MYKDF — DER KUNST DER FUGE OWN MIXDOWN, REALIZED WITH TSCORE AND SIG/ADLIB

Markus Lepper
semantics gGmbH
Berlin, DE
post@markuslepper.eu

Baltasar Trancón y Widemann
Technische Hochschule
Brandenburg, DE

ABSTRACT

The project myKdF gives music enthusiasts an interactive computer application which allows listening to *Die Kunst der Fuge* by Johann Sebastian Bach in dynamically changeable self-defined arrangements. This intends a deeper exploration of different aspects of this work by users not capable of playing it themselves on an instrument. Main result is a stand-alone Java application, which can run on any contemporary computer system.

The data model is constructed on the tScore meta-meta-model for the semantics of music notation; sound synthesis and user interaction are realized with the sig/adlib system for realtime signal processing. Furthermore, the data model has been used to produce a print publication and for automated extraction of statistical data and structural properties.

Notation systems and languages with very different contents play a fundamental role in the overall architecture.

1. INTRODUCTION

The initial idea of the project myKdF is to give music enthusiasts an interactive computer application which allows listening to *Die Kunst der Fuge* by Johann Sebastian Bach in dynamically changeable self-defined arrangements. This intends a deeper exploration of different aspects of this work by users not capable of playing it themselves on a keyboard or in a chamber music ensemble.

The second idea is to make the necessary computer model of that work available for automated analysis, by ourself and others.

Third, the strengths and weaknesses of the relatively novel and still developing products tScore and sig/adlib have heavily been tested by programming myKdF.

Notation plays a fundamental role in this project: historical sources stand against contemporary technology, and the translation between the different formats (*meta-models*) is not just mere necessity, but brings further insight into their semantic rules, expressiveness, and limitations. Here three notation meta-models have been constructed with our meta-meta-model tScore—others have been programmed ad hoc.

The software can be downloaded as a executable jar file from `http://bandm.eu/downloads/myKdF.jar`, the sources (Java, tScore, etc.) are available under CC-BY-SA-NC license on a source hosting platform.

The work is still in progress and some minor features described in the following may be not yet operative.

2. DEGREES OF FREEDOM FOR THE LISTENERS

Figure 1 shows a screen shot of the application. The user can select one of the 18 movements , arrange its visual and audio representation, and operate the player. The movements are called *contrapunctus* in the original score and in this article—abbreviated like cp 1.

The very fist idea had been to give the listeners the freedom of assigning volume, timbre (=waveform and envelope), and panorama position to each voice separately. It soon turned out that these assignments bring more didactic effect when applicable not only to voices as a whole, but to distinct musical material within.

These material layers are projections of different aspects of the composition. The very first layer is always present and contains *mere counterpoint*, that is melodic material which has not been qualified as thematic. This includes (naturally) everything which sounds in the so-called *episodes* (German: *Zwischenspiele*), in which no theme entry in any voice. The other layers are all theme entries, which can be categorized in different ways, depending on the contents of the selected contrapunctus—see section 3.2 for details.

The user can further start and pause the sequencer and move its position. They can select different tuning modes (see below). The role of the tuning parameter slider depends on the selected tuning algorithm.

Each voice can be given a panorama position. The sliders for the materials are labelled according to the selected audio representation; sliders not required in the current contrapunctus appear disabled.

For each material the volume, the wave form (sine/square/saw/triangle) of the table-driven synthesis and the sound envelope (legato, meno legato, portato, staccato) can be selected. [1]

The area under the mixer shows the formplan of the contrapunctus. The different appearances of the thematic boxes

[1] For convenience, each of these selections can be distributed to all materials of the same voice by holding the "shift" key when releasing the mouse, to the corresponding materials of all voices by "control" and to all materials of all voices by holding both keys.

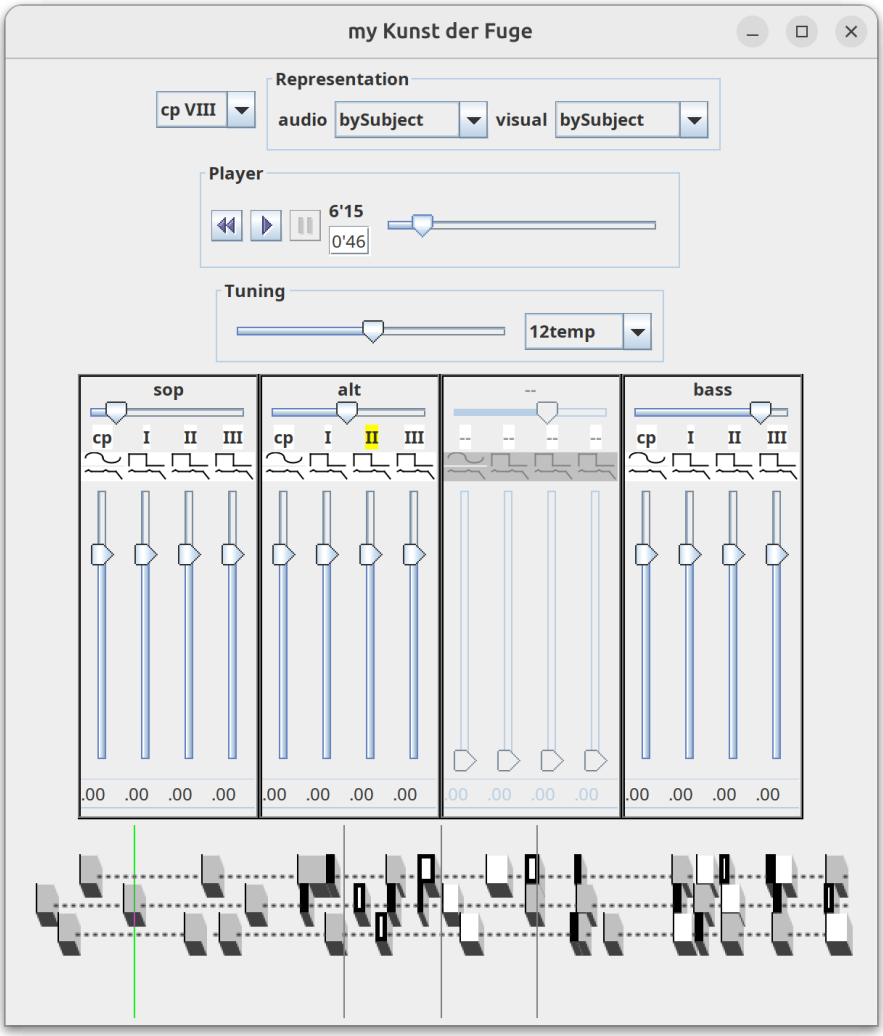


Figure 1. Screenshot of the application.

realize the selected *visual representation*. The separator lines mark the center of the contrapunctus and the positive and negative Golden Ratio. The green line is a cursor which moves synchronously with the sound.

