FunCode — Computergestützte Analyse Funktionalharmonischer Symbolsysteme

Markus Lepper[1] Baltasar Trancón y Widemann[12] Michael Oehler[3]

[1] semantics gGmbH, Berlin, DE

[2] Nordakademie, Elmshorn, DE

[3] Universität Osnabrück, DE

Vortrag im Kolloquium "Angewandte Musiktechnologie und Musikinformatik",
Universität Osnabrück, 2022-11-17

Gliederung

Gliederung

Motivation

- im Kontext unserer Re-Modellierungen kultureller, historisch entwickelter Symbolsysteme:
- Formelschreibweise für **funktionalharmonische Musiktheorie und -analyse**
  (Systeme in Gebrauch sind "folkloristisch" schein-definiert, unter- oder gar nicht spezifiziert.)
  Exakte Semantik benötigt für Diskussionsklärung *und* mögliche digitale Realisierung.

Motivation

- im Kontext unserer Re-Modellierungen kultureller, historisch entwickelter Symbolsysteme:
- Formelschreibweise für **funktionalharmonische Musiktheorie und -analyse**
  (Systeme in Gebrauch sind "folkloristisch" schein-definiert, unter- oder gar nicht spezifiziert.)
  Exakte Semantik benötigt für Diskussionsklärung *und* mögliche digitale Realisierung.
- (politische) **Notwendigkeiten:**
  "konvexe Hülle" möglichst vieler existierender (und relevanter) Varianten
- . . . Parametrisierbarkeit

Motivation

- im Kontext unserer Re-Modellierungen kultureller, historisch entwickelter Symbolsysteme:
- Formelschreibweise für **funktionalharmonische Musiktheorie und -analyse**
  (Systeme in Gebrauch sind "folkloristisch" schein-definiert, unter- oder gar nicht spezifiziert.)
  Exakte Semantik benötigt für Diskussionsklärung *und* mögliche digitale Realisierung.
- (politische) **Notwendigkeiten:**
  "konvexe Hülle" möglichst vieler existierender (und relevanter) Varianten
- . . . Parametrisierbarkeit
- *verständliche / nachvollziehbare* Spezifikations-Sprache
- überzeugender unmittelbarer praktischer Nutzen (soweit möglich !-)

Motivation

- im Kontext unserer Re-Modellierungen kultureller, historisch entwickelter Symbolsysteme:
- Formelschreibweise für **funktionalharmonische Musiktheorie und -analyse**
  (Systeme in Gebrauch sind "folkloristisch" schein-definiert, unter- oder gar nicht spezifiziert.)
  Exakte Semantik benötigt für Diskussionsklärung *und* mögliche digitale Realisierung.
- (politische) **Notwendigkeiten:**
  "konvexe Hülle" möglichst vieler existierender (und relevanter) Varianten
- . . . Parametrisierbarkeit
- *verständliche / nachvollziehbare* Spezifikations-Sprache
- überzeugender unmittelbarer praktischer Nutzen (soweit möglich !-)
- ausführbarer, unmissverständlicher *und zugleich* gut lesbarer Spezifikationstext

Motivation

- im Kontext unserer Re-Modellierungen kultureller, historisch entwickelter Symbolsysteme:
- Formelschreibweise für **funktionalharmonische Musiktheorie und -analyse**
  (Systeme in Gebrauch sind "folkloristisch" schein-definiert, unter- oder gar nicht spezifiziert.)
  Exakte Semantik benötigt für Diskussionsklärung *und* mögliche digitale Realisierung.
- (politische) **Notwendigkeiten:**
  "konvexe Hülle" möglichst vieler existierender (und relevanter) Varianten
- ... Parametrisierbarkeit
- *verständliche / nachvollziehbare* Spezifikations-Sprache
- überzeugender unmittelbarer praktischer Nutzen (soweit möglich !-)
- ausführbarer, unmissverständlicher *und zugleich* gut lesbarer Spezifikationstext
- Ansatz: **Literate Programming / Compound Document** aus
  - Spezifikationsformeln ("möglichst deklarativ" benutztes Prolog)
  - Erläuternd Text und Graphik (LaTeX/ Pgf / lilyPond)
  - Programm-Tests (Prolog)

Gliederung

Funktionalharmonische Analyse

- geht zurück auf Rameau ([1722]1965), Riemann (1882) und viele andere.

Funktionalharmonische Analyse

- geht zurück auf Rameau ([1722]1965), Riemann (1882) und viele andere.
- Grundprinzip: Einem Notentext werden Folgen von interpretierenden Symbolen zugeordnet (= "Labels"), die dessen harmonisches "Funktionieren" explizit machen.

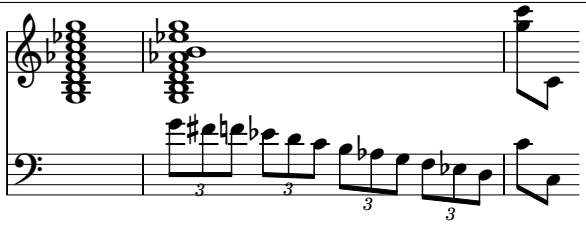
Funktionalharmonische Analyse

- geht zurück auf Rameau ([1722]1965), Riemann (1882) und viele andere.
- Grundprinzip: Einem Notentext werden Folgen von interpretierenden Symbolen zugeordnet (= "Labels"), die dessen harmonisches "Funktionieren" explizit machen.
- oder: Folgen von diesen Symbolen stehen für abstrakte Harmoniefolgen als solche, über die Aussagen möglich sein sollen.

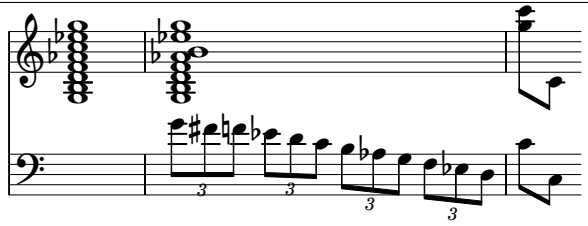
Funktionalharmonische Analyse

- geht zurück auf Rameau ([1722]1965), Riemann (1882) und viele andere.
- Grundprinzip: Einem Notentext werden Folgen von interpretierenden Symbolen zugeordnet (= "Labels"), die dessen harmonisches "Funktionieren" explizit machen.
- oder: Folgen von diesen Symbolen stehen für abstrakte Harmoniefolgen als solche, über die Aussagen möglich sein sollen.
- Beidem liegt eine (mehr oder weniger spezifische) **harmonische Theorie / "Harmonielehre"** zu Grunde.

Konkretes Beispiel



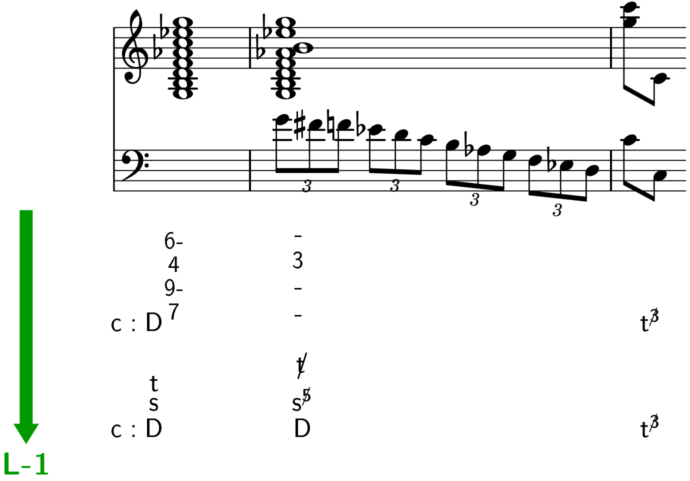
Konkretes Beispiel



	6- 4 9-	- 3 -	
c : D	7	-	t♯3
	t	♯/	
	s	s♯5	
c : D		D	t♯3

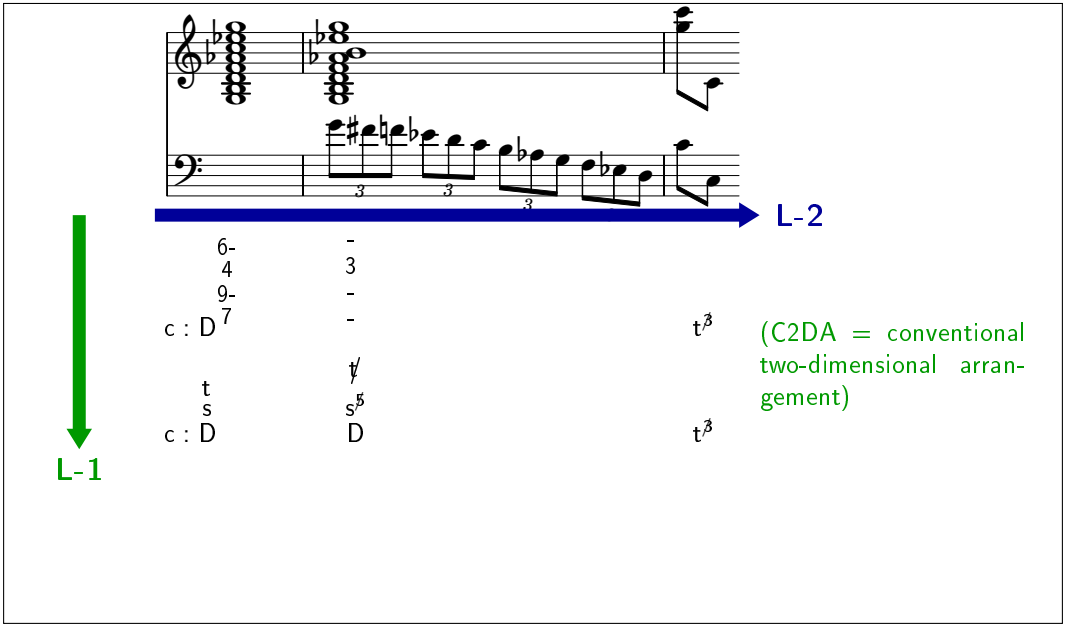
(C2DA = conventional two-dimensional arrangement)

