (in praxi scheinbar „zufällig", d.h. aus den Formbildenden Tendenzen des Materials entstandene) zunächst neuartig erscheinende Denk- und Arbeitsformen sich zum Zwecke der *begrifflichen Klärung* zu den gut erforschten prae-digitalen Formen verhalten, – sei es durch Analogie oder durch Gegensatz.

Diese Art des Vorgehens (und die damit verbundene Modellbildung) ist sicherlich eine traditionsverbundene, vielleicht sogar eine konservative.

Mag sein, daß die wirklichen Innovationen durch Kollegen erreicht werden, welche einen *gegenteiligen Ansatz* verfolgen – z.B. mit *explizitiervermeidung* der Übertragung von Begrifflichkeiten, – oder welche die folgenden Überlegungen schlichtweg für irrelevant halten und z.B. den Aufbau einer neuen, technologisch bestimmten Begrifflichkeit betreiben, – oder: Untersuchung von Rezeptionsverhalten auf der Basis von naturwissenschaftlichen Verfahren ohne ästhetologisch-historischen Ballast durchführen, etc.

Der in dieser Schrift vertretene Ansatz ist also ausdrücklich als einer von unterschiedlichen möglichen gemeint. Der Autor hofft allerdings, diesen und das Feld seiner Konsequenzen zumindest vollständig abgesteckt zu haben.

Die Entwicklung der letzten dreißig Jahre beweist, daß der Einsatz von Digitalrechnern in den verschiedensten Bereichen der kompositorischen Praxis und musiktheoretischer und -wissenschaftlicher Analytik vielerorts überraschende Erkenntnisse, tragfähige Konzepte und erfolgreiche Neuerungen mit sich gebracht hat.

5.1 Erwiesener Nutzen des Rechnereinsatzes in unterschiedlichen Arbeitsgebieten der Komposition und Realisation von Musik.

In der Tat kennt man eine Fülle von *Einzelbereichen*, in denen die ästhetischen und arbeitspraktischen Auswirkugen des (jeweils durchaus andersartigen) Einsatzes des Digitalrechners diesen überzeugend begründen.

Man könnte grob zwei Klassen sehen, einmal den *praktischen Nutzen in der täglichen Arbeit*, zum anderen aber den *mittelfristigen ästhetischen Fortschritt*.

Einige der nun in loser Reihenfolge beispielhaft aufgezählten Bereiche sind seit Jahrzehnten erfolgreich bearbeitet, andere werden z.Zt. überall auf der Welt beherzt angegangen, wieder andere sind Zukunftsmusik.

Interessant ist, das positive Effekte („Gewinne") aus so gegensätzlichen Basisstruktur-Einwirkungen wie zum einen aus neuen *Qualitäten* des Materials, zum anderen aus neuen *Quantitäten* seiner Verarbeitung hervorgehen.

Unabhängig davon geschehen die meisten Gewinne durch *Erleichterungen* oder *Neueinrichtung* von Arbeitsgängen, – andere aber auch überraschenderweise durch deren *Erschwerung* !

Wir stellen deshalb den im folgenden aufgeführten Einzelbereichen jeweils die Einordnung ihrer *Tiefenstruktur* nach den aus beiden Kriterien folgenden zwei mal zwei möglichen Fällen voran.

- [Qualitativ neue Arbeitsweisen wegen *quantitativer Erleichterung* in der Produktions-Effektivität (Beschleunigung des feed-back):]

  Durch Digitalrechner können komplexe Strukturgenerierungs-Vorschriften *mehrere Größenordnungen schneller* durchgeführt werden.

  Dadurch ist es in vielen Fällen *zum erstenmal überhaupt möglich*, mit vernünftigem Aufwand / Wartezeit *Varianten* zu entwickeln (= Evaluierungen von Strukturdefinitionen mit jeweils (leicht) anderen Parameter-Werten.)

  Das zyklische Arbeitsverfahren von „Varianten produzieren $\Longrightarrow$alle rezipieren $\Longrightarrow$entscheiden" wird somit für *komplexere* Definitionen durch den Digitalrechner überhaupt erst anwendbar.

- [Qualitativ neue Arbeitsweisen wegen *quantitativer Erleichterung* in der Produktioneeffektivität : Beschleunigung von Analyse:]
  Entsprechend zu oben können nun auch die unteren, mechanischem Arbeitsschritte der musiktheoretischen- oder -wissenschaftlichen Analyse über große Datenmengen automatisch und wirtschaftlich ausgeführt werden, – wohlgemerkt: tatächlich *nur die ersten, allereinfachsten* Arbeitsschritte einer wirklichen Analyse.

- [Diverse Gewinne durch *qualitative Erschwernis* : Zwang zur Formalisierung:]

  Paradoxerweise vermuten wir diverse Nutzen aus der qualitativen *Erschwernis*, bestehend im weitgehenden Zwang zur Formalisierung.

  Beispielsweise ...

  1. Didaktisch-produktionspraktischer Gewinn: Noch genauere / tiefere Durchdringung der eigenen Vorstellung. I.A. folgt daraus eine besseres Verständnis der Ausgangsvorstellung, daraus folgt wiederum normalerweise eine *Vereinfachung* von Gesamtarchitektur und Einzelvefahren[1]. Diese hat wiederum sowohl leichtere *Handhabbarkeit* in der praktischen Arbeit als auch stärkere Konsistenz und damit *Wirkmächtigkeit* des Endproduktes zur Folge.

     Der Rechner fungiert hier gleichsam als Materialsation von „Ockhams Messer".

  2. Didaktischer Meta-Gewinn: Dder Zwang zur Formalisierung und Reduzierung hat für das Unterrichtsgespräch i.A. (1) eine klarere und konkretere Terminologie zur Folge, und (2) das herausschälen von Grundverfahren, welche von einem Studenten zum anderen übetragbar (natürlich auch adaptierbar und fortentwickelbar) sind.

[1] So konnte z.B. im Verlauf von PGP erkannt werden. daß die ajouté-Schicht (8.2) *ohne* (ursprünglich vom Studenten vorgeschlagene) Spracherweiterung recht problemlos (als ein Sonderfall/Extremfall bezgl. der Parameterwerte) zu realisieren war.

3. Die an anderer Stelle dieser Liste auftretenden Förderungen von Wiederverwendbarkeit („reusablity"), Austauschbarkeit, Reproduzierbarkeit etc. haben allesamt ein hinreichendes Maß an zumindest grundlegender Formalisierung zur Voraussetzung!
4. praktischer Gewinn: Auch die u.e. Versionsverwaltung und Entscheidungsprotokoll (undo, rewind, preview, Anzeige etc.) sind nur auf formalisierten (genormten) Daten sinnvoll (hinreichend aussagekräftig) möglich!
5. Dies ist aber auch ein Meta-Gewinn, da der Entscheidungsgraph nun physische gespeichert werden kann und als konkretes Material der kompositionstheoretischen Analyse nun zugänglicher ist.

- [*Quantitative Erleichterung* von undo, redo, Punktuellem Durchgriff etc. :]

  Der o.e. Zwang zur Formalisierung hat u.U. einen gewissen Verlust an (spontaner) Flexibilität zur Folge.

  Dieser wird teilweise ausgeglichen durch die *wesentlich unaufwendigere* Durchführbarkeit von spontanen Experimenten bezgl. deren Umkehrbarkeit und Wiederholbarkeit („undo" und „redo/replay"):

  Wenn im klassischen analogen Bandschnitt verschiedene Versionen einer kleinteiligen Montage „ausgehört" werden sollten, waren für jede einzelne Klebestelle minutenlange Manipulationsfolgen nötig: Band einlegen→ manuell drehen und Schnittstelle heraushören→ mit Fettstift markieren→ Band herausnehmen und schneiden→ alte Klebebänder abknibbeln→ Bandschnipsel aus Versehen falsch herum einkleben etc.

  Bei mehr als einem Dutzend Schnipsel kürzer als fünf Zentimeter konnte man schon einmal, trotz aufwendiger Buchführung, die Übersicht velieren.

  Um verschieden Zwischenergebnisse vergleichbar zu machen und um Schnitte unaufwendig rückgängig machen zu können, mußten ständig Zwischen- und Sicherheitskopien überspielt werden.

  All dies ist im Falle des rechner-internen Schneidens (potentiell! s.u.) wesentlich unaufwendiger und schneller möglich (was wiederum in eine qualitativ neue Arbeitstechnik durch quantitative feed-back-Beschleunigung mündet, s.o.)

  Der *Punktuelle Durchgriff*, – also das spontane Verändern einzelner Stellen einer ansonsten algorithmisch („automatisch") generierten Gesamtstruktur im nachhinein, – ist auf den Ebenen der *Parameter*verläufe durch deren rechnerinterne Repräsentiertheit überhaupt erst möglich geworden (auf der Ebene der *Schall*verläufe waren sie in der Analogtechnik schon möglich, allerdings nur in bestimmten Situationen und nie ohne höheren Aufwand).

- [Gewinne durch die Kompensation der *Verluste* an haptischer *Qualität* :]

  Eine durchgängig rechnerinterne Repräsentation hat allerdings hohe Verluste an physischer Anschaulichkeit und leiblichen Rückkopplungswegen zur Folge:

  - Fünf vor mir liegende Schieberegler, die zugleich robust und empfindlich sind, können mit den Fingerspitzen millimeterweise einzeln gestreichelt oder mit der Handkante alle zusammen schlagartig zugezogen oder aber mit der flachen Hand förmlich geknetet werden, auf diese Weise Zufallsfunktionen sich ändernder Amplitude improvisierend[2].
  - Zeitfunktionen konnten durch die (leiblich empfindbare) Beschleunigung schwerere Massen realisiert werden (Das Cembalokonzert des Autors begann mit dem manuellen In-Drehung-Versetzen des schweren Wickels der Achtspur-Maschine, gleichsam mit dem Anwerfen eines Motors, – „Das hört man auch!")
  - Die Dicke der vor mir liegenden Bandwickel gibt nicht nur u.U. ein gutes Gefühl für das in dieser Nacht Geleistete, sondern auch unmittelbare Anschaulichkeit von Zeitproportionen verschiedener Formteile oder Schichten.
  - etc.

  All diese Ein- und Ausgabekanäle sind mit Sicherheit anschaulicher (und wegen ihrer Unterschiedlichkeit auf die Dauer auch *weniger gesundheitsschädlich*) als das Bedienen *virtueller* slider und Betrachten *virtueller* Bandwickel.

  Um die so verloren gegangene Flexibilität der Arbeitsweise auszugleichen, müssen Neuentwicklungen stattfinden, welche letztlich Gewinn und Fortschritt bringen können.

  Die *verlorengegangene* Ganzheitlichkeit der haptischen Eingabe muß durch einen *neue Ganzheitlichkeit* durch abstrakt-funktionale Superzeichenbildung ausgeglichen werden.

  Am Beispiel des Bandschnittes heißt das, daß die Schnittvorgänge auf einer *höheren Ebene* als Komponenten einer „Schnitt-Algebra" formuliert werden sollten. Die Elemente dieser Algebra (die Transformationsvorgänge auf den zu schneidenden „Bändern") werden wiederum zu Objekten und wären dann mit Mitteln einer Meta-Algebra behandelbar, d.h. könnten verglichen, übertragen, automatisch disponiert und generiert, sequenziert, in Regeln abstrahiert werden etc.[3]

[2] Hofschneider pflegte bezgl. der Bedienung der frequenzbestimmenden Drehknöpfe am SynLab zu sagen: „Nicht drehen, nur denken!"

[3] Die dem Verfasser bekannten digitalen Schnittsysteme erfüllen solche Anforderungen leider nicht, sondern spezialisieren sich auf hübsche bunte Oberflächen, – sind infolge dessen meist nur zähneknirschend bedienbar.

- [Meta-Gewinn durch *qualitativ Neues* : Abstrahierbarkeit von Reglerbewegungen:]

  Die alte Analogtechnik hatte als Eingabegeräte (u.a. deshalb, weil entgegen dem Grundkonzept der Spannungssteuerung kein Modul jemals mit *sämtlichen* Parametern und Schaltern aus wirtschaftlichen und Übersichtlichkeits-Gründen vollständig spannungssteuerbar zu bauen war) *an den* Modulen Regler, Schalter, Drehknöpfe etc.

  Deren digitale Nachfolger sind nun entweder virtuell, d.h nur innerhalb (des GUI) des Rechners realisiert, oder –besser noch, weil weniger Verlust an Haptik – physisch real, aber an den Rechner angeschlossen.

  Der Vorteil der alten Technologie bestand in der unmittelbaren Identifizierbarkeit der Regler allein durch ihren Ort im Raum: Wenn ich mir gemerkt habe, welches VCA-Modul in meiner momentanen Verschaltung welche Funktion hat, kann ich den für jeden spontanen Eingriff zuständigen Sub-Regler des Moduls sofort und reflexmäßig identifizieren, er-kennen und er-reichen.

  Rekonfigurierbare virtuelle Regler und physische Regler mit redefinierbarer Bedeutung haben diesen Vorteil hingegen nicht.

  Andererseits: ihr ganz großer Vorteil bestht in *völlig neuen Qualitäten*, entspringend aus ihrer Durchleitung durch den Rechner!

  1. Erst mit diesen „indirekten Reglern" können die durch sie während des Ablaufes einer in-time-session eingegebenen spontan improvisierten Zeitfunktionen *aufgezeichnet* werden.
  2. Damit sind auch andere im Rechner implementierten Verfahren von Protokollierung, Materialverwaltung, Versionsverwaltung, Replay etc. auf diese Zeitfunktionen anwendbar.
  3. Insbesondere können sie als Datenobjekte im nachhinein ediert werden (Punktueller Durchgriff, Montage der Ergebnisse verschiedener Sessions etc.), um dann „automatisch gereplayt"zu werden.
  4. Weiterhin kann eine improvisierte Regelkurve (im Rahmen eines replay) auf *andere Regelgrößen übertragen* werden[4] (nach evtl. Zwischenschaltung von einfachen, algorithmisch formulierten Adapterfunktionen)...
  5. oder dies improvisierte Material kann als Eingabegröße in algorithmische Weiterverarbeitung oder Materialgenerierung einfließen[5].

[4] Das historisch-aphysikalische Objekt der „vergangenen Bewegungsgeschichte" des Reglers wird im wörtlichen Sinne von dem dabei gesteuertem Gerät ab-strahiert, d.h. abgezogen und objektiviert, d.h. zu einem seinerseits handhabbaren Datenobjekt.

[5] Allerdings war es schon in VC-Studio durchaus möglich, „hi-order"-Parameter einer komplexe Strukturen liefernden Verschaltung (z.B. Dichteverläufe, harmonische Raster, Grade von Regularität/Irregularität etc.) mit Schiebereglern „haptisch" zu steuern, – wenn auch nicht in allen Fällen und nie ohne einigen Aufwand an Kreativität bei der Schaltungsdefinition.

- [Gewinn durch *qualitativ Neues* : Neue Wege für „experimentelle" Musikgenerierung:]

  Schon durch die *out-of-time* algorithmische Generierung von Klangverläufen sind völlig neue experimentelle Fragestellungen erstmalig praktisch beantwortbar, die dem Muster folgen ...

  „Wie klingt diese Formel ?".

  Die Resultate von fraktalen Wachstumsformeln, dynamischen Prozesse, zahlentheoretischen Algorithmen, state-machine-ensembles oder Agenten-Modellen etc. können nach ihrer Auswertung im Rechner allemal auch in Klang (oder in klangsteuernden Parameterverlauf) umgedeutet werden.

  Der Verfasser erinnert sich an ein äußerst einfaches, einleuchtendes und anregendes Werk von Carla Scaletti mit dem Titel Sun Surge Automata=Sun Surge Automata welches aus Knackgeräuschen synthetisiert war, die sich einem abstrakten, wachstumssimulierenden Algorithmus folgend verdichteten und zuletzt durchaus neuartige Klanglogik hervorbrachten.

  Darüberhinaus können auch rechnerinterne Darstellungen von Kunstwerken *anderer Gattung*, aber auch „neutrales Material" wie statistische Daten, digitale Landkarten, pixelcodierte Fotos o.ä. wegen des allgegenwärtigen tertium comparationis (oder „Modulationsmittel") der rechnerinternen Ziffernwelt probeweise als Klang interpretiert werden.

  So bietet der Rechner der kretativen Kombinatorik (auf Material-Ebene) völlig neuartige Betätigungsfelder.

Alle diese Einsatzweisen und -bereiche, durch die und in denen der Digitalrechner beim Komponieren und Realisieren höchlich nützlich ist, sind strukturell und inhaltlich also durchaus verschiedenartig, – sie bieten jeweils *durchaus unterschiedliche* Probleme, Lösungsansätze und Konsequenzen sowohl für Arbeitstechnik als auch für ästhetische Rezeption.

Wir behaupten aber :

Gemeinsam aber ist den allermeisten dieser Anwendungen des Digitalrechners, daß dieser seine wohltätige Wirkung um so stärker entfaltet, und – wichtiger noch – daß die durch ihn *neu auftretenden* Hemmnisse, Aufwendigkeiten und Ärgerlichkeiten, die man keinesfalls unterschlagen darf, sich um so weniger auswirken, je *höher der Integrationsgrad* eines Modul-Ensembles ist und je unproblematischer der Datenaustausch[6] zwischen den Funktionseinheiten realisiert werden kann.

Z.B. ist o.e. „Abstrahierbarkeit von Reglerkurven" eigentlich erst wirklich interessant, wenn möglichst jede Kurve von jedem Kurven-Datentyp gleich unaufwendig mit diversen (graphischen/textuellen/regelbasierten) Editoren verarbeitet werden kann, – möglichst jeder Algorithmus und jedes Klangsyntheseverfahren, das mir auf dem Rechner zur Verfügung steht, beliebig mit solchen Kurven gesteuert werden

[6] ... wobei hier unter „Daten" auch Funktionen, Regeln und ganze Programme als „Daten höherer Ordnung" verstanden werden sollen.

kann, – einfache (automatisch eingefügte) Konvertierungen zwischen den Ausgangskurven verschiedener Input-Moduln erfolgen, – die Kurvenobjekte an ein *Version Control System* angeschlossen sind, etc.

Für die anderen o.e. Bereiche lassen sich ähnliche Überlegungen anstellen, welche unsere obige Behauptung durchaus stützen und bei konsequentem Weiter-Denken letztlich zur Forderung nach der Integrierenden Digitalen Arbeitsumgebung führen, welche wir auch unten (5.4) erheben werden.

5.2 Notwendigkeit der projektspezifischen „Benutzersprache".

In den vorigen Kapiteln haben wir gesehen, daß zwei Grundtypen von Sprachen als zentrale Medien im Kompositionsprozeß auftreten: Die *Innere* Sprache des Komponisten und die *historisch* vermittelte Sprache der musikalischen Notation.

Beide sind naturgemäß dem Digitalrechner nicht zugänglich.

Vielmehr ist eine Kommunikation mit einem Rechner, und damit seine Verwendung auch im Kontext des künstlerischen Schaffensprozesses, bekanntlicherweise ausschließlich durch den Kanal einer sog. „Programmiersprache" möglich[7].

Wenn nun die Verflechtungen der verschiedenen intra-psychischen Ebenen der *Inneren* Sprache als auch die hochgradig kontextabhängige Semantik der *äußeren* musikalischen Notation als bedeutender konstitutiver Faktor in die *Substanz* eines Kunstwerkes einfließen, folgt daraus u.E., daß auch die Tiefenstrukturen der anzuwendenden Programmiersprache *Bestandteil* des ästhetischen *Materials* sind, mit welchem der Komponist sein Werk zu erstellen versucht.

In der prae-digitalen Arbeitsweise vollzieht oder erlebt der Komponist mit jedem neuen Werk eine (mehr oder weniger bewußte und mehr oder weniger weitreichende) *Neuorganisation* seiner Inneren Sprache als Teil seiner ästhetischen (Meta-)Entscheidungen.

Wir fordern deshalb grundsätzlich, daß der Einsatz einer Digitalen Datenverarbeitung im Kontext ästhetischer Produktion nicht nur die zu verwendenen Programmiersprachen kritisch auf die „formbildenden Tendenzen des Materials" hin untersucht, sondern außerdem, daß ausgehend von der Analyse der (im vorigen Kapitel benannten) Hintergrundsinformationen (= Ausgangskontext, Zielvorstellung, Thematisches Material) und der voraussichtlichen Struktur des Mittelgrundes eine *je Projekt* die Definition einer *eigenen, spezifischen* Programmiersprache als erster (zumindest: früher) Schritt der *Materialgenerierung* und als bewußte ästhetische Entscheidung innerhalb der Rechnerumgebung stattfindet.

Die so definierte Sprache nennen wir *Benutzersprache*.

Die Sprachen, in welchen diese implementiert wird, nennen wir *Infrasprachen*[8].

[7] Auch graphische Benutzeroberflächen, Betriebssystem-shells, ja sogar die Bedienungsvorschriften von Fahrkartenautomaten wollen wir im folgenden unter dem Arbeitsbegriff „Programmiersprache" zusammenfassen.

[8] Der in der Informatik teilweise gebräuchliche Ausdruck „Meta-Sprache" kollidiert leider mit

5.3 Forderungen an Benutzersprachen.

Eine Benutzersprache muß beinhalten ...

- Codierung und Bearbeitbarkeit der *Mittelgrund-Information.*

  Die Mittelgrund-Information ist (wie oben gezeigt) das Arbeitsmedium des Schaffenden.

  Verglichen mit ihnen ist die Codierung von *Vordergrund*strukturen i.A. unproblematisch.

  Diese sind im Schaffensprozeß aber lediglich mnemotechnische Hilfsmittel zum Aufspannen des psycho-internen Mittelgrundes. Solange ein Rechner sich nur im Vordergrund bewegt, bleibt der Einsatz von Papier und Bleistift normalerweise die weitaus zweckmäßigere Alternative[9].

  Auch die beim Abspielen des Notentextes (Vordergrundstruktur) stattfindenden historisch begründeten Transformationsprozesse im Denken des Interpreten haben (vorweggenommen) im Denken des Komponisten häufig *Materialfunktion.* Die einzig mögliche rechnerinterne Entsprechung dafür ist ebenfalls, die (dabei gemeinten) Mittelgrundstrukturen als Objekte konkret zu realisieren und als solche manipulierbar zu machen.

  Beispiel:

  Die Datenstruktur des *Notentextes*, wie er gedruckt auf Papier erscheint, zu implementieren, ist nur die eine Sache. Diese Aggregatform vertrete in unserem Beispiel die „Vordergrund-Funktion".

  Wenn ich nun aber in Wirk-Lichkeit eine Zwölftonkomposition erstelle, dann denke ich tatsächlich in Reihen und Modi.

  Wenn die Benutzersprache den Mittelgrund repräsentieren soll, dann muß sie folglich Objekte (nebst ihren Algebren) wie `zwoelftonReihe`, `Permutationsvorschrift`, `Ausloeschungsmoeglichkeitsanalyse`, `Transpositionsplan` etc. implementieren.

  Wenn ich hingegen funktionalharmonisch schreibe, operiere ich auf Klanggestalten und -verbindungen. Die rechnerinterne Objektwelt muß in diesem Fall Funktions-Symbole[10], Klanggestalten, Tonarten und Modulationsregeln als „first class resident" und konkret manipulierbare Gegenstände beinhalten.

unserer Verwendung der Vorsilbe „meta", welche u.a. die reflektierende, methodologische Ebene benennt. Außerdem ist „infra" einfach die Sache treffender, – obwohl das keine Entschuldigung wäre, etablierte Terminologie mit einer privaten zu umgehen.

[9] Alle uns bekannten Notationsprogramme z.B. kleben viel zu stark am Vordergrund, so daß selbst das Setzen einfacher Partituren für den intelligenten Benutzer zum Horrortrip wird:

Zu der (bereits komplexen) Dialektik von psycho-internem Mittelgrund und Übersetzungsregeln in den Vordergrund tritt als Drittes die grundsätzlich probleminadäquate *Eingabgesprache* des Notensatzprogrammes. Deren Idiosynkrasien entfalten schädliche Formbildende Tendenzen, die zu bekämpfen oder auszugleichen oft zusätzlich anstrengend ist.

[10] Das Wort „Funktion" meint hier den Fachbegriff aus der Musiktheorie, nicht den aus der Mathematik oder der Computertechnik.

Den Unterschied zwischen diesen beiden (wirk-lichen) Denk-Räumen (operationalen Objektwelten, Mittelgrund-Räumen) und dem Notentext (als Vordergrund) erkennt man deutlich an dem jeweils stattfindenen *unterschiedlichen Übersetzungsschritt* von Mittel- in den Vordergrund:

Den Übersetzungsschritt wiederum erkennt man leicht daran, daß er i.A. die Bestimmung der (stets notwendigen) *Vorzeichen* beinhaltet. Diese geschieht für unsere beiden Beispiele fast nach diametral entgegengesetzten Strategien :

Im ersten Fall (Zwölfton-Komposition) ist die Vorzeichnung nur ein praktisches Mittel, die Tasten des Klavieres zu benennen, – „Ob es, ob dis, das gelte alles gleich". Im zweiten Fall hingegen die Entscheidung zwischen „gis" oder „as" ganze Welten von Semantik tragen.

Interessanterweise scheint die Vorzeichenbedeutung und -behandlung innerhalb der großen „Familie der westlichen Musiken" sogar *skalierbar* zu sein:

Auf halber Strecke zwischen den Extremen von Serialismus und Funktionalharmonik steht z.B. die „freie Atonalität", bei der die Notation u.U. auf die stets vorhandene und wirksame *Allusion* an tonale Strukturen hinweisen kann.

Zwischen dieser und dem funktionalharmonischen Denken steht wiederum die Orthographie, die dem Mittelgrund der Jazz-Harmonielehre entspricht: Prinzipiell kann meistens eine bezgl. der Tonalität korrekte Schreibweise benannt werden, – für die häufig hinzugefügten Nonen, Undezimen, freien Leittöne und tongeschlechtfremden Terzen etc. sind wahrscheinlich oft verschiedene Vorzeichnungen gleichermaßen adäquat[11].

- Codierung und Bearbeitbarkeit von *Meta-Bestimmungen*.

  Auch Meta-Bestimmungen, von denen im vorigen Kapitel einige Beispiele gegeben wurden, sind *Material*, – können in bestimmten Projektarten sogar zum zentralen Material werden.

  Auch diese sollten (so weit als mögliche) innerhalb des Rechners realisierbar, nach außen darstellbar und von außen (algorithmisch) manipulierbar sein.

  Wenn ich z.B. für ein „musique concrète"-Werk diverse Violinisten husten lasse, dann sollte die Verwaltung der „`recording`"-Objekte die Identifikation des Interpreten beinhalten, – diese aber sollte mit einer Adress-Datei verknüpfbar sein können, oder mit der Honorar-Buchhaltung.

  Weiterhin sollte z.B. die Bandschnittliste und ihre Geschichte mit meinem generellen Arbeitsprotokoll verknüpft sein (im weitestgehend Fall: mit meinem digitalen Terminkalender, – im primitivsten Fall: nur qua file-Datum), weil die psycho-interne Identifizierung (Benennung) musikalischer Materialien oft nach dem Muster verläuft: „Spiel doch noch mal die letzte Version von vorgestern Nacht !"

[11] Womit der Verfasser als Laie im Bereich des Jazz sich auf das durchaus gefährliche Gebiet der Vermutung begibt.

Der einfachste Fall von Meta-Bestimmungen sind noch (textuelle und/oder grafische) Notizen und Kommentare, welche ich sowohl an Objekte anheften will, als auch chronologisch als Protokoll nachlesen können muß.

Die komplexesten Fälle von Meta-Bestimmungen sind z.B. codierte Regeln, welche ganze Kompositionsprogramme kontextabhängig auswählen, wie es ja im Prokekt mkSinf geplant war.

- Generische Formulierung.

  Obwohl eine Benutzersprache als eine projekt*spezifische* ästhetische materialgenerierende Entscheidung definiert ist, sollten die implementierten Verfahren, generierten Daten und gewonnenen Erkenntnisse *übertragbar* formuliert werden.

  Übertragbarkeit (auf andere Parameter oder Strukturebenen) und Kombinierbarkeit (mit anderen Verfahren oder Datentypen) sind Eigenschaften, welche sich sowohl innerhalb eines Projektes positiv auf die Flexibilität der Entscheidungsmöglichkeiten und die Vereinfachung der Gesamtarchitektur auswirken, als auch projektübergreifend das Konzept einer Integrierenden Digitalen Arbeitsumgebung (Wachsen mit jedem Teilprojekt, Dialog der Benutzer über Materialaustausch etc, s.u.) überhaupt erst ermöglichen.

5.4 Voraussetzung für Benutzersprachen : die Integrierende Digitale Arbeitsumgebung.

In den meisten Fällen kombiniert eine Benutzersprache den Umgang mit den verschiedenartigsten musikalischen Materialien (Rhythmen, Tonhöhen, Instrumente, etc.) und computerspezifischen Technologien (Parser, Grafik, Objektverwaltung, etc.) .

Die Realisierung auch nur der anspruchslosesten Benutzersprache wäre schlechthin wirtschaftlich unmöglich, müßten all diese benötigen Sub-Verfahren jedesmal von Null an neu implementiert werden.

Sie ist also nur möglich in einer Umgebung, welche funktionsbereite, flexible und kombinierbare Moduln zur Behandlung der Materialien und Verfahren der unteren Ebenen bereitstellt, welche von der Benutzersprache möglichst problemlos verwendet werden können.

Diese Umgebung nennen wir *Integrierende Digitale Arbeitsumgebung*, kurz IDA.

Wir wollen nun versuchen, die Kernforderungen an die Integrierende Digitale Arbeitsumgebung (und auch an die Infrasprache) aus der Summe unserer Erfahrungen herauszuziehen.

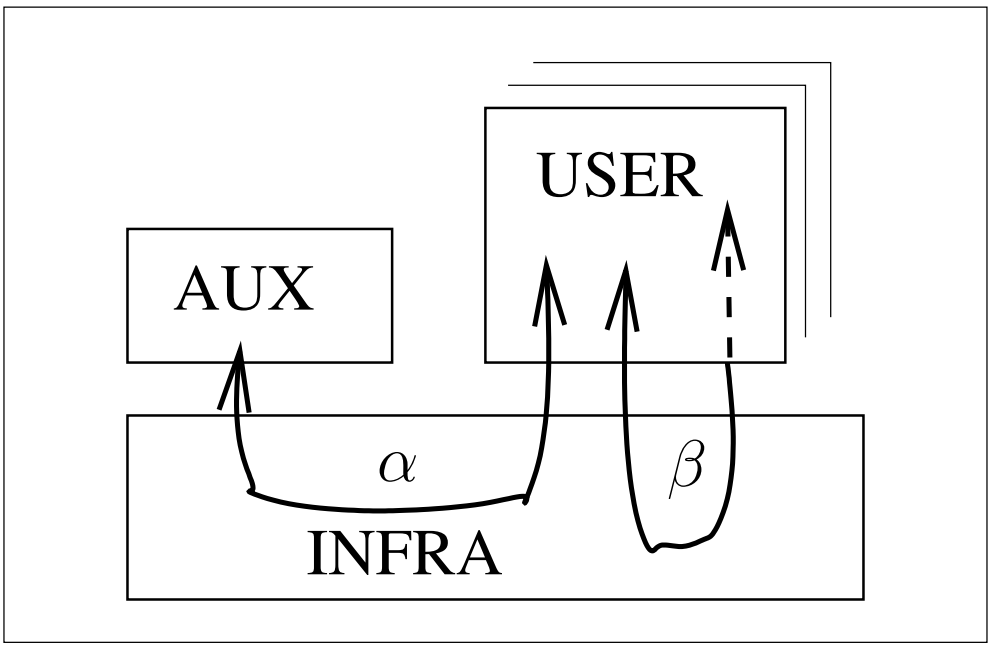


Abbildung 5.1: Prinzip der Integrierenden Digitalen Arbeitsumgebung.

5.5 Die beiden Hauptaufgaben einer IDA.

Die wesentliche Funktion einer Integrierenden Digitalen Arbeitsumgebung ist die Herstellung der *gegenseitigen Verwendbarkeit* komplexer (= aufwendig zu erstellender) Funktionseinheiten (siehe Bild 5.1).

Diese strukturell identische, primär technologisch erscheinende Fähigkeit bezieht sich nun auf zwei verschiedene Klassen von Wegen des Informations- und Kontrollflusses, mit jeweils durchaus unterschiedlichen Motivationen als auch Konsequenzen !

Zum *einen* erfordert, wie oben ausgeführt, der Zwang zur Wirtschaftlichkeit bei den Implementierungen der (allemal notwendigen) Benutzersprachen die Verfügbarkeit von eher technologisch orientierte Hilfsmoduln (Editoren, Parsergeneratoren, Objekt- und Versionsverwaltung etc.), im Bild mit AUX bezeichnet.

Ebenso sollen auch „intelligentere" und fachspezifischere Hilfsmoduln (hier z.B.: Harmoniksynthese und -analyse, Zeitorganisation, Tonsysteme, Permutationsgeneratoren, Formplanmodellierung etc.) allesamt mit *möglichst wenig Anpassungsaufwand* und unter *möglichst wenig Einschränkungen* benutzbar, kombinierbar und „verschaltbar" (=gegenseitig kontrollierbar) sein.

Um dieses möglichst einfache Kombinierbarkeit zu gewährleisten wären erforderlich (1) die *generische* (oder sonstwie polymorphe) Formulierung der Moduln, und (2) die Bereitstellung universeller *Basisalgebren*, welche einen systemweiten Austausch semantischer Information auf eindeutige Weise erst ermöglichen.

Auch die erste Gruppe scheinbar „primitiver" AUX-Moduln ist nämlich insofern keineswegs trivial, als auch sie grundsätzlich „Daten höherer Ordnung" empfangen und abliefern, – also Funktionen, Regeln, Termbäume o.ä.

Die Realisierung dieser Datenaustauschmechanismen ist Aufgabe der das gesamte System tragenden INFRA-Ebene.

Zum *anderen* aber, – ist die Kommunzierbarkeit komplexer Daten- und Programmobjekte erst einmal zwischen „Applikation" und „AUX-Bibliothek" hergestellt, – können natürlich auch „Applikationen" untereinander fleißig Daten und Kontrolle austauschen, – sie verhalten sich dann nicht grundsätzlich anders als System-Moduln !

Während die Kombinierbarkeit unterer Systemmoduln der Forderung nach unaufwendigem *praktischem* Umgang mit dem System entspringt, hat die Kombination von anwendungsorientierten Moduln durchaus *inhaltliche* Konsequenzen.

Die Funktion einer Integrierenden Digitalen Arbeitsumgebung besteht dabei *zum ersten* in der Übertragbarkeit von Algorithmen, Materialien und Strukturen zwischen den unterschiedlichen Bestimmungsebenen *ein und desselben* gerade entstehenden Werkes. Die mehrfache Anwendung desselben Materials in ganz unterschiedlichen Bestimmungsschichten eines Werkes erhöht allemal dessen ästhetische Konsistenz und Wirksamkeit.

Darüberhinaus aber, als *lernfähiges*, mit jedem Werk *wachsendes* System, unterstützt sie die Vernetzung *verschiedener Werke desselben Komponisten* durch Übernahme von (im Kontext des einen Werkes entwickelten und evtl. abstrahierend überarbeiteten) Moduln in die persönliche System-Bibliothek des Komponisten.

Drittens, durch Übereignung dieser Moduln an die auch den Kollegen zur Verfügung stehende, systemweite Bibliothek, wird eine Kommunikation *zwischen den Komponisten* im Medium des konkreten algorithmischen Materials ermöglicht[12].

[12]Die angestrebte weitgehend freie Verwendung unterschiedlicher Verfahren in verschiedensten Kontexten bedeutet keinesfalls eine Unterstützung der leider zunehmend anzutreffenden „stilistischen Supermarkt-Mentalität": man bedient sich mal hier, mal da, und kombiniert, was einem gefällt.

Dahingehende pädagogische Befürchtungen werden allerdings dadurch beruhigt, daß der *Zwang zu ökonomischem Verhalten* durchaus positive, nämlich konzentrationsfördernde Effekte verursacht.

5.6 Zu erwartender Nutzen des Einsatzes einer Integrierenden Digitalen Arbeitsumgebung.

Hinausgehend über die grundsätzliche Notwendigkeit einer IDA für die schnelle Implementierung von Benutzersprachen und die Verbesserung der dem Digitalrechner an sich eigenen Verwendungsvorteile, erkennen wir eine Reihe von speziellen Neuerungen, die nur in einer IDA sinnvoll unterstützt werden können. Die damit verbundenen Produktionsvorteile sind ...

- [Qualitativ nichts neues, aber *quantitative Erleichterung* durch automatische (integriert gesteuerte) *Datenverwaltung*:]

  Zwei Arten von Systembestandteilen tragen hauptsächlich Funktionalitäten dieser Kategorie : Die *Enzyklopädischen Bibliotheksmoduln* und die *Versionsverwaltung*.

  Erstere können beinhalten Datensätze aus allen Teilen des musikhistorisch erschlossenen corpus, – Webernsche Reihen, BACHsche Formpläne, ja ganze WAGNERsche Opern könnten so „online" in Analyse und anschließende Struktursynthese einfließen.

  Einfachere Beispiele für derart enzyklopädische Moduln sind eine „Interaktive Instrumentenkunde", bis hin zum „Instrumentations-Expertensystem", – Stimmungstabellen mit historischen Anmerkungen, – ein Lexikon der Vortragsbezeichnungen, – das aktuelle Vorwahlverzeichnis ...

  Die *Versionsverwaltung* müsste sämtliche strukturgenerierenden Modulaufrufe mitsamt deren (nicht-trivialen) Parametrisierungen protokollieren, – die dabei generierten neuen Strukturen mit dieser Information vernküpfen, – alte Versione rekonstruierbar halten, – automatisches redo ganzer Produktionsstränge mit leicht edierten Parametersätzen ermöglichen etc.

- [*Qualitativ Neues* durch quasi „materielle" Darstellung des *gesamten* Bestimmungsnetzwerkes im Rechner:]

  Die IDA könnte in ihrer Endausbaustufe die (oder: eine) vollständige Materialisierung des Bestimmungsnetzwerkes enthalten, *inklusive* der *Meta*-Bestimmungen !

  Darüber hinaus könnte sie auch die dynamische Geschichte des Bestimmungsnetzwerkes während der verschiedenen Arbeitsphasen protokollieren, also die Folge von Aufbau, Abbau und Revision der Einzelbestimmungen und Verknüpfungen, die Abarbeitungsreihenfolge der Offenene Entscheidungen, die Auswertungslauf-Ergebnisse, die resultierenden Revisionen der Parametersätze etc.

  Dadurch wäre (1) eine (nur theoretisch!) vollständige *Dokumentierbarkeit* aller Materialbildungsschritte und Entscheidungen möglich, und (2) könnte die Integrierende Digitale Arbeitsumgebung, da sie ja tatsächlich *alles* „mitbekäme", sich zu einem *ideal lernenden System* entwickeln.

- [*Qualitativ neue* Anregungen durch Rekombinierbarkeit von Algorithmen:]

  Die angestrebte (möglichst) *freie und problemlose Übertragbarkeit* (s.u.) von Algorithmen zwischen verschiedenen zu generierenden musikalischen Parametern oder Strukturebenen kann experimentelle Situationen schaffen:

  Die konkreten Ergebnisse der Auswertungsläufe von aus rein intutiven Gründen re-kombinierten Algorithmen sind oft *nicht im vorhinein* kalkulierbar (= „Blinde Verschaltung").

  Wenn z.B. ich einen mühsam zwecks Tonhöhenorganisation definierten strukturgenerierenden Algorithmus mit wenigen Zeilen Adaptercode auf Instrumentation oder Formteilproportionen übertragen kann[13], kann die Rezeptive Beurteilung der Auswertungsergebnisse dieser (in gewissem Sinne „zufälligen") Maßnahme unter günstigen Umständen (1) für langezeit als quälend unlösbar scheinende Probleme schlagartig Lösungsmöglichkeiten aufzeigen, oder sogar (2) akzeptable Lösungen quasi von selbst generieren, und damit (3) durch Wiederverwendung desselben Verfahrens in unterschiedlichen Positionen des Bestimmungsraumes die innere Konsistenz der Struktur und damit die ästhetische Wirkmächtigkeit des Werkes durchaus erhöhen.

- [*Qualitativ neue* Vefahren der Strukturdefinition durch Kombinierbarkeit von Paradigmen:]

  Sehr unterschiedliche Komponierparadigmen der prae-digitalen Elektronischen Musik mit jeweils ursprünglich ganz anders strukturierten Eingabensprachen und historisch aus verschiedenartigsten Traditionen stammend, werden bei ihrer *rechnerinternen Darstellung* mit grundsätzlich kompatiblen Datentypen modelliert.

  - „musique concrète"(mit herkömmlicherweise der „Eingabesprache" : Bandschnitt, Filtrierung, Verhallung etc.),
  - (Post-)Serialismus (mit herkömmlicherweise der „Eingabesprache" : Permutationstabellen, numerische Algorithmen etc.),
  - und „voltage-control" (mit herkömmlicherweise der „Eingabesprache" : Strippen-Ziehen und Regler-Drehen)

  ... werden so neuerdings kombinierbar :

  - Die Schnittpläne für eine musique-concrète-Operation können seriell berechnet werden.
  - VC-ähnliche Verschaltungen können serielle Indexreihen generieren.
  - Serielle Permutationsalgorithmen können signalverarbeitende Schaltungen zusammenbauen („Blinder Algorithmus"!).

[13] Dies ist verwandt mit der Arbeitsweise des „Gezielten Fehlgebrauchs", siehe unten 8.2.

– Ergebnisfunktionen von Spektralanalysen (Verfahren im Sinne der musique-concrète) können serielle Generatoralgorithmen steuern.

– etc.

Diese Kombinabilität schafft große neue Freiräume. Daß deren Verfügbarkeit nicht in Beliebigkeit verunklart, bewirken u.a. alle die o.e. Randbedingungen des Einsatzes des Digitalrechners, die zu Vereinheitlichung und Vereinfachung zwingen.

5.7 Anforderungen an eine zu entwickelnde IDA.

Zunächst sei hingewiesen, daß schon die Bezeichnung mit dem Partizip *Präsens* andeuten soll, daß es sich bei einer Integrier*enden* Digitalen Arbeitsumgebung um ein

- offenes (= hinreichend spezifiziertes und dokumentiertes),
- erweiterbares,
- lernfähiges
- und adaptierbares

System handeln soll, dessen Hauptcharakteristikum eine möglichst weitgehende *Ideologie-Neutralität* sein soll.

Mit diesem Schlagwort kann die u.E. *zentrale* Anforderung an jede zukunftsorientierte Rechnerarchitektur griffig bezeichnet werden :

Kein Kompositionsverfahren, keine Art der Materialorganisation, keine Art, Musik zu denken, sollte für die Arbeit mit einem solchen System vorausgesetzt werden; vielmehr sollten sowohl die unterschiedlichsten Verfahren und *Auffassungen des Materials* gleichberechtigt koexistieren können und neue, auch sehr persönliche Ansätze jederzeit möglichst einfach dem System hinzugefügt werden können.

Genau das gegenteilige Verhalten z.B. zu einem so primitiven Konstrukt wie dem Midi-Standard ist notwendig: Die unhinterfragte Verwendung der Klaviertasten als Stellvertreter für Notation oder die Voraussetzung der Zeitorganisation als einer westlich-metrischen sind abzulehnende Beispiele unzulässiger Vorgaben. Derartige Einschränkungen sind in *doppelter* Hinsicht unzureichend: Sie übersehen, daß es außerhalb dieser Formalismen sehr viele und gleich elaborierte andere Formalismen gibt, welche für die entsprechenden Kontexte durchaus passender sein könnten, - und sie leugnen, daß die von ihnen scheinbar unterstützten Notationsystem hinter dem scheinbar uniformen Vordergrund in Wirklichkeit durchaus verschiedenartigste Mittelgrundstrukturen darstellen sollen.

Um eine möglichst weitgehend *Ideologieneutralität* zu unterstützen, können folgende Richtlinien für die Architektur der technologisch betimmten Systembestandteile sinnvoll sein:

Es sollten (1) immer *verschiedenartigste* Basisalgebren für die Modellierung eines Problembereiches (z.B. von Tonhöhensystemen) parallel implementiert werden (Funktionalharmonisch, nicht-temperiert, indisch, physikalisch, instrumentenspezifische Tabulatur, postseriell-abstrakt etc.).

Diese sollten (2) in möglichst allen Kontexten (dritten Moduln) gleichberechtigt und unter ähnlich niedrigem Aufwand instantiierbar sein.

Wo immer möglich sollten (3) automatische Konvertierungen zwischen diesen Basisalgebren die Verwendung in verschiedenen Anwendungskontexten dem Benutzer möglichst erleichtern.

Ein zunächst speziell erscheinendes, aber für den grundsätzlichen Umgang mit einem jedem Rechnersystem in der Praxis doch zutiefst bestimmendes Problem ist das sog. „Lizenz-Problem", welches zur Verdeutlichung der Richtung von notwendiger Diskussion und wohl anzuwendender algebraisch/ergonomischer Analysmethode im folgenden Abschnitt ausführlicher vorgestellt wird.

5.7.1 Das Lizenzproblem. Der Punktuelle Durchgriff.

Ein zentrales Problem beim Einsatz des Digitalrechners in der ästhetischen Produktion ist die unbedingt notwendige Ermöglichung der Arbeitstechnik des *Direkten Durchgriffs*, auch „Lizenz" genannt.

Damit ist folgende typischerweise oft auftretende Situation gemeint:

Ein Regelsystem dient (entsprechend parametrisiert) als *Generator-Algorithmus* und liefert z.B. ein diachron aufgefaltetes Resultat, wie eine Tonhöhenfolge oder eine „Melodie" (= rhythmisch/diastematische Gestalt).

Der Komponist ist nun mit dem Ergebnis für seine konkreten Verwendungszwecke recht zufrieden, will aber an *einer einzigen Stelle* eine *punktuelle Korrektur* anbringen, z.B. eine Tonhöhe ändern.

In der „alten Satztechnik" könnte der Generator-Algorithmus z.B. das Kanonprinzip sein, parametrisiert mit der ersten Stimme und den Einsatzabständen. Eine *nach Auswertung* des Kanons zum mehrstimmigen Satz in einer einzelnen seiner Stimmen vorgenommene derartige punktuelle Korrektur nennt man in diesem Zusammenhang „Lizenz", weil dies streng genommen eine Verletzung der Kanon-Regel (sic verbum licet!) darstellt und man dafür, zumindest als Kontrapunkt-Schüler, eine „Genehmigung" braucht.

Dies übertragen auf eine rechner-internen Realisierung treten zwei Probleme auf, von denen das zweite tatsächlich schwierig zu lösen ist.

Erstens :

Der Benutzer muß stets (nicht allzu umständlich zu bedienende) technische Mittel haben, derartige Direkten Durchgriffe auszuführen. Die Ergebnisse von Evaluierungsläufen sollen also in Datentypen repräsentiert werden, für welche das System zumindest low-level browser und Editoren bereithält, – besser noch werden diese ergänzt um *regel- und musterbasierte* Kommandosprachen, welche komplexere Datenkorrekturen und sinnfälligere Formulierungen erlauben[14]. So sollten Eingaben

[14]Gerade diese Verwendung war eine wichtige Zielvorgabe und Ausgangsvorstellung beim dama-

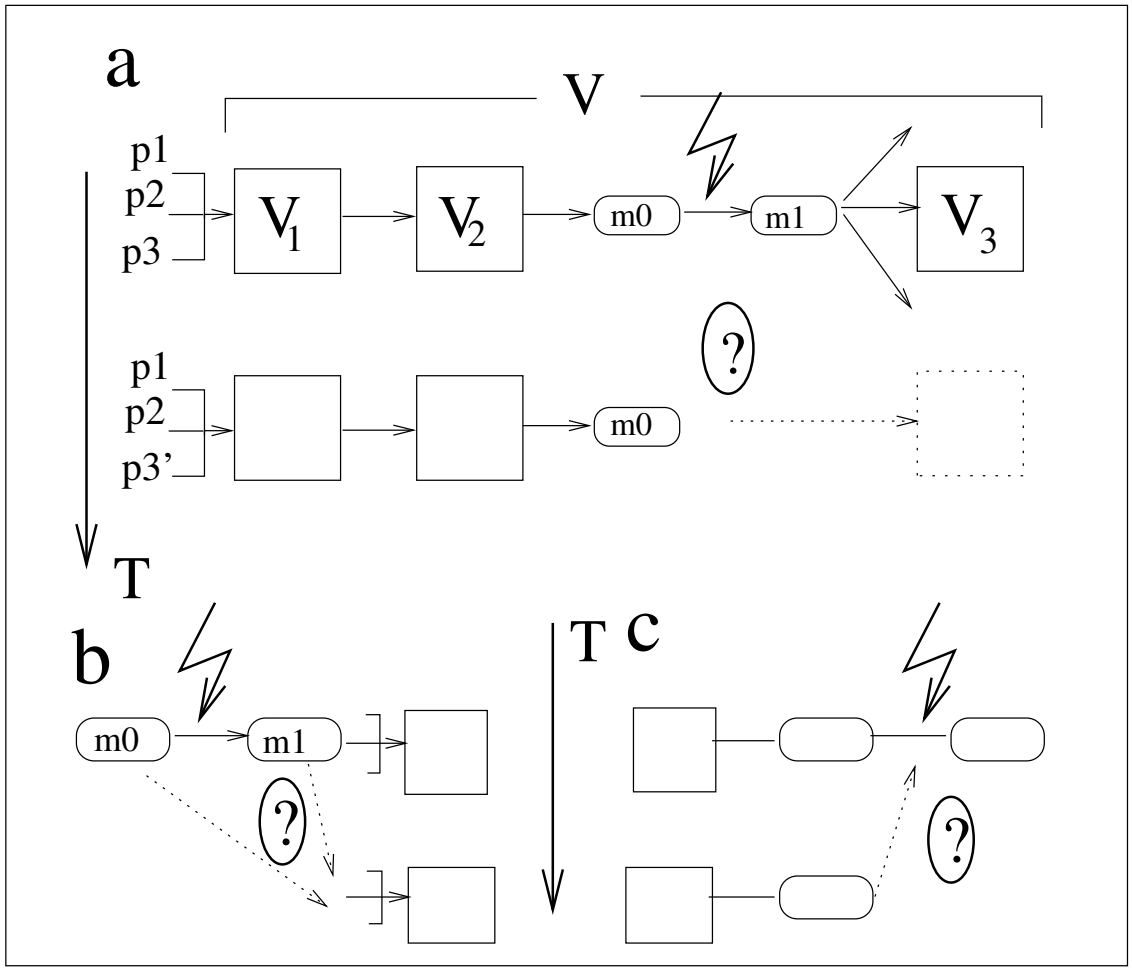


Abbildung 5.2: Direkter Durchgriff und Lizenz-Problem.

wie „Ersetze als Tonhöhen der Klasse `fis`, die nicht auf ein `g` folgen, durch das `f` derselbe Oktave!"möglich sein, anstatt (low-level) in einer grafisch dargestellten Liste alle diese Tonhöhen „anklicken" zu müssen.

Zweitens :

Ein größeres (technologisches und semantisches) Problem ergibt sich im Zusammenhang mit der *Versionsverwaltung* und automatisierter Materialentwicklung :

Angenommen, der komplex zusammengesetzte Verarbeitungsschritt $\mathcal{V}$ aus Bild 5.2-(A) soll im Rahmen des automatischen Replay / der automatischen Versionsgenerierung mit leicht veränderten Parametern (im Bild : p_3' statt p_3) wiederholt werden.

Innerhalb der „alten" Auswertung stand als Resultat von Auswertung V_2 eine diachrone Gestalt m_0, die „seinerzeit" in einem interaktiven Editierungsschritt punktuell und direkt modifiziert wurde[15].

ligen Entwurf der Sprache APOS als Interpreter !

[15] NB: *Typischerweise* können hier psycho-interne Vorstellungs-Wirklichkeit des Benutzers und

Danach wurde mit dem Material m_1 der Verarbeitungsschritt V_3 gestartet. Alle Schritte hintereinander bilden das protokollierte Verarbeitungsnetzwerk $\mathcal{V}$, welches nun re-evaluiert werden soll.

Zur Beantwortung der Frage, wie das System einen interaktiven, punktuellen Durchgriff bei der Re-Evaluierung eines komplexen Verarbeitungsnetzwerkes behandeln soll, werfen wir zunächst einen kurzen Blick auf die möglichen Inhalte von Direkten Durchgriffen, um festzustellen, daß diese grundsätzlich in *zwei unterschiedlichen Darstellungen* vorliegen können, nämlich angesiedelt im Vordergrund oder im Mittelgrund:

Bestehe z.B. unser Material m_0 aus einer Tonklassenfolge ⟨ as, d, g, f, e, cis ⟩.

Der Punktuelle Durchgriff sei geschehen, indem der Benutzer (mittels Anklicken einer grafischen Noten-Darstellung oder mit einem Listen-Editor etc.) das Auftreten der Tonklasse f durch fis ersetzt hat.

Diese *einfache Vordergrund-Aktion* kann nun (womöglich überraschenderweise) *unendlich viele Mittelgrund-Bedeutungen* repräsentieren!

Dieses einfache Austauschen eines Tones z.B. könnte in (der noch rein psycho-intern repräsentierten) Wirk-Lichkeit bedeuten...

- Ersetze jedes f durch ein fis.
- Ersetze das erste f durch ein fis.
- Ersetze den auf g folgenden Ton durch ein fis.
- Ersetze den vierten Ton durch ein fis.
- Ersetze den vierten Ton durch ein fis, falls er auf ein g folgt.
- ... etc.
- Ersetze einen zufällig gewählten Ton durch ein fis.
- Ersetze zufällig viele (≥ 1) zufällig gewählte Töne durch fis.
- ... etc.

Nennen wir den Vordergrund-Edierungsvorgang „Vordergrundeingriff" und die zugrunde liegende Mittelgrundstruktur die „Regeldarstellung" der Lizenz.

Bei geeigneter Wahl der Infra-Sprache könnten aber auch diese Mittelgrund-Informationen durchaus von der psycho-internen in die rechner-interne Darstellung übertragen werden.

---

rechner-interne Objektwelt signifikant *differieren* (was eine gewisse implementatorische Herausforderung mit sich bringt):

Kanonischerweise modellieren wir im Rechner eine solche Modifikation eines Objektes m_0 durch die Erstellung eines *völlig neuen* Objektes m_1 als Ergebnis eines Auswertungsprozesses des (trivialen) Algorithmus namens „Edieren", – der Benutzer hat jedoch in den meisten Fällen vielmehr die praktische Vorstellung, „dasselbe" Objekt „verändert" zu haben, d.h. die Vorstellung von *gleichbleibender* Identität bei verändertem Zustand, – wobei er dennoch auf den „alten" Zustand ggfls. zurückgreifen will, so daß die Identifizierung von m_0 und m_1 über ein *zweidimensionales* Koordinatensystem geschieht, dessen eine Achse die Objektidentität ist, und dessen zweite Achse die *Arbeitszeit*.

Ein *mögliches Systemverhalten* zur Modellierung des Lizenz-Problems könnte also sein :

- Jeder Edierungsvorgang (punktueller Durchgriff, Vordergrundeingriff) wird wie alle anderen Auswertungsschritte standardmäßig protokolliert.

- Jeder durch Edieren ausgeführte, protokollierte Vordergrundeingriff kann zusätzlich „klassifiziert" werden. Jeder neu protokollierte Vordergrundeingriff ist zunächst „unklassifiziert", – wenn der Benutzer nicht sofort eine Klassifikation gemäß unten beschriebenem Raster vornehmen will, um die spätere interaktive Abfrage zu vermeiden.

- *Immer*, wenn ein Auswertungsschritt / ein Netzwerk von Auswertungsschritten (mit modifizierten Parametern) re-evaluiert werden soll (= Versionsgenerierung), zu dessen Eingabeparametern ein (seinerzeit) durch Vordergrundeingriff modifiziertes Material gehort (Bild 5.2 - (B)), *oder* in dessen Verlauf mittendrin ein Vordergrundeingriff des Benutzers stattfand (Bild 5.2 - (A)), *oder* dessen Ausgangsresultat ausschließlich in punktuell modifizierter Gestalt weiterverwendet wird (Bild 5.2 - (C)), werden *alle noch nicht klassifizierten* von diesen Vordergrundeingriffen gemäß folgendem Schritt behandelt.

- Für die so gefundenen und noch nicht klassifizierten Vordergrundeingriffe betritt das Rechnersystem, ehe es den Re-Evaluierungslauf des Netzwerkes beginnt, einen *interaktiven Modus* und fordert den Benutzer auf, die Vordergrundeingriffe zu klassifizieren.

  Die „Klassifikation" einer Vordergrundaktin weist das System an, wie es bei allen zukünftigen Re-Generierungen der von ihr betroffenen Materialien diese zu beachten habe.

  Der Benutzer kann grundsätzlich zwischen folgenden Verhaltensweisen wählen:

  – Ignorieren
    Die Einzelaktion (= der Direkte Durchgriff) bezog sich tatsächlich nur auf die spezifischen Verhältnisse des Ergebnisses des *damaligen* Auswertungslaufes und soll für alle anderen (die aktuelle und die zukünftigen) Auswertungen keinerlei Bedeutung haben.

  – Abstrahieren
    Für die seinerzeit nur mittels Mausklick als Einzelaktion eingegebene Modifikation wird vom Benutzer nun die *Regel-Darstellung nachgereicht*. Diese Regeldarstellung soll in Zukunft jedesmal automatisch auf alle neugenerierten Materialien (im entsprechendem Kontext) angewendet werden.

