

*mycrsf* basierte Geisteswissenschaft

Release 1.4

# Musikwissenschaft mit $\text{\LaTeX}$



**Wie man Musikbeispiele mit Open-Source-Tools in seine wissenschaftliche Texte integriert.**

**Eine selbstreferentielle Anleitung**

Karsten Reincke

11. September 2020

*Dieses Tutorial erläutert, wie man musikalische Beispiele mit Open-Source-Mitteln in L<sup>A</sup>T<sub>E</sub>X-Texte einbettet. Dazu sieht es Notensatzsysteme, Editoren, Konverter und Tools, die Notentexte erzeugen, verändern und in den L<sup>A</sup>T<sub>E</sub>X-Text integrieren. Und es skizziert, wie man ganze 'Produktionsketten' aus Frontendsystemen, Konvertern und Backendsystemen 'zusammenstößt'. Zuletzt entsteht so eine 'Landkarte' verschiedener Wege.<sup>1</sup>*

---

<sup>1</sup>) Unser Text wird unter den Bedingungen der *Creative Commons Share Alike*-Lizenz (CC BY-SA 4.0) veröffentlicht: Sie dürfen das Material – grob gesagt – in jedem Format oder Medium vervielfältigen und weiterverbreiten, es remixen, verändern und darauf aufbauen – und zwar für beliebige Zwecke, sogar kommerzielle –, sofern Sie angemessene Urheber- und Rechteangaben machen, einen Link zur Lizenz beifügen, Ihr abgeleitetes Werk unter derselben Lizenz verbreiten und angeben, ob und wo Sie das Original in welcher Hinsicht geändert haben. Details dazu finden Sie unter ⇒ <https://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Das Recht, diese Arbeit im Rahmen des üblichen wissenschaftlichen Verfahrens zu zitieren, bleibt davon unbenommen: es gibt keine Pflicht, das zitierende Werk unter dieselbe Lizenz zu stellen. Die Bedingung der *angemessenen Urheber- und Rechteangaben* erfüllen Sie, indem Sie in ihrem Werk an prominenter Stelle den Text einfügen: *Abgeleitet vom GitHub-Projekt <https://github.com/kreincke/musicology.ltx>, das von © 2019ff K. Reincke und Kontributoren unter den Bedingungen der Lizenz CC-BY-SA veröffentlicht worden ist*

( Weil wir selbst diese Arbeit von *mycsrf* abgeleitet haben, fügen wir beispielhaft den Hinweis hinzu: *Format abgeleitet von [mind your Scholar Research Framework](http://fodina.de/mycsrf) ©K. Reincke CC BY 3.0 DE <http://fodina.de/mycsrf> )*

## Vorwort

Ich selbst hatte mir zu Beginn eines größeren musikwissenschaftlichen Projektes gewünscht, ein solches Tutorial zu haben: *LaTeX*, *BibTeX* und *JabRef* waren mir schon vertraut. Meinen optimalen Zitierstil für das Schreiben geisteswissenschaftlicher Arbeiten hatte ich bereits konfiguriert<sup>2</sup> und dokumentiert<sup>3</sup>. Unklar war mir 'nur', wie man Notentexte und musikalische Analysen in *LaTeX*-Texte einbindet.

Das sollte eigentlich nicht kompliziert sein. Man bräuchte dazu doch nur ein Notationssystem, das Notentext erfasst und das – als Teil des *LaTeX*-Quelltextes – das Notenbild in das eigentlich Werk hinein generiert. Leider schwiegen sich meine sehr guten, einschlägigen *LaTeX*-Bücher dazu aus<sup>4</sup>, selbst wenn sie auch Randbereiche behandelten<sup>5</sup>. Die entsprechende Internetrecherche überrollte mich dagegen: so viele Notationssysteme und Tools, aber kein systematischer Überblick.<sup>6</sup>

Wollte ich meine Arbeit also nicht gefährden und Sackgassen vermeiden, musste ich die gegebenen Möglichkeiten zuerst sichten. Andernfalls wäre ich Gefahr gelaufen, zuletzt doch 'aufs falsche Pferd gesetzt' zu haben. Was ich brauchte war eine technisch fundierte 'Landkarte' der Methoden und Tools, die mir den besten Weg weisen konnte. Und so traf mich – wieder einmal – die Erkenntnis:

*Was man im Internet nicht findet, muss man selbst hineinstellen  
- am Besten unter einer Lizenz, die jedem die freie Wieder- und  
Weiterverwendbarkeit garantiert.*

- 
- <sup>2</sup>) vgl. *Reincke, Karsten*: mycsrf; mind your Classical Scholar Research Framework; 2018 ⇒ <https://fodina.de/mycsrf/> – RDL: 2018-12-21, wp..
- <sup>3</sup>) vgl. *Reincke, Karsten*: Dienst am Leser, Dienst am Scholaren. Über Anmerkungsapparate in Fußnoten - aber richtig: Release 2.0; (Geistes-) Wissenschaftliche Texte mit jurabib; 2018 ⇒ <http://kreincke.github.io/mycsrf/examples/scholar-fono-de.pdf> – RDL: 2018-12-21, S. 2ff.
- <sup>4</sup>) vgl. *Voß, Herbert*: Einführung in *LaTeX* unter Berücksichtigung von pdf*LaTeX*, Xe*LaTeX* und Lua*LaTeX*; Berlin: Lehmanns Media, 2012 (= dante): ISBN 978-3-86541-462-5, S. vi ff, insbesondere 905 u. 909: das umfangreiche Register erwähnt weder Musik im allgemeinen noch LilyPond oder MusiX*TeX* im Besonderen.
- <sup>5</sup>) vgl. *Mittelbach, Frank* u. *Michel Goossens*: Der *LaTeX*-Begleiter; unter Mitarbeit v. J. Braams, D. Carlisle u. C. Rowley; mit Beitr. v. Christine Detig u. Joachim Schrod; 2., überarb. u. erw. Aufl.; München, Boston [... u.a.O.], 2005: ISBN 3-8273-7166-X, S. vii ff, insbesondere 1080 u. 1087: auch dieses umfangreiche Register erwähnt weder Musik im allgemeinen noch LilyPond oder MusiX*TeX* im Besonderen..
- <sup>6</sup>) Rühmlich die Ausnahme von *Thoma, Martin*: How to write music with LaTeX; 2012 ⇒ <https://martin-thoma.com/how-to-write-music-with-latex/> – RDL: 2018-12-19, wp.. Allerdings ging sie nicht in die Tiefe, die ich benötigte.

# Inhaltsverzeichnis

<b>Vorwort</b>	<b>3</b>
<b>0 Der Kontext</b>	<b>6</b>
0.1 Anliegen	6
0.2 Kriterien	11
<b>1 Leichtgewichtige Teillösungen</b>	<b>13</b>
1.1 Sonderzeichen: der Standard (★★★)	14
1.2 Andere Sonderzeichen: wasysym (★★★)	14
1.3 Noch mehr Sonderzeichen: musicography (★★)	15
1.4 Ganz viele Sonderzeichen: harmony (★★★★★)	16
1.4.1 Rhythmik	16
1.4.2 Harmonik	16
<b>2 Backends: Komplexe Notationssysteme</b>	<b>18</b>
2.1 ABC: einfach und vielfach genutzt (★★★)	18
2.1.1 Technische Vorbereitung	19
2.1.2 Kadenz I: einzeilig	20
2.1.3 Kadenz II: mehrzeilig	21
2.1.4 Bewertung	21
2.2 MusiX <sub>TEX</sub> : komplex, kompliziert und schön (★★★★)	23
2.2.1 Technische Vorbereitung	23
2.2.2 Kadenz I: einzeilig	25
2.2.3 Kadenz II: Zweizeilig für ein Instrument	26
2.2.4 Kadenz III: Zweizeilig für mehrere Instrumente	27
2.2.5 Einschätzung	29
2.3 PMX (und M-TX): die Nicht-Vereinfachung (★★)	30
2.3.1 Das große Versprechen: Kadenz I und III	30
2.3.2 ... der Wermutstropfen ...	33
2.3.2.1 Inkompatibilitäten	33
2.3.2.2 Unzulänglichkeiten	35
2.3.3 ... und die kleine Lösung: Kadenz II	35
2.4 Lilypond: komplex, elegant und schön (★★★★★)	39
2.4.1 Technische Voraussetzungen	41
2.4.2 Kadenz I	43
2.4.3 Kadenz II	45
2.4.4 Kadenz III	47
2.4.5 Bewertung	48
2.5 Graphiken: es geht auch manuell (★★)	49
2.6 Mup: das veraltete Auslaufmodell (★)	50
2.7 T <sub>E</sub> Xmuse: die veraltete Interimslösung (★)	51
<b>3 Frontends: die (graphische) Eingabe</b>	<b>52</b>
3.1 Konverter I	53
3.2 Editoren I	55

3.3	Exkurs: MIDI unter GNU/Linux resp. Ubuntu	57
3.4	Editoren II	60
3.4.1	ABCJ (★)	60
3.4.2	Aria Meastosa (★★)	61
3.4.3	Audimus (★)	62
3.4.4	Brahms (-)	63
3.4.5	Canorus (★★★)	63
3.4.6	Denemo (★★★)	66
3.4.7	EasyABC (★★★★)	69
3.4.8	Elysium (★★★★)	71
3.4.9	Free Clef (★)	73
3.4.10	Frescobaldi (★★★★★)	73
3.4.11	Jniz (-)	75
3.4.12	Laborejo (★)	76
3.4.13	MusEdit (-)	77
3.4.14	MuseScore (★★★★★)	78
3.4.15	MuX2d (★)	80
3.4.16	NoteEdit (★)	81
3.4.17	NtEd (★★)	82
3.4.18	Ptolemaic (★)	84
3.4.19	Rosegarden (★★★)	86
3.5	Konverter II	88
3.5.1	abc2ly	88
3.5.2	abc2mtex	88
3.5.3	ly2abc	88
3.5.4	musicxml2ly	89
3.5.5	mxml2abc	89
3.5.6	xml2abc	89
3.5.7	xml2ly	89
3.5.8	xml2pmx	90
<b>4</b>	<b>Vom Frontend nach L<sup>A</sup>T<sub>E</sub>X: die Toolketten</b>	<b>91</b>
4.1	w01: EasyABC → abc2mtex → L <sup>A</sup> T <sub>E</sub> X+MusixT <sub>E</sub> X	94
4.2	w02/w03: EasyABC → abc2ly → Frescobaldi/Elysium → ...	95
4.3	w04/w05: EasyABC → musicxml2ly → Frescobaldi/Elysium → ...	96
4.4	w06/07: MuseScore → musicxml2ly → Frescobaldi / Elysium → ...	97
4.5	Vergleich	98
<b>5</b>	<b>Fazit</b>	<b>100</b>
<b>6</b>	<b>Nachrede: Das 'vergessene Kapitel'?</b>	<b>102</b>
	<b>Abkürzungen</b>	<b>103</b>
	<b>Literaturverzeichnis</b>	<b>104</b>

## 0 Der Kontext

### 0.1 Anliegen

Es gibt 'zweieinhalb' Dinge, die man erreichen möchte, wenn man am Computer Notentexte 'schreibt':

- Zum ersten wird man seine Werke hörbar machen wollen, also in Computersound umwandeln. Dies ist – bis auf eine Ausnahme – nicht unser Thema.
- Zum zweiten wird man seine Noten ausdrucken wollen, möglichst ansprechend und so leserlich, dass spielbare Notenblätter für Musiker entstehen. Dies werden wir allenfalls implizit behandeln.
- Und schließlich wird der eine oder andere Autor Notenbeispiele in seine (musikwissenschaftlichen) Texte einbinden wollen. Wie man das in welchen Grenzen mit  $\text{\LaTeX}$  umsetzt, wollen wir hier diskutieren und demonstrieren.

Natürlich ist die Integration von Noten in wissenschaftliche Texte nicht ein grundsätzlich anderes Szenario als der bloße Druck eines Notenblattes. Denn auch eine am Computer erstellte musikwissenschaftliche Arbeit zielt zuletzt auf eine ansprechende Erscheinung. Gleichwohl fordert sie Besonderes: Anders als ein Notenblatt enthält sie vornehmlich Fließtext. Außerdem nutzt sie üblicherweise keine ganzen Werke als Beleg, sondern Ausschnitte. Zu guter Letzt fordert die Musikwissenschaft gelegentlich, in diese 'Extrakte' Zeichen oder Kommentare einfügen zu können, die selbst nicht zur 'Notensprache' gehören.<sup>7</sup>

Die beste Methode, wissenschaftliche Texte zu schreiben, bietet  $\text{\LaTeX}$  – ebenso gemessen am Druckbild, wie am Grad der Automatisierung oder am Aufwand, den (eigenen) Wissenschaftsvorgaben konsistent gerecht zu werden. Und so geht es hier – vereinfacht gesagt – um  $\text{\LaTeX}$  und Musik. Ein Blick auf CTAN-Liste mit entsprechenden Tools<sup>8</sup> verwirrt eher, als dass er hilft: Womit soll man anfangen, was braucht man wozu, was davon soll man lernen, was lohnt überhaupt die Mühe? Google beantwortet die Query  $\langle latex\ music \rangle$ <sup>9</sup> mit Links auf einige kurze Artikel

---

<sup>7</sup>) Ein Beispiel dafür ist die 'Harmonieanalyse', die ja eine eigene Fachsprache bereitstellt und erwartet, dass z.B. die Abkürzungen für *Tonika*, *Subdominante* oder *Dominante* samt Subspezifikatoren in den entsprechenden Notentext eingebunden werden.

<sup>8</sup>) → <https://ctan.org/topic/music> (RDL: 2018-12-27)

<sup>9</sup>) → <https://www.google.com/search?q=latex+music>

und auf die bekannten Tools *MusiX<sub>TEX</sub>* und *LilyPond*. Zusammen führt auch das den unbedarften Anfänger nicht weiter. Es fehlt einfach eine qualifizierte Sichtung und Anleitung, die sagt und zeigt, wie man das beste Ergebnis mit geringstem Aufwand erreicht. Diese Lücke wollen wir mit einem selbstreferentiellen Dokument schließen: Es sichtet, was es gibt, zeigt, wie geht, und beleuchtet, was warum nicht funktioniert. Dabei soll alles, was es vorgeführt, anhand des eigenen  $\text{\LaTeX}$ -Quelltexts reproduzierbar sein. Und wenn es uns damit gelingt, den einen oder anderen davon abgehalten zu haben, auf's falsche Pferd zu setzen, dann wollen wir zufrieden sein.

Den wissenschaftlichen Text unseres 'Demos' erzeugen wir mit *mycsrf*<sup>10</sup>, einem Bibliographie- und Zitierstil<sup>11</sup> für  $\text{\LaTeX}$ , der sich eng an die altphilologische Arbeitsweise anlehnt, deren Notwendigkeit aber mit modernen Anforderungen an eine geisteswissenschaftliche Arbeit zu begründen ist.<sup>12</sup> Dieses 'Framework' zu kennen oder gar zu nutzen, wird jedoch nicht vorausgesetzt, weder technisch noch ideell<sup>13</sup>: was wir hier zur Integration von Musikbeispielen sagen, kann ohne Umschweife auf andere, in und mit  $\text{\LaTeX}$  verwirklichte Zitierstile übertragen werden.

Ebenso einfach darf man das, was wir hier anhand von Linuxbeispielen erläutern, auf die Windowswelt übertragen: Wer dort mit  $\text{\LaTeX}$  arbeitet, wird unsere Ergebnisse ebenfalls direkt nutzen können.

Wie man Notenbeispiele und musikwissenschaftliche Elemente in  $\text{\LaTeX}$ -Texte einbaut, hat – neben Michael Enzenhofer<sup>14</sup> – dankenswerterweise bereits Martin

---

<sup>10)</sup> vgl. *Reincke: mycsrf*, 2018, wp..

<sup>11)</sup> Genau genommen handelt es sich dabei nicht um einen eigenständigen Zitierstil, der in das  $\text{\LaTeX}$ -System zu installieren wäre, sondern um eine Konfiguration und Erweiterung von *Jurabib*, die in einen ganzes Framework zur Erarbeitung geisteswissenschaftlicher Texte eingebunden sind.

<sup>12)</sup> Zentrale Bedingung der Wissenschaftlichkeit ist die Überprüfbarkeit und Reproduzierbarkeit. Das meint bei geisteswissenschaftlichen Arbeiten, dass Argumentationen logisch aufgebaut sind und Zitate ohne großen Aufwand in den Originalen nachgesehen werden können. Warum das wie in *mycsrf* realisiert wird, habe ich in einem anderen selbstreferentiellen Text diskutiert und dokumentiert. (Vgl. dazu *Reincke: Dienst am Leser, Dienst am Scholaren*, 2018, S. 1ff)

<sup>13)</sup> Allerdings können Sie dieses Dokument unter GNU/Linux aus den Quellen heraus selbst erzeugen. Und Sie dürfen den  $\text{\LaTeX}$ -Quelltext im Rahmen der Lizenzierung (CC BY-SA 4.0) weiterverarbeiten. Sie finden die Sourcen unter `/examples/musicology.de` im *mycsrf*-Paket (Vgl. dazu *Reincke, Karsten: Quellen zu Musikwissenschaft mit LaTeX*; 2019  $\Rightarrow$  <https://github.com/kreincke/mycsrf/tree/master/examples/musicology.de> – RDL: 2019-01-02, wp.)

<sup>14)</sup> Diese Arbeit liefert jedoch 'nur' eine eher grundsätzlich Einführung in die  $\text{\LaTeX}$ -Codierung mit einigen Hinweisen zur Einbindungen von Graphiken im Allgemeinen und zur Nutzung von *LilyPond* im Besonderen (vgl. dazu *Enzenhofer, Michael: LaTeX für Musiker*; 2016  $\Rightarrow$  [http://www.michael-enzenhofer.at/LaTeXfuerMusiker/Latex4Musiker\\_1.pdf](http://www.michael-enzenhofer.at/LaTeXfuerMusiker/Latex4Musiker_1.pdf) – RDL: 2019-01-11, S. 4ff, 31ff u. 21ff). Über die Arbeitsweise von *LilyPond* und damit über die notwendigen Schritte zur Integration spricht die Arbeit kaum, ebenso wenig über Alternativen

## 0 Der Kontext

Thoma analysiert.<sup>15</sup> Erster bietet eine Einführung in L<sup>A</sup>T<sub>E</sub>X, die das Thema 'Musik' eher kursorisch behandelt, letzterer einen Überblick über L<sup>A</sup>T<sub>E</sub>X und Musik, allerdings in Form einer HTML-Webseite. Damit kann diese 'Einführung' von ihrer Natur her 'nur' beschreiben, nicht aber zeigen. Wir hingegen wollen einen Text erstellen, der aus sich heraus und an sich reproduzierbar zeigt, was geht, wie es geht und welche Klippen man wie umschiffet.

Zu guter Letzt rücken zwei Nebenthemen in den Vordergrund:

Notentexte sind von ihrem Gegenstand her zweidimensional - wenigstens, wenn sie mehrstimmige Musik notieren: Es folgt Note auf Note, um das Nacheinander auszudrücken. Und es steht Note über Note, um die Gleichzeitigkeit von Klängen zu signalisieren. Die Schriftsprache ist dagegen im Kern eindimensional: es folgt einfach Wort auf Wort. Auch L<sup>A</sup>T<sub>E</sub>X selbst - als Auszeichnungssprache für Schriftsprachen - ist eindimensional: Im Quelltext folgt Token auf Token. Von daher geht die Integration von Notenbeispielen in einen Text immer mit einer Konvertierung einher. Es gibt Auszeichnungssprachen für Noten, die diese Konvertierung in die eindimensionale 'Wort-für-Wort'-Welt erfolgreich bewältigt haben, z.B. *MusicXML*<sup>16</sup>, *MusiX<sub>T</sub>E<sub>X</sub>*<sup>17</sup>, *PMX*<sup>18</sup> oder *LilyPond*.<sup>19</sup> Trotzdem schreiben echte Musiker lieber echte Musik, also zweidimensionale Notentexte.

Damit stellen sich sofort einige Fragen:

- Mit welchen Tools, die ihren Input in einem der 'eindimensionalen' Formate speichern, kann man Notentexte – sozusagen in gewohnter Manier – graphisch, also zweidimensional erfassen?
- Gibt es – wenn man die Auszeichnungssprachen doch direkt nutzen muss – wenigstens Hilfsmittel, die das 'Komponieren' mit diesen in sich sperrigen Auszeichnungssprachen vereinfachen?
- Und wenn man schon mit mehreren Tools arbeiten muss, um zuletzt den einen gewünschten Text zu erhalten, wie sieht dann eine gute Architektur der kooperierenden Komponenten aus, wie ihre Übergabepunkte? Wie sollte man das Ganze idealerweise 'zusammenstöpseln'?

---

zu *LilyPond*. Insofern signalisiert der Titel mehr, als der Text zuletzt bietet. Nichtsdestotrotz bleibt er ein praktikabler 'Crashkurs' in Sachen L<sup>A</sup>T<sub>E</sub>X-Kodierung.

<sup>15</sup> vgl. Thoma: *How to write music with LaTeX*, 2012, wp..

<sup>16</sup> vgl. anon. [*MakeMusic*]: *MusicXML Homepage*; o.J. [2018] ⇒ <https://www.musicxml.com/> – RDL: 2018-12-21, wp..

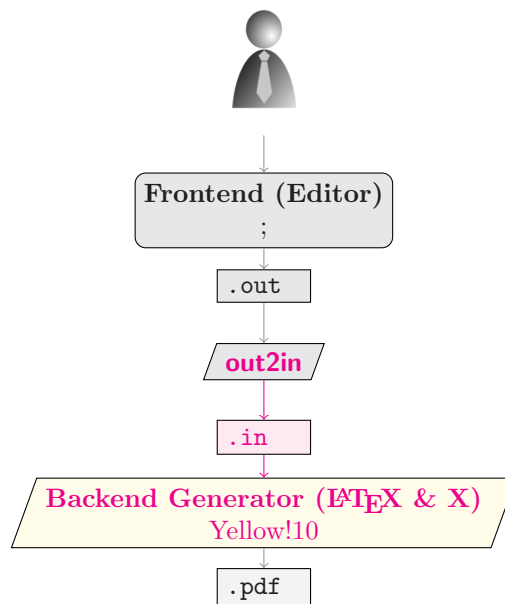
<sup>17</sup> vgl. anon. [*CTAN*]: *MusiX<sub>T</sub>E<sub>X</sub> – Sophisticated music typesetting*; o.J. [2018] ⇒ <https://ctan.org/pkg/musixtex> – RDL: 2018-12-21, wp..

<sup>18</sup> vgl. anon. [*CTAN*]: *pmx – Preprocessor for MusiX<sub>T</sub>E<sub>X</sub>*; o.J. [2018] ⇒ <https://ctan.org/pkg/pmx> – RDL: 2018-12-21, wp..

<sup>19</sup> vgl. *LilyPond Development Team*: *LilyPond. Notensatz für Jedermann*; o.J. [2018] ⇒ <http://lilypond.org/index.de.html> – RDL: 2019-01-05, wp..



Diese Punkte zielen auf mögliche Editoren, Frontends, Konverter und deren Kooperation mit den eigentlichen Notensatzsystemen, den Backends.



Die verschiedenen Aufgaben solcher Systeme auseinanderzuhalten, ist bei der Begutachtung von Lösungen essentiell. So attestiert z.B. eine ältere Sichtung von Notensatzprogrammen der „Open Source Welt“, sie habe „[...] noch immer Schwierigkeiten, ein [...] Nischenprodukt wie ein Notensatzprogramm zu entwickeln“. Und sie begründet ihr Urteil damit, dass „[...] eine neue Mark-up-Sprache zu erlernen und in dieser grafische Vorstellungskraft zu entwickeln [...] nicht jedermanns Sache (sei)“, auch wenn die Ergebnisse – wie im Falle von *Frescobaldi* und *LilyPond* – „[...] äußerst professionell anmuten (mögen)“.<sup>20</sup> Wenn man, wie hier geschehen, (graphische) Editoren und eigentliche Notensatzprogramme implizit 'gleichsetzt', dann beraubt man sich der Möglichkeit zu einem differenzierteren und pragmatischen Urteil: Wer integrierte Gesamtsysteme erwartet, bewertet komplexe Architekturen mit verteilten Teilsystem eben dieser Verteilung wegen schlechter – und erkennt, das sehr viel komplexere Systeme nach diesem Prinzip aufgebaut sind und erfolgreich betrieben werden, wie etwa *Unix* resp. *GNU/Linux*.

Wir jedenfalls werden diesen Aspekten gesondert nachgehen: Zuerst analysieren wir mögliche 'Backends' und deren Methoden, 'Noten' in L<sup>A</sup>T<sub>E</sub>X-Texte einzubinden. Sie sind insofern *Backends*, als sie textuellen Input verarbeiten, der auch über vorgelagerte, eher graphische Programme erzeugt werden könnte. Diese bilden

<sup>20)</sup> vgl. *Albrecht, Mirko*: Digitaler Notenschlüssel. Notensatzsoftware im Test; in: *Linux User*, 10 (2009) ⇒ [https://musescore.org/sites/musescore.org/files/MuseScore\\_LinuxUser\\_de\\_2009\\_10.pdf](https://musescore.org/sites/musescore.org/files/MuseScore_LinuxUser_de_2009_10.pdf) – RDL: 2019-02-22, S. 51.

## 0 Der Kontext

das 'Frontend' für den Arrangeur, Komponisten oder Musikwissenschaftler. Solche graphischen und semi-graphische Editoren und Konverter werden wir analysieren, nachdem die Backends begutachtet haben. Und ganz am Ende werden wir sehen, dass uns gerade diese Aufteilung eine besondere Flexibilität bietet, Notenbeispiele in (musik)wissenschaftliche Arbeiten einzubetten.