Konkretes Beispiel

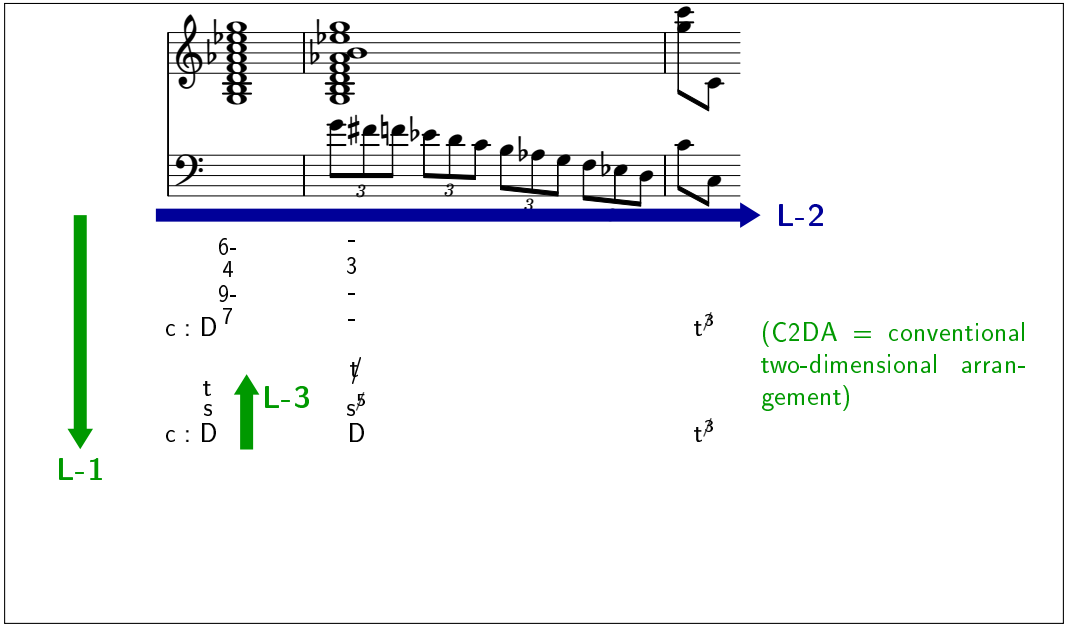


(C2DA = conventional two-dimensional arrangement)

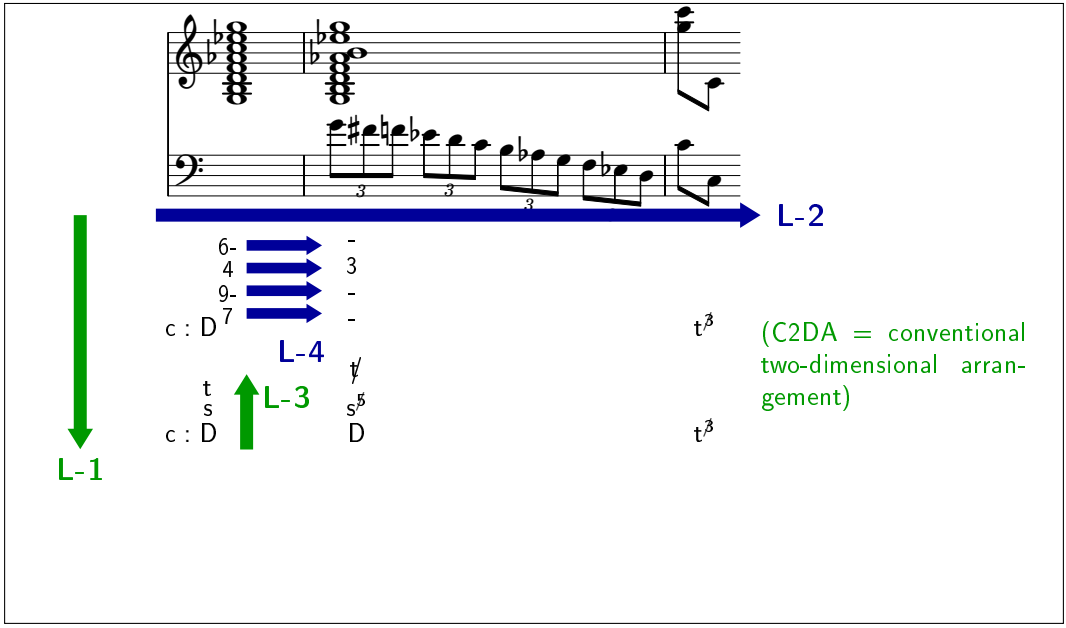
Konkretes Beispiel



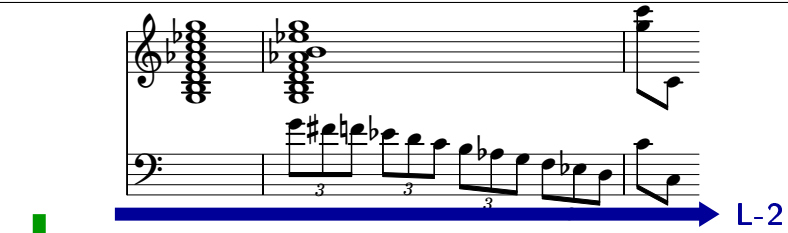
Konkretes Beispiel

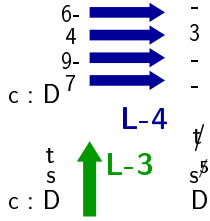


Konkretes Beispiel



Konkretes Beispiel



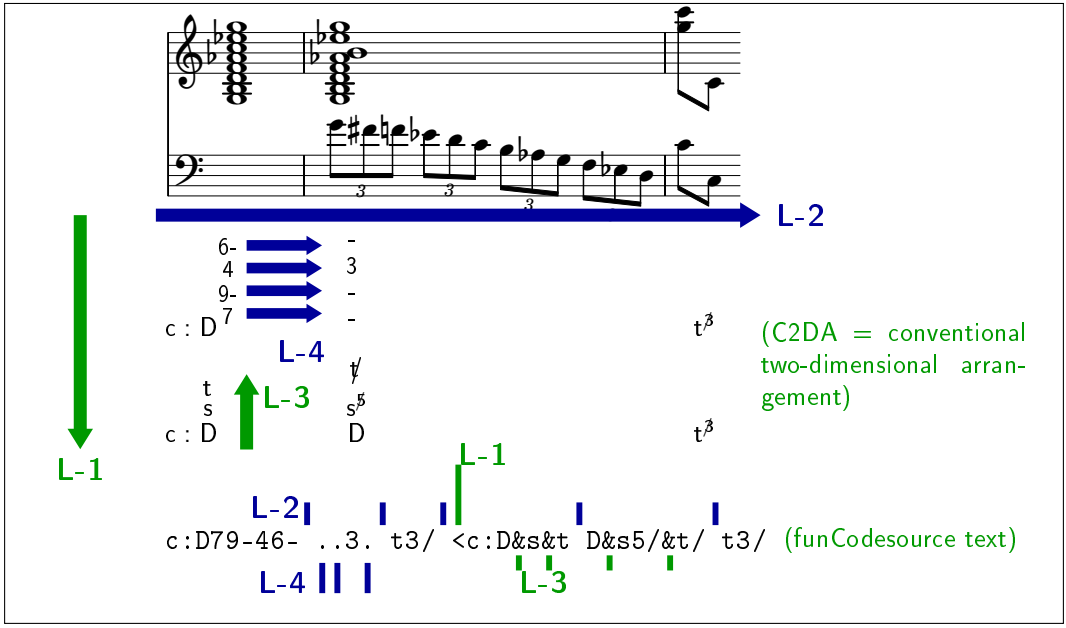


(C2DA = conventional two-dimensional arrangement)

L-1

```
c:D79-46- ..3. t3/ <c:D&s&t D&s5/&t/ t3/
```
(funCodesource text)

Konkretes Beispiel



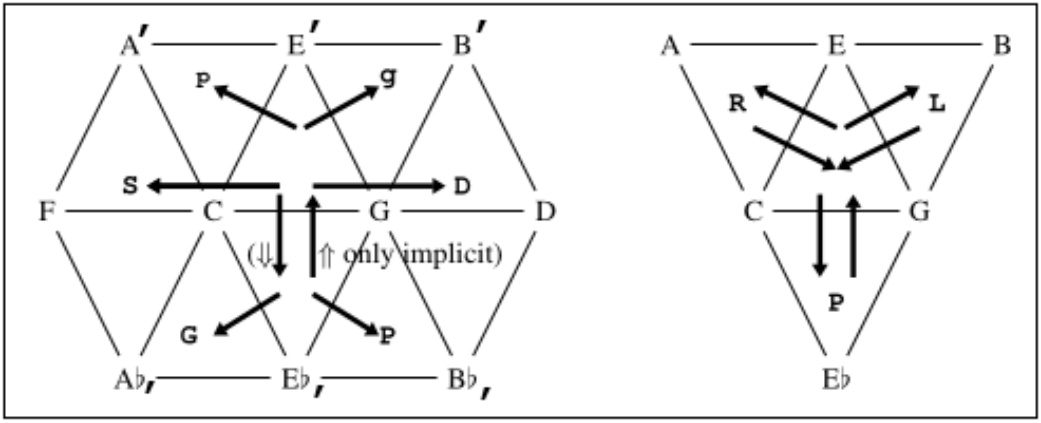
Semantiken funktionalharmonischer Symbolsequenzen

- **verschiedene Semantiken** möglich, üblich und sinnvoll
  (Kompositionsformeln, Suchpatterns, psychologische Situationen, Spannungsgefühle/Tendenzen, etc.)
  (Lepper, Trancón y Widemann und Oehler 2022a)
- Sinnvoll formalisierbar nur "S-4":
  Konkrete Menge von Tonhöhenklassen ("set of pitch classes").
  (Gleichsam: Akkorde modulo Oktavlage.)

Semantiken funktionalharmonischer Symbolsequenzen

- **verschiedene Semantiken** möglich, üblich und sinnvoll
  (Kompositionsformeln, Suchpatterns, psychologische Situationen, Spannungsgefühle/Tendenzen, etc.)
  (Lepper, Trancón y Widemann und Oehler 2022a)
- Sinnvoll formalisierbar nur "S-4":
  Konkrete Menge von Tonhöhenklassen ("set of pitch classes").
  (Gleichsam: Akkorde modulo Oktavlage.)
- Dafür wiederum verschiedene Domänen möglich: Midi Keys, "Spelled Keys".
  Am differenziertesten: **Euler-Netz**
- Koordinate im Euler-Netz = (q, t) = Exponenten "reiner Quinten" = 3/2 und "reiner Dur-Terzen" = 5/4, beides Intervalle der "Partialtonreihe".
- (Aber "reine" Intervalle *nur symbolisch*!
  Sie *beschreiben die Psychologie* auch bei temperierter Stimmung etc.)

Das Eulernetz



Zwei Ebenen der Semantik

- **Grundton** (der Funktion/des Akkordes)
  vs. Menge der klingenden Tonklassen = **Akkordkomponenten**

Zwei Ebenen der Semantik

- **Grundton** (der Funktion/des Akkordes)
  vs. Menge der klingenden Tonklassen = **Akkordkomponenten**
- `cis:` `D` `DP` globale Bezugstonhöhe
  `cis:` `D` `(D)` `DP` lokale Bezugstonhöhe

Zwei Ebenen der Semantik

- **Grundton** (der Funktion/des Akkordes)
  vs. Menge der klingenden Tonklassen = **Akkordkomponenten**
- `cis:  D  DP` globale Bezugstonhöhe
  `cis:  D  (D)  DP` lokale Bezugstonhöhe
- Grundfunktionen `S` / `T` / `D`;
  abgeleitete Nebenfunktionen `p` / `g`;
  Großschreibung vs. Kleinschreibung signalisiert "Tongeschlecht" Dur vs. moll.

