# Computermusik Vorlesung WS 18, 20

Johannes Waldmann

26. Januar 2021

## 1 Organisatorisches

**Die LV insgesamt**

- jede Woche eine Vorlesung, eine Übung
- Prüfungszulassung: regelmäßiges und erfolgreiches Bearbeiten von Übungsaufgaben (teilw. autotool)
- Prüfung:
  - (gemeinsames) Abschluß-Konzert
  - (individuelle) Dokumentation (*welche* kreative Idee wurde *wie* realisiert)

**Übungen**

- Sie benutzen die Rechner im Pool (Z430) mit dort installierter Software. — *Kopfhörer mitbringen!*

  Es ist zu empfehlen, die gleiche Software auch auf Ihren privaten Rechnern zu installieren, damit Sie selbst experimentieren und Hausaufgaben erledigen können.
- wir verwenden ausschließlich *freie* Software (Definition: siehe `https://www.gnu.org/philosophy/free-software-intro.html`) (Debian-Pakete oder selbst kompiliert). Alles andere wäre unwissenschaftlich — weil man es eben nicht analysieren und ändern kann.

**Übungs-Aufgaben**

Das sind Beispiele für Tätigkeiten, die in dieser LV (und in allen anderen) immer wieder vorkommen: nicht nur Software bedienen und Knöpfchen drehen, sondern auch:

Analysieren, Rechnen, Recherchieren, historisch einordnen, Programmieren (Synthetisieren).

1. ausprobieren: Hydrogen (Drum-Sequencer) → Rakarrack (Effekt-Prozessor)

   Audio-Routing mit qjackctl

2. Finden Sie die von Hydrogen benutzte Audio-Datei für *TR808 Emulation Kit, Kick Long*

   anhören mit `vlc`,

   konvertieren Sie mit `sox` in `wav`-Format, (Hinweis: `man sox`),

   betrachten Sie Dateiinhalt (Amplituden-Verlauf) mit

   ```
   gnuplot -persist -e "plot 'kick.wav' binary format='%int32' using 0:1
   ```

   Bestimmen Sie mittels dieses Bildes die Grundfrequenz der Schwingung. Welche weitere Information ist dazu nötig, woher bekommen Sie diese?

**Hausaufgaben (bis zur Ü in KW 43)**

1. betrachten Sie Dateiinhalt mit

   ```
   od -cx kick.wav | less
   ```

   Wo endet der Header (wo steht das erste Datenbyte)?

   Suchen Sie die offizielle WAV-Spezifikation, bestimmen Sie deren bibliografische Daten (Autor/Gremium, Ort, Jahr)

   Erzeugen Sie durch ein selbstgeschriebenes Programm (Sprache beliebig) eine wav-Datei, die einen (kurzen) Sinus-, Dreieck-, oder Rechteckton enthält,

   ansehen mit `gnuplot`, abspielen mit `vlc`,

   verwenden Sie das als Sample in Hydrogen.

2. Wie sah diese Maschine (TR808) aus?

   Welche Band führt diese Maschine im Namen? (Hinweis: `http://www.vintagesynth.com/`)

   Kann Hydrogen alle dort angegebenen Eigenschaften des Originals simulieren?

   beschreiben Sie Struktur und (einige) Elemente von *Ritchie Hawtin: Minus Orange 1*, *Aphex Twin: Flaphead* o.ä., simulieren Sie mit Hydrogen und Rakarrack.

# 2 Einleitung

**Definition Computermusik**

- *Computermusik* fassen wir auf als:

  Analyse und Synthese von Musik mithilfe der Informatik (Algorithmen, Software)

  (A: hören, verstehen; S: komponieren, aufführen)

- beruht auf Modellen aus der Musiktheorie, z.B. für
  - Erzeugung von Klängen in physikalischen Systemen,
  - das Tonmaterial:
    Tonhöhe, Konsonanz und Dissonanz, Akkorde, Skalen
  - die zeitliche Anordnung des Materials:
    Rhythmen, Melodien, Kadenzen, Kontrapunkt

**Definition Musik**

- die Kunst der zeitlichen Anordnung von Klängen.

  (Edgar Varese 1883–1965: *I call it organised sound*)

- „Kunst“ bedeutet: der Autor (Komponist, Interpret) will im Hörer Empfindungen hervorrufen

- das geht sowohl sehr direkt, Beispiele:
  - Tonreihe aufsteigend: Frohsinn, absteigend: Trübsal
  - Dissonanz $\Rightarrow$ Spannung, Unruhe; Konsonanz $\Rightarrow$ Auflösung, Ruhe

als auch indirekt, Beispiele:

- Zitat (Parodie) von Elementen andere Musikwerke:
  Anerkennung (Daft Punk ⇒ Giorgio Moroder), Aneignung (F.S.K.), Ablehnung (Punk ⇒ Prog Rock).

**Definition Pop(uläre) Musik**

- die mechanische (Aufnahme und) Vervielfältigung von Audiosignalen (seit ca. 1920, Grammophon) trennt die *Aufführung* vom ihrem *Resultat* (dem Klang)

  (Elijah Wald, An Alternative History of American Popular Music, Oxford Univ. Press, 2009)

- dadurch entsteht Popmusik, das ist etwas Neuartiges
  - statt Komposition (Klassik) oder Improvisation (Jazz): *Produktion* des Klangs in einem Studio
  - rezipiert wird nicht nur der Klang, sondern unzählige *Nebenprodukte*, insb. Bilder (z.B. Schallplattenhüllen)
    die Bedeutung wird daraus vom *Fan* konstruiert

  (Diederich Diederichsen, Über Popmusik, Kiepenheuer, 2014)

**Hörbeispiele**

- Daft Punk (Guy-Manuel de Homem-Christo und Thomas Bangalter): *Giorgio by Moroder* (LP Random Access Memories, 2013)
- Donna Summer: *I Feel Love* (Single, 1977) Produzent: G. Moroder
- Kraftwerk (Ralf Hütter und Florian Schneider): *Autobahn* (LP 1974), aufgenommen im Studio Conny Plank
- Neu! (Michael Rother und Klaus Dinger): *Hallogallo* (1972), Produzent: Conny Plank
- Stereolab (Tim Gaine, Laetitia Sadier u.a.): *Jenny Ondioline* (1993)
- Grandmaster Flash (Joseph Sadler) *The Message*(1982)
- Big Black (Steve Albini u.a.): *Kerosene* (1986)

**Plan unserer Vorlesung**

- KW 42: Klang-Erzeugung (Physik der Musikinstrumente)
- KW 43: Klang-Analyse (Spektren) und Klangveränderung
- KW 44: Analog-Synthesizer (und ihre Simulation)
- KW 45: Algebraische Beschreibung von Klängen (csound-expression)
- KW 46: Töne, Skalen, Konsonanzen, Akkorde, Kadenzen
- KW 47: Algebra Of Music (haskore), Notensatz (lilypond)
- KW 48: A Pattern Language (tidal-cycles, supercollider)

ab hier Reihenfolge noch unklar:

- KW 49: Rhythmus (breakbeat science)
- KW 50: Mathematische Musiktheorie
- KW 51: Musikgeschichte, Zwischenstand Projekte
- KW 54: Audio-Analyse
- KW 55: Digital Audio Workstations (ardour, sooperlooper)
- KW 56: algorithmische Mechanik
- KW 57: Zusammenfassung, Abschluß Projekte

# 3 Geräusch und Klang

**Begriffe**

- Geräusch:
  - erzeugt durch Schwingungen eines physikalischen Systems (z.B. Musikinstrument)
  - übertragen durch Druckschwankungen in einem Medium (z.B. Luft), durch Ohr wahrnehmbar

- Klang: . . . durch *periodische* Schwingungen . . .
- virtuelle (elektronische) Instrumente
  - simulieren den physikalischen Vorgang
  - oder speichern nur dessen Amplitudenverlauf
- Unterschied zu automatischem Spiel reeller Instrumente

**Modell einer periodischen Schwingung**

- Modell:
  - ein Körper mit Masse $m$ und Ruhelage $0$
    bewegt sich auf einer Geraden $g$,
    d.h., hat zum Zeitpunkt $t$ die Koordinate $y(t)$
  - die Rückstellkraft (bei Pendel: durch Schwerkraft, bei schwingender Saite: durch Elastizität) ist $F = -k \cdot y$.
    Notation: das ist eine Gl. zw. Funktionen (der Zeit)!
- mathematische Beschreibung
  - Geschwindigkeit $v = y'$, Beschleunigung $a = v' = y''$
  - nach Ansatz ist $a = F/m = -(k/m) \cdot y$
  - $y$ ist Lsg. der Differentialgleichung $-(k/m)y = y''$

**Numerische Näherungslösung der Dgl.**

- gegeben $k, m$, bestimme Funktion $y$ mit $-(k/m)y = y''$
- numerische Näherungslösung durch Simulation:
  ersetze Differentialgl. durch Differenzengl.
  wähle $y_0$ (initiale Auslenkung), $\Delta > 0$ (Zeitschritt),
  bestimme Folgen $y_0, y_1, \ldots, v_0 = 0, v_1, \ldots, a_0, a_1, \ldots$
  mit $a_i = -(k/m)y_i$, $v_{i+1} = v_i + \Delta a_i$, $y_{i+1} = y_i + \Delta v_i$
- ausrechnen in Haskell, anzeigen mit `https://hackage.haskell.org/package/gnuplot-0.5.5.3/docs/Graphics-Gnuplot-Simple.html`
- genaueres in VL Numerik,
  z.B.: *Stabilität* besser, wenn $y_{i+1} = y_i + \Delta v_{i+1}$

**Exakte Lösung der Dgl.**

- gegeben $k, m$, bestimme Funktion $y$ mit $-(k/m)y = y''$
- genaueres siehe VL Analysis, z.B. Ansatz von $y$ als
  - Potenzreihe mit unbestimmten Koeffizienten
  - Linearkombination von Basisfkt. mit unbest. Koeff.
- wenn man Glück hat, oder die numerische Lösung gesehen hat:
  Ansatz $y(t) = \cos(f \cdot t)$
- wir erhalten die *reine harmonische Schwingung*

**Schwingung einer Saite**

- bisher: (harmonische) Schwingung *eines* Massepunktes (mit linearer Rückstellkraft)
  jetzt: reale Saite aus vielen Punkten (Segmenten)
- $u$ : Ort $\times$ Zeit $\to$ Auslenkung
  $d^2u/(dt)^2 = c \cdot d^2u/(dx)^2$, $u(0,t) = u(1,t) = 0$, $u(x,0) = 0$
- Ansatz $u(x,t) = f(x) \cdot g(t)$
  es gibt mehrere Lösungen
- Dgl. ist linear: jede Summe von Lösungen ist Lösung
- Hermann Helmholtz: Vorl. über die mathematischen Prinzpien der Akustik, Leipzig 1898 `https://archive.org/details/vorlesungenber03helmuoft`

**Anpassung und Anwendung**

- diese Modell ist Energie-erhaltend
  tatsächlich wird aber Energie abgegeben (1.über das Medium an den Sensor, 2. durch Reibung im schwingenden Körper als Wärme an die Umgebung)
- Modellierung der *Dämpfung* z.B. durch Reibungskraft proportional zu Geschwindigkeit $F_R = r \cdot v = r \cdot y'$
  Aufstellen und Simulation der Dgl. in Übung.

- mit diesem Modell können wir beschreiben:
  - Klang einer Saite (Gitarre, Klavier, Cembalo)
    (nicht Geige)
  - Klang eines Trommelfells (Fußtrommel, nicht Snare)

**Weitere period. Schwingungen f. Instrumente**

- Wirkung der Dämpfung kann durch regelmäßige Energiezufuhr ausgeschaltet werden ($\Rightarrow$ angeregte Schwingung) z.B. das Anstoßen einer Schaukel
- Geige:

  Bewegung des Bogens führt der Saite Energie zu

  regelmäßige Unterbrechung durch Kontaktverlust Bogen–Saite bei zu starker Auslenkung
- Blasinstrumente:

  Anblasen führt der schwingenden Luftmenge Energie zu

  regelmäßige Unterbrechung durch Blatt (Oboe, Saxofon), Lippen (Trompete) oder Luftsäule selbst (Orgel, Flöte)

**Geräusch-Instrumente**

- nichtperiodisches Verhalten kann erzeugt werden durch
  - nichtperiodische Schwingung eines phys. Systems
    z.B. Doppel-Pendel, Mehr-Körper-System
    keine direkte Anwendung als Instrument bekannt,
    Simulation evtl. für virtuelle Instrumente nützlich
  - Überlagerung (fast gleichzeitiger Ablauf) sehr vieler unterschiedlicher periodischer Schwingungen
    für zahlreiche (Rhythmus)-Instrumente benutzt, z.B.
    - Maracas (Rumba-Kugel): enthalten viele kleine harte Klangkörper, die aneinanderstoßen
    - Snare (kleine Trommel): mehrere Federn, die gegen Fell der Unterseite schlagen (schnarren)

**Chaotische Schwingungen**

- wenn man das wirklich nur simulieren möchte, nicht mechanisch realisieren,
- dann kann man auch Systeme *ohne* mechanisches Äquivalent betrachten
- Bsp: die Iteration der Funktion

  $f : [0,1] \to [0,1] : x \mapsto 4 \cdot (x - 1/2)^2$

  zeigt aperiodisches (chaotisches) Verhalten
- Ü: Wertefolge ausrechnen, ansehen, anhören

**Übungen**

- Wie wird Musikgeschichte zitiert (im Klang und) im Text von: DJ Hell: *Electronic Germany* (2009)

  Wer singt auf *U Can Dance* des gleichen Albums? War früher (viel früher) in welcher Band? Wer hat dort anfangs elektronische Instrumente gespielt? Danach welchen Musikstil erfunden?

  weitere Beispiele für Musikzitate suchen, genau beschreiben, was zitiert wird, wie groß der Abstand ist (zeitlich, inhaltlich) und diskutieren, warum.
- Schwingungen (periodische, gedämpfte, chaotische)

  simulieren, Resultate ansehen, anhören (einzeln, als Drumkit in Hydrogen)

  Bsp.

```
import Graphics.Gnuplot.Simple
ys = map fst
    $ let d= 0.1
      in iterate (\(y,v) ->
         let {a = negate y;vn =v+d*a;yn=y+d*vn}
         in (yn,vn)) (1,0)
plotList[] $ take 100 ys
```

  zum Schreiben vgl `https://github.com/BartMassey/wave/blob/master/writetest.hs`

- Schwingung einer Saite simulieren: Differenzengl. aus Differentialgl. ableiten, Verlauf zeichnen mit gnuplot (`plotMesh3D`)

  verschieden Lösungen zeigen

- chaotischer (pseudo-zufälliger) Amplitudenverlauf durch Bit-Schieberegister.