3. SCORE DATA

3.1 Original Music Model

The project is based on the data files which contain the scores of the contrapunctus of *Die Kunst der Fuge*. These data are *models* which have one common definition of syntax and semantics, the *meta-model* `Score_cwn`. These in turn are instances of the *meta-meta-model* TScore. All these three architecture levels are realized as Java classes and instances. [1, 2, 3]

Our edition is almost completely identical with that by Wolfgang Graeser. [2] [4] The print version of our model can be downloaded from `http://bandm.eu/downloads/kdf.pdf`.

[2] We skip his "contrapunctus XVIII", which is just an arrangement for two harpsichords of cp 17, and use this number for the very last, unfinished movement.

TScore is a text-based formalism and its implementation for easy notation of any time-indexed data. The fundamental paradigm is co-algebraic: Voices and timepoints are distinct objects—necessary for all of them is an identifying name but no further properties. Each pair of voice and timepoint can constitute an event. Parameters can freely be attached to voices, timepoints, and events, without contributing to their identity.

The input text format follows the well-proven format of a conventional orchestra score: time flows from left to right, broken into staves which are vertically stacked; voices are vertically stacked within a staff, and synchronous events are more or less horizontally aligned.

Beyond these basic rules, everything is pluggable: The time model itself, all event parameters, their representation syntax and the resulting semantic data objects.

`Score_cwn` is a meta-model for a basic version of Common Western (Music) Notation. It supports easy notation of rhythms defined by metrics, changes of metric, functional pitches (= pair of white note and accidental = *spelled pitches (SP)* as defined by [5]), with "running octaves" and in different locales, dynamics, articulation, and ornaments.

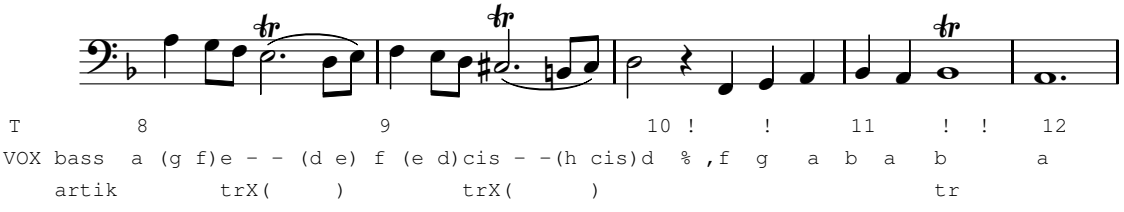


Figure 2. Historic notation vs. semantic encoding, cp 17 ms. 8–11.

```
T          156   157 158  162   163 167    168  169    172        173 174
VOX sop                                    - II                      -
VOX alt    - .III              x                        -.. IIu/2>    x
VOX tenor                           - .III                           x
VOX bass         - II      - ..x     - IIu> - x
```

Figure 3. Excerpt from the `formplan-fuge` score for cp 18. (Sign – stands for continued events; dots stand for metrical dotted notation; both are inherited from the tScore CWN generic format. For the specific symbols see Table 1.)

mat	::=	*cp* | *th* | `%`
cp	::=	`x` | (`1` | `2` | `3` | `4`) *modc*
modc	::=	`u`$^?$ & `>`$^?$
th	::=	*base* *mod*
base	::=	`D` | `C` | ((`I` | `II` | `III` | `IV`)(`D` | `C`)$^?$)
mod	::=	`u`$^?$ & `>`$^?$ & ((⋆ | /)(`2` | `3` | `4` | ...))$^?$

The symbol `%` stands for a pause; *cp* allows for different counterpoint material (only `x` and `1` are used in the context of myKdF); *base* identifies a theme and its dux or comes form; `u` indicates inversion (German *Umkehrung*); `>` indicates an incomplete entry; ⋆ and / stand for augmentation and diminuition of the rhythm by the given factor. The meta-operators _|_ stand for alternatives, _&_ for permutation and _$^?$ for optionality.

Table 1. Syntax of the `formplan-fuge` sub-language.

The syntax and semantics for all these parameters are pluggable. All parameter tracks can come in pairs, to separate Urtext and additions by the editors. Pitches must be monodic per voice; further extensions like chords and lyrics can be defined by subclassing. This has not been necessary for myKdF; contrarily, dynamics are not used at all, and articulation sparsely.

All tScore-based meta-models are ad-hoc extensible, so the source of cp 4 includes additionally the track `zach` which contains the annotations by Gerd Zacher, who relates this contrapunctus to the *Epistle to the Romans*. [6]

TScore is not intended to represent conventional notation in a one-to-one fashion, but to construct precisely defined *semantic* meta-models and models. The semantics of a notation system (= meta-model) can be defined as a *transformation network* from the graphic symbols in the written, printed, or displayed notation (on the left side of a network diagram as in Figures 5 and 6) to the audible parameters of an execution of a piece (on the right side). [7, 8] Then every encoding is a cut through such a network—so is every model and every tScore model. See for example Figure 2: We know from historically informed performance that the third trill symbol has a different meaning than the others: It does include an implicit *final turn* (German: *Nachschlag*). In case of the others, this is replaced by the explicit notes. Consequently, the semantic-oriented tScore encoding applies two different ornament keywords. These will be rendered identically in print, but differently in audio execution and statistical analysis.