Allerdings werden wir uns bei unserer Betrachtung auf die Open-Source-Welt konzentrieren. Mehr oder minder kostenintensive proprietäre Programme mögen beliebt sein, notwendig sind sie nicht. Es ist auch für den Musiker besser, nicht in die 'Vendorenfalle' zu geraten und seine Arbeiten von den Produkten einer Firma abhängig zu machen. Deshalb fokussieren wir uns hier auf die Umsetzung mit freier Software.<sup>21</sup>

---

<sup>21)</sup> Wer als Musiker proprietäre Programme nutzt, macht sich erpressbar: Auf der einen Seite investiert er Zeit und Mühe in die Erstellung von Notentexten, die er auch später noch weiterverarbeiten oder wiederverwenden will. Auf der anderen Seite muss er fürchten, die hinter dem eigenen Programm stehende Firma werde beim nächsten Versionswechsel die Lizenzgebühren so stark erhöhen, dass sie das Budget des Musikwissenschaftlers sprengen. Und selbst ohne diese finale Katastrophe wird der Wechsel zu einem anderen Anbieter mit der Zeit durch die schon investierte Arbeit so aufwendig, dass man lieber bei dem bisherigen Programm bleibt, auch wenn es teurer oder schlechter ist als die Alternativen. Damit ist man in die *Vendorenfalle* getappt. Mit freier Software kann das nicht geschehen, weil es zum Wesen freier Software gehört, dass man ohne pekuniären Aufwand an den Quellcode und an die Nutzungsrechte der ablauffähigen Programme herankommt - auch an die neueren und besseren Versionen. (vgl. dazu *Free Software Foundation: Freie Software. Was ist das? Übersetzung aus dem Amerikanischen*; o.J [2016] ⇒ <https://www.gnu.org/philosophy/free-sw.de.html> – RDL: 2018-12-21, wp..) Für Musiker hat sich die Lage mit der Etablierung von *MusicXML* allerdings etwas entspannt: Durch die Standardisierung des Dateiformates wird der Wechsel zu einem anderen, besseren Programme erleichtert. Hier muss man 'nur' noch fürchten, dass das bisher die jeweiligen Programme vorab (noch) nicht standardisierte Features benutzen oder den Standard nicht ganz konform implementiert haben. Allerdings sind solche 'Eigenarten' ein beliebtes Mittel, den zahlenden Kunden zuletzt doch wieder an sich zu binden und ihm den Wechsel zur Alternative zu erschweren. (Zur Lizenzierung von *MusicXML* vgl. auch *anon. [Wikipedia]: MusicXML*; o.J. [2018] ⇒ <https://de.wikipedia.org/wiki/MusicXML> – RDL: 2018-12-21, wp.) Dieser Kontext gibt uns auch die Gelegenheit, auf die Lizenzproblematik einzugehen: Wir werden bei den Tools jeweils kurz nachweisen, dass es sich tatsächlich um freie Software bzw. Open Source Software handelt. Dass dem so ist, ist uns wichtig. Denn aus einem solchen Nachweis ergibt sich, dass man die Software ohne Einschränkungen und unentgeltlich nutzen darf. Allerdings folgt Open Source Software dem Prinzip 'Paying by Doing'. Anstatt – wie sonst üblich – die Nutzungsrechte zu kaufen, 'erwirbt' man diese, indem man das aktiv tut, was die Lizenzen dem Nutzer auftragen. Zu wissen, was das ist, ist kein Hexenwerk, kann in Einzelfällen aber 'tricky' werden. Diese rechtlichen Rahmenbedingungen sind nicht unser Thema. Wir gehen deshalb davon aus, dass Sie sich selbst darum kümmern werden, wenn es angesagt ist. Unser Erfahrung nach gibt es dafür eine gute Daumenregel: Sie dürfen die Frage, ob und wie sie die Software lizenzkonform nutzen, solange aufschieben, bis Sie sich entscheiden, die Software an andere weiterzugeben. Oder anders gesagt: Solange Sie selbst diese Software 'nur' von irgendwoher downgeloaded oder installiert haben und anwenden, solange dürfen Sie die Beachtung der Compliance auch 'recht gefahrlos' auf später verschieben.

Dazu noch einige abgrenzende Worte:

- Damit Sie einen guten Gewinn von der Lektüre haben, sollten Sie mit der Nutzung von L<sup>A</sup>T<sub>E</sub>X unter Ihrem Betriebssystem vertraut sein. Insbesondere die Erweiterung Ihres T<sub>E</sub>X-Systems und die Installation von zusätzlichen CTAN-Paketen legen wir kommentarlos in Ihre Hände<sup>22</sup>.
- Ferner setzen wir voraus, dass Sie die Tools und Programme, die wir erwähnen, unter Ihrem Betriebssystem installieren können.
- Außerdem werden wir Sie nicht anleiten, wie man mit einem der begutachteten System Notentexte schreibt. Dazu gibt es bessere und genauere Handbücher, auf die wir gerne und dankbar verweisen. Was wir Ihnen jedoch zeigen werden, ist, wie man die mit den Tools erarbeitete Notentexte - wenn überhaupt möglich - in einen L<sup>A</sup>T<sub>E</sub>X-basierten wissenschaftlichen Text integriert.
- Und schließlich gehen wir ohne große Prüfung davon aus, dass die Programme und Tools, die behaupten, für verschiedene Betriebssysteme zu existieren, im wesentlichen gleich funktionieren. In der Open-Source-Welt ist dem üblicherweise so.

Und damit wollen wir es gut sein lassen mit dem Erwartungsmanagement.

## 0.2 Kriterien

Wer den besten Weg sucht, braucht Maßstäbe. Deshalb werden wir den Notensatzsystemen drei Referenzkadenzen vorlegen und erwarten, dass sie diese optisch ansprechend erfassen und wiedergeben können, und zwar einschließlich der zugehörigen Harmonieanalyse:

Als erste und einfachste Aufgabe beziehen wir uns auf ein Beispiel aus der Harmonielehre von Grabner<sup>23</sup>:

<sup>22)</sup> Es gibt eine Fülle guter, auch auf die Praxis ausgerichteter L<sup>A</sup>T<sub>E</sub>X-Einführungen. (etwa [Schlosser, Joachim: Wissenschaftliche Arbeiten schreiben mit L<sup>A</sup>T<sub>E</sub>X. Leitfaden für Einsteiger. 6., überarb. Aufl.; Frechen: mitp Verlag, 2016: ISBN 978-3-95845-289-3, S. 7ff](#)). Die manuelle Installation eines CTAN-Paketes ist – wenn überhaupt nötig – so schwierig nicht: Man lädt sich das Paket herunter und entpackt es in einem Ordner. Dann kopiert man diesen Ordner – nötigenfalls mit Root-Rechten – in die L<sup>A</sup>T<sub>E</sub>X-Distribution, z.B. unter `/usr/share/texmf/tex/latex/` (wobei `/usr/share/texmf` distributionsabhängig ist). Und schließlich setzt man an der Konsole / im Terminal noch einmal das Kommando `sudo texhash` ab, um das neue Paket bekannt zu machen. Wenn Sie eine gängige Distribution benutzen, sollte der Rückgriff auf solch eine Detailarbeit aber gar nicht nötig sein.

<sup>23)</sup> vgl. [Grabner, Hermann: Allgemeine Musiklehre; mit einem Nachtrag v. Diether de la Motte; 11. Auflage. Kassel, Basel \[... u.a.O.\]: Bärenreiter Verlag, 1974: ISBN 3-7618-0061-4, S. 107.](#)

## 0 Der Kontext

I      IV      V      I      IV      I

CADENZA-I: REFERENZ

Die zweite Referenzkadenz soll die Erfassung harmonisch komplexer Zusammenhänge abfordern:

CADENZA-II: REFERENZ

Und die dritte Referenzkadenz soll das in einen schwierigeren rhythmisch-metrischen Kontext einbetten:

CADENZA-III: REFERENZ

Später werden wir dann – wie angekündigt – nach guten Frontends für die  $\text{\LaTeX}$ -kompatiblen Notensatzsysteme suchen, nach Editoren. Von ihnen werden wir verlangen dass sie die Referenzkadenz II erfassen und adäquat exportieren. Wie wir die Adäquatheit begutachten, werden wir gesondert erläutern.<sup>24</sup> Und zuletzt werden wir ganze Toolketten aus Frontends, Konvertern und Backends evaluieren. Bei diesen werden wir überprüfen, ob sie die Referenzkadenz III bruchlos über alle Komponenten hinweg transportieren und adäquat darstellen.

<sup>24)</sup> → S. 57

# 1 Leichtgewichtige Teillösungen

Selbstverständlich wird sich zuletzt herausstellen, dass adäquate Systeme zur Erfassung von Musik eine gewisse Komplexität mitbringen. Das liegt in der Natur der Sache: Musik ist zweidimensional.<sup>25</sup> Das hat einige Autoren nicht entmutigt, wenigstens für Teilaufgaben einfachere Lösungen zu entwickeln, die im freien Textsatzsystem  $\text{\LaTeX}$ <sup>26</sup> genutzt werden können. Diesen werden wir zuerst nachgehen. Behalten wir aber im Hinterkopf: *Komplexität ist wie Wasser: man kann es nicht komprimieren.*<sup>27</sup>

---

<sup>25)</sup> Bei der Notation von Musik hatten wir gesagt, dass sie zweidimensional oder eindimensional codiert sein könne. Als Abfolge(sic!) von Klängen(sic!) ist sie selbst aber immer (mindestens) zweidimensional.

<sup>26)</sup> Bzgl. der Zusammenhänge von  $\text{\TeX}$ ,  $\text{\LaTeX}$ ,  $\text{\pdfLaTeX}$  und  $\text{\LuaLaTeX}$  vgl. etwa [Voß, Herbert: Die wissenschaftliche Arbeit mit \LaTeX; unter Verwendung von LuaTeX, KOMA-Script und Biber/BibLaTeX](#); Berlin: Lehmanns Media, 2018 (= dante): ISBN 978-3-86541-946-0, S. 7ff oder [Voß: Einführung in \LaTeX](#), S. 8ff. Daraus ergibt sich mittelbar auch der Open-Source-Status von  $\text{\LaTeX}$ :  $\text{\TeX}$  und  $\text{\LaTeX}$  sind zunächst einmal 'nur' eine Metasprache. Um aus Dateien, die in dieser Auszeichnungssprache formuliert sind, typographisch gestaltete *DVI*- oder *PDF*-Dateien zu erzeugen, bedarf es bestimmter Programme, nämlich `pdflatex`, `lualatex` etc. Außerdem hat sich um  $\text{\LaTeX}$  herum ein Biotop von Paketen entwickelt. Das sind fertig vorbereitete  $\text{\TeX}$ - oder  $\text{\LaTeX}$ -Dateien, die man in seine eigenen Dokumente einbindet und also mitnutzt. Solche Pakete stehen – mit den Kernprogrammen gebündelt – als komplette  $\text{\TeX}$ -Distributionen bereit. Für den europäischen Raum existieren z.Zt. zwei wesentliche Distributionen, *TeXlive* und *MiKTeX*. Diese findet man im *Comprehensive TeX Archive Network* ( $\rightarrow$  <https://ctan.org/>). Der unbedarfte Anwender tut aber gut daran, sich vorbereitete Installationspakete von sekundären Distributoren zu holen. *Linux*-Distributionen liefern diese i.d.R. mit. Ob man sein  $\text{\LaTeX}$ -System lizenzkonform nutzt, entscheidet also die Lizenzierung der Kernprogramme und die Lizenzen der eingebundenen Pakete. Diese müssen nicht unter derselben Lizenz veröffentlicht sein. Glücklicherweise werden beide – die Kernprogramme und die Erweiterungen – als *CTAN*-Pakete gehostet und unter den Bedingungen verschiedener FOSS-Lizenzen zum Download angeboten. So ist etwa  $\text{\pdfLaTeX}$  unter der GPL lizenziert ( $\rightarrow$  <https://ctan.org/pkg/pdftex>). Ganz allgemein gesagt, darf man alles, was zu  $\text{\LaTeX}$  gehört, als freie Software nutzen. Wir werden bei den Paketen jedoch kurz angeben, unter welcher Lizenz sie genau stehen.

<sup>27)</sup> Dieses wunderbare Bonmot stammt nicht von mir. Und bedauerlicherweise weiß ich auch nicht (mehr) von wem. Ich meine es etwa Anfang der 2010er Jahre in einer Keynote bei der Deutschen Telekom in Darmstadt gehört zu haben. Ich verbeuge mich also dankbar vor dem mir leider unbekannten Autor.

## 1.1 Sonderzeichen: der Standard (★★★)

Neben Zeichen für die Schriftsprache als solche bot  $\text{\LaTeX}$  - der Mathematik sei Dank - immer schon auch graphische Symbole an, die wie Schriftzeichen in einer Zeile platziert werden können.<sup>28</sup> Sie werden im Mathematikmodus notiert, haben also die Form  $\text{\$}\backslash\text{ZEICHENNAME}\text{\$}$ . Und in diesem Fundus von Sonderzeichen gibt es auch die drei Symbole  $\sharp$  ( $= \text{\$}\backslash\text{sharp}\text{\$}$ ),  $\flat$  ( $= \text{\$}\backslash\text{flat}\text{\$}$ ) und  $\natural$  ( $= \text{\$}\backslash\text{natural}\text{\$}$ ).

Diese Zeichen können ohne Zusatzpaket in einem  $\text{\LaTeX}$ -Quelltext genutzt werden. Unnötig zu erwähnen, dass sie – für sich genommen – nicht ausreichen, Musik textuell zu erfassen. Dennoch wird man in der Kombination mit anderen Ansätzen gelegentlich auf sie zurückkommen wollen.<sup>29</sup>

## 1.2 Andere Sonderzeichen: wasysym (★★★)

Gleiches gilt für das  $\text{\LaTeX}$ -Zusatzpaket *wasysym*<sup>30</sup>: Es erweitert den Vorrat an musikalischen Zeichen um Notensymbole. Wie in  $\text{\LaTeX}$  üblich, muss das Paket über ein Kommando in die Präambel eingebunden werden ( $\backslash\text{usepackage}\{wasysym\}$ ), damit man danach auf die Zeichen  $\text{\musicalnote}$  ( $= \backslash\text{eighthnote}$ ),  $\text{\musicalnote}$  ( $= \backslash\text{quarternote}$ ),  $\text{\musicalnote}$  ( $= \backslash\text{halfnote}$ ),  $\text{\musicalnote}$  ( $= \backslash\text{fullnote}$ ), und  $\text{\musicalnote}$  ( $= \backslash\text{twonotes}$ ) zugreifen kann.<sup>31</sup>

<sup>28)</sup> vgl. *Mittelbach u. Goossens: Der  $\text{\LaTeX}$ -Begleiter*, 2005, S. 543ff et passim.

<sup>29)</sup> In diesem Zusammenhang wäre auch zu erwähnen, dass Unicode und seine Encodierung UTF-8 von sich aus einen Bereich mit 'Musikzeichen' definiert hat, nämlich die Zeichen zwischen U+1D100 und U+1D1FF (Vgl. dazu *Köllerwirth, Micha: UTF-8-Codetabelle mit Unicode-Zeichen*; o.J. [2015]  $\Rightarrow$  <https://www.utf8-zeichentabelle.de/unicode-utf8-table.pl> – RDL: 2019-01-14, wp.). Wenigstens unter (pdf) $\text{\LaTeX}$  bedarf Nutzung dieser Zeichen – wenn überhaupt möglich – zusätzlicher Maßnahmen. (Zum Zusammenhang zwischen Unicode und UTF( vgl. *Kuhn, Markus: UTF-8 and Unicode FAQ for Unix/Linux*; o.J. [2009]  $\Rightarrow$  <https://www.cl.cam.ac.uk/~mgk25/unicode.html> – RDL: 2019-01-22, wp.)

<sup>30)</sup> vgl. *anon. [CTAN]: wasysym -  $\text{\LaTeX}$  support file to use the WASY2 fonts*; o.J. [2003]  $\Rightarrow$  <https://ctan.org/pkg/wasysym> – RDL: 2018-12-21, wp. Die Paketbeschreibung gibt an, dass *wasysym* unter der *LaTeX Project Public License* veröffentlicht wird. Das ist eine von der OSI anerkannte Open-Source-Lizenz ( $\rightarrow$  <https://opensource.org/licenses/LPPL-1.3c>).

<sup>31)</sup> vgl. *Kielhorn, Axel: The wasysym macro package for  $\text{\LaTeX}2\epsilon$* ; 2003  $\Rightarrow$  <http://texdoc.net/texmf-dist/doc/latex/wasysym/wasysym.pdf> – RDL: 2018-12-22, S. 2.

## 1.3 Noch mehr Sonderzeichen: *musicography* (★★)

Das Zusatzpaket *musicography*<sup>32</sup> vereinigt und erweitert die bisher erwähnten Möglichkeiten. Außerdem sagen manche Fürsprecher, dass es – im Gegensatz zu anderen Paketen – auch mit *pdflatex* druckfähige PDF-Dateien erzeuge.<sup>33</sup> Selbstverständlich muss dieses Paket seiner L<sup>A</sup>T<sub>E</sub>X-Natur nach ebenfalls in die Präambel des L<sup>A</sup>T<sub>E</sub>X-Dokumentes eingebunden werden (`\usepackage{musicography}`), um entsprechende Befehle nutzen zu können.

Anschließend erlaubt das Paket, Vorzeichen { ♭ (= `\musFlat`), # (= `\musSharp`), ♮ (= `\musNatural`), ♯♯ (= `\musDoubleFlat`), × (= `\musDoubleSharp`) }, Notensymbole { ○ (= `\musWhole`), ♩ (= `\musHalf`), ♪ (= `\musQuarter`), ♫ (= `\musEighth`), ♬ (= `\musSixteenth`), ♪. (= `\musHalfDotted`), ♪. (= `\musQuarterDotted`) ... } und Metren wie  $\text{♩}$  (= `\meterCutC`) in den Fließtext einzuarbeiten.

Gleichwohl gibt es dabei einige 'Ungereimtheiten', die den Gebrauch erschweren:

- Zum ersten muss man das Paket *MusiX<sub>TEX</sub>*, sofern man es zusammen mit *musicology* verwenden will, nach *musicology* in die Präambel einbinden. Andernfalls 'meckert' *musicology*, dass der Befehl `meterC` schon definiert sei. Bindet man *MusiX<sub>TEX</sub>* tatsächlich nach *musicology* ein, kann *MusiX<sub>TEX</sub>* den von *musicology* eingeführten Befehl offensichtlich problemlos redefinieren. Leider passen die Symbole dann – wie man hier sehen kann – von der Größe

her nicht mehr zu einander: *musicology* →  $\text{♩}$ :  $\text{♩}$  ← *MusiX<sub>TEX</sub>*.

- Außerdem sind die Symbole für die Achtel- (♪) und die Sechzehntelnoten (♬) aus Musikersicht unbrauchbar. Solche Zeichen gibt es in der Musik(wissenschaft) nicht.

Wenn man dieses Paket verwenden will, ist also eine gewisse Vorsicht angesagt.

<sup>32)</sup> vgl. anon. [CTAN]: *musicography* – Accessing symbols for music writing with pdfL<sup>A</sup>T<sub>E</sub>X; o.J. [2018] ⇒ <https://ctan.org/pkg/musicography> – RDL: 2018-12-23, wp.. Die Paketbeschreibung gibt an, dass *musicography* unter der *LaTeX Project Public License* veröffentlicht wird. Das ist eine von der OSI anerkannte Open-Source-Lizenz (⇒ <https://opensource.org/licenses/LPPL-1.3c>).

<sup>33)</sup> Vgl. dazu etwa Cashner, Andrew A.: *The musicography Package: Symbols for Music Writing with pdf<sub>late</sub>x*; 2018 ⇒ <http://ctan.space-pro.be/tex-archive/macros/latex/contrib/musicography/musicography.pdf> – RDL: 2018-12-23, S. 1. Persönlich sind uns bei der PDF-Generierung solche Irritationen mit anderen Paketen nicht begegnet. *mycsrf* nutzt auch *pdflatex*. Und unter Ubuntu 18.04 werden dabei alle Fonts eingebunden und können alles ausdrucken, was auch am Bildschirm sichtbar ist: so auch bei dem Beispiel *musicology.de*.



## 1.4 Ganz viele Sonderzeichen: harmony (★★★★★)

Das Paket *harmony*<sup>34</sup> ist ein Highlight, bietet es für den Gebrauch innerhalb einer Textzeile doch ein besonders ausgefeiltes, einfach anzuwendendes System von Musikzeichen an. Als L<sup>A</sup>T<sub>E</sub>X-Paket muss es natürlich ebenfalls zuerst mittels eines Befehls in die Präambel eingebunden werden (`\usepackage{harmony}`), bevor es Zeichen für die beiden Bereiche *Rhythmik* und *Harmonieanalyse* bereitstellt<sup>35</sup>:

### 1.4.1 Rhythmik

Zunächst enthält es gute aufgelöste Kodierungen für Taktarten  $\{ \frac{3}{4} (= \text{\Takt{3}{4}}), \frac{4}{4} (= \text{\Takt{4}{4}}), \dots, \text{\C} (= \text{\Takt{c}{0}}), \text{\C} (= \text{\Takt{c}{1}}) \}$ . Dann offeriert es nicht nur einfache Notenlängen  $\{ \text{\o} (= \text{\Ganz}), \text{\d} (= \text{\Halb}), \text{\l} (= \text{\Vier}), \text{\a} (= \text{\Acht}), \text{\s} (= \text{\Sech}), \text{\z} (= \text{\Zwdr}), \}$  – die sogar punktiert werden können  $\{ \text{\d} (= \text{\Halb\Pu}), \text{\l} (= \text{\Vier\Pu}), \dots \}$  –, sondern es stellt auch Symbole für Noten mit waagerechtem Balken zu Verfügung  $\{ \text{\a} (= \text{\AchtBL}), \text{\s} (= \text{\SechBL}), \text{\a} (= \text{\Vier \AchtBL}), \text{\s} (= \text{\Vier \SechBL}), \dots \}$ , sodass die einzelnen Elemente – als Syntagmen aneinandergereiht – zu ganzen Rhythmusketten kombiniert werden können: `\C \a \a \a \a | \a \a \a \a \a |` – eine wirklich ausgefuchste Lösung.

### 1.4.2 Harmonik

Einen ähnlich geschickten Ansatz bietet das Paket *harmony* da, wo es Symbole erzeugt, die – der Funktions- und Stufentheorie entsprechend – harmonische Zusammenhänge repräsentieren.

Den Kern bildet die allgemeine Tupelkonstruktion `\HH.X.u.v.w.z.`, mit der einfache und komplexe Aspekte dargestellt werden können<sup>36</sup>: Was zwischen den ersten beiden Punkten erscheint (X), erscheint als Hauptzeichen; dann folgt das, was darunter erscheinen soll (u) und schließlich das, was rechts oben daneben angezeigt werden soll, und zwar von oben nach unten (v w z):

$$\text{\HH.X.u.v.w.z.}$$

Der Witz ist nun, dass man nahezu beliebige Latexkonstrukte zwischen den Punkten einsetzen kann. Und wenn man nichts zwischen den Punkten einträgt, bleibt der Platz

<sup>34)</sup> vgl. *anon.* [CTAN]: *harmony – Typeset harmony symbols, etc., for musicology*; o.J. [2005]  $\Rightarrow$  <https://ctan.org/pkg/harmony> – RDL: 2018-12-21, wp.. Die Paketbeschreibung gibt an, dass *harmony* unter der *LaTeX Project Public License* veröffentlicht wird. Das ist eine von der OSI anerkannte Open-Source-Lizenz ( $\rightarrow$  <https://opensource.org/licenses/LPPL-1.3c>).

<sup>35)</sup> Für einen vollen Überblick über den Zeichenvorrat und die Kombinationsmöglichkeiten vgl. *Wegner, Dagny u. Arnim Wegner*: *harmony.sty*; 2007  $\Rightarrow$  <ftp://ftp.dante.de/pub/tex/macros/latex/contrib/harmony/harmony.pdf> – RDL: 2018-12-21, S. 4ff.

<sup>36)</sup> vgl. dazu ds., a.a.O., S. 2ff.



## 1.4 Ganz viele Sonderzeichen: *harmony* (★★★★★)

leer. So können die einfachen Symbole der Funktionstheorie  $\{ \mathsf{T} \text{ (= \texttt{\textbackslash HH.T....})}, \mathsf{Tp} \text{ (= \texttt{\textbackslash HH.Tp....})}, \mathsf{S} \text{ (= \texttt{\textbackslash HH.S....})}, \mathsf{D} \text{ (= \texttt{\textbackslash HH.D....})} \}$  mit derselben Technik erzeugt werden, wie die komplexeren  $\{ \mathsf{D}_3^9 \text{ (= \texttt{\textbackslash HH.D.3.9.7..})}, \mathsf{T}^{9b \rightarrow 8} \text{ (= \texttt{\textbackslash HH.T..9\$flat\rightarrow\$8.7..})} \}$ . Die speziellen Zeichen der Funktionstheorie, die nicht so einfach aus dem Fundus normaler Fonts gebildet werden können, stellt *harmony* gesondert zur Verfügung:  $\{ \mathsf{D} \text{ (= \texttt{\textbackslash Dohne})}, \mathsf{D} \text{ (= \texttt{\textbackslash DD})}, \mathsf{S} \text{ (= \texttt{\textbackslash DS})} \}$ . Selbstverständlich können diese ebenfalls in das allgemeine Tupelkonstrukt eingebettet werden<sup>37</sup>:

$$\mathsf{D} \text{ (= \texttt{\textbackslash DD.5\textbackslash VM.7...} \rightarrow \mathsf{D}^7_{5>}}$$

Zudem erlaubt es die Flexibilität des Grundkonstruktes (in gewissen Grenzen), Symbole für die Stufentheorie und den Generalbass zu erzeugen:

$$\mathsf{I}^{(5)}_{(3)} \quad \mathsf{III}^{(6)}_{(3)} \quad \mathsf{V}^6_4 \quad \mathsf{I}^{(5)}_{(3)} \quad \mathsf{I}^{(5)}_{3b} \quad \mathsf{III}^{(6)}_{3b} \quad \mathsf{III}^{(6)}_{(3)} \quad \mathsf{III}^{(6)}_{3b}$$

An diesem Beispiel erkennt man außerdem, dass sich die *harmony*-Konstrukte (hier `\texttttt` und `\flat`) gut mit anderen L<sup>A</sup>T<sub>E</sub>X-Elementen kombinieren lassen:

```
\begin{center}
\HH.I..\texttttt{(5)}..\texttttt{(3)}.. \
\HH.III..\texttttt{ 6 }..\texttttt{(3)}.. \
\HH.V..\texttttt{ 6 }..\texttttt{ 4 }.. \
\HH.I..\texttttt{(5)}..\texttttt{(3)}.. \
\HH.I..\texttttt{(5)}..\texttttt{ 3\$flat$ }.. \
\HH.III..\texttttt{ 6 }..\texttttt{ 3\$flat$ }.. \
\HH.III..\texttttt{ 6 }..\texttttt{ 5 }..\texttttt{(3)}. \
\HH.III..\texttttt{ 6\$flat$ }..\texttttt{ 3\$flat$ }.. \
\end{center}
```

Später werden wir zeigen, dass sich die *harmony*-Elemente ihrerseits auch gut in MusiX<sub>T</sub>E<sub>X</sub>-Syntagmen einbetten lassen. Insofern haben die Programmierer von *harmony* der Community ein mächtiges Werkzeug zur Verfügung gestellt.<sup>38</sup>

<sup>37)</sup> Vgl. dazu *Wegner u. Wegner: harmony, 2007, S. 6.*

<sup>38)</sup> Trotzdem wollen auch wir wenigstens darauf hinweisen, dass *harmony*-Konstrukte auf die Einbettung in einen Fließtext mit 12 Pt. ausgelegt sind. Bei kleineren Größen von 11PT abwärts werden die Zeilenabstände gedehnt, es entsteht ein leicht unruhigeres Druckbild. (Vgl. dazu *ds., a.a.O., S. 2.*) Das zu beklagen, wäre allerdings ein Jammern auf sehr hohem Niveau.