Zwei Ebenen der Semantik

- **Grundton** (der Funktion/des Akkordes)
  vs. Menge der klingenden Tonklassen = **Akkordkomponenten**
- `cis:` `D` `DP` globale Bezugstonhöhe
  `cis:` `D` `(D)` `DP` lokale Bezugstonhöhe
- Grundfunktionen `S` / `T` / `D`;
  abgeleitete Nebenfunktionen `p` / `g`;
  Großschreibung vs. Kleinschreibung signalisiert "Tongeschlecht" Dur vs. moll.
- `P` und `G` normalerweise mit Geschlechtswechsel;
  `c:TpP` = C-Dur → a-moll → C-Dur
  falls nicht: implizite Schreibweise für explizite Verdurung / -mollung; (Maler 1931)
  Auswertung durch Explizit-Machen des Geschlechtswechsels:
  `c:TPP` = `c:Tp⇑p⇑` = C-Dur → a-moll → A-Dur → fis-moll → Fis-Dur
- Notationstechnisch kompakt / historisch durchgesetzt / für Geübte sehr anschaulich

Semantik für Akkord-Grundtöne

Sehr einfach, gilt aber nur *nach o.e. Normalisierung*.

$$[\![\,\mathtt{D}\,]\!] = [\![\,\mathtt{d}\,]\!] = (1,0) \qquad [\![\,\mathtt{S}\,]\!] = [\![\,\mathtt{s}\,]\!] = (-1,0)$$
$$[\![\,\mathtt{g}\,]\!] = (0,1) \qquad [\![\,\mathtt{G}\,]\!] = (0,-1) \qquad [\![\,\mathtt{p}\,]\!] = (-1,1) \qquad [\![\,\mathtt{P}\,]\!] = (1,-1)$$
$$[\![\,\mathtt{T}\,]\!] = [\![\,\mathtt{t}\,]\!] = [\![\,\Uparrow\,]\!] = [\![\,\Downarrow\,]\!] = (0,0)$$

Euler-Werte aller Buchstaben im funktionalharmonischen Symbol **und in allen evtl. geschachtelten lokalen Kontexten** einfach aufaddieren!

Semantik für Akkordkomponenten

- **Default-Töne** müssen nicht bezeichnet werden.
  (meist: Dreiklangstöne Grundton, Terz und Quinte)
  (pragmatische Überlegungen: Lesbarkeit / Papier-Platz, aber auch philosophische Gründe: Unverzichtbare Grundbestandteile (Hauptmann 1873) (Lewin 1982))

Semantik für Akkordkomponenten

- **Default-Töne** müssen nicht bezeichnet werden.
  (meist: Dreiklangstöne Grundton, Terz und Quinte)
  (pragmatische Überlegungen: Lesbarkeit / Papier-Platz, aber auch philosophische Gründe: Unverzichtbare Grundbestandteile (Hauptmann 1873) (Lewin 1982))
- **charakteristische Dissonanz** = Hinzufügung (`s56+` / `D7` / `D79-`)

Semantik für Akkordkomponenten

- **Default-Töne** müssen nicht bezeichnet werden.
  (meist: Dreiklangstöne Grundton, Terz und Quinte)
  (pragmatische Überlegungen: Lesbarkeit / Papier-Platz, aber auch philosophische Gründe: Unverzichtbare Grundbestandteile (Hauptmann 1873) (Lewin 1982))
- **charakteristische Dissonanz** = Hinzufügung (`s56+` / `D7` / `D79-`)
- **akkordfremder Ton** (Vorhalt/Durchgang/Wechselnote) = Ersetzung (/Hinzufügung)

Semantik für Akkordkomponenten

- **Default-Töne** müssen nicht bezeichnet werden.
  (meist: Dreiklangstöne Grundton, Terz und Quinte)
  (pragmatische Überlegungen: Lesbarkeit / Papier-Platz, aber auch philosophische Gründe: Unverzichtbare Grundbestandteile (Hauptmann 1873) (Lewin 1982))
- **charakteristische Dissonanz** = Hinzufügung (`s56+` / `D7` / `D79-`)
- **akkordfremder Ton** (Vorhalt/Durchgang/Wechselnote) = Ersetzung (/Hinzufügung)
- Typische historisch bedingte Anti-Orthogonalität:
  Unterdrückung der Default-Töne durch akkordfremde.
  `D7 = D1357` `D6 = D136` `S6 = S136` `S56 = S1356`

Semantik für Akkordkomponenten

- **Default-Töne** müssen nicht bezeichnet werden.
  (meist: Dreiklangstöne Grundton, Terz und Quinte)
  (pragmatische Überlegungen: Lesbarkeit / Papier-Platz, aber auch philosophische Gründe: Unverzichtbare Grundbestandteile (Hauptmann 1873) (Lewin 1982))
- **charakteristische Dissonanz** = Hinzufügung (`s56+` / `D7` / `D79-`)
- **akkordfremder Ton** (Vorhalt/Durchgang/Wechselnote) = Ersetzung (/Hinzufügung)
- Typische historisch bedingte Anti-Orthogonalität:
  Unterdrückung der Default-Töne durch akkordfremde.
  `D7` = `D1357` `D6` = `D136` `S6` = `S136` `S56` = `S1356`
- Aber wiederum:
  Notationstechnisch kompakt / historisch durchgesetzt / für Geübte sehr anschaulich

Semantik für Akkordkomponenten

- **Default-Töne** müssen nicht bezeichnet werden.
  (meist: Dreiklangstöne Grundton, Terz und Quinte)
  (pragmatische Überlegungen: Lesbarkeit / Papier-Platz, aber auch philosophische Gründe: Unverzichtbare Grundbestandteile (Hauptmann 1873) (Lewin 1982))
- **charakteristische Dissonanz** = Hinzufügung (`s56+` / `D7` / `D79-`)
- **akkordfremder Ton** (Vorhalt/Durchgang/Wechselnote) = Ersetzung (/Hinzufügung)
- Typische historisch bedingte Anti-Orthogonalität:
  Unterdrückung der Default-Töne durch akkordfremde.
  `D7` = `D1357` `D6` = `D136` `S6` = `S136` `S56` = `S1356`
- Aber wiederum:
  Notationstechnisch kompakt / historisch durchgesetzt / für Geübte sehr anschaulich
- in funCode abweichend von der Tradition:
  Alle Intervallformen müssen **explizit** gegeben werden: `6-` / `6+` / `6++` etc.
  (bis auf `...3...` und `...D7...`)

Semantik für Akkordkomponenten

- **Default-Töne** müssen nicht bezeichnet werden.
  (meist: Dreiklangstöne Grundton, Terz und Quinte)
  (pragmatische Überlegungen: Lesbarkeit / Papier-Platz, aber auch philosophische Gründe: Unverzichtbare Grundbestandteile (Hauptmann 1873) (Lewin 1982))
- **charakteristische Dissonanz** = Hinzufügung (`s56+` / `D7` / `D79-`)
- **akkordfremder Ton** (Vorhalt/Durchgang/Wechselnote) = Ersetzung (/Hinzufügung)
- Typische historisch bedingte Anti-Orthogonalität:
  Unterdrückung der Default-Töne durch akkordfremde.
  `D7` = `D1357` `D6` = `D136` `S6` = `S136` `S56` = `S1356`
- Aber wiederum:
  Notationstechnisch kompakt / historisch durchgesetzt / für Geübte sehr anschaulich
- in funCode abweichend von der Tradition:
  Alle Intervallformen müssen **explizit** gegeben werden: `6-` / `6+` / `6++` etc.
  (bis auf `...3...` und `...D7...`)
- **Alle diese Mengen und Regeln sind Parameter**

Weitere Parametrisierungen

- Lokalisierung der Tonklassen-Namen
  (für den tonalen Kontext am Anfang des Tracks)
- Lokalisierung von Funktionsbuchstaben
  (Kombinationen möglich, wie counter-parallel cp für g)
- **Makros** für häufig auftretenden Kombinationen,
  z. B. `Gr` = Englischsprachige Literatur "German Sixth"
  = ein Akkord der übermäßigen Sexte, z. B. *g*+h+des+f+as
  definierbar durch `Gr := D/5-_79-`
  — politisch wichtig wegen fast immer unterspezifierter Verwendung!
- Erlaubnis höherer Intervallzahlen (Jazz-Harmonielehre: 11, 13 etc.)
- Intervall-Modifier voll definierbar

Komplexierung durch “Feature Interactions”

- Vererben von Akkordkomponenten (als Quelltext, wg. _ und ^).
  `D6-79-  D5..`
- Auch bei Geschlechtswechsel:
  `SP57-  Sp.6+`

Komplexierung durch “Feature Interactions”

- Vererben von Akkordkomponenten (als Quelltext, wg. _ und ^).
  `D6-79-  D5..`
- Auch bei Geschlechtswechsel:
  `SP57-  Sp.6+`
- Multi-Funktionen und lokale Kontexte:
  `(D&T) S` ist unproblematisch und üblich. (Distributiert)
  `(S) D&T` ist z.Zt. undefiniert – kulturell vorhanden, aber idiosynkratisch.
  `(S&S) D&T` ist z.Zt. undefiniert – kulturell nicht vorhanden, aber zweckmäßig.

Komplexierung durch “Feature Interactions”

- Vererben von Akkordkomponenten (als Quelltext, wg. _ und ^).
  `D6-79-  D5..`
- Auch bei Geschlechtswechsel:
  `SP57-  Sp.6+`
- Multi-Funktionen und lokale Kontexte:
  `(D&T) S` ist unproblematisch und üblich. (Distributiert)
  `(S) D&T` ist z.Zt. undefiniert – kulturell vorhanden, aber idiosynkratisch.
  `(S&S) D&T` ist z.Zt. undefiniert – kulturell nicht vorhanden, aber zweckmäßig.
- **Future Work**: manches noch unerforscht.