The `Score_cwn` meta-model has a second parameter track `artikHRSG` for articulation and ornaments added by the editors. We have used it in cp 8 m. 24 and 160 bass to add the thematic mordent and in cp 9 eight times to add the thematic pralltriller to the middle of theme II. These additions correspond well to Baroque practices of notating and playing. They are also possible for the thematic fragments in cp 9 soprano m. 47 and bass m. 75.

The generic input format of the tScore meta-meta-model closely follows the well-proven traditional score format and is often suitable for sight-reading, and the syntax of all meta-models can be defined with minimal redundancy. Therefore the data input of the complete *Die Kunst der Fuge* took less than four workdays!

3.2 Formplan Data

As mentioned above, at a certain point of the work it became clear that the users shall be able to influence not only the distinct voices as a whole, but also the material layers within. These layers are defined by data in another tScore-based meta-model. This format inherits all metric and rhythmic notation means from the standard CWN score, but replaces the pitch information by the codes for the materials. Table 1 shows the syntax of this language and Figure 3 an example: For instance, the measure 172 in the alto voice is filled by two events. The first is doubly dotted (duration = $7/8$) and just a hold of whatever comes before. The second event (duration = $1/8$) is an entry of theme II in its inverted form, with all durations shortened by the factor two, and incomplete. In measure 174 some unspecified counterpoint material `x` begins.

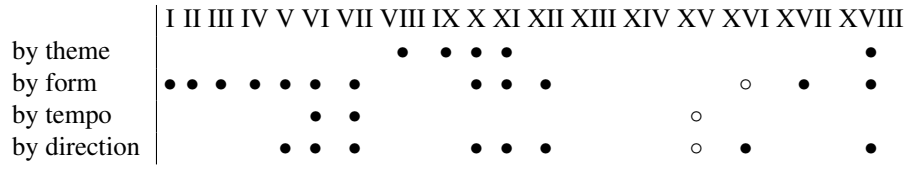


Table 2. Presentation modes offered per contrapunctus. (In XV: direction and tempo augmentation tightly coupled and appearing only once. In XVI: form hardly significant.)

A rendering can be seen in the bottom part of Figure 1.

Whenever the selected contrapunctus features theme entries which differ w.r.t. a particular category, then this category is offered to the user for visual and audio representation. These categories are

by theme for all contrapunctus where multiple themes are combined;

by form difference of dux and comes form, as fundamental for fugue theory;

by tempo whenever a theme appears in different augmentations and/or diminuitions;

by direction difference between recto and inverso, or even between all four dodecaphonic appearances: original, inverted, retrograde and inverted-retrograde. (Only the first two appear in *Die Kunst der Fuge*.)

The user can select which of these aspects shall be expressed in the visual representation in the form diagram shown below the mixer section and in the audio realization. Figure 1 shows the form diagram of cp 8 with different symbols per theme. The labelling of the audio sliders is determined by the selected audio representation mode.

In any case, the mere counterpoint is the very first category. The others are selectable for the different kinds of theme entries if and only if this aspect is realized in the selected contrapunctus, see the survey in Table 2.

When switching the contrapunctus, the audio and visual representation is selected automatically. There are different default priorities:

audio: byTheme → byTempo → byDirection → byForm
visual: byTheme → byDirection → byTempo → byForm

This is because the tempo difference is visible anyhow, by the width of the thematic blocks, without additional graphical means. (Incomplete entries are marked differently anyhow.)

4. THE sig/adlib REALIZATION

MyKdF is a stand-alone Java application. It is realized with sig/adlib, a Java library for sound synthesis and processing. It uses only the built-in means of the Java Runtime Environment (JRE). [10] For the size of the software see Table 3, measured with CLoC [9].

In principle, the sound synthesis could also be delegated to the built-in implementation of General MIDI. Then all sound manipulations could be mapped to MIDI controllers. Our approach (currently still) has the disadvantage of comparatively simple sound synthesis. But it also has significant advantages: (a) It is a totally stand-alone application which requires only a Java runtime environment, which is a JVM and a runtime library that interfaces the platform's sound system. (b) All sound synthesis is under program control and does not depend on the kind and quality of some General MIDI installation. (c) All SP-based tuning systems can be applied globally, not event-wise as for enharmonic MIDI keys.

sig/adlib singleton sources	1,797
sig/adlib parametric sources	2,985
sig/adlib expanded sources	13,593
sig/adlib/lib singleton sources	978
sig/adlib/lib parametric sources	1,774
sig/adlib/lib generated sources	6,680
myKdF Java sources	
model	281
view	499
control	68
analyses	967
tScore sources:	1,493

Table 3. Lines of code of the myKdF project, excluding blank lines and comments. Calculated by the program `cloc` [9].

4.1 The sig/adlib Network

The application follows a model–view–control architecture. The view is realized with components of the JRE Swing library, and the model is the sound-producing network constructed with sig/adlib. Its principal architecture is shown in Figure 4: The largest frame contains the circuit which is instantiated four times, once for each voice. It contains a smaller frame which is instantiated four times per voice for its material layers. (There are additional voices for the very last measures of cp 5 to cp 7 and cp 11, which are hidden to the user and left out in the diagram for readability.)

The very left column of the Figure shows the four wave tables (sine, square, triangle, and saw) which are distributed via a multiplexer to the table-driven generators. The multiplexer is switched by the user's choice in the control layer. In a similar way one of four envelopes is selected and applied to the sound.