  Vgl. *chip tunes*, *demo scene*: Klänge und Geräusche durch möglichst kurzen Maschinencode erzeugen

**Klangveränderung durch Filter**

- ein Filter ist ein Operator von $(\Omega \to \mathbb{R})$ nach $(\Omega \to \mathbb{R})$

  (eine Funktion der Zeit auf eine Funktion der Zeit,

  d.h., Filter ist Funktion zweiter Ordnung)

- Bsp: der Operator $\mathsf{scale}_s : g \mapsto (x \mapsto s \cdot g(x))$

- Bsp: der Operator $\mathsf{shift}_t : g \mapsto (x \mapsto g(x - t))$

  akustisch ist das ein *Echo*. Mehrere Echos ergeben *Hall*.

  Realisierungen:

  - Tonband-Schleife
  - Federhallstrecke

  typisch für: Gitarrenklang in Surf-Musik (Bsp: Dick Dale), Gesamtklang im (Dub) Reggae (Bsp: Lee Perry)

**Klangveränderung durch Filter**

- Operator $F$ ist *linear* (L), wenn $\forall a, b \in \mathbb{R}, g, h \in (\Omega \to \mathbb{R}) : F(a \cdot g + b \cdot h) = a \cdot F(g) + b \cdot F(h)$
- $F$ ist *zeit-invariant* (TI), wenn $\forall t \in \mathbb{R} : \mathsf{shift}_t \circ F = F \circ \mathsf{shift}_t$
- Satz: jeder LTI-Filter kann als (Limes einer unendl.) Summe von $\mathsf{shift}$ und $\mathsf{scale}$ dargestellt werden
- Satz: jeder lineare Filter operiert auch linear auf den Fourier-Koeffizienten.

- Folgerung: Obertöne werden geschwächt oder verstärkt, aber niemals „aus dem Nichts“ erzeugt.

  Das begründet den Wunsch nach nichtlinearen Filtern (Verzerrern).

**Filter in der Musik-Praxis (Fender Amp)**

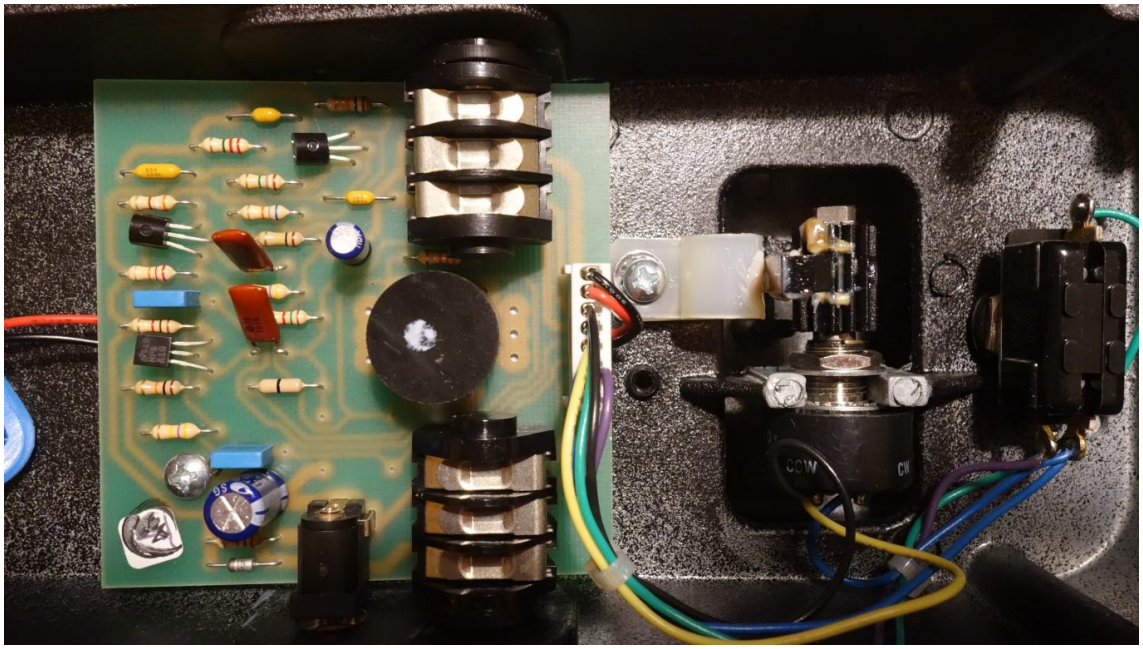

- Lautstärke (Volume): $\mathsf{scale}_s$
- period. Lautstärkeänderung (Tremolo) (Speed, Intensity)

  ist linear, aber nicht zeit-invariant
- Federhall (Reverb): $\sum_{d \in D} \mathsf{shift}_d$, linear, zeit-invariant

Tiefpaß (Bass), Hochpaß (Treble) : später genauer

**Filter in der Musik-Praxis (Wah)**

- Dunlop Crybaby GCB95

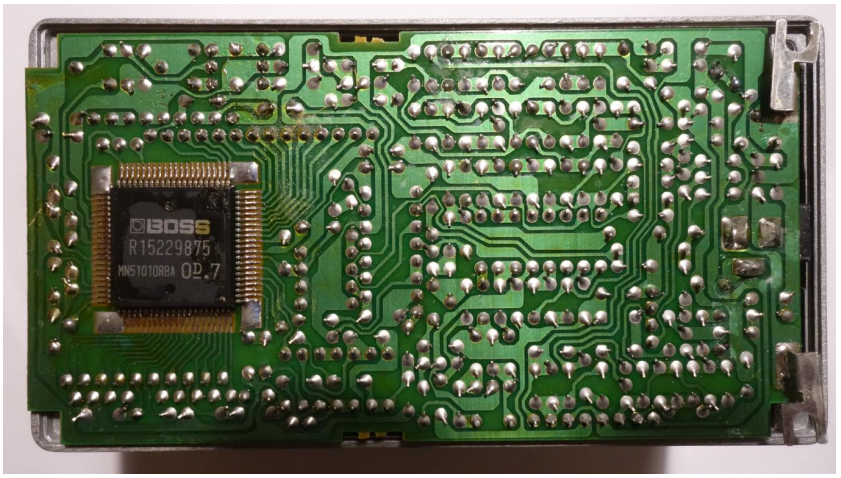


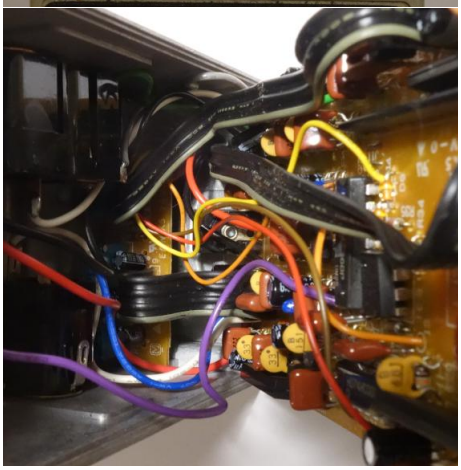

`https://www.electrosmash.com/crybaby-gcb-95`

- Bandpaß mit einstellbarer Resonanz-Frequenz
- Fußwippe (Pedal) → Zahnstange → Dreh-Potentiometer
- vgl. später: spannungsgesteuerte Filter (VCF)

**Filter in der Musik-Praxis (Echo)**

- Boss Digital Delay (DD 3, ab 1986)

- erstes Delay-Fußpedal DD 2, 1983 `https://www.hobby-hour.com/electronics/s/dd2-delay.php`
- Echo-Zeit max. 800 ms. Samplebreite 12 bit. Ü: Taktbreite 12.5 $\mu$s . . . 50 $\mu$s. Wieviel Bit werden gespeichert?
- vgl. später: Chorus (= spannungsgest. Delay), Flanger

**Chorus, Flanger, Phaser**

- Chorus: $f \mapsto f + \mathsf{shift}_d(f)$ mit $d = \epsilon \cdot \sin(\omega t)$ für $\omega$ klein
- Flanger: wie Chorus, aber $d$ kleiner

  ursprünglich realisiert durch zwei Tonbandmaschinen

für $f$, für $\mathsf{shift}_d(f)$, dabei $d$ durch Bremsen des Bandes

bei elektronischer Realisierung:

auch mit Rückführung des Ausgangssignales

- Phaser: $f \mapsto f + \mathsf{Allpass}^k(f)$, dabei $\mathsf{Allpass}(f)$:

  erhält Amplituden, verschiebt Phasen (frequenz-abhängig)

- R. G. Keen: *The technology of Phase Shifters and Flangers* 1999, `http://www.geofex.com/Article_Folders/phasers/phase.html`

**Übungsaufgaben**

- mit Hydrogen und Rakarrack Aspekte des Schlagzeugs (Rhythmus, Sound) nachbauen:

  Vivien Goldman (und New Age Steppers): *Private Armies Dub* (1981)

  (Produzent: Adrian Sherwood, vgl. *Bugaloo* (2003, video)

- Phaser: für eine Sägezahnschwingung $f$: bestimmen Sie Amplitudenverlauf (später: und Spektrum) der Schwingung $f + \mathsf{scale}_{-1}(\mathsf{shift}_d(f))$ abhängig von Parameter $d \in [0, \pi]$.

- Echo, Hall, Flanger selbst implementieren:

  Verzögern und ggf. rückkoppeln

  (WAVE-Datei lesen, bearbeiten, schreiben)

  anwenden auf: Sinus, Rechteck, Rauschen

  Ansatz: `https://gitlab.imn.htwk-leipzig.de/waldmann/cm-ws18/tree/master/kw43`

- (NEU) mit dieser Implementierung vergleichen (Steve Harris) `https://github.com/swh/ladspa/blob/master/phasers_1217.xml` (oder andere Open-Source)

# 4 Klang-Analyse

**Definition, Motivation**

- jede periodische Schwingung kann als gewichtete Summe harmonischer Schwingungen dargestellt werden

  (Jean Fourier, 180?, `http://www-history.mcs.st-and.ac.uk/Biographies/Fourier.html`)

- die Folge dieser Gewichte der Obertöne ist das *Spektrum*, das charaktetisiert die Klangfarbe

- Änderung des Amplitudenverlaufs

  linear (z.B. Filter), nichtlinear (z.B. Verzerrer)

- kann beschrieben werden als Änderung des Spektrums

**Periodische Funktionen**

- für $\Omega = [-\pi, \pi]$ betrachte $\mathsf{P} = \{f \mid f : \Omega \to \mathbb{R}\}$.
- $\mathsf{P}$ ist *Vektorraum* (Addition, Skalierung) und Hilbert-Raum
- *Skalarprodukt* $\langle f, h\rangle := \int_\Omega f(x) \cdot g(x)\, dx$, Norm $|f| = \sqrt{\langle f, f\rangle}$
- $b_1 = 1, b_2 = \sin(x), b_3 = \cos(x), b_4 = \sin(2x), b_5 = \cos(2x), \ldots$
  bilden eine *orthogonale Basis* für $\mathsf{P}$
  nach geeigneter Skalierung sogar ortho*normal*
- jedes $f \in \mathsf{P}$ eindeutig darstellbar als Linearkombination von Basisvektoren $f = \sum_i \langle f, b_i\rangle / |b_i|^2 \cdot b_i$
- weitere Voraussetzungen sind nötig (damit Integrale und Summen existieren), siehe VL Analysis
- numerisch: approximiere Integral durch Summe

**Beispiel: Rechteck-Schwingung**

- $\Omega \to \mathbb{R} : t \mapsto$ if $t < 0$ then $-1$ else if $t = 0$ then $0$ else $1$

  das ist die Signum- (Vorzeichen)-Funktion $\mathrm{sign}$

- $\mathrm{sign}(x) = (4/\pi) \sum_{k \text{ ungerade}} \frac{\sin(kx)}{k}$

  (nur ungerade Oberwellen)

Nebenrechnungen:

- $\cos(kx)$ ist gerade Funktion, $\mathrm{sign}(x)$ ungerade,
  deswegen $\langle \mathrm{sign}(x), \cos(kx) \rangle = 0$
- $\langle \mathrm{sign}(x), \sin(kx) \rangle = 2 \cdot \int_{[0,\pi]} \sin(kx) dx = [-1/k \cdot \cos(kx)]_0^\pi =$
  if $2|k$ then $0$ else $4/k$

**Beispiel: Sägezahn-Schwingung**

- $f : \Omega \to \mathbb{R} : x \mapsto x$
- numerische Bestimmung der Fourier-Koeffizienten

```
let k = 4 ; d = 0.01
in sum $ map (\x -> d * x * sin (k*x))
         [ negate pi, negate pi + d .. pi ]
==> -1.570723326585521
```

- Vermutung $f = -\frac{2}{\pi} \sum_{k \geq 1} \frac{(-1)^k}{k} \sin(kx)$ ( alle Oberwellen )

**Spektren von Audiosignalen**

- Spektrum eines Signals $f$ kann so bestimmt werden:
- teile Signal in Zeit-Intervalle (z.B. $\Delta = 1/10$ s),
  $f_i : [-\Delta, \Delta] \to \mathbb{R} : t \mapsto f(i\Delta + t)$
- wähle Frequenz-Werte $k_1, k_2, \ldots$
- bestimme Koeffizienten der Freq $k_j$ zur Zeit $i\Delta$ als $\langle f_i, k_j \rangle$
- Anzeige z.B. in `vlc`: Audio → Visualisations → Spectrum

- es gibt schnellere Algorithmen

  (diskrete Fourier-Transformation)

- das Ohr bestimmt die Fourier-Koeffizienten durch Resonanz in der Schnecke (Cochlea), Frequenz-Auflösung ist ca. 3 Hz bei 1 kHz

**Programme zur Spektral-Analyse**

- Chris Cannam, Christian Landone, and Mark Sandler: *Sonic Visualiser: An Open Source Application for Viewing, Analysing, and Annotating Music Audio Files,* in Proceedings of the ACM Multimedia 2010 International Conference.

  `https://sonicvisualiser.org/`

- Anwendungsbeispiel:

  Aphex Twin, $\Delta M_i^{-1} = -\alpha \Sigma D_i[\eta] F j_i[\eta - 1] + F\text{ext}_i[\eta^- 1]$,

  Album: Windowlicker, 1999.

  hergestellt mit Metasynth (Eric Wenger, Edward Spiegel, 1999) `http://www.uisoftware.com/MetaSynth/`,

**Spektren von Klängen/Instrumenten**

- harmonische Schwingung: keine Oberwellen,

  kommt in der Natur selten vor und ist für Musikinstrumente auch gar nicht erwünscht:

  Oberwellen ergeben interessantere Klänge,

  die auch variiert werden können

- Bsp: Gitarre: Anschlagen nahe dem Steg: viele Oberwellen, zur Saitenmitte: weniger.