## 2 Backends: Komplexe Notationssysteme

### 2.1 ABC: einfach und vielfach genutzt (★★★)

Das *ABC*-Notationssystem ist eine ASCII basierte Methode zum Dokumentieren von Musik. Initial entworfen wurde sie, um die Lieder zu erfassen.<sup>39</sup> So bezeichnet sich dieses Verfahren – weil schon lange gepflegt und vielfach genutzt – denn auch als *das* textbasierte Musiknotationssystem und als *den* Defacto-Standard für Volksmusik.<sup>40</sup> Für dieses Verfahren gibt es eine Fülle von Konvertern, die ganz verschiedene Nutzungsszenarien bedienen.<sup>41</sup> Und eines der Verwertungsszenarios ist eben die Einbettung von *ABC*-Noten in  $\text{\LaTeX}$ -Text, wie es von dem *abc*- $\text{\LaTeX}$ -Paket ermöglicht wird.<sup>42</sup>

Die Art der Notation – die natürlich für alle Verwertungsszenarios im Wesentlichen gleich ist – kann per Online-Tutorials gelernt werden, etwa dem von Steve Mansfield<sup>43</sup> oder dem von John Chambers<sup>44</sup>; die Besonderheiten im Rahmen der  $\text{\LaTeX}$ -Nutzung werden im Pakethandbuch erläutert.<sup>45</sup>

Auch wenn die *ABC*-Notation zunächst 'nur' Lieder erfassen sollte und Mehrstimmigkeit nicht unbedingt das Anliegen der initialen Programmierer gewesen ist, gibt es mit *ABC-PLUS*<sup>46</sup> inzwischen auch für mehrsystemige Partituren eine Lösung, die gut dokumentiert ist<sup>47</sup> und die Integration in  $\text{\LaTeX}$ -Texte ermöglicht.<sup>48</sup>

<sup>39)</sup> vgl. *Chambers, John*: ABC Music Notation; o.J. [2009]  $\Rightarrow$  <http://trillian.mit.edu/~jc/music/abc/doc/ABCTutorial.html> – RDL: 2018-12-27, Subpage 'Intro'.

<sup>40)</sup> vgl. *Walshaw, Chris*: abc notation home page; o.J. [2018]  $\Rightarrow$  <http://abcnotation.com/> – RDL: 2018-12-25, wp.. Im Original: „the text-based music notation system and the de facto standard for folk and traditional music“.

<sup>41)</sup> vgl. *Walshaw, Chris*: abc software packages; o.J. [2018]  $\Rightarrow$  <http://abcnotation.com/software> – RDL: 2018-12-25, wp..

<sup>42)</sup> vgl. *anon.* [CTAN]: abc – Support ABC music notation in  $\text{\LaTeX}$ ; o.J. [2016]  $\Rightarrow$  <https://ctan.org/pkg/abc> – RDL: 2018-12-25, wp..

<sup>43)</sup> vgl. *Mansfield, Steve*: How to interpret abc music notation; 2016  $\Rightarrow$  [http://www.lesession.co.uk/abc/abc\\_notation.htm](http://www.lesession.co.uk/abc/abc_notation.htm) – RDL: 2018-12-25, wp..

<sup>44)</sup> vgl. *Chambers*: abc music notation, 2018, wp..

<sup>45)</sup> vgl. *Gregorio, Enrico*: The abc package; 2016  $\Rightarrow$  <https://ctan.org/tex-archive/macros/latex/contrib/abc/abc.pdf> – RDL: 2018-12-25, wp..

<sup>46)</sup> vgl. *Gonzato, Guido*: The ABC Plus Project; o.J. [2018]  $\Rightarrow$  <http://abcplus.sourceforge.net/> – RDL: 2018-12-28, wp.. Mittlerweile scheint der Name von *ABC-PLUS* in *ABC-2* abgeändert worden zu sein.

<sup>47)</sup> vgl. *Gonzato, Guido*: Making Music with Abc 2; A practical guide to the Abc notation; [formerly: 'Making Music with Abc Plus']; 2018  $\Rightarrow$  [http://abcplus.sourceforge.net/abcplus\\_en.html](http://abcplus.sourceforge.net/abcplus_en.html) – RDL: 2018-12-28, S. XVff.

<sup>48)</sup> vgl. ds., a.a.O., S. 134.

### 2.1.1 Technische Vorbereitung

Trotz aller Einfachheit bedarf es zur erfolgreichen Nutzung der ABC-Notation in und mit einer L<sup>A</sup>T<sub>E</sub>X-Datei einiger systemischen Vorbereitungen:

- Zunächst muss – ganz unabhängig von L<sup>A</sup>T<sub>E</sub>X – das Tool *abcm2ps* installiert werden.<sup>49</sup> Es wird genutzt, um die im ersten Durchgang des PDF-Erzeugungsprozess aus der L<sup>A</sup>T<sub>E</sub>X-Datei extrahierten ABC-Daten in ein Postscriptbild umzuwandeln, das dann bei der nächsten Runde – automatisiert – anstelle des ABC-Codes in die L<sup>A</sup>T<sub>E</sub>X-Datei eingesetzt wird.
- Sofern es die T<sub>E</sub>X-Distribution nicht eh schon mitliefert, ist es nötig, das *abc*-L<sup>A</sup>T<sub>E</sub>X-Paket<sup>50</sup> manuell zu installieren.<sup>51</sup>
- Danach muss das Paket – wie bei L<sup>A</sup>T<sub>E</sub>X üblich – mit einem Kommando (`\usepackage{abc}`) in die Präambel eingebunden werden.<sup>52</sup>
- Schließlich bedürfen die L<sup>A</sup>T<sub>E</sub>X-Durchgänge zur Erzeugung des PDFs einer Modifikation: Um das Bild aus dem extrahierten Code generieren zu können, muss das generierende Programm *pdflatex* hilfsweise auch externe Programme aufrufen dürfen<sup>53</sup>, also mit der Option `--shell-escape` gestartet wird. Die entsprechende Passage aus einem Makefile<sup>54</sup> könnte so aussehen.<sup>55</sup>

`.tex.pdf:`

```
# (A) create a tmpdir for storing the abc help files
mkdir -p abc
```

<sup>49)</sup> Unter Ubuntu: `sudo apt-get install abcm2ps`

<sup>50)</sup> vgl. *anon. [CTAN]: abc in L<sup>A</sup>T<sub>E</sub>X, 2016, wp.*

<sup>51)</sup> Bei Ubuntu 18.04 ist es im Paket *texlive-music* enthalten.

<sup>52)</sup> Der Konvertierung in Graphiken wegen muss außerdem das Paket *graphicx* mittels `\usepackage{graphicx,color}` aktiviert sein.

<sup>53)</sup> Dies ist normalerweise aus Sicherheitsgründen untersagt, denn sonst könnten die Computer, die die Dokumente erzeugen, aus 'Dokumenten' heraus manipuliert werden.

<sup>54)</sup> Musikwissenschaftler, die nicht ganz so software-affin sind, mag der Begriff 'Makefile' abschrecken. Tatsächlich verbirgt sich dahinter etwas sehr Einfaches: Man kommt bei der Nutzung eines Computers oft an den Punkt, wo man das, was man immer wieder tun muss, gerne automatisieren, also mit dem Aufruf eines Befehls abgearbeitet sehen möchte. Unter Windows kann man dafür Batch-Dateien erstellen, unter Linux Shell-Skripte. Oder man nutzt eben Makefiles.

<sup>55)</sup> Wissenschaftliche L<sup>A</sup>T<sub>E</sub>X-Dokumente mit Fußnoten und bibliographischen Angaben werden eh schon über mehrere L<sup>A</sup>T<sub>E</sub>X-Durchgänge hinweg erzeugt: Jeder einzelne Durchgang lagert später benötigte Angaben in Hilfsdateien aus und liest diejenigen, die er selbst verwenden will, aus Dateien ein, die vorhergehende erzeugt haben. Verwendet man das *abc*-L<sup>A</sup>T<sub>E</sub>X-Paket generiert der erste Durchgang also nicht mehr nur die bibliographischen Hilfsdateien, sondern auch die korrespondierenden ABC-Dateien, für die er anschließend das externe Tool *abcm2ps* aufruft. Dies konvertiert die extrahierten Vorlagen dann seinerseits in Postscript- und PDF-Dateien. Und der nächste L<sup>A</sup>T<sub>E</sub>X-Durchgang integriert dann diese erzeugten Dateien in das Gesamtwerk. Wie man das schlüssig automatisiert, zeigt das Makefile dieses Tutorials (→ <http://github.com/kreincke/mycsrf/tree/master/examples/musicology.de>)

## 2 Backends: Komplexe Notationssysteme

```
# (B) the first latex pass which extracts also the abc-files
@ pdflatex $<
# (C) create the help files for the literature
@ bibtex 'basename $< .tex'
@ makeindex 'basename $< .tex'.nlo -s cfg/nomencl.list \\
  -o 'basename $< .tex'.nls
# (D) generate the abc pictures by calling the external program abcm2ps
@ pdflatex --shell-escape $<
# (E) mv the results into the tmpdir to enable the next pass to find them
mv *.ps abc/
# (F) create the final document
@ pdflatex --shell-escape $<
@ pdflatex --shell-escape $<
# (G) mv the recreated results also into the tmpdir for a better cleansing
mv *.ps abc/
# (H) cleansing the environment
rm -rf abc
```

Aus dem technischen Ablauf ergibt sich, dass das Open-Source-Tool *abcm2ps*<sup>56</sup> hier eine zentrale Rolle spielt. Nach seiner Einbettung in die gängige L<sup>A</sup>T<sub>E</sub>X-Prozedur kann und darf man also beliebig viele *ABC*-Sektionen in seinem L<sup>A</sup>T<sub>E</sub>X-Quellcode erzeugen und mit *ABC*-Notationen füllen, wobei jede Sektion mit `\begin{abc}` eröffnet und mit `\end{abc}` beendet wird:

### 2.1.2 Kadenz I: einzeilig

abc/cadenca1

CADENZA-I: ABC

Diese einzeilige Kadenz bildet das Beispiel 187 aus der Harmonielehre von Grabner nach.<sup>57</sup> Sie wird mit folgendem Code erzeugt:

```
\begin{abc}[name=abc/cadenca1]
X:1
M:none
```

<sup>56</sup>) vgl. *Moine, Jean-Francois*: Jef's page. ABC music notation software; o.J. [2018] ⇒ <http://moinejf.free.fr/> – RDL: 2019-02-27, wp.. Seine Quellen werden öffentlich gehostet. Das Github-Repository lizenziert diese über die Datei 'Copying' unter der GPL-3.0 (vgl. *[Github]: abcm2ps* [Repository]; o.J. [2019] ⇒ <https://github.com/leesavide/abcm2ps/> – RDL: 2019-02-27, wp.), einer von der OSI anerkannten Open-Source-Lizenz (→ <https://opensource.org/licenses/LPPL-1.3c>).

<sup>57</sup>) vgl. *Grabner*: Allgemeine Musiklehre, 1974, S. 107.

## 2.1 ABC: einfach und vielfach genutzt (★★★)

```
L:1/4
K: C
"T" [C4E4G4] "S" [F4A4c4] "D" [G4B4d4] ||
w: I IV V
"T" [A,4C4E4] "S" [D4F4A4] "D" [E4^G4B4] ||
w: I IV V
\end{abc}
```

### 2.1.3 Kadenz II: mehrzeilig

Als Beleg dafür, dass auch die *ABC*-Methodik mittlerweile tatsächlich mehrsystemige Konstrukte erzeugen kann, hier ein entsprechendes Beispiel:

abc/cadenca2

CADENZA-II: ABC

Es wird mit folgendem Code erzeugt:

```
\begin{abc}[name=abc/cadenca2]
X: 1
L: 1/4
K: D
M: none
%%score { RH | LH }
V: RH clef=treble name="Piano" stem=up
V: LH clef=bass stem=down
[V:RH] [F2d2] [F2^d2] [B2e2] [B2^e2] |
[V:LH] "T" [D,2A,2] "(D7)" [B,,2A,2] "Sp7" [D,2G,2] "D79" [C,2G,2] |
[V:RH] [B2f2] [e2^g2] [e2(a1)a1] [a2f2] ||
[V:LH] "Tp" [D,2F,2] "DD7" [B,,2D,2] "D4-3" [A,,2(D,1)C,1] "T" [D,2D,,2] ||
\end{abc}
```

### 2.1.4 Bewertung

Offensichtlich kommt man mit der *ABC*-Notationsmethode und  $\text{\LaTeX}$  dem (je intendierten) 'Original' recht nahe, und zwar ohne größeren Schreibaufwand: Die musikalische Notation ist (fast) intuitiv verständlich, Stufen- und Funktionssymbole werden als normale Schriftzeichen in das Notenbild integriert, und zwar über die Option, Liedtexte (Wörter) unter Noten und 'Griffsymbole' über Noten einzufügen. Gleiches gilt für die mehrzeilige Kadenz.

Man sieht diesen Beispielen aber auch an, dass das Ergebnis optisch nicht optimal ist: So wird die Breite der Notensysteme automatisch auf die Breite des Druckbereiches

## 2 Backends: Komplexe Notationssysteme

gesetzt,<sup>58</sup> was bei kürzeren Beispielen zu unschönen Dehnungen führt. Wichtiger ist jedoch, dass man nicht in der Lage ist, die eingefügten Analysesymbole typographisch der Stufen- oder Funktionstheorie entsprechend zu gestalten: Weder können hoch- und tiefgestellte Kleinsymbole oder Sonderzeichen eingefügt werden, noch die Alterationszeichen  $\sharp$ ,  $\flat$  oder  $\natural$  aus den Sonderzeichen - ganz zu schweigen von der Einbettung jener ausgefeilten Konstrukte, die das *harmony*-Paket zur Verfügung stellt.<sup>59</sup>

Außerdem muss man gelegentlich dort, wo man die Einfachheit der Notation erhalten will, kleine 'Hacks' verwenden und Unsauberkeiten in Kauf nehmen – wie wir es etwa bei dem Vorhalt in der 2. Kadenz getan haben. Solche Stellen typographisch sauber zu gestalten, würde verlangen, die Notation in mehrere Stimmen aufzublähen.<sup>60</sup> Und das hätte das Handling deutlich kompliziert.

Schließlich bliebe zu erwähnen, dass auch das ABC-Verfahren sozusagen auf einer kleinen beliebten 'Mogelei' beruht: es wird der Notentext ja nicht in  $\text{\LaTeX}$ -Code verwandelt, sondern in ein Bild, das dann in den  $\text{\LaTeX}$ -Code eingebunden wird.<sup>61</sup> Und wie immer kann die Einbindung von externen Graphiken ungünstigenfalls zu Irritationen hinsichtlich des Seitenumbruchs führen.<sup>62</sup>

Trotzdem bietet die *ABC*-Notationsmethode ein technisch ausgereiftes Verfahren mit einer sehr steilen Lernkurve, das schnell ansprechende Ergebnisse erzeugt. Wer sich für seine Verwendung entscheidet, nimmt gewisse optische und funktionelle Einschränkungen in Kauf, die in der theoretischen Musikwissenschaft nur bedingt akzeptabel sind. Allerdings eignet man sich mit der *ABC*-Notationsweise eine Technik an, die Dank der

---

<sup>58)</sup> Es soll jedoch die Möglichkeit geben, Parameter an das *abcm2ps*-Tool aus dem Notencode heraus zu übergeben, mit dem solche Aspekte zu steuern wären. Unglücklicherweise ist es uns nicht gelungen, das zu aktivieren.

<sup>59)</sup> Das kann ja auch nicht anders sein, weil die *ABC-Notation* außerhalb von  $\text{\LaTeX}$  in ein Bild umgewandelt wird, sodass eventuell noch eingebettete *(La)TeX*-Konstrukte gar nicht mehr ausgewertet würden.

<sup>60)</sup> Tatsächlich bietet ABC Plus 'nur' die Option, die Stimmen innerhalb eines Systems für das ganze Stück festzulegen, nicht aber taktweise (vgl. [Gonzato: Making Music with Abc 2, 2018, S. 49f](#)). Wir hätten also – um dem Preis einer signifikanten Mehrarbeit – die unschöne Halteklammer im Bass des zweiten Kadenz vermeiden können, wenn wir den Bass durchweg zweistimmig angelegt hätten. Sie einfach als  $\flat$  zu notieren, ist aber keine Option, weil *abc* den folgenden Schlussakkord im Bass dann unter die Vorhaltsauflösung im Diskant positioniert. Deshalb unser Hack der 'Vorhaltsverdoppelung mit gleichem Ton' (→ S. 21). Wir werden später erkennen, dass *PMX* unter demselben Positionierungsproblem leidet (→ S. 36). Allerdings bietet es die taktweise Notation in mehreren Stimmen, was die Mehrarbeit erträglich macht.

<sup>61)</sup> Die Stufen des Prozesses spiegeln sich in den Dateien, die anhand des Namensparameters gebildet werden: eine *.abs*-Datei, die korrespondierenden Postscriptdateien *.ps* und *.eps* und die finale *.pdf*-Datei.

<sup>62)</sup> Dennoch ist diese Idee natürlich nicht ehrenrührig, im Gegenteil: wir werden noch sehen, dass andere Methoden sie auch verwenden: LilyPondBook wäre ein Beispiel dafür, die manuelle Verbindung von *PMX* und *MusixTeX* die andere. Und man darf daraus sofort auch folgern, dass man selbst ebenso vorgehen kann, um den Output ganz anderer Notensatzprogramme in seine  $\text{\LaTeX}$ -Code einzubinden. Wir werden die Einbindung von Graphiken deshalb auch ganz generell beschreiben. (→ S. 49)

## 2.2 MusiX<sub>TeX</sub>: komplex, kompliziert und schön (★★★★)

vielfältigen Konverter in vielen Szenarien verwendet werden kann.<sup>63</sup>

## 2.2 MusiX<sub>TeX</sub>: komplex, kompliziert und schön (★★★★)

Das MusiX<sub>TeX</sub>-Tutorial erwähnt mehrfach, dass man Notentexte mit dieser Beschreibungssprache eher nicht 'händisch' erarbeiten wolle: Anfänger sollten - wie es heißt - nicht mit MusiX<sub>TeX</sub> beginnen, sondern lieber gleich *PMX* lernen.<sup>64</sup> Zwar könne man MusiX<sub>TeX</sub>-Befehle durchaus auch manuell in eine  $\text{\LaTeX}$ -Datei einfügen. Gleichwohl würden es die meisten weniger anstrengend finden, die dabei anstehenden Entscheidungen durch einen Präprozessor wie *PMX* treffen zu lassen.<sup>65</sup> Wollte man jedoch Fließtext und Musik in einem Dokument vereinen, dann sei die direkte Verwendung von MusiX<sub>TeX</sub> durchaus eine Alternative.<sup>66</sup>

Wir teilen diese letzte Abgrenzung. Mehr noch: wir meinen, dass es Rahmen einer musikwissenschaftlichen Arbeit deutlich produktiver ist, seine MusiX<sub>TeX</sub> encodierten Kadenzen, Motive oder Analysen etc. 'manuell' in den  $\text{\LaTeX}$ -Fließtext einzufügen, anstatt Umwege über *PMX* zu gehen. Was *PMX* betrifft, so werden wir den Schwierigkeiten später noch genauer nachgehen.

Für die direkte Verknüpfung von  $\text{\LaTeX}$  und MusiX<sub>TeX</sub> gäbe es hingegen - wie es heißt - zwei grundsätzlichen Methoden: entweder erzeuge man mit MusiX<sub>TeX</sub> Bilder seiner Notenseiten<sup>67</sup> - und zwar unabhängig von  $\text{\LaTeX}$  - und binde diese Bilder dann mit  $\text{\LaTeX}$ -Standardmitteln - unabhängig von MusiX<sub>TeX</sub> - in seinen Code ein. Oder man schreibe seinen MusiX<sub>TeX</sub>-Code direkt als Bestandteil seines  $\text{\LaTeX}$ -Quellcodes.<sup>68</sup> Beides habe Vor- und Nachteile, ganz zuletzt aber gelte dann doch:

*„On balance, the direct method (of embedding musical excerpts in  $\text{\LaTeX}$  documents) is probably to be preferred.“<sup>69</sup>*

### 2.2.1 Technische Vorbereitung

Auch die Nutzung von MusiX<sub>TeX</sub> bedarf einiger systemischen Vorarbeiten<sup>70</sup>:

<sup>63)</sup> Pars pro toto vgl. *Rosen, Paul*: ABC Notation [Wordpress Plugin]; o.J. [2018]  $\Rightarrow$  <https://de.wordpress.org/plugins/abc-notation/> - RDL: 2018-12-29, wp..

<sup>64)</sup> vgl. *Vogel, Oliver* et al.: MusiX<sub>TeX</sub>. Using  $\text{\TeX}$  to write polyphonic or instrumental music; Version 1.28. [Revised File]; 2018  $\Rightarrow$  <http://texdoc.net/texmf-dist/doc/generic/musixtex/musixdoc.pdf> - RDL: 2018-12-08, S. iii.

<sup>65)</sup> vgl. ds., a.a.O., S. 1.

<sup>66)</sup> vgl. ds., ebda.

<sup>67)</sup> z.B. im Format *EPS* = Encapsulated Postscript

<sup>68)</sup> vgl. ds., a.a.O., S. 114.

<sup>69)</sup> vgl. dazu ds., ebda.

<sup>70)</sup> Zum generellen Verhältnis von  $\text{\TeX}$ ,  $\text{\LaTeX}$ , MusiX<sub>TeX</sub> und verwandter Software vgl. auch *Tennent, Bob*: MusiX<sub>TeX</sub> and Related Software; o.J.  $\Rightarrow$  [https://icking-music-archive.org/software/htdocs/More\\_Related\\_Software.html](https://icking-music-archive.org/software/htdocs/More_Related_Software.html) - RDL: 2019-01-17, wp.

## 2 Backends: Komplexe Notationssysteme

- Sofern es die  $(La)TeX$ -Distribution nicht eh schon mitliefert, muss das MusiX $TeX$ -Paket<sup>71</sup> installiert werden.<sup>72</sup>
- Wie bei  $\LaTeX$  üblich, gilt es danach, das MusiX $TeX$ -Paket mit dem Kommando `\usepackage{musixtex}` in die Präambel einzubinden.
- Schließlich müssen auch hier die  $\LaTeX$ -Durchgänge zur Erzeugung des PDFs modifiziert werden. MusiX $TeX$  verwendet etwas, was als „three pass system“<sup>73</sup> bezeichnet wird: Beim ersten  $\LaTeX$ -Durchgang schreibt das MusiX $TeX$ -Paket Informationen über die benötigten Noten und ihre Kennzeichnungen in eine Datei, die wie das  $TeX$ -Dokument heißt, für das der  $\LaTeX$ -Durchgang gestartet worden ist – nur dass diese Auslagerungsdatei anstelle der Extension `.tex` die Exzension `.mx1` erhält. Danach muss für diese `.mx1`-Auslagerungsdatei das Tool `musicflx` aufgerufen werden. `musicflx` optimiert die Darstellung und schreibt seine Berechnungen in eine 'namensgleiche' Datei mit der Extension `mx2`. Und schließlich liest das MusiX $TeX$ -Paket beim nächsten  $\LaTeX$ -Durchgang die Optimierungsinformationen wieder ein und integriert sie in das endgültige Dokument.<sup>74</sup> Der Algorithmus ist also einfach: Findet das MusiX $TeX$ -Paket als Teil des  $\LaTeX$ -Durchgangs (noch) keine `.mx2`-Auslagerungsdatei, schreibt es die `.mx1`-Auslagerungsdatei, ansonsten verarbeitet es die gefunden `.mx2`-Auslagerungsdatei. Dies kann automatisiert werden. Schwierig daran ist nur, dass man selbst den Aufruf seiner Tools entsprechend organisieren muss. Von alleine passiert das eben nicht. Die entsprechende Passage aus einem Makefile könnte so aussehen:<sup>75</sup>

<sup>71)</sup> vgl. *anon.* [CTAN]: MusiX $TeX$  2018, wp.. Diese Paketbeschreibung gibt an, dass *MusiX $TeX$*  unter der GPL-2.0 distribuiert werde. Das ist eine anerkannte Open-Source-Lizenz (→ <https://opensource.org/licenses/GPL-2.0>).

<sup>72)</sup> Bei Ubuntu 18.04 ist es im Paket *texlive-music* enthalten.

<sup>73)</sup> Vgl. *Vogel et al.*: MusiX $TeX$ , 2018, S. 5.

<sup>74)</sup> Vgl. dazu ds., ebda. Bei näherem Hinsehen 'verknäuel't sich die Beziehung von  $\LaTeX$  und MusiX $TeX$  etwas und muss 'aufgedrösel't werden. Wenn es darum geht, welches Tool welche Datei erfolgreich auswertet, gilt es zu unterscheiden:

- MusiX $TeX$ : (1) `etex mutx-file.tex` → `mtx-file.mx1` (2) `musicflx mutx-file.mx1` → `mutx-file.mx2` (3) `etex mutx-file.tex` → `mutx-file.dvi`
- $\LaTeX$ : (1) `latex latx-file.tex` → `latx-file.mx1` (2) `musicflx latx-file.mx1` → `latx-file.mx2` (3) `latex latx-file.tex` → `latx-file.dvi`

Wichtig ist dabei, dass die Dateien *mutx* und *latx* einen verschiedenen, auf MusiX $TeX$  bzw.  $\LaTeX$  ausgerichteten Inhalt haben, obwohl es beide immer noch  $TeX$ -Dateien sind. Neben den Tools *etex* und *latex*, die – wie beschrieben – dvi-Dateien erzeugen, gibt es noch die Tools *pdflatex* und *musixtex*. Beide erzeugen direkt pdf-Dateien: *pdflatex* aus einer reinen  $\LaTeX$ -Datei, *musixtex* aus einer reinen MusiX $TeX$ -Datei. In unserem Fall erzeugen wir jedoch  $\LaTeX$ -Dateien mit MusiX $TeX$ -Anteilen. Deshalb müssen wir – sozusagen händisch – den Aufruf von *musicflx* selbst organisieren.