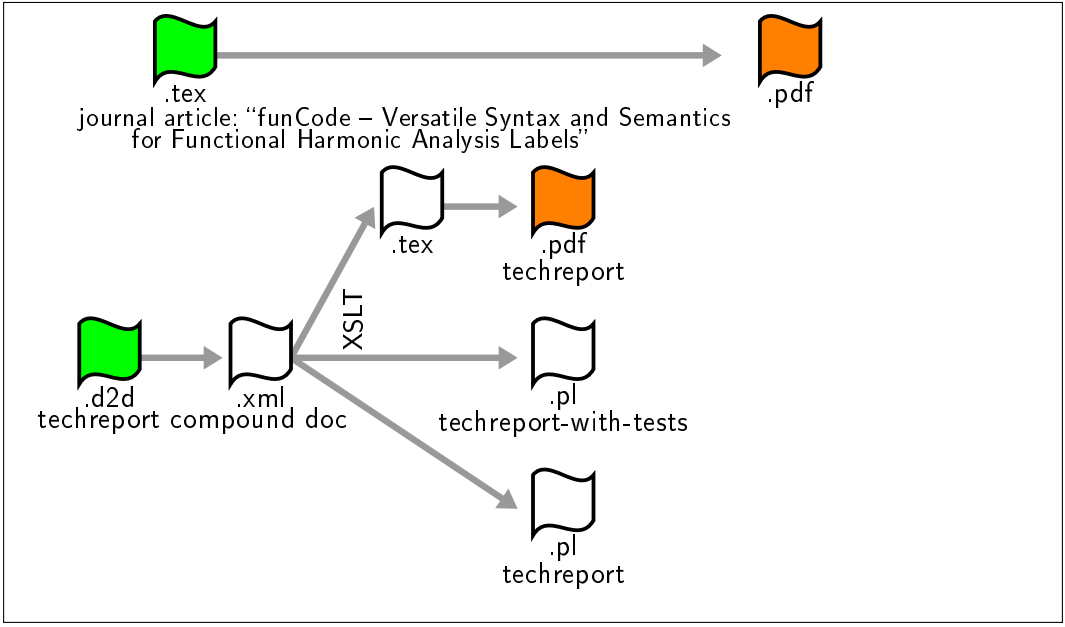
Gliederung

Gründe für Prolog

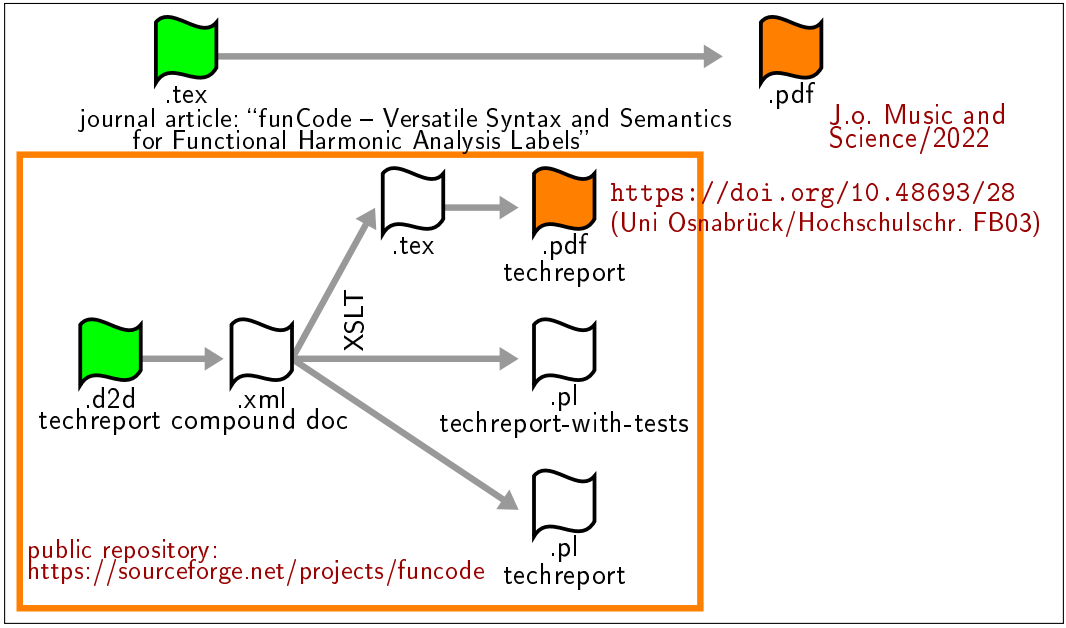
Politisch wichtig zur Akzeptanz:

- anerkannter Formalismus mit pragmatischer Komponente und dennoch exakt definierter Semantik.
  (Konkurrenzentwurf über Stufentheorie / "Roman Numerals" ist ein öffentliches Python-Programm (Nápoles López und Fujinaga 2020)).
- erster Aufschlag war in "Z":
  Viel kompakter und streng typisiert, aber für die Zielgruppe schlecht geeignet.
- erster Versuch der Publikation hatte *integrierte* Spezifikationsformeln.
  Nach Fehlschlag:
  Trennen in Journal Article und Technical Report.

Dokumente und Veröffentlichungsplan:



Dokumente und Veröffentlichungsplan:



Gliederung

Übersetzungs- und Ausführungsmodell

0. Quelltext

```
cis: t (D7) s
```

Übersetzungs- und Ausführungsmodell

0. Quelltext

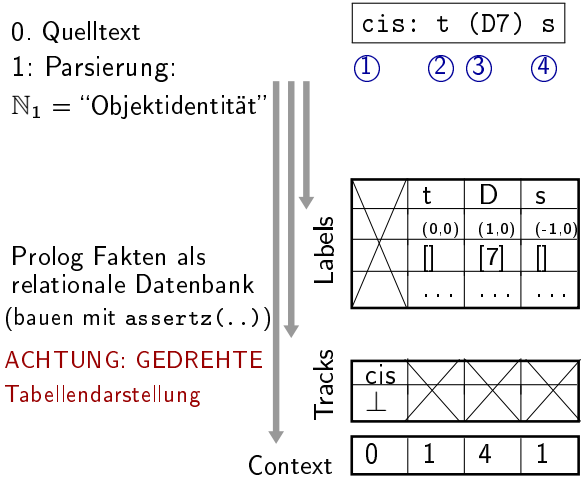
```
cis: t (D7) s
```

1: Parsierung:

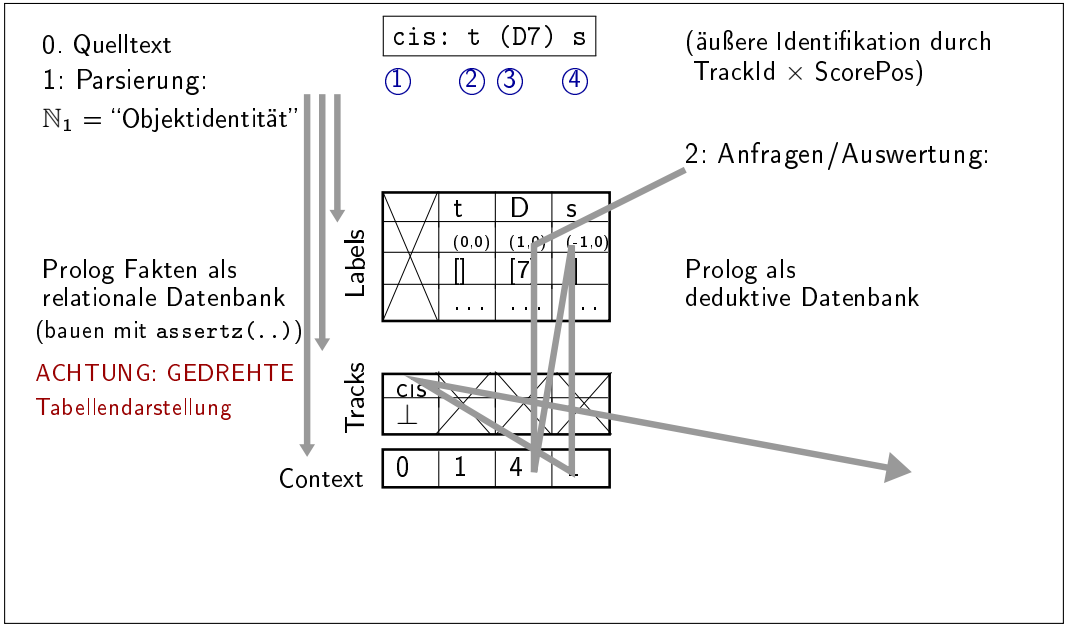
① ② ③ ④

$\mathbb{N}_1$ = "Objektidentität"

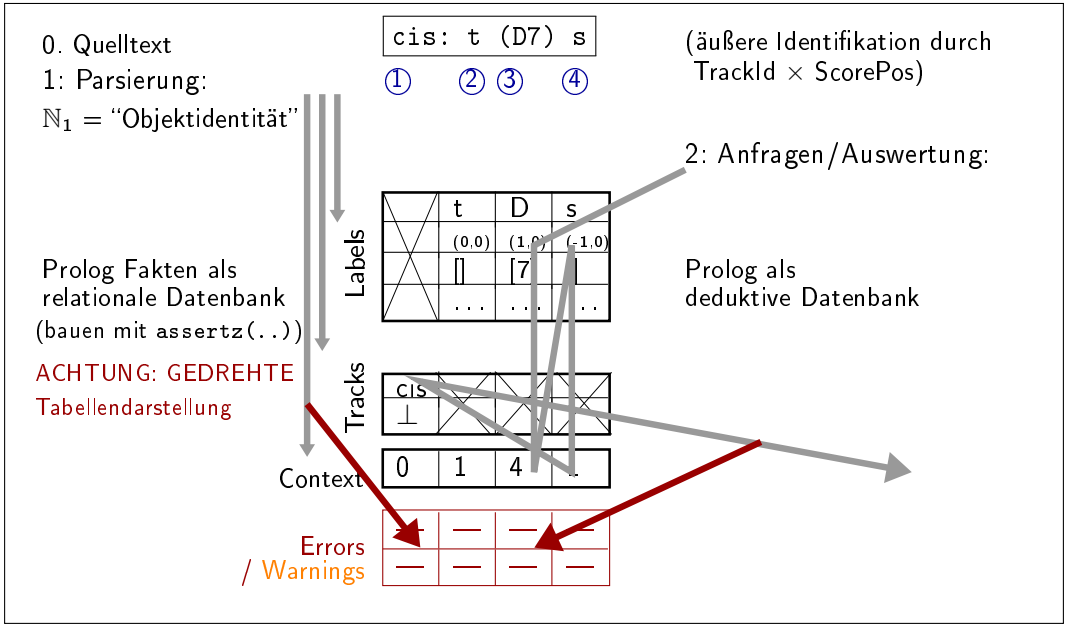
Übersetzungs- und Ausführungsmodell



Übersetzungs- und Ausführungsmodell



Übersetzungs- und Ausführungsmodell



Gliederung

Zusammenfassung der Erfahrungen

- Bequeme Entwicklungsarbeit wegen sofortiger Verifikation durch (begrenzte) Tests. (Insgesamt ca. sechs Wochen für alles, incl. Journal Artikel)
- Fundamentale Bedeutung der Test-Codes für **Regressionstests** / Grundlegendes Refactoring wegen des experimentellen Charakters ca. viermal nötig.
- Frühzeitige und ubiquitäre Behandlung von Errors und Warnings.
- Wichtigkeit der Differenzierung zwischen Stamm-Code und Parametrisierungs-Code ("grüne Kästen"), wenngleich auch nur informell.
- Lokalisierungen (Makros etc.) wurden teils durch Eingriff in die Parsierungs-Kette der *Definite Clause Grammar (DCG)* Implementierung realisiert. Dies verlangte *reverse engineering* und schien zunächst **ziemlich hacky**.