- Bsp. Schlagzeug (Trommel, Tom): Anschlag Mitte/Rand

- Bsp: Orgel: offene und gedackte Pfeifen, siehe dazu

### Aufgaben

- wie unterscheiden sich Spektren der Luftschwingungen in offenen von einseitig geschlossenen Röhren? nach: Alfred Kalähne: *Grundzuge der mathematisch-physikalischen Akustik*, Leipzig 1913, `https://archive.org/details/grundzgedermath01kalgoog`

- Fourier-Koeffizienten einer Rechteck-, Sägezahn-, Dreiecks-Schwingung bestimmen:
  - Skalarprodukte symbolisch oder numerisch bestimmen
  - Amplitudenverlauf in WAVE-Datei schreiben und Spektrum analysieren (`sonic-visualiser`)

# 5 Elektrische Schwingungen (I)

### Plan

- bisher: mechanische Schwingungen
  - Bsp: Massepunkt/Feder,
  - Anwendung: akustische Musikinstrumente
    Bsp: Saiten, Membrane, Luftsäulen
- jetzt: elektrische Schwingungen (und Filter)
  - Bsp: Oszillator (LC), Tiefpaß (RC)
  - Anwendungen:
    - Analog-Synthesizer (Robert Moog 64, Don Buchla 63)
    - Simulation von A.-S. (csound, Barry Vercoe, 1985)
  - Ziele: 1. möglichst exakte Nachbildung (des Akustischen, des Analogen), 2. völlig neuartige Klänge

### Elektrische Schaltungen

- Schaltung: gerichteter Graph,
  - Kanten sind Bauelemente
    - ohne Zustand: Widerstände, Verstärker (Transistor)

    - mit Zustand:
      Kondensator: Ladung, Spule: magnetisches Feld
  - durch jede Kante fließt Strom,
    jeder Knoten hat Potential
  - besondere Knoten: Masse (0), Eingabe, Ausgabe
- Zustandsänderung ist Funktion der Ströme und Spannungen
- Schaltung realisiert einen Operator von (Zeit $\to$ Eingabe) nach (Zeit $\to$ Ausgabe)

**Schaltung – Beispiel Tiefpaß**

- Schaltung: (0,3) node [left]$U_E$ to [short,i=$I$,*-] (1,3) to [R,l=$R$, -*] (4,3) to [C,l=$C$] (4,1) node [ground] (3,3) to [short,-*] (5,3) node [right] $U_A$ ;
- Widerstand: $U_E - U_A = R \cdot I$

  (siehe auch Kraftwerk: *Ohm Sweet Ohm*, 1975)
- Kondensator: $I = C \cdot \dfrac{dU_A}{dt} = C \cdot U'_A$
- Bsp: $U_E(t) = 1\text{V}$, $U_A(0) = 0$ (Kondensator leer)

  $C \cdot U'_A = I = (1 - U_A)/R$, Simulation, exakte Lösung
- Bsp: $U_E(t) = \sin(2\pi f t)$, $U_A(t) = ?$
- wirkt als Tiefpaß-Filter: Schwächung hoher Frequenzen

**Bemerkung zur Methodik**

- das gehört alles in das Gebiet der Analyse und Simulation analoger Schaltungen.

  das ist natürlich seit $\geq 100$ Jahren alles wohlbekannt
- man rechnet eleganterweise nicht den Amplitudenverlauf, sondern betrachtet stationäre Lösungen für harmonische Schwingungen:

  jede Kenngröße (Strom, Spannung, Widerstand usw.) ist dann eine (!) komplexe Zahl, Analyse mit Kirchhoffschen Regeln

  Amplitudenverlauf (falls überhaupt benötigt) durch Fourier(rück)transformation.
- und es gibt dafür auch (seit $\geq 50$ Jahren) passende (teure!) Software

### Weitere Filter: Hochpaß, Bandpaß

- (0,3) node [left]$U_E$ to [short,i=$I$,*-] (1,3) to [R,l=$R$, -*] (4,3) to [L,l=$L$] (4,1) node [ground] (3,3) to [short,-*] (5,3) node [right] $U_A$ ; , Spule: $U_A = L \cdot \frac{dI}{dt} = L \cdot I'$

  wirkt als *Hochpaß* (tiefe Frequenzen werden geschwächt)

- (0,3) node [left]$U_E$ to [short,i=$I$,*-] (1,3) to [R,l=$R$, -*] (4,3) to [L,l=$L$] (4,1) node [ground] (4,3) to [short,-*] (6,3) to [C,l=$C$] (6,1) node [ground] (6,3) to [short,-*] (8,3) node [right] $U_A$ ; , wirkt als *Bandpaß*

  (hohe und tiefe $f$ geschwächt, in der Nähe der Resonanzfrequenz weniger)

- Bandpaß mit Rückführung und Verstärkung:

  wirkt als *Oszillator* (schwingt auf Resonanzfrequenz)

### Weitere Filter: Allpaß

- lattice filter (d: Gitter- oder Leiter-Filter)

  (0,4) node [left] to [short,*-] (1,4) to [C,l=$C$] (5,4) to [short,-*] (6,4); (1,4) to [L,l=$L$] (3,2) to [] (5,0); (0,0) node [left] to [short,*-] (1,0) to [C,l=$C$] (5,0) to [short,-*] (6,0); (1,0) to [] (3,2) to [L,l=$L$] (5,4);

- vgl. Julius O. Smith, Physical Audio Signal Processing, W3K Publishing, `https://ccrma.stanford.edu/~jos/pasp/Allpass_Filters.html`

- angewendet im *Phaser*

### Aufgaben

- wie lautet die exakte (stationäre) Lösung für den RC-Tiefpaß mit Eingabe $U_E(t) = \sin(2\pi f t)$?

  Ansatz: $U_A(t) = a \cdot \sin(2\pi f t + \phi)$ mit unbekannten $a, \phi$.

- Experimente mit `https://gitlab.imn.htwk-leipzig.de/waldmann/circuit`, siehe auch autotool-Aufgabe.

  Simulieren Sie einen Allpaß, damit einen Phaser.

  TODO: Eingabe/Ausgabe von/nach WAVE-Datei einlesen

# 6 Elektrische Schwingungen (II)

### Spannungsgesteuerte Schaltungen

- Steuerung von System-Eigenschaften (z.B. Resonanzfrequenz, Filter-Steilheit) durch
  - Ausgabe-Spannung anderer Teilsysteme
  - Bedienerschnittstelle (Regler, Klaviatur — ebenfalls als Spannungsquellen realisiert)
- $\Rightarrow$ modularer Aufbau eines Synthesizers, Verbindung der Komponenten (= Programmierung) durch Kabel/Stecker
- Robert Moog: *Voltage Controlled Electronic Music Modules*, J. Audio Engineering Soc. Volume 13 Issue 3 pp. 200-206; July 1965, `https://www.aes.org/e-lib/browse.cfm?elib=1204`

### Spannungsgesteuerte Komponenten

- Verstärker (VCA, voltage controlled amplifier)

  eigentlich Multiplizierer: $U_A(t) = U_C(t) \cdot U_E(t)$
- Oszillator (VCO)

  Steuerspannung $\sim$ Frequenz: $U_A(t) = \sin(U_C(t) \cdot t)$
- Filter (VCF): Steuerspannung $\sim$ Resonanzfrequenz
- periodische $U_C$ mit kleiner Frequenz erzeugt durch

  LFO (low frequency oscillator)

### Steuerspannungen aus Benutzeraktionen

- einfachste Möglichkeit: Taste drücken/loslassen

  Impulslänge je nach Eingabe, Impulshöhe konstant
- mehr Ausdruck: Stärke des Tastendrucks bestimmt Impulshöhe (konstant über gesamte Länge)
- (Luxus: *gewichtete* Tastatur, simuliert Trägheit der Klavier-Mechanik)

- (Hüll)kurvenparameter für nicht-konstante Impulse:
  - Attack (Anstiegszeit auf maximale Höhe)
  - Decay (Abfallzeit bei noch gedrückter Taste)
  - Sustain (Impulshöhe nach Decay)
  - Release (Abfallzeit nach Loslassen der Taste)

**Erste Synthesizer in populärer Musik**

- spannungsgesteuerte modulare Synthesizer produziert ab 1963 (Robert Moog, Don Buchla)
  (Bsp: Buchla: *In The Beginning Etude II*, 1983?)
- erste Anwendungen (auf publizierten Aufnahmen)
  - für exotische Klänge als Verzierung in Standard-Popmusik (Byrds: *Space Odyssey*, 1967)
  - als Solo-Instrument
    * als Ersatz klassischer Instrumente, für klassische Musik (Wendy Carlos: *Switched on Bach*, 1968)
    * für neuartige, eigens komponierte Musik (Morton Subotnick: *Silver Apples of the Moon*, 1967)
- (frühere elektronische Instrumente: siehe *120 Years of Electronic Music* `http://120years.net/`)

**Theremin**

- Lev (Leon) Termen, 1922, Rußland
- wird berührungslos (!) gespielt, Prinzipien:
  - durch Handbewegung wird Kapazität eines Kondensators in einem HF-Schwingkreis (170 kHz) (!) geändert, dadurch die Frequenz der Schwingung
  - Tonhöhe: vom HF-Summensignal hört man nur die (niederfrequente) Differenzfrequenz zu einem zweiten (nicht verstimmten) Schwingkreis,
  - Lautstärke: HF-Bandpaß und Gleichrichtung erzeugt Steuerspannung für VCA
- Hörbeispiel: Captain Beefheart: *Electricity*, 1967

**Simulation mit grafischer Programmierung**

- ALSA modular synthesizer
- Komponenten (LFO, VCO, VCF, ...) auf Arbeitsfläche,
- Verbindung durch Kabel (Ausgangsgrad beliebig, Eingangsgrad 1)
- Verbindungen zur Außenwelt (Bsp.)
  - In: MCV (MIDI control voltage) Steuerspannung ist Tonhöhe der Taste eines virtuellen Keyboards (z.B. `vkeybd`) oder externen Keyboards (z.B. USB-MIDI)
    qjackctl (Alsa): `virtual keybd` output — `ams` input
  - Out: PCM,
    qjackctl (Audio): `ams` output port — `system` input port

**Übungen**

- Experimente mit *ALSA modular synthesizer*:
  - Ausprobieren LFO, VCO, VCF, MCV, ADSR (Env)
  - die zeitliche Umkehrung einer ADSR-Kurve ist im allgemeinen nicht ADSR – sondern nur für welche Parameter?
  - Nachbilden bestimmter Klänge
    - base drum,
    - snare drum,
    - der Grashüpfer in „Biene Maja“ (deutsche Tonspur der japanischen Verfilmung von 1975)
    - Becken (hihat, crash, ride)
    - Metallophon,
    - Flexaphon (Hörbeispiel: ca. bei 0:55 in Can: *Sing Swan Song*, 1972)
    - Xylophon,
    - Orgelpfeife, (Pan)Flöte
    - einzelner Wassertropfen, Regen, Wasserfall, Meer
- die Herausforderung ist: kleine Schaltung (wenige, einfache Bauteile) mit überraschendem Klang.
- Steuerung durch Sequencer, der Midi-Signale ausgibt, z.B. `http://www.filter24.org/seq24/`

# 7 Programme für Klänge

**Motivation**

elektrische Schaltungen zur Klangerzeugung ...

- real bauen (Analog-Synthesizer, Moog, Buchla, ...)
- oder simulieren. Bedienung/Beschreibung
  - grafisch (alsa modular synthesizer)
  - textuell (durch eine DSL)
    - separate DSL, Bsp: Csound
      `https://csound.com/`, Barry Vercoe 1985
    - eingebettete DSL (in Haskell): csound-expression
      Anton Kholomiov

      ```
      hall 0.5 ( usqr 6 * (sqr (400 * usaw 2.1)))
      ```

**csound-expression**

- Klangbeschreibung durch algebraischen Ausdruck

  ```
  hall 0.5 ( usqr 6 * (sqr (400 * usaw 2.1)))
  ```

- verwendet:
  - Operatoren aus Csound-API (VCO, VCF, ...)
  - (Spannungs-)Steuerung durch passende Argumente
  - Haskell (Typen, Funktionen, `*`, `map`, ...)
- wird in Csound-Ausdruck kompiliert, (ansehen mit
  `( renderCsd $ hall ...) >>= putStrLn`)
- Csound-Server interpretiert (berechnet Amplitudenverlauf)

  ```
  dacBy (setJack "ce" <> setRates 48000 0 <> setBufs 0 2041)
    $ osc 300
  ```

### CE-Beispiel: Additive Synthese

- Fourier-Darstellung der Rechteck-Schwingung

```
let f = 300
in sum $ map (\k -> (osc $ k * f) / k )
       $ map fromIntegral [1, 3 .. 9]
```

- das funktioniert, weil…
  - `k` und `f` den Typ `Sig` haben (nicht Zahl!)
  - für `Sig` die Addition definiert ist (`instance Num Sig`)
- Klang vergleichen mit `sqr f`, obere Grenze (9) variieren
- Übung: desgl. für Sägezahn-Schwingung,
  für Summe vieler harmonischer S. mit zufälliger Frequenz

### weitere Csound/CE-Beispiele und -Quellen

- Wind `https://hackage.haskell.org/package/csound-catalog-0.7.2/docs/Csound-Catalog-Wave.html#v:mildWind`
- Schlagzeuge (Hans Mikelson) `https://hackage.haskell.org/package/csound-catalog-0.7.2/docs/Csound-Catalog-Drum-Hm.html`
- Glocke (noiseBell) u.a. `https://hackage.haskell.org/package/csound-catalog-0.7.2/docs/Csound-Catalog-Wave.html#v:noiseBell`
- Anton Kholomiov: *Speed up you Csound workflow with Haskell*, Csound Journal 23 (2017) `http://csoundjournal.com/issue23/Csound_expression_paper.html`
- Types in Csound-Expression: `https://www.imn.htwk-leipzig.de/~waldmann/etc/untutorial/ce/`

### Schnittstellen für Live-Spiel: MIDI

- MIDI-Signalquelle
  - reelles Keyboard: `https://github.com/spell-music/csound-expression/issues/51`

  - virtuelles Keyboard:

    ```
    vdac $ midi $ \ m -> return $ osc $ sig $ cpsmidi m
    ```

- Signaturen:

```
midi :: Sigs a => (Msg -> SE a) -> SE a
cpsmidi :: Msg -> D
sig :: D -> Sig
osc :: Sig -> Sig
vdac :: RenderCsd a => a -> IO ()
return :: Monad m => a -> m a
```

### Schnittstelle für Live-Spiel: GUI

- Verwendung von GUI-Elementen aus Csound:

```
dac $ do
 (g,f) <- slider "f" (expSpan 20 2e4) 440
 panel g ; return $ osc f
```

- Signaturen:

```
slider :: String -> ValSpan -> Double -> Source Sig
expSpan :: Double -> Double -> ValSpan
type Source a = SE (Gui, Input a); type Input a = a
panel :: Gui -> SE () ; osc :: Sig -> Sig
dac :: RenderCsd a => a -> IO ()
```

- mehrere Element kombinieren durch `hor,ver :: [Gui] -> Gui`

### Übungen

- csound-expression:
  - die Aufgaben für alsa-modular-synthesizer (Maultrommel usw.)
  - Tonerzeugung: Beispiele anhören `https://gitlab.imn.htwk-leipzig.de/waldmann/cm-ws18/tree/master/kw46/data` und den Csound-Expression-Ausdruck raten.

    Benutzt wurden nur: osc, usaw, usqr, white, mul, at, (+), (-), (*), hall, fvdelay

    Erzeugen Sie selbst solche Beispiele! Hochladen und die anderen raten lassen.