    Das System sollte dabei stets überprüfen, ob die angegebene Regeldarstellung tatsächlich zu derselben Modifikation des Materials führt wie die seinerzeitige Einzelaktion.

– Wörtlich Übernehmen
Die Einzelaktion soll als Regel aufgefaßt werden, resp., die punktuelle Modifikation geschah seinerzeit schon durch einen Regel-Ausdruck. Dieser soll jedesmal „wörtlich" übernommen werden.

– Weiterhin Nachfragen
Direkte Durchgriffe werden voraussichtlicherweise wegen der Art des Materials/der Generierung auch zukünftig bei jeder Wieder-Auswertung des Transformationsnetzwerkes *jeweils spezifisch* nötig sein.

Durch diese Klassifizierung wird das System angewiesen, alle zukünftigen automatischen Re-Evaluierungen des Netzwerkes nach der Generierung (und/oder vor der Verwendung) des Materials, welches positionell dem damals punktuell modifizierten Material entspricht, kurz anzuhalten, und vorübergehend in einen interaktiven Modus zu gehen, welcher dem Benutzer erlaubt, die den damaligen punktuellen Korrekturen entsprechenden neuen Korrekturen anzubringen, voraufhin die automatische Auswertung fortgesetzt wird.

5.8 Forderungen an Infrasprachen.

Da die Infra-Sprache möglichst sowohl zur Implementierung des Basissystems der IDA verwendet werden soll, als auch zur Realisierung der diversen Benutzersprachen, ergeben sich zunächst natürlicherweise eine Reihe von *rein technologischen* Anforderungen[16]

Diese seien hier der Vollständigkeit halber kommentarlos genannt:

1. persistency.
2. multi-threading.
3. inter-language-working.
4. strong typing.
5. high orderness.
6. reflection.
7. polymorphism.
8. real time capability.
9. garbage collection.

Darüberhinaus ergeben sich aus unseren Strukturanalysen einige weitergehende wünschenswerte Eigenschaften:

- Durchgriffs- und Lizenzmöglichkeit, Austauschbarkeit und Uniformität der Sprachmittel auf allen Systemebenen, – dadurch potentiell monolithische Gesamtarchitektur.

[16]Der Verfasser dankt hier besonders Herrn Dipl. Inf. Wolfgang Grieskamp, Berlin. Unsere umfassenden Diskussionen waren nicht nur für diese technologischen Erkenntnisse unverzichtbare Voraussetzung, sondern trugen auch zur Klärung der ästhetischen Zusammenhänge Entscheidendes bei.

- Aber auch : Saubere Trennbarkeit von Arbeitsphasen und -zielen[17].

- Gleichwertige Unterstützung verschiedener *Programmierparadigmen*:

  Da ja (1) unterschiedlichste Probleme mit einem solchen System modelliert werden sollen, und da (2) verschiedene Benutzer und Programmentwickler für dessen Wachsen arbeiten sollen, wäre es höchst förderlich, eine gewisse *Bandbreite von Pragrammierstilen* gleichberechtigt zu erlauben : von regelbasiert/logisch über pur-funktional bis hin zu imperativ/objektorientiert.

5.9 Entwicklung einer Integrierenden Digitalen Arbeitsumgebung als gemeinsames Anliegen einer globalen Pluralität von Komponisten – das IDA-Manifest.

Alle seinerzeit am AUDIAC-Projekt Beteiligten waren (wir vermuten: sind immer noch) der Überzeugung, daß eine zukunftsträchtige Weiterentwicklung des Rechnereinsatzes in der (anspruchsvollen, experimentellen, ästhetisch engagierten) musikalischen Produktion *ausschließlich* durch die Entwicklung von *Integrierenden* Digitalen Arbeitsumgebungen möglich sein wird.

Den inhaltlichen Nutzen einer solchen Arbeitsumgebung mögen die vorangehenden Abschnitte wohl unbestreitbar dargetan haben.

Weiterhin behaupten wir, daß die Entwicklung einer solchen IDA nur durch die gemeinsame, koordinierte und vorurteilslose Kooperation möglichst vieler Interessierter und verschiedenartig Befähigter aus möglichst unterschiedlichen „Lagern" und Arbeitsbereichen durchführbar sein wird.

Drei Hinweise mögen diese Behauptung stützen:

- Die aus der Anwendung von Einzelprogrammen resultierenden Erleichterungen und auch erstmals überhaupt möglichen Syntheseverfahren (s.o., 5.1) bringen oftmals, – gerade bei Verwendung der inzwischen kommerziell gewordenen Produkten und proprietären Sub-Standards, – neuartige Erschwernisse und Einschränkungen mit sich.

  Diese gilt es zu kompensieren, ja zu bekämpfen, damit nicht neu eingeführte Zwange der Verfahrenstechnik und Engpässe im Informationsfluß ästheti-

[17] Ein wesentliches Manko der TEX/LATEX-Architektur ist ja gerade, daß (entgegen der ursprünglich geforderten Trennung von generischem MarkUp und Darstellung) die Behandlung der formalen Text-Struktur (in unserer Terminologie: des Mittelgrundes), die Kontrolle des graphischen Erscheinungsbildes *und zusätzlich* die zur Realisierung eingesetzten Programmierparadigmen ein fürchterliches, unentwirrbares Kuddel-Muddel bilden: Es kann z.B. sein, daß Schlüsselworte, die die *grafische Erscheinung* beinflussen sollen, in einigen *syntaktischen* Zusammenhängen, welche die *logische Textstruktur* realisieren sollen, korrekt ausgewertet werden, in anderen jedoch alles kaputthauen!

sche Entscheidungen überlagern. (= „Formbildende Tendenzen des Materials!", deutliches Beispiel : die impliziten forget-Funktoren !).

- Auch in den kommerziellen Betriebssystem-Umgebungen herrscht der Trend zu *integrierenden* Kommunikationsmechanismen vor (z.B. HTML/SGML, OLE, GNU-Arbeitsumgebung, microsoft office-Pakete etc.).

  Die Gemeinde der künstlerisch anspruchsvollen Software-Benutzer und -Entwickler kann es sich *weder* erlauben, diese (in ganz anderen Zusammenhängen und für andersartige Anwendungen entwickelten) Kommunikationstrukturen kritiklos zu übernehmen und so eine weitere Präformierung, ja Deformierung des ästhetischen Materials zu riskieren, – *noch* aber andererseits ihre normbildende Marktpotenz schlicht zu ignorieren und auf ihre unbestreitbare Leistungsfähigkeit gänzlich zu verzichten.

- Das technologische Niveau der – inhaltlich allerdings weiterhin primären, ja primitiven – Systemebenen (Datenrepräsentation und -kommunikation, GUI-Gestaltung, Komplexität der Treiberschalen etc.) hat sich in den letzten Jahren (wegen der Fokussierung der kommerziellen Aktivitäten auf derart verkaufsfördernde Oberflächenphänomene) gewaltig gehoben.

  Engagierte Privat-Entwickler (Komponisten, Hochschulen, Forschungsprojekte) sind nicht mehr in der Lage, – und es ist auch nicht deren Aufgabe –, auf diesen Gebieten mit der Industrie mitzuhalten.

  Nur eine ausgeweitete *Wiederverwendbarkeit* generischer und erweiterbarer Systemmoduln kann eine zeitgemäße Entwicklungsarbeit unter experimentellen Vorzeichen noch zum Erfolge führen, insoweit sie ermöglichen würde, diese auf die zentralen Inhalte zu konzentrieren.

Die Entwicklung einer Integrierenden Digitalen Arbeitsumgebung für künstlerisch engagiertes, zeitgemäßes Komponieren muß also erfolgen unter der Zielvorstellung größtmöglicher Offenheit, Erweiterbarkeit, Durchlässigkeit und Ideologieneutralität, – gleichzeitig sind anzustreben Schnittstellen zu den marktbeherrschenden de-facto-Standards, welche die Enge der übermittelbaren Datenstrukturen und die looseness der Typsysteme möglichst kompensieren, – all dies muß geschehen nach gründlicher Analyse der benötigten Infra-Strukturen unter weitest möglicher Abstraktion zwecks Offenhaltung (prinzipiell) beliebiger Neudefinitionen des musikalisch/kompositorischen Materials in der Zukunft.

Da weder große Gewinnerwartung noch breite Werbewirkung von Investition in oder Unterstützung für ein solches Projekt zu erwarten sind, – ja es den proprietären Industrieinteressen teilweise durchaus zuwider läuft, bleiben wir Komponisten auf unsere eigenen (mehr oder weniger schwachen) Mittel zurückverwiesen.

Die Gemeinschaft aller an einer solchen Entwicklung Interessierter kann also nur versuchen, ihre wenigen, weltweit verteilten Kräfte sinnvoll zusammenzuschließen, – so dezentral wie möglich und so koordiniert wie nötig.

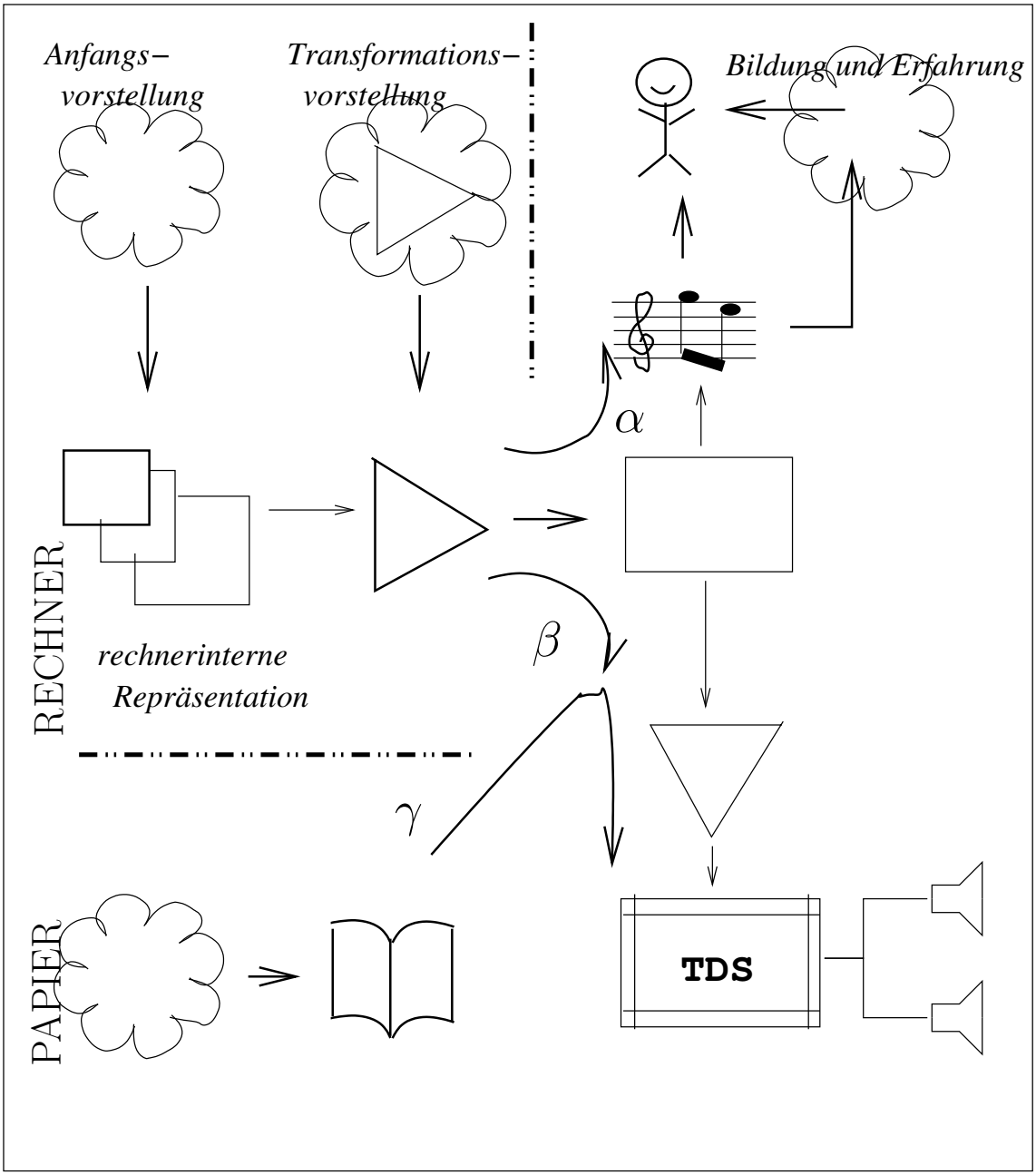


Abbildung 5.3: Verwendungsphasen des Digitalrechners : Mittelgrund-Operationen vs. TDS-Interpretation.

5.10 Konzeptionelle Trennung von Arbeitsumgebung und Instrument – Der Technische Datensatz.

Trotz des grundsätzlich anzustrebenden höchstmöglichen Integriertheit einer wünschenswerten digitalen Arbeitsumgebung (zum Zwecke größtmöglicher Freiheit von Verknüpfung) sollten auf der *konzeptuellen* Ebene zunächst einmal *sauber getrennt* werden (um danach evtl. gegenseitige Beeinflussung und Schnittmengen zu untersuchen) *zwei wesentlich unabhängige* und unterschiedliche Einsatzbereiche des Rechners innerhalb ästhetischer Produktion (siehe Bild 5.3):

Zunächst dient der Rechner als Partner im Prozeß des eigentlichen Komponierens. Unter den o.e. Rahmenbedingungen der Sinnfälligkeit kooperieren Komponist und Algorithmen auf Mittelgrundstrukturen, welche gleichzeitig im Bewußtsein des Menschen und im Wissen des Rechners repräsentiert sind.

Das Ergebnis dieser Arbeitsphase ist eine (evtl. zufälligerweise nur rechnerintern repräsentierte) *Vordergrund*struktur. Diese kann nun (α) externalisiert werden, – z.B. als Vordergrundstruktur eines gedruckten Notentextes an einen menschlichen Interpreten abgegeben werden, oder (β) im Rechner verbleiben und vom *Rechner als Instrument* abgespielt und klanglich realisiert werden.

Diese zweite Verwendung, – der Rechner als Interpret, Dirigent, Tonmeister oder Instrument – ist von der ersten a priori prinzipiell völlig trennbar; auch der umgekehrte Fall (γ) ist immer noch anzutreffen: Ein Stück wird zu Fuß komponiert und dann, Note für Note, zwecks „sequencing" o.ä. in den Rechner eingegeben.

Mag sein, daß auch der Rechner eine Form von Bewußtsein hat oder haben wird, – dies ist sowohl Definitions- als auch Glaubensfrage, – daß der uns entgegentretende *Mensch* über Bewußtsein verfügt, ist schließlich ebenfalls nur eine Hypothese.

Allemal jedoch hat der Rechner eine *Innere Sprache*, genauer: gleich dem Menschen eine Schichtung von verschiedenartigen Inneren Sprachen unterschiedlicher Differenzierung und Spezialisierung.

Ausgehend von den primitiven, linearen Datentypen (Bits und Bytes, Midi-Codierung und verschrobene Zahlenformate) über Automaten- und Prozeß-Algebren bis hin zu typisierten Kalkülen höhere Ordnung operiert jeder Rechner mit ganzen Ensembles (und Schichtungen) „Innerer Sprachen", – die uns aber nicht interessieren (können). Diese sind uns als Totalität natürlich nicht nachvollziehbar[18] und sind, ähnlich wie die Inneren Sprachen eines anderen Menschen, *als solche* nicht kommunizierbar, –allenfalls qua Modellbildung.

Während jedoch beim Abgeben eines (auch rechnergenerierten) Notentextes an einen „passenden", d.h. entsprechend geschulten Interpreten die komplizierte Gesamtheit der historisch vermittelten Mechanismen der musikalischen Notation die Semantik des Werkes jedenfalls teilweise kommunikabel macht, *fehlt* dieser Hintergrund beim Weg (β) selbstverständlich *völlig*.

[18] Interessanterweise beschäftigt sich ja ein großer Bereich der wissenschaftlichen Informatik damit, das Verhalten von Systemen zu *erforschen*, welche man eben selbst erst konstruiert hat.

Der Rechner als Interpret verfügt nicht über eine Innere Sprache, wie die des Musikers, welche (zwar nicht mitteilbar, aber) an der Erfahrung musikalischer Wirklichkeit zutiefst geschult ist.

Eine jede *explizite programmtechnische Realisierung* dieser *zweiten* Stufe von Transformationen muß im Fall (β) die Semantikübermittlung durch die historische Konvention von Notation und Instrumentaltradition (in Fall α) *vertreten* (Ein prägnantes Beispiel, auch für die in diesem Zusammenhang besonders auffällige *Realisierung Autarker Skalare*, s.u. 6.7.6, findet sich in der folgenden Fallstudie in 7.4.1).

Die unreflektierte Vermischung dieser zweiten Stufe von Transformation mit der komponierenden Mittelgrund-Transformation ist es, der wir oben entgegentraten. In der Tat ist das Verhältnis beider keinesfalls unproblematisch und unbemerkt können Rücksichtnahmen auf technische Probleme in der zweiten Transformationsstufe in die vorhergehende Materialdefinition und -behandlung einsickern.

Nützlich sind solche Interdependenzen, wenn sie bewußt als Materialbegriff thematisiert werden, – schädlich, wenn unbewußte Vorwegnahme realisationstechnischer Idiosynkrasien zur Verunklarung des eigentlichen Strukturbegriffes führt.

Das Ergebnis dieser zweiten Transformation nennen wir *Technischen Datensatz*, kurz TDS.

Ein TDS ist immer ein Ensemble von technologisch determinierten Datensätzen in den *jeweils verschiedenen Eingabe-Sprachen* aller an einer Session beteiligten Klang- oder Signalverarbeitenden Einheiten, seien es rechnerinterne (Realzeit-)Algorithmen oder externe Geräte (*Fremdgeräte*) wie nachgeschaltete Midi-gesteuerte Hallgeräte, Mischpulte, Sequenzer oder Lichtpulte, Bühneneffektoren etc.

Auch die in Bild 5.3, Weg (α) entstehenden Notentexte, z.B. für Sängerin oder Violine, können in weiterem Sinne als Technische Datensätze betrachtet werden, wenn man es übers Herz bringt, Sängerin und Violinspieler als „Fremdgerät" zu bezeichnen.

5.11 Das AUDIAC-Projekt. Die APOS-Sprache.

Das AUDIAC-Projekt war der (naturgemäß nur partiell gelungene) Versuch der Konstruktion einer Integrierenden Digitalen Arbeitsumgebung.

Es wurde initiiert wurd Dr. Helmut Zander und Prof. Dirk Reith und durchgeführt hauptsächlich von Dipl.Ing. Gerhard Kümmel, Thomas Neuhaus, Roland Borrmann, dem Verfasser und vielen anderen, – finanziert mit Mitteln von Bund, Land NRW und Sponsoren.

Entwickelt wurde die AUDIAC-Signalprozessor-Hardware und die AUDIAC-Software-Architektur als Annäherung an eine Integrierende Digitale Arbeitsumgebung.

Als (Annäherung an eine) Infra-Sprache diente APOS. APOS ist eine eigens entwickelte, auf den Erfahrungen mit smalltalk, FORTH etc. aufsetzende, musterorientierte multi-method-Programmiersprache.

Ihre Hauptdesignziele waren : (1) Uniforme Sprachverwendung auf ganz unterschiedlichen Ebenen der Monolithische Systemarchitektur, von intelligenter und adaptierbarer Benutzerschnittstelle bis hinab zur Manipulation von Hardware-Registern auf Bit-Ebene. (2) Unterstützung von rapid prototyping zu Erleichterung interaktiver Experimente von Benutzer und Entwickler[19].

Im Rahmen des AUDIAC-Projektes wurden mit Komponisten unterschiedlicher Ausrichtung und Qualifikation durchaus verschiedenartige kleinere Werke realisiert.

Die künstlerische, didaktische und programmiertechnische Arbeit bewegte sich jeweils in anderen der (zu Beginn dieses Kapitels aufgeführten) Anwendungs- und Problembereiche. Die dabei aufgestellten Lösungsansätze und erzielten (Teil-)Erfolge waren aber stets auf das Wachsen und Zusammenwachsen des Gesamtsystems als Integrierende Digitale Arbeitsumgebung hin angelegt.

In den beiden folgenden Kapiteln werden wir durch Vorstellung eines dieser Projekte versuchen, die bis hierhin eingeführte Begrifflichkeit an konkreten Beispielen anschaulich(er) zu machen.

[19]Natürlich ist auch APOS aus der Sicht professionellen Sprachdesigns lediglich ein „hack", aber eine etwas weniger grausame Ausgeburt als tcl oder LaTeX. Allemal jedoch sollte ein Programmentwickler die dünne Grenze zwischen „rapid prototyping" und „fast & dirty" beachten.

Kapitel 6

Fallstudie: Die Benutzersprache des Projektes „PGP".

6.1 Kontext des Projektes PGP.

Der Planung nach sollte das AUDIAC-Projekt PGP[1] auf der Veranstaltung des Institutes für Computermusik und Elektronische Medien im Rahmen der öffentlichen Präsentation aller Kunst- und Musikhochschulen des Landes NRW, genannt „Spektakel 94", im Juni jenes Jahres stattfinden.

Sachfremde Gründe verhinderten seine rechtzeitige Realisierung der vollständigen AUDIAC-Software, so daß der beteiligte Student seine Studie mit anderen Syntheseverfahren realisierte.

Dennoch war die analytische und sythetische Arbeit an der AUDIAC-Version recht weit fortgeschritten, und die weitere Entwicklung recht genau absehbar.

Da diese Version sich sowohl inhaltlich als auch didaktisch durchaus als Beispiel eignet, und da die anderen AUDIAC-Projekte meistenteils ja ihre klangliche Aufführung gefunden haben, so soll auch diese Arbeit nicht vergeblich gewesen sein, indem sie hier einer (hoffentlich instruktiven) Autopsie unterzogen wird.

Der ästhetische Zusammenhang der letzten AUDIAC-Projekte war das künstlerische Entwicklungsvorhaben (sic!) „Klangräume" des Institut für Computermusik und elektronische Medien, und dessen Gegebenheiten waren folglich der Ausgangspunkt der verschiedenen ästhetischen *Analysen* (semantische und soziale, bauphysikalisch/akustische, rezeptionspsychologische etc.), welche in allen Teilprojekten die ersten Schritte der *Materialgewinnung* darstellten.

Es ist also eine kurze Schilderung dieses Zusammenhanges nötig.

[1] Alle bisherigen AUDIAC-Projekte sind nach den Initialen des beteiligten Komponisten benannt, ggfl. mit angehängtem Index, – bis jetzt sind es TNP*n*, FHP0 und FHP1, JGP, THP0 und THP1.

6.2 Exkurs : Das Institut für Computermusik und elektronische Medien beim „Spektakel '94".

Der Beitrag des Institut für Computermusik und elektronische Medien zum „Spektakel '94" stand unter dem Titel „Klangräume". Dieser Titel, ernst genommen, bedeutete eine Auseinandersetzung im Spannungsfeld zwischen *virtuellen* Räumen einerseits, (– worunter man, historisch als Metapher entstanden, aber nicht ohne wahrnehmungstheoretische Begründung, so unterschiedliche Stilmittel zusammenfassen kann wie komponierte Raumverteilungen in der Mehrkanaligkeit, Komponierte Phasenlagen, Tonale Räume, Ereignisräume, Klang-Räume im Sinne von Registerbehandlung, aber auch soziologische und gebrauchshistorische Aspekte von Verwaltungsgebäude und Konzertsaal etc. –) gegenüber dem *physischen* Aufführungsort andererseits, seinen Eigenschaften und Eigenheiten.

Der von uns ausgewählte Raum, die große Halle des sog. „Bierkastens", des großen, überreichlich postmodern geratenen neuen Rathauses von Dortmund, hatte mehr von diesen als von jenen (siehe Bild 6.1).

Während in kleineren, akustisch neutralen Seitenräumen Demonstrationen, Seminare und Tonbandkonzerte liefen, führten wir in der Haupthalle eigens für die dort herrschende Marmor-und-Granit-Akustik ausgewählte größere Werke auf. Diese waren in der Lage, diesen speziellen Raum als Resonator, Klangkörper, Instrument zu benutzen, so daß diese Aufführungen (z.B. „The Bad Boys Were Prodding The Bear Through The Bars Of The Cave" von Thomas Neuhaus) für den, der die Stücke aus aus anderen Kontexten kannte, als paradigmatisch unvergessen bleiben.

Als *dritte* Schiene des umfangreichen Programmes des Institut für Computermusik und elektronische Medien liefen eigens für diesen Raum und diese Situation entwickelte interaktive Installationen und ein interaktives Instrumentalstück, das AUDIAC-Projekt JGP.

An der Gesamtveranstaltung des Institut für Computermusik und elektronische Medien waren alle Mitarbeiter und Studenten, bis hinab zum ersten Semester, in unterschiedlichster Form beteiligt.

An dem erwähnten *speziellen* künstlerischen Entwicklungsvorhaben „Klangräume" arbeiteten unter der Leitung von Prof. Dirk Reith vier Studenten (Eckert, Garavaglia, Gahn und Hansen) und die Lehrbeauftragten Thomas Neuhaus und der Verfasser. Dieses Entwicklungsvorhaben mündete in fünf aufgeführte Projekte.

6.3 Ausgangsidee des Projektes.

In diesem Zusammenhang begann also die Arbeit am Projekt PGP, welches hier vorgestellt werden soll.

Die Grundidee des komponierenden Studenten war ebenso einfach wie überzeugend. Welche Komplikationen sie nach sich zog, und wie lehrreich sie damit ist,

Abbildung 6.1: Die große Halle des Dortmunder Rathauses.

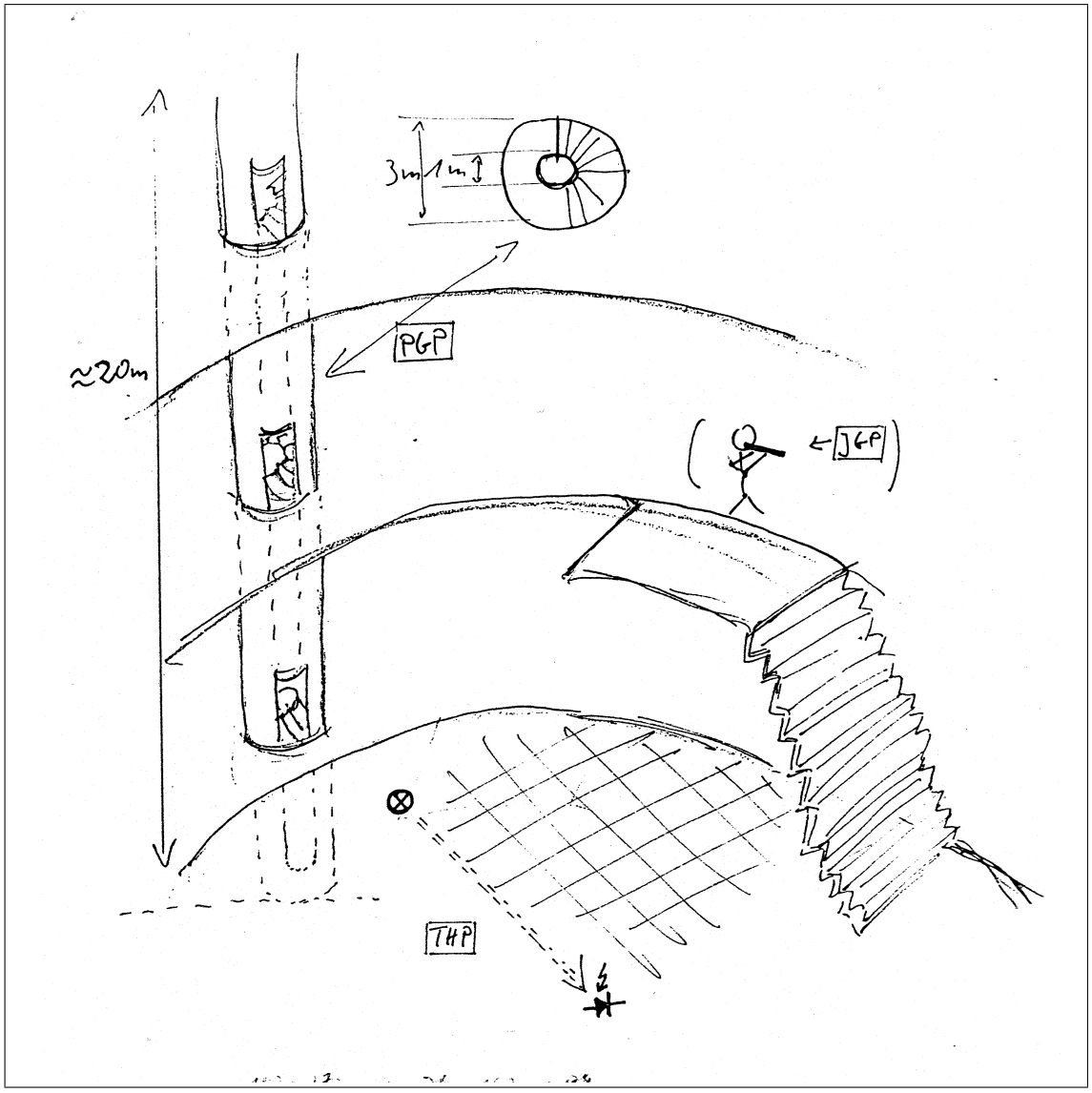


Abbildung 6.2: Schematische Darstellung der Treppenhäuser.

war anfänglich nicht abzuschätzen.

Bei der Besichtigung des zu beschallenden Raumes fielen allen Beteiligten die vier runden, ca. 3 Meter starken und 30 Meter hohen Treppenhausschächte auf (Bild 6.2). Diese sind von der Halle aus als freistehende Röhren sichtbar, welche die Ebenen der Emporen durchstoßen. Sie tragen je eine Wendeltreppe mit Absätzen[2], welche sich an ihrem Innenrand entlangzieht, ohne sich jedoch auszufüllen, sodaß über die ganze Höhe des Gebäudes im Innern der Röhren sich eine freie, oben und unten geschlossene Luftsäule erhebt.

[2]Gerade die im Laufe der Arbeit erst hinzugefügte Modellierung der regelmäßige Alternierung von Absätzen und Stufen stellte sich später als höchst wirkungsvolle Maßnahme zur Erhöhung der rezeptiven Anschaulichkeit heraus! Vgl. 6.7.8.

Der Student nun wollte mittels jeweils oben und unten in diesen Treppenröhren angebrachte Lautsprecher ein Stereosignal einspeisen.

Dieses sollte, und das war die naheliegende, aber nicht folgenlose Idee, *Schritte* von scheinbar durch das Treppenhaus auf- und absteigenden Personen wiedergeben. Man kann auch sagen: aus scheinbar auf- und absteigenden Schrittgeräuschen bestehen – was nicht ganz dasselbe ist.

6.4 Konzeptionelle Entwicklung.

Die Arbeit am Projekt PGP ging, ähnlich wie bei den anderen AUDIAC-Projekten, dreistufig vor sich : Zunächst wurden in der gesamten Arbeitsgruppe vom Studenten die Ideen und Pläne vorgestellt und gemeinsam diskutiert und analysiert, d.h. ggfls. phänomenologisch reduziert und strukturell klarer herausgearbeitet.

Unter dem Einfluß des Verlaufes und der Ergebnisse dieser Diskussion wurde dann vom Studenten das Konzept in Eigenarbeit weiterentwickelt und ausformuliert, um dann wieder vorgestellt und kritisiert zu werden.

Zunehmend wurde das Projekt PGP dann Gegenstand des Hauptfachunterrichtes, und vom Hauptfachlehrer, Herrn Thomas Neuhaus, mitbetreut.

Eine zweite Phase bestand in der eigentlichen *Implementierung*, welche vom Studenten in enger Zusammenarbeit mit dem Verfasser in tagelanger, konkreter Programmentwicklungsarbeit am Rechner durchgeführt wurden.

Programmentwicklungsarbeit bedeutet dabei zunächst Entwurf der Benutzersprache, Implementierung eines Interpreters (einschließlich der von diesem gesteuerten Klanggenerierung), und – natürlich – Debugging.

In einem evolutorischen Ping-Pong geschah aber andererseits auch die Veränderung oder Erweiterung der ursprünglichen Konzepte aufgrund der konkreten klanglichen oder technischen Erfahrungen bei den (an sich erfolgreichen) Probe- oder Produktionsläufen.

Wie immer konnten natürlich nicht alle aus den Ausgangsideen der Studenten herauskristallierten oder -kristallierbaren Ideen auch umgesetzt werden. Wenn die Implementierungsarbeit sich auch auf das primär Notwendige beschränkte, so wurde beim Entwurf jeder Architektur dennoch versucht, spätere Erweiterungen auf das jetzt schon als interessant Erkannte möglichst zu unterstützen.

Die dritte Phase war die (prinzipiell) selbstständige Arbeit des Studenten, der auf der einmal (in Schritt zwei) geschaffenene Sprachplattform dann unterschiedliche Datensätze und Partituren entwickelte, testete, verwarf oder auswählte, modifizierte und kombinierte, wobei er nur noch mit der von ihm *selbst definierten* „Benutzersprache" und wenigen Standardbefehlen beschäftigt war, also sich selbst überlassen werden konnte.

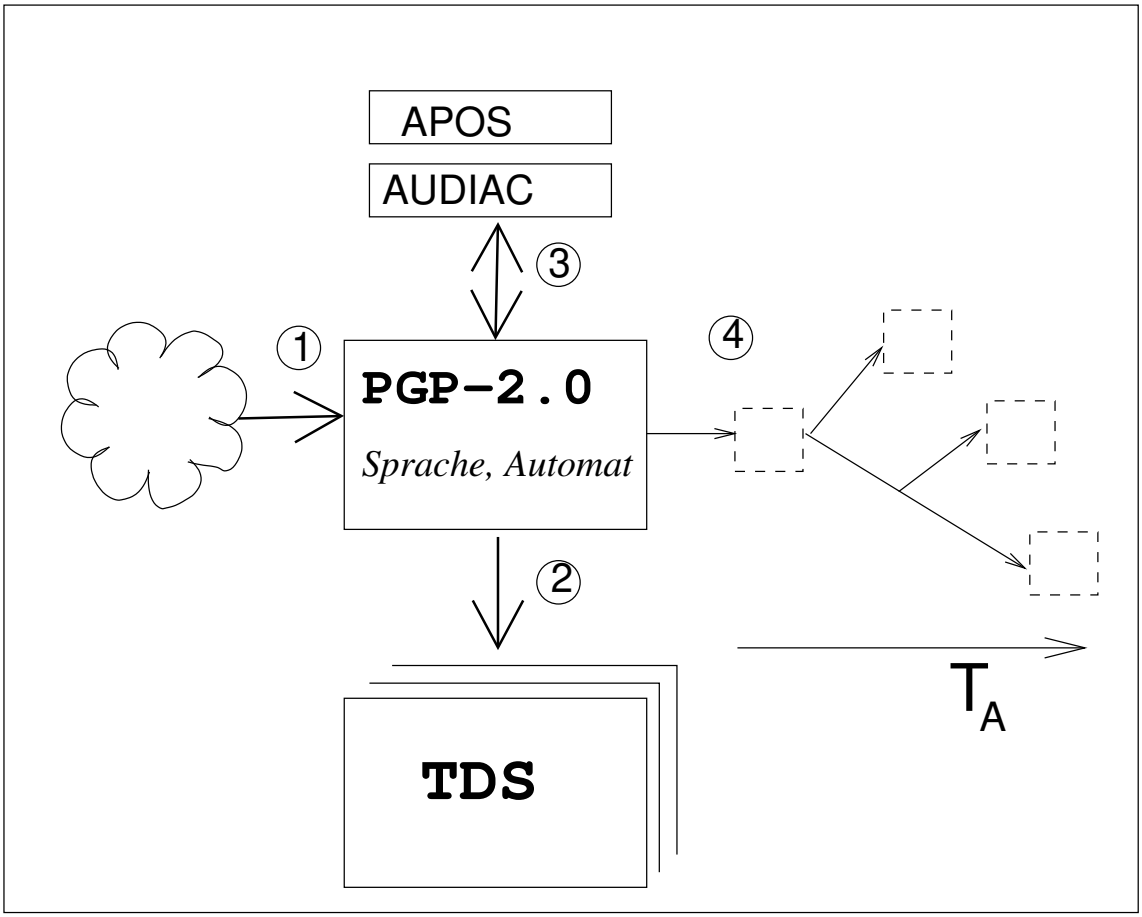


Abbildung 6.3: Einbettungen von und Blicke auf PGP.

6.5 Das Projekt PGP als Sprachentwicklung.

Es wurde behauptet, daß jede kompositorische Tätigkeit, allemal jedoch die mit Hilfe des Rechners, zwangsläufig mit Entwurf und Entwicklung von persönlichen Programmiersprachen zusammenhängt (s.o. 5.2).

Jedenfalls was die AUDIAC-basierten Projekte im „Spektakel '94" angeht, trifft dies größtenteils zu.

Als zentrales Element der rechnergestützten Realisierung des Projektes PGP ergaben sich verschiedene Versionen einer Partitursprache, genannt PGP-0.0, PGP-0.1, PGP-0.2 etc., welche die klangliche Umsetzung von Strukturvorstellungen des Komponisten elegant ermöglichen soll.

Wir werden im folgenden die ersten Versionen dieser Sprache vorstellen.

Im Besonderen werden wir vier verschiedene Blickwinkel auf die „Produkte" der realisierten Software einnehmen (siehe Bild 6.3).

1. Als erstes werden wir den Weg von der Analyse der Vorstellungen zu den ersten Entwurfsdefinitionen darstellen.

2. Dann werden wir einen Blick auf die Implementierung der tatsächlichen klanglichen Realisierungen werfen. Diese geschehen, wie im vorigen Kapitel ausführlich vorbereitet, durch die Formulierung der Transformationen von einem (beliebig) codierten „semantischen", strukturbeschreibendem Quelltext in die für den Betrieb der technischen (Fremd-)Geräte notwendigen Technischen Datensätze („TDS", s.o. Abschnitt 5.10, in Abbildung 6.3 unten), – also durch die Implementierung von Übersetzern.

3. Zum dritten werden wir die Einbettung der entstandenen Sprache in die AUDIAC-Gesamtsoftware, und die daraus folgenden arbeitspraktischen und ästhetischen Konsequenzen betrachten,

4. und zuletzt die Fortentwicklungs- und Erweiterungsmöglichkeiten späterer Versionen andeuten, deren Zweckmäßigkeit und ästhetischen Konsequenzen diskutieren.

6.6 Entwicklung von Materialbegriffen und Syntheseverfahren als Umkehrung einer abstrahierenden Analyse.

In der elektronischen Musik ist es ein Regelfall, – und auch in der konventionellen Schreibweise nicht eben selten, – daß ein später implementierter *technischer Synthesevorgang* (und damit die operationalen Objekte des Mittelgrundes) sich ergeben durch eine *genaue Umkehrung* eines vorhergehenden Analyseprozesses.

An unserem Beispiel läßt sich das schön herausarbeiten. Die Analyse verlief mehrstufig und verlief wie folgt:

Zwei Aussagen, die schon vieles determinieren, waren von Anfang an bekannt : Am Ende der Arbeit steht ein *Stereo-Datenträger*, welcher das „fertige Stück" in einer Aufführung repräsentiert, indem er in den Treppenhausröhren abgespielt wird, der eine Lautsprecher oben, der andere unten.

Dieses Signal besteht aus mehrstimmigen Verläufen von auf- und abwärts steigenden Schrittgeräuschen unterschiedlicher Klanglichkeit.

Ausgehend von diesen recht banalen Grundaussagen werden nun eine Folge von *begriffsbildenden* Abstraktionsschritten vollzogen. Diese führen letztlich zu einem handhabbaren Materialvorrat, d.h einem System von operablen Objekten und Begriffen.

Die klangliche *Realisierung* wird auf der Ebene der Datentransformationen den *Rücklauf* der Abstraktionschritte darstellen, die wir nun auf der Ebene der Modelltransformation vollziehen.

6.6.1 Abstraktion $\alpha \rightarrow \beta$: Auffassung eines akustischen Signals als Summe.

Dieser erste Abstraktionsschritt (siehe Bild 6.4) ist trivialerweise zwingend notwendig und wird in unterschiedlicher Form bei *jeder* Produktion Elektronischer Musik angetroffen : Das analoge Bandsignal (ein elektrisch codierter Schallverlauf) ist nämlich der menschlichen Modellbildung völlig unzugänglich.

Das auf den beiden Spuren eines Stereo-Datenträgers aufgezeichnete auditive Signal kann *als solches* (schon allein wegen zu hoher Datenmengen) vom menschlichen Vorstellungsvermögen nur qua Abstraktion, d.h. Begriffsbildung erfaßt werden.

Die sog. „Neue Musik", also die seriell/postseriell bestimmte Satztechnik, insbesondere aber die sog. „Elektronische Musik", hat dafür eine Reihe von durchaus unterschiedlichen Methoden entwickelt.

Z.B. kann ein auditives Gesamtsignal als elektrische Summe von gleichzeitig ablaufenden Teilprozessen in verschiedenen Frequenzbereichen aufgefaßt, d.h. konstruiert und rezipiert werden, – sozusagen als akustische Schichttorte.

Oder eine klingende Struktur besteht aus verschiedenen gleichzeitig ablaufenden Strukturen teils linearem, teils punktuellem Inhalt, wie ausgeführt von Elena Ungeheuer in (Ungeheuer, 1993).

Oder eine klingende Struktur besteht aus explizit gesetzten Ereignissen und *arbeitstechnisch nachgeschalteter prozessualer Verarbeitung*, siehe Abschnitt **??**, z.B. partiturgesteuert generierte einfache Knack-Pulse, die dann durch eine extreme Verhallung gejagt werden, welche somit quasi zum Syntheseverfahren avanciert.

In unserem Beispiel findet als Zerlegung das Verfahren von „Montage und Mischung" Anwendung. Das heißt, daß das auf den beiden Spuren des Datenträgers vorhandene Signal (in Abb. 6.4 und 6.4 unter α) beschrieben werden kann als die *elektrische Summe*[3] von beliebig, auch überlappend, *montierten Teilsignalen*. Jedes dieser Teilsignale soll *entsprechen* dem Geräusch eines menschlichen Fußes, der einen Schritt auf einer Treppe vollzieht.

Dieses „entsprechen" kann nun in der Tat *Verschiedenstes* bedeuten : Diese Teilsignale *bestehen* entweder aus einer Schallaufzeichnung eines solchen physischen Schritt-Ereignisses, oder sie sind durch Transformationen daraus *entstanden*, oder sie *konnotieren* ein solches Ereignis, oder sie sollen ein solches *bedeuten*[4].

Wie dem auch sei, entscheidend ist der hier vollzogenen *erste Abstraktionsschritt* der Modellbildung : Das analoge Bandsignal wird betrachtet als Realisierung einer Folge von *Ereignissen* (in Abb. 6.4 und 6.5 unter β), welche mit den Angaben *wann?*, *welche Geschoßhöhe?* und *welche Klangeigenschaften?* [5] parametrisiert sind. Es handelt sich also um eine *ereignisorientierte* Modellbildung. In unserem Fall ist das Ertönen eines Schrittes (was immer das sei) die Repräsentation eines Ereignisses.

Die „Ereignisse" *dieser* Materialebene sind aber noch längst nicht ein vom Komponisten handhabbares Material. Vielmehr werden nun weitere Abstraktionsschritte vollzogen, an deren Ende die von unserer Benutzersprache zu beschreibende „Wirk-lichkeit" steht.

[3]Die „elektrische Summe" ist allerdings nur auf der Ebene der Modellbildung als *Metapher* adäquat; die tatsächliche technische Abmischung mehrerer Signale zu einem gleichzeitig Erklingendem erfordert, besonders in der Rechnerinternen Synthese, wesentlich komplexere Vorgänge als eine reine Addition, so z.B. Phasenkorrekturen. Auch in der herkömmlichen analogen Mischtechnik bereits nimmt ja der Realisierenden intuitiv Lautstärkekorrekturen entsprechend der auditiv analysierten Spektralstruktur der zu mischenden Signale vor, geleitet von seiner (evtl. vorbewußten) Vorstellung des gewünschten Klangergebnisses.

[4]Diese *durchaus verschiedenen* Verhaltensweisen sind Gegenstand genauerer Analysen in (Lepper, icht).

[5]Man beachte die beliebig komplexe interne Struktur und durchaus unterschiedliche Modellierbarkeit der letzten Bestimmung. Im tatsächlichen PGP-Projekt und in vorliegender Darstellung wird dieser Bereich als „black box" lediglich importiert.

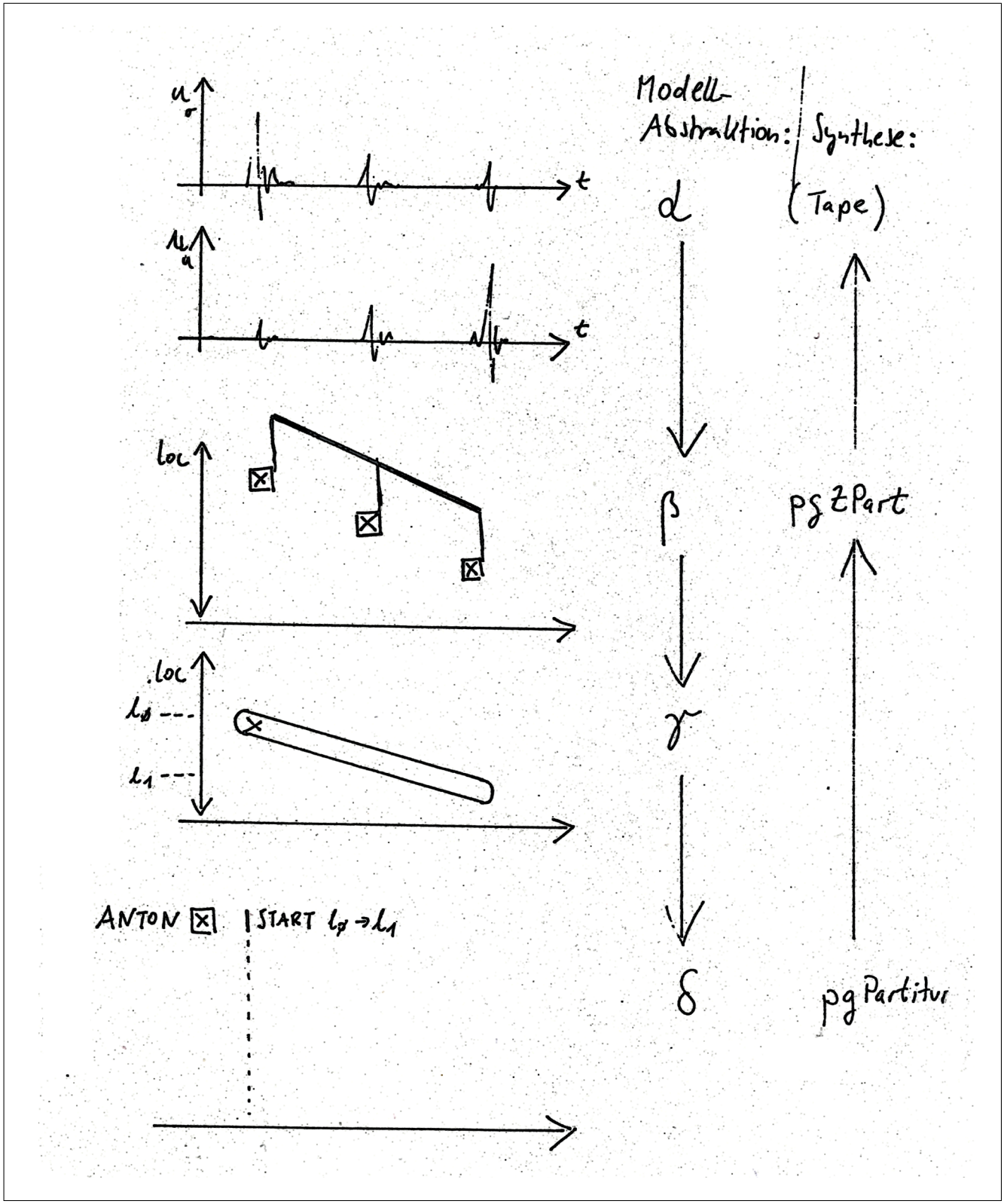


Abbildung 6.4: Materialbildende Abstraktionsschritte in PGP.

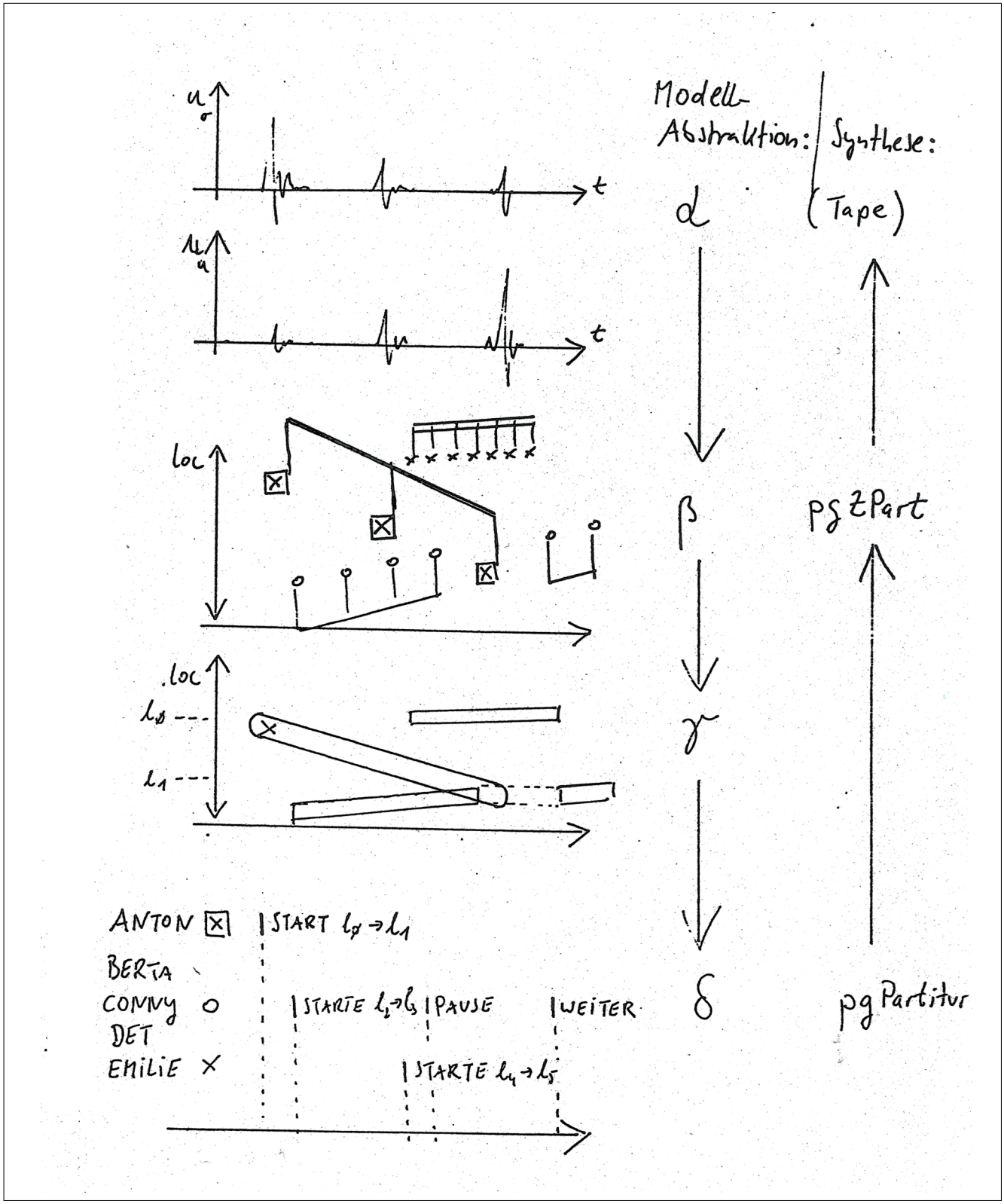


Abbildung 6.5: Mehrstimmigkeit zeigt Notwendigkeit der Abstraktionsschritte.

6.6.2 Exkurs: Ereignismodelle, Prozeßmodelle und Aggregatwechsel.

Auf den ersten Blick könnte man die eben erwähnten Abstraktionsverfahren (und die durch Umkehrung entstehenden Syntheseverfahren) klassifizieren entweder als „prozess-orientiert" (= verfahrens-orientiert) oder als „ereignis-orientiert".

Ersteres kann man vergleichen mit der „impliziten" Darstellung einer mathematischen Funktion, also qua Regel oder Formel, letzteres mit einer „expliziten", also einer Aufzählung der einzelnen Bestandteile.

In der Praxis ist allerdings der Versuch einer allzu strengen Trennung und eindeutigen Zuordnung einer Notationsweise oder „Benutzersprache" in diese Kategorien häufig inadäquat; viele Mittelgrund-Informationen sind vielmehr Mischformen, und ihre Ereignis- resp. Prozeß-„Haftigkeit" sind ein vielmehr ein *Maß*, wie stark die Einzelelemente der Beschreibung an *konkret benannte Zeitpunkte* des Gesamtverlaufes gebunden sind resp. sich auf die Gesamtdauer der beschriebenen musikalischen Struktur beziehen.

Hält man jedoch eine strenge binäre Unterscheidung (aus anderen guten Gründen) konsequent durch, so ergeben sich natürlicherweise fast immer mehrstufige (Synthese- oder Analyse-)Architekturen mit einer von Ebene zu Ebene *alternierenden* Darstellungsform: Folgen von Prozessen (ihr Beginn, Ende, ihre Modifikation) werden auf der nächst höheren Ebene zu Ereignissen. Die „Hülle" dieser Folge von Ereignissen kann wiederum als Prozeß beschrieben werden, welcher seinerseits von Ereignissen kontrolliert wird, etc.

Diese charakteristische Eigenschaft einer Architektur nennen wir *Aggregatwechsel*.

6.6.3 Die Abstraktionen $\beta \to \gamma$ und $\gamma \to \delta$.

Die im Verlauf unseres Fallbeispiels nun folgenden Abstraktionen zeigen genau diesen Aggregatwechsel: das Ereignisorientierte Modell β wird auf nächsthöherer Ebene beschrieben durch ein Prozessorientiertes Modell, welches wiederum durch Ereignisse gesteuert wird.

Die Abstraktion $\beta \to \gamma$ (in Abb. 6.4 und 6.5) faßt die eben eingeführten Ereignisse (vom Typ „ein Schritt ertönt an einer bestimmten Stellen") wiederum zu Superzeichen zusammen, macht aus dem gerade eingeführten ereignisorientierten Modell wieder ein prozessuales :

Die Schritte ihrerseits werden nämlich nicht einzeln betrachtet, sondern als Bestandteile einer Schrittfolge, eines „Ganges". Bedeutung, klangliche Realisierung und begriffliche Modellierung bestehen hier aus einem *Vorgang* der Gestalt „Person x beginnt in Stockwerk n_0 mit Tempo v etc. einen Gang in Richtung n_1".

Das heißt: jeder einzelne erklingende Schritt ist Teil eines aus vielen aufeinander folgenden Schritten bestehenden klanglichen Verlaufes, der sich *so anhört*, als würde eine *unsichtbare Person* mit einer bestimmten Zahl Schritte pro Sekunde und einer bestimmten Schrittlänge die Treppen hinauf- oder hinabsteigen, zwischendurch

anhalten, die Richtung wechseln etc. Bild 6.5 zeigt ein mehrstimmiges Beispiel, aus dem die *Zweckmäßigkeit* unserer Abstraktionsschritte (im Sinne der Handhabbarkeit) ersichtlich sein sollte.

Die Einzelereignisse werden also auf dieser Ebene γ wieder in einen *prozessualen* Begriff zusammengefasst.

Die *Steuerung und Beeinflussung* dieser semantisch/klanglichen Verläufe geschieht ihrerseits aber wiederum ereignisorientiert, indem der Komponist die *Änderungen* der Prozeßparameter zu bestimmten Zeitpunkten in der Partitur der Ebene δ als Ereignis fixiert.

Wir haben hier also auf dieser Modellebene δ einen zweiten, höheren Begriff von „Ereignis" konstruiert, von der Gestalt „Person x hält an, ... kehrt um, ... wird langsamer" etc.

6.7 Das Partiturformat. Entwicklungszustand PGP-0.2.

Für diese Partiturebene δ und für die Steuerung der Rückübersetzung $\delta \to \gamma \to \beta \to \alpha$, was ja die eigentliche Klangsynthese beinhaltet, mußte nun eine Sprache entwickelt werden. Diese nennen wir PGP-x.y.

Diese erlebte im Verlaufe der Arbeit einige Revisionen und Erweiterungen.

Für die folgenden praktischen Beispiele legen wir nun die Version PGP-0.2 zu Grunde[6].

Aus oben dargestelltem Abstraktionsgang ergaben sich genaue Forderungen an die von PGP-0.2 bereitzustellenden Formulierungsmöglichkeiten.

Einerseits muß die Sprache die notwendigen Definitionen für das klingende Material unterstützen (was dem Übergang von β nach α entspricht), andererseits die Formulierung der Partiturebene δ erlauben (deren Auswertung zu den Prozessen γ und den Einzelereignissen β führt).

Als den dazwischen vermittelnden Begriff wurde bald der einer „satztechnischen Stimme" erkannt.