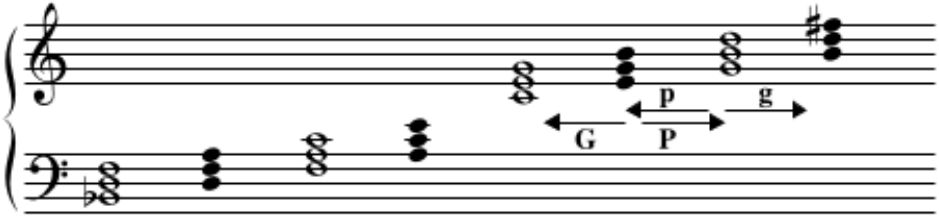
  Stellte sich dann aber als *gut eingrenzbar* heraus: Nur eine(1) grundlegende Transformations-Methode musste jeweils definiert werden. Allein diese müssen gglfls. bei neuer Prolog-Version angepasst werden.
- Single-Source Prinzip (nur *eine(1)* Datei) von Autor ML als sehr hilfreich empfunden.
- Allerdings fehlt die Editor-Unterstützung (emacs): häufiges manuelles Umschalten zwischen LaTeX- und Prolog-Mode erforderlich.
- Reaktionen von Lesern aus den beiden Fach-Communities: fehlen leider noch.

Gliederung

Grundlagen

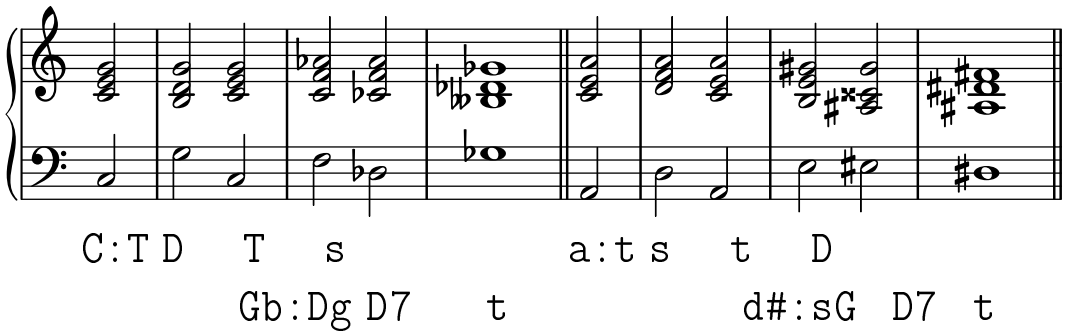
- Modulation = Akkordfolge, die eine *Verschiebung des Gefühls für das diatonische Zentrum* bewirkt.
- Es werden genannt: “diatonische”, “chromatische”, “enharmonische”. (in den deutsch-sprachigen Lehrbüchern, z.B. (Geller (2002), Hussong (2005), Acker (2009))
- Diatonische Modulation: Ein Dreiklang hat zwei verschiedene Interpretationen in zwei verschiedenen Kontexten. (Ist aber in sich immer derselbe Dreiklang mit derselben Tonklasse als Grundton!)

Diatonische Funktionen (ungefähr gleich “Stufen”)

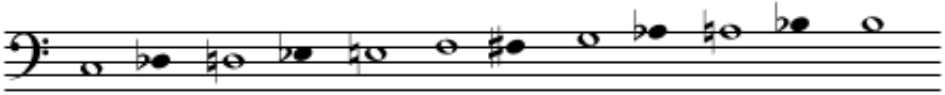



```
          C:Sp   S      Sg=Tp  T      Tg=Dp  D      {Dg}
          C:ii   IV     vi     I      iii    V      {vii}
a:{sG}    s      sP=tG  t      tP{=dG d      dP}
a:{bII}   iv     VI     i      III    v      VII
```

Weitest mögliche diatonische Modulationen (Beispiele)



Erreichbarkeit aller chromatischen Stufen (Beispiele)



C:T	**sG**	**DD**	**tP**	**Tg**	**S**	**DDg**	**D**	**sP**	**Tp**	**SS**	**Dg**
a:tP	**{Tg}**	**s**	**ssG**	**D**	**tG**	**Sg**	**dP**	**{Dg}**	**t**	**sG**	**DD**

Terms in braces {...} describe the enharmonically exchanged pitch class.

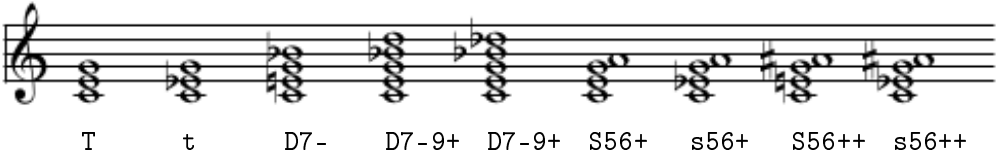
Gliederung

System i1: Charakteristische Dissonanzen

T | t | D 7- (9-|9+|) | S 5(6+|6++) | s 5(6+|6++)

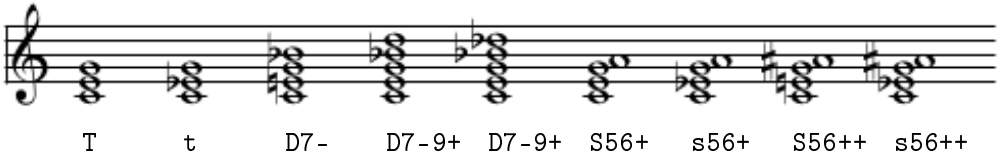
System i1: Charakteristische Dissonanzen

T | t | D 7- (9-|9+|) | S 5(6+|6++) | s 5(6+|6++)



System i1: Charakteristische Dissonanzen

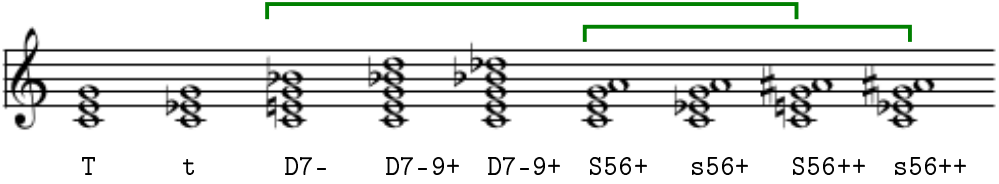
T | t | D 7- (9-|9+|) | S 5(6+|6++) | s 5(6+|6++)

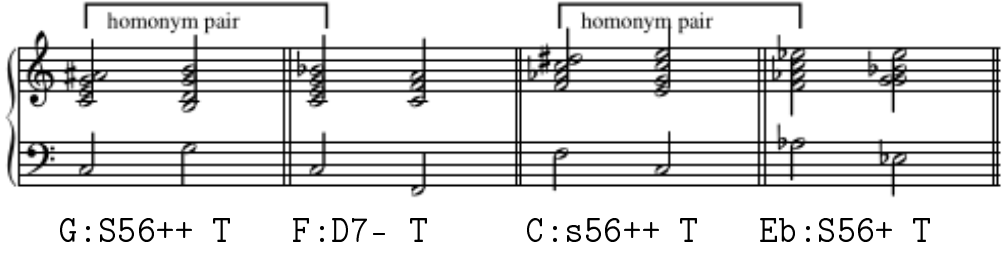


- Ab jetzt reden wir nur noch von Akkord*formen*. Also *ohne sequentiellen Kontext*.
- Bestimmte "Stufenklänge" / "Jazz-Klänge" sind *keine* funktionalen!
  (z.B. `T7+`, `t7-`, "Csus" = `T4`, etc.)
- Der berühmteste ist `s56+`, Umkehrung des `D7-` (auch "halbvermindert").
- Schon in diesem aller-einfachsten Label-System sind *zwei Homonym-Paare*!

System i1: Charakteristische Dissonanzen

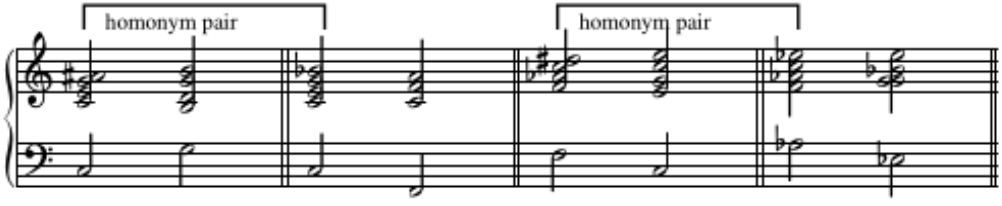
T | t | D 7- (9-|9+|) | S 5(6+|6++) | s 5(6+|6++)





Kategorien von Homonymen

pc set	=				
func. expr. root pitch class root symbol (modulo case)	= = =	≠ = ≠	≠ = =	≠ ≠ ≠	= (≠) (=)
	A-1	A-2	A-3	A-4	A-5
i1 (char.Diss.)	–	D7- ≅ S56++	S56+ ≅ s56++	–	–
	–	–			



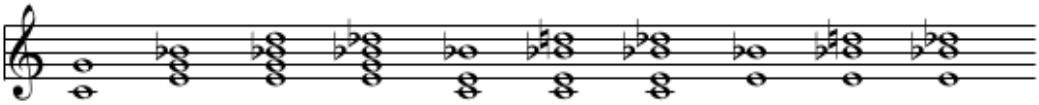
Gliederung

System i2: Verkürzungen

```
T (3/|) | t | D (1/|) (5/|) 7- (9-|9+|) | s 5(6+|6++) | S 5(6+|6++)
```

System i2: Verkürzungen

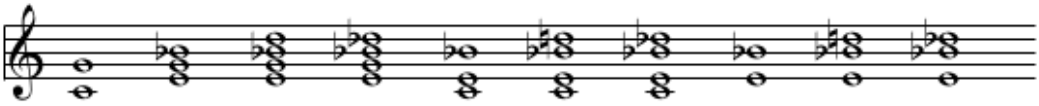
T (3/|) | t | D (1/|) (5/|) 7- (9-|9+|) | s 5(6+|6++) | S 5(6+|6++)