The third column shows the eighteen sets of score data. Each consists of one score per voice, and an additional formplan score. The former are multiplexed and fed into a sequencer, one for each voice. The first column of the data is fed into a edge detector, which recognizes every event start and triggers the envelope, the frequency calculation and changing the material register. The next columns of

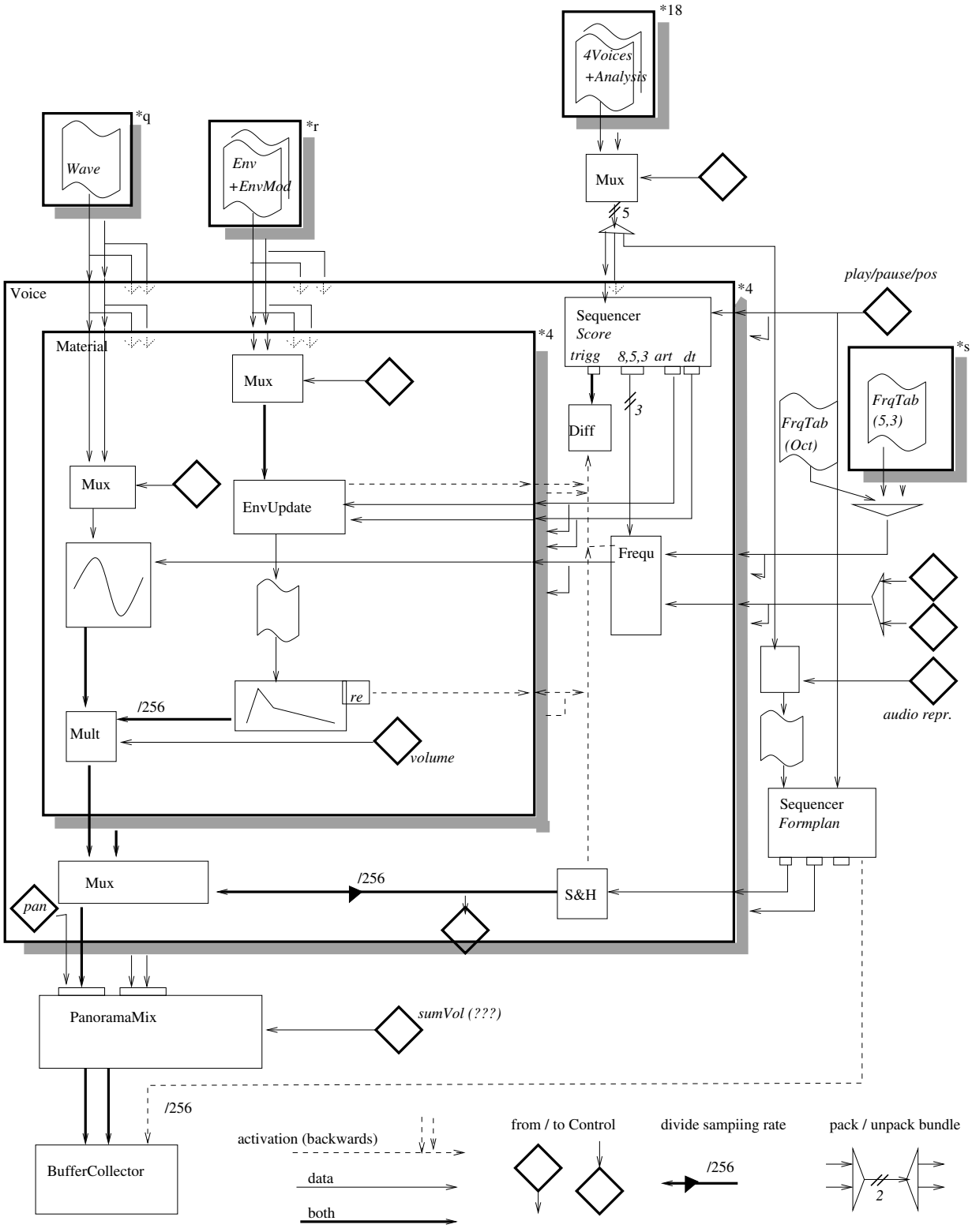


Figure 4. The sound synthesis network, realized with sig/adlib.

each voice score contain the pitch information encoded in octaves, fifth, and thirds. Last not least come the duration of the events.

The formplan data has its own format and goes to a dedicated sequencer which controls the multiplexing of the different material sounds per voice. The rightmost column shows the global tuning tables and the user's tuning controls which flow into the calculation of the technical frequency for each new event in each voice.

The bottom of the Figure shows how the four material layers are multiplexed into the output of the voice, controlled by the formplan sequences, and how the voices are mixed into the stereo output.

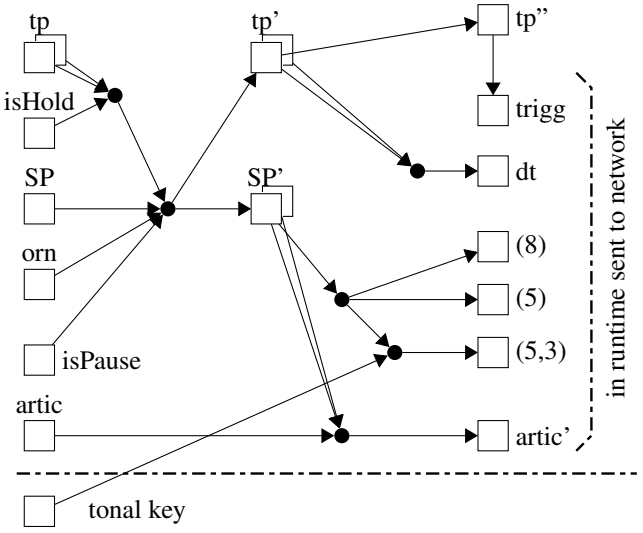


Figure 5. Translation from the tScore CWN format into sig/adlib sequencer data tables. (Staggered boxes stand for parameters of adjacent events.)

4.2 Transition from the semantic tScore score to the real-time sig/adlib score

The application consists of two parts: for realtime execution the symbolic data of the tScore world is translated into data for sig/adlib.

TScore-based data, including the `Score_cwn` meta-model as described above, contains complex, multi-faceted semantic relations. This is implemented with object-orientation and high-order functions. Contrarily, sig/adlib real-time data are fixed-length arrays of primitive values, optimized for realtime execution. Separation into these (or similar) two realms was caused by performance issues in the early days of real-time computer sound synthesis. Today it is still sensible for pragmatic reasons from software technology, applying the principles of information hiding and modularization [11]. And it serves as a cutting knife for exploring the domain theory, since the translation algorithm between these two spheres forces to make meanings and limitations on both sides explicit. For instance, the automated expansion of *ornaments* required exploring and formalizing of different rule systems. Figure 5 shows the data flow between both realms.