Dieser wird so benutzt, daß einerseits eine solche Satzstimme durch charakteristische Eigenschaften als eine solche, d.h. als selbstidentisch, wiedererkannt wird. In diesem Sinne entspricht die Satzstimme der Stimme eines Instrumentes mit einer im Ensemblezusammenhang unverwechselbaren charakteristischen Instrumentalfarbe, wie z.B. in einem Bläserquintett.

Eine „Satzstimme" entspricht aber auch einer virtuellen aber individuellen Person, die durch das Treppenhaus geht.

[6]Die älteren Versionen hatten nur kurze Lebensdauer, hier nur kurz die Eigenschaften der Vorversionen :

Version 0.0 wurde durch äquidistante Zeitsimulation ausgewertet.

Version 0.1 führte die selbstregulierende Zeitsimulation ein, außerdem eine eigene Bewegungscharakteristik für die Bewegung auf dem Absatz, zusätzlich zu der für „aufwärts" und „abwärts".

Version 0.2 führte innerhalb der Bewegungscharakteristik den Parameter „`.abweichung`" ein.

Eine solche Stimme wird benutzt wie ein Instrument in einem Bläserquintett : Jede Stimme kann zu *einem* gegebenen Zeitpunkt genau *einen* akustischen Vorgang darstellen, d.h. eine Bewegung durch das Treppenhaus, oder aber schweigen.

6.7.1 Verschiedene Begriffe namens „Stimme".

Diese „Satzstimmen" sind keinesfalls mit den „Generatorstimmen" zu verwechseln, welche eine der Ebene des letztlich realisierenden Syntheseverfahrens angehören.

Aufgabe der Zwischenebene der automatischen Datentransformationen ist nicht zuletzt ja gerade die Vermittlung zwischen Satzstimme und Generatorstimme; Sinn und Zweck des Entwurfes einer Benutzersprache ist es ja gerade, den Komponisten von der Rücksichtnahme auf technologische Gegebenheiten möglichst weitgehend freizustellen, sie bestenfalls gar nicht zur Kenntnis nehmen zu müssen.

6.7.2 Exkurs : Wahl der Nomenklaturen.

Eine wichtige Vorbemerkung zur folgenden Nomenklatur :

Begriffe wie „Stimme" und „voice", „Partitur" und „score", „Klanggestalt" und „sound" etc. sind viel benutzt und mehrdeutig. Um eine möglichst eindeutige Nomenklatur aufzubauen, die Verwechselungen ausschließt (wie z.B. die o.e. Verwechselung zwischen Generatorstimme, Satzstimme und Zwischenpartitur), wählen wir für alle diejenigen Datentypen in einem Projekt, die man allesamt naheliegenderweise mit den Namen „Stimme" oder „voice" bezeichnen würde, die aber durchaus unterschiedliche Funktionen erfüllen, statt jener möglichst exotische Namen.

Z.B. gab es in dem Projekt TNP, Bestandteil der Bühnenproduktion „Figur und Klang im Raum" des „Theaters der Klänge", eine komplexe Datenstruktur, welche Klangtypen des „VOSIM"-Syntheseverfahrens in ihrem zeitlichen Verlauf beschrieb.

Diesen Datentyp nannten wir kurzerhand „Zwerg", weil sie dementsprechend klangen. Die Objekte dieses Datentypes hießen dann, folgend einer geschlechtlich paritätisch besetzten Version der Mainzelmännchen, „Anton", „Berta", „Conny", „Det", „Emilie" und „Fritzi", – nicht etwa „Struktur0-0a", „Struktur0-0b" etc.

Die Satzstimmen im Projekt PGP nennen wir folglich „pgPerson".

Aus demselben Grund wurden die vom Studenten vorgeschlagenen Parameternamen wie „Geschwindigkeit" ersetzt durch die eindeutigeren „.schritte/sekunde" resp. „.stufen/schritt".

6.7.3 Abstraktes Partiturformat. Zweiteiligkeit.

Das zentrale Ausdrucksmittel des Komponisten bestehen in unserem Ansatz (= der anzustrebenden Integrierenden Digitalen Arbeitsumgebung) in der Formulierung von Befehlen (allg.: Sätzen) in der von ihm zu seinen Zwecken und mit der von ihm gewünschten Semantik definierten Benutzersprache (s.o. 5.2).

Im Falle von PGP-0.2 ergaben die fortschreitenden Analysen der zunächst verständlicherweise verworrenen Ausgangsvorstellungen des Studenten eine den geklärten Vorstellungen (oder, wenn man so will, den technologisch-didaktischen kompromißähnlichen Zwischenlösungen) natürlich entsprechende Trennung in zwei Bereiche :

Zum einen die (quasi vor der Realzeit stattfindende) Definition der beteiligten sog. „pgPersonen", zum anderen die Erstellung des den Ablauf beschreibenden „Drehbuches".

Sämtliche in PGP-0.2 auftretenden Bestimmungen und ihre Zusammenhänge zeigt Bild 6.6.

6.7.4 pgPerson-Definitionen.

Diese sind grundsätzlich out-of-time und beinhalten die Bestimmungen der (zunächst festen) charakteristischen Parameter der pgPersonen, als da sind...

- Schritte pro Sekunde.
- Stufen pro Schritt.
- Klanglichkeit.

Die ersten beiden Punkte sind auch Gegenstand des Drehbuches, da sie von dort aus *dynamisch verändert* werden können, und werden dort beschrieben.

Punkt drei, – die Klanglichkeitsbestimmung der letztlichen Signal-Generierung, der Syntheseschritt $\beta \rightarrow \alpha$, ist in der momentanen Implementierung und der vorliegenden Darstellung schmählich unterrepräsentiert.

In der Tat sahen die ersten Ideen des Studenten eine dreifache Bestimmung vor (Ausgewähltes Schallmaterial einerseits, Frequenz und Bandbreite eines nachgeschalteten Filters andererseits.)

Eine solche Differenzierung macht aber streng genommen nur Sinn, wenn eine *funktionale Abhängigkeit* der dynamischen Verhaltens der Filter-Parameter von den höheren Strukturebenen formulierbar ist. Diese wäre leicht dem System hinzuzufügen (s.u. 8.4.11), geschah damals nicht aus Zeitgründen und würde auch der vorliegenden Prinzipdarstellung nichts neues beitragen. Als schlichte *Konstant*werte zusätzlich zur Schallmaterialauswahl sind sie durchaus redundant, da die Filtrierung ja out-of-time durchgeführt werden können und ihr Resultat zur Gänze als Schallmaterial genommen werden kann.

In PGP-0.2 werden beliebige[7] externe digitale Schallaufzeichnungen („recordings", neudeutsch, falsch und häßlich auch „samples" genannt), in das AUDIAC-System eingelesen und einem Namen zugeordnet.

Der Parameter .klangtyp der pgPerson-Definition besteht nun aus der Referenz auf ein klangtyp-Objekt. Dieses beinhaltet jeweils drei Referenzen auf recording-Objekte (oder einen zeitlichen Ausschnitt daraus), und zwar je eine zur Darstellung des Aufwärts- und des Abwärtschreitens und eine für das Gehen auf einem Absatz.

[7] Später z.B. auch Türenschlagen o.ä. (s.u. 8.2).

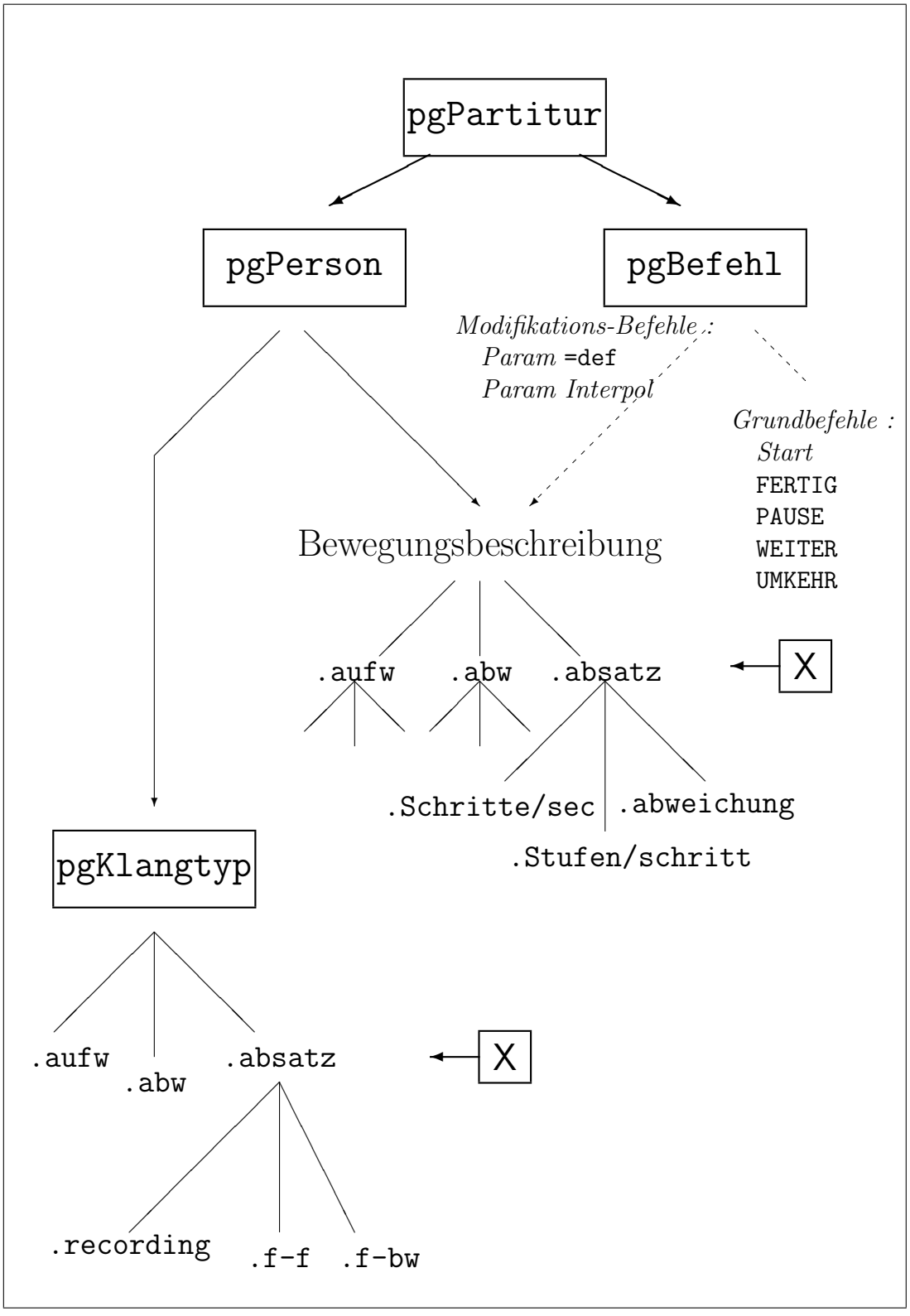


Abbildung 6.6: Bestimmungen in PGP-0.2.

6.7.5 Das Drehbuch.

Das Drehbuch, der zweite Teil der Ausdrucksmöglichkeiten des Benutzers, ist *zeitorientiert* und kommt so dem Klassischen Begriff einer „Partitur" im engeren Sinne ziemlich nahe.

Es besteht aus einer Folge von *Befehlen*[8]. Die Formen dieser Befehle bezeichnen wir als PGP-Kommando.

Bezeichen wir im folgenden diese in der Partitur auftretenden Elemente, Ereignistypen, als PGP-Kommandos. Tabelle 6.1 listet diese Befehle und ihre Parameter auf.

`BEGINNE`	*wer ?*	*wann ?*	von wo ?	wohin ?
`ENDE`	*wer ?*	*wann ?*		
`KEHRE UM`	*wer ?*	*wann ?*		
`PAUSE`	*wer ?*	*wann ?*		
`WEITER`	*wer ?*	*wann ?*		
`ÄNDERE`	*wer ?*	*wann ?*	welchen Parameter ?	auf welchen Wert ?
`INTERPOLIERE`	*wer ?*	*wann ?*	welchen Parameter ?	auf welchen Wert ?
			über welche Dauer ?	Interpolationsverfahren ?

Tabelle 6.1: PGP-0.2-Befehle

Die PGP-Kommandos zerfallen in zwei Gruppen :

- *Grundbefehle* geben an, zu welchen Zeitpunkten eine bestimmte Person einen Lauf beginnt, endet oder unterbricht,
- *Modifikationsbefehle* geben an, zu welchen Zeitpunkten eine bestimmte Person einen Parameter ihrer Fortbewegung auf welche Weise ändert.

Zu *jedem* PGP-0.2-Befehl in der Partitur muß angegeben werden, erstens *wann* er geschehen soll, das heißt, bei welchem *absoluten Zeitpunkt* innerhalb der Gesamtdauer der Partitur[9].

Zweitens muß jedesmal angegeben werden, auf welches `pgPerson`-Objekt, also auf welche Satzstimme, der Befehl sich bezieht.

Die weiteren Parameter sind je Befehl unterschiedlich, siehe Tabelle 6.1.

[8] welche jeweils als eigene APOS-Nachricht eingegeben werden.

[9] Diese Einschränkung wird in den geplanten Erweiterungen teilweise aufgehoben (s.u. 8.4.10).

6.7.5.1 Grundbefehle.

Die Grundbefehle beinhalten die Befehle...

- `BEGINNE` einen Gang.
- `ENDE`
- `KEHRE UM`
- `PAUSE` und `WEITER`, d.h. „Beginne Pause" und „Beende Pause".

Diese Ereignisse und ihre Parameter bedeuten trivialerweise:

- „Beginne einen Gang" bedeutet im Rahmen des „Dramatischen Modelles", welches einem Drehbuch entspricht : „Die angegebene Person betritt das Treppenhaus und beginnt auf- oder abzusteigen".

  Auf der Ebene des „musikalischen Modelles", d.h. der reinen satztechnischen Vorstellung, bedeutet es respektive : „Ein akustisches Signal mit den angegebenen Eigenschaften (eine im `pgPerson`-Objekt identifizierte Schallaufzeichnung, normalerweise ein Schrittgeräusch) beginnt, wiederholt zu erklingen und dabei von einem Lautsprecher zum anderen zu wandern".

  Der `Start`-Befehl ist der einzige der Grundbefehle mit zusätzlichen Parametern: ihm müssen zwei Ortsangaben namens „Start" und „Ziel" mitgegeben werden. Erstere bezeichnet das Stockwerk, aus welchem heraus die Person das Treppenhaus betritt, letztere bestimmt zum einen die initiale Bewegungsrichtung (= auf das Ziel hin), zum anderen einen Ort, an welchem dieser Gang als beendet zu betrachten ist, – die Satzstimme verstummt automatisch, sobald sie diese virtuelle Höhe erreicht[10] [11].

- „Beende einen Gang" bedeutet „Die angegebene Person verläßt das Treppenhaus", resp. „Die angegebene repetierende Stimme verstummt".
- „Kehre um" bedeutet „Die angegebene Person ändert ihre Bewegungsrichtung, und damit ggfls. ihre Bewegungsparameter, s.u.", resp. „Die angegebene repetierende Stimme ändert die Richtung der Kanalwanderung und evtl. den Repetitionsrhythmus".
- „Beginne Pause" bedeutet „Die angegebene Person hält an und ist somit nicht mehr hörbar", resp. „Die angegebene repetierende Stimme verstummt".
- „Beende Pause" bedeutet „Die angegebene Person geht weiter", resp. „Die angegebene repetierende Stimme setzt an dem virtuellen Ort wieder ein, wo sie beim Befehl „Beginne Pause" verstummte".

[10] Was interessanterweise, dank der Existenz der Befehle `KEHRE UM` und `ENDE` nicht unbedingt eintreten muß (s.u. 6.8).

[11] Das Problem der *Codierung* und Darstellung von „Höhe"-Werten ist Gegenstand der folgenden Abschnitte 6.7.8.

6.7.5.2 Modifikationsbefehle.

Die Modifikationsbefehle beinhalten die Befehle, welche die o.e. skalaren Parameter der `pgPerson`en ändern.

Bei dem ersten Befehl erfolgt die Änderung plötzlich zum angegebenen Zeitpunkt. Beim zweiten Befehl beginnt beim angegebenen Zeitpunkt eine allmähliche Änderung des Parameters auf den angegebenen Zielwert über die Dauer der angegebenen Interpolationszeit[12].

Zu beiden Befehlen muß angegeben werden, welcher Parameter verändert werden soll und auf welchen Wert er gesetzt werden soll.

Zu dem Befehl `Interpoliere` der zweiten Gruppe muß zusätzlich angegeben werden, in welcher Zeit der Zielwert erreicht werden soll, und nach welchem Verfahren interpoliert werden soll.

6.7.6 Skalare Größen und „Autarke Quasi-Physikalische Skalare".

G. M. König	PR I	1968	`regularity`
W. Hufschmidt	Trio II	1982	*Tripel von Farbwerten*
C. Barlow	Cogluotobüsişdletmesi	1980	`smoothness`

Tabelle 6.2: Berühmte Beispiele „Autarker Quasi-Physikalischer Skalare".

Sämtliche in einem kompositorischen Zusammenhang auftretenden skalaren Größen lassen sich unserer Meinung nach natürlich in drei Klassen unterscheiden:

- einheitsgewichtete *physikalische* Größen.
- reine, einheitenlose Zahlenwerte.
- Autarke (Quasi-Physikalische) Skalare.

Erstere dienen bekanntermaßen zur Modellierung von Bestimmungen der diversen physikalischen Weltmodelle. Sie sind gekennzeichnet dadurch, daß sie einen von 1.0 (eins) verschiedenen Faktor enthalten, welcher ein Produkt von Potenzen der Grundeinheiten ist. Die Grundeinheiten sind die Menge der physikalischen Grundeinheiten (Kilogramm, Meter, Sekunde, Candela etc.), ergänzt um Währungen und Normalzeit [13].

Die zweite Gruppe ist auch bekannt: reine Zahlen können (von ihrer internen Arithmetik abgesehen) eigentlich nur als *Indizes* oder *Faktoren* eine Rolle spielen.

Die dritte Gruppe behauptet der Autor als seine Entdeckung:

[12] In PGP-0.2 nicht implementiert!

[13] Auch der internationale Finanzmarkt ist ein Weltmodell, – behauptet er zumindest!

In vielen bedeutenden Werken neuerer Musik nämlich arbeitet der Komponist mit „Autarken" Skalaren. Diese bestehen zum einen aus (normalerweise total geordneten) Wertebereichen, die weder zur den Außenwelt-Größen der ersten Gruppe, noch zu den reinen Zahlen zwangsläufig bezogen sein müssen.

Zum zweiten bestehen sie aus den auf diesen Wertebereichen *vom Komponisten selbst definierten, prinzipiell beliebigen*[14]Algebren[15].

Unauffällig machen sie sich dadurch, daß diese Algebra durchaus (meist *zufälligerweise*!) zu Algebren aus den ersten beiden Gruppen isomorph sein kann, und weil naive Implementierungen sich evtl. ergebende Darstellbarkeiten ohne Skrupel zur rechnerinternen Realisierung benutzen.

Tatsächlich jedoch sind derartige Abbildungsdefinitionen höchst kritisch und sollten stets bewußt und dokumentiert, ja spezifiziert vorgenommen werden !

Tabelle 6.2 nennt einige historische Beispiele für Autarke Quasi-Physikalische Skalare.

So gibt es bei G.M. König in „Projekt Eins" aus dem Jahre ??? den Parameter „Regularität", – im Autobus-Stück von C. Barlow gibt es den Parameter „Rauhigkeit", – W. Hufschmidt operiert in „Trio Zwei" mit Tripeln symbolischer Farbintensitäten. Zu diesem illustren Kreis gehören, bei aller Bescheidenheit, auch unsere unten zu definierenden Skalaren Datentypen für die `pgPerson`-Parameter `.Ort` und `.Abweichung`.

Ebenfalls in diese Gruppe gehören die *meisten musikeigenen* Datentypen, z.B. die möglichen musikalischen Lautstärke-Skalen oder die funktionsharmonisch definierten Tonklassen.

Alle diese Datentypen teilen zwei Eigenschaften :

- Die auf ihnen definierten Strukturen können beliebig krumm und antiorthogonal sein (man s.u. die „modulo 10"-Struktur unseres „Ort"-Skalars). Operationen und Morphismen können auf ihnen beliebig definiert werden, wodurch man dann mit ihnen durchaus irgendwie „rechnen" kann, so daß sie sich „quasi-physikalisch" verhalten können.
- *Keine* irgendwie geartete Verbindung zu Datentypen aus den anderen Gruppen ist *a priori* gegeben, so daß sie „autark" zu nennen sind.

Vielmehr ist jede mögliche Verbindung zu (Produkten von) physikalischen Skalaren nur auf *eine einzige* Art möglich: Ein Datum eines Autarken Skalars wird als Eingabe-Parameter in einen *ebenfalls benutzerdefinierten* Übersetzungs-(Realisierungs-)Algorithmus gegeben[16].

[14]Z.B. hätte Hufschmidt in seiner Farblogik durchaus definierten können „`Rot + Blau = Lila` und `Lila - Blau = Grün`".

[15]Dazu gehören auch die über *mehreren* Skalaren Typen definierten Algebren, wenn mindestens eine davon ein Autarker ist, – z.B. „D-Mark mal Rauhigkeit".

[16]Dieser macht *intern* dann evtl. durchaus Gebrauch von Eigenschaften des den Autarken Typs repräsentierenden herkömmlichen Typs, sollte diese Verwendung aber als *Teil der Übersetzungsdefinition* allemal *bewußt* benutzen!

So könnte z.B. HUFSCHMIDT irgendwann spät im Kompositionsprozeß einen Übersetzungsalgorithmus aufrufen, der ein „autarkes Farbwerttripel" auf dem Pianoforte darstellt, – oder einen *anderen* Algorithmus, der dies für ein Streichquartett tut.

Diese beiden unterschiedlichen Sichten und Umgehensweisen bezeichen wir als *interne* resp. *externe* Semantik :

Die interne Semantik eines Autarken Quasi-Physikalischen Skalares wird *ausschließlich* durch die vom Komponisten frei definierten inneren Algebren bestimmt, – sie ist allemal nur *einmal* vorhanden, bleibt innerhalb des Datentyps, verhält sich „autark" und „quasi physikalisch" und weist beliebige Anti-Orthogonalitäten auf.

Die externe Semantik wird hingegen mit jedem Interpretationskontext neu definiert, – es gibt ihrer *beliebig viele*, welche in je *andere*, meist „wirklich" physikalische Datentypen transformieren.

Der Übersetzungsalgorithmus, welcher ein autarkes Datum in ein physikalisches überführt, ist allemal vom *Zielkontext* bestimmt, s.o. Abschnitt 3.4.2: Die entstehenden physikalischen Daten erhalten ihre Semantik (und damit die Korrektheit der Entsprechung zu den autarken Daten) durch den klanglichen *Produktionszusammenhang*, für den sie sozusagen einen Technischen Datensatz (TDS) bilden, – die Bestimmung „`grün`" würde z.B. für einen Synthesizer wohl anders übertragen als für ein Spinett.

Vor der Übersetzung jedoch, während der Kompositorischen Arbeit im Mittelgrund, hat der Komponist mit diesen Farbtripeln o.ä. ausschließlich gemäß seiner selbstdefinierten und autarken Algebra gerechnet, verglichen, Motive und Funktionen gebildet, permutiert, sequenziert[17] etc.

Auch hier muß die Verknüpfung mit konventionellen Skalaren explizit definiert sein: Selbst die einfache Multiplikation eines Autarken Skalars mit einem absoluten Faktor muß (mit ähnlichen Sprachmitteln wie die Übersetzung) explizit definiert werden[18].

6.7.7 Exkurs: Technische Realisierung skalarer Größen.

Die in den gängigen Computersprachen wie Pascal oder C implementierten arithmetischen Funktionen des Laufzeitsystems beziehen sich perverserweise zum großen Teil auf *zyklische* Moduln, wie z.B. der Datentyp `Integer` oder `int`. Sie sind somit von wesentlich einfacherer Struktur als die arithmetischen Typen unserer Vorstellung (z.B. vollständig axiomatisierbar).

Sie sind, auf gut Deutsch, mega-fast und ober-dirty implementiert und bilden einen Ausbund an Häßlichkeit.

[17] Das Wort „sequenzieren" soll hier den musiktheoretischen Begriff bezeichen, – „auf nächste Tonhöhen-Stufe transponiert wiederholen".

[18] NB ist die Installation einer *additiven* Verknüpfung in vielen Fällen die unbewußte Einführung eines *Morphismus*-Typs, der auf dem ursprünglichen (Objekt)-Typ operiert, der als solcher natürlicherweise in sich addiert werden kann, und der unbemerkt kanonisch auf den Objekttyp zurückrepräsentiert wird! Oder umgekehrt: Örter kann man nicht addieren, nur Abstände. Wenn aber ein Ort immer durch einen Abstand repräsentiert wird, kann diese Tatsache leicht übersehen werden

Dummerweise kann ein unreflektierter Anwender diese Datentypen bis zu einem bestimmten Punkt der Entwicklungsarbeit *scheinbar* problemlos benutzen; ihre gravierenden Nachteile, zuvörderst die strukturelle Differenz zu dem zu modellierenden Problem, werden häufig erst spät oder gar nicht wahrgenommen[19].

Obwohl die Sprache APOS sich hervorragend eignet, Datentypen mit vernunftgemäßen arithmethischen Eigenschaften zu implementieren, hauptsächlich getragen vom mathematischen Begriff der „Kategorie", mußte bis dato dank ständigen Termindruckes auf eine gründliche Bearbeitung des Themas „arithmetische Strukturen" leider verzichtet werden.

Immerhin enthält unsere Bibliothek ausschließlich sog. „sichere Arithmetik", d.h. bei Zahlenbereichsüberschreitung erfolgt keine unbemerkte modulo-Rückfaltung, sondern eine `EXCEPTION` wird ausgelöst.

Auch wir bilden also aus rein produktionspraktischen und arbeitsökonomischen Gründen die verschiedenen Skalaren Datentypen ohne typtechnisch adäquate Modellierung und unter Verzicht auf die gerade erst geforderte Explizitheit von Isomorphie-Postulaten „fast and dirty" auf die vordefinierten Datentypen `Integer` und `Fixpoint` ab[20].

[19] Der (scheinbare) Vorteil dieser modulo-Arithmetik im Vergleich zur sicheren Arithmetik und wahrscheinlich ein Grund ihres langen Überlebens ist die Gültigkeit des Assoziativ-Gesetzes!

[20] APOS 5.x enthält inzwischen `Real` statt `Fixpoint`.

	`pgPerson`	
Ort	`.wo` , `.wohin`	Angabe einer Stockwerkhöhe `Integer -20` → `80`
Schrittgeschwindigkeit	`.schritte/sekunde`	Angabe der durchschnittlichen Schritte pro Sekunde. `Fixpoint 0.0` → `32000.0`
Schrittvarianz	`.abweichung`	Maß für das Schwanken der Schritte pro Sekunde. `0` bedeutet konstant, `100` maximale Abweichung, zufällig von 50-100% . `Integer 0` → `100`
Schrittgröße	`.stufen/schritt`	Angabe der durchschnittlichen Stufen pro Schritt. `Fixpoint 0.0` → `80.0`

Tabelle 6.3: Skalare Wertebereiche in PGP-0.2.

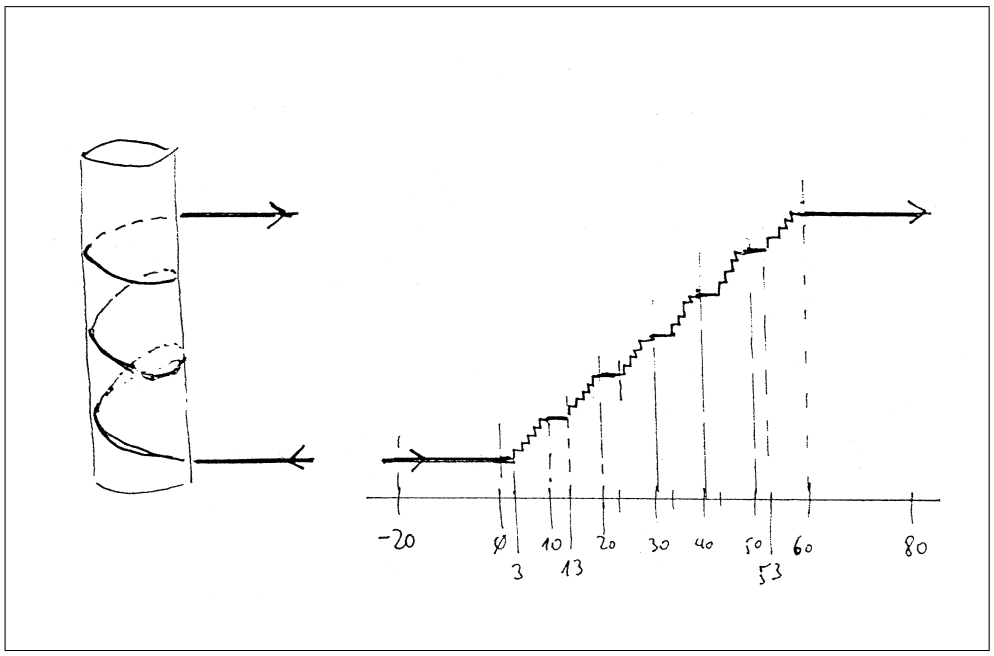


Abbildung 6.7: Darstellung der Treppenorte „Ort" als `Integer`

6.7.8 Die Skalaren Typen in PGP-0.2.

Die skalaren Parameter der `pgPerson`-Objekte und der Modifikationsbefehle zeigt Tabelle 6.3.

- Ort

  Der Ort einer virtuellen Person wird dargestellt durch eine Integer-Zahl zwischen -20 und +80.

  Dabei handelt es sich um eine Metrik, die auf der linearen Abwicklung der Treppenwendel definiert ist(siehe Bild 6.7).

  Die Zahl 0(Null) steht für den unteren Beginn des Treppenhauses, die Zahl 60(sechzig) steht für die oberste Stufe. Die Zahlen unter 0(null) und über 60(sechzig) , stehen für die zunehmende Entfernung einer Person vom Treppenhaus, nachdem sie das Ende der Stufen erreicht und dieses verlassen hat. Diese Zahlenbereiche werden als „oberes und unteres Nirwana" bezeichnet.

  Wichtig ist für die Modellierung der physischen Wirklichkeit die Tatsache, daß die Treppe *Absätze* enthält. Alle Ortsangaben, die modulo zehn(10) kleiner als drei(3) sind, werden als Orte auf einem Treppenabsatz aufgefasst, alle anderen als Stufen.

  Alle Rechenoperationen finden gleichförmig auf Orten statt, so daß ein Absatz des Treppenhauses wie drei Stufen behandelt wird. („Stufen Pro Schritt" wird automatisch zu „Drittel Absatz pro Schritt" etc.)

Die folgenden Attribute sind je `pgPerson` *dreimal* vorhanden und charakterisieren so jeweils unterschiedlich das Gehverhalten (1) abwärtssteigend, (2) aufwärtssteigend oder (3) auf dem Absatz gehend.

Konkret: Die Menge dieser Attribute eines `pgPerson`-Objektes wird identifiziert über die *Produkt*menge ...

$$\{\texttt{abwärts}, \texttt{aufwärts}, \texttt{absatz}\} \times \{\texttt{.schritte/sekunde}, \texttt{.stufen/schritt}, \texttt{.abweichung}\}$$

... als Indexmenge.

- `.Schritte/Sekunde`

  Die Anzahl der Schritte pro Sekunde wird einfach als positive Festkommazahl, d.h. als Dezimalbruch angegeben.

- `.Stufen/Schritt`

  Die *durchschnittliche* Anzahl der Stufen pro Schritt wird ebenfalls als Dezimalbruch angegeben. Die tatsächliche Zahl der Stufen für jeden einzelnen Schritt ist selbstverständlich eine *ganze* Zahl.

  Eine ganzzahlige Angabe von Stufen/Schritt bedeutet also, daß jedesmal genau diese Zahl von Stufen gegangen wird. Eine nichtganzzahlige Angabe bedeutet, daß die tatsächliche Stufenzahl der Schritte den angegebenen Wert als Durchschnitt hat.

  Eine Angabe gleich `0.0` (NullKommaNull) bedeutet ein Treten auf der Stelle (s.u., Fremdgebrauch, 8.2).

- `.Abweichung`

  Dieser Parameter wurde relativ spät eingeführt, ist aber für eine im Sinne des „Dramatischen Drehbuches" realistische Modellierung von treppensteigenden Menschen entscheidend wichtig.

  Dieser Parameter ist (in vorliegender Implementierung) mit einer Integerzahl von 0(null) bis 100(einhundert) einstellbar. Die Abweichung bezieht sich auf die Schrittfrequenz. Eine Abweichung von 0(null) bedeutet, daß die (oben als Parameter) angegebenen Schritte/Sekunde genau eingehalten werden. Eine Abweichung von 100(einhundert) bedeutet, daß die Schrittfrequenz für jeden einzelnen Schritt vom Doppelten bis zum Halben der angegebenen Schritte/Sekunde variieren kann[21].

[21] Die Implementierung PGP-0.2 ist buggy, als die Einhaltung des zugrundeliegenden, angegebenen Wertes von Schritte/Sekunde als *Mittelwert* anscheinend und ärgerlicherweise nicht garantiert ist.

6.8 Erste Kritik: Querständigkeit der vom Benutzer gewählten Faktorisierung der Parameter zur Stufenstruktur der zugrundeliegenden Modellabstraktion.

Für die Meta-Ebene der kompositionstheoretischen[22] Analyse ist nun interessant, daß der bis hierhin definierte Entwurf trotz seiner zunächst simpel anmutenden Ausdrucksfähigkeit schon Verwerfungen, „rough edges" und Unentscheidbares beinhaltet:

- Besonders aufschlußreich finden wir die Fälle, in denen der Student eine *bestimmte Gruppierung* von Parametern zu Super-Parametern vorzieht, obwohl die reine Problemananyse durchaus (auch) Argumente für andere Gruppierungen liefert:

  So werden z.B. die Parameter „*von wo?*" und „*wohin?*" vom Studenten dem `Start`-Befehl zugeordnet, ebenso die initiale Bewegungsrichtung, welche durch die Differenz implizit gegeben ist. *Nicht jedoch* werden Geschwindkeit und Schrittgröße für den gestarteten Bewegungsvorgang als Teil des `start`-Befehls implementiert. Vielmehr werden diese zunächst aus dem momentanen Zustand des `pgPerson`-Objektes entnommen und können danach durch die parameterverändernden Befehle der Befehlsgruppe zwei unmittelbar verändert werden.

  Diese Gruppierung in der Objekt-/Syntax-/Semantik-Definition kommt (1) einerseits aus einer praktischen Überlegung, andererseits (2) wiederspiegelt sie eine *Struktur der Begrifflichkeit* des Komponisten :

  Die Parameter „`.schritte/sekunde`" und „`.stufen/schritt`" werden als charakteristische Bestandteile der Personendefinition angesehen, die sich *normalerweise nicht* ändern und (zusammen mit dem ausgewählten Klangmaterial) die Identifizierbarkeit gewähren. Die Änderung eines solchen Parameters (und auch der Fall, daß dieselbe Person den einen Gang anders beginnt als den anderen,) ist der Ausnahmefall und als solcher ein „Ereignis" im doppeltem Sinne des Wortes.

  Aus ergonomischen Gründen (=1a) und wegen der grundsätzlichen Forderung nach *Redundanzminimierung* (=1b) folgt der Wunsch, sich selten ändernde Werte nicht unnötig oft hinschreiben zu müssen, – und damit o.e. Gruppierungsentscheidung.

  Darüberhinaus aber könnte es durchaus sein, daß die vom Komponisten benutzte Vorstellung von „Stimme" als (strukturelles) Konnotat die Struktureigenschaft „Als selbstidentisch Erkennbar anhand von festen Parametern" hat.

[22] ... oder auch musiktheoretischen, musikwissenschaftlichen, didaktischen, muskphilosophischen etc., – welche Kolleginnen und Kollegen sich auch angesprochen fühlen mögen ...

Konkret: Der Komponist denkt (evtl. vorbewußt) an die Gegebenheiten (genauer: Mittelgrund-Raumkoordinaten) z.B. bei einem *Bläserquintett* und überträgt, daß sich eine `pgPerson` von der anderen (über ihr Schreit-Verhalten) identifizierbar unterscheidet genau wie eine Flöte von einer Oboe (über ihr Kling-Verhalten).

- *Andererseits* aber werden die beim einmaligen `pgPerson`-Definieren (ganz zu Beginn der Partitur) angegebenen Schrittparameter *nicht* als Reset-Parameter aufgehoben und z.B. bei jedem neuen Startbefehl re-initialisiert.

  Vielmehr sind, völlig unabhängig von Gängen, der Zustand der Parameter zu jeder Zeit der Zustand der Ausgangsdefinition überlagert mit der (zeitlich richtigen Folge) *aller bis-hierigen* Modifikationsbefehle (s.o. 6.7.5.2). Dies ist nicht unbedingt konsequent und praktisch.

- Die Dreiteilung „aufwärts, abwärts, Absatz" findet sich (1) in den Bewegungsparametern von `pgPerson` und den Modifikationsbefehlen, aber auch (2) *verdoppelt* innerhalb von „`Klangtyp`" (siehe Bild 6.6, bezeichnet mit X) !

  Da zwischen `pgPerson` und `Klangtyp` eine strikte eins-zu-eins-Relation besteht, sind beide eigentlich nur zwei Teilaspekte einer einzigen „entity", – der Datenentwurf streng genommen *fehlerhaft*.

  Die implementierte Zweiteilung (und die damit verbundene Verdoppelung der Dreiteilung!) ist vielmehr eine *ergonomisch begründete* und reflektiert zwei unterschiedliche die *Arbeitsphasen* :

  Das Schritt-Klang-Material, was in der *ersten* Arbeitsphase (mühsam) extrahiert, beschnitten und behandelt wurde, kann nun im *zweiten* Schritt in *verschiedensten* Partituren (*innerhalb* derer `pgPerson`-Objekte ja nur existieren) verwendet werden.

  Da die derart zwei-geteilte Objektdefinition durchaus Auswirkungen auf Denken und Verhalten des Komponisten hat (klangliche Veränderungen innerhalb derselbe Stimme sind nur schwer zu realisieren, nämlich durch Gezielten Fehlgebrauch,s.u. 8.2), ist sie ein Beispiel für die oben (3.5.3) behauptete ästhetische Relevanz der Achse „Arbeitsphase" im kompositorischen Bestimmungsraum

- Alle `recording`-Klang-Parameter werden von den Bewegungsparametern *abgetrennt* und in den eigenen Objekttyp `klangestalt` zusammengefasst. Aus der Sicht von „Partitur" oder „Drehbuch" sind damit out-of-time und nicht beeinflußbar.

  Diese Tatsache ist ein typisches Beispiel für eine *Meta-Bestimmung*, hier bezgl. der Anwendung oder Nicht-Anwendung satztechnischer Verfahren auf physikalische Parameter.

  Diese Meta-Bestimmung trägt u.U. durchaus Semantik:

(1) Die feste Zuordnung von Klangmaterial zu `pgPerson` *materialisiert* überhaupt erst den psycho-internen Begriff von Stimme als operable Einheit im Denken des Komponisten. (2) Sie „bedeutet" für Hörer und Komponisten, – so lange der *Rezeptionsstandpunkt* das „Hörspiel" oder „Drama" (= Geschehen) ist –, daß jede „handelnde Person" zwar ihre Schrittgeschwindigkeit jederzeit ändern kann, nicht jedoch (innerhalb eines PGP-Satzes) ihr Schuhwerk wechselt !

- Noch ein Widerspruch: Die Beendigung eines Ganges kann erfolgen entweder durch (1) expliziten `ENDE`-Befehl im Drebuch, oder durch (2) Erreichen des als `.wohin` beim `START`-Befehl angegebenen „Ziel-Ortes", oder (3) durch Über- oder Unterschreiten der Grenzen des Orte-Wertebereiches.

  Alle drei Möglichkeiten sind nicht korreliert!

  Es kann häufig sein, daß Drehbuch-Grundbefehle gar nicht mehr ausgeführt werden, weil der Gang wegen (2) oder (3) schon beendet ist. Dabei wird in PGP-0.2 nicht einmal eine warning-Anzeige ausgegeben.

  Modifikationsbefehle hingegen werden weiterhin ausgeführt, um sich beim Beginn des *nächsten* Ganges verspätet in modifizierten Parameterzuständen auszuwirken. Die genaue Untersuchung solchen „Fehler-Verhaltens" ist Gegenstand des eigenen Abschnittes 8.3 weiter unten.

  Um zumindest Störungen von (1) durch (2) sicher zu vermeiden, ist ein *Trick* nötig, nämlich `.wohin` nur zur Bestimmung der initialen Bewegungsrichtung zu benutzen und auf das Maximum oder Minimum des Wertebreiches zu setzen, so daß zumindest (2) zusammenfällt mit (3).

  Häufig ist auch das Gegenteil: `.wohin` wird wg. vorhergehender `UMKEHR`- oder `ENDE`-Befehle nie erreicht, – auch hier dient das Ziel also nur zur Codierung der initialen Bewegungsrichtung (vgl. unten die Ausführungen zum „gezielten Fehlgebrauch" in 8.2). .

- Manchen dieser Anti-Orthogonalitäten haben eine gemeinsame Wurzel: Die unterschiedliche Behandlung von Raum und Zeit. Alle Befehle des Drehbuchs müssen mit der absoluten Zeit parametrisiert werden, obwohl der Benutzer vielleicht eine örtliche Bedingung formulieren will. Andererseits ist es recht aufwendig, den aktuellen virtuellen Ort einer `pgPerson` n Zeiteinheiten nach ihrem `Start`-Befehl überhaupt zu errechnen. Selbst wenn kein anderer Drehbuch-Befehl interferiert, so muß zumindest

  $$\texttt{.stufen/schritt} * \texttt{.schritte/sekunde} * n$$

  berechnet werden.

Weitere Analyse und Vorschläge zur Lösung dieser Problematiken unten in 8.3 und in 8.4.10.

6.9 Konkrete Syntax der APOS-Implementierung.

Syntaktische einfache (z.B. immer explizite Klammerung verlangende) Mikro-Sprachen können dank der APOS-eigenen *Musterbehandlung* in wenigen Zeilen Quelltext elegant implementiert werden.

Damit kommt man im kompositorischen Alltag schnell erstaunlich weit.

Hilfreich ist besonders, daß in APOS derartig implementierte Parser auf bereits typisierten Objekten und nicht auf Lexemen operieren.

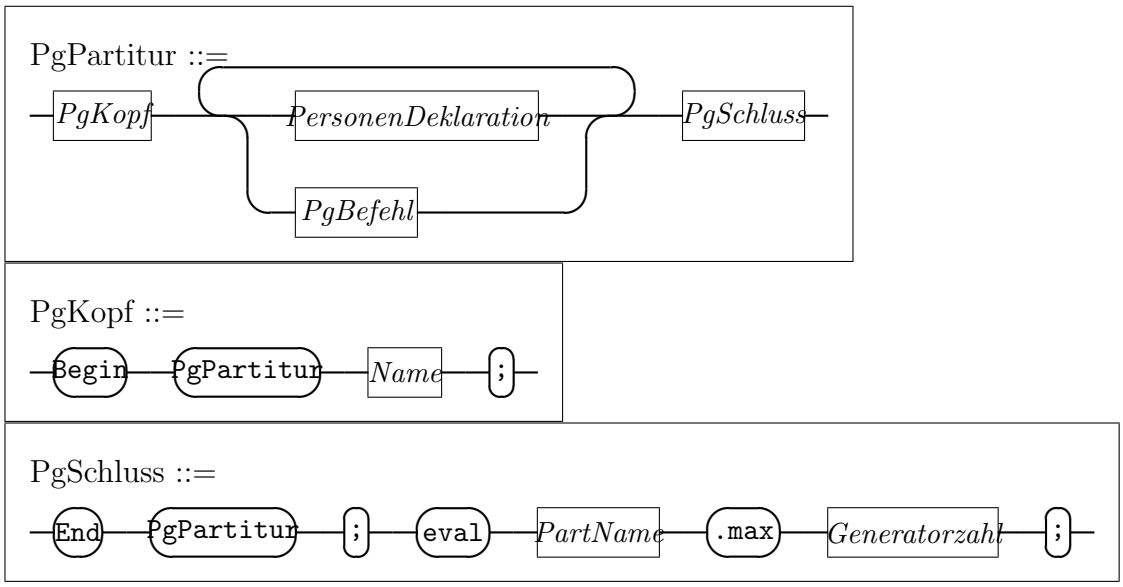
Grammatische Strukturen werden entsprechend *über Objektklassen* als ihrem terminalen Alphabet definiert (nicht über viel weniger Struktur beinhaltenden Lexem-Klassen), also über recht komplizierten Objekten, die vom System schon standardmäßig verwaltet (= internalisiert, benannt, erkannt, angezeigt, verglichen etc.) werden können.

Besonders Verletzungen von typmäßigen Kontextbedingungen werden so fast automatisch auf die „message not understood"-Exception abgebildet.

6.9.1 Partiturformat.

Vorgestellte Analyse der vom Benutzer in dieser Ausbaustufe gewünschten Ausdrucksmöglichkeiten spiegelt sich in den konkreten Syntaxgraphen der zu implementierenden Benuztersprache wieder.

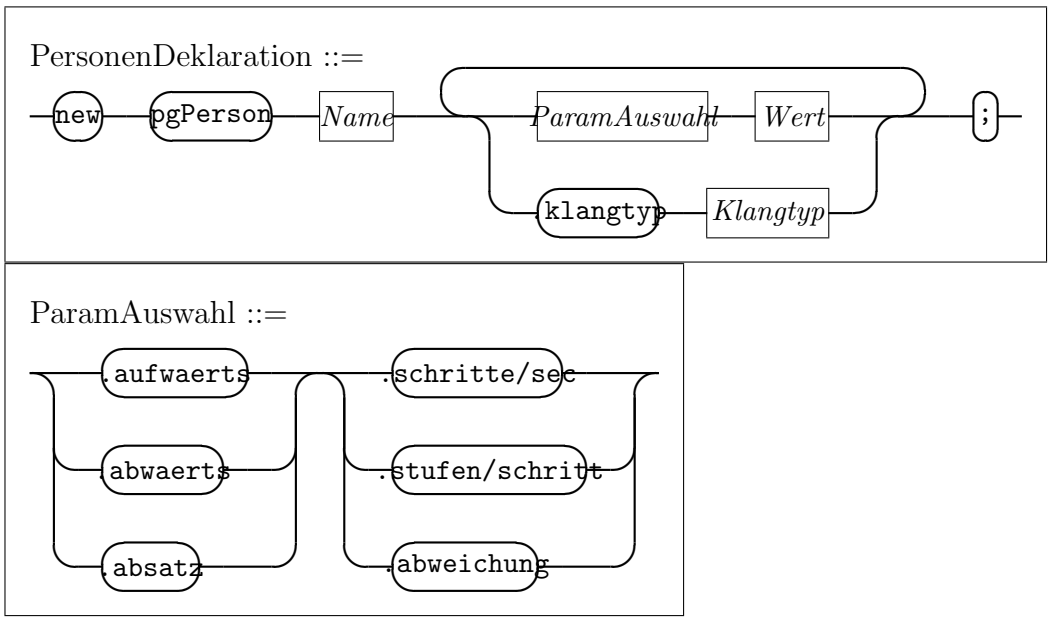
Obwohl häufig ein zweiteiliges Partiturformat intuitiv eingehalten wurde, – zuerst die Personendeklarationen, danach das „Drehbuch" bestehend aus den sog. PGP-0.2-Befehlen, – schreibt die Syntax eine derartige Trennung nicht vor. Einzig wichtige Kontextbedingung ist, daß eine `pgPerson` deklariert ist, ehe sie benutzt wird !



NB: Das in den Syntaxgraphen auftretende Semikolon ist das der APOS-Sprachebene, d.h. es bezeichnet die Stellen, an denen eine „Nachricht" im APOS-Sinne endet.

6.9.2 pgPerson-Deklaration.

Jede Personendeklaration definiert eine der Satzstimmen (pgPersonen) der Partitur, gibt ihr einen Namen und die anfänglich gültigen Parameterwerte.

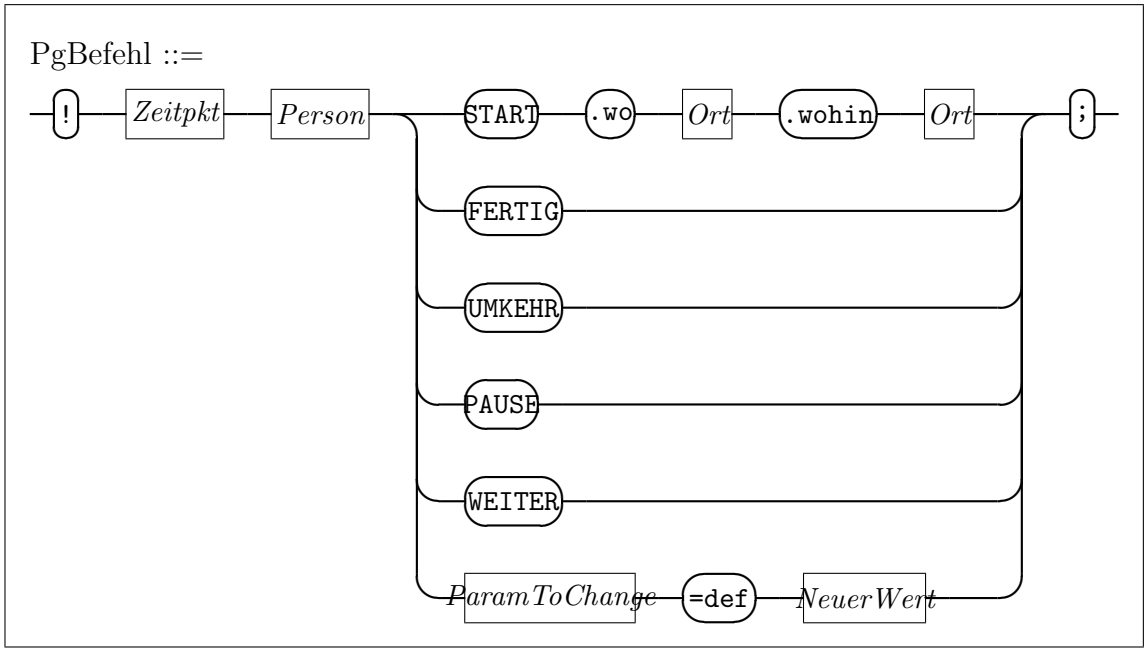


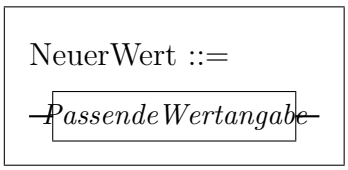
Man beachte, daß das Nonterminal „*Wert*" der *Kontextbedingung* unterliegt, daß es in einen für eine Zuweisung an den selektierten Parameter passenden Datentyp abgeleitet werden muß (also ein Ausdruck vom Typ `Integer` für den Parameter `.abweichung` und vom Typ `Fixpoint` für die anderen Parameter).

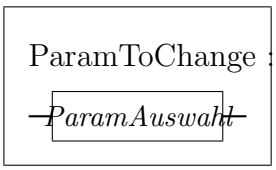
6.9.3 Drehbuch.

Der zweite Teil der Partitur enthält das „Drehbuch", also die Liste, wann welche Stimme was zu tun hat. Dieses Drehbuch ist eine Folge von sog. „PGP-0.2-Befehlen"(= Grundbefehle und Modifikationsbefehle). Diesen entsprechen die folgenden Syntaxgraphen[23] ...

[23]Die tatsächlich implementierten Syntaxgraphen verzichteten auf die Nonterminale `START` und `=def`. Außerdem ist als *ParamToChange* lediglich realisiert die (in dieser Darstellung erst unter „Erweiterungen" (8.4) präsentierte) Variante, die automatisch den Parameter der „aktuellen Bewegungsrichtung" selektiert, und die in mehrfacher Hinsicht unzureichend ist.







Auch hier bei den Parameter-Modifikationsbefehlen gilt o.e. Kontextbedingung für Typkompatibilität.

6.10 Anwendungsbeispiele im Quelltext.

6.10.1 Einstimmig.

Quelltext 1

```
1 ... ; eine erste einfache Partitur
2 ...
3 ...  new klangtyp "stoeckelschuhe-2"
4 ...  @ stoeckelschuhe-2 .aufwaerts (recording "anna-langsam-auf-stahl")
5 ...>                                  from (2.5 sec) to (4.23 sec)
6 ...  @ stoeckelschuhe-2 .abwaerts  (recording "anna-langsam-auf-stahl")
7 ...>                                  from (5.0 sec) to (6.1  sec)
8 ...  @ stoeckelschuhe-2 .absatz    (@ stoeckelschuhe-2 .aufwaerts)
9 ... ; ------------------------------------------------------------
10 ..
11 ..  new pgPartitur "einsameBerta"
12 ..  new pgPerson "berta"  .aufwaerts .schritte/sec    2.3
13 ..>                       .aufwaerts .stufen/schritt  2.0
14 ..>                       .aufwaerts .abweichung       50
15 ..>                       .abwaerts  .schritte/sec    3.3
16 ..>                       .abwaerts  .stufen/schritt  3.0
17 ..>                       .abwaerts  .abweichung      100
18 ..>                       .absatz    .schritte/sec    1.2
19 ..>                       .absatz    .stufen/schritt  1.0
20 ..>                       .absatz    .abweichung       30
21 ..>                       .KlangTyp  stoeckelschuhe-2
22 .. ;
23 ..
24 ..  !  5.0  berta Start .wo 10 .wohin 70
25 ..  ! 20.0  berta Umkehr
26 ..  ! 30.0  berta Fertig
27 ..  end pgPartitur
28 ..  eval einsameBerta  1
```

Ende Quelltext

Zum Verständnis dieses Quelltextes :

- Die erste Spalte im (alten) APOS-Batch-Format ist eine Steuerspalte: „;" leitet eine Kommentarzeile ein, „>" bedeutet, daß die folgende Zeile die *Fortsetzung* des Textes einer begonnenen Nachricht ist[24].
- Die Definition eines `Klangtyp`-Objektes hat hier denkbar einfachste Form (Angabe eines Ausschnittes einer Schallaufzeichnung) und geschieht in der konkreten Praxis durch einen *eigenen Arbeitsgang*, in dem das Klangmaterial präpariert wird. Der AUDIAC-Befehl „`recording` *(any Name)*" durchsucht alle Schallaufzeichnungen des aktuellen Projektes nach einer Aufnahme mit dem angegebenen Namen.
- Die Angabe der Zeit erfolge in Sekunden.

[24]Eine neue Zeile ohne einleitendes Steuerzeichen bedeutet ein *implizites Message-Complete-Semikolon* am Ende einer vorausgehenden (nicht-Kommentar) Zeile(n).

6.10.2 Exkurs. Die Diachrone Darstellung als Benutzereingabeformat.

Auf der Ebene der (Implementierung von) Evaluierung spielt das Konzept der „Synchronen Simulation" (s.o. 4.2) eine große Rolle (nicht zuletzt, wenn dem *Autor* des Algorithmus eine synchrone Vorstellung zur Problemformulierung naheliegt).

Spätestens, wenn ein Evaluierungslauf warning-messages an den Benutzer sinnvoll kommunizieren soll, muß der Benutzer sich evtl. dieses Denkmodell, also das Evaluieren einer „Partitur von links nach rechts" (allgemein: Die jeweils implementierte Strategie des Evaluationsalgorithmus, von dem die Fehlermeldung kommt,) *leider* zu eigen machen, um die Fehlermeldung zu verstehen.

Davon ersteinmal *unabhängig* sind aber die zeitbezogenen, zeitlich geordneten Eingabeformate, die sich in der Benutzereingabe finden. Diese nennen wir „Diachrone Darstellungen".

Je nach zu formulierender Vorstellung (Satzstruktur) können allerdings *beliebige Polyphonien* von verschieden gruppierten Diachronen Darstellungen als *Eingabeformat* durchaus angemessen sein.