T3/ D/7- D/7-9+ D/7-9- D5/7- D5/7-9+ D5/7-9- D/5/7- D/5/7-9+ D/5/7-9-

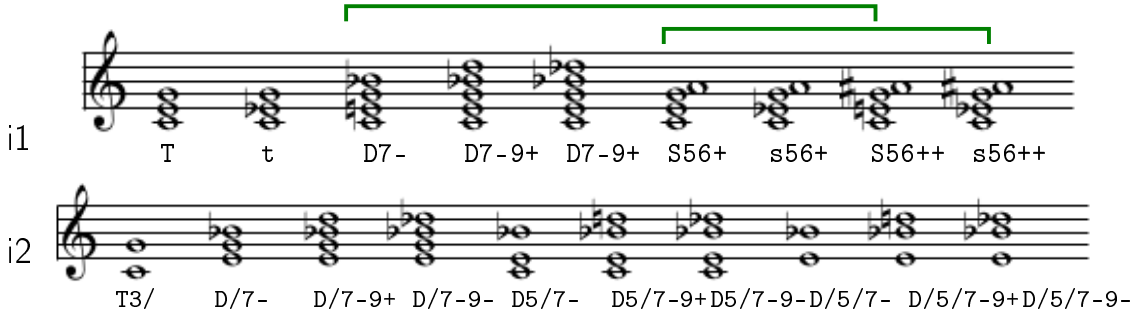
System i2: Verkürzungen

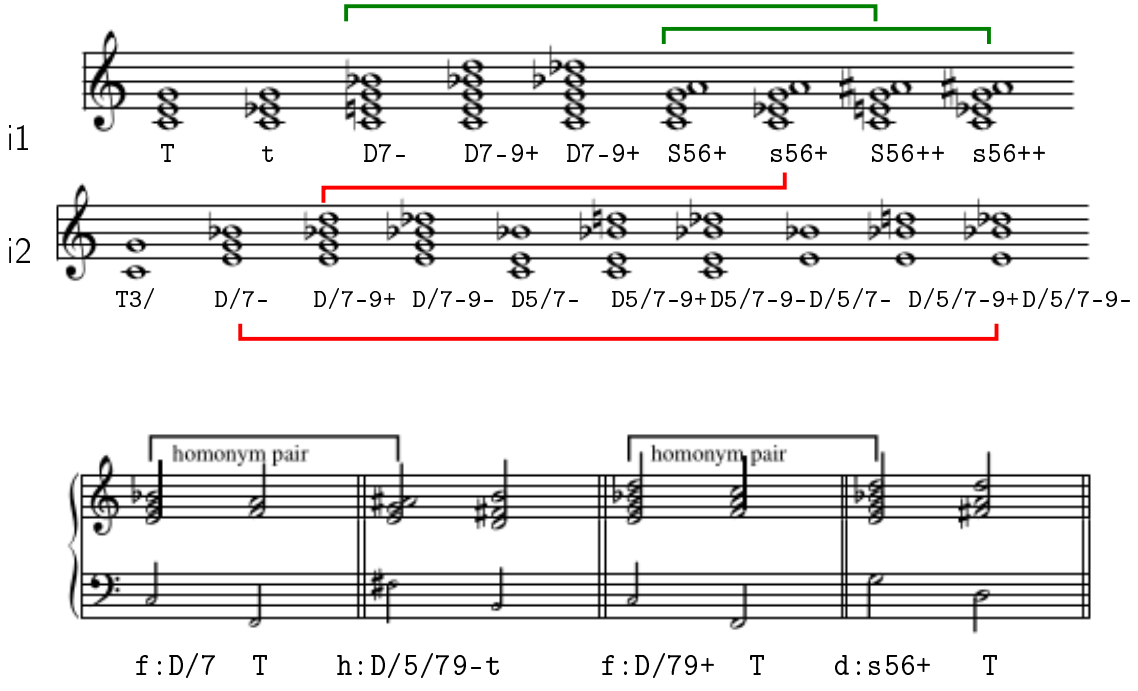
```
T (3/|) | t | D (1/|) (5/|) 7- (9-|9+|) | s 5(6+|6++) | S 5(6+|6++)
```

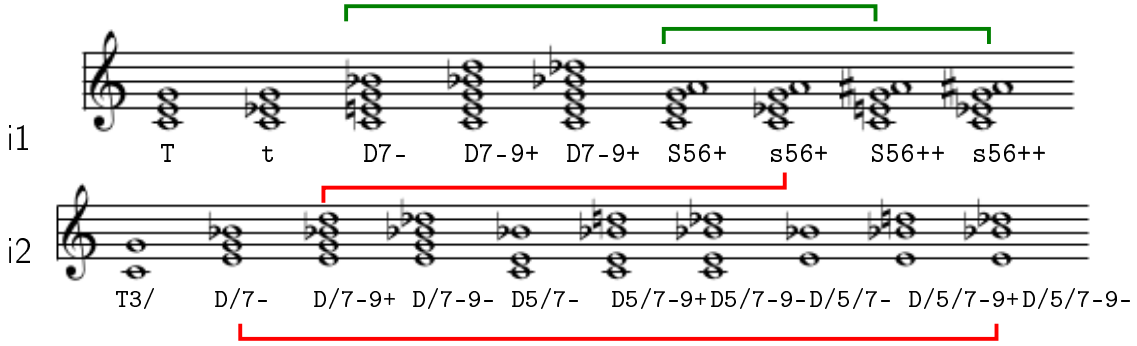


T3/ D/7- D/7-9+ D/7-9- D5/7- D5/7-9+ D5/7-9-D/5/7- D/5/7-9+D/5/7-9-

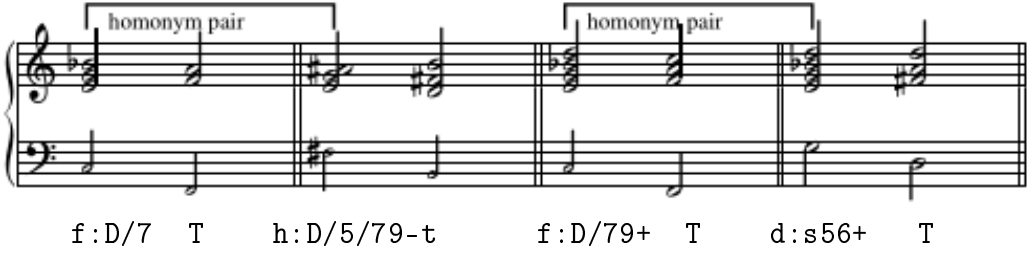
- Benennung der Akkord-Komponenten nach ihrer *ursprünglichen* Funktion: Terz, Quinte, Septime
- Prinzip schon bei Rameau (1722)
- Aber historisch nicht immer korrekt:`D/7-9+` erscheint zeitlich eher als `D7-9+`
- Entwurfsentscheidungen S-m versus S-p: `t3/`, `t5/`, `T/` ?
- Neuartige Homonyme wegen der Grundtonverkürzung


i1
T t D7- D7-9+ D7-9+ S56+ s56+ S56++ s56++
i2
T3/ D/7- D/7-9+ D/7-9- D5/7- D5/7-9+ D5/7-9-D/5/7- D/5/7-9+D/5/7-9-


i1
T t D7- D7-9+ D7-9+ S56+ s56+ S56++ s56++
i2
T3/ D/7- D/7-9+ D/7-9- D5/7- D5/7-9+ D5/7-9-D/5/7- D/5/7-9+D/5/7-9-
homonym pair
homonym pair
f:D/7 T h:D/5/79-t f:D/79+ T d:s56+ T

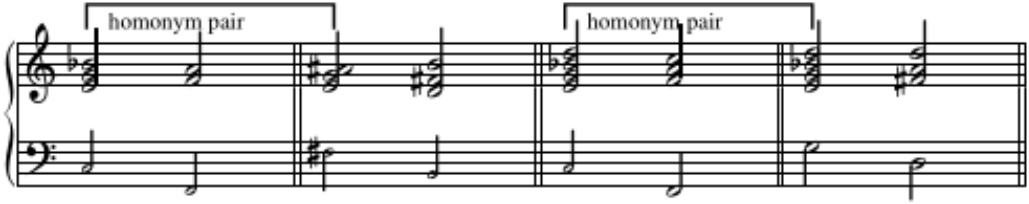


SIND DAS ALLE?



Kategorien von Homonymen

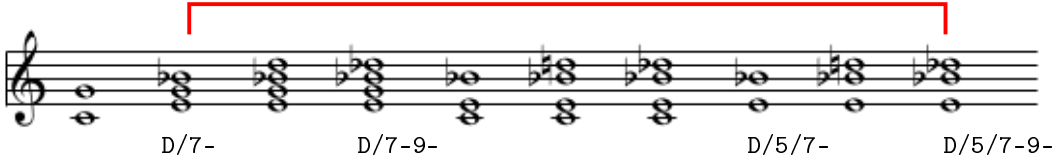
pc set	=				
func. expr. root pitch class root symbol (modulo case)	= = =	= = ≠	≠ = =	≠ ≠ ≠	= (≠) (=)
	A-1	A-2	A-3	A-4	A-5
i1 (char.Diss.)	–	D7- ≅ S56++	S56+ ≅ s56++	–	–
i2 (Verkürzung)	–	–	D/7-≅D/5/79-	D/79+≅s56+	



f:D/7 T h:D/5/79-t f:D/79+ T d:s56+ T

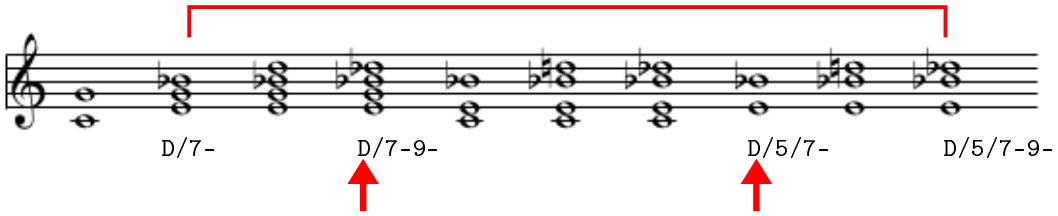
Kategorien von Homonymen

pc set	=				
func. expr. root pitch class root symbol (modulo case)	= = =	= = ≠	≠ = =	≠ ≠ ≠	= (≠) (=)
	A-1	A-2	A-3	A-4	A-5
i1 (char.Diss.)	–	D7- ≅ S56++	S56+ ≅ s56++	–	–
i2 (Verkürzung)	–	–	D/7-≅D/5/79-	D/79+≅s56+	



Kategorien von Homonymen

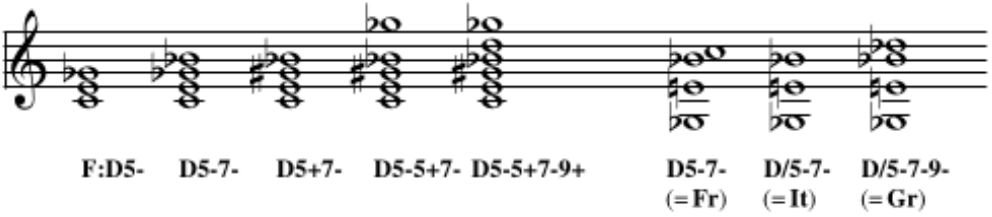
pc set	=				
func. expr. root pitch class root symbol (modulo case)	= = =	= = ≠	≠ = =	≠ ≠ ≠	= (≠) (=)
	A-1	A-2	A-3	A-4	A-5
i1 (char.Diss.)	–	D7- ≅ S56++	S56+ ≅ s56++	–	–
i2 (Verkürzung)	–	–	D/7-≅D/5/79-	D/79+≅s56+	D/79- D/5/7-



Gliederung

System i3: (Quint-)Alterationen

```
T(3/|)|t| D(1/|)(|5/|5-|5+|5-5+) 7-(9-|9+|) | s5(6+|6++)| S5(6+|6++)
```