<sup>75)</sup> Wiederum gilt: Wissenschaftliche  $\LaTeX$ -Dokumente mit Fußnoten und bibliographischen Angaben werden eh schon über mehrere  $\LaTeX$ -Durchgänge hinweg erzeugt: Jeder einzelnen Durchgang lagert später benötigte Angaben in Hilfsdateien aus und liest die, die er selbst verwenden will, aus Dateien ein, die vorhergehende erzeugt haben. Wie man beides schlüssig automatisiert, kann im Makefile dieses Tutorials eingesehen werden (→ <http://github.com/kreincke/mycsrf/tree/master/examples/musicology.de>)



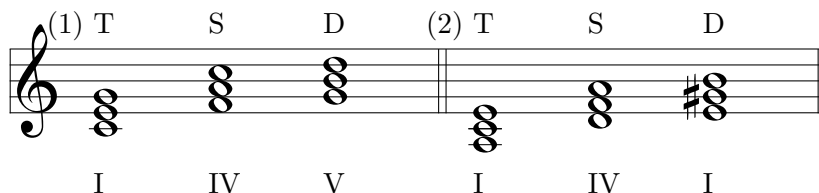
## 2.2 MusiX<sub>TEX</sub>: komplex, kompliziert und schön (★★★★)

```
.tex.pdf:
# (A) the first latex pass also creates the mx1 file
@ pdflatex $<
# (B) create the help files for the literature
@ bibtex 'basename $< .tex'
@ makeindex 'basename $< .tex'.nlo -s cfg/nomencl.ist \\
    -o 'basename $< .tex'.nls
# (C) create the mx2 file
@ musixflx $<
# (D) create the final document
@ pdflatex $<
@ pdflatex $<
```

Nach Abschluss dieser Vorarbeiten kann man dann beliebig viele MusiX<sub>TEX</sub>-Sektionen in seinem L<sup>A</sup>T<sub>E</sub>X-Quellcode erzeugen und mit MusiX<sub>TEX</sub>-Notationen füllen, wobei jede Sektion mit `\begin{music}` eröffnet und mit `\end{music}` beendet wird:

### 2.2.2 Kadenz I: einzellig

Wir beginnen auch hier wieder mit der Nachbildung des Beispieles aus der Harmonielehre von Grabner<sup>76</sup>:



CADENZA-I: MUSI<sub>TEX</sub>

Man sieht, mit MusiX<sub>TEX</sub> und L<sup>A</sup>T<sub>E</sub>X kommt man dem 'Original' wirklich nahe, näher, als mit der Rekonstruktion anhand der ABC-Methodik. Die einzellige Kadenz in der MusiX<sub>TEX</sub>-Version wird durch folgenden Code generiert:

```
\begin{music}%
\largemusicsize%
% using defaults: \instrumentnumber{1}% + \setstaves{1}{1}
% + \setclef{1}{\treble} + no bar type + \generalsignature{0}%
\nobarnumbers%
\startextract%
\setdoublebar%
\nOTES\lcharnote{10}{(1) }\uptext{T}\zchar{-10}{I}\zw{ce}\wh{g}\en%
\nOTES\uptext{S}\zchar{-10}{IV}\zw{fh}\wh{j}\en%
```

<sup>76)</sup> vgl. *Grabner: Allgemeine Musiklehre*, 1974, S. 107.


## 2 Backends: Komplexe Notationssysteme

```

\NOTES\upertext{D}\zchar{-10}{V}\zw{gi}\wh{k}\en%
\bar%
\NOTES\lcharnote{10}{(2)}\upertext{T}\zchar{-10}{I}\zw{ac}\wh{e}\en%
\NOTES\upertext{S}\zchar{-10}{IV}\zw{df}\wh{h}\en%
\NOTES\upertext{D}\zchar{-10}{I}\sh{g}\zw{eg}\wh{i}\en%
\setdoublebar%
\endextract%
\end{music}%

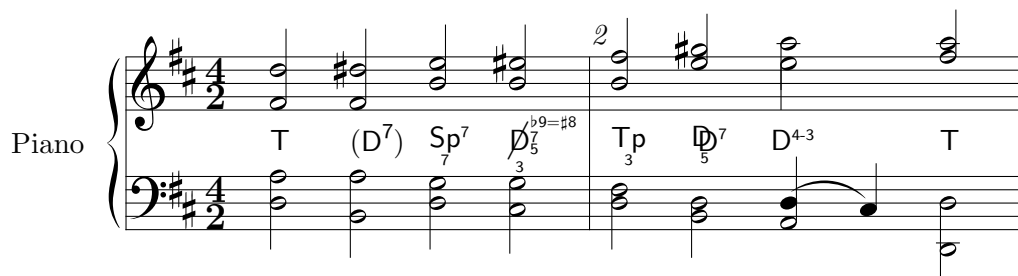
```

Zumindest prinzipiell ermöglicht es MusiX<sub>TEX</sub> auch, Notentext in den Fließtext einzubetten. Dazu unterbindet man den Absatzumbruch, indem man das Kommando

`\let\extractline\relax` in den Quellcode einfügt: . Gleichwohl kann das kaum schöne Ergebnis erzeugen, weil eine Notensatzzeile selbst mit sehr reduzierten Informationen immer noch deutlich höher ist, als eine normale Textzeile, sodass der Zeilenfluss geradezu 'stolpern' muss.

### 2.2.3 Kadenz II: Zweizeilig für ein Instrument

MusiX<sub>TEX</sub> bietet zwei Methoden der Instrumentation an: Die eine ordnet jedem Instrument eine Stimme zu, die andere fasst mehrere Stimmen resp. Systeme - wie bei einem Klaviertext - zu einem Instrument zusammen. Hier deshalb zunächst eine Kadenz für ein Instrument, das in mehreren Systemen notiert wird:



CADENZA-II: MUSI<sub>TEX</sub>

Diese Zeilen werden mit folgendem MusiX<sub>TEX</sub>-Code generiert:

```

\begin{music}
\normalmusicsize
\parindent4em
\instrumentnumber{1}
\setstaves{1}{2}
\setclef{2}{\treble}
\setclef{1}{\bass}
\setname{1}{Piano}

```

## 2.2 MusiX<sub>TEX</sub>: komplex, kompliziert und schön (★★★★)

```
\generalsignature{2}% D-DUR
\generalmeter{\meterfrac{4}{2}}
\startextract
\NOTes\zmidstaff{\HH.T....}\zh{K}\hl{a}|\zh{f}\hu{k}\en%
\NOTes\zmidstaff{(\HH.D..7...)}\zh{I}\hl{a}|\zh{f}\sh{k}\hu{k}\en%
\NOTes\zmidstaff{\HH.Sp.7...7.}\zh{K}\hl{N}|\zh{i}\hu{l}\en%
\NOTes\zmidstaff{\HH.\Dohne.3..7.9.}\zh{J}\hl{N}|\zh{i}\hu{l}\en%
\bar
\NOTes\zmidstaff{\HH.Tp.3....}\zh{K}\hl{M}|\zh{i}\hu{m}\en%
\NOTes\zmidstaff{\HH.\DD.5...7.}\zh{I}\hl{K}|\zh{l}\sh{n}\hu{n}\en%
\NOTes\zmidstaff{\HH.D....4-3.}
  \zh{H}\isluru{0}{K}\qu{K}\tslur{0}{J}\qu{J}|
  \zhl{l}\hl{o}\en%
\NOTes\zmidstaff{\HH.T....}\zh{D}\hl{K}|\zh{m}\hu{o}\en%
\setdoublebar
\endextract
\end{music}
```

Zudem zeigt dieses Beispiel sehr eindringlich, dass man in den MusiX<sub>TEX</sub>-Code direkt auch die MusiX<sub>TEX</sub>-fremden Analysensymbole von *harmony* einbetten kann.

### 2.2.4 Kadenz III: Zweizeilig für mehrere Instrumente

Bei der zweiten Methode wird jedem Instrument ein eigenes Notenliniensystem zugeordnet und die Stimmen für die Instrumente zu einer Partitur zusammengefügt. Die einzelnen Stimmen der Instrumente können selbst auch Akkorde enthalten. Dem entsprechend müssen die Noten verteilt werden.<sup>77</sup>

Das folgende Beispiel demonstriert zudem, dass MusiX<sub>TEX</sub> neben der zentrierten Darstellung über das Kommando `\startextract` auch eine linksorientierte Darstellung per `\startpiece` zur Verfügung stellt, um ganze Passagen in den eigene Text einzufügen<sup>78</sup>:

Diskant

Bass

T

D<sup>8</sup><sub>3-3<sup>b</sup></sub>

S<sub>3-3<sup>b</sup></sub>

D<sup>7</sup><sub>8-7</sub>

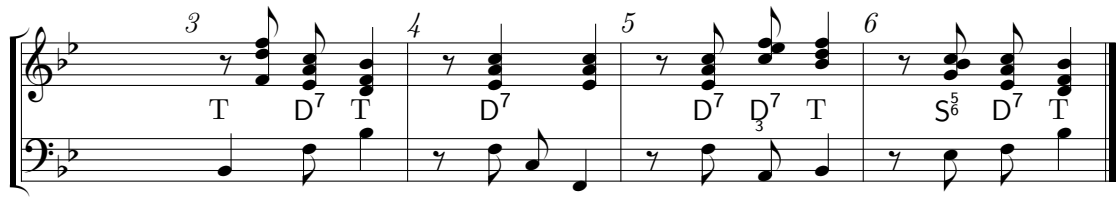
D<sub>p</sub><sub>8-8<sup>b</sup></sub>

D<sup>7</sup><sub>5-3</sub>

<sup>77</sup>) Aus Platzgründen belassen wir es hier symbolisch bei zwei 'Manualen'.

<sup>78</sup>) Bei längeren Notentexten gelingt es *musicflx* gut, ein ausgewogenes Bild zu erzeugen. Bei kürzeren muss man gelegentlich über den Parameter `\hsize=XYZmm` oder über den erzwungenen Zeilenumbruch `\alalign` in die Verteilung eingreifen.

## 2 Backends: Komplexe Notationssysteme



CADENZA-III: MUSIXTEX

Der entsprechende MusiXTEX-Code sieht so aus:

```
\begin{music}
  \smallmusicsize
  \parindent4em
  \instrumentnumber{2}
  \setstaves{2}{1}
  \setstaves{1}{1}
  \setclef{2}{\treble}
  \setclef{1}{\bass}
  \songtop{2}
  \songbottom{1}
  \setname1{Diskant}
  \setname2{Bass}
  \generalsignature{-2}
  \generalmeter{\meterfrac58}
  \startpiece
    % Takt 1/1: B: 2 8tel (Kurzschreibweise) + D: 4tel AKK
    \NOTes \zmidstaff{T} \Dqbl I b & \zq{ik}\qu{m}\en
    % Takt 1/3: B: 2 8tel explizit mit Vorzeichen + D: 4tel AKK
    \NOTes \zmidstaff{\HH.D.3-3$\flat$.8.7..}
      \ibl{0}{a}{0}\qb{0}{a}\tbl{0}\fl{a}\qb{0}{a} &
      \zq{j}\rq{1}\qu{m} \en
    % Takt 1/5: B: punktierte 16tel explizit mit Vorzeichen + D: 8tel AKK
    \NOTes \zmidstaff{\HH.S.3-3$\flat$....}
      \ibbl{2}{N}{0}\qbp{2}{N}\roff{\tbbbl{2}\fl{N}\tqb{2}{N}} &
      \zq{il}\cu{p}\en
  \bar % T2: 2*(B:2*8. + D:4. AKK) + B:2*16. + D:8. AKK
  \NOTes \zmidstaff{\HH.D.8-7.7..}
    \Dqbl M L & \zq{j1}\qu{o} \en
  \NOTes \zmidstaff{\HH.Dp.8-8$\flat$....}
    \ibl{1}{K}{0}\qbp{1}{K}\roff{\tbbbl{1}\fl{K}\tqb{1}{K}} &
    \zq{hm}\qu{o}\en
  \NOTes \zmidstaff{\HH.D.5-3.7..}
    \ibbl{2}{J}{0}\qbp{2}{J}\roff{\tbbbl{2}\tqb{2}{H}} &
    \zq{j}\rq{1}\cu{m} \en
```

## 2.2 MusiX<sub>TEX</sub>: komplex, kompliziert und schön (★★★★)

```
%\alaligne
\bar %T3 B:8.+D:8.P + B:8.P+D:8.AKK + B:8.+ D:8.AKK + B:4.+D:4.AKK
  \notes \zmidstaff{T} \qa I & \ds \zq{fk}\cu{m}\en
  \notes \zmidstaff{\HH.D..7...} \ca M & \zq{eh}\cu{j}\en
  \notes \zmidstaff{T} \qa b & \zq{df}\qu{i}\en
\bar %T4: 8. P + (B:2*8. + D:4. AKK) + B:8. + D:8.AKK + 8. P
  \notes \ds & \ds \en
  \notes \zmidstaff{\HH.D..7...} \ca{M J} & \zq{eh}\qu{j}\en
  \notes \qa F & \zq{eh}\qu{j}\en
\bar %T5: 8. P + 2*(B:8. + D:8. AKK) + 8. P + B:8. + D:8.AKK
  \notes \ds & \ds \en
  \notes \zmidstaff{\HH.D..7...} \ca M & \zq{eh}\cu{j}\en
  \notes \zmidstaff{\HH.D.3.7...} \ca H & \zq{j}\rq{1}\cu{m}\en
  \notes \zmidstaff{T} \qa I & \zq{ik}\qu{m}\en
\bar % T6: 3*(B:8. + D:8. AKK) + B:4. + D:4. AKK
  \notes \ds & \ds \en
  \notes \zmidstaff{\HH.S..6...} \ca L & \zq{g}\rq{i}\cu{j}\en
  \notes \zmidstaff{\HH.D..7...} \ca M & \zq{eh}\cu{j}\en
  \notes \zmidstaff{T} \qa b & \zq{df}\qu{i}\en
\Endpiece
\end{music}
```

### 2.2.5 Einschätzung

Die mit MusiX<sub>TEX</sub> erreichbare optische Qualität ist bestechend. Graphische Feinheiten sind bis ins Letzte hinein darstellbar. Externe Analyse- und Theoriesymbole vermag man problemlos als L<sup>A</sup>T<sub>E</sub>X-Konstrukte in den Quellcode einzubetten. L<sup>A</sup>T<sub>E</sub>X, musikwissenschaftlichen Methodik und MusiX<sub>TEX</sub> können also einfach verknüpft werden; sie stehen sich nicht gegenseitig im Weg. Und typographische Abstriche aufgrund von Unzulänglichkeiten oder Eigenarten der gewählten Tools braucht man bei dieser Konstellation nicht zu machen.

Diese Freiheit und Qualität wird mit einer aufwendigen und nicht eben intuitiven Syntax der Auszeichnungssprache MusiX<sub>TEX</sub> erkaufte: Die Verknüpfung von Achtelnoten unter einem Stamm mag als Beleg dafür gelten. Wer MusiX<sub>TEX</sub> nutzt, wird nicht nur das Handbuch wirklich verstanden haben müssen, er wird für die Details auch immer wieder darauf zurückkommen.

Der Gewinn für einen musikwissenschaftlichen Text ist allerdings groß: typographisch ist alles erreichbar. Dabei erleichtert – wie üblich – eine saubere, zeilenorientierte Programmierung das Arbeiten.<sup>79</sup> Und nach einiger Zeit wird man per Copy&Paste gewiss auch flüssiger vorankommen.

<sup>79)</sup> Wir selbst haben diese bei unserem dritten Kadenzbeispiel aus Platzgründen leider nicht vollständig umsetzen können.

## 2.3 PMX (und M-TX): die Nicht-Vereinfachung (★★)

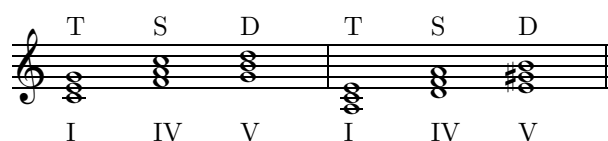
### 2.3.1 Das große Versprechen: Kadenz I und III

Wer MusiX<sub>TEX</sub> nutzt, dem drängt sich immer wieder einmal der Gedanke auf, dass das doch auch einfacher gehen müsste. Und in der Tat: Legt man für die vielen Kleinigkeiten, mit denen man bei MusiX<sub>TEX</sub> jeden Einzelfall gestalten darf, ein paar Standards fest, sollte man eine einfachere Sprache 'erfinden' können. Mit der würde man seine Notentexte formulieren. Und das Ergebnis würde zuletzt maschinell wieder in MusiX<sub>TEX</sub>-Code übersetzt werden. Genau das ist der Grundgedanke von *PMX*<sup>80</sup>, einem Präprozessor für MusiX<sub>TEX</sub>.<sup>81</sup>

Ein *PMX*-Tutorial nennt MusiX<sub>TEX</sub> eines der besten Programme für den elektronischen Notensatz.<sup>82</sup> Allerdings habe es ein nicht eben intuitives 'Look and Feel', biete kein 'WYSIWYG' und könne ob seiner Natur als symbolische Auszeichnungssprache (nicht nur Musiker) entmutigen<sup>83</sup>: Selbst nach einer sorgfältigen und sauberen Installation bleibe das Setzen einer Partitur ein mühsames Unterfangen.<sup>84</sup>

Einen Ausweg aus diesem Dilemma – so das Tutorial – biete *PMX*, das als Präprozessor den Eingabeprozess dramatisch vereinfache und so etwas liefere, was zu den einfachsten Möglichkeiten gehöre, Notenblätter elektronisch zu erarbeiten.<sup>85</sup> Die Gegenüberstellung der MusiX<sub>TEX</sub>-Variante und der *PMX*-Variante derselben Noten unterstreicht diese vollmundigen Ankündigungen.<sup>86</sup>

Betrachtet man die Sache logisch, ist die Emphase nicht unangemessen: Es gehört zum Wesen eines Präprozessors, dass er einen (meist) vereinfachten Code nimmt und auf die (meist) komplexere und kompliziertere Hauptsprache abbildet. Das hieße in diesem Fall, dass die herausragenden typographischen Fähigkeiten von MusiX<sub>TEX</sub> erhalten blieben, ohne dass man weiterhin seine überbordende Syntagmen zu tippen hätte. Und dass dem tatsächlich so ist, können auch wir an unser kleinen Grabner-Kadenz<sup>87</sup> demonstrieren:



CADENZA-I: PMX

<sup>80)</sup> vgl. anon. [CTAN]: pmx for MusiX<sub>TEX</sub> 2018, wp.. Diese Paketbeschreibung gibt an, dass *PMX* unter der GPL-2.0 distribuiert wird. Das ist eine anerkannte Open-Source-Lizenz (→ <https://opensource.org/licenses/GPL-2.0>).

<sup>81)</sup> Vgl. ds., a.a.O., S. 4ff.

<sup>82)</sup> vgl. Noack, Cornelius C.: Typesetting music with PMX; 2013 ⇒ <http://icking-music-archive.org/software/pmx/pmxccn.pdf> – RDL: 2018-12-31, S. 2.

<sup>83)</sup> vgl. ds., ebda.

<sup>84)</sup> vgl. ds., a.a.O., S. 3.

<sup>85)</sup> vgl. ds., ebda.

<sup>86)</sup> vgl. ds., ebda.

<sup>87)</sup> vgl. Grabner: Allgemeine Musiklehre, 1974, S. 107.

## 2.3 PMX (und M-TX): die Nicht-Vereinfachung (★★)

Der entsprechende kommentierte PMX-Code sieht so aus:

```
%% PREAMBLE; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% nstaves | ninstr | mtrnuml | mtrdenl | mtrnump | mtrdenp | npickup
      1      1      3      0      0      0      0
% nkeys | npages | nsystems | musicsize | fracident
      0      0      4      16      .0
% no instrument name = blank line

% tremble
t
./
% BODY: HEADER: a smaller width than the line width
w80m
%% MUSIC: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\zcharnote{-10}{I}\ \zcharnote{+10}{T}\ c04 ze zg
\zcharnote{-10}{IV}\ \zcharnote{+10}{S}\ f04 za zc
\zcharnote{-10}{V}\ \zcharnote{+10}{D}\ g04 zb zd
|
\zcharnote{-10}{I}\ \zcharnote{+10}{T}\ a03 zc ze
\zcharnote{-10}{IV}\ \zcharnote{+10}{S}\ d04 zf za
\zcharnote{-10}{V}\ \zcharnote{+10}{D}\ e04 zgs zb
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% EOF
```

Man erkennt, dass sich das Handling der Notenbezeichnungen gegenüber dem MusiX<sub>TEX</sub>-Sourcetext verschlankt hat. Noch auffälliger wird das bei unser dritten Referenzkadenz:

4

CADENZA-III: PMX

Zu ihr gehört dieser Sourcecode:

## 2 Backends: Komplexe Notationssysteme

```

%% PREAMBLE: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% nstaves | ninstr | mtrnuml | mtrdenl | mtrnump | mtrdenp | npickup
      2      2      5      8      5      8      0
% nkeys | npages | nsystems | musicsize | fracident
     -2      0      3      16      .1
Bass
Diskant
bt
./
%% BODY: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% HEADER: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\\font\ref=cmr10\def\zt#1#2{\zcharnote{#1}{\ref#2}}\def\bs{$\backslash$}\
%a smaller width than the line width
w130m
%% MUSIC: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% T1: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[1 b82 b8+ ] [1 a8 a8f ] [1 g1d g3 ] | /
\zt{-7}{T}\ b44u zd zf
\zt{-7}{D7}\ cu ze zf
\zt{-7}{S}\ b8-u ze zb+ | /
%% T2: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[1 f83 e83 ] [1 d83d d13f ] [1 c13d a32 ] | /
\zt{-7}{D7}\ c4-u ze za
\zt{-7}{Dp}\ a-u zd za+
\zt{-7}{D7}\ c8-u ze zf | /
%% T3: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
b42 f83 b43 | /
r8 \zt{-7}{T}\ [uh f85 zd zf-
      \zt{-7}{D7}\ e85 zc za ]
      \zt{-7}{T}\ d4- zf zb | /
%% T4: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r8 f83 c83 f42 | /
r8 \zt{-7}{D7}\ e44 za zc e44 za zc | /
%% T5: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r8 [1 f83 a8- ] b421 | /
r8 \zt{-7}{D7}\ [u c85 za ze c85 ze zf ]
      \zt{-7}{T}\ b44u zf+ zd | /
%% T6: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r8 [1 e83 f8 ] b41 | /
r8 \zt{-7}{D7}\ [u c85 zb zg c85 za ze ]
      \zt{-7}{T}\ b44u zf zd | /
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% EOF

```

Schon der erste Akkord aus der MusixT<sub>E</sub>X-Variante

`\N0tes \Dqbl I b & \zq{ik}\qu{m} \en`



## 2.3 *PMX* (und *M-TX*): die Nicht-Vereinfachung (★★)

reduziert<sup>88</sup> sich zum schlichteren

```
[1 b82 b8+ ] | / b44u zd zf
```

### 2.3.2 ... der Wermutstropfen ...

Trotz der so euphorisch kolportierten Vorteile wächst sich *PMX* bei näherem Hinsehen für den Musikwissenschaftler zu einer Enttäuschung aus. Gehen wir die Punkte der Reihe nach durch:

#### 2.3.2.1 Inkompatibilitäten

Zunächst müssen wir 'gestehen', dass wir in diesem Kapitel unseres 'selbstreferentiellen' Dokumentes bei der Gegenüberstellung von Notentext und generierendem Code 'geschummelt' haben: Es gibt überhaupt keine Möglichkeit, *PMX*-Code in *L<sup>A</sup>T<sub>E</sub>X*-Code einzubetten und daraus dann – sozusagen in einem Rutsch – eine PDF-Datei zu erzeugen. Das liegt in erster Linie an der Arbeitsweise von *PMX*<sup>89</sup>: Als Präprozessor nimmt das entsprechende Tool *pmxab* eine *PMX*-Datei und erzeugt daraus eine *MusiX<sub>T<sub>E</sub>X</sub>*-Datei. Diese *MusiX<sub>T<sub>E</sub>X</sub>*-Datei ist von ihrem Dialekt her aber eine vollständige *MusiX<sub>T<sub>E</sub>X</sub>*-Datei und kein inkludierbares Code-Snippet, das 1:1 in die *L<sup>A</sup>T<sub>E</sub>X*-Umgebung `\begin{music} ... \end{music}` eingebettet werden könnte. Entsprechende Versuche müssen scheitern.

Die eine, 'direkte' Lösung für dieses Problem besteht darin, zuerst – ganz *L<sup>A</sup>T<sub>E</sub>X*-unabhängig – aus der *PMX*-Datei eine Graphik zu erzeugen und diese anschließend in die *L<sup>A</sup>T<sub>E</sub>X*-Datei einzubinden<sup>90</sup>:

- Dazu ruft man für eine *PMX*-Datei zunächst den *PMX*-Präprozessor auf (z.B. `pmxab candeza1.pmx`), um die korrespondierende *MusiX<sub>T<sub>E</sub>X</sub>*-Datei `candeza1.tex` zu generieren.
- Das Resultat übergibt man dann dem *MusiX<sub>T<sub>E</sub>X</sub>*-Tool als Input, und zwar mit der Option `-g`: (`musixtex -g candeza1.tex`). So wird nicht eine *PDF*-Datei erzeugt, sondern die Postscriptdatei `candeza1.ps`.
- Danach lässt man diese Postscriptdatei in eine Encapsulated-Postscript-Datei umwandeln (`ps2eps candeza1.ps`).<sup>91</sup>

<sup>88)</sup> Um die reinen Notensymbole gegenüberzustellen, haben wir diesmal die in beiden Varianten als *T<sub>E</sub>X*-Code integrierten Analysesymbole weggelassen.

<sup>89)</sup> In zweiter Linie liegt es an den Entwicklern von *PMX*. Sie haben schlicht (noch) kein korrespondierendes *L<sup>A</sup>T<sub>E</sub>X*-Paket geschaffen. Technisch mag das herausfordernd sein, unmöglich ist es nicht: Es wird ja schon gesagt, dass der von *pmxab* erzeugte Code eigentlich 'nur' auf die richtige Weise auskommentiert werden müsse, damit er in der `\begin/end{music}`-Umgebung genutzt werden könne (vgl. *Noack: Typesetting music with PMX*, 2013, S. 94). Mithin sollte der Schritt zu einer integrierten Lösung so groß nicht sein.

<sup>90)</sup> Vgl. dazu auch *ds.*, *ebda*.

<sup>91)</sup> Eine make-basierte Automatisierung dieser Schritte enthielte als Kern wohl dies:

## 2 Backends: Komplexe Notationssysteme

- Und ganz zuletzt fügt man an der Stelle der  $\text{\LaTeX}$ -Datei, wo die *eps*-Graphik erscheinen soll, den  $\text{\LaTeX}$ -Befehl `\includegraphics{candeza1.eps}` ein<sup>92</sup>

Diese Umstände müssen – wenigstens den Musikwissenschaftler – enttäuschen: Wenn man zuletzt doch eine Graphik einbindet, die man mit einem externen Tool erstellt hat, warum sollte man sich dann die Mühe antun, zwecks Erstellung einer Graphik zuerst *PMX* zu lernen, wo es doch so viele gute Notensatzprogramme gibt, die ihre Daten – auch ohne Umweg über Shellkommandos – als Postscript- oder PDF-Datei exportieren?<sup>93</sup> Das könnte doch nur dann sinnvoll sein, wenn man wenigstens auf die gestalterischen Vorzüge und Freiheiten von *MusiX $\text{\TeX}$*  zugreifen könnte. Aber nicht einmal das ist ja der Fall – wie wir gleich sehen werden.