6.10.3 Mehrstimmig (-spurig) : Mehrere diachrone Darstellungen.

Im ersten der folgenden Texte werden zwei Stimmen voneinander unabhängig (und out-of-time), aber in sich jeweils diachron (chrono-logisch) generiert:

Quelltext 2

```
1....; eine erste zweistimmige Partitur
2....
3....  new pgPartitur "BertaUndAnton"
4....
5....  new pgPerson "anton"  .aufwaerts .schritte/sec    1.3
6....>                       .aufwaerts .stufen/schritt  1.0
7....>                       .aufwaerts .abweichung       50
8....>                       .abwaerts  .schritte/sec    2.3
9....>                       .abwaerts  .stufen/schritt  2.0
10...>                       .abwaerts  .abweichung        0
11...>                       .absatz    .schritte/sec    1.2
12...>                       .absatz    .stufen/schritt  1.0
13...>                       .absatz    .abweichung        0
14...>                       .KlangTyp  Bergstiefel-1
15...;
16...  ! 15.5  anton Start  .wo    10  .wohin   60
17...  ! 30.0  anton .aufwaerts  .schritte/sec =def 2.0
18...  ! 50.0  anton pause
19...  ! 50.0  anton umkehr
20...  ! 70.0  anton weiter
```

```
21...
22...  new pgPerson "berta"  .aufwaerts .schritte/sec    2.3
23...>                       .aufwaerts .stufen/schritt  2.0
24...>                       .aufwaerts .abweichung       50
25...>                       .abwaerts  .schritte/sec    3.3
26...>                       .abwaerts  .stufen/schritt  3.0
27...>                       .abwaerts  .abweichung      100
28...>                       .absatz    .schritte/sec    1.2
29...>                       .absatz    .stufen/schritt  1.0
30...>                       .absatz    .abweichung       30
31...>                       .KlangTyp  stoeckelschuhe-2
32...;
33...  !  5.0  berta Start .wo 10 .wohin 70
34...  ! 20.0  berta Umkehr
35...  ! 30.0  berta Fertig
36...  end pgPartitur
37...
38...  eval BertaUndAnton  2
```

Ende Quelltext

6.10.4 Mehrstimmig : *Eine* diachrone Formulierung.

Im folgenden Beispiel wird (eher homophon gedacht, oder dramatisch!) die „Gesamthandlung" *aller* Personen zusammengefaßt in *eine* diachronen Darstellung:

Quelltext 3

```
1....  new pgPartitur "BertaUndAnton_1"
2....  new pgPerson "anton"  .aufwaerts .schritte/sec    1.3
3....     ...  etc.  ...
4....  new pgPerson "berta"  .aufwaerts .schritte/sec    2.3
5....     ...  etc.  ...
6....
7....  !  5.0  berta Start .wo   10  .wohin 70
8....  ! 15.5  anton Start .wo   10  .wohin  60
9....  ! 20.0  berta Umkehr
10...  ! 30.0  anton .aufwaerts  .schritte/sec =def 2.0
11...  ! 30.0  berta Fertig
12...  ! 50.0  anton pause
13...  ! 50.0  anton umkehr
14...  ! 70.0  anton weiter
15...  end pgPartitur
16...  eval BertaUndAnton_1 2
```

Ende Quelltext

6.10.5 Mehrschichtig : Seriell/Postseriell.

Das folgenden Beispiel repräsentiert ein post-serielles Satzdenken: der Zeitpunkt eines UMKEHR-Ereignisses einer pgPerson wird unabhängig von allen anderen Grund- und von den Modifikationsbefehlen (ähnlich einer postseriellen „Parameter-Polyphonie") in einem eigenen Rhythmus (qua eigener diachroner Darstellung) gegeben:

Quelltext 4

```
1....  new pgPartitur "BertaUndAnton_2"
2....  new pgPerson "anton"  .aufwaerts .schritte/sec    1.3
3....     ...  etc.  ...
4....  new pgPerson "berta"  .aufwaerts .schritte/sec    2.3
5....     ...  etc.  ...
6....
7....  !  50.0  anton UMKEHR
8....  ! 100.0  anton UMKEHR
9....  ! 150.0  anton UMKEHR
10...  ! 200.0  anton UMKEHR
11...
12...  !  20.0  berta UMKEHR
13...  !  40.0  berta UMKEHR
14...  !  60.0  berta UMKEHR
15...  !  80.0  berta UMKEHR
16...  ! 100.0  berta UMKEHR
17...  ! 120.0  berta UMKEHR
18...  ! 140.0  berta UMKEHR
19...  ! 160.0  berta UMKEHR
20...  ! 180.0  berta UMKEHR
21...  ! 200.0  berta UMKEHR
22...
23...  !  5.0  berta Start  .wo   10  .wohin  70
24...  ! 15.5  anton Start  .wo   10  .wohin  60
25...  ! 30.0  anton  =def  .schritte/sec 2.0
26...  ! 30.0  berta  =def  .abweichung 75
27...  ! 50.0  anton pause
28...  ! 60.0  berta Fertig
29...  ! 70.0  anton weiter
30...  end pgPartitur
31...  eval BertaUndAnton_2 2
```

Ende Quelltext

Ob eine solche Partitur (mit vielen „in der Luft hängenden" UMKEHR-Befehlen) allerdings vom Interpreter überhaupt akzeptiert wird, entscheidet eine weitergehende Diskussion über den *Fehlerbegriff* als Teil einer „erweiterten Semantik" unserer Sprache, die unten in 8.3 geführt wird.

6.10.6 Dasselbe unter Verwendung von APOS-Kontrollstrukturen.

Quelltext 5

```
1....  new pgPartitur "BertaUndAnton_3"
2....  new pgPerson "anton"  .aufwaerts .schritte/sec    1.3
3....     ...  etc.  ...
4....  new pgPerson "berta"  .aufwaerts .schritte/sec    2.3
5....     ...  etc.  ...
6....
7....  apl 20.0 to 200.0 step 50.0 [ !  _0   anton UMKEHR ]
8....  apl 20.0 to 200.0 step 20.0 [ !  _0   berta UMKEHR ]
9....
10...  !  5.0  berta  .wo 10 .wohin 70
11...  ! 15.5  anton  .wo   10  .wohin  60
12...  ! 30.0  anton  .aufwaerts .schritte/sec 2.0
13...  ! 30.0  berta  =def  .abweichung 75
14...  ! 50.0  anton pause
15...  ! 60.0  berta Fertig
16...  ! 70.0  anton weiter
17...  end pgPartitur
18...  eval BertaUndAnton_3 2
```

Ende Quelltext

... sieht schon etwas hübscher aus. Mehr dazu in 8.5.

6.10.7 Algorithmisch - Seriell/Postseriell.

Quelltext 6

```
1....  new pgPartitur "BertaUndAnton_4"
2....  new pgPerson "anton"  .aufwaerts .schritte/sec    1.3
3....     ...  etc.  ...
4....  new pgPerson "berta"  .aufwaerts .schritte/sec    2.3
5....     ...  etc.  ...
6....
7....  apl 0 to 3  [ ! [RANDOM 7.0 to 170.0 ] anton UMKEHR ]
8....  apl 0 to 10 [ ! [RANDOM 7.0 to 170.0 ] berta UMKEHR ]
9....
10...  !  5.0  berta .wo 10 .wohin 70
11...  ...     etc ...
```

Ende Quelltext

... auch dazu mehr in 8.5.

Kapitel 7

Exkurs: Zur Implementierung.

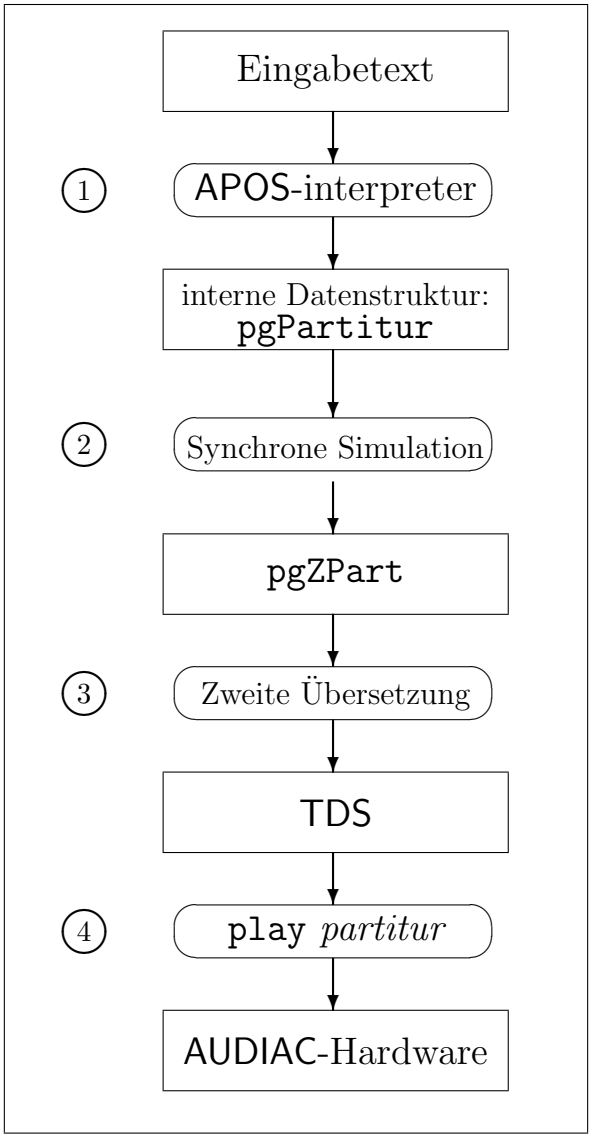


Abbildung 7.1: Die Großphasen unserer Implementierung.

Die Implementierung der Interpretation von PGP-0.2 und der letztlichen Klanggenerierung erfolgte in der musterorientierten Sprache APOS und dem AUDIAC-eigenen Mikro-Code.

Die Implementierung ist ziemlich „straight forward" und wird im folgenden nur in (leicht vereinfachten) Auszügen aus dem Quelltext zusammen mit stichpunktartigen Kommentaren vorgestellt.

Die folgende Darstellung ist idealisiert, als sie die Idiosynkrasieen der Hardware und ihrer Treiber unterschlägt. (Siehe Quelltexte olp5705/PGP.)

Bild 7.1 zeigt die einzelnen Großschritte der momentanen Implementierung:

Der Benutzertext wird (1) durch den APOS-Interpreter in eine Datenstruktur überführt. Wird für diese der `eval`-Befehl aktiviert, so findet die (2) Evaluierung durch Synchrone Simulation statt und eine Zwischenpartitur wird generiert.

Diese wird dann (3) in einen TDS (= Technischer Datensatz, s.o. 5.10) übersetzt, welcher (4) auf die AUDIAC-Hardware geladen wurde und dann spielte.

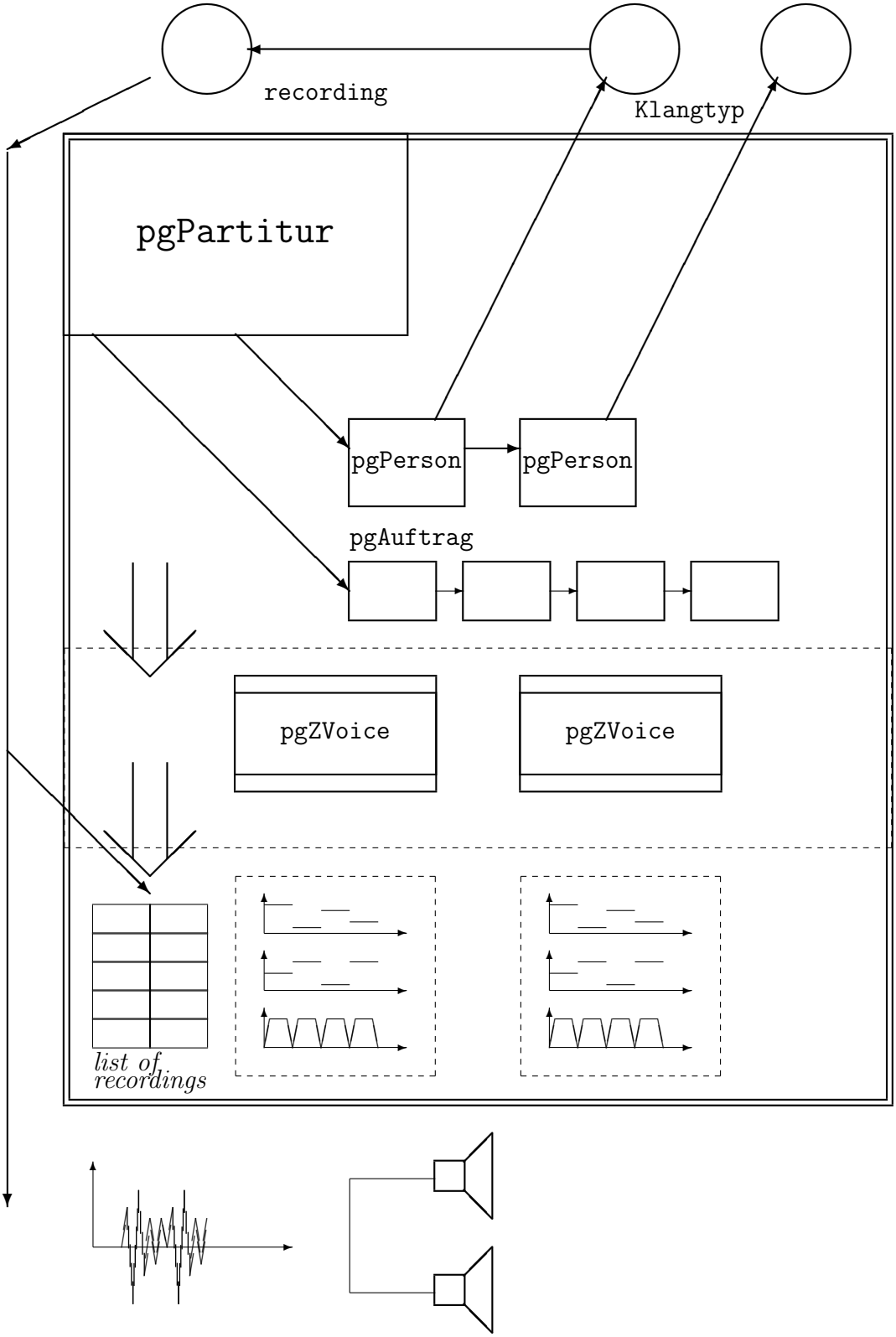


Abbildung 7.2: Die interne Objektstruktur.

				δ	$\equiv$	Benutzerbefehle
				γ	$\equiv$	„Gang" einer `pgPersond`.
Transformation	δ	$\Longrightarrow$	γ	β	$\equiv$	Synchrone Simulation
				β	$\equiv$	`pgZPart`
Transformation	β	$\Longrightarrow$	α	$\equiv$	`Pfn`-Generierung und Playing.	

Tabelle 7.1: Rück-Übertragung der Abstraktionsschritte in die rechnerinterne Synthese.

Diese Syntheseschritte entsprechen (wie angekündigt) dem Rücklauf der in Abb 6.4 und 6.5 durchgeführten analytischen Transformation. Die einzelnen Entsprechungen zeigt Tabelle 7.1.

Bild 7.2 zeigt die Objektstruktur für eine `pgPartitur`. Alles im oberen Drittel wird beim Eintreffen der Benutzerbefehle (Phase 1 aus Bild 7.1) statisch aufgebaut, – die darunterliegenden Teile werden in den folgenden Auswertungsschritten (2 und 3) generiert.

7.1 Aufbau der Objektwelt.

Quelltext 1

```
1....; // alle folgenden Quelltexte vgl. old/olp5705/pg/ PGP0A
2....
3....; -----------------------------------------------------
4....   edit (new package "pgp-2.0")
5....; -----------------------------------------------------
6....
7....   new sysconst "hoehe/geschoss"   10
8....   new sysconst "absatz/geschoss"   3
9....
10...   new sysconst "pgMaxUnend"      80.0
11...   new sysconst "oberesNirvana"   60.0
12...   new sysconst "unteresNirvana"   0.0
13...   new sysconst "pgMinUnend"     -20.0
14...
15...   new sysconst "EpsilonOrt"   1.0
16...   new sysconst "maxZeit"      32000.0
17...
18...; -----------------------------------------------------
19...
```

```
20...   new parent "pgKlangTyp"  lexobject
21...   new field   pgKlangTyp  "aufwaerts" OBJ
22...   new field   pgKlangTyp  "abwaerts"  OBJ
23...   new field   pgKlangTyp  "absatz"    OBJ
24...   new field   pgKlangTyp  "f-f"       integer
25...   new field   pgKlangTyp  "f-bw"      integer
26...; etc...
27...   enddef pgKlangTyp
28...
29...; ------------------------------------------------
30...
31...   new parent "pgPartitur" Ensemble
32...   new field   pgPartitur "dauer" fixpoint
33...   new field   pgPartitur "auftraege" Qu
34...   new field   pgPartitur "protokoll" ~access
35...; temporaer fuer Auswertung :
36...   new field   pgPartitur "aktZeit"    fixpoint
37...   new field   pgPartitur "naechsteZeit"    fixpoint
38...   new field   pgPartitur "naechstesObj"    OBJ
39...   enddef pgPartitur
40...
41...; ------------------------------------------------
42...
43...   new parent "bewegung" bytarr
44...   new field   bewegung "schritte/sec"      fixpoint
45...   new field   bewegung "stufen/schritt"    fixpoint
46...   new field   bewegung "abweichung"        integer
47...   enddef bewegung
48...
49...   new parent "PgPerson"  LEXOBJECT
50...   new field   PgPerson "klangtyp" pgKlangtyp
51...   new embed   PgPerson "aufwaerts" bewegung
52...   new embed   PgPerson "abwaerts"  bewegung
53...   new embed   PgPerson "absatz"    bewegung
54...; temporaer fuer Auswertung :
55...   new field   PgPerson "aktIpols" Qu
56...   new field   PgPerson "running"  OBJ
57...   new field   PgPerson "pause"    OBJ
58...   new field   PgPerson "richtung" FTOK
59...   new field   PgPerson "wo"       pgOrt
60...   new field   PgPerson "wohin"    pgOrt
61...   new field   PgPerson "demnaechst" fixpoint
62...   new field   PgPerson "generator"  OBJ
63...   enddef PgPerson
```

```
64...
65...   dm [ new pgperson (any name) ]
66...>      [ init .aktipols [back[_0_s1_2]] ]
67...
Ende Quelltext
```

Anmerkungen:

1.1 Quelltext 1 Zeile 3 ff. definiert zunächst einmal einige System- oder „Natur"-Konstanten.

1.2 Die Konstante „`epsilonOrt`" in Quelltext 1 Zeile 15 kontrolliert die Genauigkeit der Vergleichung zwei `pgort`-Werten.

Daß der *Gleichheitsbegriff* eines benutzerorientierten Datentyps jedesmal speziell definiert werden muß, ist fast der Regelfall. Zweckmäßig wäre eine generische algebraische Struktur, z.B. hinreichend vernünftige Unterstützung von Kategorien oder zumindest speziellerer Operator-Strukturen.

Da unsere APOS-Implementierung das aus zeitlichen Gründen leider noch nicht kann, müssen derartige Probleme dummerweise immer „zu Fuß" erledigt werden (s.o. Abschnitt 6.7.7).

1.3 Die Objektklasse „`Klangtyp`" (deren Definitinion in Quelltext 1 Zeile 20 beginnt) und die entsprechenden Methoden sind eigentlich *außerhalb* von PGP-0.2 definiert und gehören im Gesamtprojekt zu einer *eigenen vorgelagerten* Arbeitsphase. Sie stehen hier etwas „provisorisch"[1].

1.4 Die „Records"vom Typ „`pgPerson`" (Quelltext 1 Zeile 49 ff.) *simulieren* während des Evaluierungslaufes (der ja als diachrone Simulation erfolgen soll) die jeweils momentanen Zustände der nach der Paritursemantik agierenden „Personen".

1.5 Das Feld „`.auftraege`" in Quelltext 1 Zeile 33 heißt so aus historischen Gründen, - es geht zurück auf das ersten brain-storming mit dem Studenten.

[1] ... und werden NB in der Felddefinition von `pgPerson` momentan auch nicht benutzt, s. Quelltext `??` !

Quelltext 2

```
1....
2....   new parent "PgAuftrag"  Obj
3....   new field   PgAuftrag "partitur"  QuItem
4....   new field   PgAuftrag "wann"    fixpoint
5....   new field   PgAuftrag "wer"     pgPerson
6....   new field   PgAuftrag "ausgefuehrt" OBJ
7....   enddef pgAuftrag
8....
9....
10...   new parent "Pg-ParamChng"  pgAuftrag
11...   new field   Pg-ParamChng "welcher"         ftok
12...   new field   Pg-ParamChng "richtung"        ftok
13...   new field   Pg-ParamChng "wert"            ~access
14...   enddef Pg-ParamChng
15...
16...   new parent "Start"      pgAuftrag
17...   new field   start       "wo"         pgOrt
18...   new field   Start       "wohin"      pgOrt
19...   enddef pg-StartAuftrag
20...
21...   new parent "Pg-Interp" Pg-ParamChng
22...   new field  Pg-Interp  "dauer"  fixpoint
23...   new field  Pg-Interp  "modus"  obj
24...   new field  Pg-Interp  "fuerWen" QuItem
25...; nur temporaer fuer ausertung :
26...   new field  Pg-Interp  "startwert" univskal
27...   enddef  Pg-Interp
28...
29...   enddef(new parent "Umkehr" pgauftrag)
30...   enddef(new parent "Pause"  pgauftrag)
31...   enddef(new parent "Weiter" pgauftrag)
32...   enddef(new parent "Fertig" pgauftrag)
33...
34...
35...   link (.partitur (any pgAuftrag)) to (.auftraege (any pgPartitur))
36...   link (.fuerwen  (any pg-interp)) to (.aktIpols (any pgPerson))
37...
38...; ------------------------------------------------
39...
```

Ende Quelltext

1.6 Die (erst ab APOS-5.0 standardmäßig eingebaute) Baumschreibweise macht die Klassenstruktur unterhalb von `pgAuftrag` deutlicher :

Quelltext 3

```
1....
2....  new parentTree
3....>  "PgAuftrag"  Obj     "partitur"  QuItem    "ausgefuehrt" OBJ
4....>                       "wann"      fixpoint  "wer"       pgPerson
5....>
6....>   |  "Pg-ParamChng"   "welcher"   ftok      "richtung"    ftok
7....>                       "wert"      ~access
8....>   |  |   "Pg-Interp"  "dauer"     fixpoint  "modus"       obj
9....>                       "fuerWen"   QuItem    "startwert" univskal
10...>   |  "Start"          "wo"        pgOrt     "wohin"       pgOrt
11...>
12...>   |  "Umkehr"
13...>   |  "Pause"
14...>   |  "Weiter"
15...>   |  "Fertig"
16...;
17...   link (.super (any pgAuftrag)) to (.auftraege (any pgPartitur))
18...
19...; ------------------------------------------------
20...
```

Ende Quelltext

1.7 Die *Objekte* aus den *Subklassen* von `pgAuftrag` werden beim Parsieren der Benutzereingaben generiert und verknüpft, um die Struktur der Benutzereingabe festzuhalten.

Gleichzeitig dienen die Klassen-Objekte (= „`parent`-Objekte") als Schlüsselworte zur Parsierung der Benutzereingaben.

Die Grammatikregeln oben vorgestellter Definition von PGP-0.2 finden wir leicht wieder in Quelltext 4 Zeile 22, Quelltext 4 Zeile 34, Quelltext 5 Zeile 12, Quelltext 5 Zeile 18, Quelltext 5 Zeile 30 etc.

Man erkennt z.B. deutlich, daß in Quelltext 5 Zeile 16 das „Schlüsselwort" als Parameter eines `new`-Befehls verwendet wird, welches in Quelltext 5 Zeile 12 ff. geparsed wurde.

Quelltext 4

```
; HILFSROUTINEN :

 dm [ new pgPartitur (any name) ]
>    [ with [_0_s1_2]
>           [' init .auftraege __0 ; back __0]]
;
 dm [ ~do .protokoll (any pgPartitur) (any ~tail) ]
>    [ showcr ['_r3] ;
>      if [@ _1_2] ['@ _1_2 [''  ['splice __1] [''_r3] ] ]
>                  ['@ _1_2 [''[''_r3] ] ]
>    ]
;
; ------------------------------------------------------------------------
;  1.  EINGABESYNTAX
;      =======
;  1.1  partiturrahmen und kopf
; ------------------------------------------------------------------------

 new syslatch "actPgp" pgpartitur

 new token "begin"
 dm [ begin pgPartitur (any name) ]
>    [ with [new _r1]
>           ['GROW __0 ; CONSULT __0 ;
>             @ actpgp __0 ]]

 dm [ end pgPartitur ]
>    [ if [@ actpgp]
>         ['close consulting __1 ; close growing __1 ;
>           @ actpgp NULL]
>         ['!XP "status, no score open !"]
>    ]
;
 dm [ new pgPerson (any name) (any ftok)(any ~tail) ]
>    [ with [back[_0_1_2]]
>           ['init __0_r3 ;
>             apl ['' .aufwaerts .abwaerts .absatz ]
>                 ['' apl [''' .schritte/sec .stufen/schritt]
>                         [''' test __0___0____0 ]
>                  ]]]
;
```

```
42...   dm [ test (any pgPerson) (any ftok)(any ftok) ]
43...>     [ if [[@@ _1_2_3] ?eq (fixpoint 0.0)]
44...>         [' !XP "status, KEIN EINTRAG in " _1 " fuer " _r2 ]
45...>      ]
46...;
47...   dm [ init (any pgPerson) .klangtyp (any pgKlangTyp) (any ~tail) ]
48...>     [ _0_1_2_3 ; _0_1_r4 ]
49...;
50...   dm [ init (any pgPerson) .klangtyp (any pgKlangTyp) ]
51...>     [ @ _2_1_3 ]
52...   dm [ init (any pgPerson)(any ftok)(any ftok)(any fixpoint)(any ~tail)](
53...>  dm [ init (any pgPerson)(any ftok)(any ftok)(any integer)(any ~tail)]
54...>     [ _0_1_2_3_4 ; _0_1_r5 ])
55...;
56...   dm [ init (any pgPerson) .aufwaerts .schritte/sec (any fixpoint) ](
57...>  dm [ init (any pgPerson) .abwaerts  .schritte/sec (any fixpoint) ](
58...>  dm [ init (any pgPerson) .absatz    .schritte/sec (any fixpoint) ](
59...>  dm [ init (any pgPerson) .aufwaerts .stufen/schritt (any fixpoint) ](
60...>  dm [ init (any pgPerson) .abwaerts  .stufen/schritt (any fixpoint) ](
61...>  dm [ init (any pgPerson) .absatz    .stufen/schritt (any fixpoint) ](
62...>  dm [ init (any pgPerson) .aufwaerts .abweichung     (any integer) ](
63...>  dm [ init (any pgPerson) .abwaerts  .abweichung     (any integer) ](
64...>  dm [ init (any pgPerson) .absatz    .abweichung     (any integer) ]
65...>     [ @ _3 [@ _2_1] _4 ])))))))))
66...;
```

Ende Quelltext

1.8 Quelltext 4 Zeile 22 und Quelltext 4 Zeile 34 bilden die *Benutzerschnittstelle*, weil sie einen *Syntaxgraphen der Sprache* PGP-0.2 realisieren!

1.9 Allen PGP-0.2-Befehle ist gemeinsam, daß sie mit Zeitpunkt und Person attributiert sind, und in eine Liste eingefügt werden. Diese allen Befehlen gemeinsamen Operationen werden realisiert von Quelltext 5 Zeile 6 ff., auf welche alle „Teil-Parser" zurückgreifen.

1.10 In Quelltext 4 Zeile 22 wurde das `pgPartitur`-Objekt als lexikalisches Ensemble lesend und schreibend geöffnet (Quelltext 4 Zeile 24), so daß alle in Quelltext 4 Zeile 34 neu erzeugten Person-Objekte im Lexikalischen Baum unmittelbarer Unterknoten des zu ihnen gehörenden Partitur-Objektes sind.

1.11 Beim Aufbau der Datenstruktur könnten noch mehr Konsistenztestes durchgeführt werden als jetzt (nämlich fast gar keine !). Siehe auch unten die ausführliche Diskussion des *Fehlerbegriffes* unter 8.3.

Quelltext 5

```
1....; ------------------------------------------------------------------------
2....;  1.  EINGABESYNTAX
3....;  1.2  einzelne Auftraege
4....; ------------------------------------------------------------------------
5....
6....>  dm [ new pgAuftrag (any fixpoint) (any pgPerson)  ]
7....>     [ with [new _1]
8....>           [' @ .wann __0_2 ; @ .wer __0_3 ;
9....>              @99 [@ .auftraege [actpgp]] __0 ;
10...>              back __0]]
11...;
12...   dm [ ! (any fixpoint) (any pgPerson) Pause  ](
13...>  dm [ ! (any fixpoint) (any pgPerson) Weiter ](
14...>  dm [ ! (any fixpoint) (any pgPerson) Fertig ](
15...>  dm [ ! (any fixpoint) (any pgPerson) umkehr ]
16...>     [ back [new _3_1_2 ] ] )))
17...;
18...   dm [ ! (any fixpoint) (any pgPerson) start
19...>                     .wo (any block) .wohin (any block) ]
20...>     [ with [new _3 _1_2]
21...>           [' @ _3__0 [pgOrt _4] ;
22...>              @ _5__0 [pgOrt _6] ;
23...>              back __0]]))
24...;
25...   dm [ ! (any fixpoint) (any pgPerson) .wo  (any integer) ]
26...>     [ with [back[new pg-ParamChng _1_2 ]]
27...>           [' @ .welcher __0 _3 ; @ .wert __0_4 ]]
28...;
29...   new loctok "~pchng"
30...
31...   dm [ ! (any fixpoint) (any pgPerson) .aufwaerts (any ftok) (any ~tail) ](
32...>  dm [ ! (any fixpoint) (any pgPerson) .abwaerts  (any ftok) (any ~tail) ](
33...>  dm [ ! (any fixpoint) (any pgPerson) .absatz    (any ftok) (any ~tail) ]
34...>     [ ~pchng _r1 ]
35...;
36...   dm [ ~pchng (any fixpoint) (any pgPerson) (any ftok)
37...>                              .schritte/sec    (any fixpoint) ](
38...>  dm [ ~pchng (any fixpoint) (any pgPerson) (any ftok)
39...>                              .stufen/schritt  (any fixpoint) ](
40...>  dm [ ~pchng (any fixpoint) (any pgPerson) (any ftok)
41...>                              .abweichung  (any integer) ]
42...>     [ new pg-ParamChng _r1 ]))
43...;
```

```
44...   dm [ ~pchng (any fixpoint) (any pgPerson) (any ftok)
45...>                    .schritte/sec (any fixpoint)   in (any fixpoint) ](
46...>  dm [ ~pchng (any fixpoint) (any pgPerson) (any ftok)
47...>                    .stufen/schritt  (any fixpoint) in (any fixpoint) ](
48...>  dm [ ~pchng (any fixpoint) (any pgPerson) (any ftok)
49...>                    .abweichung (any integer)   in (any fixpoint) ]
50...>     [ new pg_interp _r1 ]))
51...;
52...
53...   dm [ new pg-paramChng  (any fixpoint) (any pgPerson)
54...>               (any ftok)(any ftok) (any univskal)  ]
55...>     [ with [back[_0_1_2_3]]
56...>            [' @ .richtung  __0 _4 ;
57...>               @ .welcher   __0 _5 ;
58...>               @ .wert __0_6  ]])
59...;
60...   dm [ new pg-interp  (any fixpoint) (any pgPerson)
61...>               (any ftok)(any ftok) (any univskal) in  (any fixpoint) ]
62...>     [ with [back[_0_1_2_3_4_5_6]]
63...>            [' @ .dauer  __0 _7 ;
64...>               @ .modus  __0 NULL ]])
65...;
66...
67...
68...
```

Ende Quelltext

Quelltext 6

```
1....; ------------------------------------------------------------------------
2....;  1.3.  ANZEIGEN  der partiturobjekte
3....; ------------------------------------------------------------------------
4....  dm [ list (any pgPartitur) ]
5....>    [ open show _1 ;
6....>      apl all pgPerson in _1 ['_0__0] ;
7....>      showcr ;
8....>      apl [@ .auftraege _1]['_0__0] ;
9....>      close show
10...>    ]
11...; ------------------------------------------------------------------------
12...  dm [ list (any pgPerson) ]
13...>    [ show [[@ .name _1] xx__ 20 ] ; show .klangtyp _1 ; showcr ;
14...>      _r0 use .aufwaerts ; _r0 use .abwaerts ; _r0 use .absatz ; showcr ]
15...;
16...  dm [ list (any pgPerson) use .aufwaerts ](
17...> dm [ list (any pgPerson) use .absatz](
18...> dm [ list (any pgPerson) use .abwaerts ]
19...>    [ show [[@ .name _3]xx__ 12] ;
20...>      shfo [@ _3_1] ; showcr ]))
21...;
22...  dm [ list (any pgAuftrag) ]
23...>    [ show [@ .wann _1] "  " [[@@ _1 .wer .name] xx__ 20] "  " ]
24...;
25...  dm [ list (any umkehr) ](
26...> dm [ list (any weiter) ](
27...> dm [ list (any pause)  ](
28...> dm [ list (any fertig) ]
29...>    [ _0_s1 ; showcr [@@ _1 .parent .name] ])))
30...;
31...  dm [ list (any pg-Paramchng) ]
32...>    [ _0_s1 ; _r0 use .richtung ; _r0 use .welcher ;  showcr ]
33...;
34...>    [ if [@ _3_1] [' show ['@ .name _3] "  " __1 ]
35...>                  [' show "  " ]  ] ---}
36...;
37...  dm [ list (any pg-StartAuftrag) ]
38...>    [ _0_s1 ; _r0 use .wo  ; showcr ;
39...>      show ["" xx__ 36 ] ; _r0 use .wohin ; showcr ]
40...;
41...  dm [ list (any pg-Interp) ]
42...>    [ _0_s1 ; showcr "      IN " [@ .dauer _1] ]
```

Ende Quelltext

7.2 Eval-Befehl nebst Objektstrukturen. Synchrone Simulation.

Quelltext 7

```
1....; ------------------------------------------------------------------------
2....;  2.  AUSWERTEN der partiturobjekte
3....;  2.0   definition der DATENSTRUKTUR des ergebnisses.
4....; ------------------------------------------------------------------------
5....
6....   dm [ @+ .protokoll actpgp (any ~tail) ]
7....>     [ ~do _1 [@ _2][aktuelleZeit] _r3 ]
8....;
9....
10...  new parent "pgZevent" bytarr
11...  new field   pgZevent "wann"   fixpoint
12...  new field   pgZevent "hoehe"  fixpoint
13...  new field   pgZevent "sPer"   fixpoint
14...  new field   pgZevent "r"      ftok
15...  new field   pgZevent "wer"    pgPerson
16...; NORMIERTE umrechnungen:
17...;  (temporaere, hier eigentl. UNSICHTBARE felder)
18...  new field   pgZevent "nWann"  D32
19...  new field   pgZevent "nsper"  D32
20...  new field   pgZevent "record" obj
21...  new field   pgZevent "pause"  D32
22...;   ETC.
23...  enddef pgZevent
24...
25...  new typedarr "pgZPart" of 2000 pgZEvent
26...   new field   pgZPart    "num"        integer
27...   new field   pgzpart    "wer"        pgPerson
28...   new field   pgzpart    "freigabe"   ~access
29...  enddef pgZPart
30...
31...   dm [ listshow (any pgzevent) ]
32...>     [ show [@ .wann _1] "  " ;
33...>       apl ['       .hoehe .sper][' show __0_1] ;
34...>       show "  " ;
35...>       _r0 [@ .r _1] ;
36...>       showcr [@@ _1 .wer .name]
37...>     ]
```

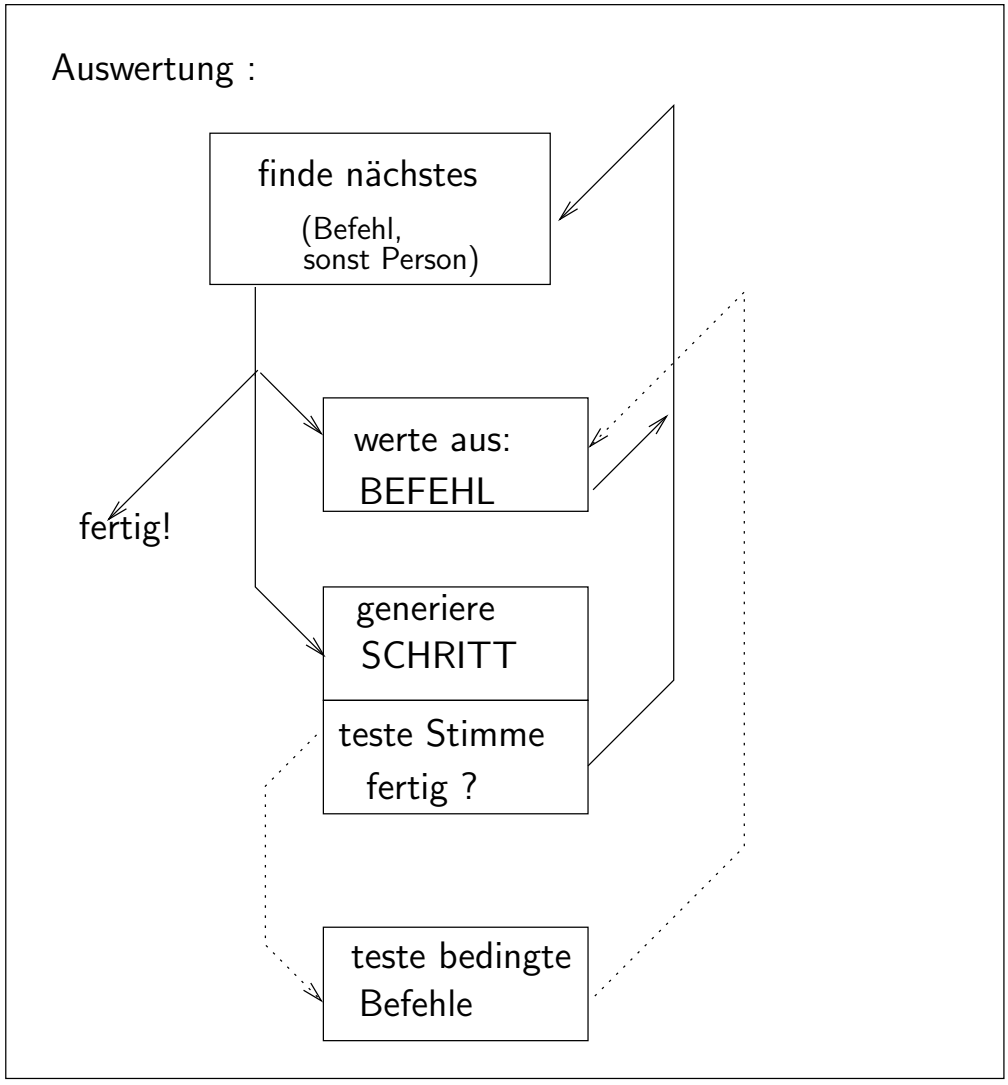


Abbildung 7.3: Der interne Ablauf eines eval-Schrittes.

```
======================== pgZpart : $OUT0 ========================
0    2.0       .hoehe   +  10.0   .sPer  +  0.83332   0  anton
1    2.83332   .hoehe   +  11.0   .sPer  +  0.83332   0  anton
2    3.66665   .hoehe   +  12.0   .sPer  +  0.83332   0  anton
3    4.49998   .hoehe   +  13.0   .sPer  +  0.83332   0  anton
4    5.13389   .hoehe   +  14.0   .sPer  +  0.63391   +  anton
5    5.87567   .hoehe   +  15.0   .sPer  +  0.74177   +  anton
6    6.58721   .hoehe   +  16.0   .sPer  +  0.62869   +  anton
7    7.21591   .hoehe   +  17.0   .sPer  +  0.79887   +  anton
8    8.01478   .hoehe   +  18.0   .sPer  +  0.67294   +  anton
9    9.79861   .hoehe   +  19.0   .sPer  +  1.11088   +  anton
10  10.63194   .hoehe   +  20.0   .sPer  +  0.83332   0  anton
... etc.
```

Abbildung 7.4: Beispiel eines `pgZPart`-Objektes.

```
38...;
39...   dm [ listshow (any pgzevent) .aufwaerts ]
40...>     [ show invers " + " normal ]
41...   dm [ listshow (any pgzevent) .abwaerts ]
42...>     [ show invers " - " normal ]
43...   dm [ listshow (any pgzevent) .absatz ]
44...>     [ show invers " 0 " normal ]
45...;
```

Ende Quelltext

Quelltext 8

```
1....
2....; ------------------------------------------------------------------------
3....;  2.  AUSWERTEN der partiturobjekte
4....;  2.1  hauptverteiler.
5....; ------------------------------------------------------------------------
6....
7....   new token "eval"
8....   new token "ZEITFORSCHALTUNG"
9....
10...   dm [ zeitfortschaltung (any pgpartitur) (any fixpoint) ]
11...>     [
12...{ --- HIER  real time scheduling nachher einfuegen !!!!
13...----------------}
14...        @ .aktzeit _2_3 ]
15...;
16...;                                anzahl d. generatoren
17...   dm [ eval (any pgpartitur) (any integer) ]
18...>     [ @ actpgp _1 ;
19...>       init _r0 ;
20...>       ~do _0_1 ; @ actpgp NULL ]
21...;
22...   dm [ init eval (any pgpartitur) (any integer) ]
23...>     [ @ .aktZeit _2 0.0 ;
24...>       grow _2 ;
25...>       apl 0 to [pred _3] [' init _2 pgzpart __0] ;
26...>       close growing ensemble ]
27...;
28...   dm [ init (any pgpartitur) pgzpart (any integer) ]
29...>     [ with [in _1 make _2 ["$OUT" ++ _3]]
30...>            [' @ .num __0_3 ]  ]
31...;
32...; ------------------------------------------------------------------------
```

```
33...; "HARTE" SCHLEIFE :
34...
35...   dm [ ~do eval (any pgpartitur) ]
36...>     [ _r1 [find next _2] ;
37...>       ifN [[aktuelleZeit] ?> [@ .dauer _2]]
38...>           ['_< _r0] ]
39...;
40...   dm [ eval (any pgpartitur) NULL ]
41...>     [ @+ .protokoll actpgp  " KEINE SCHRITTE oder AUFTRAEGE mehr ! " ;
42...>       showcr "auswertung fertig !!" ;
43...>       __< NOP ]
44...;
```

Ende Quelltext

Quelltext 9

```
1....   dm [ ~fnn pgperson (any pgpartitur) fixpoint#aktzeit
2....>                    (any pgperson)  (any obj) (any fixpoint) ]
3....>     [ if [ AND [@ .running  _4]
4....>                [not [@ .pause  _4]]
5....>                [[@ .demnaechst _4] ?>= #aktzeit]
6....>                [[@ .demnaechst _4] ?<  _6]       ]
7....>           [' _0_1_2_3 [.suc .ensemble _4] _4 [@ .demnaechst _4] ]
8....>           [' _0_1_2_3 [.suc .ensemble _4] _r5 ]
9....>     ]
10...;
11...   dm [ ~fnn pgperson (any pgpartitur) fixpoint#aktzeit
12...>                    (any lexobject)  (any obj) (any fixpoint) ]
13...>     [  _0_1_2 [.suc .ensemble _4] _r5 ]
14...;
15...   dm [ ~fnn pgperson (any pgpartitur) fixpoint#aktzeit
16...>                    NULL             (any obj) (any fixpoint) ]
17...>     [  back _5 ]
18...;
19...   dm [ find next (any pgpartitur) pgperson ]
20...>     [ ~fnn _3_2 [@ .actzeit _2] [@0 .subs _2] NULL [maxzeit]]
21...; ------------------------------------------------------------------------
22...   dm [ ~fnn pgauftrag (any pgpartitur) fixpoint#aktzeit
23...>                    (any pgauftrag)  (any obj) (any fixpoint) ]
24...>     [ if [ AND [not [@ .done   _4]]
25...>                [[@ .wann  _4] ?>= aktzeit]
26...>                [[@ .wann  _4] ?<  _6]       ]
27...>                [' _0_1_2_3 [.suc .partitur _4] _4 [@ .wann _4] ]
28...>                [' _0_1_2_3 [.suc .partitur _4] _r5 ]
29...>     ]
```

```
30...;
31...   dm [ ~fnn pgauftrag (any pgpartitur) fixpoint#aktzeit
32...>                      NULL             (any obj) (any fixpoint) ]
33...>     [  back _5 ]
34...;
35...   dm [ find next (any pgpartitur) pgauftrag ]
36...>     [ ~fnn _3_2 [@ .actzeit _2] [@0 .auftraege _2] NULL [maxzeit]]
37...; ------------------------------------------------------------
38...;
39...; STRATEGIE bei GLEICHEM zeitpunkt :
40...;     zuerst   auftraege
41...;    (dann     regeln)
42...;     dann     personen = event-generierung
43...
44...   dm [ find next (any pgpartitur) (any pgauftrag) (any pgperson) ]
45...>     [ if [ [@ .wann _3] ?<= [@ .demnaechst _4] ]
46...>          [' back _3]
47...>          [' back _4] ]
48...;
49...   dm [ find next (any pgpartitur) NULL (any obj) ]       [ back _4 ]
50...   dm [ find next (any pgpartitur) (any pgauftrag) NULL ] [ back _3 ]
51...;
52...   dm [ find next (any pgpartitur) ]
53...>     [ _r0 [_r0 pgauftrag] [_r0 pgperson] ]
```

Ende Quelltext

2.0 Ergebnisse des Auswertungslaufes sind zum einen ein Protokoll, wann welcher „Auftrag" tatsächlich ausgeführt wurde, zum anderen eine Zwischenpartitur je Generatorstimme (siehe Bild 7.4), die für jedes einzelne Ereignis eine Zeile mit allen notwendigen „technischen" Angaben enthält. Die Deklaration dieser Objekttypen geschieht in Quelltext 7 Zeile 6 resp. Quelltext 7 Zeile 10.

2.1 Beim Auswerten eines `eval`-Befehles werden als erstes soviel `pgZPart` - Objekte erzeugt, wie der Befehlsparameter als maximale Anzahl von Generatorstimmen angibt (Quelltext 8 Zeile 25).

Die aktuelle Zeit wird auf „`0000.0000`" gesetzt.

2.2 In der Hauptschleife (Quelltext 8 Zeile 35) wird nun in „simulierter Realzeit" jedesmal der zeitlich früheste anstehende Evaluierungsschritt ausgeführt. Dies ist entweder ein `pgAuftrag` (Befehl der vormals geparsten Benutzereingabe) oder die Generierung eines Schritte in der `pgZPart` für eine momentan laufende `pgPerson`.

Quelltext 10

```
;  ------------------------------------------------------------------------
;   2.  AUSWERTEN der partiturobjekte
;   2.2  auswerten der AUFTRAEGE.
;  ------------------------------------------------------------------------

    new loctok "~killipol" "~readipol"  "~generate"

    dm [ eval (any pgpartitur) (any pgAuftrag) ]
       [ ZEITFORTSCHREITUNG _1  [@ .wann _2] ;
         ~do _0_2 [@ .wer _2] ;
         @ .done  _2 OK ]
;

    dm [ ~do eval (any pg-StartAuftrag) (any pgperson) ]
       [ if [@ .running _3]
            ['!XP "parameter, person rennt schon/noch !"] ;
         @ .wo _3 [@ .wo _2] ;
         @ .wohin _3 [@ .wohin _2] ;
         @ .pause _3 NULL ;
         @ .running _3 OK ;
         @ .demnaechst _3 [@ .wann _2] ;
{ - ATTENTION  ASSUME bei gleichem ".wann" wert werden ERST die auftraege
                 und DANN die (gleichzeitigen) STIMMEN ausgewertet !! }
         case [[@ .wo _2] CMP [@ .wohin _2]] of
              ((compare : "equal")) ['!XP "keine richtung, wo = wohin !"]
              ((compare : "above")) ['@ .richtung _3 .abwaerts]
              ((compare : "lower")) ['@ .richtung _3 .aufwaerts] ;
         @+ .protokoll actpgp "startauftrag ausgewertet fuer " _3 ]
;
;  ------------------------------------------------------------------------

    dm [ ~do eval (any UMKEHR) (any pgPerson) ]
       [ @ .richtung _3 [NOT[@ .richtung _3]] ]
;
    dm [ not .aufwaerts ] [ .abwaerts ]
    dm [ not .abwaerts ]  [ .aufwaerts ]
;
    dm [ ~do eval (any PAUSE) (any pgPerson) ]
       [ @ .pause _3 OK ]
;
    dm [ ~do eval (any WEITER) (any pgPerson) ]
       [ @ .demnaechst _3 [@ .wann _2] ;
         @ .pause _3  NULL ]
```

```
44...;
45...   dm [ ~do eval (any FERTIG) (any pgPerson) ]
46...>     [ @ .running _3 NULL ;
47...>       @+ .protokoll actpgp " PERSON  " _3
48...>              " beendet durch explizites ''FERTIG'' " ]
49...;
50...   dm [ ~do eval (any pg-PARAMCHNG) (any pgPerson) ]
51...>     [ _r0 use [@ .welcher _2] ]
52...;
53...   dm [ ~do eval (any pg-paramchng) (any pgPerson) use .wo ]
54...>     [ @ _5_3 [@ .wert _2] ]
55...;
56...   dm [ ~do eval (any pg-paramchng) (any pgPerson) use .schritte/sec ](
57...>  dm [ ~do eval (any pg-paramchng) (any pgPerson) use .abweichung ](
58...>  dm [ ~do eval (any pg-paramchng) (any pgPerson) use .stufen/schritt ]
59...>     [ @ _5 [@ [@ .richtung _2]_3] [@ .wert _2] ;
60...>       ~killIpol _3 [@ .richtung _2][@ .welcher _2]  ]))
61...;
62...; hier <FEHLT> die behandlung von "aktuelle richtung" und "all" !
63...
64...   dm [ ~do eval (any pg-interp) (any pgPerson) ]
65...>     [ ~killipol _3 [@ .richtung _2][@ .welcher _2] ;
66...>       shift qu _2 into [@ .actipol _3] ;
67...>       @ .startwert _2 [@@ _3 [@ .richtung _2][@ .welcher _2] ]
68...>     ]
69...;
```

Ende Quelltext

2.3 Das Auswerten eines anstehenden `pg-ParamChng` geschieht einfach, indem die den Zustand der „Person" (=Satzstimme) repräsentierenden Felder des `pgPerson`-Objektes entsprechend geändert werden (Quelltext 10 Zeile 56 ff.)

Dabei wird insbesondere ein evtl. gerade aktiver interpolationsauftrag „gekillt", da der explizite (spätere) Benutzerbefehl Priorität hat.

2.4 Ein Interpolationsbefehl `pg-interp` wird ausgewertet, indem ein evtl. gerade aktiver Interpolationsbefehl für dieselbe Person und Parameter gelöscht wird und der neue `pg-interp` einfach als „aktiv" markiert (hier: umgekettet!) wird (Quelltext 10 Zeile 64).

Die Auswertung ist also bezogen auf die Objektstruktur der eingegebenen „abstrakten Syntax" *destruktiv*, – eine Implementierung, welche eine „geclonte" Datenstruktur abarbeitet/abbaut, wäre wohl sauberer.

Quelltext 11

```
;  ------------------------------------------------------------------------
;   2.  AUSWERTEN der partiturobjekte
;   2.3  auswerten der personen, generierung der SCHRITTE.
;  ------------------------------------------------------------------------

    dm [ eval (any pgPerson) ]
       [ @ aktuellezeit [@ .demnaechst _1] ;
         if [@ .pause _1]['!XP "consistency, pausierende person selektiert ?"];
         if [?is .absatz [@ .wo _1]]
            [' _r0 .absatz] ['_r0 ['@ .richtung _1]]
       ]
;
    dm [ eval (any pgPerson) (any ftok) ]
       [ _r0 [@ _2_1] ]
;
    dm [ eval  (any pgPerson) (any ftok) (any bewegung) ]
       [ ~READIPOL _r1 ;
         if [[@ .abweichung _3] ?eq (integer 0) ]
            [' _r0 ['1.0 / ['@ .schritte/sec _3]]  ]
            [' _r0 ?>  [[@ random] ?> 0.5] ]
       ]
;
    dm [ eval  (any pgPerson)(any ftok)(any bewegung) ?> NULL ]
       [ _0_1_2_3[   [1.0 / [@ .schritte/sec _3]]
                   * [random [1.0 / [1.0 + [[@ .abweichung _3] / 100 ]]]
                             1.0
                     ]] ]
;
    dm [ eval  (any pgPerson)(any ftok)(any bewegung) ?> OK ]
       [ _0_1_2_3[   [1.0 / [@ .schritte/sec _3]]
                   * [random 1.0
                             [1.0 + [[@ .abweichung _3] / 100 ]]
                     ]] ]
;
```

```
35...;                                          tatsaechliche schrittDauer !
36...   dm [ eval  (any pgPerson)(any ftok)(any bewegung)(any fixpoint) ]
37...>     [ ~generate  .wann  [@ .demnaechst _1]
38...>                  .hoehe [@ .wo _1]
39...>                  .sPer  _4
40...>                  .typ   [@@ _1_2 .klangtyp]
41...>                  .r     _1
42...>                  .wer   _1 ;
43...>       if [test .wohin _1 [@ .wo _1][@ .wohin _1]]
44...>          [' __< @ .running _1 NULL ] ;
45...>       _0 .wo _1 [@ .stufen/schritt _3] [@ .richtung _1] ;
46...>       += .demnaechst _1_4 ;
47...>       show "."
48...>      ]
49...;
50...; ------------------------------------------------------------------------
```

Ende Quelltext

Quelltext 12

```
1....
2....   dm [~readipol (any person) .aufwaerts (any bewegung) ](
3....>  dm [~readipol (any person) .abwaerts  (any bewegung) ](
4....>  dm [~readipol (any person) .absatz    (any bewegung) ]
5....>     [ apl [@ .actipol _1]
6....>           [' if [' ['@ .richtung __0] ?eq _2]
7....>                 ['' _r0__0 ] ]]
8....;
9....   dm [~readipol (any person) ftok#r  bewegung#bew pg-interp#ipol ]
10...>     [ _r0 [@ .modus #ipol][@ .startwert #ipol][@ .wert #ipol]
11...>           [@ .wann  #ipol][@ .demnaechst _1][@ .dauer #ipol] ]
12...;
13...   dm [~readipol (any person) ftok#r  bewegung#bew pg-interp#ipol
14...>                NULL  univkskal#startwert univksal#zielwert
15...>                fixpoint#startzeit fixpoint#nun fixpoint#zielzeit ]
16...>     [ @ [@ .welcher #ipol] #bew
17...>         [#startwert + [    [#zielwert - #startwert]
18...>                         *  [ [#now - #startzeit] / #dauer ]
19...>                       ]]]
20...;
```

```
21...;  NULL steht fuer linear, OK fuer exponentiell <FEHLT> PROVIS!!
22...   dm [~killipol (any person) ftok#ri ftok#welcher ]
23...>     [ _r0 [@0 .actipol _1]]
24...   dm [~killipol (any person) ftok#ri ftok#welcher NULL ]
25...>     [ nop ]
26...   dm [~killipol (any person) ftok#ri ftok#welcher pg-interp#ipol]
27...>     [ with [.suc .fuerwen #ipol]
28...>           [' if ['    ['['@ .welcher #ipol]?eq #welcher]
29...>                   and ['['@ .richtung #ipol]?eq #ri] ]
30...>                ['' @- .fuerwen #ipol ] ;
31...>             _0_1_2_3 __0 ]]
32...;
```

Ende Quelltext

2.5 Das Auswerten einer `pgPerson` bedeutet das generieren eines Schritt-Geräusches. Von Quelltext 11 Zeile 6 bis Quelltext 11 Zeile 48 werden die Parameter gesammelt und bestimmt. Insbesondere wird in Quelltext 11 Zeile 17 durch Aufruf von Quelltext 12 Zeile 2 überprüft, ob Interpolationsaufträge aktiv sind und die entsprechenden Werte im `pgPerson`-Objekt aktualisiert werden müssen.

2.6 Sind alle Parameter beisammen, so wird in Quelltext 11 Zeile 37 zu guter Letzt eine Zeile in der `pgZpart` generiert.

2.7 Abschließend wird der Zeitpunkt des *nächsten* Schrittes (sofern keine Aufträge davor interferieren) berechnet und der `.ort` der `pgPerson` entsprechend weitergeschaltet (Quelltext 11 Zeile 45).

2.8 Die bequemere Schreibweise „`klasse#paramname`" mit *benannten* Parametern (z.B. Quelltext 12 Zeile 9) steht erst ab APOS-5.x zur Verfügung.

Quelltext 13

```
1....
2....   dm [ eval  .wo (any pgperson) (any fixpoint) .aufwaerts ]
3....>     [ += _2_3_4 ]
4....   dm [ eval  .wo (any pgperson) (any fixpoint) .abwaerts ]
5....>     [ -= _2_3_4 ]
6....;
7....   dm [ ?is .absatz (any pgOrt) ]
8....>     [     [     [[[_2 + 1000.0](trasa u32) DIV $1.0](trasa INTEGER)]
9....>             MOD [hoehe/geschoss]]
10...>       ?<  [absatz/geschoss]
11...>     ]
12...;
```

```
13...    dm [ test .wohin  (any pgperson) (any pgort)(any pgort) ]
14...>      [ if [_4 ?<= [pgMinUnend]]
15...>           [' @+ .protokoll actpgp " PERSON   " _3
16...>                  " erreicht UNTERES nirwana. "  ;
17...>              __< back OK] ;
18...>        if [_4 ?>= [pgMaxUnend]]
19...>           [' @+ .protokoll actpgp " PERSON   " _3
20...>                  " erreicht OBERES nirwana. "  ;
21...>              __< back OK] ;
22...>        if [[ABS[_5 - _4]] ?<= [EpsilonOrt]]
23...>           [' @+ .protokoll actpgp " PERSON   " _3
24...>                   " hat ''.WOHIN'' erreicht. "  ;
25...>              __< back OK] ;
26...>         back NULL    ]
27...;
```

Ende Quelltext

Quelltext 14

```
1....;  2.3.2 HILFSROUTINEN  allokation und freigabe von
2....;   generatorstimmen
3....; ------------------------------------------------------------------------
4....
5....  dm [ ~generate  .wann (any fixpoint) .hoehe (any integer)
6....>                 .sper (any fixpoint) .typ   (any pgklangtyp)
7....>                 .r    (any ftok)     .wer   (any pgperson) ]
8....>    [ _0 _2_4_6_8_a_c [find pgzpart for _b_2] ]
9....;
10...  dm [ find pgzpart for pgperson#pers fixpoint#wann  ]
11...>    [ apl all _1 in [@ .ensemble _3]
12...>         [' if [' and [' [' @ .wer __0] ?eq #pers]
13...>                      [' [' @ .freigabe __0] ?<= #wann] ]
14...>               ['' ___< back __0] ;
15...>      apl all _1 in [@ .ensemble _3]
16...>         [' if [' [' @ .freigabe __0] ?<= #wann ]
17...>               ['' ___< back __0] ;
18...>      back NULL ]
19...;
20...; ==> wg lesbarkeit der pgzpart fuer menschen :
21...;        moeglichst kein ueberfluessiger wechsel
22...;        der generatorstimme bei selber pgperson !!
23...;        Suche nach der AELTESTEN freigabe <FEHLT> !
```

```
24...  dm [ ~generate  fixpoint#wann integer#hoehe
25...>                 fixpoint#sper pgklangtyp#typ
26...>                 ftok#r        pgperson#wer   pgzpart#part)   ]
27...>    [ if [[@ .freigabe #part] ?> #wann ]
28...>         [' @+ #part .dura  [#wann -[@ .freigabe #part]]
29...>                     .wer ~~Pause ] ;
30...>      @+ #part .dura [@@ #typ #r .length] .wer #wer .mat [@@ #typ #r]
31...>                      .r #r  .wo #hoehe ;
32...>      @ .wer #part #wer ;
33...>      @ .freigabe [#wann + [@@ #typ #r .length]] ;
34...>    ]
35...;
36...  dm [ ~generate  fixpoint#wann integer#hoehe
37...>                 fixpoint#sper pgklangtyp#typ
38...>                 ftok#r        pgperson#wer   NULL  ]
39...>    [ !XP "memory, keine generatorstimme frei !" ]
40...;
```

Ende Quelltext

2.9 Das Eintragen in eine Generatorstimme erfolgt, indem zunächst eine freier Generator gesucht wird (Quelltext 14 Zeile 10).