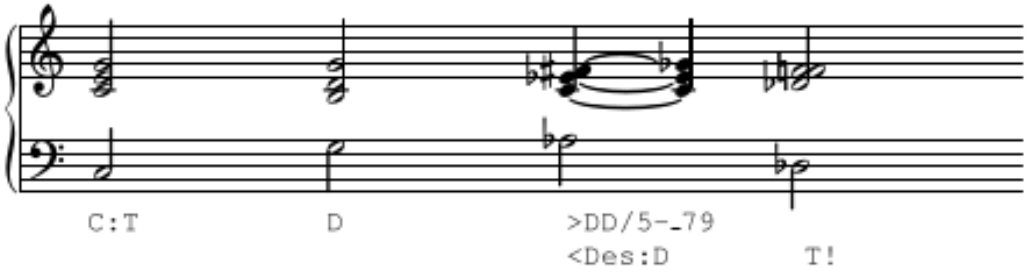
- insgesamt 37 Akkordformen, langsam wird's unübersichtlich!?
- manuelles Mining gerade noch möglich:

Patterns from 3+ up to 7- →	5/	5	5-	5+	5+5-
					2
		3	4	2	2
Patterns from 7- up to 3+ ↓	6	3	2	4	2
					2
		3	4	2	2
	6	3	2	4	2
D/7 6	**6**	6	6	6	6
					2
		3	4	2	2
	6	**3**	2	4	2
3	3	**3**	3	3	3
D/79- 3	3	**3**	3	3	3
					2
		3	**4**	2	*2*
	6	3	**2**	4	*2*
4	4	4	**4**	4	*4*
D7 2	2	2	**2**	2	*2*
					2
		3	4	**2**	*2*
	6	3	2	**4**	*2*
2	2	2	2	**2**	*2*
D/79+ 4	4	4	4	**4**	*4*
					2
		3	*4*	*2*	**2**
	6	3	*2*	*4*	**2**
2	2	2	*2*	*2*	**2**
2	2	2	*2*	*2*	**2**
D79+ 2	2	2	*2*	*2*	**2**

Kategorien von Homonymen

pc set	=				
func. expr. root pitch class root symbol (modulo case)	= =	= ≠	≠ =	≠ ≠	= (≠) (=)
	A-1	A-2	A-3	A-4	A-5
i1 (char.Diss.)	–	D7- ≅ S56++	S56+ ≅ s56++	–	–
i2 (Verkürzung)	–	–	D/7-≅D/5/79-	D/79+≅s56+	D/79- D/5/7-
i3 (Alteration)	–	–	D7 ≅D/5-79- D5-79+ ≅D5+79+ ≅D5-5+7 ≅D/5-5+79+ (+ 8 weitere Paare)	D/5-79- ≅S56++	D5-5+79+ D5-7 (="Fr") ≅D/5+79+

Konstruiertes Beispiel: Modulation durch Quintalteration



Gliederung

System i4: Akkordfremde Töne

- historisch entstanden durch “eingefrorene” Stimmführung
- Durchgänge, Nebennoten, Vorhalte.
  (Suspensionen = Vorhalte von unten, lt. Hentschel u. a. 2020)
- manuelles Mining nicht mehr möglich.

	pc	T	t	S	s	D
7^	11	replaces 1				
2-	1			replaces 1		
2+	2			replaces 1		
2^	2	replaces 3	replaces 3	replaces 3	replaces 3	replaces 3
2^^	3	replaces 3		replaces 3		replaces 3
4	5	replaces 3	replaces 3	replaces 3	replaces 3	replaces 3
4+	6	replaces 3		replaces 3		replaces 3
4^	6	replaces 5				replaces 5
5-	6					**own** for 5
5+	8					**own** for 5
6-	8	replaces 5				replaces 5
6+	9	replaces 5		**own**		replaces 5
6++	10			**own**		
6^	9					replaces 7-
7-	10			replaces 6-/+		**own**
7+	11			replaces 6+/++		
9-	1					**own**
9+	2					**own**
10-	3					replaces 9+/-

Regulärer Ausdruck für akkordfremde Töne

```
  T ( (7^ (1|) (2-|2+|)) | ((1|) (2-|2+)) | 1/ |)
    ( ((2^|2^^) (3|) (4|4+|)) | ((3|) (4|4+)) | 3/ |)
    ( ((4^) (5|) (6-|6+|)) | ((5|) (6+|6-)) | 5/ |)
| t ( (7^ (1|) (2-|2+|)) | ((1|) (2-|2+)) | 1/ )
    ( ((2^) 3 (4|) ) | (3 4) |)
    ( ((4^) (5|) (6-|6+|)) | ((5|) (6+|6-)) | 5/ |)
| S ( (7^ (1|) (2-|2+|)) | ((1|) (2-|2+)) | 1/ |)
    ( ((2^|2^^) (3|) (4|4+|)) | ((3|) (4|4+)) | 3/ |)
    (4^|5|4^5 |)
    (6+|6++|7-|7+)
| s ( (7^ (1|) (2-|2+|)) | ((1|) (2-|2+)) | 1/ |)
    ( ((2^) 3 (4|) ) | (3 4) |)
    (4^|5|4^5)
    (6+|6++|7-|7+)
| D ( (7^ (1|) (2-|2+|)) | ((1|) (2-|2+)) | 1/ |)
    ( ((2^|2^^) (3|) (4|4+|)) | ((3|) (4|4+)) |)
    ( (4^ (5|) (6-|6+|)) | ((5|)(6-|6+)) | 5/|5-|5+|5-5+| |)
    (6^|7-) (9-|9+|10-|)
```

Kategorien von Homonymen

pc set	=				
func. expr. root pitch class root symbol (modulo case)	≠ = =	≠	= ≠	≠	= (≠) (=)
	A-1	A-2	A-3	A-4	A-5
i1 (char.Diss.)	–	D7- ≅ S56++	S56+ ≅ s56++	–	–
i2 (Verkürzung)	–	–	D/7-≅D/5/79-	D/79+≅s56+	D/79- D/5/7-
i3 (Alteration)	–	–	9 Paare 1 Quadrupel	D/5-79- ≅S56++	D5-5+79+ D5-7 ≅D/5+79+
i4 (akkordfremde)	s56++≅s57- 218,352	74.652	1.005.224	604.321	704

Anfragen an die Software

- *addFunctions*(*Id*, *R*)

expands the given regular expression *R* and binds all resulting functional labels to the atomic identifier *Id*.

- *allClassNumbers*(*Id*, *R*)

returns in *R* the list of all generated pc set class numbers.

- *numberOfTerms*(*Id*, *R*)

returns in *R* the number of all functional expressions bound to *Id*.

- *writeText*(*Id*)

displays all functional expressions bound to *Id*, sorted by pc set classes.

- *writeHomonyms*(*Id*)

displays all pairs of interpretation for the same pc set class, ordered by their class numbers.

- *writeHomonymsSurvey*(*Id*)

displays the number of homonym pairs by the tonic displacements and categories.

- *writeSymmetrics*(*Id*)

displays the number of rotation symmetric pc sets bound to *Id*, sorted by the shift interval.

- *writeSetNormalized*(*L*)

takes a list *L* of atoms representing pc classes and prints a normalised representation.

- *writeFunction*(*L*)

parses a function source text and displays the resulting pc set.

```
?- writeText(i2).
-- 05 =021h=33 <57> -------
T3/ (51000)
-- 06 =041h=65 <66> Symm=6 -------
D1/5/7- (22000)
-- 026 =045h=69 <246> -------
D5/7- (21000)
-- 036 =049h=73 <336> -------
D1/5/7-9- (22000)
D1/7- (81000)
-- 0236 =04Dh=77 <2136> -------
D5/7-9- (21000)
-- 046 =051h=81 <426> -------
D1/5/7-9+ (22000)
-- 0246 =055h=85 <2226> -------
D5/7-9+ (21000)
-- 037 =089h=137 <345> -------
t (00000)
-- 047 =091h=145 <435> -------
T (00000)
-- 0258 =125h=293 <2334> -------
s56+ (50000)
D1/7-9+ (T1000)
-- 0358 =129h=297 <3234> -------
s56++ (50000)
S56+ (80000)
-- 0368 =149h=329 <3324> -------
S56++ (80000)
D7- (80000)
-- 0369 =249h=585 <3333> Symm=3
-------
D1/7-9- (21000)
-- 02369 =24Dh=589 <21333> -------
D7-9- (20000)
-- 02469 =255h=597 <22233> -------
D7-9+ (20000)
true
```

```
?- writeHomonyms(i3).
==== 06=041h=65=<66> Symm=6 =============
D1/5/7- -(6)-> D1/5/7- (2000) (2000)
==== 026=045h=69=<246> ================
D1/5-7- -(6)-> D5/7- (2100) (1000)
==== 036=049h=73=<336> ================
D1/5/7-9- -(6)-> D1/7- (2000) (1000)
==== 046=051h=81=<426> ================
D1/5/7-9+ -(6)-> D1/5+7- (2000) (2100)
==== 0246=055h=85=<2226> ==============
D1/5-5+7- -(6)-> D5/7-9+ (2200) (1000)
==== 0248=115h=277=<2244> ==============
D1/5-7-9+ -(6)-> D5+7- (2100) (1100)
==== 0258=125h=293=<2334> ==============
D1/5+7-9- -(6)-> D1/7-9+ (2100) (1000)
s56+ -(3)-> D1/7-9+ (0000) (1000)
D1/5+7-9- -(3)-> s56+ (2100) (0000)
==== 0358=129h=297=<3234> ==============
s56++ -(3)-> S56+ (0000) (0000)
==== 0268=145h=325=<2424> Symm=6 ========
D5-7- -(6)-> D5-7- (1100) (1100)
D1/5+7-9+ -(6)-> D1/5+7-9+ (2100) (2100)
D5-7- -(2)-> D1/5+7-9+ (1100) (2100)
D1/5+7-9+ -(4)-> D5-7- (2100) (1100)

==== 0368=149h=329=<3324> ==============
D1/5-7-9- -(6)-> D7- (2100) (0000)
D7- -(2)-> S56++ (0000) (0000)
S56++ -(4)-> D1/5-7-9- (0000) (2100)
==== 02468=155h=341=<22224> ============
D5+7-9+ -(4)-> D5-5+7- (1100) (1200)
D5-7-9+ -(6)-> D5-5+7- (1100) (1200)
D5-7-9+ -(2)-> D5+7-9+ (1100) (1100)
D5-5+7- -(2)-> D1/5-5+7-9+ (1200) (2200)
D1/5-5+7-9+ -(6)-> D5+7-9+ (2200) (1100)
D1/5-5+7-9+ -(4)-> D5-7-9+ (2200) (1100)
==== 0369=249h=585=<3333> Symm=3 ========
D1/7-9- -(6)-> D1/7-9- (1000) (1000)
D1/7-9- -(3)-> D1/7-9- (1000) (1000)
==== 02469=255h=597=<22233> ============
D1/5-5+7-9- -(6)-> D7-9+ (2200) (0000)
==== 02468t=555h=1365=<222222> Symm=2 ====
D5-5+7-9+ -(6)-> D5-5+7-9+ (1200) (1200)
D5-5+7-9+ -(4)-> D5-5+7-9+ (1200) (1200)
D5-5+7-9+ -(2)-> D5-5+7-9+ (1200) (1200)
true
```