Critical points are (a) ornament resolution, which replaces one single input note by a sequence of multiple events; (b) the recognition of pitch repetitions, which is necessary for sharpening the articulation, while genuine articulation marks are rare in Bach's source text; (c) the translation of functional pitch information into combinations of interval coordinates.

The main difference to MIDI real-time execution is that *future information* is easily available for sound rendering, like the intended note duration and subsequent pitch repetition.

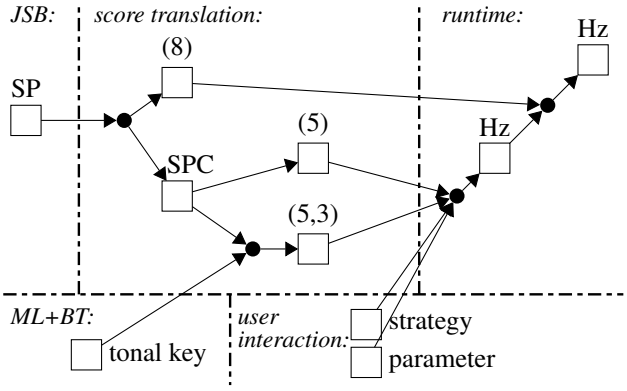


Figure 6. Tuning information flow. SP = spelled pitches. SPC = spelled pitch classes. (8), (5), (3,5) = interval coordinates.

4.3 Tuning

Figure 6 shows the data flow for a single event: The spelled pitch is translated into interval coordinates when the scores are translated.

The octave coordinate is used in any case. Other coordinates are calculated for the pitch class modulo octave. For SPs these classes are the *spelled pitch classes (SPCs)* [5].

The further coordinates specify multiples of fifths and thirds (= Euler coordinates) to realize pure tuning, and fifths only for all other tunings (equally tempered, mean tone, transposition invariant, etc.) Compared to SPs, these encodings allow much easier arithmetic operations like comparison, transposition, or tuning calculation.

Depending on the tuning strategy and its parameters, both selectable by the user interactively, even when playing, the effective frequencies are calculated.

The (5,3) encoding for the *pure tuning* interpretation of the SPCs requires additional information about the currently ruling *tonal key*. Currently this has been added manually by the authors, but could be supplied by an analysis algorithm in future versions of the software.

4.4 The Envelope Update Language (EUL)

For all envelope curves needed in a sound synthesis project, we offer the realization by a two-level language approach, see Table 4.

The lower level is a list of segment descriptors which control in realtime the changes of the output value of the envelope generator, segment by segment. Read as a table, each line contains an operation code and one or two numeric values, interpreted according to that opcode.

On a second level, the values used therein (d, f, v, and i in the Table) are given as an expression following the syntax definition *exp*, including realtime input parameters. In the current implementation, these are sampled once when a segment is started. (But this could change in a different future application context.) The expressions can use all score and all interactive parameters. The main aesthetic difference between realtime MIDI driven ADSRs and envelopes in a score-driven context is that the future length of the event is known and allows adaptive sound shaping.

dtSeg_	d		d gives the end point of this segment relative to its start.
dtEnv_	d		d gives the end point of this segment relative to the start of the complete envelope.
_fin		f	f shall be reached at the end of the segment; calculate the necessary increment accordingly.
_inc		i	i is the increment value which is added to the current value during the whole segment.
_inc		0	hold the current output value for the duration of the segment.
dtSeg_fin	0	v	jump to the value v and start next segment immediately.
inc_fin	i	f	add the increment i until final value f is reached.
end	–	–	keep output value; end processing.
ferm	–	–	keep output value; wait for external trigger to continue with next segment.

exp ::= *value* | *project* | *arith1* | *arith2*
| *log* | *if* | *comp* | *select*
value ::= `input`(*ident*) | `const`(*digits*)
project ::= `project` *exp exp exp exp exp*
// `project`$(a, b, c, d, e) = \frac{a-b}{c-b} * (e - d) + e$.
arith1 ::= (`not` | `neg` | `inv`) *exp*
arith2 ::= (`add` | `sub` | `mul` | `div` | `min` | `max`) *exp exp*
log ::= (`and` | `or` | `xor`) *exp exp*
if ::= `if` *exp exp exp*
comp ::= `equals` *exp exp exp*$^?$
// third parameter is the epsilon.
| `range` *exp exp exp*
// whether $a \leq c \leq b$
select ::= `select` *reference_to_envelope_table*

Table 4. Envelope Update Language (EUL): segment descriptor tables and expression language for the values.

The expression `select` is evaluated the same way, but operates on a meta-level: it selects from a multitude of table definitions. This is useful to switch to an alternative envelope form—for instance for extremely short or pianissimo events.

Figure 7 shows an example.

EUL is currently realized as a deeply embedded domain-specific language—the expression layer is interpreted by a stack machine whenever an event starts. This shall be replaced by a compiler into sig/adlib primitives, which is much faster.

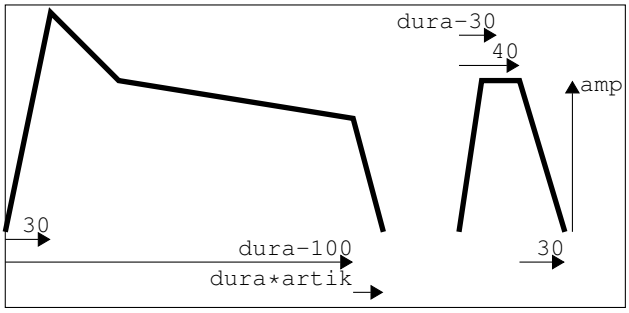
4.5 User Interactions Protocol

In sig/adlib, run-time user interaction are handled by `Post` objects. These can receive data from various sources, like GUIs, MIDI devices, sequencers, or programmed agents, and deliver it to synthesizers, protocol files, visual GUI feedback, etc. This architecture allows easy switching of the data source, even at runtime, and connecting it to a dynamic multitude of drains.

Every such Post object has a unique name and can write its realtime behavior to a protocol file. These files can be converted into the myKdF arrangement score format, see Figure 8 for an example and Table 5 for the syntax definitions.