Ist dies erfolglos, so wird eine `EXCEPTION` generiert (Quelltext 14 Zeile 39), da die Benutzer-Partitur mit der angegebenen Anzahl von Generatoren nicht realisierbar ist, – was bei überlappenden Schritten (= Schallaufzeichnungsdauer > Schrittfrequenz) leicht möglich ist.

2.10 Nach Eintrag in die Generatorstimme wird der frühest mögliche Zeitpunkt berechnet und in das Feld `.freigabe` eingetragen, zu dem die Generatorstimme für ein weiteres Ereignis wieder zur Verfügung steht (Quelltext 14 Zeile 33).

2.11 Mit den erzeugten `pgZPart`-Objekten ist bereits ein *Technischer* Datensatz generiert. Dieser ist aber ein allgemeiner und muß zum „Abspielen" auf unterschiedlichen Geräten unterschiedlich weiterübersetzt werden.

Für die AUDIAC-Hardware müssen sog. „`PFN`s" erzeugt werden (s.u. 7.3.4), was im übernächsten Abschnitt dargestellt wird.

7.3 Exkurs: PFG, `Pfcb` und `Pfn`.

7.3.1 Der Hardware-PFG.

Der PFG war ein wesentlicher Bestandteil der AUDIAC-Hardware-Architektur, auf den seine Erfinder, Dipl.-Ing. Gerhard Kümmel und Dr. Helmut Zander, besonders stolz waren, – mit Recht[2].

Der PFG (= Parameter-Funktions-Generator) ist eine Hardware-Komponente, welche 256-fach gemultiplext stückweise linear definierte Funktionen über der Zeit (mit einer Genauigkeit von 1/4 der eingestellten System-sampling-rate) erzeugt.

Die Realisierung der Hardware erfolgte durch Register- und Ram-Bausteine und eine auf raffinierte Weise mehrfach genutzte ALU, - die Steuerung bestand in einem (festgelegtem) Micro-code.

Bild 7.5 zeigt den schematischen Aufbau der PFG-hardware, Bild 7.7 zeigt den Quelltext des Microprogrammes, Tabelle 7.2 listet die Microprogramm-Bits und 7.8 das Impulsdiagramm (aktive Bits als Ziffer!). Bei letzteren stehen die caselables „`IO`", „`NO`", etc. für die Fallunterscheidung von Kontrollbits der `Pfn` „kreuz" Bool'scher Wert des Testes auf „Zielwert erreicht oder überschritten?", also für die Fälle $\{\texttt{ImmediateReturn}, \texttt{Next}, \texttt{Hold}, \texttt{Return}\} \times \texttt{Boolean}$.

Bild 7.6 zeigt den (leicht vereinfachten) Algorithmus, der in die Pfg-Hardware gegossen wurde, in einem Pseudo-Pascal.

Die zu generierenden Funktionen wurden als Folge von Segmenten angegeben, von denen jeweils sechs(6) pro gemultiplextem PFG in die Hardware geladen werden konnten.

Jedes Segment konnte entweder als linear interpolierendes Segment mit definierter Steigung und definiertem Zielwert angegeben werden, oder als konstant gehaltenes Segment mit definiertem Wert und definierter Länge.

An das Erreichen des Endes eines Semgentes konnte jeweils die Auslösung eines *Interrupt*-Signals an den Kontrollrechner (= APC) gebunden werden, welches innerhalb der AUDIAC-Gesamtarchitektur dann mit Mitteln des ATOS-Realzeit-Betriebssystems konfigurierbare Reaktionen auslösen konnte.

Bild 7.9 zeigt das Beispiel einer PFG-Funktion und die entsprechenden Datensätze.

7.3.2 Treiber-Ebene: `Pfcb` und `pfServ`.

Über der Hardware lag die Ebene `Pfcb`, welche von der Beschränkung auf sechs Semgmente (und der abweichenden Behandlung des letzten Segmentes) abstrahierte.

Realisiert wurde diese Ebene einerseits durch eine Menge von Bibliotheksfunktionen, andererseits durch einen im Hintergrund laufenden Server-Prozeß („`pfServ`"), welcher das interrupt-gesteuerte automatische Nachladen der Hardware-Segmente unsichtbar für den Benutzer der `pfcb`-Schnittstelle leistete.

[2]Die Urheberschaft kann grob wie folgt zugeordnet werden: Die ursprüngliche Konstruktion ist Erfindung von H. Zander und G. Kümmel, spätere Verbesserungen und Erweiterungen (24-Bit-Breite, statische Segmente etc.) sind Erfindungen von G. Kümmel und dem Verfasser.

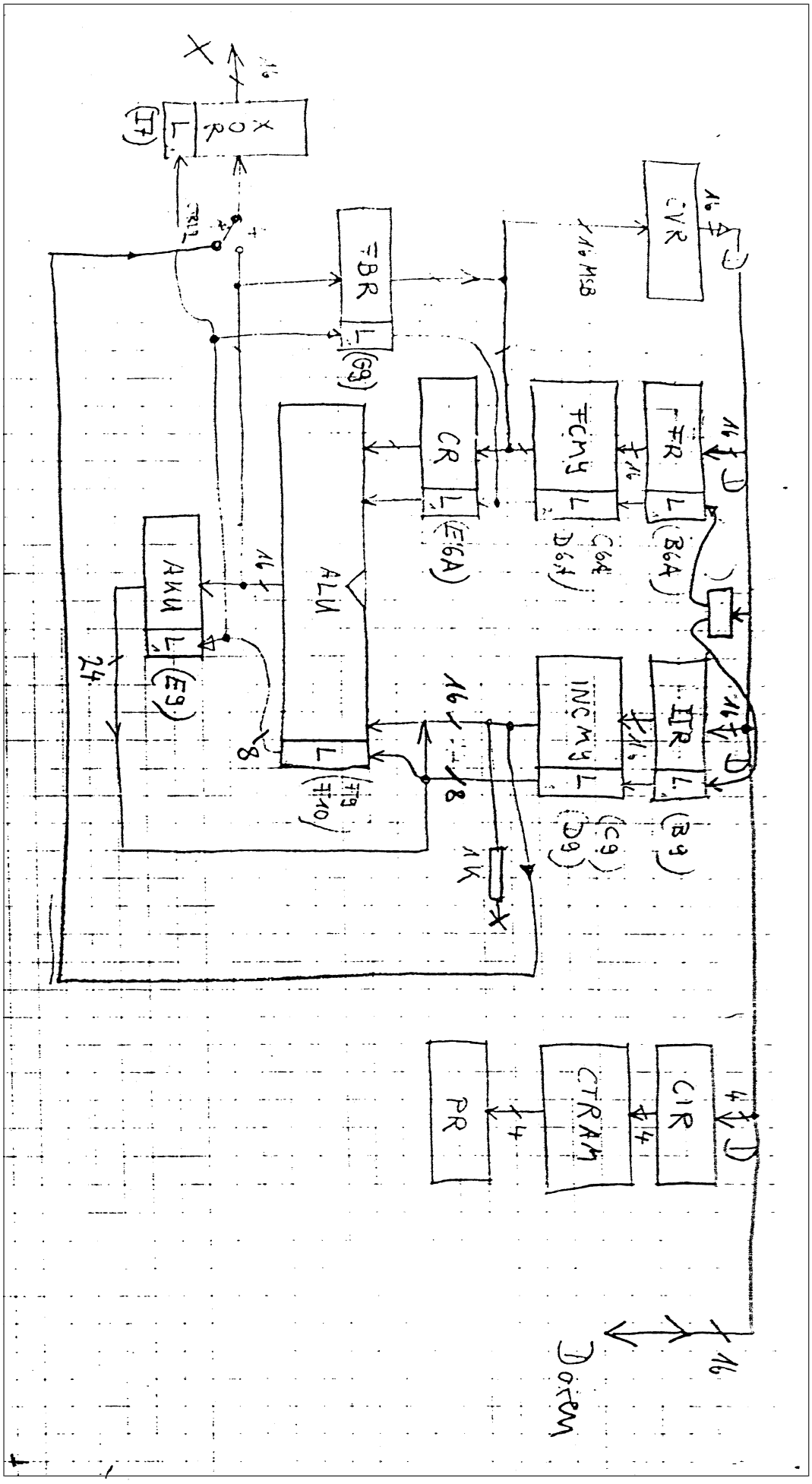


Abbildung 7.5: Schematischer Aufbau des PFG.

```
bit   name     <d>
 0    /IAW      1     IARAM - write
 1    /IRAE     1     IARAM - CS
 2     FRCK     0     FMAR-clock   (zerhackt !)
 3     RST//CA  1     IAROM umschalter (firstseg=1, CurAdr=0)
 4    /IAE      1     IAROM enable
 5    /ACE      1     counter-pal output enable
 6     ACCK     0     counter pal clock
 7     FK       0     Fifo Clock
 ------------------------------------------------------------------------
 8     CCK      0     CR-Reg-Clock
 9     AK       0     AKU clock
10    /AE       1     AKU enable
11     VS0      0     PFG alu ctrl : 00 = 0000h
12     VS1      0                    01 =         ???? ASK GK ( )
                                     10 = a-b
                                     11 = a+b
13     FBCK     0     FBR-Clock
14    /FBE      1     FBR enable
15    /CFW      1     CURFIN-memory write
 ------------------------------------------------------------------------
16     CTCK     0     Ctrl-pipeline-register clock
17    /IEN      1     INC-memory CS          nur  STAT=FALSE  mode
18     TS0      1     PAL 6  sequencing  00 =  clocke I0 I2
19     TS1      1                        01 =  clocke I1 I3 I4
                                         11 =  halte alle
20     VXC      0     X-Out-Register Clock   nur  STAT=FALSE  mode
21      n.c.    .
22     STCK     0     X-Out-Register Clock   nur  STAT=TRUE   mode
23    /STEN     1     INC-memory CS          nur  STAT=TRUE   mode
 ------------------------------------------------------------------------
```

Tabelle 7.2: Die Microprogramm-Bits des PFG.

```
input data [1 ... 6] ;
input cur ;
input output   actseg ;

local fin := data[actseg] .fin ;
local inc := data[actseg] .inc ;
local ctr := data[actseg] .ctr ;
local boolean done ;

output out ;

if IMMEDIATE_RETURN ∈ ctr then begin
  actseg := 1 ;
  out := cur ;
  end
else if HOLD ∈ ctr then
  out := cur ;
else begin
  if STAT ∈ ctr then
    new := cur + (-1) ;
  else
    new := cur + inc ;
  done := ( sign (cur - fin) ≠ sign (new -fin) )
  if done then begin
    if NEXT ∈ ctr then
      actseg := actseg + 1 ;
    else // RETURN ∈ ctr
      actseg := 1 ;
    if INT ∈ ctr then
      sendInterruptToApc
    end ; // done
  if STAT ∈ ctr
    out := inc ; // ((inc XOR fin) & 0xFFFF00) | (inc & 0x0000FF)
  else // non-static segment
    if done then
      out := fin ;
    else
      out := new ;
  end ;
```

Abbildung 7.6: Der Algorithmus der Hardware-PFG.

```
 Fall
 V Zyklus
 V V    RECHENEINHEIT                        PROGRAMMSTEUERUNG
 V V
 V 0                                         a IAROM[cur]     -> FMAR;
 V-------------------------------------------------------------------------
   1    b CUR -> CR                          c IARAM[fin/inc] -> FMAR;
                                               IARAM[fin/inc] -> PAL8;
 -------------------------------------------------------------------------
   2    d FIN -> CR                 // hardware Mux:
        e CR(=cur) + INC     = NEW  // ipol mode                          **a
    ODER: CR(=cur) + ffff.ff = NEW  // static mode
                       f  NEW -> AKU, FBR   g S1=sign(cur) -> I0;
                       f  NEW -> XOR        g S0=sign(inc) -> I2;         **z
                                            h CTRAM -> PR
 -------------------------------------------------------------------------
   3    i CR(=fin) - AKU(=new)               j S1=sign(fin) -> I1;
                                             j S0=sign(new) -> I4;
                                             j S2=sign(fin-new) -> I3;
 -------------------------------------------------------------------------
   4                                         k IAROM[cur]     -> FMAR;     (B)
                                             l COUNTUP(PAL8)
        m CR(=fin) + INC -> XOR                                  **d, **c, (C)
 -------------------------------------------------------------------------
 -------------------------------------------------------------------------
I0 5    n CUR -> CR                                                        (G)
I0 6    o CR(=cur) + INC(=0.0) -> XOR      p IAROM[firstSeg] -> IARAM      **b
I0 7    z UPDATE
 -------------------------------------------------------------------------
N0 5    q FBR(=new) -> CUR
N0 6
N0 7    z UPDATE
 -------------------------------------------------------------------------
R0 5    q FBR(=new) -> CUR
R0 6
R0 7    z UPDATE
 -------------------------------------------------------------------------
H0 5    n CUR -> CR                                                        (H)
H0 6    o CR(=cur) + INC(=0.0) -> XOR                                      **b
H0 7    z UPDATE
 -------------------------------------------------------------------------
I1 5    n CUR -> CR                                                        (G)
I1 6    o CR(=cur) + INC(=0.0) -> XOR       p IAROM[firstSeg] -> IARAM  **b
I1 7    z UPDATE
 -------------------------------------------------------------------------
N1 5    r CR(=fin) + INC(=0.0) -> XOR, FBR  u FIFO-WR                (E),**b
N1 6    s FBR(=fin) -> CUR                  v COUNTER -> IARAM
N1 7    z UPDATE
 -------------------------------------------------------------------------
R1 5    r CR(=fin) + INC(=0.0) -> XOR, FBR  u FIFO-WR                 (F),**b
R1 6    s FBR(=fin) -> CUR                  v IAROM[firstseg] -> IARAM
R1 7    z UPDATE
 -------------------------------------------------------------------------
H1 5    n CUR -> CR                                                           (H)
H1 6    o CR(=cur) + INC(=0.0) -> XOR                                         **b
H1 7    z UPDATE
 -------------------------------------------------------------------------
```

Abbildung 7.7: Der Micro-Code des PFG.

```
                          I0,1  N0,R0 H0,1  N1    R1
bit   name     <d>   01234  567   567   567   567   567
 0    /IAW      1    .....  .0.   ...   ...   .0.   .0.
 1    /IRAE     1    .00..  .00   ..0   ..0   .00   .00
 2     FRCK     0    .11..  1..   1..   1..   1..   1..
 3     RST//CA  1    0...0  0..   0..   0..   0..   0..
 ---------------------------------------------------------------------
 4    /IAE      1    0...0  00.   ...   ...   ...   00.
 5    /ACE      1    .....  ...   ...   ...   00.   ...
 6     ACCK     0    ..1..  ...   ...   ...   ...   ...
 7     FK       0    .....  ...   ...   ...   1..   1..
 ---------------------------------------------------------------------
 8     CCK      0    ..11.  .1.   ...   .1.   ...   ...
 9     AK       0    ...1.  ...   ...   ...   ...   ...
10    /AE       1    ...0.  ...   ...   ...   ...   ...
11     VS0      0    ..1.1  .1.   ...   .1.   1..   1..
 ---------------------------------------------------------------------
12     VS1      0    ..111  .1.   ...   .1.   1..   1..
13     FBCK     0    ...1.  ...   ...   ...   .1.   .1.
14    /FBE      1    .....  ...   00.   ...   00.   00.
15    /CFW      1    .....  ...   .0.   ...   .0.   .0.
 ---------------------------------------------------------------------
16     CTCK     0    ...1.  ...   ...   ...   ...   ...
17    /IEN      1    ....0  .00   ..0   .00   0.0   0.0
18     TS0      1    ..0..  ...   ...   ...   ...   ...
19     TS1      1    ..00.  ...   ...   ...   ...   ...
 ---------------------------------------------------------------------
20     VXC      0    ...1.  ..1   ...   ..1   .1.   .1.
21    /ARE      1    .0000  00.   00.   00.   00.   00.
22     STCK     0    ...1.  1..   1..   1..   1..   1..
23    /STEN     1    ..0.0  .00   ..0   .00   0.0   0.0
 ---------------------------------------------------------------------
```

Abbildung 7.8: Der Micro-Code des PFG als Impulsdiagramm.

Die Bibliotheksfunktionen[3] zur Behandlung von `Pfcb`s sind:

- `new pfcb (any integer)(any integer)(any integer)`
  Erzeugt ein neues pfcb-Objekt und initialisiert die diversen memory-mapped Hardware-Adressen. Angegeben werden mußte : Apu-Nummer, Prok-Nummer, Pfg-Nummer.

- `void pfHold (PfcbP p)`
  Hält einen laufenden PFG an (= *logisches* Anhalten auf Benutzerebene, muß in den Zustand der physikalischen Maschine umständlich hineinprojiziert werden, wobei komplexe race-conditions aufgelöst werden müssen !).

- `void pfCont (PfcbP p)`
  Läßt einen angehaltenen PFG weiterlaufen.

- `fixpoint pfRead (PfcbP p)`
  Lese momentan ausgegebenen Wert.

- `void pfWrite (PfcbP p, fixpoint value)`
  Setze unmittelbar den momentan ausgegebenen Wert eines PFGs. Dieser ist (normalerweise) angehalten, oder aber führt in seinem Lauf einen entsprechenden Sprung durch, um dann mit dem neuen Wert (folgend dem geladenen Programm) weiterzuarbeiten.

- `void pfFeed (PfcbP p)`
  Ausführen des Nachladens, wird nur vom Hintergrundprozess `pfServe` aufgerufen !

- `void pfLink (PfcbP p, PfTripleP dat, int len, bool loopmode )`
  Linken eines ungelinkten PFG an einen `Pfn` (s.u.) und aktivieren des Hintergrundprozess zum Nachladen.

- `void pfDislink (PfcbP p)`
  Anhalten des PFG und Rückgängigmachen eines evtl. vorangegangenen Linkens.

- `void pfEndferm (PfcbP p, int cnt)`
  Hebe die Halte-Wirkung der nächsten `cnt` in der `pfn` enthaltenen Fermaten-Bits auf.

- `void pfMultiEndferm (int maske, int cnt, PfcbP p0, ...)`
  Hebe die Halte-Wirkung des je `pcfb` nächsten anliegenden Fermaten-Bits auf, und zwar für alle `pfcb`s aus der angegebenen Liste, für die im `maske`-Wert das Bit mit derselben Position eins(1) ist.

[3]Die erste Funktion ist als APOS-Methode realisiert in A41PFD0, die übrigen als sog. „race functions", also ATOS-taugliche Assembler-Bausteine in den Dateien h3/PFD-4111.ASM und h4/PFNU4109.ASM.

- `void pfnUpdate (PfcbP p, pfTripleP dat, int segnum, fixpoint val, fixpoint dura )`
  Verändert das Segment der angegebenen `Pfn` mit der angegebenen Nummer so, daß der neue Endwert und die neue Dauer gelten, – egal ob das Segment statisch oder interpolierend ist.

  Falls dieses Segment bereits in die Hardware von `p` geladen ist, dann verändere auch den Hardware-Zustand entsprechend !

7.3.3 Benutzer-Ebene: `vPfg` und `Pfn`.

Noch eine Ebene darüber, innerhalb der Systemschale der *virtuellen Geräte*, lagen die dem Benutzer sichtbaren virtuellen Pfgs (= `vPfg`[4]).

Die Subklassen von `vPfg` widerspiegeln die sehr unterschiedlichen Verwendungszwecke und -verhaltensweisen, für die ein (Hardware-)PFG tatsächlich eingesetzt werden konnte.

Ein PFG, ergänzt um die Funktionalität der `pfcb`-Ebene, kann nämlich benutzt werden z.B. ...

- als Sequenzer,
- als Hüllkurven-Generator („one shot ADSR"),
- als Zeitgeber,
- als Funktionsgenerator[5],
- als Differenzier- oder Integrierglied,
- etc.

In unserer Fallstudie PGP wird lediglich die Sequenzer-Funktion benutzt.

7.3.4 PFN, das Darstellungsformat für Funktionen für die PFGs.

Die Datenstruktur, welche auf dieser Ebene die abzuspielende Funktion beschreibt, ist eine sog. `Pfn` (= Parameter FuNktion).

Ihre numerische Ausgabe konnte einerseits im Binärformat erfolgen (= technische Bitmuster zum Einladen in die Hardware: `fin`- und `inc`-Werte), andererseits im benutzerorientierten, besser lesbaren Format (Angabe von Dauer und Zielwert und der Entscheidung, ob ein interpolierendes Annähern oder ein statisches Halten des angegebenen Zielwertes gemeint ist.)

Die Bibliotheksfunktionen[6] zur Behandlung von `Pfn`s sind:

- `new pfn (any name)(any integer)`
  Erzeugt neue Pfn mit angegebener maximaler Anzahl von Segmenten.

[4] Diese hießen in der realen Implementierung leider auch nur `Pfg`.

[5] Bei einer System-sampling-Frequenz von 40 kHz kann ein Pfg im „`loop`"-Modus ein zyklisches Signal von maximal 1.600 Hz erzeugen, bei einer sampling-Frequenz von 10 kHz.

[6] Realisiert als APOS-Methoden in `h4/PFN0`.

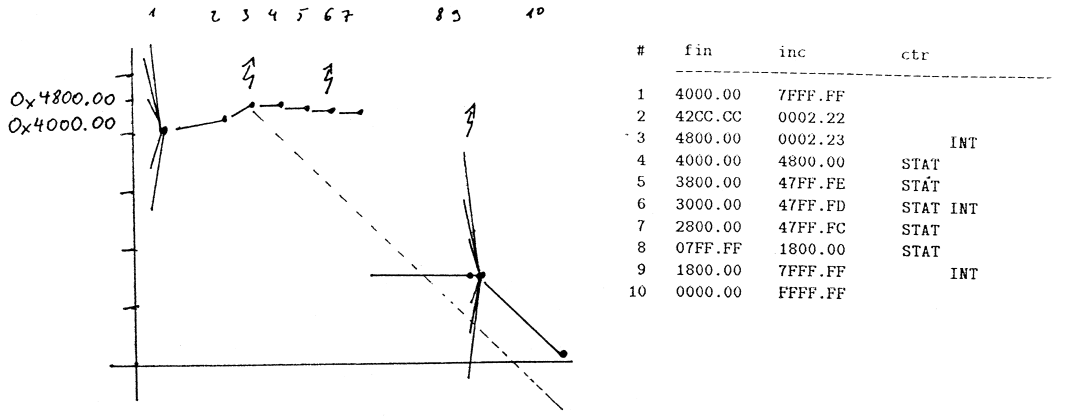


Abbildung 7.9: Beispiel für eine PFG-Funktion nebst `Pfn`-Darstellung.

- `open (any pfn) (any fixpoint)`
  Erzeugt ein initiales Segment, welches „sofort", also in maximal einem Schritt, den PFG auf den angegebenen Zielwert zieht.

- `PUT (any pfn) (any fixpoint) .inc (any fixpoint)`
  „Direkter Durchgriff" auf eine Ebene tiefer: erzeugt ein Semgment mit angegebenem Zielwert und angegebenem Inkrement.

- `@+ .ctr (any pfnWord)(any PfnBit)`
  Setze ein Kontrollbit der Treiber- und Benutzer-Ebene (gibt's nur eins: „`FERMATE`") oder ein Kontrollbit der technischen Ebene („`INT`". Direkter Durchgriff !).

- `PUT (any pfn) (any fixpoint) .dt (any fixpoint)`
  Erzeugt ein oder zwei Segmente, welche sich in der angegebenen Zeit dem Zielwert nähern. Da in der Hardware nur ganzzahlige Steigungen möglich sind, müssen i.A. zwei Segmente erzeugt werden, deren Steigung um eins (= `$0000.00100`) differiert und die zusammen die erwünschte gerade Verbindung zwischen letztem Wert und Zielwert durch eine Knickfigur annähern[7]

[7]Falls das vorhergehende Segment im „`static`"-Mode war, wird dieses im Nachhinein um einen Schritt verkürzt und ein non-static Segment eingeführt, damit die Funktion der fin- und inc-Register

- `PUT (any pfn) (any fixpoint) STATIC .dt (any fixpoint)`
  Erzeugt ein oder zwei Segmente. Das erste zieht den PFG „sofort"auf den angegebenen Zielwert, das zweite hält diesen für die angegebene Zeit (abzüglich dem evtl. eben verbrauchten Schritt).

- `PUT (any pfn) (any fixpoint) RAMP .inc (any fixpoint) .dt (any fixpoint)`
  Erzeugt eine „Rampe" aus einem ansteigenden, einem flachen und einem absteigenden Segment mit maximaler Steigung `.inc` zum knackfreien Erreichen eines Werte.

- `close (any pfn)`
  `close (any pfn) .loop`
  Beendet die Defintion einer `Pfn`[8]

- `copy (any pfn)to (any pfn)(any fixpoint)->(any fixpoint)`
  Bildet einen *Ausschnitt* aus der ersten `Pfn` zwischen den angegebenen Zeitpunkten (wobei Segmente auch *teilweise* übernommen werden. Diese Zerlegung erfordert eine recht komplexe Simulation des in Hardware gegossenenen Auswertungsalgorithmus.) und hängt diesen Ausschnitt hinten an die zweite `Pfn` an.

- `list (any pfn)`
  `list (any pfn) for users`
  Alphanumerische Anzeige: Erste Form zeigt die technischen Daten (`fin`, `inc`, `ctr`). Form zwei konvertiert sie in lesbare Form (Zielwert und Dauer).

wieder dem non-static-Mode entsprechend ausgetauscht wird. Man sieht: Ausschnitte aus Pfn-Objekten sind *leider nicht kontextfrei komponierbar*!!

Falls die Steigung allzu gering ist, werden beliebig viele static-Segmente erzeugt, welche die gewünschte flache Bewegung annähern. Man sieht: Die Länge der low-level-Datenstrukturen (und damit der Speicherbedarf) ist aus der Länge der Funktionen auf Benutzerebene *leider nicht* trivial errechenbar.

[8]Leider mußte auf dieser Ebene schon die für die Treiber-Ebene notwendigen Interrupt-Bits eincodiert werden, da diese für die Verwendung auf beiden Ebenen gemuliplext wurden.

Derartige Anti-Orthogonalitäten hatte die AUDIAC-Hardware zuhauf und der Verfasser hat leider lehrreiche Jahre seines Lebens auf deren Ausbügeln verwenden müssen.

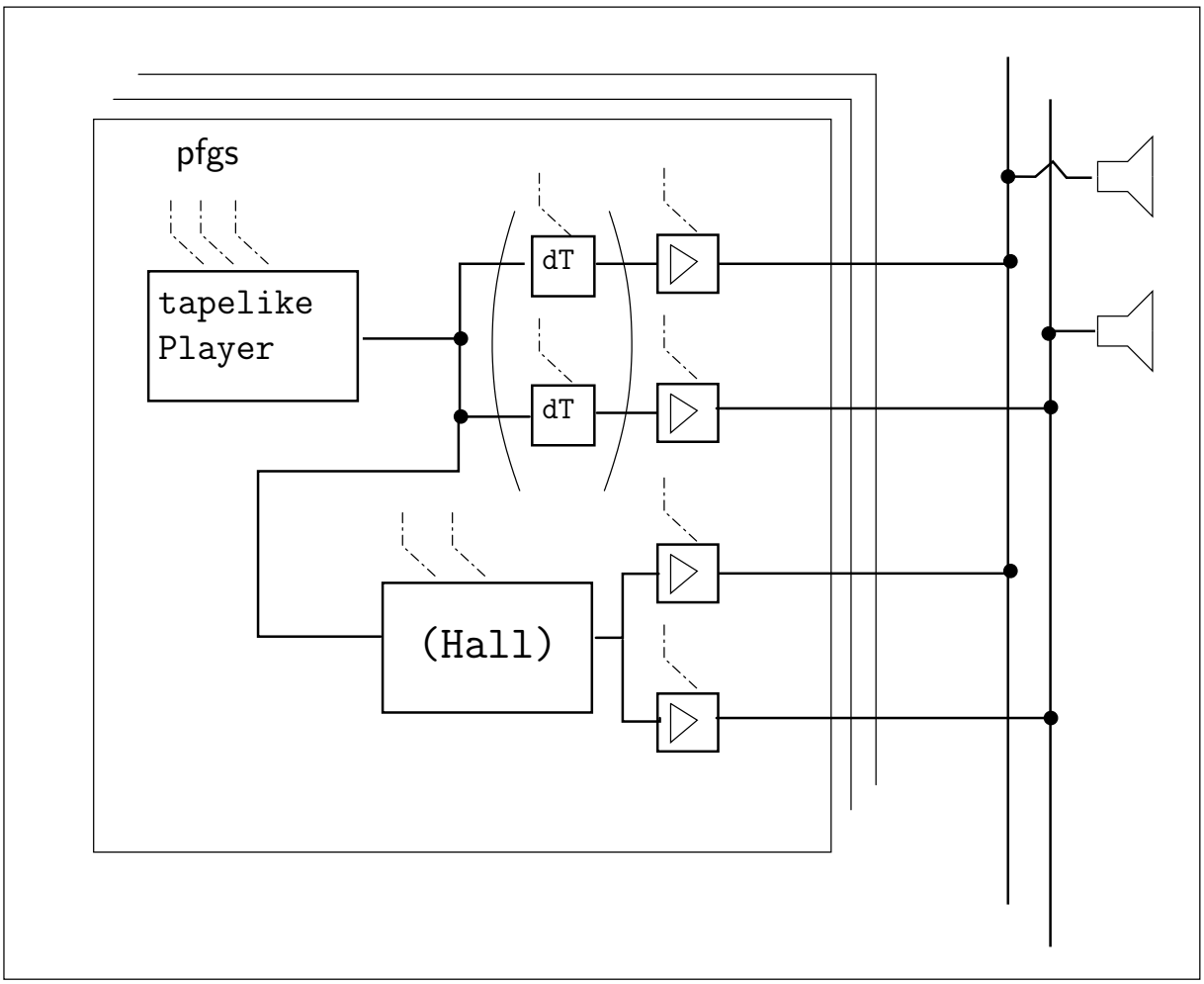


Abbildung 7.10: Die AUDIAC-Schaltung für PGP.

7.4 Abspielen. Klangsynthese mittels Realzeit-Audio-Processing.

Bild 7.10 zeigt die Konfiguration, welche im Projekt PGP im AUDIAC-Realzeitsystem (=Signal- und Kontrollprozessor, also APUs, APCs, AOC etc.) abgespielt wird, um die letztliche Klangliche Realisierung durchzuführen.

Man erkennte (je Generatorstimme) ein „`player`"-Device[9], welches die vorbereitete Schallaufzeichnung wiedergeben soll.

Sein Ausgang wird über regelbare Abschwächer auf die Summenschienen für die beiden Lautsprecher geleitet. Über diese Lautstärkeregelung wird eine erste Komponente der regelbaren Raumposition realisiert.

[9]Dies ist von der Klasse „`tapelike-player`", weil es wie eine Bandmaschine in verschiedenen Geschwindigkeiten gleich gut wiedergibt und stufenlose „Tape-Transpositionen" ermöglicht. Diese Möglichkeit wurde hier nicht verwendet, jedoch erfolgreich in anderen Projekten.

Zusätzlich könnte hier ein z.B. `delay`-Glied eingeschaltet werden, um zwecks Übersetzung der Raumposition auch die Phasenlage der beiden Lautsprechersignale im Feinstbereich zu regulieren. Überhaupt sind auf der Ebene der *zweiten* Übersetzung (von `pgZPart` nach `Tfn`) *unterschiedliche* Realisierungen des Autarken Quasi-Physikalischen Skalars „`pgOrt`" in verschiedenste „elektrische" Größen möglich, – je nach realisierender `konfiguration` (Vgl. unten 7.4.1)!

Parallel dazu wird das Ausgangssignal des `player` durch ein beliebiges (in unserem Fall recht primitives) Hallgerät gejagt und dessen Ausgang ebenfalls abgeschwächt den Lautsprechern zugemischt.

Alle dynamischen Parameter der Schaltung werden durch `Pfg`s gesteuert. Die `PFN`s für diese werden aus der `pgzpart` in einem zweiten Übersetzungsschritt abgeleitet (7.5). Sollten zusätzliche dynamische Parameter in einer erweiterten Konfiguration benötigt werden (z.B. Delay- oder Hallparameter), so ist dieser Übersetzungsschritt ebenfalls anzupassen.

Die pro Generatorstimme benötigten PFGs (und entsprechend `PFN`s) sind ...

- `player : .start`
  die Startadresse des zu spielenden Materials im MAU-RAM.
- `player : .length`
  die Länge in samples des zu spielenden Materials im MAU-RAM.
- `OO : .a0`
  `OU : .a0`
  `HO : .a0`
  `HU : .a0`
  die Amplituden der vier Abschwächer für Originalsignal/Hallsignal auf dem oberen/unteren Lautsprecher.

Bild 7.11 zeigt schematisch den `PFN`-Satz für eine Generatorstimme und das resultierende Signal (Von den vier (4) Amplitudenfunktionen ist nur eine dargestellt).

Einige Anmerkungen für Insider:

3.1 `.start` und `.length`-Parameter werden aus dem im Schrittevent (mittels Auswahl des `pgPerson`-Objektes) angegebene `pgKlangtyp`-Objekt errechnet.

Dazu werden *sämtliche* referierten `recording`s in *eine einzige* Mau-Ram-Partition (= „`mauPart`") konkateniert, die unterschiedlichen Startadressen *selbst* errechnet und (qua direktem Durchgriff auf die normalerweise dem Benutzer unsichtbare technische Ebene der `ressource`-Verwaltung) in die `.start-PFN`s einkodiert.

3.2 Die Übersetzung von `pgOrt` in die Amplituden der Abschwächer erfolgt über das Lesen aus *vier unterschiedlichen* Lautstärke-Kennlinien (siehe Bild 7.12, erzeugt in Quelltext 15 Zeile 22), – je eine für Originalsignal und Hall, je oben und je unten.

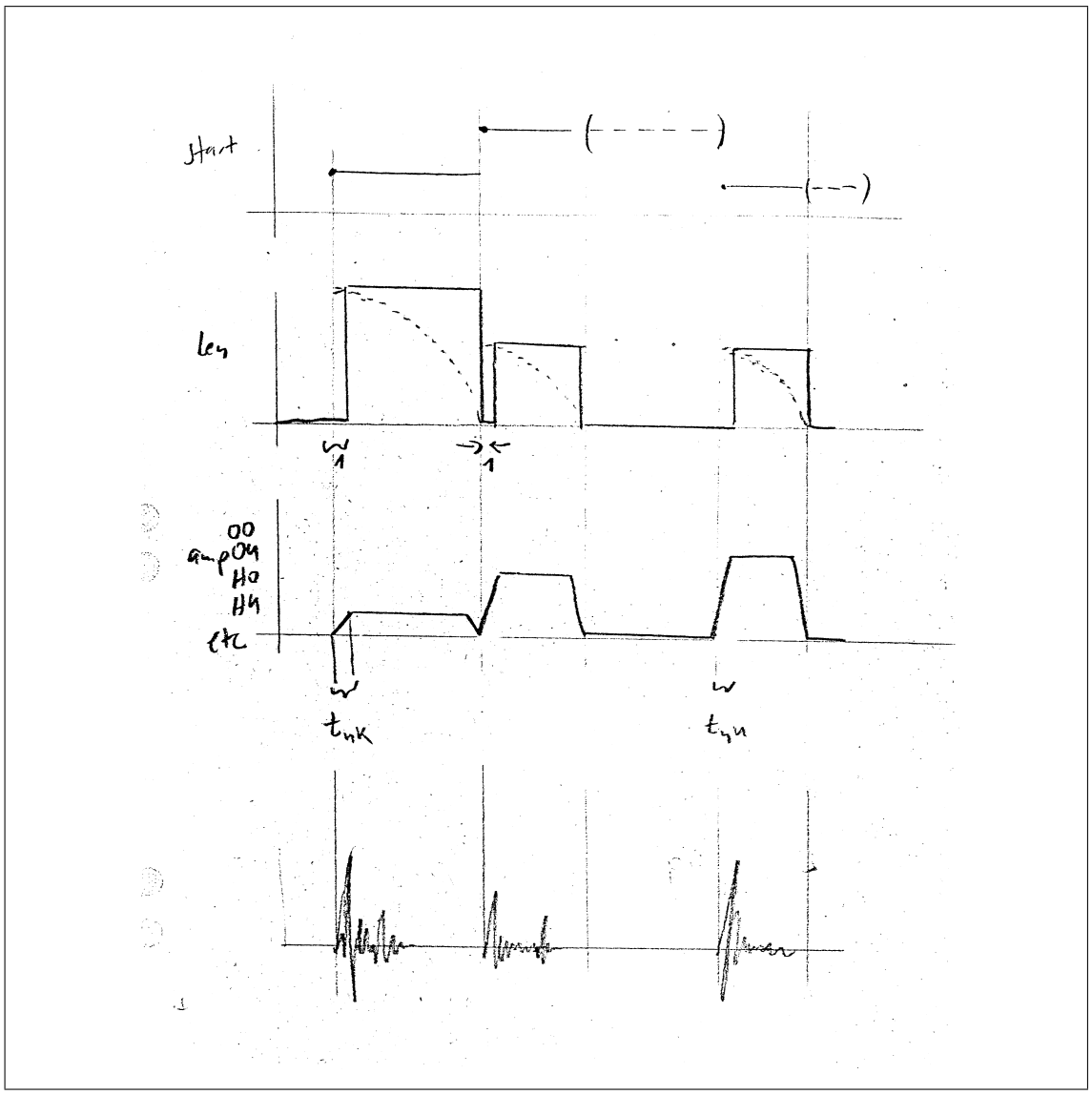


Abbildung 7.11: Der TDS für die AUDIAC-Schaltung. Gestalt der `Pfn`-Daten.

Diese können jeweils *einzeln* modifiziert werden. So können die Überlegungen am Schreibtisch und Experimente im Studio an die tatsächliche akustischen Verhältnisse in den zu beschallenden Treppenhäusern recht einfach angepasst werden.

3.3 Aus diesen Kennlinien wird nur ein Scheitelwert entnommen. Die tatsächlichen Amplitudenfunktionen bilden eine Rampe auf diesen Wert hin, deren begrenzte Steigung knackfreies Umschalten ermöglicht.

3.4 Diese Kennlinien waren ausgezeichnete Kandidaten zur geplanten Anbindung an die grafisch edierbaren `f<t>`- und `f<xy>`-Funktionen, welche leider aus bekannten Gründen nicht mehr durchgeführt werden konnte.

3.5 Man beachte, daß die `.length`-Funktion für jeden Schritt kurz auf Null (`00.0000`) gezwungen wird. Dies ist nötig, um den internen Ausgabezeiger des `player` mit der außen anliegenden Wavetable-Startadresse „`.start`" zu synchronisieren. Dieses verursacht aber auch ein kurzzeitiges „auf der Stelle spielen" und somit einen Gleichstromanteil, welcher durch entsprechende Gestaltung der `recording`-Objekte gleich Null (`0000.00`) werden sollte.

3.6 Die Zuordnung zwischen den `PFN`s eines TDS einer pgzpart und den PFGs einer Generatoreinheit der Schaltung geschieht mittels Methode ...

```
link (any pgzpart) to pgpKonfig (any integer)
```

... die ihrerseits von ...

```
play  (any pgPartitur)
```

... aufgerufen wird.

Deren Definitionen sowie den Quelltext, der die `konfig` erstellt, brauchen wir hier nicht anzuführen.

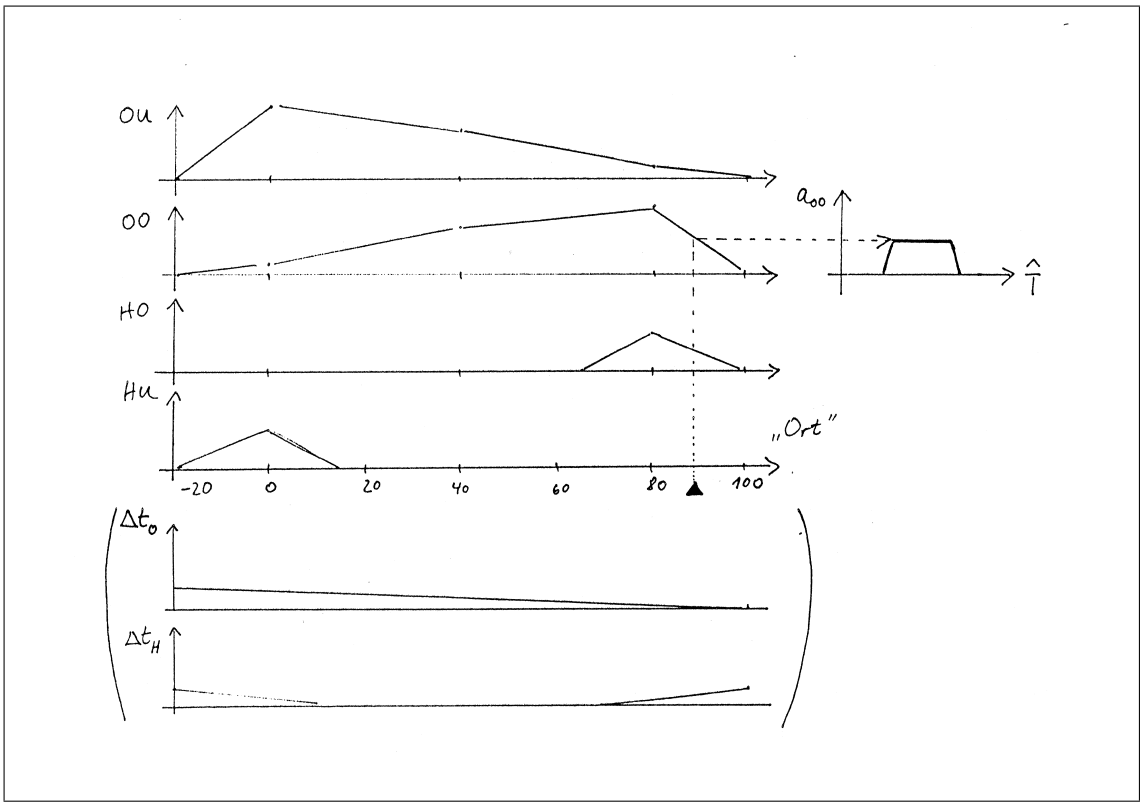


Abbildung 7.12: Kennlinien zur Realisierung von Ort durch Lautstärke.

7.4.1 Exkurs: Die Kontextabhängigkeit von Semantik und Übersetzung von Autarken Quasi-Physikalischen Skalaren.

Die verschiedenen Übersetzungsmöglichkeiten des Parameters Ort bieten ein schönes Beispiel für die Kontextabhängigkeit von Realisierung und Semantik-Definition solcher Quasi-Physikalischer Autarker Skalare.

Im Kontext unserer oben vorgestellten Schaltung wird die „gemeinte" Funktion „Zeit nach `Ort`" übersetzt in vier Amplitudenfunktionen.

Falls, wie oben angedeutet (7.4), Delay-Glieder hinzugefügt würden, können auch zwei Amplitudenfunktionen, zwei Delaydauern-Funktionen (= Phasen-Faktoren) und diverse Hallparameter-Funktionen das Ergebnis einer korrekten Übersetzung sein.

Falls wir hingegen unsere „Drehbücher" in andere Medien übersetzen, so könnte der selbe Parameter `Ort` gar als Kamera-Brennweite, Kamera-Schärfe oder Beleuchtungsfunktion übersetzt werden, – der neue TDS derselben Benutzerfunktion hätte ein ganz anderes Format.

7.5 Quelltexte der TDS- (=PFN-) Generierung.

Quelltext 15

```
1....; // alle folgenden Quelltexte vgl. old/olp5705/pg/ PGP2
2....   new token "fill"
3....
4....   new typedarr "amp<h>" wtb
5....   enddef amp<h>
6....
7....   dm [ fill (any amp<h>) in (any integer) to (any integer) ]
8....>     [ _r0 use [@99 _1]]
9....;
10...   dm [ fill (any amp<h>) in (any integer) to (any integer)
11...>                         use (any integer) ]
12...>     [ _r0 [ [_5 - _7] / _3 ]  ]
13...;
14...   dm [ fill (any amp<h>) in (any integer) to (any integer)
15...>                         use (any integer) (any fixpoint) ]
16...>     [ apl (integer 1) to [PRED _3]
17...>         [' @+ _1 ['_7 + ['__0 * _8]] ] ;
18...>       @+ _1_5 ]
19...;
```

```
20...   edit (new ensemble "pgAmpKennlinien")
21...
22...   new amp<h> "ampOU" 101
23...   @+ ampOU 0
24...   fill ampOu in   20 to (maxint)
25...   fill ampOu in   30 to 10000
26...   fill ampOu in   30 to 0
27...   fill ampOu in   20 to 0
28...
29...   new amp<h> "ampOO" 101
30...   apl ampOu [@ ampOO [100 - _1] _0]
31...
32...   new amp<h> "ampHU" 101
33...   @+ amphU 0
34...   fill amphu in   10 to (maxint)
35...   fill amphu in   10 to 0
36...   fill amphu in   80 to 0
37...
38...   new amp<h> "ampHO" 101
39...   apl ampHu [@ ampHO [100 - _1] _0]
40...
41...   end edit pgAmpKennlinien
42...   end edit userlex
43...; ------------------------------------------------------------------------
44...   new parent "pgRecording" lexobject
45...   new field   pgRecording "wo"      u32
46...   new field   pgRecording "wieLang" u32
47...   enddef pgRecording
48...; ------------------------------------------------------------------------
49...
50...   new latch "ampSegPerZPart"  2.5
51...   new sysconst "pfMaxDura"  (d32 $007f.fffc)
52...;; new sysconst "pg_playDropOut"  (+ (pf_mindt) (pf_mindt) (pf_mindt) )
53...   new sysconst "pg_playDropOut"  (pf_mindt)
54...   new token "sec"
55...
56...   dm [ list (any pgzpart) .nwann ]
57...>     [ open show _1 ;
58...>       apl _1 ['show __1 ;
59...>                apl [''.nwann .nsper .pause]
60...>                    [''show ___0__0] ;
61...>                showcr ] ;
62...>       close show ]
```

Ende Quelltext

Quelltext 16

```
1....
2....   dm [ give pfn (any pgZpart) ]
3....>     [ grow [new ftImpl for _2] ;
4....>       apl _2 ['@ .NWANN __0
5....>                  ['['['@ .wann __0] sec -> sample#] ROUNDUP $0.0004] ;
6....>                @ .NSper __0
7....>                  ['['['@ .sper __0] sec -> sample#] ROUNDUP $0.0004]
8....>              ] ;
9....>       apl _2 0 to [pred [valid .max _2]]
10...>                   ['~check .pause __0 ['@ _2['succ __1]] __1 ] ;
11...>       @ .pause [@99 _2] [pfMaxDura] ;
12...>       apl ['.hoehe .klangtyp .sper ]
13...>           [' compile pfn _2__0 ] ;
14...>       apl all pfn in [growing ensemble][' CLOSE __0] ;
15...>       close growing ensemble ;
16...>       showcr ['_r0] " DONE"
17...>     ]
18...;
19...   consult compare
20...   dm [ ~check .pause (any pgzevent)(any pgzevent)(any integer) ]
21...>     [
22...>        case [     [@ .nwann _3](trasa u32)
23...>               CMP [[@ .nwann _2] + [@ .nsper _2]](trasa u32) ]
24...>        of (lower)
25...>            ['if ['    [+ [@ .nwann _2] [@ .nsper _2]]
26...>                  ?eq [+ [@ .nwann _3] [pf_MinDt]] ]
27...>                [''__< -= .nsper _2 [pf_MinDt] ] ;
28...>              if ['    [+ [@ .nwann _2] [@ .nsper _2]]
29...>                   ?eq [+ [@ .nwann _3] [pf_MinDt][pf_MinDt]] ]
30...>                [''__< -= .nsper _2 [[pf_MinDt]+[pf_MinDt]] ] ;
31...>              !XP "consistency, UEBERLAPPENDE SCHRITTE"
32...>            ]
33...>           (equal) ['@ _1_2 (d32 $0000.0000) ]
34...>           (above) ['@ _1_2 ['  [@ .nwann _3]
35...>                                - [[@ .nwann _2] + [@ .nsper _2]] ] ] ;
36...>         show "c"
37...>      ]
38...;
39...   close consulting compare
```

```
40...; ------------------------------------------------------------------------
41...{--
42...   dm [ (any pgOrt) -> integer ]
43...>     [ [ [_0 + [pgMINUNEND]](trasa u32) DIV $1.0
44...>       ](trasa INTEGER)]
45...;  --}
46...   dm [ pgOrt (any fixpoint) -> integer ]
47...>     [ [ [_1 - [pgMINUNEND]](trasa u32) DIV $1.0
48...>       ](trasa INTEGER)]
49...;
50...{--
51...   dm [ (any fixpoint) sec -> pfSteps ]
52...>     [ [ [fixpoint 5000 0] * _0 ] trasa D32 ]
53...
54...; F and D - ASSUME srate = 50 khz ~ 20 usec
55...   dm [ (any fixpoint) sec -> pfSteps ]
56...>     [ [ ((fixpoint 5000 0)*(fixpoint 10 0)) * _0 ] trasa D32 ]
57...; --}
58...
59...; F and D - ASSUME srate = 50 khz ~ 20 usec
60...   dm [ (any fixpoint) sec -> sample# ]
61...>     [ [ (fixpoint (50 * 1000)  0) * _0 ] trasa D32 ]
62...;
63...; ------------------------------------------------------------------------
```

Ende Quelltext

Quelltext 17

```
1....   new token ".pause"
2....
3....   dm [ ~do .pause (any d32) (any pfn)(any amp<h>)
4....>                             (any pfn)(any ~tail)  ]
5....>     [ _0_1_2_3_4 ;_0_1_2 _r5 ]
6....;
7....   dm [ ~do .pause (any d32) (any pfn)(any amp<h>) ]
8....>     [ put _3 $0.0000 static .dt _2 ]
9....;
10...
11...; ------------------------------------------------------------------------
12...
13...   dm [ compile  pfn (any pgZpart) .hoehe ]
14...>     [ _r0 use pgAmpKennlinien ]
15...;
```

```
16...   dm [ compile  pfn (any pgZpart) .hoehe use (any ensemble)]
17...>     [ with [[@ .itemcnt _2] * [AmpSegPerZPart]]
18...>           [' _0_2_3  ['open['new pfn "aoo" __0]]
19...>                      [_5 : "ampOO"]
20...>                      ['open['new pfn "aou" __0]]
21...>                      [_5 : "ampOU"]
22...>                      ['open['new pfn "aho" __0]]
23...>                      [_5 : "amphO"]
24...>                      ['open['new pfn "ahu" __0]]
25...>                      [_5 : "amphU"]
26...>            ]]
27...;
28...   dm [ compile (any pgZpart) .hoehe (any pfn)(any ~tail) ]
29...>     [ with [@@ _1 (INTEGER 0) .nwann]
30...>           [' if ['__0 ?> (d32 $0.0) ]
31...>                 ['' ~do .pause __0 _r3 ]] ;
32...>       apl _1 [@ .indxMin _1] to [pred [valid .max _1]]
33...>           ['
34...> show "h" ;
35...>            _0__0_r2 ;
36...>            _0__0 ['@ _1['succ __1]] _r2 ] ;
37...>       _0 [@99 _1] _r2 ;
38...>       ~do .pause [pfMaxDura] _r3 ;
39...> showcr ]
40...;
41...;;;>       _0_2 .pause [pfMaxDura] _r3 ]
42...;
43...   dm [ compile (any pgZevent)(any pgZevent) .hoehe (any pfn)(any ~tail) ]
44...>     [ if [[@ .pause _1] ?eq (d32 $0.0)] ['__< NOP] ;
45...>       ~do .pause [@ .pause _1]  _r4 ]
46...;
47...
48...; - ASSUME recording IST LANG GENUG fuer ALLE .sper-werte !!
49...
50...   dm [ compile (any pgzEvent) .hoehe (any pfn)(any ~tail) ]
51...>     [ _0_2 [@ .hoehe _1][@ .nsper _1]_r3 ]
52...;
53...   dm [ compile .hoehe (any fixpoint) (any d32) (any pfn)(any ~tail) ]
54...>     [ _0_1 [pgOrt _2 -> integer] _r3 ]
55...;
```

```
56...   dm [ compile .hoehe (any integer)(any d32) (any pfn)(any amp<h>)
57...>                                              (any pfn)(any ~tail) ]
58...>     [ _0_1_2_3_4_5 ; _0_1_2_3 _r6 ]
59...;
60...   dm [ compile .hoehe (any integer)(any d32) (any pfn)(any amp<h>) ]
61...>     [ put _4  [pfformat xx__  [@ _5_2](TRASA u16)]
62...>               static .dt _3 ]
63...;
```

Ende Quelltext

Quelltext 18

```
1....
2....   dm [ compile  pfn (any pgZpart) .klangtyp ]
3....>     [ with [[@ .itemcnt _2] * [AmpSegPerZPart]]
4....>           [' _0_2_3  ['open['new pfn "f-f" __0]]
5....>                      .f-f
6....>                      ['open['new pfn "f-bw" __0]]
7....>                      .f-bw
8....>            ]]
9....;
10...
11...;;   SUBOPTIMAL bezg. pfn-laenge !!
12...
13...   dm [ compile (any pgZpart) .klangtyp  (any pfn)(any ~tail) ]
14...>     [ with [@@ _1 (INTEGER 0) .nwann]
15...>           [' if ['__0 ?> (d32 0.0)]
16...>                 ['' _0 .klangtyp [@@ _1 (integer 0) .wer .klangtyp]
17...>                     __0 _r3 ]
18...>           ] ;
19...>       apl _1 [@ .indxMin _1] to [pred [valid .max _1]]
20...>              ['
21...> show "k" ;
22...>                _0__0 ['@ _1['succ __1]] _r2 ] ;
23...>       _0_2 [@@ [@99 _1] .wer .klangtyp] [pfMaxDura] _r3 ;
24...> showcr ]
25...;
26...
27...   dm [ compile (any pgZevent)(any pgZevent) .klangtyp (any pfn)(any ~tail) ]
28...>     [ _0_3 [@@ _1 .wer .klangtyp] [[@ .nwann _2] - [@ .nwann _1]]  _r4 ]
29...;
30...; - ASSUME differenz IST POSITIV !!!
```

```
31...   dm [ compile .klangtyp (any pgKlangtyp) (any d32) (any pfn)(any ftok)
32...>                                                    (any pfn)(any ~tail) ]
33...>     [ _0_1_2_3_4_5 ; _0_1_2_3 _r6 ]
34...;
35...   dm [ compile .klangtyp (any pgKlangtyp) (any d32) (any pfn)(any ftok) ]
36...>     [ put _4  [pfformat _xx_  [@ _5_2](TRASA u16)]
37...>                static .dt [_3 ROUNDUP $0.0004] ]
38...;
39...; - ASSUME sind alles INTEGER in FVMY !!!
40...; ------------------------------------------------------------------------
41...
42...   dm [ compile  pfn (any pgZpart) .sPer ]
43...>     [ APL _2 ['@ .record __0 ['@@ __0 .WER .KLANGTYP ['@ .r __0]] ] ;
44...{                !!!!!!!!!!! hier evtl MEHR AUSWAHL !! ?? }
45...>       with [[@ .itemcnt _2] * [AmpSegPerZPart]]
46...>            [' _0_2_3  ['open['new pfn "bufstart" __0]]
47...>                       ['open['new pfn "buflen" __0]]
48...>             ]]
49...;
50...
51...   dm [ compile (any pgZpart) .sper (any pfn)(any pfn) ]
52...>     [ with [@@ _1 (INTEGER 0) .nwann]
53...>            [' if ['__0 ?> (d32 0.0)]
54...>                  ['' put _3 $0.0 static .dt __0 ;
55...>                      put _4 $0.0 static .dt __0
56...>                  ]] ;
57...>       _0_1_2 (integer 0) _r3 ]
58...;
59...   dm [ compile (any pgZpart) .sper (any integer)(any pfn)(any pfn) ]
60...>     [ _0_1_2_3 [@@ _1_3 .nwann] _r4 ]
61...;                                                  AB WANN !!
62...   dm [ compile (any pgZpart) .sper (any integer)(any d32)
63...>                                                  (any pfn)(any pfn) ]
64...>     [
65...> show "s" ;
66...>       if [_3 ?eq [valid .max _1]]
67...>          ['__< ~do _r1] ;
68...>       ifn [AND  [[@@ _1_3 .record] ?eq [@@ _1 [succ _3] .record]]
69...>                 [[@@ _1_3 .nsper]  ?eq [@@ _1 [succ _3] .nsper]]
70...>                 [[@@ _1_3 .pause]  ?eq (d32 $0.0)] ]
71...>           [' ~do _r1 ;
72...>              __< _0_1_2 [succ _3] _r5 ] ;
73...>       _0_1_2 [succ _3] _r4 ]
74...; - ASSUME all pfnwerte sind 32 Bit in FVMY !!!
```

```
75...
76...   dm [ ~do(any pgzpart).sper(any integer)(any d32)(any pfn)(any pfn) ]
77...>     [ _0_2 [@ _1_3] [  [[@@ _1_3 .nwann] + [@@ _1_3 .nsper]] - _4] _r5 ]
78...;
79...   dm [ ~do .sper (any pgzevent)(any d32)(any pfn)(any pfn) ]
80...>     [ put _4 [pfformat xxx_ [@@ _2 .record .wo ]]
81...>              static .dt _3 ;
82...>       put _5 $0000.00 static .dt [pg_playDropOut] ;
83...>       put _5 [pfformat xxx_ [@@ _2 .Nsper](trasa u32)]
84...>               static .dt [_3 -[pg_playDropOut]] ;
85...>       if [[@@ _2 .pause] ?> (d32 $0.0)]
86...>          ['put _4 $0.0 static .dt [@@ _2 .pause] ;
87...>            put _5 $0.0 static .dt [@@ _2 .pause] ]
88...>     ]
89...;
90...
91...{--
92...   dm [ ~do(any pgzpart).sper .pause (any d32) (any d32) (any pfn)(any pfn) ]
93...>     [ put _6 $0.0 static .dt [_5 - _4] ;
94...>       put _7 $0.0 static .dt [_5 - _4] ]
95...;
96...-------------}
```

Ende Quelltext

Kapitel 8

Weitergehende Betrachtungen – Bemerkungen zu Gehalt und Verwendung von Benutzersprachen.