  - zum Vergleich — Hörbeispiel: *Autechre*: Perlence (Album: Quaristice, 2008)
  - GUI benutzen `https://github.com/spell-music/csound-expression/blob/master/tutorial/chapters/FxFamily.md#ui-stompboxes`

- zu Autechre: `https://www.factmag.com/2017/02/25/autechre-gear-synths-samp`
  lustige Kritik an Max/MSP: `https://news.ycombinator.com/item?id=22346556`

- Vorschlag Masterarbeit(en): Automatisierung der Rate-Aufgabe.

  `https://gitlab.imn.htwk-leipzig.de/waldmann/computer-mu/-/issues/4` siehe auch `https://imweb.imn.htwk-leipzig.de/~waldmann/edu/modul/seminar/autograde/`

# 8 Harmonielehre

### Motivation, Plan

- bisher: Geräusche,

  Töne (Grundfrequenz, Obertöne, Spektren)

- heute: welche Töne klingen gut
  - zusammen (in Akkorden),
  - nacheinander (in Melodien)?

- später:
  - Folgen von Akkorden (Kadenzen),
  - Führung mehrerer Stimmen (Kontrapunkt)

  und Notation dafür (algebraisch, grafisch – Partituren)

### Klassische Literatur

- Hermann von Helmholtz: *Die Lehre von den Tonempfindungen als physiologische Grundlage für die Theorie der Musik*, Vieweg 1863. `https://reader.digitale-sammlungen.de/de/fs1/object/display/bsb10598685_00366.html`

- (von H.H. zitiert) Leonhard Euler: *Tentamen novae theoriae Musicae*, Petropoli, 1739. http://eulerarchive.maa.org/pages/E033.html,
  vgl. Patrice Bailhache: Music translated into Mathematics, https://web.archive.org/web/20050313140417/http://sonic-arts.org/monzo/euler/euler-en.htm
- Hugo Riemann: *Katechismus der Harmonielehre*, 1890 https://archive.org/details/katechismusderh00riemgoog/

**Die Naturtonreihe**

- bei schwingender Saite, schwingender Luftsäule

  kommen neben Grundton $f$ ganzzahlige Obertöne vor,

  bilden die Naturtonreihe $f, 2f, 3f, 4f, 5f, \ldots$

- einzelne Obertöne lassen sich durch passende Spielweise betonen (isolieren) (z.B. Flageolett)

  Bsp: Canned Heat: *On the Road Again*, 196?. (Flageolett-Töne im Intro)

- die Naturtonreihe bis $15f$ reduziert (durch Halbieren)

  | 1, | 9/8, | 5/4, | 11/8, | 3/2, | 13/8, | 7/4, | 15/8, | 2 |
  |---|---|---|---|---|---|---|---|---|
  | c | d | e | $\approx$ f | g | $\approx$ a♭ | $\approx$ b♭, | $\approx$ b | c’ |

  11/8: das Alphorn-Fa

- wie stimmt man Instrumente mit mehreren Saiten?

**Konsonanz**

- wie stimmt man Instrumente mit mehreren Saiten $f, g, \ldots$

  $g$ *nicht* als Oberton von $f$,

  sondern wir wollen neue Töne. Welche?

- Töne wie z.B. 300 Hz, 315 Hz
  - klingen nicht gut zusammen (sondern rauh, dissonant)
  - das Ohr nimmt die Schwebung (mit 15 Hz) wahr
- Schwebungen zw. 10 Hz und 40 Hz sind unangenehm

  (nach Helmholtz: 33 Hz ist am schlimmsten)

  konsonante Töne haben keine solchen Schwebungen

**(Vermeiden von) Schwebungen**

- $f, g$ konsonant $:=_{\text{def}}$ keine Schwebung geringer Frequenz zwischen Obertönen von $f$ und Obertönen von $g$.
- $S(f,g) := \min\{|a \cdot f - b \cdot g| : a, b \in \mathbb{N}, af \neq bg\}$
  Bsp: $S(300, 315), S(270, 375), S(270, 360) \ldots$
- Satz: $S(f,g)$ ist der ... von $f$ und $g$.
  (Begriff und Berechnung bekannt aus 1. Semester)

**Konsonanz**

- $f, g$ konsonant, wenn $\gcd(f,g)$ groß ...
  - absolut: $\gcd(f,g) > 40$Hz
  - relativ: $\gcd(f,g)/\max(f,g) \to$ groß
- Hör-Eindruck: $(80, 120)$ gegenüber $(480, 520)$
  spricht für die relative Definition.
- Def: $R(f,g) := \gcd(f,g)/\max(f,g)$
  hier $\gcd(f,g)$ auch für $f, g \in$ definiert als $\min_{a,b}\{af - bg\}$
- Aufg: Bestimme $1 = f_0 < f_1 \ldots < f_k = 2$
  mit $W(f) = \min\{R(f_i, f_j) \mid 0 \leq i < j \leq k\}$ maximal
  Bsp: $k = 3$. Für $1, 4/3, 5/3, 2$ ist $W(f) = \ldots$, geht besser?

**Die Töne nach Pythagoras**

- nach Pythagoras (ca. 500 v.Chr.) konstruiere Tonmenge
  - beginne mit 1 (Grundfrequenz)
  - multipliziere mit 3/2,
  - multipliziere mit 1/2, falls $> 2$

$$\underset{c}{1}, \quad \underset{g}{\tfrac{3}{2}}, \quad \underset{d}{\tfrac{9}{4} \to \tfrac{9}{8}}, \quad \underset{a}{\tfrac{27}{16}}, \quad \underset{e}{\tfrac{81}{32} \to \tfrac{81}{64}}, \quad \underset{b}{\tfrac{243}{128}}, \quad \underset{f\sharp}{\tfrac{729}{256} \to \tfrac{729}{512}}, \quad \underset{c\sharp}{\ldots}$$

- *pentatonische* Skala: die ersten 5 Töne dieser Reihe,

  nach Frequenzen geordnet $c, d, e, g, a$

  Bsp: The Monochrome Set: *Iceman*, Album: Spaces Everywhere, 2015. (Intro, Skala von $d$: $\{d, e, f\sharp, a, b\}$)

- *diatonische* Skala: die ersten 7 Töne dieser Reihe

  geordnet $g, a, b, c, d, e, f\sharp$

**Die Töne nach Pythagoras**

- (bisher) aufsteigend $c, g, d, a, e, b, f\sharp, c\sharp, g\sharp, d\sharp, a\sharp, e\sharp, b\sharp, f\sharp\sharp, \ldots$
- absteigend: beginne mit $2$, mult. mit $2/3$, ggf. mit $2$

  $2, \quad \frac{4}{3}, \quad \frac{8}{9} \to \frac{16}{9}, \quad \ldots$
  $c', \quad f, \quad\quad b\flat, \quad\quad e\flat, \quad a\flat, \quad d\flat, \quad g\flat, c\flat, f\flat, b\flat\flat, \ldots$

- vereinige die jeweils ersten 7 Töne, ordne.

  $c, d\flat, d, e\flat, e, f, \underline{g\flat, f\sharp}, g, a\flat, a, b\flat, b, c'$

- Abstände sind

  $2^8/3^5 \approx 1.053$ (zw. $b$ und $c'$), $3^7/2^{11} \approx 1.068$ (zw. $b\flat$ und $b$),

  sowie <u>einmal</u> $3^{12}/2^{19} \approx 1.013$

**Eigenschaften der Stimmungen**

- die pythagoreische Reihe enthält 12 exakte 3:2

  $g\flat \to d\flat \to a\flat \to e\flat \to b\flat \to f \to c \to g \to d \to a \to e \to b \to f\sharp$

- schließt nicht: $1 \neq (3/2)^{12}/(2/1)^7$ (pythagoreisches Komma)
- Konsonanzen aus der Naturtonreihe fehlen, z.B. 4:5:6.

  angenähert durch $c : e : g = 1 : \frac{81}{64} : \frac{3}{2}$

  der Fehler $\frac{81}{64}/\frac{5}{4}$ ist das diatonische Komma

- *reine Stimmung*: 4:5:6 für spezielle Akkorde:

  Tonika c,e,g, Subdominante f,a,c, Dominante g,b,d.

- *gleichtemperierte St.*: das pythagoreische Komma wird geschlossen, d.h., g:c $= 2^{7/12} \approx 1.4983$

**Die diatonische Skala**

- mit $f = 2/3, g = 3/2 \cdot c, d = 9/8 \cdot c, \ldots$
  $c \overset{T}{-} d \overset{T}{-} e \overset{S}{-} f \overset{T}{-} g \overset{T}{-} a \overset{T}{-} b \overset{S}{-} c'$
  die Abstände sind: T: Ganzton, S: Halbton
- hiervon sind die Intervallbezeichnungen abgeleitet:
  Sekunde, Terz, Quarte, Quinte, Sexte, Septime, Oktave.
- große Terz ($2T$) $c - e, \ldots$, kleine Terz ($T + S$): $e - g, \ldots$
- der *Modus* beschreibt eine zyklische Verschiebung:
  - ionisch (Dur): $c, d, \ldots$,
  - äolisch (Moll): $a, b, \ldots$.

**Akkorde (Dreiklänge)**

- Grundformen (konsonant):
  - Dur (große Terz, kleine Terz) $C = \{c, e, g\}$
    in C-Dur-Skala enthalten: $C, F, G$
  - Moll (kleine Terz, große Terz) $C^- = \{c, e\flat, g\}$
    in C-Dur-Skala enthalten: $D^-, E^-, A^-$
- Modifikationen (dissonant):
  - vermindert: (kleine, kleine) $C^0 = \{c, e\flat, g\flat\}$
    in C-Dur-Skala enthalten: $B^0$
  - vergrößert: (große, große) $C^+ = \{c, e, g\sharp\}$
    nicht in C-Dur-Skala enthalten.

**Akkorde (Vierklänge)**

- Dreiklang plus Septime (kleine oder große)
- Bsp: $C^7 = \{c, e, g, b\flat\}, C^{\text{maj7}} = \{c, e, g, b\}$
- skalen-eigene Vierklänge:
  $C^{\text{maj7}} = \{c, e, g, b\}, D^{-7} = \{d, f, a, c\}, E^{-7}, F^{\text{maj7}}, G^{-7}, A^{-7}, B^{-7(\flat 5)}$
- simple Realisierung in `electribe 2`:
  - Dur-Skala, 4 Noten pro Akkord (Grundton, $+2, +4, +6$),
    da kann überhaupt nichts schief gehen, ...
  - auch bei „frei improvisierter“ Melodie nicht
    (XY-Pad: X ist Tonhöhe (aus Skala), Y ist Arpeggio)
  - das klingt aber doch beliebig,
    woher kommt die musikalische Spannung?

**Aufgaben**

1. bestimmen Sie die Frequenzverhältnisse für C-Dur, d-Moll und e-Moll in der C-Dur-Skala bei Stimmung
   - diatonisch
   - rein
   - gleich temperiert

   und vergleiche Sie akustisch (csound-expression)
2. Konstruktion der chromatischen Töne nach Paul Hindemith (Unterweisung im Tonsatz, 1937):
   (a) zu jedem Ton aus der Obertonreihe des Grundtons (c) werden mögliche Grundtöne bestimmt. Bsp: $5 \cdot c = 4 \cdot ?$.
   Dabei Multiplikation mit $1 \ldots 6$, Division durch $1, (2), 3, (4), 5$, mit Identifikation von Oktaven.
   Welche Töne entstehen aus c?

(b) Dieser Vorgang wird für jeden der entstandenen Töne wiederholt.

Welche neuen Töne enstehen? Sind die Abstände gleichmäßig (oder fehlen noch Töne)? Vergleich mit pythagoreischer Skala.

3. was hat H. Helmholtz auf S. 291f. gerechnet/gezeichnet? Rekonstruieren Sie die „einfachste mathematische Formel“, erzeugen Sie daraus die Diagramme, vergleichen Sie mit denen im Buch

4. was hat L. Euler gerechnet? (Helmholtz S. 349, Fußnote) Überführen Sie die dort zitierte rekursive Definition der Stufenzahl in eine explizite Formel.

   Bestimmen Sie die Stufenzahl der Akkorde aus der 1. Aufgabe.

   Wo steht die Definition im Originaltext von Euler?

5. mit csound-expression oder alsa-modular-synthesizer (Module: CV: Random, Quantizer)

   - Akkorde (Dreiklänge, Vierklänge) erzeugen.
   - Akkorde aus einer Skala zufällig aneinanderreihen,
   - dazu eine zufällige Melodie aus dieser Skala

6. zur Stimmung der Gitarre:

   - man kann die unteren (tiefen) Saiten so stimmen: Saite mit Flagoelett bei 1/4 = nächst-höhere Saite mit Flageolett bei 1/3.

     Welches Intervall ist das? Wenn man bei tiefem E beginnt und alle Saitenpaaare so stimmt, welcher Ton ist dann auf der 6. (höchsten) Saite?

     Das Intervall zwischen 4. und 5. Saite wird bei üblicher Stimmung um einen halben Ton verringert.
   - Es werden gern auch abweichende Stimmungen verwendet, vgl. `http://www.sonicyouth.com/mustang/tab/tuning.html` Warum?

     Bsp: Sonic Youth: *Hyperstation*, Album: Daydream Nation (1988)

     Das Bild auf der Hülle ist `https://www.gerhard-richter.com/en/art/paintings/photo-paintings/candles-6/candle-5195/`
   - Wie ist die Gitarre im Intro von *On the Road Again* gestimmt?