Zunächst müssen wir aber der Wahrheit Genüge tun und darauf hinweisen, dass der Autor des *PMX*-Tutorials die Irritationen in Sachen *PMX* / *MusiX $\text{\TeX}$*  und  $\text{\LaTeX}$  kennt und auch anspricht: Seinem eigenen Bekunden nach würde beide Ansätze konkurrierend und überschneidend das Layout gestalten und viel Spezialbefehle zueinander inkompatibel definieren, unabhängig davon, dass sie bei auf  $\text{\TeX}$  aufsetzen.<sup>94</sup> Und so kommt er zu dem Schluss:

*„While with modern implementations of resources are no longer a serious problem, the incompatibility problems are, and their resolution would be a major programming task. So there have, to this day, not been any serious efforts to provide a truly merged version of  $\text{\LaTeX}$  with *PMX*.“<sup>95</sup>*

Verwendet man *PMX* trotzdem, bleiben zuletzt doch noch Wünsche offen:

---

```
.pmx.eps:
    @ make \${< 'basename \${< .pmx'.tex
    @ musixtex -g 'basename \${< .pmx'.tex
    @ ps2eps 'basename \${< .pmx'.ps
.pmx.tex:
    @ pmxab \${<
```

Vgl. dazu [Reincke: musicology.de](https://musicology.de), 2019, Makefile aus dem Verzeichnis 'pmx'.

<sup>92)</sup> Hier muss man beachten, an der Stelle, von wo aus man die *eps*-Datei einlesen will, nicht (versehentlich) auch eine 'gleichnamige' *pdf*-Datei abgelegt zu haben. In diesem Fall lädt `\includegraphics` nämlich letztere. Und das wird im  $\text{\LaTeX}$ -Dokument zu unerwarteten Seitenumbrüchen führen, nach deren Ursache man u.U. lange sucht.

<sup>93)</sup> Zu Erinnerung: *ABC* geht implizit genauso vor. Insofern träge dieses Argument eben auch *ABC*. Allerdings punktet *ABC* durch seine vielen Konverter und Tools und durch seine lange Tradition.

<sup>94)</sup> vgl. [Noack: Typesetting music with PMX](#), 2013, S. 93.

<sup>95)</sup> vgl. [ds., a.a.O.](#), S. 93f. Man muss an dieser Stelle auch im Kopf behalten, dass *PMX* nicht für die Nutzung in  $\text{\LaTeX}$  entwickelt worden ist. Es ging den Entwicklern darum, besser und einfacher qualitativ hochwertige Notenblätter zu erzeugen. Und für diesen Zweck ist die Lösung *PMX* / *MusiX $\text{\TeX}$*  sehr wohl geeignet. Nur ist die Gestaltung solcher Blätter meist nicht Aufgabe von Musikwissenschaftlern.

## 2.3 PMX (und M-TX): die Nicht-Vereinfachung (★★)

### 2.3.2.2 Unzulänglichkeiten

Die erste kleinere Unzulänglichkeit betrifft das Erscheinungsbild: Die Kadenz-III in der *PMX*-Variante<sup>96</sup> wirkt optisch weniger klar als die *MusiX<sub>TEX</sub>*-Variante.<sup>97</sup> Dies liegt am 5/8-Takt. Der kann in eine 3er- und eine 2er Hälfte oder in eine 2er und eine 3er Hälfte aufgeteilt werden. Und das 3. Achtel der 3er-Hälfte kann auftaktisch gemeint sein. Die Entscheidungen, die *PMX* – sozusagen nach Schema F – automatisch trifft, werden solchen kontextbedingten Feinheiten nicht gerecht.<sup>98</sup>

Die zweite Unzulänglichkeit betrifft die Methode, wie die Analysesymbole in den *PMX*-Code eingettet werden: *PMX* bietet zwar die Möglichkeit, Stücke auf *PMX*-Level textuell zu kennzeichnen oder Texte über oder unter den Systemen einzufügen.<sup>99</sup> Wenn es aber – wie bei der Musikwissenschaft notwendig – darum geht, Symbole und Texte graphisch variabel einzufügen, dann möchte *PMX*, dass der User auf „Inline *TEX* commands“ zurückgreift, also unter den *L<sup>A</sup>T<sub>E</sub>X*-Level auf den *TEX*-Level 'zurückfällt',<sup>100</sup> die Nutzung einer vereinfachten *PMX*-Syntax wird dann um den Preis einer komplizierten Eingabe von *TEX*-Syntagmen erkaufte.<sup>101</sup>

Was schon bei den ABC-Lösungen galt<sup>102</sup>, gilt damit auch hier: Aus der *L<sup>A</sup>T<sub>E</sub>X*-Welt können weder hoch- und/oder tiefgestellte Kleinsymbole oder Sonderzeichen eingefügt werden, noch die Alterationszeichen  $\sharp$ ,  $\flat$  oder  $\natural$  aus den Sonderzeichen - ganz zu schweigen von der Einbettung jener ausgefeilten Konstrukte, die das *harmony*-Paket zur Verfügung stellt. Denn das Konvertierungstool *pmxab*, das *MusiX<sub>TEX</sub>*-Code aus *PMX*-Code ableitet, arbeitet ja noch außerhalb von *TEX*. Und das Konvertierungstool *musixtex* versteht 'nur' *MusiX<sub>TEX</sub>*-Code, nicht aber *L<sup>A</sup>T<sub>E</sub>X*-Syntagmen.

### 2.3.3 ... und die kleine Lösung: Kadenz II

Trotz all dieser Nicklichkeiten kann es für den Musikwissenschaftler gelegentlich sehr wohl Sinn machen, im Kontext einer *L<sup>A</sup>T<sub>E</sub>X*-Arbeit<sup>103</sup> auf *PMX* zurückzugreifen: Wenn es nämlich gilt, viele längere Musikbeispiele zu verwenden, wird das manuelle Tippen des *MusiX<sub>TEX</sub>*-Codes zu einer langwierigen, fehlerträchtigen Angelegenheit werden.

Eine Lösung dafür bestünde darin, zuerst den reinen Notentext mit *PMX* zu erstellen und dann mit *pmxab* in *MusiX<sub>TEX</sub>*-Code umzuwandeln. Danach griffe man auf die

---

<sup>96</sup>) → S. 31

<sup>97</sup>) → S. 28

<sup>98</sup>) Gelegentlich wird man diese allerdings auch für reine Geschmacksfragen halten dürfen.

<sup>99</sup>) vgl. Noack: *Typesetting music with PMX*, 2013, S. 61f.

<sup>100</sup>) vgl. ds., a.a.O., S. 76f.

<sup>101</sup>) Weil *TEX* 'zu' kompliziert war, wurde *L<sup>A</sup>T<sub>E</sub>X* 'erfunden'; und weil *PMX* mit Liedtext 'zu' kompliziert war, entstand *M-Tx* (vgl. anon. [*CTAN*]: *M-Tx – A preprocessor for pmx*; o.J. [2018] ⇒ <https://ctan.org/pkg/m-tx> – RDL: 2019-01-04, wp.) – einschließlich eines Handbuches (vgl. Laurie, Dirk: *M-Tx: Music from Text*; 2016 ⇒ <http://ctan.space-pro.be/tex-archive/support/m-tx/doc/mtxdoc.pdf> – RDL: 2019-01-04, S. 3ff)

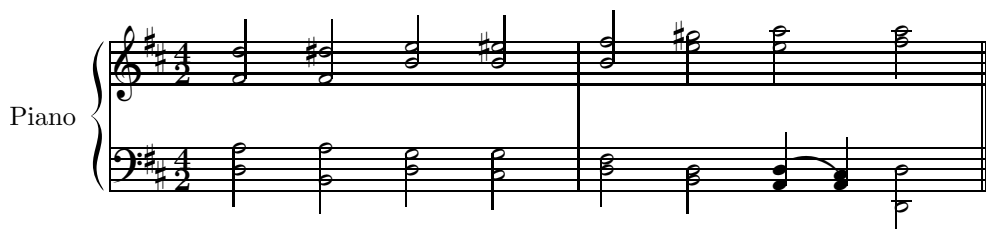
<sup>102</sup>) → S. 22

<sup>103</sup>) mit inkludiertem *MusiX<sub>TEX</sub>*-Package

## 2 Backends: Komplexe Notationssysteme

Option zurück, – wie im Tutorial als Möglichkeit angedeutet – „manuell“ bestimmte Zeilen des automatisch generierten Codes auszukommentieren und so etwas zu erzeugen, was tatsächlich in die `\begin{music}...\end{music}`- $\text{\LaTeX}$ -Umgebung kopiert werden kann.<sup>104</sup> Dieses Verfahren wollen wir noch kurz vorführen:

Wir wissen ja schon, dass wir die (beschränkten) Symbole der Harmonieanalyse, die wir auf *PMX*-Level einzugeben vermögen, bei einer Einbettung in einen  $\text{\LaTeX}$ -Code werden nicht verwenden können, eben weil es ja keine  $\text{\LaTeX}$ -Konstrukte sind, sondern  $\text{\TeX}$ -Syntagmen. Deshalb notieren wir diesmal auf *PMX*-Level nur den Notentext. Das 'reine' *PMX*/*MusiX $\text{\TeX}$* -Verfahren würde daraus die folgende Graphik erzeugen:<sup>105</sup>



CADENZA-II: PMX

Ausgegangen wird dabei von folgendem *PMX*-Code:

```
% PREAMBLE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% nstaves | ninstr | mtrnuml | mtrdenl | mtrnump | mtrdenp | npickup
      2      1      4      2      4      2      0
% nkeys | npages | nsystems | musicsize | fracident
      2      0      3      16      .1
Piano
bt
./
% BODY: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% HEADER: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%a smaller width than the line width
w130m
% MUSIC: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% T:1-1      T:1-2      T:1-3      T:1-4
      d23l za+  b-1 za+  d-1 zg      c-1 zg+ | /
      f24u zd+  f-u zd+s  bu  ze      bu zes | /
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% T:2-1      T:2-2      T:2-3      T:2-4
      d23l zf  b-1  zd  (u d4 za2 c4 za2 )  d2-1 zd+  /
      b24u zf+  e  zgs  e  za      f  za | /
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% EOF
```

<sup>104)</sup> vgl. *Noack: Typesetting music with PMX*, 2013, S. 94.

<sup>105)</sup> *PMX*-Code  $\rightarrow$  *pmxab*  $\rightarrow$  *MusiX $\text{\TeX}$* -Code  $\rightarrow$  *music $\text{\TeX}$*  -g  $\rightarrow$  PS-Code  $\rightarrow$  *ps2eps*  $\rightarrow$  EPS-Graphik

## 2.3 PMX (und M-TX): die Nicht-Vereinfachung (★★)

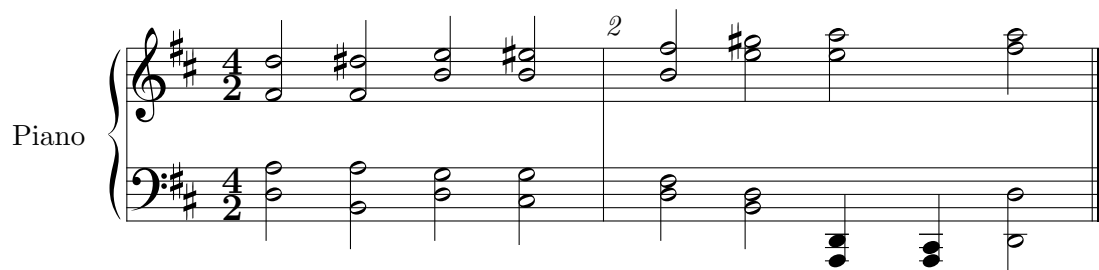
Lässt man diesen *PMX*-Code von `pmxab` verarbeiten, entsteht ein `MusixTeX`-Code, bei dem man, will man ihn erfolgreich in eine `MusixTeX`-in-`LaTeX`-Umgebung `\begin{music}` ... `\end{music}` einbetten, noch einige Zeilen auskommentieren und einige Werte ändern muss:

```
\begin{music}
%%ignore:%% \input musixtex
%%ignore:%% \input pmx
%%ignore:%% \setmaxslurs{24}\setmaxinstruments{24}%
\normalmusicsize %%% instead of: %%% \smallmusicsize%
%%ignore:%% \nopagenumbers
%%ignore:%% \tracingstats=2\relax
%%ignore:%% \hsize=369pt
%%ignore:%% \vsize740pt
\def\nbinstruments{1}
\setstaves12
\setclef1{60}
\setname1{Piano}
\generalsignature{ 2}%
\generalmeter{\meterfrac{4}{2}}%
\parindent 4em %%% instead of %%% 37pt
%%ignore:%% \elemskip1pt\afterruleskip1.000pt\beforeruleskip0pt\relax
%%ignore:%% \stafftopmarg0pt\staffbotmarg5\Interligne\interstaff{10}\relax
%%ignore:%% \readmod{cadenca2}
%%ignore:%% \startmuflex
\startpiece
%%ignore:%% \addspace
%%ignore:%% \afterruleskip%
%%ignore:%% \znotes|\zcharnote{16}{\titles{2.0}{0}{0}{0}}\en%
% Bar count 1
\pnotes{4.00}\zh{'A'}\hl{'D'}\zh{'A'}\hl{'B'}\zh G\hl D\zh G\hl C|\zh{'d'}%
\hu{'f'}\bigsh{'d'}\zh d\hu{'f'}\zh{'e'}\hu b\bigsh e\zh e\hu b\en%
% Bar count 2
\xbar
\pnotes{4.00}\zh{'F'}\hl D\zh D\hl B|\zh{'f'}\hu b\bigsh g\zh g\hl e\en%
\pnotes{2.83}\zq A\qu D \zq A\qu C|\zh{'a'}\hl{'e'}%
%%ignore: \isu0{'D'}{.8} \tslur0A
\en%
\pnotes{4.00}\zh{'D'}\hl{'D'}|\zh{'a'}\hl{'f'}\en%
\Endpiece
%%ignore:%% \vfill\eject\endmuflex
%%ignore:%% \bye
\end{music}%
```

Vor einer erfolgreichen Nutzung diese Codes muss man in seinem `LaTeX`-Header außerdem noch die Regel `\newcommand{\pnotes}[1]{\notes}` formuliert haben, damit aus der

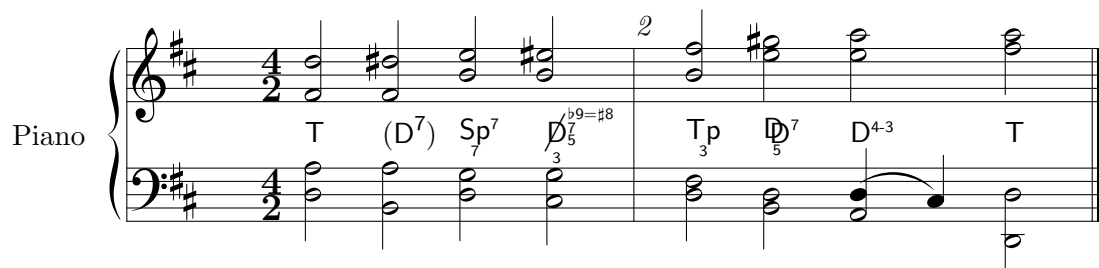
## 2 Backends: Komplexe Notationssysteme

Einbettung auf  $\text{\LaTeX}$ /MusiX $\text{\TeX}$ -Ebene – also ohne Einbindung einer externen Graphik – das folgende Bild entsteht:



CADENZA-II: PMX+MusiX $\text{\TeX}$ -IN- $\text{\LaTeX}$

An dieser Version erkennt man jedoch auch, dass die Vorstellung, man brauche bloß mal eben Zeilen auszukommentieren, der Sachlage nicht gerecht wird: Damit obiger Code überhaupt kompilierbar wurde, mussten wir die Vorhaltsklammer im Bass aus dem Code herausnehmen, was dort unmittelbar zu einer ungewollten Oktavierung führte. Tatsächlich wird man – falls man den MusiX $\text{\TeX}$ -Code in seinen  $\text{\LaTeX}$ -Code einfügen will, den `pmxab` aus der `pmx`-Datei erzeugt – ein wenig 'Tuning'-Zeit einplanen müssen. Geht man diesen Weg, kann man allerdings auch die  $\text{\LaTeX}$ -basierten Elemente einer Harmonieanalyse darin verwenden. Man hat also in der Tat die gestalterische Freiheit auf  $\text{\LaTeX}$ -MusiX $\text{\TeX}$ -Level gewonnen, ohne den komplexen und komplizierten MusiX $\text{\TeX}$ -Code als Ganzes selbst entworfen haben zu müssen. Die Grundarbeit erledigt man auf `PMX`-Level, die Feinheiten fügt man manuell ein:



CADENZA-II: PMX+MusiX $\text{\TeX}$ -IN- $\text{\LaTeX}$ ++

Hilfreich sind dabei sicher Geduld, ein geschulter Blick und der Wille zum 'Codeclearing':

```
\begin{music}%
\normalmusicsize %% instead of: %%% \smallmusicsize%
\def\nbinstruments{1}
```

## 2.4 Lilypond: komplex, elegant und schön (★★★★★)

```

\setstaves12
\setclef1{60}
\setname1{Piano}
\generalsignature{ 2}%
\generalmeter{\meterfrac{4}{2}}%
\parindent 4em %%% instead of %%% 37pt
\startpiece
% Bar count 1
\notes
  \zmidstaff{\HH.T....}\zh{'A}\hl{'D}%
  \zmidstaff{(\HH.D..7...)\zh{'A}\hl{'B}%
  \zmidstaff{\HH.Sp.7...7.}\zh G\hl D
  \zmidstaff{\HH.\Dohne.3.$\flat$9=$\sharp$8.7.5.}\zh G\hl C |%
  \zh{'d}\hu{'f}\bigsh{'d}\zh d\hu{'f}%
  \zh{'e}\hu b\bigsh{e}\zh e\hu b%
\en%
% Bar count 2
\bar
\notes
  \zmidstaff{\HH.Tp.3....}\zh{'F}\hl D%
  \zmidstaff{\HH.\DD.5...7.}\zh D\hl B |%
  \zh{'f}\hu b\bigsh g%
  \zh g\hl e%
\en%
\notes
  \zmidstaff{\HH.D....4-3.}%
  \zh{H}\isluru{0}{K}\qu{K}\tslur{0}{J}\qu{J}|%
  \zh{'a}\hl{'e}%
\en%
\notes
  \zmidstaff{\HH.T....}\zh{'D}\hl{'D} |%
  \zh{'a}\hl{'f}%
\en%
\Endpiece
\end{music}

```

## 2.4 Lilypond: komplex, elegant und schön (★★★★★)

*LilyPond* möchte guten „Notensatz für jedermann“ anbieten: Als elektronisches „Notensatzsystem“ – so das Entwicklungsteam – wolle es „[...] Notendruck in (bester) Qualität“ ermöglichen, mithin „[...] die Ästhetik handgestochenen traditionellen Notensatzes mit computergesetzten Noten [...] erreichen“.<sup>106</sup> In einem besonderen Artikel haben die LilyPond-Entwickler darge-



<sup>106)</sup> vgl. *LilyPond Development Team: LilyPond Homepage, 2018, wp.* Lizenziert ist *LilyPond*

## 2 Backends: Komplexe Notationssysteme

stellt, was das systemisch bedeutet<sup>107</sup> und welchen Konsequenzen sich daraus für ein Notensatzprogramm ergeben.<sup>108</sup> Der daraus erwachsende Anspruch ist hoch:

*„LilyPond wurde geschaffen, um die Probleme zu lösen, die wir in existierenden Programmen gefunden haben und um schöne Noten zu schaffen, die die besten handgestochenen Partituren imitieren.“<sup>109</sup>*

Wer die entsprechenden Techniken erfolgreich anwenden will, kann auf ein einfach strukturiertes Lerntutorial<sup>110</sup> und ein kürzeres Nutzungshandbuch<sup>111</sup> zurückgreifen. Letztlich wird er sich allerdings auch das umfangreiche Notationshandbuch<sup>112</sup> bereitlegen wollen.

Wie die bisher diskutierten Systeme erwartet *LilyPond*, dass man Code schreibt, keine Noten: Hier wie da ist der Texteditor das bevorzugte Werkzeug, um Musik im entsprechenden 'Dialekt' zu notieren. Trotzdem gibt es Unterschiede, die über die bloße Syntax hinausgehen:

Die wichtigste Eigenart dürfte sein, dass Lilypond konsequent zwischen Musik und Druck unterscheidet: Wer in D-Dur ein *fis* einfügen möchte, kann sich hier nicht auf die zu Beginn spezifizierte Tonart 'berufen', er muss trotzdem **fis** tippen, nicht **f**, und zwar an jeder Stelle, wo er *fis* meint. Diese Abkehr von der musikalischen Tradition hat einen gewichtigen Vorteil: Lilypond kann bei Alterationen die nötigen Vorzeichen automatisch setzen. In *g-moll* erhält die Note *f* bei eingegebenem **fis** automatisch ein Kreuz, in D-Dur nicht.<sup>113</sup>

'Augenfällig' ist zudem, dass *LilyPond* seine Elemente konsequent in einer 1:n-Beziehung anordnet: Das Notenheft besteht aus einem oder mehreren Stücken, das Stück besteht aus einem oder mehreren Notensystemen, ein Notensystem besteht aus einer oder mehrerer Stimmen, die Stimme kann solistisch oder akkordisch sein. Das Datenmodell ist mithin als Baum<sup>114</sup> ausgelegt. Und syntaktisch haben die Ebenen je eigene Markanten.

---

unter der GPL → <http://lilypond.org/gpl.html>, einer anerkannten Open-Source-Lizenz (→ <https://opensource.org/licenses/GPL-2.0>).

<sup>107</sup> vgl. *LilyPond Development Team*: Aufsatz über den automatischen Musiksatz; o.J. [2012] ⇒ <http://lilypond.org/doc/v2.18/Documentation/essay.de.pdf> – RDL: 2019-01-08, S. 5ff.

<sup>108</sup> vgl. ds., a.a.O., S. 8ff.

<sup>109</sup> vgl. ds., a.a.O., S. 2.

<sup>110</sup> vgl. *LilyPond Development Team*: LilyPond. Learning Manual; o.J. [2012] ⇒ <http://lilypond.org/doc/v2.18/Documentation/learning.de.pdf> – RDL: 2019-01-05, S. 20ff.

<sup>111</sup> vgl. *LilyPond Development Team*: LilyPond. Usage; o.J. [2018] ⇒ <http://lilypond.org/doc/v2.18/Documentation/usage.de.pdf> – RDL: 2019-01-09, S. 1ff.

<sup>112</sup> vgl. *LilyPond Development Team*: LilyPond. Notationsreferenz; o.J. [2012] ⇒ <http://lilypond.org/doc/v2.18/Documentation/notation.de.pdf> – RDL: 2019-01-05, S. 1ff.

<sup>113</sup> vgl. *LilyPond Development Team*: LilyPond Learning Manual, 2012, S. 21. In der Konsequenz wird man sich allerdings auch an – wenigstens für Deutsche – überraschende 'Töne' wie **bes**, **beses** oder **bis** gewöhnen müssen.

<sup>114</sup> vgl. anon. [Wikipedia]: Baum (Graphentheorie); o.J. [2019] ⇒ [https://de.wikipedia.org/wiki/Baum\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Baum_(Graphentheorie)) – RDL: 2019-01-11, wp..



## 2.4 Lilypond: komplex, elegant und schön (★★★★★)

Das macht das Lesen und Verstehen von *LilyPond*-Code auf Dauer einfacher, es entsteht ein klarerer Sourcetext.<sup>115</sup>

Systemisch gesehen hat LilyPond (heute) nichts (mehr) L<sup>A</sup>T<sub>E</sub>X, MusiX<sub>T</sub>E<sub>X</sub> oder T<sub>E</sub>X zu tun: es nutzt seine eigene Eingabesprache und seine eigene Maschine zum Erzeugen des Notenbildes: Als „Standardausgabeformat“ – heißt es – seien PDF<sup>116</sup> und PS<sup>117</sup> gesetzt; außerdem könnten SVG<sup>118</sup>-, EPS<sup>119</sup>- und PNG<sup>120</sup>-Dateien erzeugt werden.<sup>121</sup>

### 2.4.1 Technische Voraussetzungen

*LilyPond* sagt selbst, dass man Notenbeispiele in Form von Graphiken auch manuell in den L<sup>A</sup>T<sub>E</sub>X-Text einfügen könne, einfach indem man – zuerst und unabhängig von L<sup>A</sup>T<sub>E</sub>X – die Graphiken mit `lilypond` erzeuge und sie danach mit L<sup>A</sup>T<sub>E</sub>X-Mitteln einbinde.<sup>122</sup> Bei vielen Notenbeispielen kann das allerdings aufwendig werden, insbesondere, wenn man manuell die Länge der Notenzeilen und die Graphikbreite auf die gewünschte Zeilenlänge des Dokumentes ausrichten muss.

Deshalb bietet *LilyPond* nicht nur das Tool `lilypond` zur Erzeugung ganzer Notenblätter in den genannten Formaten an<sup>123</sup>, sondern auch das Tool `lilypond-book`: dieses „automatisiert“ die manuelle Integration, indem es die „[...] Musik-Schnipsel aus Ihrem Dokument (extrahiert), [...] `lilypond` (aufruft) und [...] die resultierenden Bilder in Ihr Dokument (einfügt)“, wobei es „[...] die Länge der Zeilen und die Schriftgröße dabei [automatisch] (dem) Dokument (anpasst)“. <sup>124</sup>

Daraus folgt sofort, dass man auch hier einiges vorzubereiten hat, wenn man *LilyPond* erfolgreich verwenden will:

► Zunächst muss man – ganz unabhängig von L<sup>A</sup>T<sub>E</sub>X – *LilyPond* installieren<sup>125</sup>. Dieses Paket stellt dann auch `lilypond-book` bereit.

► Im Gegensatz zu *ABC* oder *MusiX<sub>T</sub>E<sub>X</sub>* braucht *LilyPond* nicht in der L<sup>A</sup>T<sub>E</sub>X-Präambel aktiviert zu werden. Denn `lilypond-book` muss ja immer vor und unabhängig von L<sup>A</sup>T<sub>E</sub>X aufgerufen werden: Es generiert erst den eigentlichen L<sup>A</sup>T<sub>E</sub>X-Code, der dann keine *LilyPond*-Sektionen mehr enthält. Also kann man – direkt nach der Installation – den *LilyPond*-Quelltext eines jeden Notenbeispiels in je einer eigenen (virtuellen) Umgebung `\begin{lilypond}... \end{lilypond}` editieren. Virtuell sind diese Umgebungen insofern, als L<sup>A</sup>T<sub>E</sub>X ja nichts von *LilyPond* weiß.