This conversion is crucial (a) to make the protocol easy readable, and (b) to make it editable for modified replay.

(This part of the system is currently still under construction.)




```
if (range(0, 70, input(dura)),
    select(short), select(normal))
normal:
  dtSeg_fin const(30)       const(1.0)
  fin_Inc   input(amp)      const(0.005)
  dtEnv_inc sub(input(dura), const(100))
                            const(-0.004)
  dtSeg_fin mul(input(dura), input(artic))
                            const(0)
short:
  inc_fin   const(0.01)     input(amp)
  dtEnv_inc max(sub(input(dura),const(30)),
                const(40))
                            const(0)
  dtSeg_fin const(30)       const(0)
```

Figure 7. Two envelopes and their EUL source.

`VOX` *voiceName* *voiceData**
`VOX` *voiceName−matName* *matData**
voxData ::= `pan` (0..100) `/`$^?$
// Panorama position and interpolation select.
matData ::= (`sine` | `square` | `saw` | `triangle`)$^?$
$\&_{[;]}$ (`pizz` | `port` | `menoLeg` | `legato`)$^?$
$\&_{[;]}$ (0..100 `/`$^?$)$^?$
//Wave, envelope, and loudness with interpolation select.

Table 5. Syntax of the user protocol sub-language. The meta-operator $\_\&_{[;]}\_$ stands for permutation separated by `;`—for the others see Table 1,

5. SCORE PUBLISHING

The `Score_cwn` meta-model includes an automated translation into lilyPond source text.[12] Our print edition of *Die Kunst der Fuge* can be downloaded from `http://bandm.eu/downloads/kdf.pdf`. Page turns are optimized for piano playing; being in four staves and old keys, this is challenging but enjoyable.

```
PARS myInterpretation
  cp = 4
  audioRepr = byDirection visualRepr = byDirection
T        1              12   13      25            29     30       50
VOX sop  pan0/                                            pan100
    cp   sin;pizz;30/   49/  port    70                    50
    I    port;saw;60                                 square
VOX alt  pan20
    cp   sin;pizz;30
   (etc.)
```

Figure 8. A user interpretation protocol, converted to the corresponding tScore format.

6. STATISTICAL EVALUATION

Our models of *Die Kunst der Fuge* are also available for computer-assisted analysis. The following questions have addressed so far:

Tonal Material. Table 6 shows the SPCs found in every contrapunctus. The first contrapunctus obviously takes the role of laying the basis for the whole architecture: It contains twelve pitch classes, one for each key of the keyboard octave. Thus the first contrapunctus is the only which could be played in the traditional meantone temperament. The very last SPC is introduced not before m. 60. It completes the tonal spectrum by turning to the sub-dominant, which is a conventional signal for "we come to the end of the piece".

The maximum is reached with 17 SPCs in cp 4 (which was composed later [6]) and the triple fugue cp 8. This maximum set contains all white keys once and all black keys twice. The covered part of the circle of fifths ends immediately before the first SPC which notates a white key with an accidental: above c-flat and below e-sharp.

The very last contrapunctus, probably planned as a quadruple fugue, stays in its unfinished state below the maximum. This is a severe contrast to its harmonic complexity, which even includes far-out pseudo functions (German *Scheinfunktionen*) as in ms. 224–225. So perhaps here the maximum SPCs or perhaps even more would have appeared in a complete version.

False relations and diatonic splits. The English term *false relation* stands for two notes with the same generic pitch class and two different accidentals, appearing in two different voices. If these notes are adjacent then the German term is *Querstand*—when they are even overlapping it is considered a more pointed effect and called *diatonische Spaltung (diatonic split)*. One big motivation for constructing the data model was the statement of Ploeger [13] that there is no diatonic split in *Die Kunst der Fuge*. Knowing of one from playing, we wanted to search systematically and applied an algorithm to the data model. The rich results are out of scope of this article and will be published shortly.

Symmetric Chords, Grips, and Noteheads. According to Zacher [6] and our own results [14], in many works a composer applies a *second system* of specific and hidden rules, additionally to the conventional and obvious, to increase the complexity of the result and to make their necessary decisions easier. (Zacher: "[Zur] Diversifikation [= zur] Absicherung gegen das routinierte Einerlei der Fugenkomposition.") We have always assumed that *graphical symmetry* plays such a role in *Die Kunst der Fuge*. This could be proven by automated analysis.

	g♭	d♭	a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯	a♯	
1:				e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯			12
2:			a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		14
3:		d♭	a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯	a♯	16
4:	g♭	d♭	a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯	a♯	17
5:				e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		13
6:		d♭	a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		15
7:			a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯			13
8:	g♭	d♭	a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯	a♯	17
9:			a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		14
10:				e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		13
11:	g♭	d♭	a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		16
12:					h♭	f	c	g	d	a	e	h	f♯	c♯	g♯			11
13:					h♭	f	c	g	d	a	e	h	f♯	c♯	g♯			11
14:		d♭		e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯			13
15:				e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		13
16:			a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯			13
17:			a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯			13
18:		d♭	a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		15

Table 6. Spelled pitch classes contained in the 18 contrapunctus.

7. ACKNOWLEDGEMENT

Many thanks to Wolfgang Grieskamp, Sammamish, WA, for substantial financial support.

Many thanks to the anonymous reviewers for valuable hints for future work.

8. REFERENCES

[1] M. Lepper and B. Trancón, "tscore: Makes computers and humans talk about time," in *Proc. KEOD 2013, 5th Intl. Conf. on Knowledge Engineering and Ontology Development*, J. Filipe and J. Dietz, Eds., instincc. Portugal: scitePress, 2013, pp. 176–183, http://markuslepper.eu/papers/tscore2013.pdf[20231003].

[2] ——, "Translets —— parsing diagnosis in small dsls, with permutation combinator and epsilon productions," in *Tagungsband des 35ten Jahrestref-*

fens der GI-Fachgruppe "Programmiersprachen und Rechenkonzepte, ser. IFI Reports, vol. 482. University of Oslo, 2018, pp. 114–129, http://urn.nb.no/URN:NBN:no-65294[20230920].