     Der 4. Ton bleibt liegen, wird bei Beginn des Themas verschoben. Wohin, warum?

     in csound-expression nachbauen! Spezifikation von Tonfolgen vgl.

```
notes = fmap temp $ fmap (220 * ) [1, 5/4, 3/2, 2]
q = mel [mel notes, har notes]
dac $ mix $ sco oscInstr q
```

https://github.com/spell-music/csound-expression/blob/master/tutorial/chapters/ScoresTutorial.md#functions-for-sequenti

# 9 Algebraische Komposition

**Einleitung**

- klassisch: Musikstück repräsentiert durch Partitur,
  - Ton repräsentiert d. Note, bezeichnet Tonhöhe, -dauer
  - Tempo, Klangfarbe, Lautstärke
    - durch weiter Annotationen spezifiziert
    - oder nicht, d.h., dem Interpreten überlassen
  - Komposition:
    - Noten nebeneinander bedeutet Töne nacheinander
    - Noten (Zeilen) übereinander: Töne (Stimmen) gleichzeitig
- jetzt: Musikstück repräsent. d. (abstrakten Syntax-)Baum

**Literatur, Software**

- Paul Hudak , Tom Makucevich , Syam Gadde , Bo Whong: *Haskore Music Notation - An Algebra of Music*, JFP 1995, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.8687

  Partitur kompiliert zu MIDI-Strom, der von Hard- oder Software-Synthesizer interpretiert wird

  Donya Quick et al.: http://euterpea.com/

- Anton Kholomiov: Csound-Expression Tutorial – Scores, https://github.com/spell-music/csound-expression/blob/master/tutorial/chapters/ScoresTutorial.md

  Partitur wird durch Csound-Instrumente interpretiert, P. kann Csound-spezifische Elemente enthalten

## Partituren als Abstrakte Syntaxbäume

- ```
  data Music a
    = Prim (Primitive a)
    | (Music a) :+: (Music a)   -- sequentielle K.
    | (Music a) :=: (Music a)   -- parallele K.
    | Modify Control (Music a)
  data Primitive a = Note Dur a | Rest Dur
  data Control = Tempo Rational
    | Transpose AbsPitch
    | Instrument InstrumentName | ...
  ```
- Beispiele

  ```
  Prim (Note 1 60) :: Music AbsPitch
  Prim (Note 1 (C,4)) :: Music Pitch
  Modify (Transpose 4)
     (Prim (Note 1 (C,4)) :=: Prim (Note 1 (G,4)))
  ```

## Konstruktion von Partituren

- ```
  tr = transpose
  minor x = chord [x, tr 3 x, tr 7 x]
  bass x = tr (-12) $ line
    [ scaleDurations (1/2) x, tr 7 x, tr (-5) x ]
  pat x = chord
    [ instrument AcousticGrandPiano
      $ line [ Rest qn, minor x, Rest qn, minor x ]
    , instrument AcousticBass $ bass x ]
  theme = line $ map (\x -> pat $ x 2 qn) [ a,e,d,a ]
  ```
- benutzerdefinierte Namen (`tr`), Funktionen (`minor`), Standard-Funktionen (`map`)
- beschreibt diesen AST:

```
(Modify (Instrument AcousticGrandPiano) (Prim (Rest (1 % 4)) :+: ((Prim (Note
(1 % 4) (A,2)) :=: (Modify (Transpose 3) (Prim (Note (1 % 4) (A,2))) :=: (Modify
(Transpose 7) (Prim (Note (1 % 4) (A,2))) :=: Prim (Rest (0 % 1))))) :+: (Prim
(Rest (1 % 4)) :+: ((Prim (Note (1 % 4) (A,2)) :=: (Modify (Transpose 3) (Prim
(Note (1 % 4) (A,2))) :=: (Modify (Transpose 7) (Prim (Note (1 % 4) (A,2))) ...
```

## Von Partitur zu Interpretation

- ```
  data MEvent = MEvent
    {  eTime    :: PTime, -- onset time
       eInst    :: InstrumentName, -- instrument
       ePitch   :: AbsPitch, -- pitch number
       eDur     :: DurT, -- note duration
       eVol     :: Volume, -- volume
       eParams  :: [Double] } -- optional other parameters
  type Performance = [ MEvent ] -- aufsteigende onsets
  ```

- ```
  musicToMEvents
     ::  MContext -> Music1 -> (Performance, DurT)
  musicToMEvents
        c@MContext{mcTime=t, mcDur=dt} (m1 :+: m2) =
   let (evs1,d1) = musicToMEvents c m1
       (evs2,d2) = musicToMEvents c{mcTime = t+d1} m2
   in  (evs1 ++ evs2, d1+d2)
  ```

**Partitur und Interpretation**

- die Konstruktoren der Partitur definieren die Signatur $\Sigma$ einer Algebra
- die Partituren sind Terme über dieser Signatur
- Performances bilden die Trägermenge $D$ einer $\Sigma$-Algebra
- die Aufführung (`perform :: Music a -> Performance`) ist die Interpretation von Term nach Algebra:

  dabei wird jedes Symbol aus $\Sigma$ durch eine Funktion über $D$ ersetzt, Bsp:

  - `:+:` durch ++ (Verkettung),
  - `:=:` durch `merge` (Zusammenfügen)

**Eigenschaften der Operationen**

- für die zweistelligen Kompositionen

  ```
  seq2, par2 :: Score -> Score -> Score
  seq2 = (:+:), par2 = (:=:)
  ```

- `seq2` ist semantisch assoziativ: für alle $p, x, y, z$

```
perform p (seq2 (seq2 x y) z)
  = perform p (seq2 x (seq2 y z))
```

neutrales Element? kommutativ? Desgl. für `par2`

- gelten Distributiv-Gesetze? (Nein).
- Ü: wann sind `par2 (seq2 a b) (seq2 c d)` und `seq2 (par2 a c) (par2 b d)` semantisch gleich?

**Historische Formen der Mehrstimmigkeit**

- Cantus Firmus (feststehende Melodie, die anderen Stimmen sind Verzierung)
- Kontrapunkt (Note gegen Note): Vorschriften zur Konstruktion der Begleit-Stimmen, u.a.
  - Konsonanzen zu bestimmten (schweren) Zeitpunkten
  - keine Parallelen (gleichmäßiges Auf- oder Absteigen)
- Fuge (Flucht, die Stimmen fliehen voreinander) alle Stimmen sind aus *einem* Thema konstruiert durch
  - zeitlichen Versatz (Kanon), zeitliche Spiegelung
  - Versatz der Tonhöhe, Skalierung des Tempos, ...
- Kadenzen (Akkordfolgen) mit untergeordneten Stimmen

**Kanon**

- eine Stimme wird mehrfach zeitlich versetzt
- Beispiel: Karl Gottlieb Hering (1766-1853): *C A F F E E*

  [name=caffee,program=abcm2ps,options=-O= -B4] X: 1 K: F M: 3/4 ”1.c2 A2 F2 — F2 E2 E2 — E2 G2 E2 — F E F G F2 — w:C A F F E E trink nicht so viel* Caf* fe ”2.Ä A c c A A — B A B c B2 — G G B B G G — A G A B A2 — w:nicht für Kin-der ist der Tür* ken* trank L: 1/4 ”3.F F F — G G G — C C c — c C F — w:sei doch kein Mu-sel-man der das nicht las-sen kann
- Übung: 1. Harmonien bestimmen, 2. programmieren

### Fuge

- eine anspruchsvolle Form des Kontrapunktes. alle Stimmen sind aus *einem* Thema konstruiert durch
  - zeitlichen Versatz (wie im Kanon)
  - Versatz der Tonhöhe
  - zeitliche Spiegelung
  - Tonhöhen-Spiegelung
  - Skalierung des Tempos
- Johann Sebastian Bach (1685–1750): *Die Kunst der Fuge*, Contrapunctus XV - Canon per Augmentationem in Contrario Motu, (Solist: Pierre-Laurent Aimard, 2008)
  `http://www.mutopiaproject.org/ftp/BachJS/BWV1080/contrapunctusXV/`
- Ü: Operatoren in Partitur erkennen, implementieren

### Akkorde (Ton-Inhalt)

- Grundformen: Dur und Moll

```
major x d =
  Par [ Note x d , Note (x+4) d, Note (x+7) d ]
minor x d =
  Par [ Note x d , Note (x+3) d, Note (x+7) d ]
```

- mit Septime (kleiner, großer)

```
major7 x d = Par
 [ Note x d, Note (x+4) d, Note (x+7) d, Note (x+10) d ]
major7maj x d = Par
 [ Note x d, Note (x+4) d, Note (x+7) d, Note (x+11) d ]
```

- weitere Varianten durch Umstellen (anderer Grundton); Hinzufügen, Ändern, Weglassen von Tönen

**Die Kadenz**

- lateinisch cadere = fallen
- die Voll-Kadenz (Quinten abwärts) in C-Dur:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3kl. | $C$ | $F$ | $B^0$ | $E^-$ | $A^-$ | $D^-$ | $G$ | $C$ |
| 4kl. | $C^7$ | $F^7$ | $B^{-7(\flat 5)}$ | $E^{-7}$ | $A^{-7}$ | $D^{-7}$ | $G^7$ | $C^7$ |
| Ton | I | IV | VII | III | VI | II | V | I |
| | T | S | | Dp | Tp | Sp | D | T |

- verkürze, ersetze $B^0 = \{b, d, f(, a)\}$ durch $G = \{g, b, d(, f)\}$,
  ergibt die Kadenz: $C, F, G, C = T, S, D, T$
- Tonika (1,3,5), Dominante (5,7,9),Subdominante (4,6,8),
- Dur: $T = (c, e, g)$, $S = (f, a, c)$, $D = (g, b, d)$,
- Moll: $t = (c, e\flat, g)$, $s = (f, a\flat, c)$, $d = (g, b\flat, d)$

**Funktions-Harmonik**

- Betrachtung der Akkorde nach ihrer (vermuteten, häufigen) Funktion in musikalischer Phrase.

  nach Hugo Riemann (1849–1919):

  T These, S Antithese, D Synthese.
- in einer Kadenz können Akkorde durch Parallelen vertreten werden
- die Parallelen (mit gleicher großer Terz)

  $Tp = (-1, 1, 3) = (a, c, e)$, $tP = (3, 5, 7) = (e\flat, g, b\flat)$,

  entsprechend $Dp, dP, Sp, sP$
- The Beatles: *Penny Lane*: T, Tp, Sp, D (C, Am, Dm, G)
- harmonische Analyse einer Stelle aus Bach: BWV 268

**Vermischte Dokumente zur Harmonielehre**

- Über Hugo Riemann, von dessen Sohn Robert: `http://www.hugo-riemann.de/`
- Kritik an Riemann durch Heinrich Schenker (1868–1935) `https://web.archive.org/web/20120403032916/http://www.schenkerdocumentsonline.org:80/profiles/person/entity-000712.html`
- Kritik an einer Kritik an Schenkers Theorie der *Urlinie* `https://web.archive.org/web/20160731145955/http://schenkerdocumentsonline.org/documents/other/OJ-21-24_1.html`

**Kadenzen in der Popmusik**

- Kadenz T S (T) D T
- Beispiele: tausende, u.a. Beach Boys: *Little Honda*, 1964 (Version von Yo La Tengo, 1997)

  Strophe: $DDDD|GGDD|AADA$
- The Jesus and Mary Chain: *Upside Down*, 1984 (auf Creation Records). Strophe: $4 \cdot (GGGC)\ 4 \cdot C\ 4 \cdot G$
- Beatles: *Tomorrow Never Knows*, 1966.
- Lou Reed: „One chord is fine. Two chords is pushing it. Three chords and you're into jazz."
- Thelonius Monk: *Round Midnight*, 1944

**Übungen**

1. zu Folie „Partitur und Interpretation":
   (a) Warum „schwach monoton", nicht stark?
   (b) Welche Rechnung muß im Zweig `Par2 x y ->` stattfinden? Wie werden die Teilresultate verknüpft?
   (c) Welches ist der abstrakte Datentyp für `[Event]` (welche Operationen gehören zur API)? Welche effiziente Implementierungen dafür kennen Sie?

2. Fragen von Folie „Eigenschaften der Operationen“

3. zu Bach: Contrapunktus XV (canon per augmentationem in contrariu motu)

   (a) Bestimmen Sie die globale zeitliche Struktur der Komposition.

   Der 1.Takt der 1. Stimme erscheint (gedehnt und gespiegelt) in Takt 5 und 6 der 2. Stimme. Wo noch?

   Was zeigt der Trennstrich nach Takt 52 an?

   (b) Bestimmen Sie die Tonhöhen-Abbildung (Spiegelung) von erster zu zweiter Stimme.

   Lesehilfe: Der Violin-Schlüssel bezeichnet das G (der Kringel, zweite Notenlinie von unten), der Baß-Schlüssel bezeichnet das F (der Doppelpunkt, zweite Notenlinie von oben)

4. Programmieren Sie das Thema von Jean-Michel Jarre: Oxygen Pt. 2 als eine Verschmelzung von zwei einfachen Melodien

5. Programmieren Sie den CAFFEE-Kanon (3 Stimmen, jede mit eigenem Instrument).

   (a) Ergänzen Sie `https://gitlab.imn.htwk-leipzig.de/waldmann/cm-ws18/blob/master/kw47/Caffee.hs`

   Beschreibung der Bibliotheks-Funktionen: `https://github.com/spell-music/csound-expression/blob/master/tutorial/chapters/ScoresTutorial.md`

   (b) Benutzen Sie eine Darstellung (d.h., Unterprogramme), die die lokale Strukur ausnutzt, z.B.: zweite Häfte der 2. Zeile ist Transposition der ersten Hälfte.

   Wir verschieben nicht chromatisch (2 Halbtöne), sondern diatonisch (1 Ton in der F-Dur-Skala).

6. Realisieren Sie auf ähnliche Weise eine Voll-Kadenz

   (a) effizient programmieren unter Benutzung der Skalen-Numerierung

   (b) eine dazu passende Melodie programmieren

   Hinweis: jede Melodie (aus Skalentönen) paßt

# 10 Performing with Patterns of Time

### Überblick

- Quelle: Thor Magnusson und Alex McLean: P.w.P.o.T, Kap. 14 in: Oxford Handbook of Algorithmic Music, OUP 2018, `https://slab.org/publications/`

  Software: `https://tidalcycles.org/`

- algebraische Beschreibung von periodischen Verläufen (Parameter für Klänge), eingebettete (in Haskell) DSL

- Back-end: `https://supercollider.github.io/` James McCartney, 1996–

- Tidal benutzt SC zum Abspielen von Samples

- Tidal ist System für live-coding (durch ghci-Kommandos)

**Tidal - Beispiel**

- Sound-Server (supercollider) starten

```
sclang dirt_startup.scd
```

- Ghci starten

```
ghci
:script BootTidal.hs
```

- Klänge ausgeben

```
d1 $ s "bd [sn sn]"
d2 $ s "[jvbass*2]*2" |*| n "0 1 2 3 4"
hush
```

**Grundlagen Tidal (Modell)**

- ein Muster `m :: Pattern a` beschreibt eine periodische Abbildung von Zeit nach `a`

- elementares Muster: `pure x` mit Periode 1

- `c :: ControlMap` Parameter zum Sample-Abspielen

  s: Verzeichnis, n: Datei-Nummer, begin, end, speed ...