<sup>115)</sup> vgl. *LilyPond Development Team: LilyPond Learning Manual*, 2012, S. 40ff.

<sup>116)</sup> Portable Document Format

<sup>117)</sup> Postscript

<sup>118)</sup> Scalable Vector Graphics

<sup>119)</sup> Encapsulated PostScript

<sup>120)</sup> Portable Network Graphics

<sup>121)</sup> vgl. *LilyPond Development Team: LilyPond Notationsreferenz*, 2012, S. 481.

<sup>122)</sup> vgl. *LilyPond Development Team: LilyPond Nutzungsreferenz*, 2018, S. 20.

<sup>123)</sup> samt aller anderen Outputformate wie *midi* u.Ä.m.

<sup>124)</sup> vgl. *ds.*, *ebda.*

<sup>125)</sup> Unter Ubuntu: `sudo apt-get install lilypond lilypond-data`

## 2 Backends: Komplexe Notationssysteme

► Schließlich muss man noch organisieren, dass den eigentlichen  $\text{\LaTeX}$ -Durchgängen zur Generierung der PDFs ein `lilypond-book`-Aufruf vorausgeht. Das kann wieder in einem Makefile organisiert werden.

Leider steckt der Teufel dabei – wie so oft – im Detail:

`lilypond-book` nimmt eine – wie wir jetzt wissen – gewissermaßen 'unechte'  $\text{\LaTeX}$ -Datei, die noch *lilypond*-Sektionen enthält, und erzeugt die entsprechenden Graphiken, bevor es die *lilypond*-Sektionen durch die korrespondierenden 'include-Graphik'-Befehle auf  $\text{\LaTeX}$ -Ebene ersetzt. Das ist der Grund, warum `lilypond-book` immer als erstes prozessiert werden muss.

Ruft man `lilypond-book` ohne weitere Parameter für eine ('unechte')  $\text{\LaTeX}$ -Datei mit der Extension *.tex* auf, beschwert es sich, dass es seine Inputdatei überschreiben müsste und verweigert die Weiterarbeit. Dies wird gelöst, indem man es mit der Option `--out IHRDIR` aufruft. Dann erzeugt `lilypond-book` einen Ordner namens *IHRDIR* und sammelt darin alle Materialien ein, die es für eine  $\text{\LaTeX}$ -compilierbare Version benötigt.

Unglücklicherweise muss man `lilypond-book` dabei ein wenig unter die Arme greifen:

Tatsächlich evaluiert und bearbeitet `lilypond-book` erfolgreich alle Dateien mit der Extension *.tex*, insbesondere auch die, die per *input*-Befehl in die Hauptdatei eingebunden worden sind. Die Aufteilung eines  $\text{\LaTeX}$ -Textes in mehrere 'Snippets' bedeutet für `lilypond-book` kein Problem. Und `lilypond-book` kopiert auch alle gefundenen Daten – ggfls. überarbeitet – korrekt in den Arbeitsordner, den man dem Tool auf der Kommandozeile mit gegeben hat, und zwar unter Beibehaltung der Ordnerstrukturen, sodass die include-Referenzen in den kopierten Dateien nicht ins Leere zeigen. Gleichwohl gibt es Dateien, die nicht in den Blick von `lilypond-book` geraten und die es darum auch nicht mit in den Arbeitsordner übernimmt. Trotzdem werden natürlich auch diese Dateien benötigt, wenn man innerhalb des Arbeitsordners erfolgreich einen  $\text{\LaTeX}$ -Lauf starten will. Prominentestes Beispiel für solche Dateien sind die *bib-Files* und ausgelagerte Konfigurationsdateien.

Deshalb muss der  $\text{\LaTeX}$ -User als Nutzer von `lilypond-book` – sozusagen manuell – die fehlenden Dateien in den von `lilypond-book` erzeugten Arbeitsordner kopieren, bevor er den 'normalen'  $\text{\LaTeX}$ -Erzeugungsprozess aufruft.

Eine entsprechendes Makefile könnte so aussehen:

```
# source directory
SRCD=source
# lilypond working directory
LPWD=lily

prod:
  # (1) create a copy of your source directory as lilypond directory
  @ cp -rd ${SRCD} ${LPWD}
  # (2) change into the lilypond directory and call lilypond for your
```

## 2.4 Lilypond: komplex, elegant und schön (★★★★★)

```
# file & let the results be written into a temporary directory
@ ( cd ${LPWD} && lilypond-book --out ../tmp your-latex-file.tex )
# (3) lilypond has created & collected the sources it needs into the
# tmp dir but unfortunately not all. Hence,
# the missed parts must still be copied into that tmp dir manually
# (4.a) ensure that all dirs really exist we need
@ mkdir -p tmp/bib/ tmp/cfg tmp/pics
# (4.b) ensure that also the missed files can be found in the tmp dir
@ cp -rd ${SRCD}/bib/* tmp/bib/
@ cp -rd ${SRCD}/cfg/* tmp/cfg/
@ cp ${SRCD}/Makefile tmp/
( cd tmp && make your-latex-file.pdf && mv your-latex-file.pdf ../ )
rm -rf ${LPWD}
rm -rf tmp
```

### 2.4.2 Kadenz I

Und damit dürfen wir die Früchte der Vorarbeit ernten: Es ist bereits absehbar, dass die eigentliche Herausforderung zuletzt nicht mehr die Nutzung von *LilyPond* als solches sein wird, sondern die Einbindung jener Symbole in den Notentext, die für den Musikwissenschaftler so wichtig sind. *LilyPond* selbst bietet 3 Verfahren an, (echten) Text in die Noten zu integrieren:

1. Man darf 'Liedtext' unter ein Notensystem setzen, wobei als Text dort auch – in einfacher Form – Analysesymbole erscheinen können. können.<sup>126</sup>
2. Man darf Makros mit Text an Noten anhängen, die dann – je nach Vorzeichen '˘' oder '˙' über oder unter der Note erscheinen. Diese Makros gibt es in einer verkürzten und einer expliziten Version. Beide sind Sonderfälle einer generellen Markupsprache.<sup>127</sup>
3. Man darf komplexer strukturierte Syntagmen der erwähnten generalisierten Markupsprache an Noten anhängen, mit denen es dann auch möglich wird, feinere Texte differenziert zu gestalten und einzufügen.<sup>128</sup>

Dabei wird der Musikwissenschaftler nicht auf die L<sup>A</sup>T<sub>E</sub>X-Sonderzeichen oder die schönen *harmony*-Konstrukte zurückgreifen können. Denn *LilyPond* arbeitet ja L<sup>A</sup>T<sub>E</sub>X-unabhängig. Also wird über die Brauchbarkeit von *LilyPond* letztlich die Frage entscheiden, ob und wie gut man die Symbole der Harmonieanalyse, die *harmony* bereitstellt, mit den Mitteln von *LilyPond* nachbilden kann und welcher Aufwand dabei entsteht.

Gehen wir diese Dinge der Reihe nach durch, beginnend – wie kaum anders zu erwarten – mit der einzeiligen Grabner-Kadenz<sup>129</sup>: Die ersten beiden Akkorde sind mit einfachen

<sup>126)</sup> vgl. *LilyPond Development Team*: LilyPond Learning Manual, 2012, S. 31ff.

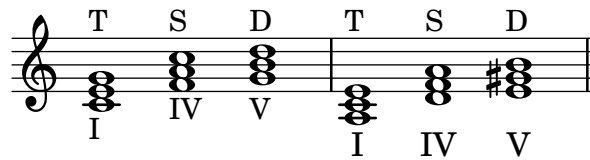
<sup>127)</sup> vgl. *LilyPond Development Team*: LilyPond Notationsreferenz, 2012, S. 211ff.

<sup>128)</sup> vgl. ds., a.a.O., S. 218ff.

<sup>129)</sup> vgl. *Grabner*: Allgemeine Musiklehre, 1974, S. 107.

## 2 Backends: Komplexe Notationssysteme

Markups klassifiziert, der dritte mit der entsprechenden abgekürzten Version. Und unter den letzten drei Akkorden sind die Symbole als 'Liedtext' eingefügt worden:



CADENZA-I: LILYPOND

Man sieht, dass die einfacheren Markup-Konstrukte noch auf einer Linie ausgerichtet werden müssten. Das ist bei der Methode 'Liedtext' schon implizit – im Rahmen der Implementation – erledigt worden. Der zugehörige Quellcode sieht so aus:

```
\begin{lilypond}
\version "2.18.2"
\header { tagline = "" }
\score {
  \new Staff {
    \relative c' {
      \time 3/1
      <c e g>1 _\markup {I} ^\markup {T}
      <f a c> _\markup {IV} ^\markup {S}
      <g b d> _"V" ^"D"
      |
      { <<
        {
          <a, c e> ^"T"
          <d f a> ^"S"
          <e gis b> ^"D"
        }
        \addlyrics {
          I IV V
        }
      } >>
    }
    \bar "||"
  }
}
\layout {
  \context {
    \Staff
    \remove Time_signature_engraver
  }
}
```

## 2.4 Lilypond: komplex, elegant und schön (★★★★★)

```
}  
}  
\end{lilypond}
```

An diesem Code erkennt man gut die systematische 1:n-Struktur eines *LilyPond*-Quelltextes: Der **score** hat eine Stimme (**staff**). All ihre Noten beziehen sich auf **c'**. Und sie hat vier Abschnitte, nämlich drei Akkorde `< ... >`, gefolgt von einer strukturierten Einheit. Dieser Komplex besteht seinerseits aus zwei Teilen, die 'gleichzeitig' (= übereinander) abgedruckt werden sollen: nochmals 3 Akkorde und die 3 Symbole, die darunter erscheinen.

Es zeigt sich aber auch, dass die naheliegenden Methoden, Funktionsanalysesymbole in die Noten zu integrieren, inhaltlich gesehen nicht ausdrucksreich genug sind und optisch nicht zufrieden stellen. Hier wird man etwas tun müssen.

### 2.4.3 Kadenz II

*LilyPond* ist nicht nur 'waschechte' *Open-Source-Software*, sondern kommt zudem mit einer Erweiterungssprache daher.<sup>130</sup> Diese zu nutzen, um die Möglichkeiten des  $\text{\LaTeX}$ -Paketes *harmony* nachzubilden, artet jedoch in echte Programmierarbeit aus und dürfte einem Musikwissenschaftler kaum mehr zuzumuten sein - wohl aber denen, die in beiden Welten zuhause ist, also uns.

Um *LilyPonds* bisherigen Mangel in Sachen musikwissenschaftlicher Kennzeichnung auszugleichen, haben wir also begonnen, eine *LilyPond*-Library zu entwickeln, die man einfach in seine *LilyPond*-Datei mit dem Befehl `\include` hinzulädt. Sie sollte im wesentlichen dieselbe Funktionalität für *LilyPond* bereitstellen, die man via *Harmony* in  $\text{\LaTeX}$  und *MusixTeX* verwenden kann.<sup>131</sup> Und aus Respekt vor der Vorlage sollte diese Bibliothek *Harmonyli* heißen, was als *LilyPond*-Datei – nicht ganz unabsichtlich – zu dem Namen `harmonyli.ly` führt.<sup>132</sup> Mittlerweile dürfen wir sagen, dass sich diese Bibliothek – auch und gerade dank der Unterstützung durch die Community – zu einem musikwissenschaftlichen Tool entwickelt hat, dass über die Fähigkeiten von *Harmony* hinausgeht und mindestens ebenso gut dokumentiert ist.<sup>133</sup>

Hat man diese Bibliothek eingebunden, kann man die entsprechende Symbole der Harmonieanalyse in seinen Notentext hinzufügen und folgendes Notenbild erzeugen:

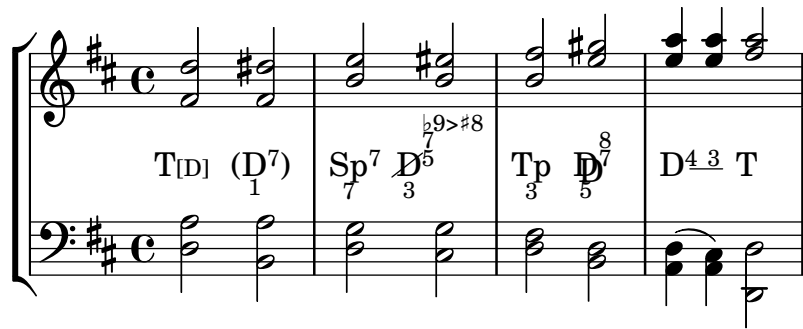
<sup>130)</sup> vgl. dazu *anon.* [Wikipedia]: GNU Guile; o.J. [2019]  $\Rightarrow$  [https://de.wikipedia.org/wiki/GNU\\_Guile](https://de.wikipedia.org/wiki/GNU_Guile) – RDL: 2019-01-14, wp..

<sup>131)</sup>  $\rightarrow$  S.16

<sup>132)</sup>  $\rightarrow$  <https://github.com/kreincke/harmonyli.ly>

<sup>133)</sup> vgl. *Reinke, Karsten*: *harmonyli.ly*. *Harmonical Analysis Symbols in LilyPond Scores*. Tutorial; 2019  $\Rightarrow$  <https://github.com/kreincke/harmonyli.ly/doc/tutorial.pdf> – RDL: 2019-11-22, S. 1ff.

## 2 Backends: Komplexe Notationssysteme



### CADENZA-II: LILYPOND

Der entsprechende *Lilypond*-Code sähe dann so aus:

```
\begin{lilypond}
\version "2.18.2"

\header { tagline = "" }
\include "lilypond/harmonyli.ly"

\score {
  \new StaffGroup {
    \time 4/4
    <<
      \new Staff {
        \relative d' {
          \clef "treble" \key d \major \stemUp
          < fis d'>2 < fis dis'>2 < b e>2 < b eis>2 |
          < b fis'>2 < e gis >2 < e a >4 < e a >4 < a fis>2 \bar "||"
        }
        \addlyrics {
          \markup \setHas "T" #'(("C"."D")("fr" . " "))
          \markup \setImHas "D" #'(("B"."1")("a" . "7")("fr" . " "))
          \markup \setHas "Sp" #'(("B"."7")("a" . "7")("fl" . " ")("fr" . " "))
          \markup \setHas "D" #'(("T"."x")("B"."3")("a" . "5")("b" . "7")("c" . "-9>+8")("fr" . " "))
          \markup \setHas "Tp" #'(("B"."3")("fl" . " ")("fr" . " "))
          \markup \setHas "D" #'(("T"."d")("B"."5")("a" . "7")("b" . "8")("fr" . " "))
          \initTextSpan " "
          \markup \openZoomRow "D" #'(("a"."4")("fl" . " "))
          \startTextSpan
          \markup \expZoomRow #'(("a"."3")("fr" . " "))
          \stopTextSpan
          \markup \setHas "T" #'(("fr" . " "))
        }
      }
      \new Staff {
        \relative d {
          \clef "bass" \key d \major \stemDown
          < d a'>2 < b a'>2 < d g>2 < cis g'>2 |
          < d fis>2 < b d>2 << { a4( a4) } { d4( cis4) } >> < d, d'>2 \bar "||"
        }
      }
    >>
  }
  \layout {
    \context { \Lyrics \consists "Text_spanner_engraver" }
  }
  \midi {}
}

\end{lilypond}
```

## 2.4 Lilypond: komplex, elegant und schön (★★★★★)

### 2.4.4 Kadenz III

Und so fehlt noch die dritte Kadenz in der Version, die *LilyPond* unter Nutzung der Bibliothek *karmonyli.ly* erzeugen kann:

Chord symbols for the first system:

T[B]     $D_3^{\frac{8}{3}}$  3-     $S_3^{\frac{3}{3}}$  3-     $D_1^{\frac{8}{1}}$  7     $Dp_3^{\frac{3}{3}}$  3-     $D_5^{\frac{7}{5}}$  3    T     $D^7$  T

Chord symbols for the second system:

$D^7$      $D^7$   $D_3^7$  T     $S_5^6$   $D^7$  T

Für diese ist folgender Quellcode zuständig:

```
\begin{lilypond}
\version "2.18.2"
\header { tagline = "" }
\include "lilypond/harmonyli.ly"

\score {
  \new StaffGroup {
    \time 5/8
    <<
    \new Staff {
      \relative c'' {
        \clef "treble" \key bes \major \stemUp
        <bes d f>4 <c es f>4 <bes es bes'>8 |
        <c es a >4 <a f' a>4 <c es f >8 |
        r8 <f, d' f>8 <a c es>8 <f bes d>4 |
        r8 <es a c>4 <es a c>4 |
        r8 <es a c>8 <c' es f>8 <bes d f>4 |
        r8 <g bes c>8 <es a c>8 <d f bes>4
      }
    }
    \new Staff {
      \relative c {
```

## 2 Backends: Komplexe Notationssysteme

```

\clef "bass" \key bes \major \stemDown
bes8[ bes'] a[ as] g16.[ ges32] |
f8[ es] d8.[ des16] c16.[ a32] |
bes4 f'8 bes4 |
r8 f8 c f,4 |
r8 f' a,8 bes4 |
r8 es8 f bes4 \bar "||"
}
}
\new RhythmicStaff {
c4 c8[ c8] c16.[ c32] | c8[ c8] c8.[ c16] c16.[ c32] |
c4 c8 c4 | r8 c2 | r8 c8 c8 c4 | r8 c8 c8 c4 |
}
\addlyrics{
\markup \setHas "T" #'(("C"."B")("fr" . " "))
\initTextSpan " "
\markup \openZoomRow "D" #'(("B"."3")("a"."3")("b"."7")("c"."8")("f1" . " "))
\startTextSpan
\markup \expZoomRow #'(("a"."3-")("fr" . " "))
\stopTextSpan
\initTextSpan " "
\markup \openZoomRow "S" #'(("B"."3")("a"."3")("f1" . " "))
\startTextSpan
\markup \expZoomRow #'(("a"."3-")("fr" . " "))
\stopTextSpan
\initTextSpan " "
\markup \openZoomRow "D" #'(("B"."1")("a"."1")("b"."7")("c"."8")("f1" . " "))
\startTextSpan
\markup \expZoomRow #'(("a"."7")("fr" . " "))
\stopTextSpan
\initTextSpan " "
\markup \openZoomRow "Dp" #'(("B"."3")("a"."3")("f1" . " "))
\startTextSpan
\markup \expZoomRow #'(("a"."3-")("fr" . " "))
\stopTextSpan
\initTextSpan " "
\markup \openZoomRow "D" #'(("B"."5")("a"."5")("b"."7")("f1" . " "))
\startTextSpan
\markup \expZoomRow #'(("a"."3")("fr" . " "))
\stopTextSpan
\markup \setHas "T" #'()
\markup \setHas "D" #'(("a"."7"))
\markup \setHas "T" #'()
\markup \setHas "D" #'(("a"."7"))
\markup \setHas "D" #'(("a"."7"))
\markup \setHas "D" #'(("B"."3")("a"."7"))
\markup \setHas "T" #'()
\markup \setHas "S" #'(("a"."5")("b"."6"))
\markup \setHas "D" #'(("a"."7"))
\markup \setHas "T" #'()
}
>>
}
\layout { \context { \Lyrics \consists "Text_spanner_engraver" } }
}
\end{lilypond}

```

### 2.4.5 Bewertung

Vergleicht man das 'Druckbild', das *LilyPond* erzeugt, mit den Noten, die MusiX<sub>TEX</sub> im Verbund mit L<sub>A</sub>T<sub>E</sub>X generiert, fällt in der Tat auf, dass die *LilyPond*-Noten irgendwie



## 2.5 Graphiken: es geht auch manuell (★★)

’weicher’, ’dichter’ und lesbarer sind: *LilyPond* wollte die Qualität des guten manuellen Notensatzes in den elektronische Notensatz einbringen.<sup>134</sup> Das ist gelungen: auch wenn sich *MusiX<sub>TEX</sub>* und *LilyPond* von der graphischen Erscheinung her nicht viel geben – den Output eines der beiden nicht exzellent zu nennen, wäre unangebracht –, so kommt doch das Druckbild von *LilyPond* einen Tick ’augenfreundlicher’ daher.

Musikwissenschaftler werden gut mit *LilyPond* ’klarkommen’: diese Notationsweise zu lernen, ist einfach, sie anzuwenden leicht. In dieser Hinsicht bietet *LilyPond* mehr als *MusiX<sub>TEX</sub>*. Und im Verbund mit *harmonyli.ly* wird *LilyPond* auf höchstem Niveau zu einem vom Forschungsgegenstand her adäquaten Tool für Musikwissenschaftler.

## 2.5 Graphiken: es geht auch manuell (★★)

Lässt man die bisher diskutierten Techniken Revue passieren, so fällt auf, dass einige von ihnen die generelle Techniken verwenden, fertige Graphiken in den  $\text{\LaTeX}$ -Code zu integrieren, anstatt Notentext über ein  $\text{\LaTeX}$ -Modul prozessierbar zu machen: *ABC*<sup>135</sup> verwendete diese ’kleine Mogelei’ ebenso wie *PMX*<sup>136</sup> und *LilyPond*.<sup>137</sup> *LilyPond* selbst hat beschrieben, was man tun muss, wenn man so vorgehen will:

*„Wenn Sie in ein Dokument Grafiken Ihres Musiksatzes einfügen möchten, so können Sie genauso vorgehen, wie Sie andere Grafiken einfügen würden: Die Bilder werden getrennt vom Dokument im PostScript- oder PNG-Format erstellt und können dann in  $\text{\LaTeX}$  oder HTML eingefügt werden.“*<sup>138</sup>

Und dabei geht es um eine sehr einfache  $\text{\LaTeX}$ -Technik:

- Als erstes muss man – wie zu erwarten – in der  $\text{\LaTeX}$ -Präambel ein spezielles  $\text{\LaTeX}$ -Paket aktivieren, und zwar mit dem Befehl `\usepackage{graphicx,color}`.
- Danach braucht man nur noch an den Stellen, wo die Graphiken erscheinen sollen, den Befehl `\includegraphics{PATH-To-YOUR-PICTURE}` einzugeben. Wichtig ist dabei, dass man die Extension der Graphik nicht an die Graphik anzuhängen braucht: liegen an der stelle verschiedene Typen derselben Graphik (PNG, EPS oder PDF), verwendet  $\text{\LaTeX}$  eines davon.<sup>139</sup>

Wer diesen Weg geht, muss drei Tücken im Auge behalten:

<sup>134)</sup> vgl. *LilyPond Development Team: LilyPond und der automatische Musiksatz*, 2012, S. 8ff.

<sup>135)</sup> → S. 22

<sup>136)</sup> → S. 33

<sup>137)</sup> → S. 42

<sup>138)</sup> vgl. *LilyPond Development Team: LilyPond Nutzungsreferenz*, 2018, S. 20.

<sup>139)</sup> Dazu noch zwei kleine Hinweise: (a) Wenn die Notensatzprogramme Postscriptgraphiken exportieren, muss man das Ergebnis in der Regel noch in EPS-Dateien konvertieren, um sie git in eine  $\text{\LaTeX}$ -Dokument einbinden zu können. Unter Linux dient dazu das Kommando `ps2eps`. (b) Wenn verschiedene Versionen eines Bildes als Dateien mit gleichem Hauptnamen und divergierender Formatextension im Ordner liegen, sollte man sehr wohl die gewünschte Dateiendung angeben. Andernfalls sucht sich  $\text{\LaTeX}$  nämlich irgendeines heraus, was unangenehme Seiteneffekte haben kann, nach deren Ursache man dann länger sucht.

1. Die Bildgröße muss händisch auf die Druckbreite ausgelegt werden, sei es über ein Graphikprogramm, sei es über Parametrisierung des Befehls `\includegraphics`.
2. Die Auflösung der Graphik muss groß genug angelegt werden. Die übliche Auflösung von Bildern im Internet (72 dpi / pixel) ist nicht druckadäquat, es sollten mindestens 300 dpi sein.
3. Der Zeilenumbruch muss manuell überwacht werden: zu lange Bilder werden oft auf die nächste Seite verlegt und erzeugen so Leerraum.<sup>140</sup>

So bleibt den Musikwissenschaftlern zuletzt immer noch der Ausweg, eine Notendatei mit irgendeinem externen Pogramm zu erstellen, in diese mit einem beliebigen Graphikprogramm die Analysesymbole 'manuell' einzufügen und das Ergebnis mit dieser Methode in den L<sup>A</sup>T<sub>E</sub>X-Text einzubinden.

### 2.6 Mup: das veraltete Auslaufmodell (★)

Zu erwähnen bliebe schließlich noch das „music publication program“, auch „Mup“ genannt.<sup>141</sup> Es wird – genau wie „ABC“, „LilyPond“ oder „MusixT<sub>E</sub>X“ – der Klasse der „Markup-Notensatzprogramme“ zugerechnet.<sup>142</sup> Das passt zur Selbstdarstellung: seine Entwickler sagen, es nähme eine Textdatei als Input und erzeuge daraus eine qualitativ hochwertige PostScript-Datei zum Drucken von Notentexten.<sup>143</sup> Obwohl anfänglich proprietäre Software, ist das Programm seit 2012<sup>144</sup> echte Opensource-Software geworden.<sup>145</sup> Es liegt ebenso in Form verschiedener Distributionspakete vor, wie im Quelltext.<sup>146</sup>

Unglücklicherweise liefern nicht alle Distributionen *Mup* fertig integriert mit.<sup>147</sup> Zumindest einige der genannten Methoden, es manuell nachzuinstallieren, funktionieren nicht. Und die Kompilation aus den Quellen läuft – jedenfalls unter Ubuntu 18.04 – in Fehler, die – wenn überhaupt – nur mit größeren Eingriffen in das System aus dem Weg zu räumen wären. Man darf also von größeren Installationshürden ausgehen.

Ob sich diese Hürden zu nehmen lohnt, wagen wir – im Kontext von Musikwissenschaft

<sup>140)</sup> Die Alternative wäre, die Graphik in eine floating Umgebung einzubetten. Dann könnte sie aber auf einer anderen, nicht zum Text passenden Seite erscheinen und müsste gesondert referenziert werden.

<sup>141)</sup> vgl. *Arkkra Enterprises*: [Homepage of] Mup [, the] music publication program; o.J. [2017ff] ⇒ <http://www.arkkra.com/> – RDL: 2019-01-26, wp..

<sup>142)</sup> vgl. *anon.* [Wikipedia]: Liste von Notensatzprogrammen; o.J. [2019] ⇒ [https://de.wikipedia.org/wiki/Liste\\_von\\_Notensatzprogrammen](https://de.wikipedia.org/wiki/Liste_von_Notensatzprogrammen) – RDL: 2019-01-17, wp..