8.1 Praktischer und ästhetischer Gehalt einer Benutzersprache.

Eine Benutzersprache in unserem Sinne ist allemal eine (kleine) Kompositionssprache. Als solche ist sie (1) ein Mittel der Kommunikation zwischen Mensch und Maschine, (2) bereits selbst eine komplexe (ästhetische, materialgenerierende) *Entscheidung*, da sie verschiedene Bahnen möglicher Materialverknüpfung (durch je unterschiedlich starkes Entgegenkommen) deutlich vorherbestimmt.

Besonders das Kriterium, wie weit die Formulierung eines klanglichen Vorganges explizit auf die Zeit (die einzelnen Zeitpunkte) bezug nimmt, wie also Diachrone Darstellung und out-of-time-Darstellung gewichtet sind, ist ein wichtiges Kriterium zur Unterscheidung der verschiedenen Komponsitionssprachen und Hinweis auf die Adäquatheit einer Sprache für ein gegebenes Problem und die Art, auf welche der benutzende Komponist dieses Problem ansonsten behandelt.

8.2 Gezielter Fehlgebrauch.

Gezielten Fehlgebrauch nennen wir ein kreatives Verhalten, welches in der antiken Technologie der Spannungsgesteuerten Klangsynthese vielfach eingesetzt wurde oder werden musste :

Übersteuerte Filter wurden als Signalgeneratoren eingesetzt, – aus LF-Sägezähnen und einstellbaren Schmitt-Triggern wurden Rhythmusgeneratoren, – Bandmaschinen dienten als Echo-Geräte, – sich im Ring triggernde Hüllkurvengeneratoren bildeten Sequenzer etc.

Allgemein ist Gezielter Fehlgebrauch das Verwenden eines Werkzeugs zu Zwecken (oder in Grenzbereichen), für die es eigentlich nicht ausgelegt ist, welches aber durchaus *anderweitig nicht zu erzielende* Ergebnisse zeitigt.

Obige Beispiele Gezielten Fehlgebrauchs aus der Praxis des Analog-Studios waren darüberhinaus keineswegs nur Notlösungen wegen der ärgerlichen Beschränktheit der damals nur zur Verfügung stehenden Technologie, sondern vielmehr sowohl form- und material-bildende kompositorische Maßnahmen, als auch höchlich geistbildend, da sie zwangen, sowohl über die Funktionsmöglichkeiten der beteiligten Geräte nachzudenken, als auch die Vorstellungen der intendierten Strukturen genauer zu hinterfragen und evtl. modifizierend konkreter einzugrenzen.

Ein Tonhöhen-Sequenzer z.B., der damals durch Überlagerung abgeschwächter Sub-Audio-Wellen „improvisiert" werden mußte, hat halt andere Verhaltensweisen (und eröffnet teilweise sogar *weitere* Gestaltungsmöglichkeiten) als ein modernes, leistungsfähiges Sequencer-Programm, – die dort unvermeidbaren leichten Schwebungen wirken form- und materialbildend.

Überraschender Weise kann auch bei der Verwendung digitaler Formulierungen auf Gezielten Fehlgebrauch nicht verzichtet werden:

Angenommen, der Benutzer von PGP möchte im Sinne des Direkten Durchgriffs entgegen seinem Grundkonzept (= „Schritte als Teile von Gängen") als *licenca* („ausnahmsweise") an einer bestimmten Stelle ein *einmaliges* Schallereignis vereinzelt erklingen lassen (sog. „Hinzugefügtes", oder „ajouté"), z.B. eine zuschlagende Tür. Dies ist im Kontext von PGP-0.2 möglich, indem ein „Gang" beschrieben wird, der zwangsläufig nur einen(1) „Schritt" enthält. Dies kann man z.B. durch Angabe einer Schrittweite erreichen, welche zwangsläufig nach einem Schritt den virtuellen Ort der `pgPerson` aus dem erlaubten Wertebereich führt und so den Gang beendet.

Weiterhin ist darauf zu achten, daß `.schritte/sekunde` so klein gewählt wird, daß die Dauer der verwendeten Schallaufzeichnungen immer in die Periode der imaginären Schritte paßt.

Zum dritten müssen wir, obwohl wir darauf achten, daß der (nicht stattfindende) Schritt immer aufwärts geht, auch für „`.absatz`" dieselben Werte eintragen, da wir ja an beliebiger Stelle (ohne auf Absatz oder Stufe achten zu wollen) unsere Tür schlagen lassen wollen !

Gegenbeispiel ist das `telefon`, also ein im Intervall von fünf(5) Sekunden repetierendes Geräusch, welches sich normalerweise nicht von der Stelle bewegt und solange fortdauert, bis sich einer seiner erbarmt.

Dies könnte modelliert werden durch (1) Wahl einer entsprechenden „Schritt"-Frequenz in Verbindung mit (2) einer Schrittweite von 0(null).

Quelltext 1

```
1....; lizenz: ausnahmsweise singulaere ereignisse
2....
3....  new pgPartitur "TuerUndTelefon"
4....
5....  new pgPerson "tuer"    .aufwaerts .schritte/sec       0.1
6....>                        .aufwaerts .stufen/schritt  1000.0
7....>                        .aufwaerts .abweichung         0
8....>                        .absatz    .schritte/sec       0.1
9....>                        .absatz    .stufen/schritt  1000.0
10...>                        .absatz    .abweichung         0
11...>                        .KlangTyp  Tuerschlagen_1
12...;
13...
14...
15...  new pgPerson "telefon" .aufwaerts .schritte/sec       0.2
16...>                        .aufwaerts .stufen/schritt     0.0
17...>                        .aufwaerts .abweichung         0
18...>                        .absatz    .schritte/sec       0.2
19...>                        .absatz    .stufen/schritt     0.0
20...>                        .absatz    .abweichung         0
21...>                        .KlangTyp  TelofonPiepen_1
22...;
23...
24...  ! 20.0  telefon .wo 70 .wohin 10000
25...  ! 35.0  tuer    .wo 20 .wohin 10000
26...  ! 45.0  tuer    .wo 70 .wohin 10000
27...  ! 50.0  telefon ENDE
28...
```

Ende Quelltext

8.2.1 Kritik.

Leider benötigt der Gezielte Fehlgebrauch häufig Informationen über das *interne* Verhalten, – bei den alten Analoggeräten wie bei modernen Algorithmen.

Diese werden heute wie damals i.A. durch Versuch und Irrtum gewonnen.

In obigem Beispiel „`tuer`"(Niedrige Schrittfrequenz = lange Ereignisdauer) ist z.B. wichtig, daß (1) die *Implementierung* der `PFN`-Generierung bzgl. der Ausdehnung des Klangmaterials *überlappende* Schritte möglicht (qua Unabhängigkeit der Generatorstimmen von den logischen `pgPerson`en), und daß (2) der Auswertungsalgorithmus das sofortige Ende des Ganges wg. der unmöglichen räumlichen Lage des „zweiten Schrittes" *sofort* erkennt. Darum wird die Stimme „`tuer`", gleich mit der Erzeugung des Schallereignisses, unverzüglich in den Zustand „unbenutzt" gesetzt

und kann jederzeit (sogar vor Ablauf des Klangmaterials!) wiederbenutzt werden, unabhängig davon, *wann* der erste und einzige „Schritt" normalerweise beendet wäre (– was für uns auch nur schwer auszurechnen ist) !

Aus der Sicht professionellen Programmierens ist der Gezielte Fehlgebrauch deshalb abzulehnen, weil er in diesen Fällen mit *nicht spezifizierten* Eigenschaften der Maschine operiert, also bei evtl. Reimplementierungen nicht mehr funktioniert, und auch nicht ohne Bezug auf die konkrete Implementierung dokumentierbar ist[1].

Spezifiziertes und implementiertes Systemverhalten stimmen halt nur in den allertrivialsten Fällen überein.

Die Semantiken von Implementierung und Spezifikation, also von Objekt und innerer Vorstellung, haben neben der gemeinsamen Schnittmenge durchaus *je eigene Bereiche* von Bestimmungen, die im je anderen *nicht enthalten* sind, wenngleich ihnen auch dort nicht widersprochen wird.

Nur im Bereich der gemeinsamen Schnittmenge darf streng genommen ein Programm überhaupt nur eingesetzt werden.

Wenn der Komponist aber bereit ist, auf Dokumentierbarkeit und Rekonstruierbarkeit zu verzichten[2], ist der Gezielte Fehlgebrauch allerdings ein wichtiges Verhalten im kreativem Umgang mit ästhetischem Material.

8.3 Konsistenz, Vollständigkeit und Fehlerbegriff.

Ein bekanntes grundlegendes Problem der Maschine-Mensch-Kommunikation besteht darin, daß die Maschine ausschließlich Inkonsistenzen in der ihr vom Menschen präsentierten Problembeschreibung erkennen kann, – also, daß zwei Anforderungen syntaktisch oder kontext-semantisch nicht gleichzeitig erfüllbar sind.

In allen nicht-trivialen Fällen jedoch kann sie normalerweise nicht erkennen, worin der Fehler besteht, d.h. welche von den beiden widersprechenden Angaben tatsächlich einem Eingabefehler entspringt.

Schlimmer noch: Der *Fehlerbegriff* selbst ist problematisch und (besonders in unserem Fall) stark abhängig vom *Gebrauchskontext*.

[1] Auch die o.e. Verwendung eines analog-elektronischen Filters als Signalgenerator funktioniert halt nur mit bestimmten Filter-Geräten eines bestimmten Herstellers und ist nicht mit „Filtern an sich" rekonstruierbar!

[2] Auch die *längerfristige Reproduzierbarkeit* von Verfahren ist beim Gezielten Fehlgebrauch wesentlich stärker gefährdet als bei Anwendungen, die einer Spezifikation folgen. (1) So schalten z.B. die neueren Telefunken-Bandmaschinen im Ggs. zu ihren Vorgängern den Wiedergabe-Verstärker aus („muting"), sobald im laufenden Betrieb die Wiedergabegeschwindigkeit umgestellt wird und der Capstan die neue Nennfrequenz noch nicht erreicht hat. Die seinerzeit möglichen hübschen Jaul-Effekte sind somit verunmöglicht.

(2) Das Telcom-Rauschunterdrückungs-System ist normalerweise je Kanal nur einmal vorhanden, kann also im Aufnahmebetrieb nicht auch die *Hinterband-Ausgabe* dekomprimieren. Deshalb kann eine Bandmaschine mit Telcom normalerweise nicht mehr für *Band-Echos* eingesetzt werden.

Der Fortschritt zu mehr Bedienungskomfort und Leistungsfähigkeit in den Standard-Anwendungen führt also zum Verlust an Möglichkeiten im Gezielten Fehlgebrauch.

Das Fehlererkennungsverhalten (der Fehlerbegriff) einer Implementierung gehört nämlich bei dem Typus unseres Vorgehens (die Entwicklung von Benutzersprachen als „minimalistische Sprächlein", die möglichst schnell konkretes, rezipierbares ästhetisches Material liefern sollen) normalerweise zu dem Bereich, welcher zwar Eigenschaft der Implementierung ist, aber vorher nicht ausgiebig diskutiert und spezifiziert wurde.

Die Konsistenzbedingungen der eingebbaren Datenstrukturen ist somit nur zum Teil enthalten im spezifizierten Systemverhaltens, zum anderen aber in den der vorbewußten Vorstellungen des intendierten „normalen" Umgangs mit dem Programm[3].

Konkret: Da der Student ursprünglich sich verständlicherweise nur vorgestellt hat, sinnvolle und nachvollziehbare Eingabetexte zu verfassen, so hat auch die (mehr oder weniger verbalisierte) Spezifikation unserer Benutzersprache keine Angaben über das Systemverhalten bei Sonderfällen wie ...

- `WEITER` ohne vorheriges `PAUSE`:
  die Realzeitsimulation stößt auf den Befehl „Ende der Pause", obwohl die entsprechende Stimme z.Zt. gar nicht pausiert.
- `PAUSE` ohne vorheriges `START`:
  die Realzeitsimulation stößt auf den Befehl „Pausiere den Gang", aber die entsprechende Stimme läuft z.Zt. gar nicht. Wie soll der Auswertungsalgorithmus sich dann verhalten, wenn *nach* dem `PAUSE`-Befehl ein `START`-Befehl folgt: Wird die Pause durch den Start-Befehl überschrieben oder gilt sie weiterhin (= sie wird „gemerkt", ist „sticky") und der Algorithmus wartet auf ein entsprechendes `WEITER`?
- Parameter-Modifikationsbefehle, während eine Stimme pausiert: Sollen diese (vorbereitend) ausgeführt werden, oder ignoriert, oder gar als Fehlerfall gemeldet werden ?
- Wie werden mehrere `UMKEHR`-Befehle zu gleicher Zeit evaluiert, – werden sie zu einem einzigen zusammengefaßt oder einzeln hintereinander ausgeführt ? Eine solche Situation kann durch algorithmische Textgenerierung (Zufallsauswahl ohne Wiederholungssperre) nämlich durchaus eintreten, z.B. in ...

  ```
  apl 1 to 2 [! bertha [@ random [' 12.0 15.0 22.3 25.3 ]] UMKEHR ]
  ```

- Was bedeutet ein `START`-Befehl für eine `pgPerson`, die bereits läuft / noch nicht beendet ist ?

  Dies scheint endlich ein Fall zu sein, der eindeutig als Falscheingabe zu behandeln ist. Sobald wir jedoch (weiter unten, 8.4.10) die regelbasierte Partiturgenerierung einführen, könnte ein Ignorieren eines solchen Befehls schon wieder die sinnvollere Reaktionsweise sein !

[3]Typischerweise erschwert das vorbewußte Wissen um den korrekten (gefahrlosen) Gebrauch eines Programmes dessen Entwickler sehr, eigene Fehler selbst zu finden. Man lasse seine Software immer vom unerfahrensten Anwender testen! Dem Autor z.B. gelingt es normalerweise, jeden macintosh mit wenigen Mausklicks „abzuschießen".

8.3.1 Gebrauchskontextabhängigkeit sinnvollen Fehlerverhaltens.

Die Entscheidung über das Systemverhalten in all diesen Fällen, also der konkrete *Fehlerbegriff*, ist in höchstem Maße abhängig vom Verwendungskontext:

Nennen wir eine Eingabe, die der Benutzer tätigt, die aber *nicht seinen Intentionen* entspricht, eine *Falscheingabe*. Die allermeisten Falscheingaben, die ein Rechner überhaupt nur erkennen kann, sind *Tippfehler*. Der Interpreter sollte selbstverständlich so viele Falscheingaben wie möglich erkennen und den Anwender warnen resp. die Eingabe zurückweisen. Falscheingaben sind aber meist abhängig vom Kontext:

Ein Wert von `0.0` oder `0.0001` für `.schritte/sekunde` oder für `.stufen/schritt` ist bei „normaler" Verwendung der Benutzersprache wahrscheinlich eine Falscheingabe, nicht jedoch bei o.e. Gezieltem Fehlgebrauch (`tuer` und `telefon`). Ähnliche Unterschiede gelten bei der Reihenfolge der Zeitangaben in der Befehlsfolge:

Falls der Benutzer nun tatsächlich eine PGP-0.2-Partitur im Sinne eines „Drehbuches", also streng diachron schreiben will, so wäre er für einen Fehlercheck sehr dankbar, da bei dieser Art der Verwendung eine Nicht-Einhaltung des monotonen Wachsens der Zeitpunkt-Angaben ja einen Tippfehler bedeutete.

Falls er allerdings „stimmenweise" polyphon schreibt, wäre eine zeitrichtige Anordnung nur im Bereich einer zusammenhängenden Stimme durchzuhalten, für die unsere Sprache allerdings überhaupt kein syntaktisches Konstrukt vorsieht.

Falls sein Quelltext parameterweise hingeschrieben oder als Ausgabe eines anderen Algorithmus automatisch generiert wird, ist die beliebige zeitliche Mischbarkeit der Befehle zwingende Voraussetzung für die freie Komponierbarkeit solcher Algorithmen.

Welcher Fehlerbegriff soll nun implementiert werden ?

Eine völlig liberale Reaktion kennt keine „ungewöhnliche Situationen" und findet für alle syntaktisch korrekten Kombinationen eine (mehr oder weniger sinnvolle) default-Semantik. Sie teilt allerdings ungewöhnliche Situationen, welche u.U. auf Fehleingaben zurückzuführen sind, dem Benutzer gar nicht mit.

Eine zu strenge Auslegung würde jedoch z.B. automatische Quelltextgenerierung erheblich verkomplizieren. Wird z.B. wie oben in 6.10.7 eine Folge von Befehlen mehrschichtig *unabhängig* von einander erzeugt, so ist es halt sehr praktisch (und wohl auch gemeint), daß dabei auftretende „stumme" Befehle (also solche ohne unmittelbare akustische Konsequenz, wie die überflüssigen `UMKEHR`-Befehle in unserem obigen Beispiel, welche bei herkömmlicher Schreibweise wohl Hinweis auf ein Schreibfehler wären) einfach vom Auswertungsalgorithmus ignoriert werden.

Da die wenigsten potentiellen Fehler rein syntaktisch sind, scheidet eine *Schichten-Lösung* (Praeprozessor wie lint) wohl leider aus.

Einzig eine Parametrisierung des Interpreters durch Angabe eines Fehler-Levels oder von zu erkennende Fehler-Gruppen könnte die erforderliche Anpassung an den Verwendungskontext tatsächlich realisieren.

8.4 Weiterentwicklung der Partiturformates.

8.4.1 Problematik von Erweiterungen.

Im Laufe der praktischen Arbeit mit unserer Benutzersprache entstehen ständig Ideen für Modifikationen, Verbesserungen und Erweiterungen.

Eine Grundidee der APOS/AUDIAC-Philosophie war es ja gerade, solch kleine interaktive Sprachen in wenigen Zeilen implementieren und ad hoc jederzeit anpassen zu können.

Dennoch ist der Wunsch nach Erweiterung immer kritisch zu betrachten:

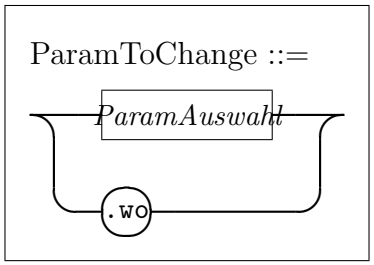
Jede Kompositionssprache ist ja bereits ästhetisches *Material*, da sie die Ausdrucksmittel vorstrukturiert (s.o.). Gerade durch die Bündelung des Möglichen und den Verzicht auf Beliebigkeit trägt sie zur Konsistenz und zum Gesicht des letztlich entstehenden Werkes bei[4]. Dieser *ästhetische Gehalt* wird nicht zuletzt durch das charakterisiert, was *nicht* formulierbar ist – wie in der bisher entwickelten Sprache z.B. Synchronisierungen zwischen verschiedenen Stimmen, Klangliche Veränderungen während eines Ganges etc.

Die evolutorisch entstehenden Wünsche nach Hinzufügung von ad hoc benötigten Ausdrucksformen sollten deshalb immer daraufhin hinterfragt werden, ob sie nicht zur Aufweichung des Grundkonzeptes beitragen, und ob sie nicht besser im Rahmen der Entwicklung eines *ganz neuen* Konzeptes von Anfang an berücksichtigt werden sollten.

Dennoch tauchten kleinere Erweiterungen auf, die schnell zu realisieren wären und in unterschiedlichen Zusammenhängen großen Nutzen brächten. Sie werden im folgenden lediglich vorgestellt; ihr Sinnzusammenhang geht entweder aus bereits Erwähntem hervor oder wird weiter unten diskutiert.

8.4.2 nice-to-have: Sprünge.

Die Parameter-Auswahl für die Modifikationsbefehle könnte zum Graphen ...



... erweitert werden.

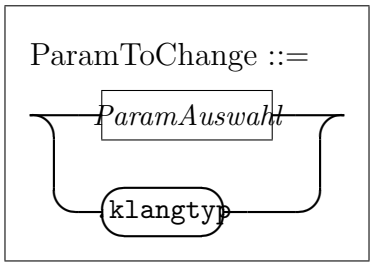
Der resultierende Modifikationsbefehl bedeutet ein unmittelbares Ändern der momentanen (virtuellen) Position einer `pgPerson`.

[4] „Lizenzen" sollten allerdings immer einfach anzuwenden sein. Gerade aber *wegen* der Strenge der Grundtextur sind sie von einem Rezipienten überhaupt erst *als solche* erkennbar.

Dies ermöglichte das „para-physikalische Phänomen der Translokation" darzustellen (s.u. 8.6.2): Eine gehende Person verschwindet plötzlich auf einer Geschoßhöhe, um auf einer anderen (mit völlig identischer Bewegungsrichtung, Bewegungsparameter etc., – also wiedererkennbar) aufzutauchen und ihren Gang, als wäre nichts geschehen, fortzusetzen.

8.4.3 nice-to-have: Schuhwerk-Wechsel.

Die Parameter-Auswahl für die Modifikationsbefehle könnte zum Graphen ...



... erweitert werden.

Der resultierende Modifikationsbefehl bedeutet ein unmittelbares Ändern des für die Person verwendeteten Klangmaterials.

Der praktische Zweck wäre, daß für o.e. singuläre ajouté- Konstruktionen (`tuer` und `telefon`) nicht mehr jeweils eine `pgPerson` pro Material aufwendig definiert werden müsste, sondern eine einzige `x`-Stimme jeweils anderes Material einmal zum klingen bringt :

Quelltext 2

```
1....
2....  new klangtyp "tuer"
3....  @ tuer     .aufwaerts (recording "zuschlagende_tuer)
4....>                                  from (0.0  sec) to (4.1 sec)
5....  @ hund     .aufwaerts (recording "koeter_anschlagend")
6....  @ teller   .aufwaerts (recording "porzellan_auf_fliesen_fallend")
7....>                                  from (0.0  sec) to (7.1 sec)
8....  apl [ tuer teller hund ] [@ .absatz _0 [@ .aufwaerts _0]]
9....;
10...
11...; ----------------------------------------
12...  new pgPartitur "TuerUndHund"
13...
14...  new pgPerson "x-voice" .aufwaerts .schritte/sec       0.1
15...>                        .aufwaerts .stufen/schritt  1000.0
16...>                        .aufwaerts .abweichung         0
17...>                        .absatz    .schritte/sec       0.1
18...>                        .absatz    .stufen/schritt  1000.0
19...>                        .absatz    .abweichung         0
20...>                        .KlangTyp  NULL
21...;
```

```
22...
23...  dm [ !  (any fixpoint) (any klangtyp) .wo (any integer) ]
24...>    [ !  _1 x-voice .klangtyp _2 ;
25...>      !  _1 x-voice START .wo _4 .wohin 10000 ]
26...
27...  ! 35.0  tuer    .wo 20
28...  ! 45.0  hund    .wo 50
29...  ! 45.01 tuer    .wo 70
30...  ! 60.0  teller  .wo 30
31...
```

Ende Quelltext

8.4.4 nice-to-have: Differenzierung von rechtem und linkem Fuß.

Eine alternierende Durchzählung aller Schritte je `pgPerson` (nebst entsprechender Spalte in der `ZPart` und Differenzierung der Klangmaterialen) ermöglichte, rechten und linken Fuß unterschiedlich zu realisieren.

Nur diese Erweiterung erlaubte es, einen „Mann mit Holzbein" oder – weniger makaber – mit Gipsfuß darzustellen.

Das Format unserer `pgZPart`-Zwischentabelle müsste zur Unterstützung der Auswahl zwischen den beiden den Füßen entsprechenden `recording`s um eine Spalte („`.isLeft`" o.ä.) erweitert werden[5].

8.4.5 nice-to-have: Zwei Arten von Absatz-Gehen.

Folgt man der Zielstellung einer möglichst realistischen Darstellung, so sollte es *zwei* Sätze von Bewegungsparametern für das Gehen auf dem Absatz geben: Je nachdem, ob die übergeordnete Bewegung aufwärts oder abwärts gerichtet ist, wird ein gehender Mensch sich auf einem Treppenabsatz leicht anders verhalten, – besonders dann, wenn Schrittfrequenz und -größe sich für beide Richtungen stark unterscheiden.

8.4.6 nice-to-have: Ortsangaben als Befehlsparameter.

Ein grundlegendes Manko unseres Entwurfes besteht wie erwähnt darin, daß einige Angaben zeitbasiert sind, andere aber raumorientiert, und daß die Relationen zwischen diesen Bestimmungen nur umständlich explizierbar sind.

In einigen vom Studenten gern gestalteten Abläufen stellte sich als wünschenswert, ja notwendig heraus, Grund- und Modifikationsbefehle (wie z.B. `UMKEHR`) nicht an einen Zeitpunkt gebunden in die Partitur einzutragen, sondern ihre Ausführung an das *Erreichen eines Ortes* zu binden.

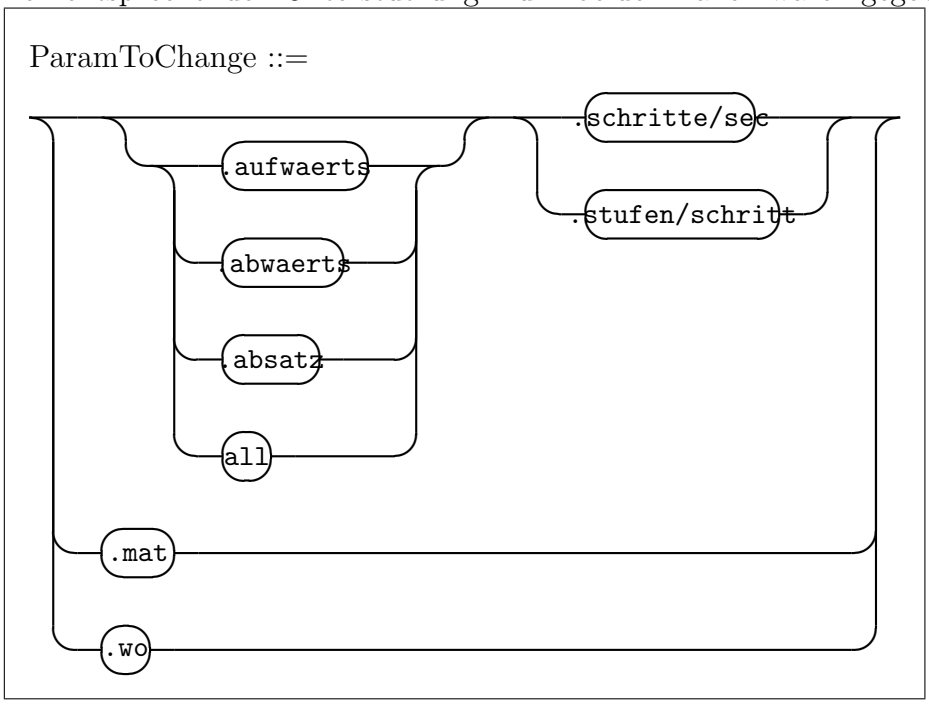
[5] Wird die rechts/links-Information sogar mit ld 3 Bit codiert, so könnte durch den Eintrag „`rechts_UND_links`" auch ein hüpfendes Kind (oder Kaninchen) realisiert werden!

Diese Forderung bedeutet aber einen grundlegenden *Paradigmenwechsel*: Das simple Modell einer linearen Simulation kann natürlicherweise *maximal einer* Dimension folgen. Die Realisierung einer zweiten Dimension kann nur nach dem Modell „Generator und Filter" hinzutreten. Die eigenständige Codierung des Wunsches nach positionsabhängigen Modifikationen wäre im Kosten/Nutzen-Sinne zu aufwendig. Vielmehr ist sie kanonisch im dem unten vorgestellten weitergehenden Ansatz einer *regelbasierten Partitur* enthalten (s.u. 8.4.10).

8.4.7 nice-to-have: Relative Parameter-Modifikation.

Ein *zentrales Defizit* des bisherigen Ansatzes ist, daß der Benutzer nicht (oder nur unter großem Rechenaufwand, welcher den automatischen Evaluierungslauf noch einmal zu Fuß nachvollziehen muß) auf den *momentanen* Zustand einer `pgPerson` zu einem gegebenen Zeitpunkt der simulierten Realzeit zugreifen kann: Wir wissen normalerweise nicht, ob eine `pgPerson` (zu einem gegebenen Zeitpunkt) auf einem Absatz oder einer Stufe geht, bei komplexeren Partituren noch nicht einmal, ob sie aufwärts oder abwärts geht.

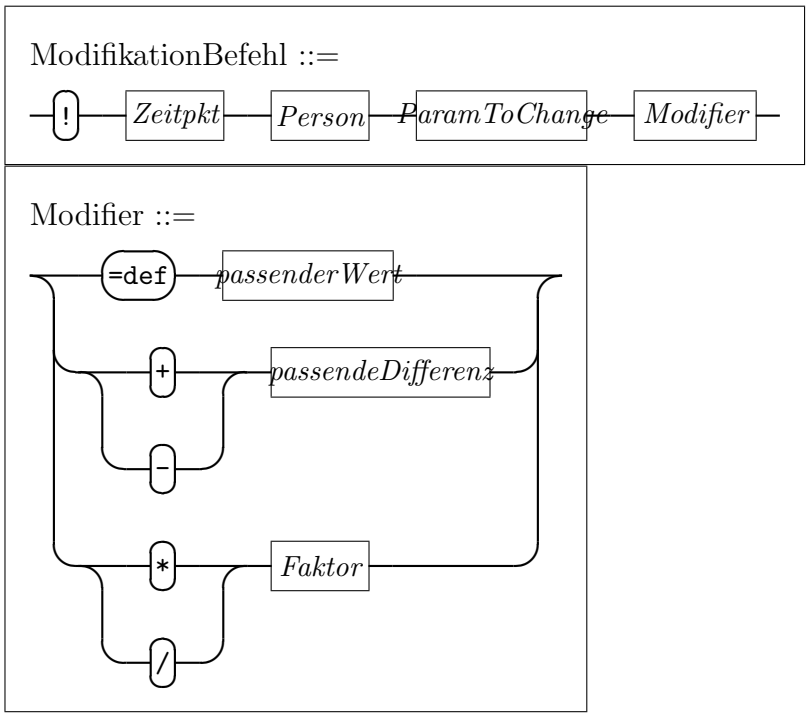
Unsere bisherigen Modifikationsbefehle sind folglich nur bedingt sinnvoll: Entweder man ändert jedesmal explizit den Parameter für alle drei Bewegungsrichtungen, oder man kann nicht sicher sein, daß die Änderung sich zum Zeitpunkt ihrer Ausführung überhaupt auswirkt. Eine andere Verbesserung wäre, eine Parametermodifikation auf die (uns allerdings unbekannte) *momentane* Bewegungsrichtung sich beziehen zu lassen[6]. Eine entsprechende Unterstützung für beide Fälle wäre gegeben durch ...



[6]Dies ist in der Tat in PGP-0.2 entgegen vorliegender Darstellung und fehlerhafterweise implementiert.

Die Verwendung von `all` bedeutet die Veränderung des angegebenen Parameters für *alle* Richtungen und den Absatz; das Weglassen des Praefixes bedeute eine Parameterveränderung für die *die Stufen* der *aktuellen* Bewegungsrichtung, – nicht den Absatz.

Beide Erweiterungen geben wahrscheinlich im bisherigen Zusammenhang, der eine *absolute* Angabe des neuen Wertes verlangt, wenig Sinn. Nötig sind vielmehr *relative* Modifikationen, so daß ein übergeordneter Syntaxgraph heißen könnte ...



Im folgenden Beispiel geschieht es gleichzeitig, daß `anton` die Schrittfrequenz aller Bewegungsrichtungen verdoppelt und `berta` beginnt, in der momentanen Richtung eine Stufe mehr je Schritt zu nehmen und plötzlich ganz gleichmäßig schreitet.

Quelltext 3

```
1....
2....  ...
3....  !  50.0  anton  all  .schritte/sekunde   * 2.0
4....  !  50.0  berta       .stufen/schritt     + 1
5....  !  50.0  berta  all  .abweichung      =def 0
6....  ...
7....
```

Ende Quelltext

8.4.8 Konzept-Erweiterung: Lesen von Werten zur Simulationszeit.

Eine *grundsätzliche*, sehr mächtige Erweiterung unseres Konzeptes ist dank APOS relativ günstig zu realisieren.

Als Hauptdefizite, welche verschiedenen Oberflächenproblemen zugrunde liegen, wurden erkannt (1) die mangelnde Korrelation zwischen Ort und Zeit und (2) die fehlende Information über den Zustand der virtuellen Personen zu einem gegebenen Zeitpunkt (der simulierten Realzeit).

Beides behebt die Einführung von lazy-evaluated-expressions als Bestandteil von `Start-` und Modifikations-Befehlen.

Wie unten gezeigt (8.5) können schon jetzt beliebige Ausdrücke in der Wirtssprache (hier: APOS) als Bestandteile unserer Befehle benutzt werden: an Stelle von festen Werten stehen Ausdrücke; das Resultat ihrer Auswertung ersetzt den festen Wert, z.B.:

```
! 10.0 anton  .schritte/sekunde (1.0 + (22.4 / 12.0))
```

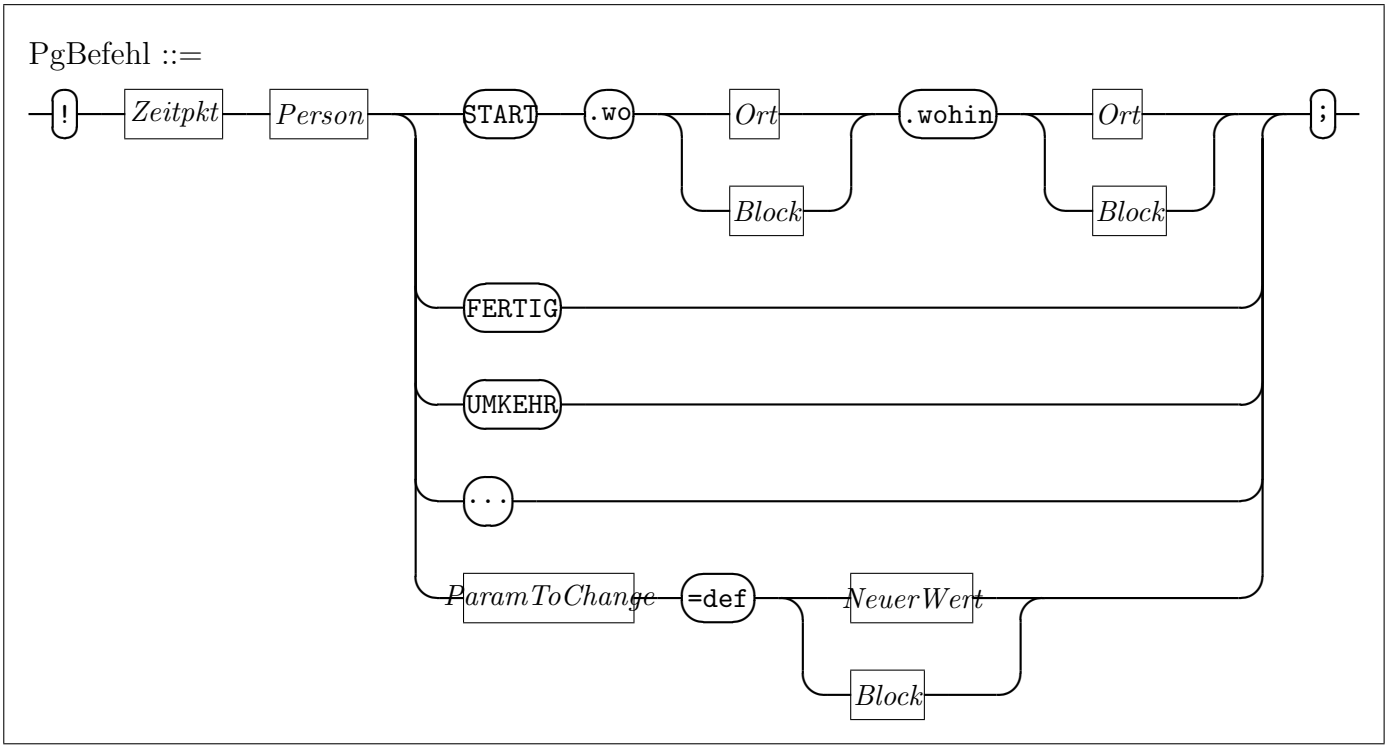
Wir erweitern unseren Interpreter nun dahingehend, daß er (1) *quotierte Ausdrücke* als Parameterwerte beim Eingeben der Partitur zuläßt, und (2) wenn er im Auswertungslauf, d.h. bei der Diachronen Simulation, auf einen solchen Ausdruck trifft, diesen *dann erst* auswertet, so daß der „aktuelle" Zustand aller `pgPerson`en (zu diesem simuliertem Zeitpunkt) in die Berechnung einfließen kann[7].

Dazu ist zusätzlich auf der Ebene der Spezifikation nötig anzugeben, wie Zustandswerte von `pgPerson`en *gelesen werden* können[8].

Da nicht ausgewertete Ausdrücke in APOS als sog. *Blockobjekte* realisiert werden, hieße die neue Version des ursprünglichen Syntaxgraphen :

[7] Dabei wird leider das *Typsystem* verlassen, da zur automatischen Typisierung von Funktionsobjekten (= „Blöcke") eine recht aufwendige abstrakte Evaluierung nötig wäre.

[8] Dies bedeutet bzgl. einer APOS-Implementierung, die Feldnamen anzugeben, welche die Werte enthalten. Das APOS-Konstrukt „ `@` *feldname Objekt* " liest den momentanen Wert eines Feldes eines APOS-Objektes. Das APOS-Methode „ `@ .posType (any pgPerson)` " liefert den Wert `.absatz`/`.aufwaerts`/`.abwaerts` für den nächsten Schritt.



Zunächst einmal ist die Erweiterung von eben (relative Wertveränderung) in dieser Erweiterung inbegriffen. Obiges Beispiel läßt sich schreiben :

Quelltext 4

```
1....
2....  apl [.aufwaerts .abwaerts .absatz]
3....>     [ ! 50.0  anton _0 .schritte/sekunde
4....>                       =def [' ['@ _0 .schritte/sekunde anton ]  * 2.0 ]
5....
6....  with [@ .posType berta]
7....>     [ ! 50.0  berta _0 .stufen/schritt
8....>                       =def [' ['@ _0 .stufen/schritt berta ]  + 1 ]
9....
10...  apl [.aufwaerts .abwaerts .absatz]
11...>     [ ! 50.0  berta _0 .abweichung =def 0 ]
12...
```

Ende Quelltext

Dies bedeutet im Vergleich zu oben offensichtlich keine Vereinfachung. Die grundlegend *neue Qualität* der Spracherweiterung zeigt sich bei andersartigen Beispielen :

Quelltext 5

```
1....  new token "gleichrum?"
2....  dm [gleichrum? (any pgPerson)(any pgPerson) ]
3....>    [[@ .richtung _1] ?eq [@ .richtung _2]]
4....
5....  new token "momentane"
6....  dm [momentane .schritte/sekunde (any pgPerson) ](
7....> dm [momentane .stufen/schritt   (any pgPerson) ](
8....> dm [momentane .abweichung       (any pgPerson) ]
9....>    [@ _1 [@ [@ .richtung _2] _2]]))
10...;
11...  ...
12...  ! 50.0 berta fertig
13...  ! 50.0 berta start .wo [@ .wo berta] .wohin [@ .wo anton]
14...
15...  ! 50.0 berta .schritte/sekunde =def
16...>          [ if [gleichrum? anton berta]
17...>               ['['momentane  .schritte/sekunde anton] * 1.2]
18...>               ['momentane  .schritte/sekunde berta]          ]
19...;
20...  ! 50.0 berta .stufen/schritt =def
21...>          [ if [gleichrum? anton berta]
22...>               ['['momentane  .stufen/schritt anton] * 1.2]
23...>               ['momentane  .stufen/schritt berta]          ]
24...; ... etc.
```

Ende Quelltext

Dies ist ein Beispiel einer „Nacheile"[9]: Berta vollzieht evtl. einen explizit formulierten `UMKEHR`-Befehl (= `FERTIG`-Befehl, unmittelbar gefolgt von einem neuen `START`-Befehl auf `anton`s Position zu, Quelltext 5 Zeile 12), um sich auf Anton zuzubewegen. Falls Anton sich von ihr entfernt, achtet sie darauf, schneller zu sein als er, um ihn einzuholen.

Für dieses Beispiel muß die *Semantik* erweitert werden um die Bestimmung, daß auch wenn Anton z.Zt. nicht aktiv geht, die gelesenen Parameter (`.wo`, `.richtung` etc.) einen verwendbaren Wert beinhalten, z.B. den letzen Zustand reflektieren.

Unsere Lösung ist allerdings nicht hinreichend : `berta` läuft ja nur auf `anton`s *momentanen* Standort zu, – dieser wird zu ihrem „Zielpunkt", so daß sie an diesem ihren Gang beenden wird, und dem `anton` auch nicht weiter nachlaufen.

Korrigieren wir dies, indem wir `berta` zu dem jeweiligen Ort-Maximum schicken, – je nachdem, in welcher Richtung `anton` sich befindet, – so wird sie einfach jenen *überholen*, – was auch nicht das ist, was wir wollten.

Befriedigende Modellierungen dieses Problems sind nur mit der unten eingeführten *Regelbasierung* zu erreichen (s. 8.4.10).

[9]Netter terminus technicus der Drogenfahndung im niederländisch-deutschem Grenzgebiet.

8.4.9 Weiterentwicklung des Materialbegriffes durch weitergehende Abstraktion $\delta \rightarrow \epsilon$: Szenarien.

Die Abstraktionsfolge $\alpha \rightarrow \beta \rightarrow \gamma \rightarrow \delta$ erzeugte einen wachsenden Materialbegriff: das durchgängige Stereosignal (α) wurde vertikal und horizontal *zerlegt* in Einzelschritt-Geräusche (β) ; diese wurden in Gänge von Personen *zusammengefaßt* (γ) ; diese wurden *gegliedert* durch die (auf jeweils *eine* Person bezogenenen) gangverändernden Verhaltens-Ereignissen (δ) , deren Abfolge unser „Drehbuch" ausmacht.

Eine sich natürlich anschließende *nächste Abstraktion* ist die Entwicklung von *Szenarien*[10]: Dabei werden *mehrere Personen* durch abstrakte Grundmuster sozialen Verhaltens *zusammengefaßt*: Sich Begegnen, sich Vermeiden, sich Nachfolgen, sich Treffen – all diese Situationsmuster können unabhängig von konkreter Klang- und Zeitgestalt als abstrakte Muster definiert und dann mit den unterschiedlichsten Materialien und Parametern instantiiert werden.

Diese „Szenarien" bilden eine neue Ebene von Material und entstehen durch die Abstraktion $\delta \rightarrow \epsilon$. Bild 8.2 zeigt einige typische Beispiele.

In vorliegendem Kontext (PGP-0.2 in APOS) sind sie durch Verwendung der Abstraktionsmechanismen der Wirtssprache (= APOS) häufig recht elegant zu realisieren, *ohne* daß die Benutzersprache und der Interpreter unbedingt erweitert werden muß[11].

Hier ein kleines Beispiel einer parametrisierbaren Szene, abstrahiert zu einer APOS-Methode :

Quelltext 6

```
1....
2....  new token "anpassen"
3....
4....  dm [ anpassen MIN (any zeitpunkt) (any pgperson)(any pgperson) ](
5....> dm [ anpassen MAX (any zeitpunkt) (any pgperson)(any pgperson) ]
6....>    [ _r0 .stufen/schritt ; _r0 .schritte/sekunde ; _r0 .abweichung ])
7....;
8....  dm [ anpassen(any token) (any zeitpunkt)
9....>                           (any pgperson)(any pgperson)(any ftok) ]
10...>    [ ! _2_3 ['_1 ['momentane _5_3]['momentane _5_4]] ;
11...>      ! _2_4 ['_1 ['momentane _5_3]['momentane _5_4]]
12...>    ]
13...;
14...  new token "treffen_gegenlauf"
15...
```

[10] Der Verfasser dankt Kollegen NEUHAUS, Essen, für die fruchtbare Diskussion, die zu der Entwicklung dieser Begrifflichkeit führte.

[11] Die oben eingeführten Erweiterungen sollten allerdings vorliegen und werden im folgenden teilweise benutzt, da ohne sie eine Formulierung wegen der o.e. expliziten zusätzlichen Nachvollziehens des Auswertungsvorganges (zwecks Bestimmung der „aktuellen" Position, Bewegungsrichtung etc.) doch recht umständlich wäre.

```
16...  dm [treffen_gegenlauf zeitpunkt#abWann
17...>     pgperson#A ort#A-Start fixpoint#A-SchrittGr fixpoint#A-SchrittFr
18...>     pgperson#B ort#B-Start fixpoint#B-SchrittGr fixpoint#B-SchrittFr
19...{lies:PERSON     STARTPUNKT  STUFEN/SCHRITT       SCHRITTE/SEKUNDE
20...     }
21...>              dauer#PausenDauer  ]
22...>    [ with [#abWann + [abs [ [#A-Start  - #B-start]
23...>                             /[  [#A-SchrittGr * #A-SchrittFr]
24...>                                +[#B-SchrittGr * #B-SchrittFr]
25...>                               ]] ]]
26...>           [' ! #abWann #A .stufen/schritt   =def #A-SchrittGr ;
27...>              ! #abWann #A .schritte/sekunde =def #A-SchrittFr ;
28...>              ! #abWann #A START .wo #A-Start .wohin #B-Start ;
29...>              ! #abWann #B .stufen/schritt   =def #B-SchrittGr ;
30...>              ! #abWann #B .schritte/sekunde =def #B-SchrittFr ;
31...>              ! #abWann #B START .wo #B-Start .wohin #A-Start ;
32...>              ! __0 #A FERTIG  ;
33...>              ! __0 #B PAUSE ;
34...>              ANPASSEN MIN __0 #A #B  ;
35...>              ! ['__0 + #pausenDauer] #A START ['' @ .wo  #A]
36...>                                              .wohin #A-Start ;
37...>              ! ['__0 + #pausenDauer] #B WEITER
38...>            ]]
39...; ------------------------------------------------------
40...   treffen_gegenlauf   12.0 anton  10  1.4   2.0
41...>                           berta  50  1.2   1.8   4.0
42...
43...   treffen_gegenlauf   20.0 conny  30  1.0   1.0
44...>                           det    70  2.0   2.2  10.0
45...;
```

Ende Quelltext

Anmerkungen:

- Weil die in APOS zur Anwendung kommende an die DE BRUIJN-Notation angelehnte Schreibweise von Parametern und lokalen Variablen zwar recht kompakt und schlackenlos, aber auch ziemlich schlecht lesbar ist, zeigt Bild 8.1 die Bezeichnungen der Parameter unseres „instantiierbaren Szenetyps".
- Quelltext 6 Zeile 19 ist eine Kommentarzeile, welche die Bedeutung der darüberstehenden Methoden-Parameter erklärt, was trotz der Neueinführung benannter Parameter in APOS leider recht häufig nötig ist. Das Steuerzeichen „geschweifte Klammer auf" in der Steuerspalte (erste Spalte des Textes) beginnt einen Kommentar, der (im Ggs. zum Semikolon) eine APOS-Nachricht nicht beendet. Diese wird vielmehr nach der schließenden Klammer fortgesetzt.

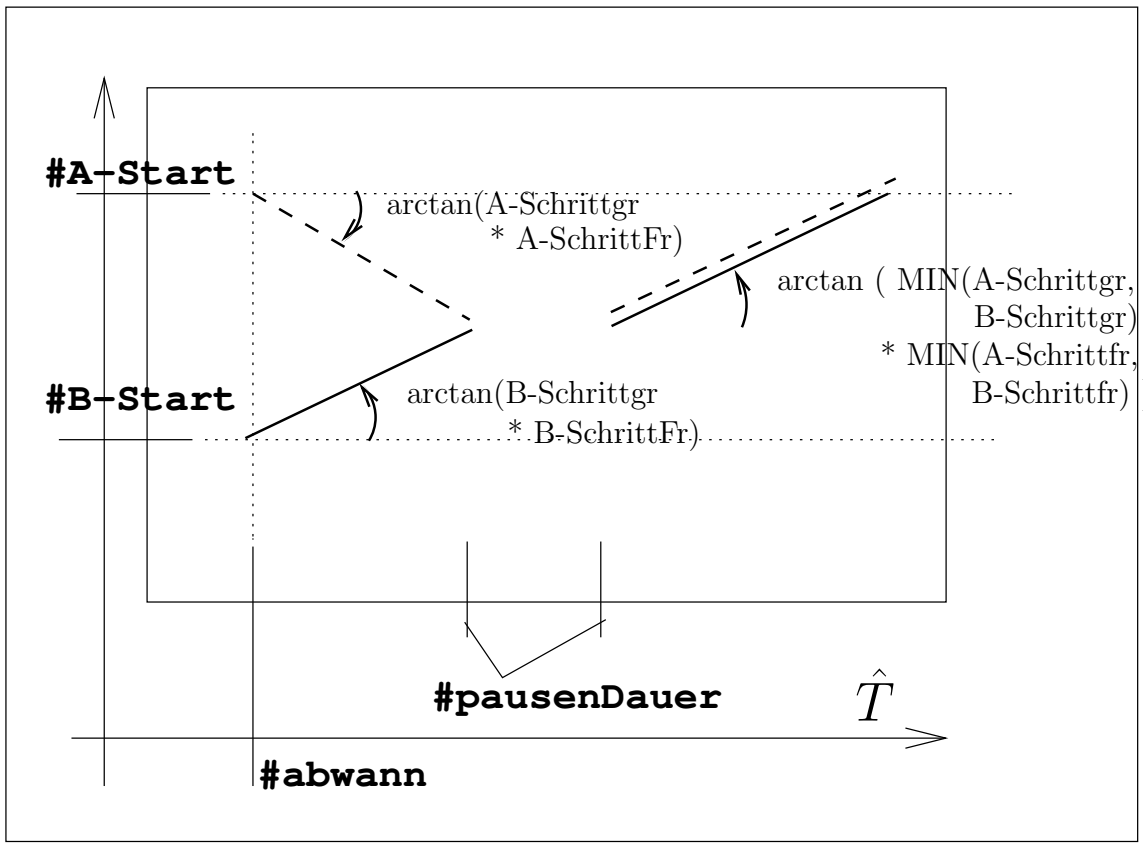


Abbildung 8.1: Parameter unserer Beispielszene.

- Für die `pgPerson_6` reicht in Quelltext 6 Zeile 32 nicht der `UMKEHR`-Befehl, da damit kein neuer Wert für `.wohin` gesetzt würde – diese Person soll ja mit Person_2 für den zweiten Teilgang einen gemeinsamen Endpunkt haben.

  Die angewandte Formulierung hat darüber hinaus zur Folge, daß beim Vorgang `ANPASSEN` die eben gehörten Parameter der ursprünglichen Bewegungsrichtung verwendet werden, und nicht die ungehörten der neuen Richtung.

- In Quelltext 6 Zeile 40 beginnt der Drehbuchtext für zwei sich räumlich und zeitlich überlappende „Begenungsszenen", welche ohne die Abstraktion qua Methode nur ziemlich umständlich und unter Vorüberlegungen mit Papier und Bleistift überhaupt formulierbar wären.

- Das oben eingeführte *Lesen der Parameterwerte zur Simulationzeit* (und damit eine einfachere Berechnung) kann leider nicht benutzt werden, da die *Zeitpunkte* der Befehle ja zur Partitur-Definitions-Zeit, also *vor* jedem Auswertungslauf bestimmt sein müssen.

  Diese Einschränkung ist grundlegend und schwerwiegend und führt (neben anderen) zu der Überlegung des folgenden Abschnittes.

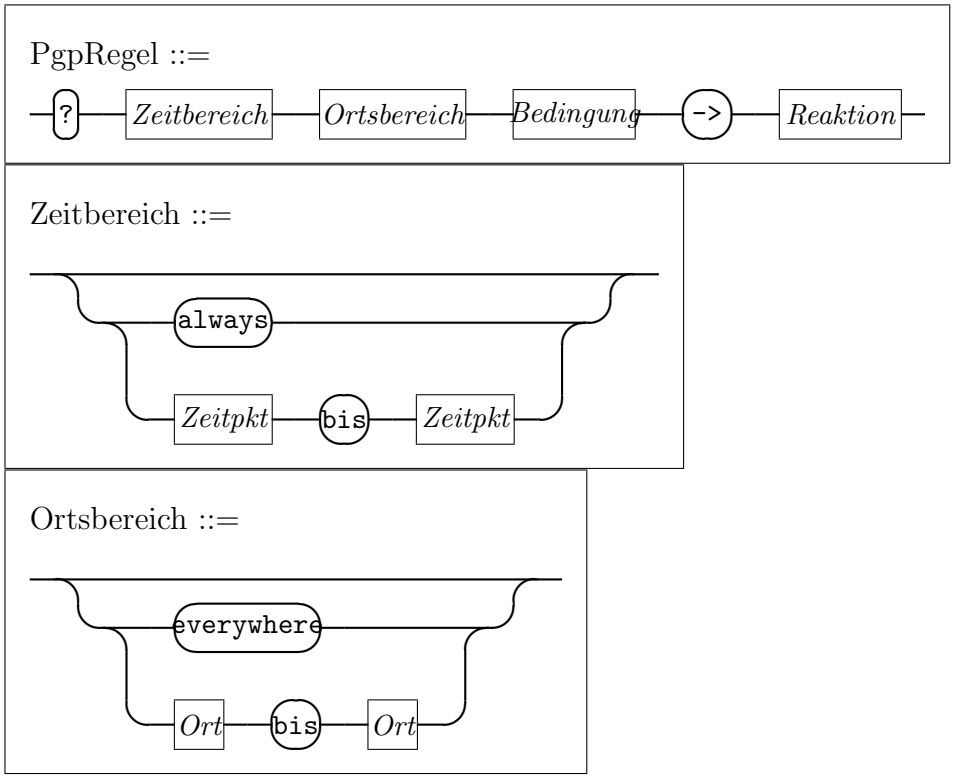
8.4.10 Abstraktion $\epsilon \rightarrow \zeta$: Erweiterung um Regelauswertung.

Mehrfach schon (siehe z.B. 8.4.2) wurde erwähnt, daß in bestimmten Fällen die Bindung eines Befehles an einen *Ort* statt einen Zeitpunkt wünschenswert wäre. Wie eben gesehen kann es auch sinnvoll sein, die Zeitpunkte von Grund- und Modifikationsbefehlen errechnen zu lassen, ausgehend von Zuständen, die natürlicherweise erst zur Simulationszeit (ohne den großen Aufwand einer Vor-Simulation) bekannt sind.

Aus alldem folgt die Idee, unser Partiturformat um das wesentlich flexiblere Instrument der *regelbasierten* Formulierung zu erweitern.

Die Erweiterungen beträfen Syntax, Semantik und Auswertung.

Die *Syntax* könnte erweitert werden um Konstrukte wie ...