[3] M. Lepper, *Example Instances of the TScore Projekt Infrastructure*, 2023, http://markuslepper.eu/sempart/tscoreInstances.html[20230920].

[4] J. S. Bach, *Die Kunst der Fuge BWV 1080*. Wiesbaden: Breitkopf und Härtel, 1924, *edt.* W. Graeser. [Online]. Available: https://s9.imslp.org/files/imglnks/usimg/1/1d/IMSLP469177-PMLP5843-Bach,_J.S._-_The_Art_of_the_Fugue.pdf

[5] J. Hentschel, F. C. Moss, A. McLeod, M. Neuwirth, and M. Rohrmeier, "Towards a Unified Model of Chords in Western Harmony," in *Music Encoding Conference Proceedings 2021*, S. Münnich and D. Rizo, Eds. Humanities Commons, 2022, pp. 143–149.

[6] G. Zacher, "Der geheime Text des Contrapunctus IV von Bach," *Beiträge zur Gregorianik*, vol. 13/14, 1992.

[7] M. Lepper, *de Linguis Musicam Notare*. Osnabrück: EpOS, 2021, https://www.epos.uni-osnabrueck.de/viewer/web/?id=150.

[8] M. Lepper and B. Trancón, "Critical semantic properties of music notation datasets," in *Proceedings of Music Encoding Conference 2024*, Denton, TX, USA, 2024, https://works.hcommons.org/records/ppg74-g6223.

[9] A. Danial, "cloc – count lines of code (v1.92)," Tech. Rep., 2021, https://github.com/AlDanial/cloc [20231003]. [Online]. Available: https://doi.org/10.5281/zenodo.5760077

[10] B. Trancón y Widemann and M. Lepper, "Sig adlib — mostly compositional clocked synchronous data-flow programming in java," in *Tagungsband des 35ten Jahrestreffens der GI-Fachgruppe Programmiersprachen und Rechenkonzepte*, ser. IFI Reports, vol. 482. University of Oslo, 2018, pp. 31–48. [Online]. Available: http://urn.nb.no/URN:NBN:no-65294

[11] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Commun. ACM*, vol. 15, no. 12, p. 1053–1058, Dec. 1972. [Online]. Available: https://doi.org/10.1145/361598.361623

[12] H. W. Nienhuys and J. Nieuwenhuizen, "Lilypond, a system for automated music engraving," in *Proceedings of the XIV Colloquium on Musical Informatics*, Firenze, 2003, pp. 167–172.

[13] R. Ploeger, *Studien zur systematischen Musiktheorie*. Lilienthal, Bremen: Eres, 1990.

[14] M. Lepper, "Das Zweite System," *senza tempo (blog)*, 2015, http://senzatempo.de/ston2015081400.html.

A. APPENDIX: CAVEATS FOUND WHEN DEFINING THE FORMPLAN DATA

Writing a graphic representation of the form of a particular fugue is always a *creative process of interpretation.* This is understood from an abstract philosophical point of view, but it becomes strikingly obvious as soon as this formplan is notated in a concrete data format like tScore. Especially some rough edges indicate fundamental aesthetic problems and properties:

A.1 Recto versus inverso in cp 1–cp 4

Taking cp 1 to cp 4 each on their own, they only have one theme each which appears in only one direction. This thus can be called recto and no modifier must be appended to any theme code.

But in the context of *Die Kunst der Fuge* as a whole, these themes are the main theme appearing as recto in the former two and as inverso in the latter two of these contrapunctus. Therefore it could also be sensible to mark *all* thematic entries in cp 3 and cp 4 by the modifier `u`. We did not do so, because it would not add any flexibility to the software for its users. But it must be taken into account when e.g. applying analyses to the data model as a whole.

In cp 5 to cp 7 both forms are combined, so that the placement of the modifier `u` is out of any doubt.

A.2 Recto versus inverso in cp 8 and cp 11

Even more complicated is the relationship of cp 8 and cp 11. Cp 8 introduces two new themes, which it it later combines with the main theme of the work. This happens in form of a *metalepsis*—several things are first presented as unrelated, and only later, in retrospection, the combination becomes logical, even unavoidable.[6]

The themes II and III both appear only in one direction, which most naturally is received as their recto. But later they are combined with (a rhythmic variant of) the main theme in its inverso direction, when compared to its initial appearance in the overall work. We decided to use the undecorated codes `II` and `III` for the directions in which the new themes appear here for the first time, and the decorated `Iu`, which reflects the larger context.

In cp 11 now the same three themes appear again, but inverted: It starts with theme I in its recto form, followed by II and III inverted, now labelled as `I`, `IIu`, and `IIIu`. Later in this movement all themes appear in *both* directions: `Iu`, `II`, and `III` appear additionally. This confirms that our choices for the labels in cp 8 where sensible. Alternatively, all three themes in cp 8 can be labelled with `u`, but this does not correspond to the psychological effect: We do not expect an inversion to sound earlier than the original.

A.3 Automatic detection of incomplete theme entry

The implemented formplan meta-model allows to specify a numeric limit, by a parameter setting like

```
minimumCompletum = 4 1/8
```

Every theme entry shorter than this is automatically tagged as incomplete. This does work well in most contrapunctus,

but interestingly not in cp 7: Some entries have a shortened *initial* note, i.e., they start at a weaker metric time point than usual, see tenor ms. 23, soprano ms. 38 and 42, alto ms. 51, bass ms. 47.

Similar, there is an acceleration for a short subsection in the entry alto ms. 14.

In all these cases the overall length is shortened, but the entry cannot be called incomplete, which is a psychological term related to a premature *ending*.

A.4 Problematic limits of theme entries

All themes in *Die Kunst der Fuge* are clearly delimited, except the theme III in cp 8 and cp 11, see Figure 9. This is a sequence (in the musical sense of the word: repetition with transposition) which can break off after different numbers of repetitions. The fact that it is really a full-fledged theme and must be honored as such is doubtlessly clarified by the facts that (a) it appears in a fixed combination with both other themes (German: *kontrapunktischer Verband*), that (b) is subject to a maximal stretta in cp 11 ms. 170, and that (c) it is the only structure in the whole work which contains the very significant pitch repetition. But when exactly do the entries of this theme end? And when do they start?