<sup>143)</sup> vgl. *Arkkra Enterprises*: Mup,2017, wp..

<sup>144)</sup> vgl. ds., ebda.

<sup>145)</sup> vgl. *Arkkra Enterprises*: Mup Licence; o.J. [2017ff] ⇒ <http://www.arkkra.com/doc/license.html> – RDL: 2019-01-26, wp.. Bei der Lizenz handelt es sich um eine Instanziierung der BSD-3-Clause Lizenz (⇒ <https://opensource.org/licenses/BSD-3-Clause>)

<sup>146)</sup> vgl. *Arkkra Enterprises*: Obtaining a copy of Mup Version 6.6; o.J. [2017ff] ⇒ <http://www.arkkra.com/doc/obtain.html> – RDL: 2019-01-26, wp..

<sup>147)</sup> jedenfalls nicht Ubuntu-18.04.

## 2.7 $\text{T}_{\text{E}}\text{X}$ muse: die veraltete Interimslösung (★)

und  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  – zu bezweifeln: Man müsste bei Mup eine nächste Auszeichnungssprache lernen und bekäme am Ende doch nur Graphiken, die in bekannter Manier<sup>148</sup> in den  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ -Text einzubinden wären. Andere Ausgabeformate werden in der Featureliste ebenso wenig erwähnt, wie die Möglichkeit, Harmonieanalysesymbole in den Notentext zu integrieren.<sup>149</sup>

Mag die *Mup*-Notationstechnik über die Zeit auch noch so große Verdienste erworben haben, für den heutigen Musikwissenschaftler wird sie erst dann (wieder) interessant, wenn sie auch eine solche Analysemethodik anböte. Darum belassen wir es bei diesem bloßen Hinweis auf *Mup* und reichen das spannende Abenteuer einer detaillierten Evaluation an die programmierenden Musiker oder musizierenden Programmierer weiter, die den sympathischen Außenseiter aktivieren mögen.

## 2.7 $\text{T}_{\text{E}}\text{X}$ muse: die veraltete Interimslösung (★)

Wer nach Notationssystemen für Musik als Teil von  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ -Texten sucht, wird gelegentlich auch auf *T<sub>E</sub>Xmuse* stoßen. Wengstens die CTAN-Übersicht erwähnt den Ansatz noch.<sup>150</sup> Allerdings bekundet die Paketbeschreibung der letzten und einzigen Veröffentlichung, dass es sich nur um ein „interim release“ handle und dass das Programm „strictly limited“ ausgeliefert werde<sup>151</sup>. Und auch die zugehörige Anleitung betont, das Programm sei noch „incomplete“: „[...] as it stands it can be called a ‘first stage’“.<sup>152</sup>

Obwohl das Paket graphisch ansprechende Beispiele enthält<sup>153</sup>, stehen Aufwand zur Integration und Nutzung auf der einen Seite und Nachhaltigkeit auf der anderen nicht in einem angemessenen Verhältnis: Außer dem erwähnten initialen Upload von 2005 hat es (bisher) kein weiteres Update oder Upgrade gegeben. Das Paket in diesem Zustand zu verwenden, hieße (z.Zt.) auf ein falsches Pferd zu setzen.

---

<sup>148</sup>) → S.49

<sup>149</sup>) vgl. *Arkkra Enterprises*: The Mup Music Publication program [Featurelist]; o.J. [2017ff] ⇒ <http://www.arkkra.com/doc/mupfeat.html> – RDL: 2019-01-26, wp..

<sup>150</sup>) → <https://ctan.org/topic/music> RDL: 2018-12-27

<sup>151</sup>) vgl. *anon.* [CTAN]:  $\text{T}_{\text{E}}\text{X}$ muse – Music typesetting system using  $\text{T}_{\text{E}}\text{X}$  and METAFONT; o.J. [2005] ⇒ <https://ctan.org/pkg/textmuse> – RDL: 2018-12-27, wp.. Die Paketbeschreibung gibt an, dass *musicography* unter der *LaTeX Project Public License* veröffentlicht wird. Das ist eine von der OSI anerkannte Open-Source-Lizenz (→ <https://opensource.org/licenses/LPPL-1.3c>).

<sup>152</sup>) vgl. *Garcia, Federico*:  $\text{T}_{\text{E}}\text{X}$ muse’s Main Loop; 2005 ⇒ <http://ftp.fau.de/ctan/macros/textmuse/Doc/pdf/mainloop.pdf> – RDL: 2018-12-27, S.1.

<sup>153</sup>) → <https://ctan.org/tex-archive/macros/textmuse/Samples/pdf/> RDL: 2018-12-27

### 3 Frontends: die (graphische) Eingabe

Abgesehen von der letzten Variante, werden bei den hier diskutierten Methoden die Noten zuletzt textbasiert codiert: um zu komponieren oder zu arrangieren benötigt man einen Texteditor. Dessen Output wird *ABC*, *LilyPond*, *MusiX<sub>TEX</sub>* oder *PMX* als die Backendsysteme übergeben, die daraus den eigentlichen Notentext erzeugen.

Die Arbeit mit reinen Texteditoren ist nicht das, was man unter einer musikerfreundlichen Arbeitsumgebung verstehen würde. Die Sprache der Musiker sind Noten, keine mehr oder minder kryptischen Syntagmen. Und so gibt es denn auch eine Reihe von graphischen Programmen, bei denen der Musiker entweder Noten 'schreibt', nicht Text, oder bei denen seine Texteingaben wenigstens unmittelbar visualisiert werden, ohne dass er zuvor selbst das Backend aufrufen müssten. Erstere bezeichnen wir als graphische oder visuelle Editoren, letztere als semi-graphische.<sup>154</sup>

Damit darf man fragen, ob und wie man Opensource-Notensatzprogramme dazu nutzen kann, den zuletzt im L<sup>A</sup>T<sub>E</sub>X-Dokument benötigten Code aus dem abzuleiten, was man in und mit diesen Editoren eingegeben. Diese Frage ist im Einzelfall – und nicht nur für Opensourceprogramme – einfach zu beantworten:

- Zum einen muss man überprüfen, ob es der Editor erlaubt, den Inhalt in einem der benötigten Formate zu speichern.
- Und zum anderen muss man testen, ob der Editor den Inhalt auch hinreichend 'verlustfrei' exportiert.

Dem werden wir nachgehen. Allerdings wollen wir noch etwas vorausschicken:

Frontends von Notensatzsystemen nutzen meist ein natives Dateiformat und exportieren ihren Inhalt ggfls. in Fremdformate. Sie agieren implizit als Konverter. Daneben gibt es noch eine Reihe von expliziten Konvertern, die eine Datei in dem einen Format einlesen und in einem anderen wieder abspeichern.

Sollte also ein Notensatzprogramm von sich aus das eine oder andere der in unserem Kontext benötigten Formate nicht oder nur unzureichend exportieren, bliebe immer noch die Möglichkeit, einen eigenständigen Konverter zwischenzuschalten. Zu wissen,

---

<sup>154</sup>) Für den unbedarften Nutzer sieht es manchmal so aus, als sei die Erzeugung des Notentextes integrierter Bestandteil des Frontends. Wäre dem so, schiene der Begriff 'Editor' irgendwie ungerecht. Tatsächlich verküpfen solche 'integrierten Entwicklungsumgebungen' meist aber den Editor 'nur' sehr geschickt mit der eigentlichen 'Notengenerierungsmaschine', dem Backend. Deshalb ist es in unserem Kontext sehr wohl sinnvoll, von Backends und Frontends zu sprechen und unter letzteren im Wesentlichen Editoren zu verstehen, die die eingegebenen Daten in verschiedenen Formaten abzuspeichern vermögen, – selbst wenn sie sich selbst eher als IDE, als *Integrated Development Environment* verstehen.

welche expliziten Konverter es (in welcher Qualität) gibt, könnte mithin die Anzahl der gut zu nutzenden Notensatzsysteme erhöhen.<sup>155</sup>

## 3.1 Konverter I

Wer nach Konvertierungssoftware sucht, findet drei wichtige Anlaufstellen:

Für die *ABC*-Notation gibt es eine anregende und umfängliche „Softwareseite“, die u.a. auflistet, mit welcher Software welche Formate aus *ABC*-Code erzeugt und aus welchem anderen Formaten *ABC*-Code generiert werden kann.<sup>156</sup> Für *MusiX<sub>TEX</sub>* und *PMX* bietet das Icking-Music-Archive eine ähnliche, wenn auch nicht ganz so extensive Webseite.<sup>157</sup> Und für *Lilypond* listet dessen Nutzungshandbuch eine Reihe von Konvertierungsoptionen auf.<sup>158</sup>

Diese Anlaufstellen verweisen auch auf *MusicXML*<sup>159</sup>, das sich selbst als „das Standardformat für den Austausch von digitalen Musiknoten“<sup>160</sup> bezeichnet.<sup>161</sup> In diesem Sinne listet die *MusicXML*-Homepage denn auch selbst eine Fülle von Software auf, die dieses Format zu verarbeiten vermag.<sup>162</sup>

Führt man diese Quellen zusammen, entsteht in etwa folgendes Bild von Konvertierungsvarianten:

- 
- <sup>155)</sup> Wollte man diesem Gedanken in letzter Konsequenz nachgehen, müsste man zunächst eine Liste aller Notationsformate erstellen und dann nach Konvertern suchen, die diese 'anderen' auf die Formate *ABC*, *MusiX<sub>TEX</sub>*, *PMX* oder *LilyPond* abbilden. Das kann beliebig komplex werden. Wir konzentrieren uns hier auf die freie Software. Formate, die proprietäre Programme verwenden, geraten also gar nicht erst in unseren Blick. Von daher würde eine Übersicht der Konverter, die wir erstellen, niemals alle Wege abdecken. Insofern erlauben wir uns, die Konverter 'nur' cursorisch zu sichten.
- <sup>156)</sup> vgl. *Walshaw: abc software packages*, 2018, wp..
- <sup>157)</sup> vgl. *Tennent, Bob: MusiX<sub>TEX</sub> and More Related Software*; o.J. ⇒ <https://icking-music-archive.org/software/htdocs/> – RDL: 2019-01-17, wp..
- <sup>158)</sup> vgl. *LilyPond Development Team: LilyPond Nutzungsreferenz*, 2018, S. 42ff.
- <sup>159)</sup> vgl. *anon. [Wikipedia]: MusicXML*, 2018, wp..
- <sup>160)</sup> vgl. *anon. [MakeMusic]: MusicXML Homepage*, 2018, wp..
- <sup>161)</sup> Trotzdem gibt es Alternativen, auch wenn sie in unserem Kontext (noch) keine Rolle spielen. So zielt etwa *MusixJSON* – als Format publiziert in einer github-Markdown-Datei – auf einen vereinfachten Transport übers Netz. (⇒ <https://github.com/soundio/music-json> RDL: 2019-01-17). Und es existieren bereits Konverter, wie etwa *musicJSON2abc* (⇒ <https://github.com/freakimkaefig/musicjson2abc> RDL: 2019-01-17). Die Grundidee zur Einführung dieses Formates dürfte folgende gewesen sein: XML ist syntaktisch sehr extensiv. In Zeiten kooperierenden Computer bedarf es aber auch eines schlanken Formates. Als eine solche Einkodierung hat sich längst die JSON-Notation etabliert. Mit ihr lassen sich – im Rückgriff auf das http-Protokoll – Daten im standardisierten Verfahren durchs Web übertragen. Um Musik in Notenform zu übermitteln, bedürfte es also nur eines 'JSON-Dialektes', damit man die bereits existierenden Programmieretechniken wiederverwenden könnte.
- <sup>162)</sup> vgl. *anon. [MakeMusic]: MusicXML Software*; o.J. [2018] ⇒ <https://www.musicxml.com/software/> – RDL: 2019-01-17, wp..

### 3 Frontends: die (graphische) Eingabe

	nach:	ABC	Mu- siX <sub>T</sub> E <sub>X</sub>	PMX	LilyPond
von:	ABC	×	abc2mtex	∅	abc2ly
	Mu <sub>si</sub> X <sub>T</sub> E <sub>X</sub>	∅	×	(∅)	∅
	PMX	∅	(pmxab)	×	<del>pmx2ly</del>
	LilyPond	ly2abc	∅	∅	×
	MusicXML	xml2abc, mxml2abc	∅	xml2pmx	xml2ly musicxml2ly

Diese Tabelle ist wie folgt zu lesen:

- Dass es keine Konverter gibt, die ihren Input auf dasselbe Inputformat abbilden – also etwa von *ABC* nach *ABC* oder von *Mu<sub>si</sub>X<sub>T</sub>E<sub>X</sub>* nach *Mu<sub>si</sub>X<sub>T</sub>E<sub>X</sub>* –, liegt in der Natur der Sache. [→ ×]
- Dass kein Konverter von *Mu<sub>si</sub>X<sub>T</sub>E<sub>X</sub>* nach *PMX* existiert, überrascht nicht. Schließlich ist *PMX* der Präprozessor für *Mu<sub>si</sub>X<sub>T</sub>E<sub>X</sub>*, nicht umgekehrt. Und dass als Konverter von *PMX* zu *Mu<sub>si</sub>X<sub>T</sub>E<sub>X</sub>* das bereits diskutierte PMX-Standardtool *pmxab* erwähnt wird, liegt in der Natur des Konzeptes. [→ (eingeklammert)]
- Der Konverter *pmx2ly* wird heute nicht mehr mit Lilypond ausgeliefert; es gibt Stimmen im Netz, die sagen, dass er nicht mehr weiter gepflegt werde und deshalb aus dem Bestand genommen worden sei.<sup>163</sup> [→ ~~durchgestrichen~~]
- Für einige Kombinationen haben wir (noch) keine Konverter gefunden, obwohl es prinzipiell sinnvoll wäre, wenn es solche gäbe. [→ ∅]
- Die Konverter von *LilyPond* nach *ABC* resp. von *MusicXML* nach *ABC* scheinen auf den ersten Blick nachrangig zu sein: Denn wenn wir beispielsweise schon eine *LilyPond*-Datei hätten, brächte eine Umwandlung ins *ABC*-Format per *ly2abc* keinen Gewinn mehr, weil der Musikwissenschaftler im *LilyPond*-Format mehr auszudrücken vermag, als im *ABC*-Format. Indirekt spielen diese Konverter aber sehr wohl eine Rolle. Denn mit ihnen ließe sich eine *LilyPond*- resp. *MusicXML*-Datei ins *ABC*-Format und von dort aus mittels des Konverters *abc2mtex* ins *Mu<sub>si</sub>X<sub>T</sub>E<sub>X</sub>*-Format transferieren. [→ unterstrichelt]

So ergeben sich – ganz prinzipiell – direkte und indirekte Wege, Dateien in das Format eines unserer Backendsystem *ABC*, *LilyPond*, *PMX* oder *Mu<sub>si</sub>X<sub>T</sub>E<sub>X</sub>* zu überführen:

<sup>163</sup>) → <https://lists.gnu.org/archive/html/lilypond-user/2008-08/msg00056.html>  
RDL 2019-01-17

VON	→	VIA	→	NACH
.ly	→	ly2abc	→	.abc
.xml	→	xml2abc mxml2abc	→	.abc
.abc	→	abc2ly	→	.ly
.xml	→	xml2ly musicxml2ly	→	.ly
.xml	→	xml2pmx	→	.pmx
.abc	→	abc2mtex	→	.tex
.ly	→	ly2abc	→ .abc → abc2mtex →	.tex
.pmx	→	pmxab	→	.tex
.xml	→	xml2abc, mxml2abc	→ .abc → abc2mtex →	.tex

Und daraus folgen – wiederum ganz prinzipiell – einfache Seiteneffekte für die Analyse freier graphischer Notensatzsysteme<sup>164</sup> für Musikwissenschaftler:

- Gäbe es ein sehr gutes Frontend mit *ABC*-Export, dann ließe sich dieses, sofern *abc2mtex* resp. *abc2ly* gut funktionieren, indirekt auch als Frontend für *MusiX<sub>TEX</sub>* resp. *LilyPond* nutzen.
- Gäbe es ein sehr gutes Frontend mit *MusicXML*-Export, dann ließe sich dieses, sofern *xml2pmx* resp. *xml2ly* oder *musicxml2ly* gut funktionieren, indirekt auch als Frontend für *MusiX<sub>TEX</sub>* resp. *LilyPond* nutzen.

In welchen Grenzen diese Möglichkeiten existieren, werden wir später verifizieren.

## 3.2 Editoren I

Die bereits erwähnte umfangreiche Sichtung von Notensatzprogrammen klassifiziert die Kandidaten nach den Merkmalen „kostenpflichtig“ versus „Open Source“ und „WYSIWYG-Benutzeroberfläche“ versus „Markup-Notensatz“ versus „Sequencer mit Notensatzfunktion“.<sup>165</sup> Wir bezeichnen sie als graphische resp. textbasierte 'Frontends' für Backendsysteme. Das ist berechtigt, weil die gelisteten Notensatzprogramme in der Regel Eingaben annehmen und das Ergebnis an die bisher erwähnten Techniken der Backendsysteme 'durchreichen': Das graphische oder textbasierte Editieren einer *ABC*-Notendatei ist das eine, die Umwandlung der Datei in eine PDF- oder PS-Datei mit 'Noten' das andere. In dieser Liste von Notensatzprogrammen finden sich dann die freien Frontends *Aria Maestosa*, *Brahms*, *Canorus*, *Denemo*, *Laborejo*, *Mup*, *MuseScore*,

<sup>164)</sup> Zur Erinnerung: Bei der Auswahl der Programme folgen wir der Definition für freie Software. (Vgl. dazu *Free Software Foundation: Freie Software*, 2018, wp.)

<sup>165)</sup> vgl. anon. [*Wikipedia*]: Notensatzprogramme, 2019, wp..



### 3 Frontends: die (graphische) Eingabe

*NoteEdit*, *NtEd*, *Rosegarden*.<sup>166</sup> Außerdem trifft man im Netz noch auf die Tools *Frescobaldi*, *muX2d* und *Elysium/Eclipse*. Für die *ABC-Notationsmethode* stehen die Opensource-Frontends *EasyABC* und *ABCJ* zur Verfügung.<sup>167</sup> Die wohl umfangreichste Sammlung von Musiksoftware liefert die Site *MusicXML*<sup>168</sup>, einfach weil sie schlicht auflistet, welche Software MusicXML-Dateien liest, schreibt oder liest und schreibt. Diese Liste enthält mithin auch andere Programme als solche für den Notensatz. Und sie listet proprietäre und freie Applikationen. Von den freien Notensatzprogrammen dieser Liste evaluieren wir noch *Audimus Notes*, *Free Clef*, *MusEdit*, *Jniz* und *Ptolemaic*.<sup>169</sup>

*Aria Maestosa*, *Brahms* und *Rosegarden* sind in erster Linie **Sequencer**, *Frescobaldi* und *Elysium/Eclipse* bieten 'nur' eine textuelle Eingabeschnittstelle, machen das Ergebnis der Eingabe aber direkt als Bild sichtbar. Die anderen dürfen als visuelle Notensatzprogramme angesehen werden.

Systematisch kann man die Lage so darstellen:

Frontend		Import						Change		Export						
Programm	Seite	ABC	LilyPond	Midi	MusicXML	MusiX <sub>TeX</sub>	PMX	graphisch	textuell	ABC	LilyPond	Midi	MusicXML	MusiX <sub>TeX</sub> / PMX	PDF / PS / PNG ...	SOUND
ABCJ	60	✓		✓					✓	✓		✓				
Aria Mae.	61			✓				✓		✓		✓				
Audimus N.	62			✓	✓			✓				✓	✓			
Brahms	63							?								
Canorus	63			✓	✓			✓			✓	✓	✓		✓	
Denemo	66		✓	✓	✓				✓		✓	✓	✓		✓	✓
EasyABC	69	✓		✓	✓				✓	✓		✓	✓		✓	
Elysium	71		✓						✓		✓				✓	✓
Free Clef	73				✓			?					✓			
Frescobaldi	73	✓	✓	✓	✓				✓		✓	✓			✓	✓
Jniz	75							✓			✓	✓	✓		✓	
Laborejo	76							✓	✓		✓	✓			✓	
MusEdit	77				✓			?					✓			
MuseScore	78			✓	✓			✓				✓	✓		✓	✓
muX2D	80				✓			?					✓			
NoteEdit	81							?								
NtEd	82			✓	✓			✓			✓	✓			✓	
Ptolemaic	84				✓			✓								
Rosegarden	86			✓	✓				✓		✓	✓	✓		✓	✓

Eine solche Fülle von Möglichkeiten fordert geradezu dazu auf, praktisch zu überprüfen, welche davon den Anforderungen von Musikwissenschaftlern (am besten) gerecht werden.

<sup>166)</sup> vgl. anon. [Wikipedia]: Notensatzprogramme, 2019, wp..

<sup>167)</sup> vgl. Walshaw: abc software packages, 2018, wp..

<sup>168)</sup> vgl. anon. [MakeMusic]: MusicXML Software, 2018, wp..

<sup>169)</sup> Die Evaluation von Programmen, die dezidiert für Android oder iOS oder Windows gedacht sind oder die spezielle Zwecke verfolgen – also etwa *Audovia*, *Candezii*, *Chaconne*, *Crescendo*, *Jfugue*, *flabc*, *Finale Notepad*, *Mc MusicEditor*, *Notation Pad*, *Ossia Viewer*, *Score Creator* – überlassen wir mit einer gewissen Willkür späteren Darstellungen.



### 3.3 Exkurs: MIDI unter GNU/Linux resp. Ubuntu

Deshalb werden wir testen, wie diese Tools mit der Referenzkadenz II umgehen können, in wie weit sie die Symbole der Harmonieanalyse aufzunehmen vermögen und ob sie diesen Inhalt in einem Format abspeichern können.<sup>170</sup>

Schon an dieser Tabelle fällt natürlich sofort ins Auge, dass es keine Editoren (mehr) gibt, die *MusiX<sub>TEX</sub>*-*PMX*-Dateien öffnen oder schreiben. Angesichts der Qualität und Komplexität von *MusiX<sub>TEX</sub>* ist das ein Desiderat. Wir werden deshalb nach der Sichtung der Editoren noch überprüfen, ob und in wie weit für andere Format und Zwecke gedachte Editoren über die Einschaltungen von Konvertern mittelbar dazu 'überredet' werden können.

### 3.3 Exkurs: MIDI unter GNU/Linux resp. Ubuntu

Dieser Übersicht zufolge können einige der Frontends für Notensatzsysteme mit *MIDI*-Dateien umgehen. Das impliziert oft die Möglichkeit, sich auch die Musik hinter den Noten 'über das Frontend' anzuhören. Manche Systeme bringen dazu bereits alles Nötige mit, andere erwarten eine darauf ausgelegte Umgebung. Und bei letzteren haben manche Distributoren die angeforderten Tools sauber mit paketierte, andere lassen Lücken.

Wie man Noten hörbar macht, ist eigentlich nicht unser Thema. Allerdings spielt es – sofern man nach dem besten Frontend für einen Musikwissenschaftler sucht – durchaus eine Rolle, ob die Frontends der Notensatzsysteme wenigstens prinzipiell auch das Hören von Musik erlauben. Wir werden deshalb – sehr kurz und cursorisch – am Beispiel von Ubuntu erläutern<sup>171</sup>, was notwendig ist, um *MIDI*-Datei mit freier Software über das System erklingen zu lassen. Dabei zielen wir weder auf Vollständigkeit noch Adäquanz im Detail. Das Prinzip allerdings sollte so weit klar werden, dass jeder mit der Hilfe von 'Dr. Google' sein eigenes System anpassen kann.

*MIDI* steht für „Musical Instrument Digital Interface“. *MIDI*-Dateien sind mithin keine Audiodateien im 'normalen' Sinne. Sie bieten keine digitalisierten Akustikdaten. Vielmehr enthalten sie klangunabhängige Beschreibungen von 'Stimmen', die – sozusagen 'willkürlich' – einem Sound, einem Instrument zugeordnet werden. Sie sind „musikalische Steuerinformationen für elektronische Instrumente“. <sup>172</sup> Um *MIDI*-Datei abzuspielen, braucht man also einen Player, der – im Verbund mit dem Computersoundsystem – nicht nur digitalisierte 'Analogdaten' in reale Analogdaten umwandelt<sup>173</sup>, sondern der

<sup>170)</sup> Konkret werden wir unsere Referenzkadenz-II jeweils mit den *LilyPond*-kompatiblen Frontends erfassen, das Resultat von dort aus als *LilyPond*-Datei exportieren und diese wiederum vom Eclipse-Plugin *Elysium* einlesen lassen. *Elysium* leitet aus dieser ursprünglich exportierten Datei automatisch eine PDF-Datei ab, die genauso aussehen sollte, wie das Original. Stimmt also diese Datei mit den Erwartungen überein, betrachten wir den ursprünglichen Export als verifiziert. Die anderen Exportformate werden wir entsprechend überprüfen. Einzelheiten dazu vermerken wir bei den entsprechenden Fällen.

<sup>171)</sup> getestet für Ubuntu 18.04 und 20.04

<sup>172)</sup> vgl. dazu anon. [Wikipedia]: Musical Instrument Digital Interface; o.J. [2019] ⇒ [de.wikipedia.org/wiki/Musical\\_Instrument\\_Digital\\_Interface](https://de.wikipedia.org/wiki/Musical_Instrument_Digital_Interface) – RDL: 2019-02-02, wp..

<sup>173)</sup> Die Widersprüchlichkeit dieses 'Wordings' lässt sich leicht auflösen: Musik ist zuletzt ein

### 3 Frontends: die (graphische) Eingabe

auch die soundunabhängige Beschreibung und eine Soundbibliothek zusammenbringt, um daraus die finalen analogen Musikdaten zu erzeugen.

Unter GNU/Linux bringt das Soundsystem – unabhängig davon, ob es rein auf *ALSA* basiert<sup>174</sup> oder dieses im Verbund mit *Pulse-Audio*<sup>175</sup> als „Sound-Middleware“<sup>176</sup> bereitstellt<sup>177</sup> – nicht von sich aus die Fähigkeit mit, *MIDI*-Dateien abzuspielen. Um das zu ermöglichen, kann man seinem Player – wenn er es denn erlaubt – ein zusätzliches *MIDI*-Plugin spendieren. Oder man installiert einen generell nutzbaren *MIDI*-Server. Für beides ein Beispiel:

Der *vlc* ist ein sehr vielseitiger Mediaplayer.<sup>178</sup> Um ihn samt *MIDI*-Player-Plugin zu installieren, genügt ein Kommando. Außerdem sollte man das Soundsystem noch um eine bessere Soundbibliothek erweitern:

```
sudo apt-get install vlc vlc-plugin-fluidsynth
sudo apt install fluid-soundfont-gm fluid-soundfont-gs
```

Das reicht aber noch nicht, um auch in Zukunft alle *MIDI*-Dateien automatisch vom *vlc* abgespielt zu bekommen. Oft ist schon ein Player für alle Medien-Dateien zuständig und *MIDI*-Dateien werden den Medien-Dateien zugerechnet, obwohl das Soundsystem – wie wir gelernt haben – von sich aus ja gar keine *MIDI*-Dateien versteht. Also ist es reiner Zufall, ob die voreingestellte Defaultapplikation *MIDI* 'kann' oder nicht: es hängt nur davon ab, ob sie selbst ein *MIDI*-Plugin hat und ob es auch installiert ist.