Die *Semantik* stützt sich ab auf die zugrunde liegende Semantik der Funktionsobjekte der Wirtssprache (hier: der Blöcke in APOS):

Bedingung ist ein Funktionsausdruck nach Boolean. Falls dieser (im evtl. angegebenen Zeitbereich) nach „wahr" evaluiert, dann wird das *Reaktions*-Programm durchgeführt. Dieses wirkt sich aus durch *Seiteneffekte*, da es leicht modifizierte Befehle unserer ursprünglichen Drehbuch-Sprache enthält. Die Modifikation besteht darin, daß die Angabe eines Zeitpunktes normalerweise nicht auftritt, sondern sich die Befehle in der *Reaktion* implizit auf den Zeitpunkt des Wahr-Werdens der *Bedingung* beziehen.

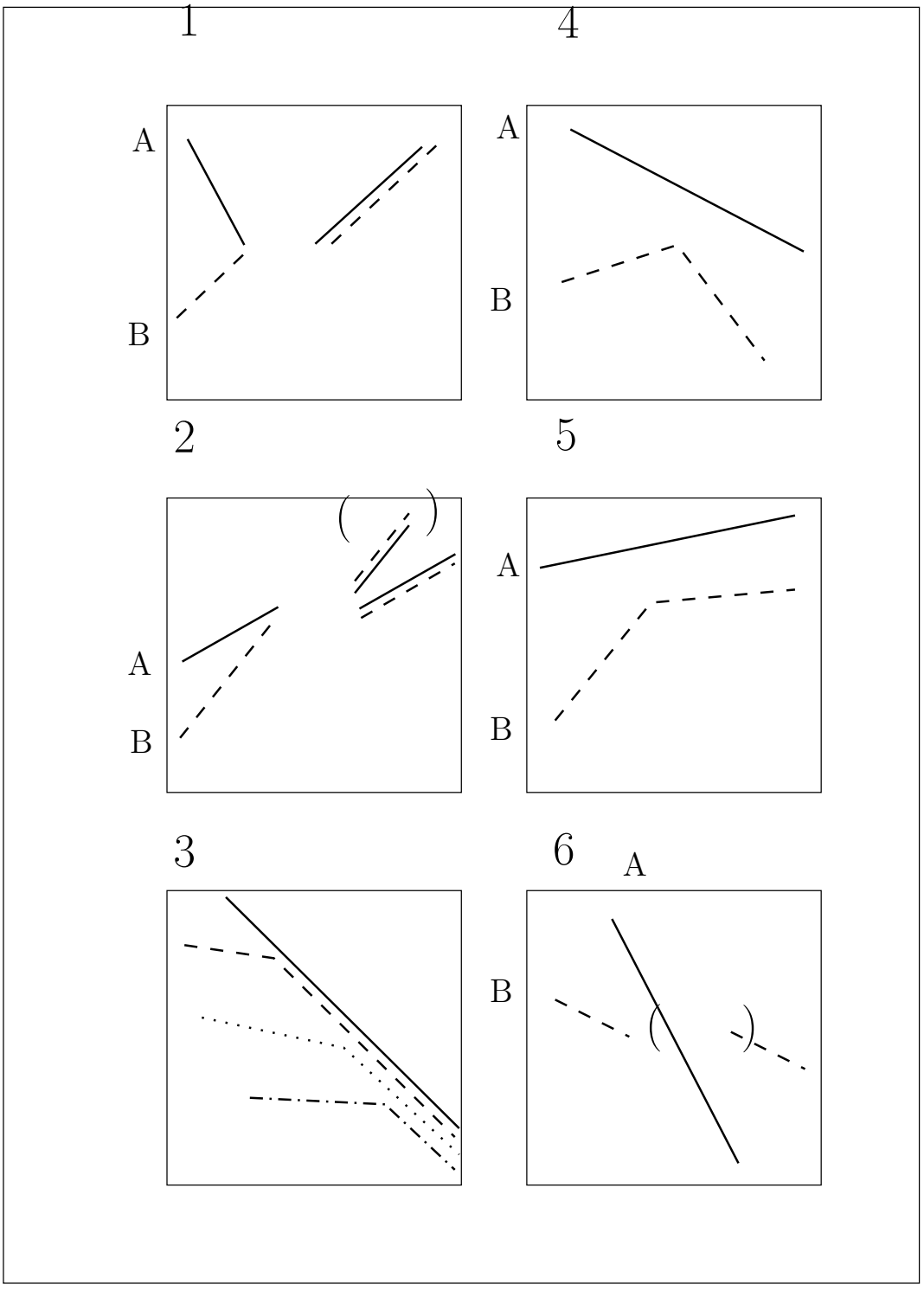


Abbildung 8.2: Beispiele von Szenen-Typen: Treffen und Meiden.

Die *Bedingung* kann explizit pgPersonen benennen und deren momentanen Parameterzustand abfragen. Sie kann aber auch Variablen enthalten (hier geschrieben als „_0", „_1" etc.). Solche Platzhalter werden der Reihe nach mit allen *momentan aktiven* pgPersonen belegt (bei Angabe einer Raumbegrenzung nur mit denen, die sich gerade in diesem Raum befinden). Falls für eine Belegung oder Belegungskombination die Bedingung wahr wird, kann dann im *Reaktionsblock* über die gleichen Platzhalter auf diese zugegriffen werden.

Die zu implementierende *Auswertung* fügt einfach zwischen die Schritte der (evtl. vorhandenen explizit an den Auswertungszeitpunkt gebundenen) Befehlsausführung und die Generierung der evtl. anstehenden Klanggenerierung als weiteren Schritt die Auswertung aller momentan aktiven Regeln ein (siehe Bild 7.1).

Noch *nicht geklärt* bei diesem Modell ist die Semanktik bezg. der Fragen, ob z.B. eine Regel nach einmaliger Erfüllung der Bedingung weiterhin für andere Stimmen geprüft werden soll, – ob diese Entscheidung evtl. vom Benutzer je Regel angegeben werden kann, – ob aus Performanzgründen auch eine Einschränkung auf Personen angegeben werden kann, – ob alle Bedingungen (innerhalb eines Prüfungsschrittes) sich auf denselben unveränderten Ausgangszustand beziehen, oder ob die Auswirkungen der Reaktionsfunktion einer ausgeführten Regel *unmittelbar* den zu prüfenden Zustand für die weiteren Regeln beeinflußt, oder gar bereits getestete und verworfenen Bedingungen nach jedem Reaktionsblock neu getestet werden sollen, bis das System sich (in diesem Auswertungszeitpunkt) stabilisiert.

Gerade die letzte Frage führt in ein weites Feld von Möglichkeiten und Gefahren (wie werden z.B. unendliche Zyklen vermieden?). In der professionellen Informatik gibt es zu diesen Typen von Systemen allerdings ausgefeilte Modelle und weitreichende Forschungsergebnisse, welche im Falle einer konkreten Spezifikation (und Implementierung) studiert, verwendet oder berücksichtigt werden können.

Illustrieren wir unsere Vorstellung durch Beispiele :

Quelltext 7

```
1....  new token "abstand" "langsamer-als" "schneller-als"  "anpassen-an"
2....
3....  dm [ abstand (any pgperson)(any pgperson) ]
4....>    [ ABS [[@ .ort _1] - [@ .ort _2]]  ]
5....
6....  dm [ (any pgperson)=def anpassen-an (any pgperson)(any fixpoint) MIN](
7....> dm [ (any pgperson)=def anpassen-an (any pgperson)(any fixpoint) MAX]
8....>    [ apl [' .schritte/sekunde .stufen/schritt]
9....>          [' __0_0_1 ['_5 ['['momentane __0 _3] * 4]
10...>                          ['momentane __0 _1]
11...>                    ]]])
12...;
13...  dm [ (any pgperson)=def langsamer-als (any pgperson) ]
14...>    [ _0_1  anpassen-an  _3 (fixpoint 0.8) MIN ]
15...  dm [ (any pgperson)=def schneller-als (any pgperson) ]
16...>    [ _0_1  anpassen-an  _3 (fixpoint 1.2) MAX ]
```

```
17...
18...; ------------------------------------------------------------
19...
20...
21...  ? always everywhere [[@ .ort anton]?eq [@ .ort berta]]
22...>            -> [.richtung berta =def [@ .richtung anton] ;
23...>                berta anpassen-an anton 1.0 min ;
24...>                berta anpassen-an anton 1.0 max ]
25...;
26...; ------------------------------------------------------------
27...
28...  ? always everywhere[   [[momentane .richtung anton]?eq .aufwaerts]
29...>                       AND[[[@ .ort anton] MOD 20] ?eq  2]     ]
30...>            -> [ anton PAUSE ;
31...>                 ! [[now] + [random 4.0 to 7.0]] anton WEITER
32...>               ]
33...;
34...; ------------------------------------------------------------
35...
36...  new token "momentane" "geschwindigkeit"
37...  dm [ momentane geschwindigkeit (any pgperson) ]
38...>    [ [_0 .schritte/sekunde _2] * [_0 .stufen/schritt _2] ]
39...;
40...  new token "gehtVoran"  "folgt"  "kommtEntgegen"
41...
42...; -- ASSUME
43...;     momentane .richtung liefert nur NICHT-NULL, wenn
44...;     die pgperson gerade aktiv ist !
45...
46...  dm [ (any pgperson) folgt (any pgperson) ]
47...>    [ OR  [ AND  [[momentane .richtung _0] ?eq .aufwaerts]
48...>                 [[momentane .richtung _2] ?eq .aufwaerts]
49...>                 [[@ .ort _0] ?< [@ .ort _2]]             ]
50...>          [ AND  [[momentane .richtung _0] ?eq .abwaerts]
51...>                 [[momentane .richtung _2] ?eq .abwaerts]
52...>                 [[@ .ort _0] ?> [@ .ort _2]]             ]
53...>    ]
54...;
55...  dm [ (any pgperson) gehtVoran (any pgperson) ]
56...>    [ _2 folgt _0 ]
57...;
```

```
58...dm [ (any pgperson) kommtEntgegen  (any pgperson) ]
59...>    [ OR  [ AND  [[momentane .richtung _0] ?eq .aufwaerts]
60...>                 [[momentane .richtung _2] ?eq .abwaerts]
61...>                 [[@ .ort _0] ?< [@ .ort _2]]             ]
62...>          [ AND  [[momentane .richtung _0] ?eq .abwaerts]
63...>                 [[momentane .richtung _2] ?eq .aufwaerts]
64...>                 [[@ .ort _0] ?> [@ .ort _2]]             ]
65...>    ]
66...;
67...
68...; ------------------------------------------------------------
69...
70...  ? always everywhere [     [_0 folgt conny]
71...>                         AND [[abstand conny _0 ] ?< 10 ] ]
72...>            -> [ conny =def schneller-als _0]
73...;
74...  ? always everywhere [     [_0 gehtvoran  conny]
75...>                         AND [[abstand conny _0 ] ?< 10 ] ]
76...>            -> [ conny =def langsamer-als _0]
77...;
78...  ? always everywhere [     [_0 kommtEntgegen  conny]
79...>                         AND [NOT [any pgperson ['__0 folgt conny]]]   ]
80...>            -> [ ! conny UMKEHR ]
81...;
82...  ? always everywhere [     [_0 kommtEntgegen  conny]
83...>                         AND [_1 folgt conny]             ]
84...>            -> [ ! conny PAUSE ; ! [[now] + 5.0] conny WEITER  ]
85...;
86...; ------------------------------------------------------------
87...
88...? always everywhere [ AND [NOT[_0 ?eq aufseher]]
89...>                           [[@ .richtung _0] ?eq [@ .richtung aufseher]] ]
90...>             -> [! trillerpfeife ;  hinzueilen aufseher _0]
91...
92...  ? always everywhere [AND [[abstand _0_1] ?< 5]
93...>                           [NOT[_0 ?eq aufseher]]
94...>                           [NOT[_1 ?eq aufseher]]      ]
95...>             -> [! trillerpfeife ;  hinzueilen aufseher _0]
96...
97...
98...; ------------------------------------------------------------
99...
100 . ;
```

Ende Quelltext

Die Einführung von Regeln in unsere Partitur ermöglicht eine *Weiterentwicklung des Materialbegriffes* durch weitergehende Abstraktion ($\epsilon \rightarrow \zeta$): Aus den konkreten „Szenen " von oben (= Materialebene ϵ), welche stets ähnliche Strukturen mit stets anderen Abmessungen ermöglichten, werden nun *Verhaltensweisen*, welche in unterschiedlichsten Kontexten sich stets anders auswirken (= Materialebene ζ).

Quelltext 7 Zeile 21 pp. zeigt eine derartige Verallgemeinerung einer Begegnungs-Szene (ähnlich wie Bild 8.2, 3): *Immer*, wenn Berta Anton begegnet, ändert sie ihre Bewegung und geht in gleichem Tempo mit ihm weiter. Semantisch ist also Anton entweder der Vorgesetzte und Berta ein Radfahrer, oder Berta empfindet für Anton eine zarte Regung und verhält sich dementsprechend ungeschickt.

Quelltext 7 Zeile 28 pp. modelliert, daß Anton alt und müde ist, und beim Aufsteigen *immer* alle zwei Stockwerke auf einem Absatz anhalten muß. Nach 4 bis 7 Sekunden geht er weiter. Dies ist ein Beispiel für die *Selbst-Modifikation* des Partiturobjektes durch Hinzufügung neuer, in der Zukunft liegender Befehle.

Die Methode „`[now]`" liefert immer den momentanen Zeitpunkt bei Auswertung der Reaktion.

Quelltext 7 Zeile 70 zeigt ein komplizierteres Verhalten: Conny ist sehr schüchtern und vermeidet jede Begegnung. Sobald Conny bemerkt, daß er (sie ?) näher als zehn Einheiten an einen Vorgänger herangekommen ist, senkt er seine Geschwindigkeit unter dessen; sobald ein Nachfolger näher herankommt, erhöht Conny seinen Schritt. Kommt ihm jemand entgegen, ohne daß ihm jemand folgt, so kehrt er um. Befindet er sich allerdings zwischen zwei auf ihn zu gehenden, so pausiert er (hinter einer Säule ?), um nach fünf Zeiteinheiten weiterzulaufen (falls nicht die Regel ein neuerliches Pausieren herbeiführt).

Der oben beschriebene Auswertungsmechanismus realisiert einen Existenzquantor, indem er für alle aktiven Personen alle Regeln testet, erlaubt aber nicht die automatische *Negation* des Existenzquantors. Deshalb gebrauchen wir explizit und zusätzlich die APOS-Methode „`[any pgPerson (any block)]`", die eine beliebige (momentan aktive!) `pgPerson` liefert, welche die durch den Block gegebene boolsche Bedingung erfüllt.

Quelltext 7 Zeile 88 modelliert einen Gefängnishof mit im Kreise schreitenden Häftlingen. Nur der „Aufseher" bewegt sich entgegengesetzt zur Hauptrichtung. Sobald ein anderer ebenfalls diese Richtung einschlägt, ertönt die Trillerpfeife (singuläres Ereignis, realisiert ähnlich wie oben im Beispiel 8.4.3) und der Aufseher eilt an den Ort des Fehlverhaltens. (Die Definitionen der nötigen Hilfsfunktionenen sind hier weggelassen.)

Dieselbe Reaktion geschieht, wenn zwei Personen (von denen keiner der Aufseher ist) sich zu nahe kommen (Quelltext 7 Zeile 92).

8.4.11 Erweiterung um Realzeit-Verarbeitung.

Der letzte Schritt der sich natürlicherweise anbietenden Folge von Erweiterungen ist die Hinzufügung eines dynamischen Realzeit-Verhaltens. Die AUDIAC-Architektur wurde ja gerade in Hinblick auf einen möglichst homogenes und unaufwendigen Übergang zwischen out-of-time und Realzeit entwickelt[12].

Im Rahmen des Konzeptes „Klangräume" beim „Spektakel '94" des ICEM (s.o. 6.2) wäre die Grundidee des Studenten (s.o. 6.3) am sinnvollsten als eine *interaktive Installation* zu realisieren gewesen :

1. Auf der Ebene des *Klangmaterials* ist eine signalverarbeitende Systemkonfiguration ohne Probleme denkbar, welche die Schritte (und anderen Geräusche) *des Besuchers* der Installation selbst in Realzeit aufnimmt und als Schallmaterial der von einer PGP-0.2-Partitur gesteuerten Klangstruktur verwendet.

   Man kann sich z.B. eine Zusammenschaltung von Endlos-Rekorder, Amplituden-Demodulator und Schmitt-Trigger vorstellen, wobei letzterer einen Kopiervorgang von erkannten Schrittgeräuschen aus dem Endlos-Puffer in einen (von mehreren) Material-Puffern steuert. Aus letzteren werden dann die `Player` für den PGP-0.2-Prozeß gespeist.

   Die Realisierung erfolgte im AUDIAC-Zusammenhang größtenteils auf der *Microcode-Ebene*.

2. Auf der Ebene der *Low-Frequency-Steuerung* gibt er ebenfalls eine natürliche Realzeitlösung : Die Bewegung des Besuchers[13] kann im einfachsten Falle den Start einer fertigen PGP-0.2-Paritur auslösen.

   In Verbindung mit dem eben eingeführten regelbasierten Partiturformat könnten diese besuchergenerierten Eingangsimpulse aber auch in die formulierten *Bedingungen* einer qua Regeln definierten dynamischen Partitur eingehen und so abhängig vom Hörerverhalten je andere Klangverläufe produzieren.

   Dieses Verfahren wurde von uns im Projekt THP0 angewandt und ebenfalls im Rahmen von „Spektakel '94" vorgestellt.

Die Einfachheit der Übertragung unserer Konzepte auf eine Realzeitzusammenhang resultiert nicht zuletzt durch die Anwendung der Verfahrens der *Synchronen Simulation* schon in der out-ot-time-Version: Der Übergang zu Realzeit besteht nämlich im Kern lediglich in der *schlichten Ersetzung* der Simulationszeit durch Realzeit, also in der Ersetzung der in jedem Schritt aufgerufenen Zeit-Fortschaltungs-Funktion durch eine an einen phsyikalischen Zeitgeber gebundene `waitDt`-Funktion (siehe Quelltext 8 Zeile 10 in Kapitel 7).

[12] Tatsächlich konnte der Student ja (nach dem verletzungsbedingten Ausfall des Verfassers) das Projekt ohne jeglichen Realzeitanteil und ohne den Komfort einer eigenen Sprache auch ohne das AUDIAC-System realisieren.

[13] ... detektiert entweder akustisch, durch die gleiche Schaltungskombination wie oben, oder durch zusätzliche Lichtschranken etc., welche qua MIDI leicht an das AUDIAC-System angeschlossen werden können, wie im Projekt TNP und THP0 geschehen.

0	Erstentwurf (= Bewegungsparameter nur aufwärts/abwärts, Diachrone Simulation noch in *festen* Zeitintervallen).
1	Zusätzlich Modellierung der Treppenabsätze in den Bewegungsparametern.
2	Modellierung von Aufwärts /Abwärts zusätzlich auch im Klangmaterial.
3	Ungleichförmiger statt gleichförmiger Zeitsimulation (Selbstgesteuertes Zeitraster) .
4	Abweichungsfaktor im Rhythmus hinzugefügt.
5	Erweiterung der Parameter-Modifikationsbefehle (`.klangtyp`, `.wo`).
6	Zur Simulationszeit auszuwertende Ausdrücke für Parameterwerte.
7	Genauere Modellierung der abstrahierten Wirklichkeit (Absatz aufwärts $\neq$ Absatz abwärts, anatomischer Unterschied von Rechts und Links).
8	Regelbasierte Partitur.
9	Realzeit (Klangmaterial und Regelauswertung).

Tabelle 8.1: Durchgeführte und denkbare Ausbaustufen von PGP.

8.4.12 Zusammenfassung.

Leider konnte von all diesen Erweiterungen wegen der schweren Erkrankung des Autors keine einzige mehr realisiert werden. Obwohl die unterschiedlichen Ideen wohl durchaus unterschiedlichen ästhetischen und praktischen Nutzen haben, geht aus ihnen u.E. doch beeindruckend hervor, welch *verschiedenartigste* (zu Beginn der Arbeit ungeahnte) Entwicklungsmöglichkeiten, ästhetischen Gundfragestellungen und technischen Probleme aus einer doch recht simplen Grundidee folgen können, wenn man diese konsequent weiterdenkt.

Tabelle 8.1 faßt noch einmal die stattgehabten und denkbaren Ausbaustufen der Sprache PGP zusammen.

8.5 Konjugierbarkeit von Moduln.

8.5.1 Politische und ästhetische Begründung der Notwendigkeit.

Die Bedeutung und Verwendung von Benutzersprachen sowohl im Rahmen des AUDIAC-Projektes als auch in jeder Integrierenden Arbeitsumgebung sollten keinesfalls eingeschränkt sein auf die Durchführung eines konkreten Projektes. Vielmehr sollten Sprachdefinitionen und -realisierungen, Erkenntnisse und Materialien, Datenstrukturen und -sammlungen (soweit sinnvoll) *dauerhafter Bestandteil* der so mit jedem Projekt wachsenden Integrierenden Digitalen Arbeitsumgebung werden.

Somit dient die Integrierende Digitale Arbeitsumgebung als *Kommunikationsplattform* zwischen Komponisten, wobei das Kommunikationsmedium nicht theoretische Darstellungen oder abgeschlossene Kompositionen (in ihrer Vordergrund-Gestalt) sind, sondern der „noch lebende", also der Verwendung und Fortentwicklung fähige *Mittelgrund* des dynamischen kompositorischen Materials.

Dies sollte nicht nur dem kompositionstheoretischen Austausch (auf der Meta-Ebene) dienen, sondern auch die konkrete Benutzung der Arbeitsergebnisse anderer, – wir sehen prinzipiell weder ein hohes moralisches Verdienst noch unmittelbaren ästhetischen Nutzen darin, das Rad zum x-ten Male zu erfinden[14].

Der „Integrierende Charakter" einer Arbeitsumgebung dient allen Beteiligten: Durch die Benutzung bereits in älteren Projekten realisierter Funktionen und Daten konzentriert sich eine Neuentwicklung auf ihre spezifische Aufgabe. So wurden z.B. in manchen der obigen Anwendungsbeispiele verschiedene Zufallsgeneratoren benutzt, deren Entwicklung mit der Sprache PGP-0.2 nun überhaupt nichts zu tun hat.

Ähnliches gilt jedenfalls für low-level Funktionalitäten wie Visualisierung der entstehenden Ereignislisten, graphische Editoren von Datenstrukturen, Verwendung von Frequenztabellen und genormter Lautstärkekennlinien, Parsergeneratoren etc., aber auch von höher angesiedelten Hilfsmoduln wie Permutationsgeneratoren, Harmonik-Algebren, Stimmungssystemen, rhythmischen und metrischen Konzepten etc.

Wir behaupteten (5.4), daß die Arbeit auf einer derartig offenen Plattform, welche den Austausch zwischen verschiedenartigsten Projekten möglichst begünstigt, *nutzenbringend* ist sowohl für die konkrete Arbeit am Einzelprojekt, als auch für den theoretischen Diskurs der Meta-Ebene.

8.5.2 Technologische Aspekte der Modul-Konjugation.

Auf der *technologischen Ebene* entspricht diesen Absichten der Vorgang der *Komposition von Moduln / Sprachen.* Die entsprechende Eigenschaft heißt „Komponierbarkeit".

Diese Nomenklatur wäre in dieser Abhandlung wohl mißverständlich[15].

Reden wir also hier von z.B. „Konjugieren" statt „Komponieren im Sinne der Rechnertechnologie".

Bild 8.3 zeigt eine mögliche Einteilung in verschiedene *Typen von Kommunikationswege* zwischen den beteiligten Subsystemen, welche die folgenden Beispiele und Bemerkungen zu den technischen Implikationen gliedert.

[14] Wobei es aber durchaus didaktische Gründe oder ästhetisch-strukturelle Feinheiten geben kann, die in bestimmten Fällen eindeutig *gegen* eine Wiederverwendung vorhandener Materialien sprechen können!

[15] Die mehrfache Belegung gleicher Namen mit verschiedenen Begriffen aus unterschiedlichen Fachgebieten ist ein häufiges Problem bei allen interdisziplinären oder synästhetischen Texten. Am saubersten wäre noch eine indizierte Schreibweise wie „$\mathsf{Komponieren}_{\mathsf{Inf}}$, $\mathsf{Komponieren}_{\mathsf{Mat}}$, $\mathsf{Komponieren}_{\mathsf{Mus}}$".

8.5.3 Typ α: Benutzersprache verwendet Objektbegriff der Infra-Sprache.

Eine *erste* wichtige und nützliche Eigenschaft, welche die Implementierung eine musikalischer Software in einer *symbolisch orientierten (Interpreter-)Sprache* mit sich bringt, und welche z.B. den verschiedenen in LISP oder Smalltalk realisierten Projekten wie auch vorliegender APOS-Implementierung zugute kommt, besteht darin, daß die Benutzersprache den grundlegenden *Objektbegriff* der Infrasprache erbt und die grundlegenden Verfahren zur Objektgenerierung, Lexikalik-Verwaltung, Visualisierung und Editierung etc. nicht neu erfinden muß.

Als besonders nützlich stellt sich meist heraus, daß *textuelle Eingabe* von diesen Sprachsystemen automatisch in Referenzen auf die interne Objektwelt übersetzt werden. Die Syntax- und Grammatikdefinitionen der zu realisierenden Benutzersprache setzen also nicht auf der unteren Ebene von terminalen Textzeichen auf, sondern auf Folgen oder gar Bäumen von *Objektreferenzen*.

Dies hat besonders in Verbindung mit „rapid protoyping" normalerweise nur Vorteile[16], den (1) sind diese Objektreferenzen schon vom System auf Korrektheit (im Sinne z.B. von lexikalischer oder numerischer Definiertheit) überprüft, und (2) kann die Verwaltung von Objektgenerierung und Sichtbarkeitsbereichen dem Infra-System delegiert werden.

Technische Voraussetzung für den letzten Punkt ist allerdings, daß der Äußere Interpretierer, die Lexem-Regeln, die Sichtbarkeitslogik, etc. *rekursiv* aus der Evaluierung von Sprachausdrücken heraus *hinreichend kontrollierbar und modifizierbar* sind, – „unter Programmkontrolle" stehen.

[16] Wenngleich auch mit weiterer Sprachverfeinerung oft der Wunsch nach stärkerer Beeinflußbarkeit des terminales Schriftbildes aufkommt.

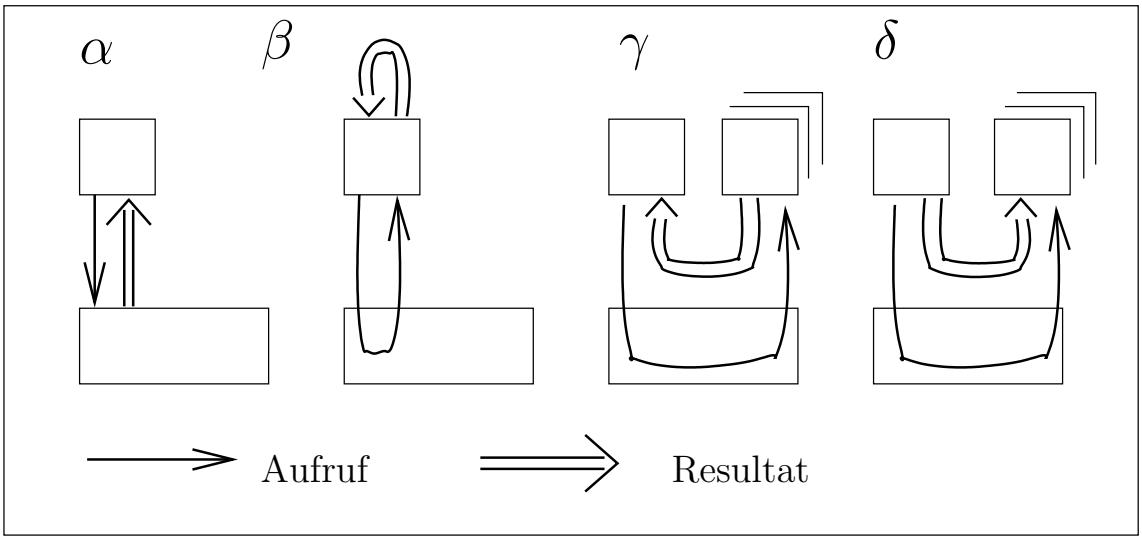


Abbildung 8.3: Mögliche Klasseneinteilung von Modul-Komposition.

8.5.4 Typ α: Benutzersprache verwendet Basis-Algebren der Infra-Sprache.

Die Bestandteile von Eingabesätzen in der Benutzersprache sind also allemal Objektreferenzen, welche in allen symbolorientierten Sprachen nicht nur durch lexikalische Übersetzung eines Eingabetextes, sondern auch durch Evaluierung von Ausdrücken generiert werden können.

Daraus folgt (natürlicherweise und ganz ohne zusätzlichen Aufwand!), daß generell *alle Komponenten* von Eingabesätzen der Benutzersprache als *Ausdrücke in der Infra-Sprache* oder sogar als Ergebnisse ganzer Programmaufrufe angegeben werden können.

Technische Voraussetzung dafür ist allerdings, daß die *Datentypen* der zu berechnenden Komponenten in denen der Basis-Moduln der Infra-Sprache enthalten sind. *Beispiele* findet man im vorigen Kapitel (z.B. Quelltext 6 Zeile 7).

8.5.5 Typ β: Benutzersprache verwendet Kontrollstrukturen der Infra-Sprache.

Auch der umgekehrte Fall ist sehr zweckmäßig: Durch Benutzung der Kontrollstrukturen der Infra-Sprache werden Ausdrücke der Benutzersprache in die Auswertung eines Ausdruckes der Infra-Sprache eingebettet; die Infra-Sprache ruft quasi die Benutzersprache auf.

Dies ist allemal sinnvoll, wenn wie bei PGP-0.2 die Auswertung von Benutzersprache-Ausdrücken Seiteneffekte hat, wie hier der Aufbau der Kommando-Datenstruktur. Somit sind durch Verwendung der Kontrollstrukturen der Infra-Sprache bequeme „Automatisierungen" von Eingabesequenzen unaufwendig möglich, – von simpler „Makro-Programmierung" bis hin zu intelligenten „Vorschalt-Algorithmen".

Technische Voraussetzung dafür ist allerdings, daß die Infra-Sprache Funktionen und Funktionen höherer Ordnung auf der Objektebene als first-class-resident behandelt.

Die Verwendung des grundlegenden Applikationsmechanismus in APOS (die „`with`"-Nachricht, welche ein Blockobjekt als Anfrage auswertet und dabei formale Parameter durch die angegebenen aktuellen Objektreferenzen ersetzt) kann im einfachsten Falle benutzt werden, um mehrfach auftretende Konstanten konsistent einzusetzen:

Quelltext 8

```
1....    with  10.53  20.05  22.777 [
2....>     ! _0 berta start .wo  10 .wohin 100 ;
3....>     ! _0 anton start .wo 100 .wohin  10 ;
4....>     !          _1 berta umkehr ;
5....>     !                _2 anton umkehr ;
6....>     !                _2 berta stop    ]
```

Ende Quelltext

Dies läßt sich im Sinne einer „Szene" (s.o. 8.4.9) weiter zu einer Methodendefinition abstrahieren, um dann mehrfach instantiiert zu werden :

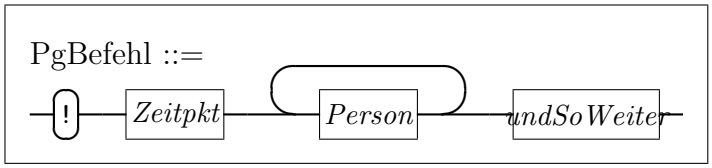
Quelltext 9

```
1....  new token "keineBegegnung"
2....
3....  dm [ keineBegegnung (any TimeStamp)(any TimeStamp)(any TimeStamp) ]
4....>    [ ! _1 berta start .wo  10 .wohin 100 ;
5....>      ! _1 anton start .wo 100 .wohin  10 ;
6....>      !          _2 berta umkehr ;
7....>      !                _3 anton umkehr ;
8....>      !                _3 berta stop    ]
9....
10...   keineBegegnung 10.53  20.05  22.777
11...   with 30.8
12...>      [ keine Begegnung _0 [_0 + 12.0] [_0 + 22.0] ]
13...;
```

Ende Quelltext

Die folgenden Beispiele zeigen *externe Spracherweiterungen*, also neue Konstrukte, welche ohne Eingriff in die Definition und Auswertung einer Benutzersprache durch vorgeschaltete Algorithmen die Ausdrucksmittel des Benutzers erweitern.

Um z.B. einen gleichzeitigen und identischen Befehl abzusetzen, der für mehrere Personen gelten soll, wäre für Quelltext 10 Zeile 3 ff. eine Syntaxerweiterung nötig :



Die Schreibweisen in Quelltext 10 Zeile 10 ff. und Quelltext 10 Zeile 18 ff. sind hingegen unmittelbar mögliche externe Spracherweiterungen durch Verwendung der Kontrollstrukturen (hi-order-funktionen) der Infra-Sprache APOS:

Quelltext 10

```
; ohne Spracherweiterung von pgp nicht moeglich :

   ! 10.5 anton berta umkehr
   ! 20.7 anton berta fertig

; ---------------------------

; dank apos sofort moeglich :

    apl [anton berta] \
        [  ! 10.5 _0 umkehr \
           ! 20.7 _0 fertig ]
;
; ---------------------------

; dank apos ist folgende "externe spracherweiterung" moeglich :

    dm [! (any timestamp) (any Block) (any ~tail) ] \
       [ apl _2 [' _0_1 __0 _r3] ]

; nun ist  moeglich :
   ! 10.5  [anton berta]   umkehr
   ! 20.7  [anton berta]   fertig
```

Ende Quelltext

Eine weitere höchst sinnvolle Erweiterung unserer Benutzersprache, ebenfalls *extern* realisierbar, also außerhalb der Sprachimplementierung, ist ein Befehlsformat, welches einen Gang von a nach b in einer *gegebenen Zeitdauer* (bei gegebener Schrittgröße oder Schrittfrequenz) beschreibt:

Quelltext 11

```
  dm [ ! (any zeitp) (any pgPerson) .wo (any ort) .wohin (any ort)
>                          in (any sekunden) .stufen/schritt (any fixpoint) ]
>     [  ! _1_2 all .stufen/schritt      =def _a ;
>        ! _1_2 all .schritte/sekunde    =def  [ [_6 - _4] / [ _8 * _a ] ] ;
>        ! _1_2 start _3_4_5_6 ]
;
  dm [ ! (any zeitp) (any pgPerson) .wo (any ort) .wohin (any ort)
>                          in (any sekunden) .schritte/sekunde (any fixpoint) ]
>     [  ! _1_2 all .schritte/sekunde =def _a ;
>        ! _1_2 all .stufen/schritt   =def  [ [_6 - _4] / [ _8 * _a ] ] ;
>        ! _1_2 start _3_4_5_6 ]
;
```

Ende Quelltext

Typischerweise würde man in APOS eine Stufe weiter abstrahieren, wodurch der Schreibaufwand vermindert, der Leseaufwand aber (zugegeben !) erhöht wird:

Quelltext 12

```
1....   dm [ ! (any zeitp) (any pgPerson) .wo (any ort) .wohin (any ort)
2....>          in (any sekunden) .stufen/schritt (any fixpoint) ]
3....>     [ _r0  .schritte/sekunde ]
4....;
5....   dm [ ! (any zeitp) (any pgPerson) .wo (any ort) .wohin (any ort)
6....>          in (any sekunden) .schritte/sekunde (any fixpoint) ]
7....>     [ _r0  .stufen/schritt ]
8....;
9....   dm [ ! (any zeitp) (any pgPerson) .wo (any ort) .wohin (any ort)
10...>                        in (any sekunden)(any ftok)(any fixpoint)(any ftok) ]
11...>     [  ! _1_2 all _9  =def _a ;
12...>        ! _1_2 all _b  =def  [ [_6 - _4] / [ _8 * _a ] ] ;
13...>        ! _1_2 start _3_4_5_6 ]
14...;
```

Ende Quelltext

8.5.6 Konjugationstyp γ: Unsere Benutzersprache verwendet andere Benutzersprache.

Eine *weitere* wichtige und nützliche Eigenschaft, welche die Realisierung musikalischer Software in einer *lernfähigen* symbolisch orientierten (Interpreter-)Sprache mit sich bringt, besteht darin, daß sie als *Bibliotheksmodul* eine integrierten Bestandteil eines neuen „Datenbank-Zustandes" bilden kann, und so systemweit zur Verfügung und Verwendung gestellt wird.

Verschieden Benutzersprachen und Synthese-Moduln können so auf unterschiedlicher Komplexitätsebene kooperieren. Ein Modul kann ein anderes quasi intern (als normale Bibliothek) aufrufen und sein Geladensein voraussetzen, – es kann aber auch der Benutzer zwei Moduln, die nicht unbedingt zusammengehören und die in verschiedenen Kontexten entstanden sind, in einem dritten konjugieren.

Allen Fällen ist gemein, daß die Kommunikation zwischen den Moduln *über* Datenstrukturen (incl. syntaktischer Konstrukte) der Infra-Sprache geschieht!

Technische Voraussetzungen sind also, daß (1) die Semantiken der zu konjugierenden Moduln auf die (allen bekannte) Semantik der Objekte und Basismoduln der Infra-Sprache abbildbar sind, (2) daß die Infra-Sprache auch die benötigten Kommunikationsmittel höherer Ordnung bereitstellt, und natürlich (3), daß die Schnittstellen der Moduln ausreichend dokumentiert sind, – was keinesfalls ein triviales Problem darstellt.

Im Falle unserer Benutzersprache PGP-0.2 ist der einfachste Fall, daß ein mit anderen Moduln (oder „Programmen", wie Klangsynthesesprachen oder Sample-

Editoren, etc.) generiertes Programm als Schallmaterial in unseren Wiedergabe-Kontext eingespeist wird.

Die Struktur der übermittelten Information ist dabei höchst einfach, nämlich eine (Mono-)Amplituden-Funktion über der Zeit.

Aber auch kompliziertere Datenstrukturen kann PGP-0.2 aus anderen Moduln übernehmen, wie in folgendem durch den Aufruf eines Zufallsgenerators (siehe auch oben Quelltext 7 Zeile 31).

Quelltext 13

```
1....  ! 10.5   (@ random [anton berta conny]) umkehr
2....  ! 20.5   (@ random [anton berta conny]) umkehr
3....  ! 22.5   (@ random [anton berta conny]) umkehr
4....  ! 27.5   (@ random [anton berta conny]) umkehr
```

Ende Quelltext

... oder, kombiniert mit den Kontrollstrukturen der Infra-Sprache :

Quelltext 14

```
1....  apl [ 10.5 20.5 22.5 27.5 ]
2....>     [ ! _0 [@ random ['anton berta conny]] umkehr ]
3....;
4....  splice apl [              [ 50.6 70.0  80.44 ]
5....>              (permutation [anton conny berta])  ]
6....>            [ ! _0_1  fertig ]
```

Ende Quelltext

8.5.7 Konjugations-Typ δ: Die Sprache PGP-0.2 als „Backend" – automatisch Quelltext generieren.

Den oben als mögliche Infra-Sprache aufgezählten (existierenden) Sprachen ist ein bestimmtes *Reflexionsprinzip* gemeinsam, daß nämlich „*Sätze der Sprache*" (über dem terminalen Alphabet der Objektreferenzen) kanonisch überführbar in Datenobjekte sind, et vice versa.

Dies hat den Effekt, daß Sätze einer Benutzersprache auch als Ergebnis der Auswertung eines Terms einer ganz anderen Sprache sein können, – das Ausgabedatum eines Algorithmus x kann den Datentyp „gültige PGP-0.2-Partitur" haben.

Da dies durchaus eine andere Qualität als der oben unter γ beschriebene „gleichberechtigte" Austausch "passiver" Datenstrukturen ist, geben wir dieser Art von Konjugation die eigene Bezeichnung δ.

Obwohl dies im Falle unseres PGP-0.2 zugegebenermaßen nicht sehr anschaulich ist, kann man sich doch z.B. ein Kompositionsmodul vorstellen, welches einzelne Sinustöne sowohl melodisch als auch im Raumort verteilt, und zur klanglichen Realisierung des letzteren sich der Schaltung des PGP-Projektes bedient und für diese (als einfachste Steuermethode) einen Datensatz in der Sprache PGP-0.2 automatisch generiert.

8.6 Bemerkungen zu Rezeption und Semantik.

8.6.1 Zur Semantik des Grundmodells.

Die Ausgangssituation der Wahrnehmung seitens eines Besuchers unserer Installation ist zunächst *paradox*:

Wir können annehmen, daß zum Zeitpunkte des Startes einer dünn besetzten, einstimmig und „realistisch" beginnenden Partitur ein Mensch sich in den Wendeltreppen aufhält. Dieser vernähme zunächst entgegenkommende Schrittgeräusche und erwartet, eine reale Person erscheinen zu sehen.

Spätestens wenn die Geräusch an ihm „vorbeigelaufen" sind, klärt sich ihre artifizielle Natur. Das Paradox der „unsichtbaren Schauspieler" wird aber relativ schnell als solches akzeptiert. Ab dann sind drei primäre, sehr unterschiedliche Rezeptionsschienen möglich, deren Auswahl und Dominanz im konkreten Fall sehr verschieden sein kann und vom konkreten Klangverlauf des Werkes einerseits und vom Interesse des Hörers andererseits bestimmt werden:

1. Aktives Spiel mit akustischen Erfahrungen.
2. Mischung von Realität und Fiktion.
3. Hörspiel.

Zu 1:

Unmittelbar sich anbietende Wahrnehmungsschiene (im Falle einer o.e. „dünnen" Partitur) ist wohl die Erfahrung (und Er-gehung) der akustischen Phänomene. Diese entspricht der allgemeinen Funktion des modernen Kunstwerkes als Vermittler von rezeptionstheoretischer Aufklärung und Verunsicherung und findet auf unserer Materialebene $\alpha \to \beta$ statt.

Die technische Realisierung unserer simulierten Schritte ist ja keinesfalls perfekt. Einmal als solche erkannt, kann ihre bewußte Beobachtung und der Vergleich mit den eigenen Schrittgeräuschen und denen anderer Anwesender durchaus die Aufmerksamkeit des Hörer auf die Feinstruktur akustischer Alltagserfahrung lenken und ihm so bewußteres Erleben tagtäglicher Wirklichkeit vermitteln[17].

Selbst eine perfekte Realisierung des von uns gewünschten Effektes[18] wäre auch nur für bestimmte Positionen perfekt. Durch aktives Gehen durch das Treppenhaus kann der Hörer also die unterschiedlichen Resonanzverhältnisse seiner eigenen

[17] Wenn man vom Essener Hauptbahnhof Richtung Volkshochschule am neuen „otelo"-Verwaltungsgebäude vorbeigeht, sieht man sein Spiegelbild in den vergoldeten Fensterscheiben dessen Erdgeschosses. Normalerweise sind diese unregelmäßig abwechselnd geschlossen und gekippt. In ersteren erblickt man sein normales Abbild, in letzteren aber, wenn man den richtigen Abstand vom Gebäude einhält, sieht man einen mißgestalteten Gnom, bei dem der Kopf direkt auf den Hüften sitzt! Es ist ein riesiger Spaß, in zügigem Tempo vor dieser protzigen Front hin- und herzugehen und sich vom ständigen Umspringen der eigenen Gestalt überraschen zu lassen.

Unter Dutzenden von Passanten fand der Verfasser *einen einzigen*, der dieses kostenlose Lachkabinett beim Vorbeigehen ebenfalls sofort bemerkte und zu genießen wußte. Die anderen alle *sahen* den Effekt noch nicht einmal und hielten uns beide wohl für betrunken.

[18] Besonders kritisch ist das „Vorübergehen" einer Schrittfolge am Hörer.

Schritte und unserer elektronischen Signale recht abwechslungsreich untersuchen, er kann sogar bis hinter die Lautsprecher steigen und auf das *Echo* des Signals in der zentralen Luftsäule der Wendeltreppen lauschen etc.

Zu 2 :

Falls eine dünngesetzte Partitur abläuft und der Hörer sich nicht alleine im Treppenhaus aufhält, kommt als zusätzliche Quelle von Irritationen die Mischung zwischen realen und virtuellen Schritten hinzu.

Zu der Erweiterung der Erfahrungsschiene (1) durch die Wahrnehmung der akustischen Differenzen kommt eine Erhöhung der Aufmerksamkeit für die Geh-Charakteristika der anwesenden Mithörer, also für ihre Physis und Verhaltensweisen.

Zu 3 :

Eine etwas dichter gesetzte Partitur mir mehreren Stimmen kann, wenn die Zuhörer sich selbst leise verhalten, als *Hörspiel* mit konkreten, aber abstrakt dargestellten Inhalten rezipiert werden.

Hier befinden wir uns auf unseren Materialebenen δ und ϵ.

Denn schon so primitive Strukturverläufe wie oben in Bild 8.2 haben in Verbindung mit der Situation „Gänge im Treppenhaus" einen überraschend engen Bereich von semantischen Interpretationsmöglichkeiten, sobald man sie als *konkrete Hörspielhandlung* begreift.

Mögliche Deutungen der Einzelszenen in Bild 8.2 sind ...

1. Anton war auf dem Weg in Bertas Büro, Berta aber auf dem Weg zu Anton. Sie treffen sich, reden stehend eine Weile (was in unserem Modell durch *Stille* repräsentiert wird!) und gehen dann gemeinsam in Antons Büro.

2. Berta erreicht den langsamer in dieselbe Richtung gehenden Anton, beide reden eine Weile und gehen dann gemeinsam weiter, wobei der Untergebene (euphemistisch: „Mitarbeiter") sich der Ganggeschwindigkeit des Vorgesetzten anpaßt. Also ist Anton hier der Chef, was auch der langsameren Gangart entspricht. (Der gegenteilige Fall steht in der Klammer.)

3. Mittagspause! Alle treffen sich auf dem Weg zur Kantine.

4. Berta geht Anton entgegen. Sobald sie ihn kommen hört, macht sie schleunigst und beschleunigt kehrt.

5. Berta geht aufwärts. Sobald sie Anton vor sich aufwärts gehen hört, verlangsamt sie ihren Schritt, um ihm nicht zu begegnen.

6. Berta hört, daß Anton sie einholen wird. Sie unterbricht ihren Gang (hinter einer Säule?) und setzt ihn fort, nachdem Anton vorbei ist.

Man sieht, daß schon die primitivsten Schallereignisse teilweise Inhalte bis in den sozialen und psychischen Bereich nahelegen, ohne daß man überinterpretiert.

8.6.2 Allmähliche Verschiebung von Rezeptionsstandpunkt und Semantik.

Sobald das Paradoxe der Situation (= „Hören von unsichtbaren Personen") vergessen und die Darstellungsform als quasi-Hörspiel eingeordnet und als „normal" akzeptiert ist, wird der Hörer diese als „realistisch" bezeichnen.

Von diesem Realismus kann nun über konkrete und skalierbare Maßnahmen eine schrittweise Fortbewegung stattfinden. Diese Maßnahmen kann man zunächst in qualitative und quantitative unterscheiden.

Wir nennen nur einige Beispiele, um danach die Konsequenzen zu diskutieren.

- Die Schrittgröße kann über das menschenmögliche Maß hinaus gesteigert werden (was aber wohl nur in einem Prozeß auch nachvollziehbar und somit mitteilbar ist).
- Die Schrittfrequenz kann absurd beschleunigt (oder evtl. verlangsamt) werden.
- Zwischen der *Wiedergabelautstärke* und dem Charakter des aufgenommen Materials kann es absurde Disproportionen geben.
- Die „paraphysikalische Translokation" kann durch ein Springen innerhalb eines (eigentlich als zusammenhängend empfundenen) Ganges endlich als möglich demonstriert werden (s.o. 8.4.6).
- Normalerweise immer gleichmäßig auftretende Geräusche (z.B. da Telefonklingeln aus einer ajouté-Schicht) können schrittweise (zunächst vorbewußt, dann stärker) verwackelt und ungleichmäßig werden.
- Die Bewegungscharakteristik von Personen kann widerspruchsvoll sein: Sich abmühender Greis auf den Absätzen und gleichzeitig junger Hüpfer auf den Stufen. Dies ist ein Beispiel für „psychologische Absurdität".
- Dazu gehören auch Gänge, deren Motivation absurd erscheint, z.B. x-fach wiederholtes schnelles Hochlaufen und langsames Hinabsteigen auf immer derselben Strecke, o.ä.
- Die den Wiedergaberaum bildende physische Treppe besteht aus Granitstufen; das Schallmaterial kann aber aus Tritten auf Stahlstufen oder Holztreppen bestehen.
- Diese verschiedenen Materialien können auch *gleichzeitig* (für verschiedene `pgPerson`en) erklingen, ...
- oder aber die Stufen werden mit einem Material, die Absätze mit einem anderen Material gebildet (Besonders solche Inkonsistenzen können der „didaktischen Funktion" des Kunstwerkes dienen, Aufmerksamkeit für die Alltäglichkeit zu wecken).

- Sobald eine `pgPerson` einmal als identifizierbar erkannt ist, besteht eine Irritation, wenn z.B. `.aufwaerts` und `.absatz` mit unterschiedlichem Schuhwerk erklingt.

Alle diese Irritationen treten allerdings nur dann auf, wenn der satztechnische Zusammenhang sie als *Abweichung vom Möglichen* auch nachvollziehbar macht! Das Paradoxe der o.e. Ausgangssituation ist längst als normal akzeptiert, nun treten neue Paradoxe auf.

Eine große Gruppe von Maßnahmen besteht aus solchen, mit denen quantitative Veränderungen der *Abmessungen der Realität* jenseits der normalen Erfahrung herbeigeführt werden. Das Ergebnis kann als „*surreal*" bezeichnet werden im dem Sinne, daß ein surreales Modell den strengen Regeln einer eigenen Realitätsdefinition treu folgt, – nur sind diese Regeln nicht die unseres Universums. Es ist, als wären die sog. Naturkonstanten lediglich mit anderen Größen defininiert.

Eine andere Gruppe erinnert (was die Konsequenzen in der Wahrnehmung betrifft) teilweise an die *frühen Comic-Filme*, in welchen oft die Meta-Ebene des darstellenden Mediums aufgedeckt wurde, indem sie verlassen oder durchbrochen wurde : Bei Verfolgungsjagden schießen Figuren über den Rand der Zeichenfläche, – bei Abstürzen in Schluchten greift im letzten Moment die Hand des Zeichners ein, um den Helden aufzufangen, – Figuren beseitigen ihre Gegenspieler einfach mittels Radiergummi etc. All diese Stilmittel dienen der didaktischen Kunstwerk-Funktion im besten BEETHOVENschen Sinne: Aufklärung über die Mechanismen der Kunst mittels ihrer Durchbrechung.

8.6.3 Umschlagen (Zurückschlagen) der Rezeptionsebenen.

Eine weitere Übertreibung der pseudo-physikalischen Gegebenheiten (Lautstärke, Schrittfrequenz etc.) bedeutet zunächst einen Wechsel vom Surrealen ins Irreale, – führt dann aber natürlicherweise zu einem typischen *Umschlag von Quantität in Qualität* , nämlich zur *Verklanglichung*: Immer mehr Personen, die immer schneller laufen, können zunächst noch als „Szene"rezipiert werden.

Dauern derartig dichte Klangteppiche aber längere Zeit an, so kann der entscheidende Schritt geschehen, daß die *Wahrnehmung* des Hörers umschlägt: Interferenzstrukturen, die Resonanz der Luftsäule, die Filterfunktionen des Gebäudes, die beim Durchlaufen des Schallfeldes sich verschiebenden Formanten etc. treten dann *schlagartig* in den Fokus von bewußter Wahrnehmung und Interesse. Das Hörspiel wird zur Klangsäule, – ihr Abschreiten wird zu einem anderen, nunmehr stärker sinnlichem als intellektuellem aktivem Rezeptionsverhalten.

Eine quantitative Übersteigerung der mühsam konstruierten Materialebene ζ schlägt also zurück in ein Wahrnehmen auf der Ebene α!

Auch ohne massive Verdichtung, z.B. durch extrem *regelmäßige Wiederholungen* (mit evtl. leichten, zunächst unmerklichen Phasenverschiebungen), welche mit der Zeit als „nicht mehr realistisch" erkannt werden, kann eine kontrollierte Verschiebung des Wahrgenommenen erreicht werden, – hier: vom Hörspiel zur rhythmischen Pattern-Komposition.

8.6.4 Zitieren externer Semantik.

Durch geeignete Wahl von Schallmaterial und Satzstruktur können darüberhinaus durchaus konkrete Bedeutungen zumeist sozialer Natur zitiert und angesprochen werden:

- Erst wenige, dann immer mehr zum Ausgang *rennende* Schritte stehen wohl weniger für „Mittagspause" als vielmehr für „Panik bei Feueralarm".
- Ein einsamer, metallisch stapfender Nachtwächter kann durchaus „suspense" im Sinne HITCHCOCKs hervorrufen.
- Eine Gruppe von gehenden Menschen, die zunehmend im Gleichtakt abwärts marschiert, unter zunehmender Angleichung von Bewegungsparameter, Schuhwerk und Klangmaterial, kann den heutzutage so beliebten kollektiven Marsch in den Abgrund symbolisieren.
- Eine „in ruhig festem Schritt" marschierende Vielzahl von Kampfstiefeln[19], die plötzlich zu laufen beginnen, bedeutet wahrscheinlich eine SA-Horde, die ein Opfer erspäht hat. Ihre Absicht und ihre folgende Aktivität wird allemal in höchstem Maße gegenwärtig, ohne daß man die Schreie der Geprügelten zu hören braucht.
- Um mit einem nicht so düsteren Beispiel zu endigen:

  Sich entsprechend dem Wechsel von Absatz und Stufe je Stockwerk wiederholendes rhythmisches Hüpfen und Trippeln steht eindeutig für ein *Kind*, welches spielerisch versucht, die für diese Treppe adäquate Fortbewegungsmethode zu entwickeln.

[19] Der Verfasser meint, daß in allen derartigen Fällen die klassische Differenzierung von „Konnotation" und „Denotation" interessanterweise nicht adäquat ist.

Kapitel 9

Beiträge zu einem Begriff des musikalischen Kunstwerkes.

9.1 Grenzen vernunftgemäßer Untersuchung.

Jede Untersuchung ästhetischer oder künstlerischer Phänomene mit den *Mitteln der Vernunft* stößt hinsichtlich dem Bereich des Erforschbaren und der Gültigkeit der Ergebnisse durchaus an Grenzen[1].

Vielleicht wird es nie möglich sein, und wahrscheinlich ist es weder sinnvoll noch notwendig, mit rein vernunftgemäßen Mitteln zu erklären, warum eine bestimmte funktionalharmonische Abfolge wie T-Tp oder T-d eine konkrete absolute intra-psyschische Gestimmtheit zur Folge hat.

Auch bilden der soziale und historische Kontext des Musik-Erlernens, – Kindheit, Ahnung, Pubertät, Leid und Festlichkeiten, Schlachtengesang und Liebeslied, – in jedem Menschen ein je eigenes Semanik-Geflecht heran, welches sehrwohl ein tatsächliches, *substantielles Material* darstellt, ohne jemals trans-personal mitteilbar zu sein, geschweige denn der vernünftigen Analyse zugänglich.

Mehr noch als in ethischen Fragen, wo die Vernunft eine unabhängige Axiomatik *neben* sich dulden muß, ist hier ihre demütige Selbstbeschränkung gefordert, wo sie solche vorbewußten Verstrickungen als *konkretes Material*, auf dem sie zu operieren hat, *unter* sich als gegeben anerkennen muß, – und *über* sich die Wirklichkeit tagtäglicher kultureller Praxis als Rahmenbedingung sieht.

Wenn diese Beziehungen und Erlebnisgehalte aus Sicht der Vernunft sich deshalb zwar im Nebel der Unbestimmbarkeit verlieren können, so sind sie dennoch für die Spiritualität des Menschen, welche spätestens im Tode der wahrscheinlich wichtigere Teil unserer Existenz sein wird, dennoch höchst *konkrete* und konstitutive Bestandteile, da *nur sie* (bestenfalls) ein unvermitteltes Eingebunden-Sein in die

[1] Begrenzt ist eine solche Untersuchung des im *engeren* Sinne „Ästhetischen" selbstverständlich auch dadurch, daß von ihr nicht bearbeitete Gebiete (produktionstechnische, wirtschaftliche, medienpolitische, soziologische, etc.) auf die konkrete Rezeption und Wirksamkeit ästhetischer „Produkte" (= häßlichstes neudeutsches Modewort !) stets einen enormen Einfluß haben, welcher den der substantiellen Gegebenheiten eines Werkes leider sogar weit übersteigen kann!

überpersönliche Kontinuität des Menschengeschlechtes auf seinem Weg durch die Zeit, und damit in die Ewigkeit, mittels quasi körperlicher Empfindung unwiderlegbar als wahr beweisen können.

Künstlerisches Erleben, Arbeit, Sexualität und Gebet sind die einzigen tragfähigen Brücken für den Menschen zu seiner Bestimmtheit.

Keine Beschränktheit jedoch entbindet die Vernunft, ihre analytische Tätigkeit bis an ihre Grenzen fortzuführen, und dann auch aus deren Ergebnissen Folgerungen zu ziehen, welche durchaus *gültige Randbedingungen* sein können für Gebiete, auf denen sie selbst nicht mehr zu urteilen hat.

In diesem Sinne sind die folgenden abschließenden Bemerkungen gemeint.

9.2 Transzendentalanalytische begründete existentielle Probleme des Kunstschaffenden.

Während der Arbeit an einem Werk ist das tätige Erleben der Mittelgrund-Strukturen eine Auffaltung der Innenwelt des Arbeitenden. Diese wird manche Zeit als quälend und verunsichernd, andere Zeit aber als überaus lustvoll erlebt, – manchmal beides zugleich.