Its motif kernel has three repetition notes and one singleton. As exposed in cp 8 alto ms. 40, it jumps down a third from the repetitions to the singleton, and goes up a second to the next repetition, see Figure 9. In the formplan score this is labelled as `III`.

With its re-appearance in cp 11, these two interval directions are maintained, but the sizes are exchanged, see Figure 9, tenor ms. 90: the melody goes down a second and up a third. Thus the overall direction of the sequence is inverted, but not the directions internal to the motif. Not before ms. 130 soprano a verbatim inversion appears, which jumps the third upward after the repetition. We decided to mark both versions which go globally upward as `IIIu`, but of course this decision is subjective. (Perhaps the set of possible decorations in the formplan meta-model should be enhanced by some freely useable variant indication?)

In all of its versions, finding the boundaries of the entries of this theme is problematic. In the formplan scores, we set the end points by melodic criteria, corresponding to their role in the application, namely to define the timepoints when to switch the sound register. The end points of theme III in cp 18 have similar problems, see for instance m. 161 pp.

Also the starting points of the entries of theme III cp 11 are ambiguous, but less problematic to define: We choose points shortly before the first significant repetition, which corresponds to the listener's recognition.

A.5 Problematic automated offer of representation modes

The representation modes offered to the user when selecting a movement, as shown in Table 2, are derived from the entries in the formplan data.

Cp 18 has one single incomplete entry (also called *false entry*, German: *Scheineinsatz*) we marked as diminuition

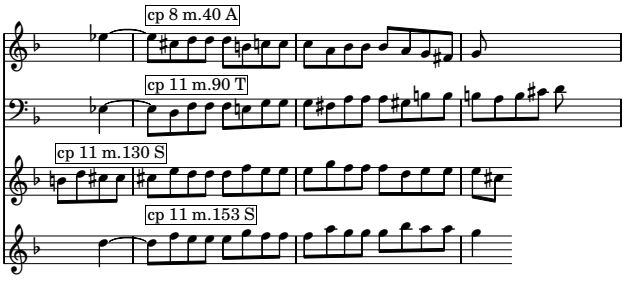


Figure 9. Ambiguous definitions of theme III and IIIu.

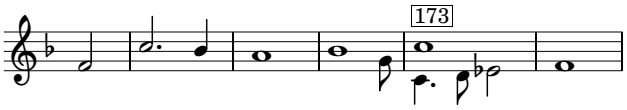


Figure 10. Diminuition entry in cp 18 m. 173 "IIu/2>" or "Iu>".

of theme II, namely ms. 173 alto, see Figure 10. As a consequence, the implementation offers the theme representation mode byTempo as appropriate. But when the user selects this for visual representation, they immediately see that a different tempo appears only once, and thus this mode is hardly useful for sound control. Since this entry is incomplete, it can also be heard as one of theme I, the overall main theme—it functions as a kind of proof object. So it should not be left completely unmarked.

A.6 Borderline cases between theme entry and derived counter-point material

A fundamental problem, often arising in the psychological analysis of listening and in didactic presentation, is drawing a line between regular theme entries and contrapuntal material extracted therefrom.

From a formal point of view, every fragmentation (German *Abspaltung*) from the *start* of a theme can also be counted as an incomplete entry. Whether this is appropriate, depends on the psychological situation of the recipient. There are famous moments where this dichotomy is played with, see Beethoven, Eroica, Finale, ms. 143–145 and Bruckner, Fifth Symphony, Finale, ms. 258–262.

Here we have a different situation in cp 5, ms. 53–56 and ms. 65–69, where the complete setting *as a whole* can be heard either as thematic *stretta* or as counterpoint material derived from (a head fragment of) the main theme.

Similar cases shows cp 8 with theme II at m. 89 and m. 114, and with theme III at m. 74 (= first solo theme entry) versus m. 77 (= theme entry or mere counterpoint?).

In cp 11 we find multiple such situations with respect to theme III, see ms. 97–99, etc. Again our formplan data aims for satisfying audio results.

The shortest possible fragment which can be heard as an incomplete entry is the sequence C3–F2 in cp 18 m. 168.

A.7 Overlapping Abspaltung

The formalization of analysis and mark-up also shows the limitations of the chosen meta-model. The crucial ms. 14–

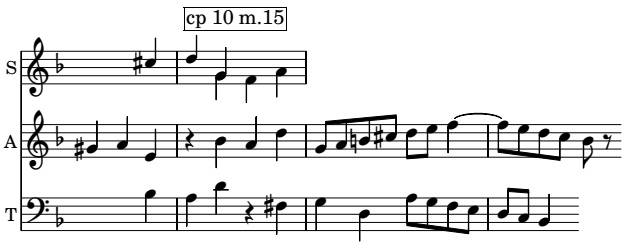


Figure 11. Overlapping incomplete entries in voice S.

16 in cp 10 show a stretta between recto and inverso in alto and tenor, together with an incomplete entry in the soprano, see Figure 11. But indeed these few notes can even be heard as *two* incomplete entries which *overlap*: the third note G4 is the last note of the first and the first of the second entry. Such an overlap is not foreseen in the metamodel. Future applications to models outside *Die Kunst der Fuge* perhaps require to extend it.

A.8 Insignificance of the dux/comes form

While the distinction between dux and comes form is highly relevant for the architecture of many contrapunctus, there are others where it can be neglected:

In cp 8 there the second entry of the main subject (bass m. 99) *programmatically* shows the comes form. We did not mark it as such in the formplan because it is the only one.

In cp 11 the main theme appears nearly always in the dux form. The comes form nevertheless is present, so the entries are coded completely in the formplan data. Changing the form only changes the interval of the very first anacrusis (in contrast to the standard process as applied in the very first comes).

This also holds for cp 16, and also in only few entries: S ms. 9, A ms. 48, B ms. 63.

In cp 17 the figured version of the theme after the first development group (= all entries after ms. 21) have all the same constant form. Our formplan score contains the qualifiers `D` (7 times) and `C` (only twice), which is correct in the strict formal sense, but aesthetically questionable.