- Funktionen zum Abspielen:

  ```
  d1, d2, ... :: Pattern ControlMap -> IO ()
  ```

- ```
  let p = pure $ M.fromList [("s", VS "bd")]
  d1 p
  queryArc p (0,3)
    [(0>1)|s: "bd",(1>2)|s: "bd",(2>3)|s: "bd"]
  ```

**Transformation von Mustern**

- Bsp. verwenden `p = run 3`, mit `queryArc p (0,1)`

  ```
  ==> [(0>1/3)|0, (1/3>2/3)|1, (2/3>1)|2]
  ```

- Transformation der Werte `fmap (\ x -> x+1) p`

  ```
  ==> [(0>1/3)|1, (1/3>2/3)|2, (2/3>1)|3]
  ```

- zeitliche Verschiebung `(1/3) <~  run 3`

  ```
  ==> [(0>1/3)|1, (1/3>2/3)|2, (2/3>1)|0]
  ```

- zeitliche Streckung `slow 2 p`

  ```
  [(0>2/3)|0, (2/3>1)-4/3|1, 2/3-(1>4/3)|1, (4/3>2)|2]
  ```

**Parallele Komposition**

- parallele Komposition (Vereinigung der Ereignismengen) `stack :: [Pattern a] -> Patter`

  ```
  fast 3 $ stack [ slow 2 (s "sn"), slow 3 (s "bd") ]
  ```

- parallele Komposition mit Kombination der Werte

  ```
  (<*>) :: Pattern (a -> b) -> Pattern a -> Pattern b
  ```

  Struktur von: beiden Seiten (`<*>`), links (`<*`), rechts (`*>`)

  - `p |+| q = (pure (+) <*> p) <*> q`
  - `(#) = (|>)` = Struktur von links, Werte von rechts
    vgl. `p # gain 0.7 # gain 0.3` mit `p # gain 0.7 |* gain 0.3`
  ```

### Sequentielle Komposition

- Nicht wie in z.B. `:+:` in Euterpea,

  weil ein Tidal-Muster kein Ende hat!

- Verschränkung von Mustern: `cat :: [Pattern a] -> Pattern a`

  für $p = \texttt{cat}[p_0, .., p_{k-1}]$ bedeutet $p[0 \ldots 1] = p_0[0 \ldots 1], p[1 \ldots 2] = p_1[0 \ldots 1], p[2 \ldots 3] = p_2[0 \ldots 1], p[3 \ldots 4] = p_0[1 \ldots 2], p[4 \ldots 5] = p_1[1 \ldots 2], \ldots$

- Denkmodell: jedes Muster ist Tonbandmaschine (Spur), bei $\texttt{cat}[p_0, .., p_{k-1}]$ spielt der Reihe nach jede Maschine $p_i$ für 1 Einheit

### Die Muster-DSL von Tidal

- zusätzlich zur bisher beschriebenen eDSL:

  konkrete Syntax für Operationen, Transformationen:

  Bsp: `"[[bd sn?]*2, [hc?*2 ho]*4]"`

  - Hintereinander: in <>: `cat`, in `[]`: `fastcat`,
  - Komma in (beiden) Klammern: `stack` (außen)
  - `x*3` bedeutet: `fast 3 x`, `x/2` bedeutet: `slow 2 x`
  - `x?`: `degrade x` (einige Ereignisse weglassen)

- `s $ fromString "[bd sn]"` ist äquivalent zu

  `s $ fastcat [ pure "bd", pure "sn" ]`

  mit `:set -XOverloadedStrings` : `s "bd sn"`

- damit kürzere Notation, aber Vorteile der eDSL (statische Typisierung, Funktionen, HO) werden aufgegeben

### Audio-Effekte in Tidal

- Ausdrucksmittel sind hier (z.B. ggü. csound-expression) absichtlich beschränkt, Schwerpunkt von Tidal ist die Kombination von (zeitlichen) Mustern, nicht von Effekten

- ein globaler Effekt-Weg, Parameter in `ControlMap`

```
d1 $ sound "sn"
  # delay 0.7 # delaytime (2/3) # delayfeedback 0.7
  # room 0.7  # size 0.9
  # cutoff 400 # resonance 0.7
```

- Parameter sind auch Muster, z.B., `size "0.5 0.9"`
- durch `orbit <string>` unabhängige Effektstrecken

```
stack [ .. # orbit "0", .. # orbit "1" ]
```

**Live Coding**

- der eigentliche Erfindungsgrund und Anwendungsfall von Tidalcycles ist „live coding“: sichtbare Arbeit am Quelltext (wird projiziert) während der Aufführung
- ghci sendet Ereignisse an Backend (supercollider) — falls Eingabe typkorrekt war.
  falls nicht: läuft bisheriges Muster weiter.
- die harte Variante ist *from scratch live coding*: Editor ist anfangs leer (und der gesamte Text bleibt auf einer Bildschirmseite, $\leq 10$ Zeilen)
- Erweiterung (Corona!): kollaboratives Live-Coding: gemeinsames Editieren eines Quelltextes

**Beispiel: Tidal von Kindohm (Mike Hodnick)**

- Kindohm at International Conference on Live Coding, October 15th 2016, at The Spice Factory, Hamilton, Ontario, Canada. `https://www.youtube.com/watch?v=smQOiFt8e4Q`
- vgl. `http://iclc.livecodenetwork.org/2015/papers.html`, `http://iclc.livecodenetwork.org/2016/papers.html`
- aktuelles Album: *Mesabi Range* `https://nadarecs.bandcamp.com/album/kindohm-mesabi-range`

### Übungen

1. Tidal installieren und starten

   (a) jack richtig configurieren, siehe `https://gitlab.imn.htwk-leipzig.de/waldmann/cm-ws18#hinweise-zur-richtigen-konfiguration-von-audi`

   (b) SuperDirt installieren

   (c) dann SC-Server starten mit `sclang superdirt_startup.scd`

   (d) Tidal-Cycles ist installiert, dann

   ```
   ghci
   :script BootTidal.hs
   d1 $ s "bd sn"
   hush
   ```

2. Tidal benutzen

   (a) Types in Tidal-Cycles: `https://www.imn.htwk-leipzig.de/~waldmann/etc/untutorial/tc/`

   (b) für einige Audio-Files (`https://gitlab.imn.htwk-leipzig.de/waldmann/cm-ws18/tree/master/kw49/data`) den Tidal-Quelltext erraten. Hinweis: benutzt wurden `s "casio:1"`, `fast`, `speed`, `rev`, `every`, `room`

   (c) Steve Reich: *Piano Phase* nachbauen.
   Hinweis: chromatische Tonfolgen so möglich: `s "sine" |+| speed (fmap (\i -> 2*`

   (d) Mike Hodnick: Deconstructing D-Code, `https://blog.mikehodnick.com/deconstructing-d-code/`

   (e) Antonio Carlos Jobim, Newton Mendonca: *One Note Samba*, Rec. Stan Getz, Charlie Byrd, 1962, LP *Jazz Samba*. Der Stil wurde als *Bossa Nova* bekannt.

      i. Welche Rolle spielt der festgehaltene Ton ($f$) im jeweiligen Akkord? ($D^{-7}D^{\flat 7}C^{-7}B^{7\flat 5}$)

      ii. Programmieren Sie den Rhythmus (Stick ab 1:28 min)

### Planung der Abschluß-Projekte

- Ziel: Methoden aus der Vorlesung benutzen, um eine musikalische Wirkung zu gestalten. Diese *live* vorführen.
- Ideen für Projekte:

1. (Standard: jedes Vorlesungsthema, siehe auch Übungsaufgaben)
2. Verknüpfung von zwei verschiedenen Themen
3. „Hacks“, d.h., Verwendung einer Methode/eines Werkzeugs zu einem nicht bestimmungsgemäßen Zweck
4. Verknüpfung mit Themen aus anderer Vorlesung, z.B. Robotik
5. Bezug zu Leipzig, z.B.
   - J. S. Bach, H. Riemann,
   - Lipsi `https://de.wikipedia.org/wiki/Lipsi_(Tanz)`,
   - Musikautomaten `https://mfm.uni-leipzig.de/dt/dasmuseum/Publik_6onlinepub.php`

- Projekt besteht aus Bericht und Vorführung. Je 2 (bis 3) Personen sollen zusammenarbeiten. Bis KW 51 Gruppen/Themen nennen, bis KW 54 Abstract und Gliederung vorlegen. Zu Vorlesungsende abzugeben sind Bericht (PDF) sowie Arbeitsversionen der Quelltexte und Audiodateien. Können bis Vorführung noch überarbeitet werden. Im Bericht sind individuelle Beiträge zu markieren, pro Person ca. 5 Seiten.

- Bewertet werden

  1. Plan: was soll stattfinden, wie soll es wirken?
  2. Inhalt: Bezug zu Themen, Methoden, Werkzeugen aus der Vorlesung, ggf. durch eigene Recherchen ergänzt
  3. Form: wissenschaftliches Schreiben, vgl. Simon Peyton Jones: `https://www.microsoft.com/en-us/research/academic-program/write-great-res`
  4. Technik/Vorführung: stimmt mit Beschreibung überein, wurde geübt, ohne Verzögerungen präsentiert, Präsentation ist nachvollziehbar (Quelltexte, Befehle, Eingaben/Ausgaben sind live sichtbar)

- (WS18) Vorführung zu geeigneter Zeit an geeignetem Ort (Beschallungs- und Video-Technik, Getränke, Gäste). Vorschläge?

  (WS20) Corona! Also livestream.

# 11 Rhythmus, Breaks, Sampling

**Motivation, Plan**

- technisch (Tidalcycles):
  - Wiederholung: Konstruktion von zeitlicher Struktur
    parallel (stack), verschränkt (cat)
  - neu: Modifikation von Struktur (every, rotate)
- Ziel, Anwendung: Analyse und Synthese von Rhythmen in der Musik, besonders: Tanzmusik

  was klingt gut (Beispiele) und warum (Gründe)
- was gut klingt, wird zitiert (gesamplet)

### Ungerade Rhythmen

- welche Rhythmen kommen in der (europ.) Praxis vor?

  4/4 (Polka, Rock'n'Roll), 3/4 (Walzer), aber nicht nur!
- Zwiefacher

  `https://www.br.de/mediathek/video/woher-kommt-der-zwiefache-verzwickter-tar`
  `584f862a3b467900119cdb27` (6 min: ein Zwiefacher über den Zwiefacher)
- Tanz = rhythmische Bewegung, aber ... `https://www.br.de/mediathek/video/`
  `alpha-retro-gestatten-sie-la-bamba-rheinlaender-wiggle-twist-av:`
  `5e848ef5eb0248001cd940da` (24 min 24 s)

### Ungerade Rhythmen – Hörbeispiele

- Norma Tanega: *You're Dead*, 1966
- Paul Desmond (rec. Dave Brubeck): *Take Five*, 1959

```
stack [ s "superpiano"
       >| n "[[~ e4'min7] ~ e4'min7 ~ b3'min7 ]"
  , s "gretsch:4" >| gain "1*10"
  , s "sn:4"      >| gain "0 1 1 0 1"
  , s "clubkick"  >| gain "1 0 0 1 0"   ]
```

- Lalo Shifrin: *Mission Impossible* (Theme) 1966

- Billy Goldenberg: *Kojak* (Theme) 1973
- Erich Ferstl: *Alpha Alpha* (Thema) 1972
- Blind Idiot God: *Slackjaw* (1992)

**Weitere zeitliche Operatoren in Tidal**

- schneller (fast), langsamer (slow),

```
fast, slow :: Pattern Time
  -> Pattern a -> Pattern a
```

- rückwärts (rev) — Samples laufen trotzdem vorwärts
- verschieben: später ~>, früher <~,
  Bsp: `"0 0 0 0.1 0" ~> "1 0 0 1 0"`
- Anwendung: `(swingBy 0.2 5 $ gain "1*10")`

```
swingBy x n =
   inside n (withinArc (Arc 0.5 1) (x ~>))
inside n f p = fast n $ f $ slow n p
```

**Operatoren zweiter Ordnung**

- nur anw., wenn Zyklus-Nummer die Bedingung erfüllt

```
when :: (Int -> Bool)
     -> (Pattern a -> Pattern a)
     ->  Pattern a -> Pattern a
```

- implementiere damit `every  :: Pattern Int->(Pattern a->Pattern a)->Pattern a->P`
- nur auf den rechten Kanal anwenden

```
jux  :: (Pattern ControlMap -> Pattern ControlMap)
     -> Pattern ControlMap -> Pattern ControlMap
```

- diskutiere: ganz außen `jux rev` — welche Patterns überleben das? (mit 1. ganz einfacher, 2. ganz beliebiger, also gar keiner Struktur)

### Synkopen

- Def: unerwartete Verschiebung (der Betonung) von Ereignissen in einem Muster

```
gain "[1 0.7]*4"
gain "[1 0.7 0.7 1 0.7 0.7 1 0.7]"
```

- Konstruktion von synkopischen Mustern:
  - mit den gezeigten Operatoren (deswegen wurden sie definiert)
  - nach anderen mathematischen Prinzipien,
  - (später) systematisch oder zufällig kombiniert (algorithmische Komposition, Aleatorik)
- empirische Analyse von Rhythmen/Synkopen

### What Makes a Good Rhythm Good?

- Godfried T. Toussaint, *The rhythm that conquered the world*, 2011 `http://cgm.cs.mcgill.ca/~godfried/rhythm-and-mathematics.html` clave son: $[1, 0, 0, 1,\ 0, 0, 1, 0,\ 0, 0$
- die erste Hälfte ($s = 10010010$) ist eine *maximal gleichverteilte* Bitfolge (mit $e = 3$ Einsen und Länge $n = 8$)

  Def: für alle $k$ gilt: für alle Teilfolgen $u, v$ von $s^\omega$ der Länge $k$ gilt: $|u|_1 - |v|_1 \in \{-1, 0, 1\}$ (mit $|u|_1 =$ Anzahl der 1 in $u$)
- Satz: für alle $0 \leq e \leq n$ existieren solche Folgen

  Beweis: Konstruktion ähnlich zu Euklidischem Algor.
- Notation in Tidalcycles: API: `euclid 3 8 1`, innere Sprache: `"1(3,8)"`.

### Syncopation . . . in Groove Music

- Witek et al.: *Syncopation, Body-Movement and Pleasure in Groove Music*, 2014 (Text S2, Figure S6) `https://doi.org/10.1371/journal.pone.0094446`

  exakt beschrieben ist das nicht, aber man kann raten:
  - Zeitpunkt $e = p/2^k$ hat metrisches Gewicht $w(e) = -k$

  - Synkope = benachbarte $e_1$ (Note) und $e_2$ (Pause oder Note auf anderem I.) mit $w(e_1) < w(e_2)$ , warum?

- die naheliende Frage ist dann: Muster mit gegebenem Synkopations-Grad automatisch erzeugen

  1. irgendwie, 2. durch (Tidal-)Operatoren

**Amen Brother**

- The Winstons: *Amen Brother*, 1969.

  (Schlagzeug: Gregory Sylvester Coleman) Break: 1:25

  [name=amen,program=abcm2ps,options=-O=] X: 1 K: F M: 4/4 L: 1/16 F2F2 c3c z c FF c3c — F2F2 c3c z c FF c3c — F2F2 c3c z c F2 z2 c2 — z c FF c3c z c F2 z2 c2 —

  angeblich „most sampled track in the history of music“

- weitere Beispiele für *break beat*-Drumming

  - The Meters (dr: Ziggy Modeliste), Bsp: *Cissy Strut* 1969
  - Booker T and the MGs (dr: Al Jackson Jr), Bsp: *Melting Pot* 1971

**Überblick Sampling**

- Def: Sample = Amplitudenverlauf eines Audio-Signals
- Anwendung 1: Optimierung der Synthese
  - Klänge im Voraus synthetisieren, abspeichern,
  - wenn Software-Simulation des physikalischen Systems nicht in Echtzeit möglich ist
  - Bsp: Synclavier (1977, 32 MB Speicher) `http://www.vintagesynth.com/misc/synclav.php`
- Anwendung 2: musikalische Aussage
  Audio-Signal wird aus erkennbarer Quelle *zitiert*:
  - einzelner Klang (eines bestimmten Instrumentes)
  - *zusammenhängende* Klänge (Teil eines Musikstückes)

### Samples in Tidal-Cycles

- Sample als Audio-Datei (wav), wird bei Start von Supercollider geladen, siehe auch `https://tidalcycles.org/Custom_Samples`
- `d1 $ s "breaks152"` spielt das Sample ab
  - in Original-Länge $l$ und -Tempo
  - beginnend zu jeder (!) Tidal-Periode $p$

  d.h. auch selbst-überlappend, falls $l > p$
- ```
  stack [ fast 4 $ s "hc"
    , s "breaks152" # begin 0 # end 0.3 # speed 0.9 ]
  ```
  begin und end sind relativ zu $l$,

  fast und speed sind unabhängig

### Übung

1. weitere Beispiele für ungerade Takte in der Pop/Rockmusik analysieren

   Go-Betweens: *Cattle and Cane*, 1983.

   Soizweger Zwoagsang *Ja wer koan Zwiefachn ka*, 2015. `https://www.br.de/mediathek/video/soizweger-zwoagsang-ja-wer-koan-zwiefachn-ka-av:5a3c777c185c080018d22d08`
2. Das Bjorklund-Verfahren, siehe `https://hackage.haskell.org/package/tidal-1.6.1/docs/src/Sound.Tidal.Bjorklund.html` und Papers von G. Toussaint.

   Beispiel $E(5, 8)$.

   Begründen Sie, daß die Implementierung die Spezifikation (im Skript) erfüllt.
3. bestätigen Sie die angegebenen Vorkommen von $E(k, n)$ z.B. bei lateinamerikanischen Rhythmen.

   Hörbeispiele Bossa Nova:

   - Stan Getz/Joao Gilberto 1964,
   - Quincy Jones: Big Band Bossa Nova 1962;
  ```

- Senor Coconut (Uwe Schmidt, Atom TM): El Baile Aleman, 2000.