Der eigene psycho-interne Mittelgrund, im Werk und im Erfahren der Eigengesetzlichkeiten des Materials quasi ver-objektiviert, *überlappt* dabei zum einen (in fast erotisch zu nennender Intensität) mit den psycho-internen Mittelgrundstrukturen längst verstorbener Kollegen und Vorläufer, zum anderen mit den ewigen Gesetzen der formalen Bestimmungen von Wahrnehmung schlechthin.

Im Moment des „release" jedoch, wenn die primäre Arbeitsphase am Werk beendet ist, kollabiert diese n-dimensionale Mittelgrundstruktur zu einem (anscheinend) *flachen*, ja häßlichen „Ding".

Das Werk in der Aggregatform, in der es erst einmal nach außen tritt und sich bewähren soll, ist ja nur eine magere Projektion einer Hinter- und Mittelgrundsstruktur, die bis vor kurzem noch mein Denken ganz erfüllte, – ein Mikro-Kosmos, geschmückt mit Lichtspielen, irisierenden Beziehungen, tiefen Allusionen und geheimnisvollen Querverbindungen.

Der psycho-interne Mittelgrund gleicht dem Blick *in* einen Diamanten, die flache Partitur nur dessen Fotografie.

Das aus diesem Gefühl von Schalheit möglicherweise entstehende Gefühl der Vergeblichkeit kann nur durch einen *dynamischen* Begriff vom Kunstwerk widerlegt werden.

Der Mittelgrund nämlich wird *im Hörer* – zwar unvollständig und bestenfalls ähnlich – wieder zum Leben erweckt werden.

9.3 Ein Kunstwerk ist Syntax.

- Ein jedes Kunstwerk selbst *ist* Syntax.
- Ein jedes Kunstwerk ist selbst *Sprache*, nicht Gesprochenes.

Die wesentliche Zweckbestimmung nämlich des Kunstwerkes („Dinges an sich" namens Kunstwerk) ist es, von jedem es wahrnehmenden Menschen jedesmal neu *weitergedacht* zu werden.

In diesem Sinne *ist* das Kunstwerk eine Menge von Ableitungs*regeln*, die in verschiedene Richtungen fortgesponnen werden können, – also eine Syntax; es ist somit eine strukturierte Menge von unbegrenzt vielen neuen Möglichkeiten, – also eine Sprache, von denen die gerade erklungenen Tonfolge nur *eine* prototypische Variante von vielen war (mit vielleicht bestimmten auszeichnenden Eigenschaften).

Der Verfasser z.B. ist als Kind oft durch den Wald gehüpft, und hat dabei denkender und/oder singender Weise die Durchführung des ersten Satzes jener Fünften Sinfonie (qua Variantenbildung) problemlos auf die doppelte Taktzahl gebracht.

Genau determiniert aber das Kunstwerk die unterschiedlichen Wahrscheinlichkeiten der verschiedenen Entwicklungsrichtungen, das Kunstwerk prägt dem kreativen Geiste dessen, der es justament wahrgenommen hat, prägt der Gesamtheit der Objekte seines inneren Handelns seine spezielle Wahrscheinlichkeitsmetrik auf.

Das Kunstwerk in seinem jeweiligen Wahrgenommenwerden wirkt also wie ein Tensor, welcher mehr oder weniger erfolgreich die Metrik des Wahrscheinlichkeitsraum meiner Denkmöglichkeiten verformt.

Der Verfasser kennt z.B. verschiedene Stufen des Rekapitulierens musikalischer Eindrücke : vom eidetischen Wiederhören einer Schallproduktion, einschließlich aller Kieckser des dritten Hornes, über ein Empfinden der gemeinten Satzstruktur, des reinen Modulationsplanes frei von jedem Schallmaterial, bis hin zum Empfinden der Summe aller satztechnischen Möglichkeiten, quasi als statische Landschaft synchron vor dem Inneren Ohr ausgebreitet.

Im Rahmen der lebenslangen Musikrezeption tritt neben den pädagogischen Gesamteffekt noch ein interessantes Einzelphänomen : Dem Autor z.B. geht nach dem (seltenen) Hören von Beethovens Fünfter Sinfonie stundenlang — dessen Sechste nicht aus dem Ohr.

Das eine Werk verlangt nach dem anderen zur Ergänzung, die Werke in Folge bilden ihrerseits ein Kunstwerk; Beethovens letzte Klaviersonaten bilden augenscheinlich eine Meta-Sonate, aber Beethovens gesamtes Klavierwerk kann ebenfalls den Begriff „Kunstwerk" in userem oben aufgestellten Sinne beanspruchen.

Eine wichtige Komponente des emanzipatorisch/didaktischen Grundkonzeptes z.B. bei Beethoven besteht ja gerade in der Nachvollziehbarkeit der strukturgenerierenden Vorgänge, also in ihrer *Fortsetzbarkeit* durch den Hörer.

(Die prägnantesten Beispiele sind u.E. die Coda des vierten Satzes der Achten Sinfonie oder die Durchführung des letzten Satzes des cis-moll-Quartettes.)

Diese Fähigkeit zum inneren Weiterdenken (und damit die Kenntnis potentieller Varianten) nämlich *befähigt erst* den Hörer, die vom Komponisten (auf die-

ser Betrachtungsebene gleichsam „willkürlich") gesetzten Eingriffe, Modifikationen, Unterbrechungen und Umleitungen z.B. eines Modulationsganges, *erstens* als dessen bewußten Entscheidungen überhaupt zu erkennen, – *zweitens*, sich zeitweilig als „nicht einverstanden" zu erklären, – wodurch die kommunikative Spannung zwischen Komponist und Hörer erst entsteht –, und *drittens*, im Nacherleben den Modulationsweg halt *anders* weiterzuspinnen.

Der Lustgewinn beim Erleben des späten BEETHOVEN resultiert ja immer aus einem (bewußt oder unbewußt wahrgenommenen) *Erkenntnisschritt*, und somit einem *Lernschritt*, der darin besteht, daß eine zunächst als widersinnig oder gar unmögliche erlebte satztechnische Behauptung nach Darlegung und Durchführung ihrer Konsequenzen sich letztlich doch als berechtigt, sinnvoll und organisch herausstellt, — die Widerlegung des Vorurteils ist konstitutiv für diese Musik.

(Man denke an die Modulation nach A-Dur in der Exposition des letzten Satzes des späten F-Dur-Quartettes op. 135.)

Beim auswendigen Spielen Bachscher Klavierwerke kann es passieren, daß man bei einer der typischen „Weichenstellen" falsch abbiegt. So kann man z.B. im ersten oder dritten der vier Duette aus den Klavierübungen durch reine Nachlässigkeit neue wunderbare kontrapunktische Vertauschungen der beiden Stimmen spielen, die nicht im Notentext stehen, und die dennoch von Bach sind.

Das Kunstwerk ist somit eine Kommunikation auf einer Meta-Ebene, indem es selbst eine *Sprache* kommuniziert.

Diese Sprache wird dem Hörer zur eigenen Verwendung zur Verfügung gestellt und kann, ja wird sein Handeln beeinflussen.

Insofern sind die kunsthistorisch bekannten Konzepte des „offenen Kunstwerkes", der „offenen Form" etc. zu kurz gegriffen, als die Offenheit im diesem Sinne per se zum Kunstwerk gehört.

Dieser Sprachcharakter des Kunstwerkes rechtfertigt, ja fordert eine Produktionsweise, die auf der Tiefenstruktur operiert, fordert die „out-of-time"-Arbeit, das heißt Materialdispositionen, Planungen und Entscheidungen auf der Ebene des statischen, die Dynamik generierenden Netzwerkes von Bestimmungen.

9.4 Musik als konkret erlebbare transzendentale Analyse.

„Musik ist höhere Offenbarung als alle Philosophie", – dieser Ausspruch eines bekannten Kollegen ist keinesfalls ein schwammiges Sentiment, sondern meint einen konkreten, exakt beschreibbaren Wirkungsmechanismus.

Transzendentale Analyse ist eine vernunftgemäße Untersuchung der Grenzen der Vernunft mit den Mitteln der Vernunft. Diese Analyse ist somit (ex positivo und ex negativo) eine Untersuchung der Möglichkeiten menschlichen Verhaltens und Erlebens schlechthin.

Das Material der Musik ist die Zeit.

Indem die Musik diese Zeit gliedert, staucht und knetet, sie so lange schlagen will, bis sie ihr Geheimnis preisgibt[2], unterzieht sie diese einer transzendentalen (d.h. auf ihre *formalen* Bestimmungen hinzielenden) Analyse, — diese aber findet statt in *praktischer* Form, also hautnah erlebbar.

Die beim Erleben von Musik gewonnenen Erkenntnisse sind somit Erkenntnisse über die Grundbedingung menschlicher Existenz und über die Möglichkeiten unseres Denkens und Wahrnehmens, – Erkenntnisse über Relativität, Vergänglichkeit, Vorurteil, Scheinbarkeit und Manipulierbarkeit.

Die schmerzhafte Falschheit der C-Dur-Scheinreprise im ersten Satz des B-Dur Quartettes op. 130, welches während des Hörens einer Schallplatte den Verfasser dieser Zeilen dem Komponisten zuschreien ließ „Nein! Nicht! Dies ist das FALSCHE Dur!", – und die Erlösung, welche nur durch das entsagende c-moll hindurch (Takt 123 ff) den Hörer wieder seine wahre Bestimmung (Es-Dur) erreichen läßt, – ist Lehrbeispiel für (1) menschliches Irren, Verirren und *Manipulierbarkeit*, und (2) als Prototyp der Aussage „Nur durch das moll geht's zum wahren Dur!" ein Beweisversuch für die Fruchtbarkeit des bewußt ertragenen Leidens.

9.5 Musik als tragische Kunst.

Die Meta-Analyse, stattfindend im Vergleich mehrmaligen Erlebens, stattfindend beim Versuch des Erinnerns, liefert weitere existentielle Erkenntnisse, welche die Auseinandersetzung mit Kunst und Musik als eine letztlich *tragische*, zum (scheinbaren) Scheitern verurteilte Bemühung offenbaren.

Das tatsächliche Erleben eines Werkes läßt sich weder wiederholen noch fixieren, – weder das allererste, „virginale" Hören, noch das eben geschehene ...

Die *eigentliche Wirkung* der C-Dur-Kadenz vor Brünnhildes Erweckung, – wenn sie aufgrund glücklicher Umstände und entsprechender Disposition des Hörers denn eintritt, – verlangt zu ihrer Entstehung den langen, entsagungsvollen Vorgang der Rezeption der ersten beiden Siegfried-Akte.

Man spule einfach, nach dem „Heil Dir, Sonne", das Tonband zurück und wiederhole die letzten zweihundert Takte, – der überwältigende Glanz der harmonischen Disposition wird sich auf diese Weise keinesfalls mehr einstellen.

Das Kunstwerk in seiner Existenzform als Modell indiziert also ein *eigenes Modell* je Mensch und je Mal.

Gerade in der Musik offenbart sich das notierte/fixierte Kunstwerk als *tragisch*, – im Gegensatz zum erlebten Werk, bei welchem das Verrinnen der Zeit (entweder durch Erfahren neuer Erkenntnisse über unser Lustvermögen, oder durch eher körperliche Erfahrung, wollüstiges Massieren und Verformen unseres physiologischen Zeitempfindens) als durchaus beglückend empfunden werden kann und die Vergänglichkeit und Uneigentlichkeit des letztlich Gemeinten so überdeckt.

[2]C.f. Stanislaw Lem, Sterntagebücher, 28. Reise, Schilderung der letzten Forschungen des Jeremias Tichy, (Lem, 1971).

Besonders schlimm ist die Erfahrung, daß das allererste Hören, welches uns bei den meisten Werken den überwältigendsten Eindruck vermittelte, nie mehr rekonstruierbar ist.

Bei jeder Rezeption wird ein Begriffssystem aufgebaut, welches, einmal entstanden, die neuerliche, nächste Rezeption in ganz andere Bahnen lenkt. Erfolgreich sind deshalb verständlicherweise diejenigen Werke, welche auf die Bahnen des zweiten und dritten Hörens verlocken und wegen Komplexität *bei Nachvollziehbarkeit* ihrer Mittelgrundstruktur auch weiterhin Genuß und Erkenntnis mit jedem Mal zu steigern versprechen.

Wegen dieser (quasi rekursiven) Modifikationen der auf einander folgenden Rezeptionsvorgänge in der persönlichen Geschichte des Hörers muß der Versuch der Konstruktion eines Kunstwerkes als der Versuch einer *Fixierung* zwangsläufig scheitern; gerade in der Kunst gibt es keinen dauernden Besitz, sondern nur ständiges Erwerben.

Auch der Künstler „besitzt" sein „eigenes" Werk *nicht*, auch er kennt nur die diversen Modelle der verschiedenen Schichten seines Denkens, – jedes nur partiell gültig und wandelbar.

Weder der Hörer noch der Autor „*hat*" das Werk !

Was ich gestern (oder in meiner Jugend oder gerade eben noch) deutlich und klar empfunden und aufs deutlichste ausgebreitet gesehen habe, ist im nächsten Moment schon (als Empfindung) verloren und nur noch bestenfalls als abstrakte Bergrifflichkeit mir selbst kommunikabel.

Der Künstler *verliert* also das Kunstwerk; – was er in seiner Inneren Sprache so deutlich und klar besessen hat, verliert er in dem Moment, wo er es in materielle Aggregatformen und somit andere Sprachen übersetzen muß.

Einerseits scheint künstlerisches Schaffen genau im Kampf gegen diesen Verlust zu bestehen: jahrhundertelang entwickelte künstlerische Praxis, Routine, Tradition und gewachsene „Form" scheinen die Substanz des Gemeinten zunehmend exakter festhalten zu können.

Andererseits jedoch bleibt der Verlust unabwendbar, da der künstlerischen Tätigkeit als solcher wesentlich immanent.

Diesen Widerspruch zu akzeptieren, kreativ auszuhalten und produktiv umzusetzen ist jedes künstlerisch Schaffenden persönliches Problem und Aufgabe.

Wissenschaft, Technologie und künstlerische Praxis und Lehre aber haben dennoch den Sinn, diesen Verlust zu bekämpfen, ihn immer weiter hinaus zu schieben, um die Fronten, an denen das Scheitern letztlich stattfinden *muß*, immer deutlicher heraustreten zu lassen.

Anhänge.

Anhang A

Kurzübersicht der APOS-Sprachkonstrukte.

A.1 Äußere Interpretation.

Zum besseren Verständnis u.a. der in Kapitel 7 aufgeführten Quelltext-Ausschnitte folgt hier eine Kurzübersicht der Grundkonstruktionen von APOS[1].

Wir halten uns im folgenden möglichst knapp und informell und vertrauen auf die Anschaulichkeit der Beispiele des Haupttextes und die richtige Intuition des erfahrenen Lesers.

1. In APOS ist die kleinste Befehlseinheit eine *Nachricht*.
2. Diese ist eine Folge (= nichtleeres n-Tupel) von *Objektreferenzen*.
3. Eine *Anfrage* ist ein endlicher, zyklenfreier Baum, dessen Blätter Objektreferenzen tragen. Die Nachfolger jedes innere Knotens sind geordnet. Den in jedem inneren Knoten beginnende Unterbaum des Baumes nennen wir *Unteranfrage* der Anfrage.
4. Ein *Äußerer Interpretierer* übersetzt (korrekte) Zeichenfolgen in Anfragen.
5. Der im folgenden verwendete Äußere Interpretierer erkennt[2] ...
   - 5-1 zusammenhängende Ziffernfolgen als *Zahlenkonstanten*
   - 5-2 in Doppelquotierung eingeschlossene Zeichen als *Zeichenkettenkonstanten*,
   - 5-3 runde und eckige Klammern werden als solche erkannt, und
   - 5-4 alle zusammenhängenden Folgen aller übrigen Zeichen als *Lexikalische Identifizierung* (= *LexRequest*).

[1] Die folgende Liste könnte sehr wohl das didaktische Raster eines kurzen Einführungskurses abgeben!

[2] Vereinfacht dargestellt! Die tatsächliche Syntax der unterschiedlichen Lexeme ist selbstverständlich komplexer. Sie enthält „escape characters", Angabe von Zahlenbasis und -exponent etc.

6. Der im folgenden verwendete Äußere Interpretierer zerlegt zunächst eine Buchstabenfolge in Teilstücke, gemäß diesen vier Kategorien.

   Diese Teile werden dann je in eine Objektreferenz übersetzt, und zwar ...

   6-1 Zahlenkonstanten in Referenzen auf Zahlenobjekte,
   6-2 Zeichenkettenkonstanten in Referenzen auf `name`-Objekte.
   6-3 Klammern liefern keine Objektreferenz, sondern bedeuten, – wie man vermuten kann –, daß der in ihnen enthaltene Text eine *Unteranfrage* (= Unterbaum) beschreibt, der an dieser Stelle in die (ansonsten linear von links nach rechts auf derselben Ebene wachsende) Anfrage (= Baum) einzufügen ist[3].
   6-4 `lexRequest`s werden durch eine Referenz auf das Objekt ersetzt, welches im momentanten Kontext-Zustand des Äußeren Interpretierers unter diesem Namen bekannt ist[4].

7. Arithmetische Operatoren, Zahlenwerte, Funktions- und Objektnamen, Parameterreferenzen, Schlüsselworte für Kontrollstrukturen etc. sind *sämtliche gleicherweise und unterschiedslos* als „Objektreferenzen" behandelt.

8. Unter der Voraussetzung, daß im aktuellen Lexikalischen Kontext Objekte namens „`gaga`", „`+`", „`show`", „`if`" und „`++--08gaga`" bekannt sind, sind folgende Zeilen Beispiele syntaktisch korrekter Anfragen:

```
                4 + ( 44 + 444 )
show ( ((gaga )) ++--08gaga ) 44 44
 ( show ) ( ( if  ) ) ((( show )))
```

[3]Wahr ist: matchende Paare von runden und eckigen Klammern werden ersetzt durch eine Referenz auf ein `block`-Objekt, welches sich aus der Äußeren Interpretation der zwischen ihnen stehenden Zeichenfolge ergibt.

[4]1. Existiert kein solches Objekt, so wird normalerweise eine `EXCEPTION` ausgegeben, und zwar eine „`lex-Not-Found-EXCEPTION`".

2. Der sog. Lexikalische Kontext besteht aus einem Lifo von sog. `ensembles`. Diese entsprechen partiellen Funktionen von `name` nach `Lexobject`. Die Überlagerungen der Funktionen der einzelnen „konsultierten" Ensembles bilden die aktuelle `name`→`lexobject`-Funktion, so daß Objekte in später konsultierten Ensembles gleichnamige Objekte in früher konsultierten Ensembles unsichtbar machen.

3. Die Methoden zur Manipulation des Lexikalischen Kontextes sind ...

```
consult (any ensemble)
  consulting ensemble
close consulting ensemble
close consulting (any ensemble)
        list consulting
```

usw.

4. Bitte experimentieren Sie.

9. Die entstandene Anfrage (= Baum von Objektreferenzen) wird vom Äußeren Interpretierer normalerweise dem *Inneren Interpretierer* zur Auswertung übergeben.

10. Der *Innere Interpretierer* liefert als Ergebnis der Auswertung einer Anfrage ein *Resultat*. Diese wird vom Äußeren Interpretierer auf dem Bildschirm zur *Anzeige* gebracht.

A.2 Innere Interpretation – Auswertung einer Anfrage.

11. Eine Anfrage, welche mit einer Referenz auf das ausgezeichnete Objekt `quote` (= „`'`") beginnt, liefert als Resultat ihrer Auswertung ein *Blockobjekt* zurück.

    Ein Blockobjekt ist ein n-Tupel von Objektreferenzen.

    Zählen wir die Komponenten der quotierten Anfrage und des resultierenden Blockobjektes beide beginnend mit null(=0), so gilt:

    Steht in der auszuwertenden Anfragen an der Position $n > 0$ eine Objektreferenz, so steht im Blockobjekt (= Resultat der Auswertung) an der Position $n-1$ dieselbe Objektreferenz[5].

    Steht in der auszuwertenden Anfragen an der Position $n > 0$ eine Unter-Anfrage (also in der externen Repräsentation: ein Klammerausdruck), so steht im Blockobjekt (= Resultat der Auswertung) an der Position $n-1$ das Resultat der Auswertung dieser Unteranfrage (= des Klammerinhaltes)[6].

12. Statt der Kombination „runde Klammer, Quote" können auch einfach *eckige Klammern* geschrieben werden[7].

    Beispiele:

```
    ' 4 [5 6]
⟹ [ 4 [5 6] ]
 ' 4 [5 (3 + 3) ]
⟹ [ 4 [5 6] ]
```

[5] Man beachte, daß das `semikolon` hier als normale Objektreferenz behandelt wird.

[6] Im Gegensatz zu z.B. LISP erstreckt sich die Wirkung einer Quotierung also *nicht in die Tiefe*. In APOS werden die in einer quotierten Klammer enthaltenen Klammern normalerweise ausgewertet, so sie nicht ihrerseits explizit quotiert sind. Das spart uns die Notwendigkeit eines „de-quote"-Mechanismus.

[7] Dargestellt werden hier und im folgenden die *Externen Repräsentationen* auszuwertender Anfragen, evtl. interne Zwischenformen des schrittweise abgearbeiteten Anfrage-Baumes und des Resultates der Auswertung nach dem Schema

$$\textit{Anfrage} \Longrightarrow \textit{Zwischenzustand} \Longrightarrow \textit{Resultat}$$

Evtl. *Bildschirmausgaben* als *Seiteneffekte* einer expliziten `show`-Nachricht werden hingegen durch einen `Rahmen` gekennzeichnet.

13. Eine Anfrage, welche *nicht* mit einem `quote` beginnt, und welche auf der *obersten Klammerebene* (= als unmittelbaren Nachfolgeknoten der Wurzel des Anfrage-Baumes) mindestens eine Referenz auf das Objekt „`;`" (= „*Semikolon*") hat, wird ausgewertet, indem *zunächst* die Folge der Komponenten der Anfrage (Objektreferenzen und Unteranfragen) *links vom linkesten* Semikolon wie eine Anfrage (= *Teilanfrage* gemäß dem folgenden Schritt 14 ausgewertet wird, *und zeitlich danach* (sic!) die Folge der Objektreferenzen rechts von diesem Semikolon wiederum diesem Schritt 13 unterzogen wird.

    Das Resultat der Gesamtanfrage ist das Resultat der *zeitlich letzten* Auswertung nach Schritt 14 (also gleich dem Resultat der Auswertung der Anfrage rechts vom rechtesten Semikolon!).

14. Anfragen, welche nicht mit einem „`quote`" beginnen und (auf oberster Klammerebene) *keine* Objektreferenz auf ein Semikolon enthalten, aber *wiederum Anfragen* (externe Repräsentation: Klammerpaare), werden ausgewertet, indem von links nach rechts alle diese sog. Unteranfragen *ersetzt werden* durch das Resultat ihrer Auswertung.

    Die so entstehende vereinfachte Anfrage wird dann nach Schritt 15 ausgewertet, welcher auch das Resultat der Auswertung bestimmt.

A.3 Innere Interpretation – Auswertung einer Nachricht.

15. Eine Anfrage, welche nicht mit einem `quote` beginnt, die keine Referenz auf das Semikolon enthält und auch keine Unteranfragen enthält (die also ein „flaches" n-Tupel von Objektreferenzen ist, = deren äußere Repräsentation keine Klammern enthält), wird vom Inneren Interpretierer ausgewertet, indem die Folge von Objektreferenzen als *Nachricht* betrachtet wird, und diese Nachricht ausgewertet wird.

    Das Resultat der Auswertung der Anfrage ist gleich dem Resultat der Auswertung dieser Nachricht.

16. Die *Auswertung* einer Nachricht liefert *immer genau eine* Objektreferenz als *Resultat* zurück.

17. Die Auswertung einer Nachricht geschieht, indem eine entsprechende Methode gesucht wird. Wird eine entsprechende Methode gefunden, so wird diese auf die Nachricht angewandt, andernfalls wird eine „`Methode-Not-Found-Exception`" ausgelöst.

18. Die Anwendung einer Methode auf eine Nachricht liefert immer genau eine Objektreferenz als *Resultat* zurück; diese bestimmt das Resultat der Auswertung der Nachricht.

19. Dies sind Beispiele für Anfragen, die unmittelbar als jeweils eine Nachricht ausgewertet werden.

```
  4 + 488
⟹  492
  5 ?< 3
⟹ NULL
showCr  555
  555
⟹  OK
```

20. Beispiele für Geschachtelte Anfragen und ihre Auswertung (siehe oben Schritt 14) :

```
4 + ( 44 + 444 )
⟹  4 + 488
⟹  492
```

21. Beispiele für Teilanfragen (sequentiell mit Semikolon gemäß Schritt 13) :

```
show 44 ; show "  gaga " ; show 33 ; show "\n"
⟹ 44  gaga  33
⟹  OK
```

Und in Kombination :

```
  show (44 + 33 ) ; show "  gaga "
⟹  show  77 ; show "  gaga "
⟹ 77
⟹  show "  gaga "
⟹ gaga
⟹  OK
```

22. Eine Nachricht der Gestalt ...

    `with` $oref_0$ $oref_1$ ... $oref_n$ *Block*

    ...wobei *Block* für eine Referenz auf ein Block-Objekt steht und „$oref_0$ $oref_1$ ... $oref_n$" für eine beliebig lange (auch leere) Folge von beliebigen Objektreferenzen heiße `with`-Nachricht.

    Eine solche `with`-Nachricht wird ausgewertet, indem eine neue Anfrage *generiert* wird und danach ausgewertet. Das Resultat dieser Auswertung bestimmt das Resultat der `with`-Nachricht.

    Die neue Anfrage entsteht aus dem in der Nachricht referiertem Blockobjekt, indem

    - Jedes Auftreten der Objektreferenz „`_0`" durch $oref_0$ ersetzt wird,
    - jedes Auftreten der Objektreferenz „`_1`" durch $oref_1$ ersetzt wird,
    - ...
    - jedes Auftreten der Objektreferenz „$\_n$" durch $oref_n$ ersetzt wird[8],
    - und jedes Auftreten einer Objektrefenz, die aus *mehreren* Unterstrichen gefolgt von einer Ziffer besteht, um einen Unterstrich gekürzt wird[9].

    Beispiele :

```
          with 3 4 [ _0 + _1]
            ⟹     3 + 4
                ⟹ 7
with 3 4 [ with _1 [' __0 + __0 ] ]
  ⟹     with  4 (' _0 + _0 )
  ⟹           4  +  4
        ⟹       8
```

23. Eine Nachricht der Gestalt ...

    `if` $oref$ $Block_t$ $Block_f$

    ...wobei $Block_t$ und $Block_f$ für Referenzen auf je ein Block-Objekt stehen und $oref$ für eine beliebige Objektreferenz, (sog. `if`-Nachricht) wird wie folgt ausgewertet:

    Ist $oref$ gleich dem ausgezeichneten Objekt „`NULL`", welches in APOS für *logisch falsch* steht, ist die Auswertung der `if`-Nachricht gleich der Auswertung der Nachricht ...

[8] Das `Block`-Objekt fungiert also im Rahmen einer `with`-Nachricht als „Lambda-Ausdruck", – die Art der Variablen-Identifizierung durch Nummerierung entspricht der bekannten De Bruijn-Notation.

[9] Diese naheliegende Lösung fand der Verfasser dann wieder in den TEX-Makro-Parametern, erfunden von Donald E. Knuth.

`with irgendwas` $oref$ $Block_f$

Ist $oref$ ungleich „`NULL`", so ist die Auswertung der `if`-Nachricht gleich der Auswertung der Nachricht ...

`with irgendwas` $oref$ $Block_t$

Die Auswertung einer Nachricht der Gestalt ...

`if` $oref$ $Block_t$

... entspricht der Auswertung der Nachricht ...

`if` $oref$ $Block_t$ `[ NOP ]`

Die Auswertung einer Nachricht der Gestalt ...

`ifN` $oref$ $Block_t$

... entspricht der Auswertung der Nachricht ...

`if ( NOT` $oref$ `)` $Block_t$ `[ NOP ]`

Beispiel :

```
if (3 ?> 4) [ show "wahr" ] [ show "falsch" ]
                    ⟹falsch
```

24. *Wichtige Kontrollstruktur* ist die Objektreferenz „`__<`" (= „`continue`").

    Beginnt eine *Nachricht* mit einem `continue` (= $n > 1$ underscores gefolgt von einem Kleiner-Zeichen), so besteht ihre Auswertung darin, daß (unter Übernahme des Resultates) eine *um das führende Continue verkürzte Nachricht* ausgewertet wird.

    Der *Programmfluß* kehrt aber *nicht mehr hinter das letzte Semikolon* zurück, wie er es ohne den `continue`-Operator täte (siehe 13), sondern die Programmfortsetzung nach Auswertung der verkürzten Nachricht findet statt ...

    - bei zwei(2) underscores hinter dem *vor*-letzten (während der Ausführung der Anfrage dynamisch ausgeführten) Semikolon,
    - bei drei(3) underscores hinter dem *vor-vor*-letzten Semikolon,
    - etc.

Beispiele :

Quelltext 1

```
1....          show "1111111111111111" ;
2....>         ifn (verify "weitermachen?")
3....>              [ __< showcr "abgebrochen !" ] ;
4....>         show "2222222222222222" ;
5....>         ifn (verify "weitermachen?")
6....>              [ __< showcr "abgebrochen !" ] ;
7....>         show "3333333333333333" ;
8....>         ifn (verify "weitermachen?")
9....>              [ __< showcr "abgebrochen !" ] ;
10...>         showcr "fertig!"
11...;
```

Ende Quelltext

25. Weitere Kontrollstrukturen sind nicht mehr Bestandteil des Sprachkerns (= der Kernsprache), sondern der diversen Bibliotheken.

    Beispiele :

```
                      apl 0 to 2 [ showcr _0 ]
⟹  with 0 [showcr _0] ; with 1 [showcr _0] ; with 2 [showcr _0]
                              ⟹    0
                                   1
                                   2
                    apl tokenlex  [ showcr _0 ]
                              ⟹  add
                                   all
                                   any
                                   apl
                                  ...
```

26. Diese „`apl`"-Befehle werten für die zu besuchenden Objekte der Reihe nach jeweils eine `with`-Nachricht aus. *Dazwischen* steht (logischerweise) ein *gedachtes Semikolon.*

    Auf dieses bezieht sich im folgenden Beispiel der `continue`-Operator.

Quelltext 2

```
1....          apl tokenlex [show _0 ; show "                  "  ;
2....>                       if [verify "weiter ?"]
3....>                           [' showcr ]
4....>                           [' __< showcr "abgebrochen" ]
5....>                      ]
6....;
7....
```

```
8....          add                weiter ? Y
9....          all                weiter ? Y
10...          any                weiter ? N abgebrochen
11...
12...;
```
Ende Quelltext

27. Klassissches Beispiel für geschachtelte Quotierung : der *Tannenbaum* :

Quelltext 3

```
1....          with (userinput Integer "bitte halbe baumbreite eingeben")
2....>             [ apl 1 to _0
3....>                  [' apl 0 to [' _0 - __0 ]
4....>                          ['' show " " ] ;
5....>                     apl 1 to [' + __0 __0 (-1) ]
6....>                          ['' show "^"  ] ;
7....>                     showcr ];
8....>               apl 1 to 2
9....>                  [' apl 0 to [' _0 - 1 ]
10...>                          ['' show " " ] ;
11...>                     showcr "!" ]
12...>              ]
13...;
14...
15...  bitte halbe baumbreite eingeben  : 6
16...           ^
17...          ^^^
18...         ^^^^^
19...        ^^^^^^^
20...       ^^^^^^^^^
21...      ^^^^^^^^^^^
22...     ^^^^^^^^^^^^^
23...           !
24...           !
25...;
```
Ende Quelltext

A.4 Klassen- und Objektwelt.

28. Die Menge der *Klassen* einer APOS-*Objektwelt* bildet einen endlichen, nicht-zyklischen Baum mit der Wurzel „`any ~access`". Die (gesehen aus Richtung der Wurzel) einer Klasse unmittelbar vorangehende Klasse heißt *Superklasse* dieser Klasse.

29. Die Menge der Klassen und die Menge der Objekte einer APOS-Objektwelt sind disjunkt.

30. Jedem Objekt einer APOS-Objektwelt ist eine Klasse als Basis-Klasse zugeordnet. Wir sagen kurz „Ein Objekt o der Klasse c", wenn c die durch die Basisklassen-Funktion zugeordnete Klasse des Objektes o ist.

31. Zur *Erzeugung neuer Objekte* innerhalb der APOS-Objektwelt gibt es sog. *Parent*-Objekte. Diese verstehen eine „`new`"-Nachricht, welche ein neues Objekt generiert. Jedes `Parent`-Objekt beschreibt Struktur und Verhalten aller unter seiner Verwendung generierten Objekte.

32. Die Menge der `Parent`-Objekte einer APOS-*Objektwelt* bildet einen endlichen, nicht-zyklischen Baum mit der Wurzel „`~access`". Das (gesehen von der Wurzel) einem `parent`-Objekt unmittelbar vorangehende `parent`-Objekt heißt *Superparent* dieser Klasse.

33. Jedes `Parent`-Objekt hat eine eigene singuläre Basisklasse. Superklasse dieser Klasse ist die Basisklasse des Superparent.

34. Alle mit einem gegebenen Parent `PAR` erzeugten Objekte (also mittels „`new PAR ...`" erzeugt) heißen *kids* des Parent-Objektes.

35. Die Basisklasse aller kid-Objekte desselben Parent-Objekts ist identisch und heißt „`kidclass`" dieses Parents[10].

    Superklasse der kidclass eines parent ist die kidclass des superparent.

36. Objekte haben normalerweise eine (konventionelle, PASCAL-`RECORD` oder C-`struct`-ähnliche) innere Feldstruktur. Jedes Feld eines Objektes „beinhaltet" zu jedem Zeitpunkt eine Objektreferenz.

    Mathematisch repräsentieren Objekte also n-Tupel von Objektreferenzen mit qua *Feldnamen* adressierbaren Komponenten.

37. Jedes `Parent`-Objekt ererbt Verhaltensweisen und Feldstruktur seines Superparent-Objektes.

    Alle kids eines Parent-Objektes erben Verhaltensweisen und Feldstruktur der kids des superparents des parent-objekts.

[10] Die Basisklasse eines Kids eines Parents kann auch eine singuläre unmittelbare Subklasse der Kidclass des Parent sein, falls die Nachricht „`own`" bezgl. dieses Kid-Objektes ausgewertet wird, welche diesem eine eigene Klasse mitgibt.

38. Die Methoden zur Definition neuer Parent-Objekte heißen ...

    `new Parent` *Name superParent*
    `new Field` *parent Name Parent*
    `enddef` *parent*

    Beispiele finden sich oben im Haupttext, Kapitel 7.

39. Feldnamen werden ggfls. automatisch um den führenden Punkt ergänzt. Der *at-operator* „`@`" liest und schreibt Feldinhalte.

    Beispiele:

```
new latch "gaga"
  @ .value gaga
     ⟹NULL
@ .value gaga 888
  @ .value gaga
     ⟹888
```

A.5 Methodenfindung.

40. Die *Menge aller Methoden* einer APOS-Objektwelt ist eine Menge von Paaren, deren erste Komponente jeweils ein *Klassenmuster* ist, und deren zweite Komponente ein Blockobjekt ist.

    Ein Klassenmuster ist ein endliches, nicht-leeres n-tupel von Klassen. Letzte Komponente eines Klassenmusters kann die *Pseudo-Klasse* „`any ~tail`" sein.

41. Ein Klassenmuster „passt" auf eine Nachricht, wenn an jeder Position des Klassenmusters eine Klasse steht, die identisch mit der Klasse der Objektreferenz ist, oder mit einer Klasse, die durch n-fache Anwendung der Superklassen-Funktion aus dieser hervorgeht.

    Die pseudo-klasse „`any ~tail`" paßt auf jede *nicht-leere*, endliche Folge beliebiger Objektreferenzen.

42. Falls zwei Klassenmuster M_1 und M_2 auf eine Nachricht passen, dann „passt M_1 besser" auf die Nachricht als M_2, wenn für die *linkeste Position* $(= n)$, an welcher beide Muster differieren, gilt:

    Die in M_1 an Position n auftretende Klasse steht im Klassenbaum *näher* an der Klasse der Objektreferenz der Nachricht an Position n als die in M_2 an Position n stehende Klasse.

    Die pseudo-Klasse „`any ~tail`" paßt immer schlechter als jede andere Klassenangabe in einem Klassenmuster.

43. Eine Nachricht ($= oref_0 oref_1 \ldots oref_n$) wird ausgewertet, indem (1) in der APOS-Objektwelt die Methoden mit dem auf diese Nachricht *am besten passenden* Klassenmuster gesucht wird, und dann die Nachricht

    `with` $oref_0 oref_1 \ldots oref_n$ *block*

    ausgeführt wird, wobei *block* das in der Methode referierte Blockobjekt bedeutet.

44. Wird für eine auszuwetende Nachricht *keine passende Methode* gefunden, so wird eine *Method-Not-Found*-`EXCEPTION` ausgelöst.

Anhang B

Glossar.

Aggregatwechsel.

A. bezeichnet das häufige Phänomen, daß in der zeitlich-logischen Aufeinanderfolge der „Aggregatzustände" des (wesentlich psycho-internen) Materials des Komponierens, resp. im Datenfluß der auf einander aufbauenden Bearbeitungsvorgänge, *ereignisbasierte* und *prozeßbasierte* Darstellungs- und Existenzformen regelmäßig alternieren.

Ausgangskontext.

A. bezeichnet in unserem Modell des Komponierens den Teil des Hintergrundes, der die seitens des Rezipienten *vorauszusetzende* Sprachkompetenz, die vorliegende Materialerfahrung, die bei ihm möglichen Konnotationen etc. beinhaltet.

Auswertungslauf.

Ein A. ist ein Vorgang der Transformation eines *implizit* gegebenen Regelwerkes in eine *explizite*, meist diachrone Darstellung.

Zentrale Funktion eines A.es ist i.A., daß die Regeldefinitionen seitens des Komponisten mit den „physikalischen" Randbedingungen des Kontextes „verrechnet" werden. Meist ermöglicht erst das Ergebnis eines A.es die Verifikation, ob die implizite Regeldarstellung tatsächlich die vom Komponisten intendierte Struktur oder Wirkung beschreibt. So bewirkt ein A. normalerweise einen Erkenntnisgewinn und evtl. eine *Revision* des in ihm ausgewerteten Bestimmungsnetzwerkes.

Autarke Quasi-Physikalische Skalare.

Ein A.Q.P.S. ist ein vom Komponisten eingeständig (meist werkspezifisch) definierter skalarer (oder n-dimensionaler) Wertebereich, für den zusätzlich beliebig komplexe oder abstruse algebraische Operationen gesetzt werden können.

Durch diese Operationen kann verhält sich dieses Skalar „quasi-physikalisch". Seine Übertragung auf real physikalische Größen muß jedoch *explizit* definiert werden und ist i.A. in unterschiedlichen Interpretations- und Realisationskontexten durchaus unterschiedlich.

Beispiel:

Ich kann durchaus ein „Maß an Regularität" von fünf Stufen definieren, diese fünf Stufen mit den Zahlen $1, 2, 3, 4, 5$ bezeichnen, eine kanonische Bijektion in die (an sich *zufälligerweise*) gleichnamigen ganzen Zahlen konstruieren (genannt „Betragsfunktion") und dann eine „Addition" nennbare Verknüpfung definieren, welche qua jener kanonischen Abbbildung und ihrer Umkehrung als „Additions"-Ergebnis den *Mittelwert*, das *Maximum*, die *Modulo-Addition* oder irgend etwas anderes als Resultat definiert.

Die so aufgebaute Struktur nennen wir „*Interne Semantik*" des A.Q.P.S.

Die *Externe Semantik* eines A.Q.P.S. ergibt sich jeweils anders bei jeder Übersetzung seiner Werte in einem strukturgenerierenden Kontext.

Unser „Regularitäts-Skalar" könnte z.B. bei serieller Umsetzung einen Wahrscheinlichkeitswert steuern, der bestimmt, ob und wann in einer an sich zeilenweise gelesenen Indexmatrix ein Zeilensprung auftritt.

In einem voltage-control-Kontext könnte unser Skalar in den Abschwächungsfaktor eines Rauschsignals übersetzt werden, welches qua S&H eine somit mehr oder weniger weit ausschlagende Zufallsfunktion generiert.

Bestimmungsraum, Bestimmung, Zeitplan, Werkfaser.

In unserem (stark vereinfachenden) Modell des Kompositionsprozesses gehen wir aus von Einzelbestimmungen und Bestimmungen, welche durch Verknüpfen anderer Bestimmungen komplex zusammengesetzt sind.

Bestimmungen, welche die Strukturierung von Teilen der ablaufenden Werkzeit bewirken, nennen wir *Zeitpläne*.

Bestimmungen, welche die Ebene der Produktion selbst zum Gegenstand haben (z.B. Entscheidungen über anzuwendende Satztechniken, über das Aufschieben oder Vorziehen von Entscheidungen im Arbeitsverlauf, über die zur Fällung einer noch offenen Entscheidung heranzuziehenden Kriterien etc.) nennen wir *Meta-Bestimmungen*.

Bestimmungen, welche verschiedene Bestimmungen mit einem Zeitplan verknüpfen, nennen wir *Werkfaser* und halten sie für die eigentlichen „building blocks" des musikalsichen Kunstwerkes während seiner psycho-internen Entstehung.

Diese Werkfasern sind nun plaziert in einem n-dimensionalen *Bestimmungsraum*. Dessen *Achsen* zerfallen in drei(3) Klassen :

- Die ablaufende Realzeit des geplanten (entstehenden / letztlich vollendeten) Werkes, auch *Werkzeit* genannt, von uns mit $\hat{T}$ bezeichnet.
- Die ablaufende Realzeit des Arbeitsvorganges selber, gegliedert in *Abeitsphasen*, von uns mit T_A bezeichnet.
- Die *Parameter-Achsen* des Werkes, welche durchaus stilabhängig, ja werkspezifisch jedesmal anders definiert werden können.

Diachrone Darstellung.

Eine jede Darstellung musikalischer Strukturen und Materialien, welche (mit einer ihrer Achsen) dem *linearen Verlauf der Zeit* explizit folgt.

empirisch realistisch und transzendental idealistisch.

KANTs Kurzformel für seine (auch unserer Modellbildung zu Grunde liegende) Ausgangs-Hypothese, daß zwar einerseits die Welt (als „Ding an sich") durchaus ohne uns existent ist (was immer das heiße !?) und „ihren eigenen Regeln" folgt, – aber als solche uns niemals zugänglich ist, da jede Erfahrung (als einziger Möglichkeit, mit dieser Welt in Kontakt zu treten) vom Erfahrenden und vom Erfahrenen gleichermaßen und *unauflösbar* bestimmt ein *wesentlich Drittes* ist.

Wir unterscheiden in diesem Sinne Ding-Lichkeit als unerfahrbare, menschenunabhängige Welt, und Wirk-Lichkeit als *psycho-interne Modellbildung*[1], welche die einzig für uns relevante Welt jedesmal und in jedem Menschen neu konstruiert.

Ergonomischer Informationsgehalt.

Ein Maß für den Aufwand, aus einer gegebenen Darstellung eine bestimmte Art von (in ihr rein „physikalisch" zweifellos enthaltener) Information zu extrahieren und dem Menschen erfahrbar zu machen.

Beispiel: Vergleicht man ein Partitur und (vollständiges, wörtliches) zwei-systemiges Particell eines Bläserquintettes, so sind die auftretenden Akkordformen in diesem sofort (auf einen Blick!) abzulesen, – der Stimmverlauf des Hornes jedoch einfacher in jener.

Fehlgebrauch, Gezielter.

Das Vorgehen, ein Gerät oder Verfahren für Anwendungszwecke oder auf Materialien zu verwenden, für die es eigentlich nicht entwickelt wurde, – so lange als das entstehende Resultat ästhetisch sinnvoll und anders auch nicht zu erzielen ist.

G.F. ist in der Geschichte der elektronischen Musik schon fast die Regel : Die ersten Experimente elektronischer Klangsynthese z.B. wurden mit Funktionsgeneratoren durchgeführt, die industriellen Test- und Materialprüfzwecken dienen sollten.

Hinter-, Mittel- und Vordergrund.

Oberste Gliederungsstufe in unserem Modell des Komponierens :

Der *Hintergrund* beinhaltet Ausgangs- und Zielkontext, sowie das *Thema*, also die Festlegung des zentralen Arbeitsbereiches (was u.U. sogar so abstrakt sein kann wie ein anzuwendendes satztechnisches Verfahren).

Der *Vordergrund* beinhaltet das angestrebte Ergebnis in (scheinbar) mitteilbar fixierter Form (fertige Schallaufzeichnung, veröffentlichter Notentext etc.)

Die Definition des Vordergrundes ist tatsächlich lediglich produktions-praktisch / wirtschaftlich möglich, – substantiell hingegen höchst kritisch, da das „transzendentalanalytisch dingliche Werk" in der Rücktransformation der Vordergrundstruktur in psychointernen Mittelgrund existiert.

Den *Mittelgrund* bilden (1) alle psycho-internen Objekte, welche während des Produktionsprozesses das eigentliche Material des kompositorischen Denkens bilden, – Motive, Pläne, Strukturformeln, Umgangsregeln, Satztechniken, etc. – und in dem sich alle künstlerischen Entscheidungen abspielen, was insbesondere bei der Planung des Einsatzes des Digitalrechners zu berücksichtigen ist, – aber auch (2) die im Rezeptionsprozeß (re-)konstruierte (psycho-interne) Substanz des Werkes.

[1] *Diese* ist einziger Ort der Gültigkeit von „Ursache und *Wirk*ung", – deshalb diese Nomenklatur.

Ideologie-Neutralität.

I.-N. ist eine grundlegende und unverzichtbare Forderung an Integrierende Digitale Arbeitsumgebungen, und an überhaupt alle möglichen Formen von automatisierter Unterstüzung, daß sie sich bezogen auf die grundlegenden ästhetischen Ausrichtungen der sie benutzenden Musiker möglichst neutral verhalten und die verschiedensten Stile, Hintergründe, Methoden und Absichten versuchen, gleichermaßen gut zu unterstützen.

Innere Sprache.

I.S. ist ein psycho-internes System von operablen Identitäten (= autarken Symbolen) und Verknüpfungsregeln, in welchem (1) wahre und falsche Sätze gebildet werden können, und (2) aus vorhandenen Sätzen weitere abgeleitet werden können.

Das Wirken einer I.S. braucht keinesfalls bewußt zu geschehen.

Jeder Mensch verfügt über ein ganzes Ensemble I.S.n.

Eine I.S. kann innerhalb des Menschen auf ganz unterschiedlichen Seins-Ebenen angesiedelt sein, – so hat z.B. jeder Pianist eine I.S., deren operable Einheiten die Stellungen von Hand und Finger sind und deren Verknüpfungsregeln äußerst genau definierbar sind, ohne normalerweise jemals bewußt wahrgenommen oder angewandt zu werden.

In diesem Sinne (was allerdings lediglich ein Nomenklaturphänomen ist!) wäre menschliches Denken *immer sprachgebunden* und verfügten Digitalrechner über eine *psycho-interne Realität* !

Konjugierbarkeit, Konjugation, Konjugieren von Algorithmen.

Aus technischen, naheliegenden Gründen der Nomenklatur *Ersatzworte* für die Begriffe *Komponierbarkeit* oder *Kompositionalität*, *Komposition* oder *Komponieren von Algorithmen* aus der Informatik.

Lizenz-Problem, Punktueller Durchgriff.

In unserem Zusammenhang besteht das L.P. darin, daß in einer ganzheitlich generierten, durch Auswertung impliziter Regeln entstandenen (meist diachronen) Struktur, an *ad hoc* vom Komponisten bestimmten Positionen Veränderungen vorgenommen werden, welche das eigentlich zugrunde liegende Generator-Prinzip punktuell negieren[2].

Diese Aktion des Komponisten nennen wir „punktuellen Durchgriff".

Das L.P. Problem wird beim Einsatz des Digitalrechners besonders in Verbindung mit Versionsverwaltung und redo-Mechanismen relevant.

Modellbildung.

Im *weiteren* Sinne aufgrund der „empirisch realistischen und transzendental idealistischen" Hypothese die einzige Möglichkeit von Erfahrung : Jede Wahrnehmung trägt sofort (darin besteht sie!) zur Modifikation unseres persönlichen, psycho-internen Modelles von der Welt bei.

[2]Es gibt in der Musikgeschichte auch einen allgemeineren Begriff von Lizenz, welcher das grundsätzliche Abweichen von angeblich akademisch sanktionierten Satzregeln beinhaltet („con alcune lizence"). Diesen Begriff meinen wir hier nicht!

Im *engeren* Sinne der einzig mögliche Weg von Erkenntnis : Ein vermuteter Zusammenhang innerhalb eines Teilbereiches der (vermuteten!) Dinglichkeit wird psycho-intern modelliert. Dieses Modell wird danach verifiziert, indem es auf das eine, große „Modell Welt" (psycho-intern aufgespannt durch die Erfahrung) angewandt wird und die Ergebnisse dieser Anwendung mit den Vorhersagen des Modelles verglichen werden.

post-seriell.

Der Begriff P. bezeichnet sowohl einen kompositorischen Stil und eine Auffassung von Musik, als auch die damit übereinstimmenden Verfahren.

Letztere sind am einfachsten erklärt und führen auch zu der Namensgebung : Nach (= „post") den Erfahrungen des stengen Serialismus war es möglich, *verschiedenste heterogene Medien* kompositorisch auf allen Ebenen zu verwenden, unterschiedlichste *Materialien* (ohne Ähnlichkeiten vorfinden zu müssen) frei zu verknüpfen und sie mit nämlichen handwerklichen Methoden zu behandeln.

Beispiele :

- Zeitungsseiten, Texte der Artikel oder Proportionen des Layouts können zu Formplänen von Werken werden;
- Gedichte, Kursbücher oder politische Pamphlete können permutiert, dekomponiert, rhythmisiert oder abstrahiert werden, – also behandelt wie vormals eine Dauernreihe (und davor eine Zwölftonreihe und davor ein Thema);
- Sprachformate und Spezfikationstechniken sind a priori gleichberechtigt : genaue Auskomponierung, Aleatorik, Happening oder Gedankenexperiment, Bühnenhandlung oder lightshow, – keine Form von Medium oder Verwendung eines Mediums wird ausgeschlossen[3].

Als tertium comparationis können dabei dienen der Zahlbegriff, der Zeitbegriff, semantische Definitionen (Metaphern-Setzungen) etc.

Besonders die musikalische *Analyse* profitiert von den post-seriellen Techniken, als weitgehend „vorurteilsfreie", d.h. respektlos angewendete phänomenologische Verfahren durchaus neue Erkenntnisse auf struktureller Ebene auch über „ältere" Musik liefern können.

Rezeptionsstandpunkt.

Mit R. bezeichnen wir den Teil der Ausgangsbedingungen im Rezipienten, der sich mit jedem Male der Rezeption des nämlichen Werkes durchaus *spontan ändern* kann, – aber auch durch die bisherige Rezeptionsgeschichte dieses und anderer Werke beeinflußt wird.

Konkretes Beispiel (und mitnichten nur Metapher!) ist der räumliche Abstand, den man beim Betrachten eines Gemäldes einnimmt und dessen Änderung i.A. zu völlig unterschiedlichen Wahrnehmungen führt.

Hufschmidt nennt den R. „die Brille", durch die man ein Werk rezipiert, und empfiehlt grundsätzlich, diese öfters zu wechseln.

[3]Das neuerdings als so zeitgemäß angepriesenen „multimedia-Denken" ist in der Tat eine bereits durchaus patinierte Erfindung der 50er und 60er Jahre !

Höchst Sublimes fließt in die Bestimmung des R. ein, aber auch das Banalste : Die köstlichste Tristan-Steigerung kann zur Qual werden, wenn man sich vorher nicht um seine Blase kümmert.

Selbstidentität.

Innerhalb unserer „empirisch realistischen und transzendental idealistischen" Modellbildung ist S. immer etwas *Behauptetes*!

Die Identität mit sich selbst eines (vermuteten) Dinges wird als *Arbeitshypothese explizit* jeweils eigens und neu vorausgesetzt, um an diesem hypothetischen Selbstidentischen (Unwandelbaren) die unterschiedlichsten Erscheinungsformen der Wirklichkeit in Beziehung zu setzen.

Z.B. ist „BEETHOVENs Fünfte Sinfonie" ein derart konstruiertes, „fiktives" selbstidentisches Ding.

Symbol, symbolisch.

S. bedeutet hier fast das Gegenteil des oft benutzten, alltäglichen Verständnisses. Unser Gebrauch von S. bedeutet *nicht* „Zeichen *für* etwas anderes" zu sein, sondern im Gegenteil *nur selbstidentische* Einheit unserer Vorstellung, sozusagen ein Denk-Atom, – ein psycho-internes Objekt, mit dem man nach bestimmten Regeln halt spielen (verknüpfen etc.) kann.

Dieser Begriff von S./s. ist dem des „Symbols" in „symbolischen Programmiersprachen" wie LISP o.ä. verwandt.

Abgesehen von der reinen *Tatsache* des „geregelten Spielens" können wir über Gehalt und Konnotation der (psycho-internen) Symbole nichts wissen.

Evtl. kennen wir die Spielregeln der Verknüpfbarkeit , – doch selbst das ist nicht zwingend.

Synchrone Simulation.

Eine Unterart des rechnergestützt automatischen (oder auch manuellen, psycho-internen) *Auswertungslaufes*, der in seinen logischen Schritten den zeitlichen Schritten des zu generierenden (oder eines eingegebenen !) Materials folgt.

Eine Implementierung als s.S. ist dem menschlichen Denken meist recht adäquat und normalerweise leichter zu debuggen als out-of-time operierende Struktursynthesen, da ihre Logik der vom Komponisten oft geübten, anschaulichen Vorstellung des zeitlichen Nacheinander folgt.

transzendentale Geschiedenheit.

Kurzformel für die Tatsachen, daß (1) die „Welt an sich" unserer Erfahrung *ihrem Wesen nach* immer unzugänglich bleiben muß, und daß (2) die inneren Vorstellungswelten, konkreten Empfindungen und diversen Inneren Sprachen anderer Menschen niemals uns direkt vermittelbar sein können (= die strukturelle Einsamkeit des menschlichen Bewußtseins).

Utilitarisches Wahrheitsindiz.

Kurzformel für die Tatsachen, daß (nachdem erkannt ist, daß die unterschiedlichsten Modellbildungen der „Welt an sich" gleichberechtigt adäquat sein können) nur noch die *Nützlichkeit* eines Modelles einen Art von „Wahrheitsbegriff" konstituieren kann, – genauer: das einzige Kriterium für einen (vielleicht als solchen gar nicht benötigten oder wünschenswerten) Wahrheitsbegriff ist die *Brauchbarkeit* der von einem bestimmten Weltmodell gelieferten *Vorhersagen*.

Das u.W. belegt die Unbrauchbarkeit der reinen Vernunft zur Fundierung von Ethiken und hatte so historisch die übelsten Konsequenzen.

viriginales Hören.

Das erste (relevante) Erleben eines Kunstwerkes, welches unsere innere Vorstellungswelt (zumeist) so einschneidend beeinflußt, wie es kein weiteres Hören desselben Werkes je wieder wird tun können.

NB muß das virginale Hören nicht unbedingt das physisch erste Mal sein, sondern das erste Mal, bei welchem wir tatsächlich ergriffen werden.

Werkfaser-Polyphonie.

W.P. bezeichnet die Eigenschaft der Mittelgrund-Strukturen (fast aller anspruchsvolleren Werke), daß verschiedene Werkfasern auf verschiedenen Bestimmungsebenen die nämlichen Intervalle der ablaufenden Werkzeit mit *unterschiedlichen, unabhängigen und parallelen Zeitplänen* bestimmen.

Zielkontext.

Z. ist ein wichtiger Teil der von uns als *Hintergrund* zusammengefassten Werksubstanz, beinhaltend die formalen, praktischen, wirtschaftlichen, arbeitstechnischen und -zeitplanerischen, kulturpolitischen etc. Überlegungen, welche in die Randbedingungen eines Werkes und somit in seine Mittelgrundstruktur einfließen.

Literaturverzeichnis

Barlow, C. (1980). Cogluotobüsişdletmesi.

Kant, I. (1781). *Kritik der reinen Vernunft*.

König, G. M. (1968). *PR I*.

Lem, S. (1971). *Dzienniki Gwiadowe*. Czytelnik, Warszawa.

Lepper, M. A. G. (unveröffentlicht). Bedeutung und wahrnehmung elektronisch erzeugter klänge im spannungsfeld von konnotat und denotat.

Reith, D. (1980). Nested loops i.

Sayers, D. (1930). *Whose Body?* Victor Gollancz Ltd., London.

Scaletti, C. (1987). sunsurgeautomata.

Ungeheuer, E. (1993). Musik als wahrnehmungsexperiment.

Zacher, G. (1993). *Bach gegen seine Liebhaber verteidigt*. Number 79/80 in Musik-Konzepte. edition text + kritik, München.

Bildnachweise

Das Photo auf Seite 115 stammt von Dipl-Ing. Roland Masslich, Essen.

Die Zeichnung des PFG auf Seite 174 ist von Dipl-Ing. Gerhard Kümmel, Essen.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z