Survey output i2 (Char.Diss + Verkürzung)

```
?- writeHomonymsSurvey(i2).
Survey of homonyms
Complexity limits min,maxXeach,sum = compl(0,0,0,0)/compl(11,11,11,11)
/compl(0,0,0,0)/compl(11,11,11,11):
cat |      0      1      2      3      4      5      6      sum
A-1 |      0      0      0      0      0      0      0        0
A-2 |      0      0      1      0      0      0      0        1
A-3 |      0      0      0      1      0      0      1        2
A-4 |      0      0      0      1      0      0      0        1
A-5 |      0      0      0      1      0      0      2        3
true
```

Survey output i3 (Char.Diss + Verkürzung + Quintalter.)

```
?- writeHomonymsSurvey(i3).
Survey of homonyms
Complexity limits min,maxXeach,sum = compl(0,0,0,0)/compl(11,11,11,11)
/compl(0,0,0,0)/compl(11,11,11,11):
cat |      0      1      2      3      4      5      6      sum
A-1 |      0      0      0      0      0      0      0        0
A-2 |      0      0      1      0      0      0      0        1
A-3 |      0      0      3      1      3      0     10       17
A-4 |      0      0      0      2      1      0      0        3
A-5 |      0      0      1      1      1      0      5        8
true
```

Survey output i4 (. . . + akkordfremde)

```
?- writeHomonymsSurvey(i4).
Survey of homonyms
Complexity limits min,maxXeach,sum = compl(0,0,0,0)/compl(11,11,11,11)
/compl(0,0,0,0)/compl(11,11,11,11):
cat |      0      1      2      3      4      5      6      sum
A-1 | 208738      0   1792    766   3730      0   3326   218352
A-2 |     30    427  45160     68    517  27794    656    74652
A-3 |      0 160657 154364 181955 241043 195659  71657  1005335
A-4 |  37372 114458 102806 101623 108923  77406  61733   604321
A-5 |      0      1     53     63    167      1    419      704
true
```

Komplexitätsmaße

c1 the number of own chord pitches which are affected (by a non-chord note, by alteration, or by cancellation).
c2 the number of additional non-chord pitch classes which sound in the chord.
c3 the number of own chord pitches which sound together with a related non-chord pitch.
c4 the number of pitches resulting from more than one chord component.
c5 the number of chord notes which are affected by *two* non-chord notes (mostly one from above and one from below).
c6 the number of chord notes from c5, which are themselves sounding (=the intersection of the contributors to c3 and c5).

Requests filtered by complexity

- *writeHomonyms*($Id, J, K, C_1, C_2, C_3, C_4$)

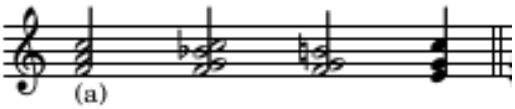
as above, but filters by tonic distance intervals (J is a list of integers from 0 to 6, giving the required half tone distances), categories (K is list of integers) and complexities (C_1 is a list of minimal complexities $\langle c_1, c_2, c_3, c_4 \rangle$; they filter each functional label individually; C_2 a list of maximal complexities; C_3 and C_4 give minima and maxima for the *sum* of both labels.

All filters can be completely opened by supplying the special atomic value `all`.

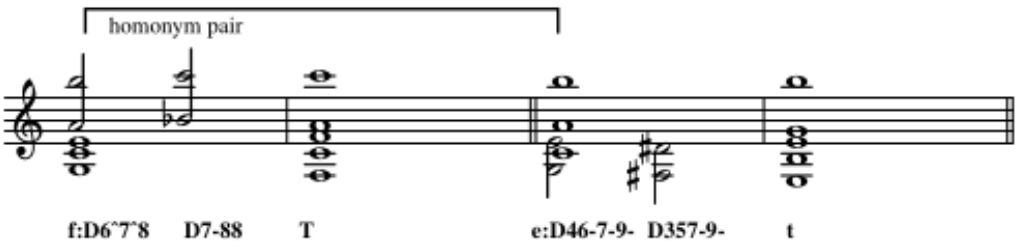
- *writeHomonymsSurvey*(Id, C_1, C_2, C_3, C_4)

displays the number of homonym pairs by the tonic displacements and categories, after filtering by complexity

Konstruierte Beispiele

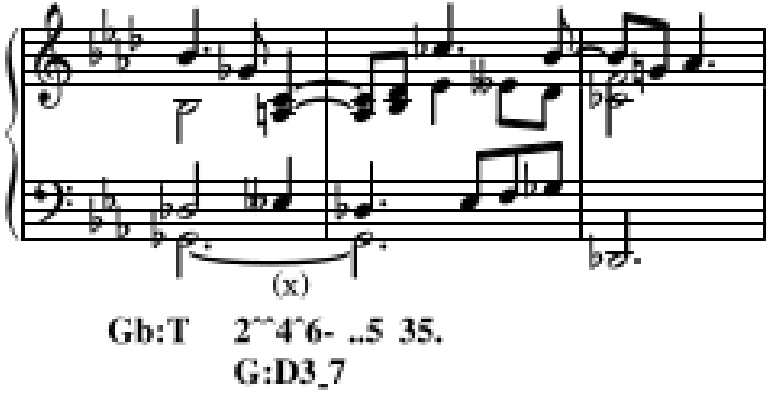


f:T D[4]7
c:D5/7[2++4] !..33 T3_



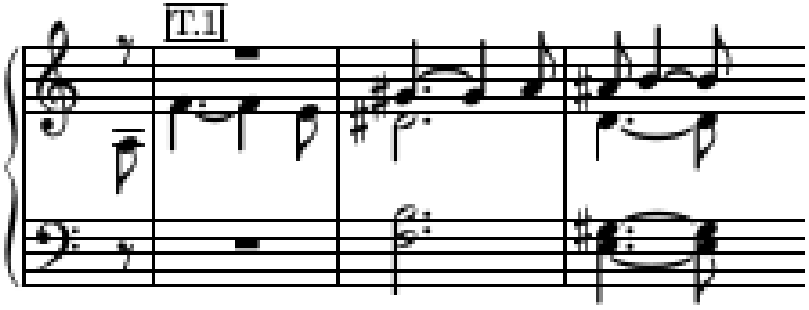
Tatsächliches Beispiel

(aus “Tristan”, zweiter Akt, woher sonst?)



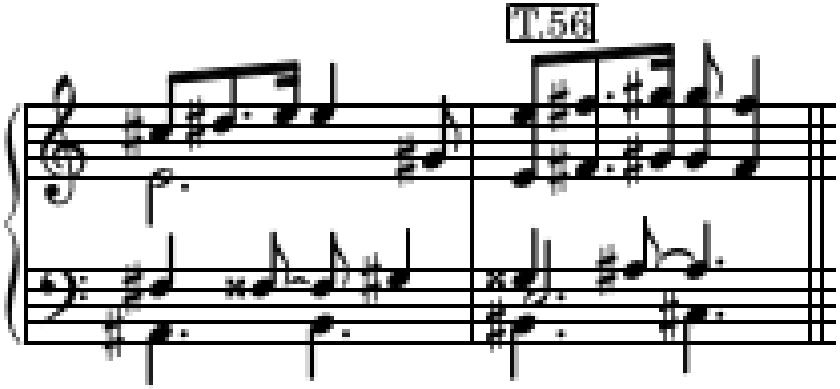
Complexity (x) = (3,3,0,0), bei nur 4 Akkordtönen!

Tatsächliches Beispiel



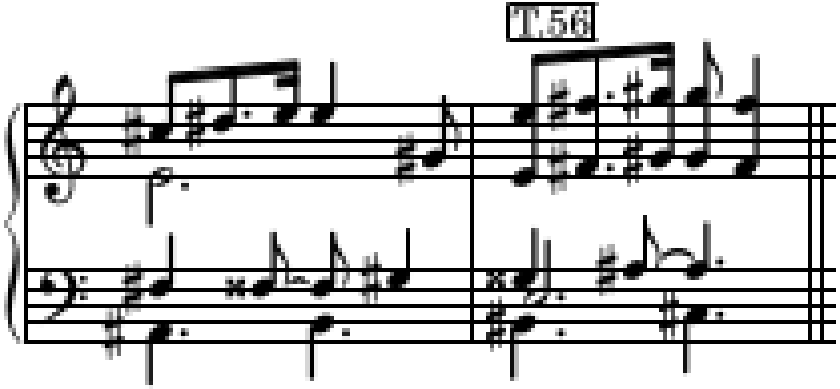

```
a:t6- .5 DD5-_6^ .7 D4^7- 5.
dis:s56+
```

Tatsächliches Beispiel




```
h:D3_7 ..5+  s5_2^^ .3  D/74^_2^^ ..3 .5_.9+
```

Tatsächliches Beispiel




```
h:D3_7 ..5+  s5_2^^ .3  D/74^_2^^ ..3 .5_.9+
```

a:t

Quellen

Acker, H. (2009). *Modulationslehre*. Kassel: Bärenreiter. ISBN: 978-3-7618-2126-6.

Geller, D. (2002). *Modulationslehre*. Wiesbaden: Breitkopf und Härtel. ISBN: 3-7651-0368-3.

Hauptmann, M. (1873). *Die Natur der Harmonik und Metrik*. Leipzig: Breitkopf und Härtel.

Hentschel, J. u. a. (2020). *Standard for harmonic annotations developed at the Digital and Cognitive Musicology Lab at École Polytechnique Fédérale de Lausanne*. Techn. Ber. [Accessed 02-Mar-2022]. URL: https://github.com/DCMLab/standards.

Hussong, H. (2005). *Untersuchungen zu praktischen Harmonielehren seit 1945*. Berlin: Verlag im Internet GmbH.

Lepper, M., B. Trancón y Widemann und M. Oehler (2022a). „funCode — Versatile Syntax and Semantics for Functional Harmonic Analysis Labels“. In: *Music and Science* 5. DOI: 10.1177/20592043221085659.

– (2022b). *funCode 1.0 Technical Report*. https://doi.org/10.48693/28. Universität Osnabrück.

Lewin, D. (1982). „A Formal Theory of Generalized Tonal Functions“. In: *Journal of Music Theory* 26.1, S. 23–60. ISSN: 00222909. URL: http://www.jstor.org/stable/843354.

Maler, W. (1931). *Beiträge zur durmolltonalen Harmonielehre*. München: Leuckart.

Nápoles López, N. und I. Fujinaga (2020). „Harmalysis: A Language for the Annotation of Roman Numerals in Symbolic Music Representations“. In: *Music Encoding Conference Proceedings 2020*. Hrsg. von E. De Luca und J. Flanders. Humanities Commons, S. 83–85. DOI: 10.17613/380x-dd98.

Rameau, J.-P. ([1722]1965). *Traité de l'Harmonie*. New York: Broude.

Reimar, H. (2021). *Die Natur der Harmonik*. Universitätsverlag Rhein-Ruhr.