4. Arbeiten mit (eigenen) Samples in Tidal, vgl. `https://tidalcycles.org/Manipulating_samples` experimentieren Sie mit `breaks165`, `bev` (wie angegeben), `led` (schwierig)

5. implementieren Sie Ideen aus: Nick Collins: *Algorithmic Composition Methods for Breakbeat Science*, 2001 `https://composerprogrammer.com/research/acmethodsforbbsci.pdf`

6. autotool-Aufgaben zu Euklidischen Rhythmen (gleichmäßige Verteilung von Ereignissen/Zahlen in einem Raster).

   Entwickeln Sie eine Theorie für den zweidimensionalen Fall, vgl. `https://gitlab.imn.htwk-leipzig.de/autotool/all0/issues/562`

# 12 Algorithmische Komposition

**Motivation**

- klassische Partitur beschreibt das Musikstück *extensional* (durch Angabe der zu spielenden Töne)
- jetzt: *intensional* (durch Angabe einer Vorschrift (Algorithmus) zur Bestimmung der zu spielenden Töne )
- z.B. algebraische Ausdrücke (par, seq)
- jetzt auch: *randomisierte* Algorithmen zur Komposition
- Aufführung durch Maschinen *oder Menschen*
- Quelle (Übersicht): Gerhard Nierhaus: *Algorithmic Composition*, Springer 2009, `https://gerhardnierhaus.com/books-on-ac`

**Geschichte der Alg. Komposition (Beispiele)**

- mit Würfeln und Tabellen:
  - Johann Philipp Kirnberger: *Der allzeit fertige Menuetten- und Polonaisen-Komponist*, 1757

  - Carl Philipp Emanuel Bach: *Einfall einen doppelten Contrapunct in der Oktave von sechs Tacten zu machen ohne die Regeln davon zu wissen*, 1758

- mit Rechenmaschinen
  - Lejaren Hiller, Loenard Isaacson: *Illiac Suite*, 1955
- das Ziel ist hier immer die Nachahmung bekannter Musikstile

**Geschichte der Alg. Komposition**

- hier geht es um wirklich neue Musik:
- Iannis Xenakis: `http://iannis-xenakis.org/`

  *Metastasis*, 1955; Buch *Formalized Music — Thought and Mathematics in Music*, 1963,
- Gottfried Michael Koenig `http://www.koenigproject.nl/`, *Projekt 1* 1964, *Projekt 2* 1966, *Sound Synthesis Program* 1971

  RU-Prinzip: inspiriert von der Unwiederholbarkeit von Reihenelementen („unregelmäßig“) einerseits und den gruppenbildenden Multiplikations-reihen („regelmäßig“) andererseits.

  Beispiele: `http://koenigproject.nl/music-excerpts/`

**Komposition und Constraints**

- die Kompositions-Aufgabe: bestimme eine (bestmögliche) Partitur, die diese Bedingungen (Constraints) erfüllt:
  - Randbedingungen
    (Anzahl Stimmen, Tonart, Metrum, Anzahl Takte)
  - musikalische Regeln (keine Dissonanzen, Parallelen)
  - ggf. Ähnlichkeit zu Vorlagen
    (Bsp: eine Fuge im Stil von Bach)
  - ggf. Vermeidung der Ähnlichkeit zu Vorlagen
    (Bsp: eine Fuge, aber anders als die vorige)
- maschinelle Lösung dieser Aufgabe durch
  - exakte Verfahren (vgl. VL Constraint-Programmierung)
  - statistische Näherungsverfahren (sog. maschinelles Lernen)

**Modelle für musikalische Eigenschaften**

- Constraint: Häufigkeiten aufeinanderfolgender Töne

  Modell: stochastischer endlicher Automat (Markov-Prozeß)

- Constraint: Häufigkeiten globaler Strukturelemente

  Modell: stochastische generative Grammatik

- Constraint: spannendes Verhältnis zwischen mehreren Stimmen

  Modell: Zweipersonenspiel

- Constraint: Regelmäßigkeit ohne Wiederholungen

  Modell: zellulärer Automat, Lindenmayer-System

**Algorithmische Komposition und Kreativität?**

- wenn die Kompositionsarbeit scheinbar durch einen Computer übernommen wird — welche Rolle haben: der Komponist? der Interpret? der Hörer?
- der kreative Vorgang ist: Komponist schreibt das Programm (wenigstens: wählt Programme aus und stellt die Parameter ein)
- bei *live coding* ist das ein zentraler Aspekt (Publikum sieht den Bildschirm des Komponisten)
- vgl. aber Joseph Schillinger: *The Mathematical Basis of the Arts*, 1943. (S. 17: fünf Erscheinungsformen der Künste) `https://archive.org/details/TheMathematicalBasi`

**Pseudozufall in Tidalcycles**

- Pseudozufallsgröße im Intervall $[0, 1]$

```
rand :: Fractional a => Pattern a
rand = Pattern (\(State a@(Arc s e) _) ->
 [Event (Context []) Nothing a
  (realToFrac $ (timeToRand ((e + s)/2) :: Double))])
```

- Transformation auf Intervall $[l, r]$ durch `range l r rand`
- ganze Zahlen $[0, 1 \ldots m - 1]$ durch `irand m`

- Auswahl aus einer Liste (mit möglicher Implementierung)

```
choose :: [a] -> Pattern a
choose xs = (xs !!) <$> irand (length xs)
```

**Stetige und diskrete Muster**

- `rand`, `irand` und daraus konstruierte (`choose`) Muster sind stetig (continuous): haben für jeden (reellen) Zeitpunkt einen Wert,
- `ControlPattern` muß immer diskret sein,

  denn OSC-Nachrichten sind diskret.

  Diskretisierung durch: `segment 4 rand`
- pseudo-zufällige Melodie (links z.B. `s "superpiano"`)

```
... >| n (segment 4 $ irand 12)
... >| n (scale "major" $ segment 4 $ irand 7)
```

- pseudo-zufällige Akkordfolge (mit falscher Stimmführung)

```
... >| n (scale "major" $ segment 4 $ irand 7)
    |+ n "[0,2,4]"
```

**Kampf dem Determinisimus**

- der „zufällige Wert“ ist Funktionswert des Mittelpunktes des abgefragten Intervalls:

  gleiche Fragen → gleiche Antworten

```
queryArc
  (stack [segment 2 $ irand 8, segment 2 $ irand 8])
  (Arc 0 1)
[[](0>½)|5,[](½>1)|1,[](0>½)|5,[](½>1)|1]
```

- zur parallelen Komposition unabhängiger Stimmen:

  die Zeit für den Generator lokal verschieben

```
stack [ segment 2 $ irand 8
      , segment 2 $ rotR 1 $ irand 8 ]
```

**Stochastische Sprachen**

- (klassische) Sprache über Alphabet $\Sigma$ ist Abbildung $L : \Sigma^* \to \{0, 1\}$ (die Zweiermenge)
- stochastische Sprache über $\Sigma$ ist Abbildung $L : \Sigma^* \to [0, 1]$ (das Intervall reeller Zahlen)

  $L(w) \approx$ die Wahrscheinlichkeit, mit der $w \in L$
- Plan: stochastische Sprache

  für $\Sigma =$ Elementar-Ereignisse (z.B. Noten)

  - so definieren, daß interessante $w \in \Sigma^*$ hohe Wahrscheinlichkeit haben
  - durch endliches Objekt (Automat, Grammatik) repräsentieren

**Stochastische Automaten, Markov-Prozesse**

- ein endlicher stochastischer Automat $A$ besteht aus:
  - Zustandsmenge $Q$
  - Initialvektor $I \in (Q \to [0, 1])$,
  - Transitionsmatrix $T \in (Q \times Q \to [0, 1])$.

  wobei $I$ stochastischer Vektor ($\sum_{q \in Q} I(q) = 1$)

  und $T$ stoch. Matrix (jede Zeile ist stoch. Vektor)
- stochastischer Prozeß erzeugt Wort $w = q_0 q_1 \ldots q_n \in Q^n$:

  wähle $q_0 \in Q$ nach Verteilung $I$,

  wähle $q_{k+1} \in Q$ nach Verteilung $T(q_k)$.
- Anwendungen in der Musik: $Q =$ Noten, $Q =$ Akkorde.
- dabei wird aber die globale Struktur (nach Riemann: die Funktion der Akkorde) ignoriert!

### Stochastische Grammatiken

- Kontextfreie Grammatik beschreibt Satzbau von natürlichen (und künstlichen) Sprachen *und Musik*

  Jeder Satz hat Subjekt und Prädikat $\approx$ jede Kadenz beginnt und schließt mit Tonika.

- stochastische CFG: wie klassisch, zusätzlich zu jeder Variablen $l$ ein Wsk-Vektor $\vec{v}$ über alle Regeln, $[(v_1, l \to r_1), \ldots, (v_n, l \to r_n)]$

- Bsp: mittlere Wortlänge für $[(1/2, S \to b), (1/2, S \to aSS)]$

- Ü: Beziehung stoch-CFG/Markov-Prozesse

  (MP $\approx$ endlicher Automat = Typ-3-Grammatik $\subseteq$ Typ-2-Grammatik)

### Kombination stochastischer Methoden

- Donya Quick and Paul Hudak: *Grammar-Based Automated Music Composition in Haskell*, `http://functional-art.org/2013/quick.pdf`, Workshop on Functional Art, Music, Modeling and Design (FARM)

### Lindenmayer-Systeme

- D0L-System besteht aus

  Axiom $S \in \Sigma$, Funktion $\phi : \Sigma \to \Sigma^*$ mit $S \sqsubset \phi(S)$

  $F = (0, \{0 \mapsto 01, 1 \mapsto 0\}), \quad G = (a, \{a \mapsto abc, b \mapsto ac, c \mapsto b\})$

- Aristide Lindenmayer (1925–1989), `http://algorithmicbotany.org/papers/#abop`

- definiert Folge $S, \phi(S), \phi^2(S), \ldots$ mit $i < j \Rightarrow \phi^i(S) \sqsubset \phi^j(S)$

  d.h. Folge hat Limes $\phi^\omega(S) \in \Sigma^\omega$

- $w_F = 0100101\ldots$, $w_G = abcacbabcbacabc\ldots$

- Satz: $w_G$ ist *quadratfrei*, d.h., enthält kein Teilwort $uu$.

- Anwendung: Tonfolge konstruiert aus $w_G$ enthält keine benachbarten Wiederholungen

**Regelmäßige Unregelmäßigkeit: Vuza-Kanons**

- Def: A Vuza canon is a non periodic factorization of the cyclic group $\mathbb{Z}_N$ in two factors.
  - $\mathbb{Z}_N = \mathbb{Z} \pmod N$, Bsp: $\mathbb{Z}_6 = \{0, 1, 2, 3, 4, 5\}$, $3 + 4 = 1$
  - *factorization* bedeutet hier: $\mathbb{Z}_N = A \oplus B$ für $A, B \subseteq \mathbb{Z}_n$ und alle Einzelsummen verschieden
    Bsp: $\mathbb{Z}_6 = \{0, 3\} \oplus \{0, 1, 2\}$
  - Menge $M$ ist *non periodic*: $M \oplus x = M \iff x = 0$
    Bsp: $M = \{0, 3\}$ ist periodisch, denn $M \oplus \{3\} = M$.

  Es gibt keine solche Zerlegung von $\mathbb{Z}_6$, aber von $\mathbb{Z}_{72}, \ldots$
- musikalische Anwendung: $A =$ Impuls (Rhythmus), $B =$ Verschiebungen (Einsätze im Kanon).
- D. Vuza 1991 ... *regular* ... *unending canons*, Franck Jedrzejewski 2013 `https://arxiv.org/abs/1304.6609`

**Übung**

1. Algorithmische Komposition in Euterpea,

   Beispiel: `http://www.donyaquick.com/interesting-music-in-four-lines-of-c`

2. Implementieren Sie Steve Reich: Piano Phase (1967).

   Aufführung (Peter Aidu, 2006, solo!) `https://archive.org/details/top.09`,

   Beschreibung: `https://en.wikipedia.org/wiki/Piano_Phase#First_section`

3. Stochastische Musik:

   `https://gitlab.imn.htwk-leipzig.de/waldmann/cm-ws18/blob/master/kw48/stoch.hs`

   zufällige Permutation: Implementierung vervollständigen

4. deterministische nichtperiodische Musik

   benutzen Sie `https://gitlab.imn.htwk-leipzig.de/waldmann/cm-ws18/blob/master/kw48/dnp.hs`

# 13 Automatische Musik-Analyse

### Motivation, Überblick

- – Synthese: Modell $\Rightarrow$ Klang, Musikstück
  – Analyse: Modell $\Leftarrow$ Klang, Musikstück
- Aspekte:
  – Spektrum eines Klangs
  – Tempo, Akkordfolge, Struktur eines Musikstücks
- Anwendungen:
  – Wiedererkennen eines Musikstücks
  – Zuordnung zu Epoche, Stil, Komponist
  – Vorschlagen (bekannter) ähnlicher Stücke
  – Synthese (neuer) ähnlicher Klänge, Stücke
  – Audio-Kompression

### Werkzeuge zur Audio-Analyse (Bsp.)

- The Vamp audio analysis plugin system `https://www.vamp-plugins.org/`

  ```
  sonic-annotator
    -d vamp:qm-vamp-plugins:qm-tempotracker
    foo.wav   -w csv --csv-stdout
  ```

  (auch für sonic-visualiser, audacity)

- Centre for Digital Music, Queen Mary Univ. London: QM Vamp Plugins `https://vamp-plugins.org/plugin-doc/qm-vamp-plugins.html`

  vamp:qm-vamp-plugins:qm-barbeattracker, ..:qm-segmenter:segmentation, ..:qm-tempotracker:tempo

- Matthias Mauch: Chordino und NNLS Chroma `http://www.isophonics.net/nnls-chroma`

  vamp:nnls-chroma:chordino:simplechord

**Spektral-Analyse (DFT)**

- grundsätzlich (Wdhlg): Darstellung einer periodischen Funktion $f : [0, 1] \to \mathbb{C}$ als $\sum_{k\in\mathbb{N}} c_k(t \mapsto \exp(2\pi ikt))$
- Anpassung für zeitdiskrete Signale (d.h., Vektoren) $x : [0, 1 \ldots n-1] \to \mathbb{C}$

  Darstellung (Dekodierung) $x_t = (1/\sqrt{n}) \sum_{k=0}^{n-1} c_k \exp(2\pi ikt/n)$

  $x = M \cdot c$ mit Matrix $M_{t,k} = (1/\sqrt{n}) \exp(2\pi ikt/n)$
- Koeffizientenvektor bestimmen aus $M^{-1}x = c$

  Satz: $M^{-1}_{t,k} = (1/\sqrt{n}) \exp(-2\pi ikt/n)$

  Bew: $(M^{-1} \cdot M)_{p,r} = (1/n) \sum_q \exp(-2\pi ipq/n) \exp(2\pi iqr/n)$
- Kodierung: $c_k = (1/n) \sum_{l=0}^{n-1} x_l \exp(-2\pi ikl/n)$

**Anwendung zur Audio-Analyse**

- Signal in Blöcke zerlegen

  (z.B. Samplerate 44.1 kHz, jeder Block 512 Samples, Blocklänge ist dann 12 ms, Blockfrequenz 86 Hz)
- DFT für jeden Block einzeln, ergibt Funktion

  Block-Index $\times$ Frequenz-Index $\to$ Koeffizient (in $\mathbb{C}$)

  (Spektrogramme in sonic-visualiser)
- Implementierung: $M \cdot c$ (Matrix mal Vektor) schneller ausrechnen unter Ausnutzung der Struktur von $M$

  (fast Fourier transf., Cooley und Tukey 1965, Gauß 1805)

  `https://gitlab.imn.htwk-leipzig.de/waldmann/cm-ws18/tree/master/dft`

**Audio Fingerprinting**

- Avery Wang: *An Industrial Strength Audio Seach Algorithm*, ISMIR 2003, `http://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf`
- Musikstück $\to$ Spektrogramm $\to$ Fingerabdruck (FP)

- Vergleich des FP eines unbekannten Musikstücks mit (vielen) bekannten FP aus Datenbank
- gewünschte FP-Eigenschaften: temporally localized, translation invariant, robust, sufficiently entropic
- Implementierung: Spektrogramm-Spitzen, Ankerpunkte, FP ist Menge der Differenz-Vektoren zu nahen Punkten.
- schnelle Erkennung von Diagonalen im Diagramm (Scatterplot) der Diff.-V. in Anfrage/in Datenbankeintrag

**Verlustbehaftete Audio-Kompression**

- Original ist Funktion $\mathbb{R} \to \mathbb{R}$, i.A. keine endliche Repräsentation möglich, wird nur erreicht durch:
  - Zeit-Raster durch Abtastung (Sampling) (z.B. 44.1 kHz)
  - Wert-Raster durch Zahlendarstellung (z.B. 16 Bit)
- Platzgewinn durch geringere Samplefreq., ger. Bitbreite.
- bei vorgegebenem Platz: höhere Qualität ist möglich durch Ausnutzung physiologischer Eigenschaften
- MP3, AAC, Opus, ...: Fourier-Transf. $\to$ Darstellung der Koeffz. mit zeit- und frequenz-abhängiger Bitbreite

  Europ. Broadcasting Union: AC-3, `https://www.etsi.org/deliver/etsi_ts/102300_102399/102366/`

**Parameterbestimmung für Markov-Modelle**

- Wdhlg: Markov-Prozeß $A$ mit Zuständen $Q = \{1, \ldots, n\}$
  - gegeben durch Initialvektor $I_A : Q \to [0, 1]$
  - und stochastische Transitionsmatrix $T_A : Q \times Q \to [0, 1]$.

  definiert Funktion $f : Q^* \to [0, 1]$
- suchen Verfahren zur Lösung der Aufgabe:

  - gegeben: Wertepaare $g = \{(w_1, y_1), \dots\}$
  - gesucht: $A = (I_A, T_A)$ mit $g(w) \approx f_A(w)$.
- Variante: gegeben: Mengen $P, N \in Q^*$,
  gesucht: $A$ mit $\forall w \in P : f_A(w) \gg 0$, $\forall w \in N : f_A(w) \approx 0$.
  Bsp: $P$: Takte aus Melodien von Bach, $N$: ... Beatles.

**Parameterbest. durch Gradientenabstieg**

- geg.: (differenzierbare) Zielfunktion $f : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$
  ges.: ein $x \in \mathbb{R}^n$ mit $f(x)$ minimal
- Anwendung: $(I_A, T_A) \mapsto \sum_i (g(w_i) - f_A(w_i))^2$
  falls min. Wert 0, dann löst $A$ die Aufgabe (vorige Folie)
- Verfahren: bestimme Gradient $\nabla f = (\partial f / \partial x_i)_i$
  (durch *automatische Differentiation*, `https://www.imn.htwk-leipzig.de/~waldmann/edu/ss18/ki/folien/#(202)` )
  verändere Kandidaten $x$ um $-c \cdot \nabla f$.
- das wird derzeit gern *tiefes Lernen* oder gar *KI* genannt.
- für diese Anw.: $I_A$ und Zeilen von $T_A$ stochastisch:
  Ansatz $J$ beliebig, $I_p = J_p^2 / (\sum_q J_q^2), \dots$

**Versteckte Markov-Modelle**

- stochastischer Automat $A = (I, T)$ mit Zuständen $Q$
  mit stoch. Ausgabe-Matrix $O : Q \times \Sigma \to [0, 1]$
- definiert Funktion (Wsk.) $h : \Sigma^* \to [0, 1]$
  $h(w) = \sum_{q_1 \dots q_k \in Q^{|w|}} f_A(q_1 \dots q_k) \cdot O(q_1, w_1) \dots O(q_k, w_k)$
-
  - $Q$ : Laute einer natürlichen Sprache
  - $A$ : Häufigkeit von Laut-Folgen in Menge von Wörtern
  - $\Sigma$: Merkmals-Vektoren von Spektrogramm-Abschnitten
  - $h$: Realisierung von Lauten

- Aufgaben (allgemein):
  - geg: $A, O, w \in \Sigma^*$, ges.: der größte Summand in $\sum_{q_1 \ldots q_k}$
  - geg: Wertepaare aus $h$, ges.:$A, O$

**Bestimmung harmonischer Strukturen**

- José Pedro Magalhães, W. Bas de Haas: *Functional Modelling of Musical Harmony*, ICFP 2011 `https://github.com/haas/harmtrace`

  „Given a sequence of chord labels, the harmonic function of a chord in its tonal context is automatically derived."

- Spektral-Analyse $\rightarrow$ chord labels, diese jedoch fehlerbehaftet. Wie reparieren?

- benutze Baumstruktur, beschrieben durch Grammatik

  und fehlerkorrigierenden Parser (Swierstra 2009) `https://hackage.haskell.org/package/uu-parsinglib`

- `harmtrace parse -g pop -c "C:maj C:maj  F:maj G:maj C:maj"`

**Übung**

- Diskutieren Sie das Geschäftsmodell (wer bezahlt womit wofür?) von kommerziellen Musikerkennungsdiensten. Siehe Abschnitt 1 des zitierten Artikels von Wang (2003).

  Was steht im 1. Absatz von Abschnitt 3.3?

- Diskrete Fourier-Transformation:

  `https://gitlab.imn.htwk-leipzig.de/waldmann/cm-ws18/tree/master/dft`

  Vergleichen Sie die akustische Wirkung der Rasterung der Amplituden mit Rasterung der Fourier-Koeffizienten. (Beispiel im README)

  Überprüfen Sie experimentell und beweisen Sie: $M \cdot M^{-1} =$ Einheitsmatrix

  Ändern Sie cm-ws18/dft/Main, so daß das Rastermaß für Koeffizienten geringer und hoher Frequenzen unterschiedlich eingestellt werden kann.

  Extremfall: Koeffz. ab einer gewissen Grenzfrequenz (d.h., ab einem gewissen Index, z.B. halbe Fensterbreite) durch 0 ersetzen.

- Stochastische Musik in Tidalcycles: benutzen Sie `https://tidalcycles.org/index.php/markovPat`

  Schreiben Sie Übergangsmatrizen für Melodien (Tonhöhenanderungen bevorzugt in kleinen Schritten) und Baß (bevorzugt größere Sprünge). Töne sollen aus vorgegebener Skala sein oder zu vorgegebener Akkordfolge passen.

  Erzeugen Sie eine globale Struktur durch Kombination von Markov-Mustern verschiedener Frequenz.

- Ton- und Akkord-Analyse:

  ausprobieren, Ausgabe von chordino als Eingabe für Sequencer verwenden

  siehe `https://gitlab.imn.htwk-leipzig.de/waldmann/cm-ws18/blob/master/README.md#vamp-plugins-sonic-visualiser-sonic-annotator`

- Beispielprogramm zur Bestimmung eines Markov-Modells `https://gitlab.imn.htwk-leipzig.de/waldmann/cm-ws18/tree/master/learn-markov`

# 14 Analyse von Rhythmen

**Beispiele, Anregungen**

- Grosvenor Cooper and Leonard B. Meyer, *The Rhythmic Structure of Music*, 1960 `https://press.uchicago.edu/ucp/books/book/chicago/R/bo24515499.html`

- Godfried T. Toussaint, *Musical Rhythm and Mathematics*, `http://cgm.cs.mcgill.ca/~godfried/rhythm-and-mathematics.html`

- Mike Adamo, *the breakbeat bible*, 2010, `https://www.mikeadamo.com/the-breakbeat`

- Florent Jacquemard `https://jacquema.gitlabpages.inria.fr/` *A Supervised Approach for Rhythm Transcription...* `https://hal.inria.fr/hal-01315689v2`, *A Structural Theory of Rhythm Notation...*,

# 15 Notensatz

**Übersicht**

- schriftliche Fixierung von Musik zum Zweck der Archivierung und späteren Aufführung

- zweidimensionale Notation:
  - für jede einzelne Stimme (Notensystem)
    - Ordinate (nach oben) (Notenlinie): Tonhöhe
    - Abszisse (nach rechts): Zeit

    Zeit auch durch Form der Noten (Köpfe, Hälse)
  - für mehrere Stimmen:
    - Notensysteme übereinander bedeutet Gleichzeitigkeit
- Notensatz mit Computer:
  *grafisch* (WYSIWIG) oder *programmatisch* (algebraisch)

**Notensatz mit Lilypond**

- kompiliert algebraische Musik-Beschreibung (Programmtext) zu grafischer Darstellung (PDF) sowie zu MIDI-Datenstrom
- Warum? `http://lilypond.org/essay.html`
- Wie? `http://lilypond.org/doc/v2.19/Documentation/contributor-big-page.html#overview-of-lilypond-architecture`
  - Implementierung in C++
  - Anwender-definierte Erweiterungen in LISP (Scheme)
  - Backend benutzt Metafont `https://web.archive.org/web/20110927042453/http://www.tex.ac.uk/ctan/systems/knuth/dist/mf/mf.web`

**Sematik von Lilypond (lokal)**

```
\version "2.18.2" \header { } \score {
  \relative c' { c4 d e f | g a b c }
  \layout { }  \midi { }
}
```

- lokale Struktur (Folge von Noten einer Stimme)
  - Einzelnote (Bsp: `c''4`): Name, Oktave, Zeit

  - fall Zeit nicht angegeben, dann vorigen Wert benutzen
    bei Taktstrich: Zeit muß Vielfaches des Taktes sein
  - relative Notation: immer die nächstliegende Oktave

  damit muß man für Melodien *wesentlich* weniger schreiben als bei Haskore (u.ä.)

- globale Struktur . . . nächste Folie

**Semantik von Lilypond (global)**

- Zusammensetzung von Teil-Partituren:
  - sequentiell: hintereinander
  - parallel: Operator `<< ... >>`

    ```
    \relative c' {c4 d e <<f a c>> | <<g b d>> a b c}
    ```

- Notensysteme (für mehrstimmigen Satz)

```
<< \new Staff{\relative c' { c4 d e f | g a b c }}
  \new Staff {\relative c''{ g  a g f | e d d c }}
>>
```

- Wiederholungen:
  - notiert (Wiederholungszeichen in Partitur): `repeat volta 2 {c1}`
  - expandiert: `repeat unfold 2 {c1}`

**Semantik von Lilypond: Unterprogramme**

- Namen zur Bezeichnung von Teilpartituren

```
foo = \relative c' { c4 d e f | g a b c }
\score { << \new Staff { \foo }
            \new Staff { \transpose c e \foo }
         >> }
```

- vergleichbar zu Unterprogrammen, *aber*
  - Unterprogramme haben keine Argumente

  - Unterprogramme sind global (außerhalb `\score{..}`)
  - es gibt keine Bedingungen und Verzweigungen

  Das ist ein riesengroßer Rückschritt im Vergleich zu Haskore u.ä.

- Work-around: man kann LISP-Code schreiben

**Semantik von Lilypond (Kontexte)**

- `\new Staff`, `\new Voice`, ... legt *Kontexte* an,
- zu jedem Kontext gehören mehrere *engraver*,

  diese verarbeiten *musical expression* zu grafischen Elementen: `Clef_engraver`, `Key_engraver`, `Note_heads_engraver`, ...
- benannter Kontext kann „von außen“ benutzt werden

```
<< \new Staff { \new Voice = "foo" {} }
   \new Staff { \new Voice \relative c' {c4 d e f} }
   \context Voice = "foo" \relative c'{  g a b c }
>>
```

- Anwendung: Text (Silben) unter Melodie (Noten)

**Anwendung von Kontexten**

- Text (Silben) unter Melodie (Noten)
- ```
  << \new Staff { \new Voice = "foo" {} }
     \new Lyrics = "here"
     \context Voice = "foo"
        \relative c' { g a b c }
     \context Lyrics = "here" {
       \lyricsto "foo" { fas- ci- na- ting }
     } >>
  ```
- spezifiziert wird:
  - wo der Text erscheint: `\new Lyrics`
  - nach welcher Stimme er ausgerichtet wird: `\lyricsto`
  ```

**Übung**

1. Lilypond-Beispiele `https://gitlab.imn.htwk-leipzig.de/waldmann/cm-ws18/tree/master/lilypond`
2. einfache Beispiele ausprobieren

   ```
   lilypond basic.ly
   evince basic.pdf
   timidity basic.midi
   ```

3. *Chase the Devil* analysieren, modifizieren, ergänzen,
4. Schlagzeug hinzufügen

   ```
   \new DrumStaff { \drummode { bd4 bd } }
   ```