Der Ausweg aus dem Dilemma ist einfach: Man macht den *vlc* zum Standardtool für *MIDI*-Dateien. Dazu klickt man mit der rechten Maustaste<sup>179</sup> auf eine *MIDI*-Datei und wählt 'Eigenschaften' aus. In dem dann erscheinenden Dialog geht man zum Reiter *Öffnen mit*, wählt aus der Liste möglicher Applikationen den *vlc* aus und bestätigt die Wahl über den Button 'Set as Default'. Danach werden auch alle anderen *MIDI*-Dateien automatisch mit dem *vlc* 'geöffnet' – und das heißt in unserem Fall: abgespielt.

Will man *MIDI*-Dateien dagegen über die Shell abspielen, benötigt man einen *MIDI*-Server mit Playerfunktionalität, will sagen: einen „software synthesizer“.<sup>180</sup> Dazu

---

Strom sich überlagernder Frequenzen. Sie zu digitalisieren bedeutet, diesen Strom – wie die Gurke für den Gurkensalat – in Scheiben zu zerschneiden: Jede Scheibe wird ein Tupel von Zahlen, der die je zum Schnitzeitpunkt klingenden Frequenzen repräsentiert. Und wie beim Gurkensalat, so auch bei der Musik: je feiner die Scheiben, desto besser das sensorische Ergebnis, desto besser die *digitalisierten Analogdaten*. Unabhängig davon kann Musik aber auch auf einer Metaebene als System digital so beschrieben werden, dass man daraus – wie bei *MIDI* – unter Rückgriff auf externe Zutaten wieder ein Strom *digitalisierten Analogdaten* ableiten kann, die dann ihrerseits über das Computersoundsystem zu 'realen' Analog'daten' werden.

174) → [https://de.wikipedia.org/wiki/Advanced\\_Linux\\_Sound\\_Architecture](https://de.wikipedia.org/wiki/Advanced_Linux_Sound_Architecture)

175) → <https://www.freedesktop.org/wiki/Software/PulseAudio/>

176) → <https://de.wikipedia.org/wiki/PulseAudio>

177) → <https://wiki.ubuntuusers.de/Soundsystem/>

178) → <https://www.videolan.org/vlc/>

179) bei Linkshändern die linke

180) → <https://wiki.ubuntuusers.de/TiMidity/>

### 3.3 Exkurs: MIDI unter GNU/Linux resp. Ubuntu

bietet sich *timidity* an<sup>181</sup> Auch in diesem Fall sollte man die besseren Soundbibliotheken installieren. Dann muss man *timidity* allerdings noch dazu 'überreden', diese auch zu nutzen: man konfiguriert *timidity* entsprechend, in dem man in der Konfigurationsdatei `/etc/timidity/timidity.cfg` den Wert `freepats.cfg` durch den Namen der Soundbibliothek, hier `fluidr3_gm.cfg` ersetzt. Ein entsprechendes Shellskript könnte so aussehen:

```
#!/bin/bash
sudo apt-get install timidity
sudo apt install fluid-soundfont-gm fluid-soundfont-gs
# optional für ein Timidity eigenes Desктоoptool
sudo apt-get install timidity-interfaces-extra
```

Dann öffnet die Konfigurationsdatei mit `sudo vi /etc/timidity/timidity.cfg`, kommentiert die Zeile `source /etc/timidity/freepats.cfg` aus und löscht das Kommentarteichen vor `source /etc/timidity/fluidr3_gm.cfg` respektive fügt diese Zeile ein.

Danach sollte man in einer Shell über das Kommando `timidity MIDIDATEI.midi` seine *MIDI*-Datei erfolgreich abspielen können. Der Befehl `timidity -ig MIDIDATEI.midi` öffnet sogar ein graphisches Interface. Gelegentlich erwartet das eine oder andere Frontend, dass es seine Daten an einen *MIDI*-Server übergeben kann. Auch dafür kann man *timidity* benutzen. Man startet ihn von der Shell oder im Bootprozess mit der Option `iA` und erhält als Output eine Liste von Ports, auf denen *timidity* als Server lauscht. Konkret kann das so aussehen:

```
> timidity -iA
TiMidity starting in ALSA server mode
Opening sequencer port: 129:0 129:1 129:2 129:3
```

Die Programme, die ihre *Midi*-Musidaten an einen solchen Server übergeben wollen, muss man dann nur noch entsprechend konfigurieren.

Hat man all dies erledigt, kann man auch die Frontends der Notensatzsysteme zum Anhören nutzen, die zwar generell *MIDI*-bereit sind, die selbst aber keine eigene oder nur eine unvollständige *MIDI*-Umgebung mitbringen. Insbesondere sollten nun auch *elysium*, *frescobaldi* und *rosegarden* als *MIDI*-'Player' funktionieren. Allerdings wird man diesen über ihr jeweiliges Konfigurationsmenue<sup>182</sup> meist noch mitteilen müssen, dass sie über den erzeugten *Timidity*-Port kommunizieren sollen.

Bleibe noch zu erwähnen, dass bestimmte Distributionen das *gststreamer*-System als Audio- und Videointerface nutzen.<sup>183</sup> Seine Fähigkeiten erwibt dieser Layer über Plugins. Hat man also kein *gststreamer-MIDI*-Plugin installiert, kann keines der Programme, die diesen Audiolayer nutzen, *MIDI*-Dateien abspielen. Unglücklicherweise werden

<sup>181</sup>) → <http://timidity.sourceforge.net/#info>

<sup>182</sup>) oft erreichbar über 'Edit/Preferences'

<sup>183</sup>) → <https://gststreamer.freedesktop.org/>

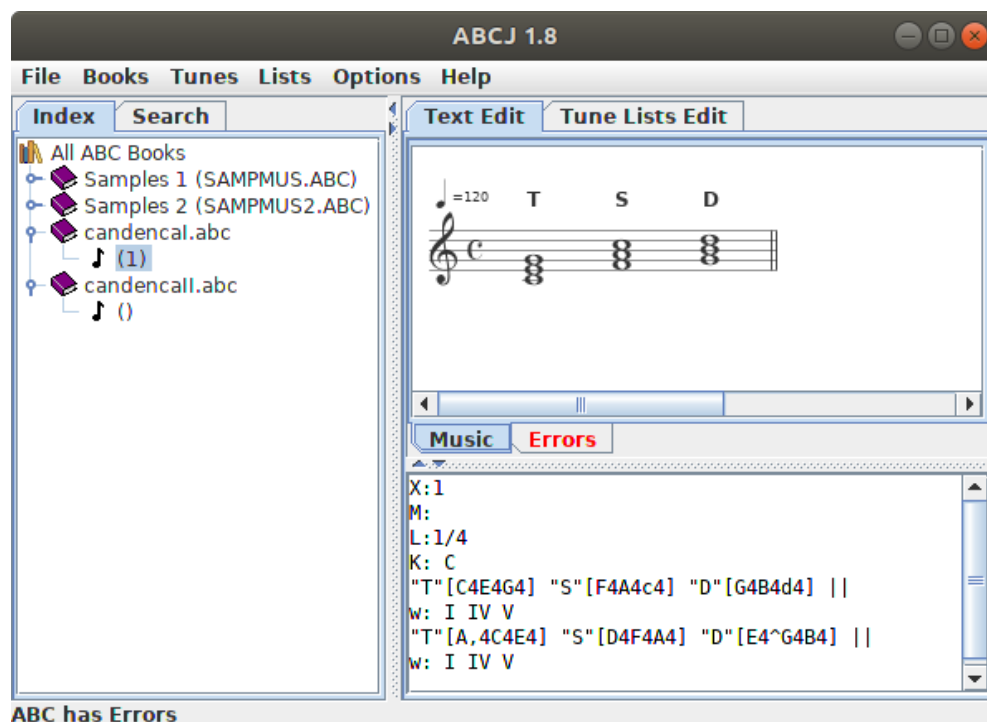
### 3 Frontends: die (graphische) Eingabe

gstreamer-MIDI-Plugins nur über die *gstreamer-plugins-bad*-Serie angeboten, nicht über die *gstreamer-plugins-good*-Serie.<sup>184</sup> Mithin wird zumindest der Purist auch dann noch Programme finden, die MIDI-Dateien nicht abspielen können, wenn er sein System auf der tieferen *ALSA/PulseAudio*-Ebene längst schon MIDI-fähig gemacht hat.<sup>185</sup>

## 3.4 Editoren II

### 3.4.1 ABCJ (★)

*ABCJ* bezeichnet sich als „Java Editor, Player und Bibliothekar für Musikdateien“.<sup>186</sup> Das Programm wird als JAR-File mit graphischem Installer zum Download angeboten; sofern man die einfachen Vorbedingungen erfüllt hat<sup>187</sup>, läuft die Installation reibungslos durch. Die herausragende Eigenschaft von *ABCJ* dürfte sein, dass es *ABC* notierte Musikstücke in 'Playlisten' erfasst, für deren einzelne Einträge es den zugehörigen *ABC*-Code editierbar macht und daraus *on the fly* das entsprechende Notenbild ableitet:



<sup>184</sup>) → <https://gstreamer.freedesktop.org/documentation/plugins.html>

<sup>185</sup>) Für tiefer gehende MIDI-Aspekt unter Linux vgl. *Felix, Ted*: *Ted's Linux MIDI Guide*; 2018 ⇒ <http://tedfelix.com/linux/linux-midi.html> – RDL: 2019-02-10, wp..

<sup>186</sup>) vgl. *Spencer-Jowett, Steve*: *The ABCJ Page*; o.J. ⇒ <http://abcj.ganderband.com/> – RDL: 2019-02-04, wp. Im Original: „a pure-Java editor, player and librarian for music files stored using Chris Walshaw's ABC notation“. Die Homepage gibt auch an, dass „ABCJ is freely available here under the terms of the GNU public license“ - also unter einer anerkannten Open-Source-Lizenz (→ <https://opensource.org/licenses/alphabetic>).

<sup>187</sup>) vgl. ds., ebda.

Damit hätte man tatsächlich einen (semi)graphischen Editor, bei dem man unten den ABC-Text eingibt und oben das graphische Notenbild erhält. Einige Eigenarten dieses Verfahrens zwingen uns allerdings, an *ABCJ* nur einen Stern zu vergeben:

- Zum ersten kann er nicht mit allen Syntagmen umgehen. So vermag er unsere Referenzkadenz I, die wir mit dem Konverter *abcm2ps* haben erfolgreich erzeugen können, nicht vollständig in Noten umzuwandeln, den Code unser Referenzkadenz II überhaupt nicht<sup>188</sup>: So bald der ABCJ-Parser das nächste Token nicht erkennt, bricht er die Umwandlung ab. Das liegt in der Natur eines Parsers. Nicht 'natürlich' ist hingegen, dass er nicht alles, was syntaktisch (in *ABC-Plus*) erlaubt ist, auch versteht.
- Zum zweiten bietet *ABCJ* als Exportformat nur *ABC*-Code und *MIDI* an. Ihn als Brücke in andere Welten zu nutzen, ist damit nur sehr bedingt möglich.
- *ABCJ* tritt als semi-graphischer Editor an.<sup>189</sup> Seine Editorfunktionen sind allerdings recht rudimentär, bieten keine Code-Completion und nur ein spartanisches Syntaxhighlighting.

Damit scheidet *ABCJ* für Musikwissenschaftler als Frontend aus. So gerne wir dem Programm auch mehr Reputation gezollt hätten – es ist echte freie Software – und so erkennbar viel Arbeit sein Autor hineingesteckt hat, so wenig ist es leider für unsere Zwecke geeignet.

### 3.4.2 Aria Maestosa (★★)

*Aria Maestosa* ist Open-Source-Software und bezeichnet sich selbst als „midi sequencer/editor“, der *MIDI*-Datei erzeugen, editieren und abspielen könne<sup>190</sup>. Er bringt eine Anleitung zur Nutzung<sup>191</sup> und eine zur Installation<sup>192</sup> mit. Die letzte Version stammt aus dem Jahr 2017.<sup>193</sup>



- <sup>188</sup>) Rein logisch gesehen, könnte der Fehler auch auf unser Seite liegen. Wir könnten unzulässigen Code genutzt haben, den 'unser' Tool *abcm2ps* – wie auch immer – noch zu tolerieren weiß. Der Effekt ist aber derselbe: *ABCJ* steht nicht ohne Zusatzüberlegungen zur Verfügung.
- <sup>189</sup>) Zur vorgreifenden Erinnerung: Das allein ist kein Ausschlusskriterium. Im Gegenteil: wir werden mit *Frescobaldi*, *EasyABC* und *Elysium* andere Frontendsystem vorführen, die dasselbe Konzept verfolgen. Allerdings sind deren textuelle Editierfunktionen reicher.
- <sup>190</sup>) vgl. anon. [SourceForge]: Aria Maestosa; o.J. [2017] ⇒ <http://ariamaestosa.sourceforge.net/> – RDL: 2019-02-03, wp.. Die Homepage sagt auch, Aria Maestosa werde unter der GPL-2.0 distribuiert. Das Programm ist mithin Open-Source-Software (→ <https://opensource.org/licenses/alphabetical>).
- <sup>191</sup>) vgl. anon. [SourceForge]: Aria Maestosa Manual; o.J. [2017] ⇒ <http://ariamaestosa.sourceforge.net/man.html> – RDL: 2019-02-03, wp..
- <sup>192</sup>) vgl. anon. [SourceForge]: Aria Maestosa Installation; o.J. [2017] ⇒ <http://ariamaestosa.sourceforge.net/building.html> – RDL: 2019-02-03, wp..
- <sup>193</sup>) → <https://sourceforge.net/projects/ariamaestosa/files/>. *Aria Maestosa* wird mittels des Kompatibilitätslayers *wxwidgets* kompiliert. Für Ubuntu gibt es nur veraltete Pakete. Der Versuch, die Version von 2017 aus den Quellen zu installieren – was ansich sehr gut beschrieben

### 3 Frontends: die (graphische) Eingabe

Das Besondere an *Aria Maestosa* ist sicherlich, dass es das Schreiben von Noten in erster Linie nicht über einen „Score Editor“ (semi-)graphisch ermöglicht, sondern über das Spielen von Instrumenten: ein „Piano Editor“ simuliert die Eingabe der Töne z.B. über die Klaviatur, ein „Guitar Editor“ über die Bünde einer Gitarre.<sup>194</sup> Das Programm liest und schreibt im Wesentlichen *MIDI*-Dateien, als Exportformat soll *aiff*<sup>195</sup> zu Verfügung stehen.<sup>196</sup> Neben dem Editieren und Abspielen bietet *Aria Maestosa* auch eine Druckfunktion an, womit es in den Bereich der Notensatzprogramme hineingreift, ohne vom Anspruch her wirklich eines zu sein.

Für den Musikwissenschaftler dürfte dieses Programm seiner spezifischen Ausrichtung wegen und ob seiner begrenzten Im- und Exportfunktionen als generelles Tool kaum in Frage kommen. Das kann man nicht dem Programm vorhalten: es tut das, was es tun will – und das offensichtlich gut. In unserem Kontext ist es uns deshalb noch zwei Sterne wert. In einem anderen könnten es mehr sein.

#### 3.4.3 Audimus (★)

*Audimus* ist Teil eines Projektes „[...] to develop educational software for music teachers and their students“.<sup>197</sup> Die Notationssoftware firmiert unter dem Namen *audimus-notes*. Es handelt sich um ein Javaprogramm, das – nach Installation einer geeigneten Java-Runtime-Umgebung – einfach mit dem Kommando `java -jar Audimus.jar` gestartet werden kann, ein entsprechendes Shellskript liegt bei. Die letzte ausgereifte Version stammt vom September 2007, ein letztes Folgerelease aus dem August 2008.<sup>198</sup>

Das Alter macht sich bemerkbar. Die Software kann gestartet werden, das graphische Frontend funktioniert aber nicht (mehr): es scheinen Fonts und Zeichenelemente zu fehlen. Damit ist dieses Notensatzprogramm, das MusicXML lesen und schreiben soll, für Musikwissenschaftler heute nicht mehr verwendbar. Sein Output hätte eh nur über Konverter mit  $\text{\LaTeX}$  verbunden werden können. Aus Gründen des Respekts möge es jedoch noch einen Stern erhalten.

---

ist – scheitert auf einem moderneren GNU/Linux daran, dass das Kompilat eine veraltete Bibliothek erwartet, die zu aktivieren nicht mehr ohne radikalen Systemumbau möglich ist.

<sup>194)</sup> vgl. anon. [SourceForge]: *Aria Maestosa*, 2017, wp..

<sup>195)</sup> Audio Interchange File Format

<sup>196)</sup> vgl. Guepewi: *Aria Meastosa*; 2017  $\Rightarrow$  <https://www.paules-pc-forum.de/forum/thread/177505-aria-maestosa/> – RDL: 2019-02-03, wp..

<sup>197)</sup> vgl. anon. [SourceForge]: *Audimus* Homepage; o.J. [2008]  $\Rightarrow$  <https://sourceforge.net/projects/audimus/> – RDL: 2019-01-31, wp.. Die SourceForge-Seite nennt als Lizenz die GPL-2.0, womit das Programm unter einer anerkannten Open-Source-Lizenz distribuiert wird ( $\rightarrow$  <https://opensource.org/licenses/alphabetical>).

<sup>198)</sup> vgl. anon. [SourceForge]: *Audimus* Download; o.J [2008]  $\Rightarrow$  <https://sourceforge.net/projects/audimus/files/> – RDL: 2019-01-31, wp..



### 3.4.4 Brahms (-)

Wikipedia zählt *Brahms* zu den „Sequenzern“, die „[...] neben ihrem Hauptanwendungsfeld der Audio- und MIDI-Bearbeitung auch Notensatzfunktionalitäten (beinhalten)“.<sup>199</sup> Wir sind geneigt, von 'Geistersoftware' zu sprechen:



Das Programm *Brahms* wird in einer älteren Sichtung auf ein Sourceforgeprojekt verlinkt<sup>200</sup>, in der aktuelleren aber nicht.<sup>201</sup> Suchmaschinen beantworten den Request 'brahms musicsoftware' im Wesentlichen mit obigen Listen, deren Duplikate und dem Sourceforgeprojekt.<sup>202</sup> Die Homepage dieses *Brahms-Sourceforge-Projektes* nutzt dann – beruhigender- und letztlich irreführenderweise – ein Logo mit Noten.<sup>203</sup> Dennoch hat (dieses) *Brahms* (von sich aus) nichts mit Musik zu tun: es sei ein „[...] Modular Execution Framework (MEF) for executing integrated systems built from component software processes“, ein „SystemML-ready execution client“.<sup>204</sup> Mittlerweile gibt es dazu ein neueres Github-Repository, das per 'Fork' aus dem Sourceforgeprojekt entstanden ist. Und unter Github beschreibt sich *Brahms* – direkt und ganz ohne Notenlogo – als „simulation execution engine“.<sup>205</sup>

Tatsächlich ist (dieses) *Brahms* kein Sequencer mit Notensatzfunktion. Und wenn es solch ein Programm früher einmal gegeben hat – wir vermuten eher, dass das Logo und die Beschreibung frühere Begutachter in die Irre führte –, dann sind die Repositories durcheinander geraten. Oder aber das richtige Notensatz- und Sequencerprogramm *Brahms* wird so versteckt gepflegt, dass auch der unbedarfte Musikwissenschaftler es nicht einfach und schnell genug finden wird. So läuft alles auf dasselbe hinaus: heute ist *Brahms* in dieser Funktion nicht nutzbar.

### 3.4.5 Canorus (★★★)

*Canorus* ist ein Notensatzprogramm<sup>206</sup>, das gelegentlich auch als Nachfolger von *NoteEdit* gehandelt wird.<sup>207</sup> Die vorletzte Version (0.7.2) aus dem Jahr 2015 wurde noch als „leichtgewichtige Alternative“ zu umfangreicheren Notensatzprogrammen



<sup>199)</sup> vgl. anon. [Wikipedia]: Notensatzprogramme, 2019, wp..

<sup>200)</sup> vgl. Callon, Gordon J.: Music Notation Software; o.J. [2009] ⇒ <http://www.acadiau.ca/~gcallon/www/others.htm> – RDL: 2019-02-12, wp..

<sup>201)</sup> vgl. anon. [Wikipedia]: Notensatzprogramme, 2019, wp..

<sup>202)</sup> → <https://www.google.com/search?q=brahms+musicsoftware> (RDL 20190209)

<sup>203)</sup> vgl. anon. [SourceForge]: Brahms Homepage; o.J. [2013] ⇒ <http://brahms.sourceforge.net/home/> – RDL: 2019-02-12, wp..

<sup>204)</sup> vgl. anon. [SourceForge]: Brahms Documentation: What is Brahms; o.J. [2013] ⇒ <http://brahms.sourceforge.net/docs/What%20is%20BRAHMS.html> – RDL: 2019-02-12, wp..

<sup>205)</sup> vgl. anon. [Github]: BRAHMS-SystemML/brahms, a modular execution middleware; o.J. [2018] ⇒ <https://github.com/BRAHMS-SystemML/brahms> – RDL: 2019-02-12, wp..

<sup>206)</sup> vgl. anon. [SourceForge]: Canorus - music score editor. [Homepage]; o.J. [2019] ⇒ <https://sourceforge.net/projects/canorus/> – RDL: 2019-01-27, wp..

<sup>207)</sup> vgl. anon. [Wikipedia]: Canorus; o.J. [2019] ⇒ <https://de.wikipedia.org/wiki/Canorus> – RDL: 2019-01-23, wp..

### 3 Frontends: die (graphische) Eingabe

bezeichnet.<sup>208</sup> Der letzte Releasekandidat (0.7.3) stammt vom Juni 2018.<sup>209</sup> Reviews der neueren Version stehen noch aus.<sup>210</sup> Vom Typ her gehört *Canorus* zu den graphischen Editoren: es erlaubt die Bearbeitung von Noten, ohne Text 'programmieren' zu müssen.

Die Homepage von *Canorus* ist die Sourceforge-Projektseite.<sup>211</sup> Aktuell wird das Programm nicht in allen Distributionen angeboten<sup>212</sup>; externe Pakete gibt es eher für ältere Programmsammlungen.<sup>213</sup> Dies dürfte einen einfachen Grund haben: Der – von 2019 aus gesehen – vorletzte veröffentlichte Release-Candidat 0.7.2 stammte aus dem Jahr 2015, der letzte aus 2018.<sup>214</sup> Über drei Jahre tat sich – von außen gesehen – in Sachen Weiterentwicklung also wenig. In der Opensourcewelt ist das üblicherweise ein Zeichen dafür, dass das Projekt 'eingeschlafen' ist.<sup>215</sup> Die Veröffentlichung vom Juni 2018 kam dann zu spät für z.B. die Ubuntu 18.04, der LTS-Distribution.<sup>216</sup> Gleichwohl sind wir sicher, dass das Programm in kommenden Distributionen wieder aufgenommen wird.

So bleibt Anfang 2019 im Wesentlichen nur die Installation aus den Quellen, falls man *Canorus* nutzen will. Das sollte allerdings Programmierer und Systemkenner nicht sehr herausfordern. Das Softwarepaket liefert eine Readme-Datei mit, die für verschiedene Umgebungen die nötigen Befehle auflistet.<sup>217</sup> Alternativ bleibt nur, auf die nächste Distribution zu warten, die uns dieses Programm wieder ohne Eigenarbeit zur Verfügung stellt.

Außer der Visualisierung am Bildschirm bietet *Canorus* selbst keinen Notensatz. Für die anderen Zwecke nutzt es *LilyPond* als Backend. Oder anders gesagt: *Canorus* fungiert als graphisches Frontend für das textuell arbeitende *LilyPond*. Das, was man mit *Canorus* erarbeitet, speichert es in einem eigenen XML-Format. Importieren kann man (gegenwärtig) *MusicXML*- und *Midi*-Dateien, exportieren auch Graphiken und *LilyPond*-Dateien<sup>218</sup>.

<sup>208)</sup> vgl. Kreussel, Peter: Neue Software; in: EasyLinux, 03 (2015) ⇒ <http://www.linux-community.de/ausgaben/easylinux/2015/03/neue-software-teil-1-2/2/> – RDL: 2019-01-27, wp..

<sup>209)</sup> vgl. anon. [SourceForge]: Canorus - music score editor. [Files]; o.J. [2019] ⇒ <https://sourceforge.net/projects/canorus/files/> – RDL: 2019-01-27, wp.. Dem Download beigelegt ist der Text der GPL-3.0. Das Programm wird also unter einer anerkannten Open-Source-Lizenz distribuiert (⇒ <https://opensource.org/licenses/alphabetical>).

<sup>210)</sup> Stand 01/2019

<sup>211)</sup> vgl. anon. [SourceForge]: Canorus Homepage, 2019, wp..

<sup>212)</sup> so nicht in Ubuntu 18.04

<sup>213)</sup> vgl. anon. [Repology]: Versions for canorus; o.J. [2019] ⇒ <https://repology.org/metapackage/canorus/versions> – RDL: 2019-01-27, wp..

<sup>214)</sup> vgl. anon. [SourceForge]: Canorus Files, 2019, wp..

<sup>215)</sup> Auch vor 2015 soll es schon Unterbrechungen bei der Entwicklung gegeben haben vgl. anon. [Ubuntu]: Canorus; o.J. [2014] ⇒ <https://wiki.ubuntuusers.de/Canorus/> – RDL: 2019-01-27, wp.

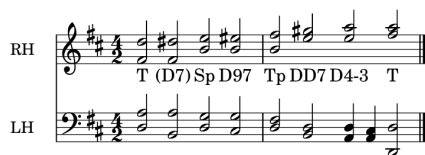
<sup>216)</sup> Long-Term-Service-Distributionen erscheinen seltener (bei Ubuntu alle 2 Jahre), werden dafür aber länger mit Updates versorgt.

<sup>217)</sup> Für Ubuntu 18.04 konnten wir verifizieren, dass die Kompilation einfach durchläuft, wenn man die geforderten Zusatzpakete – wie beschrieben – installiert.

<sup>218)</sup> Laut Homepage gäbe es auch einen Im- und/oder Export für *ABC*- oder *MusiX<sub>TEX</sub>*-



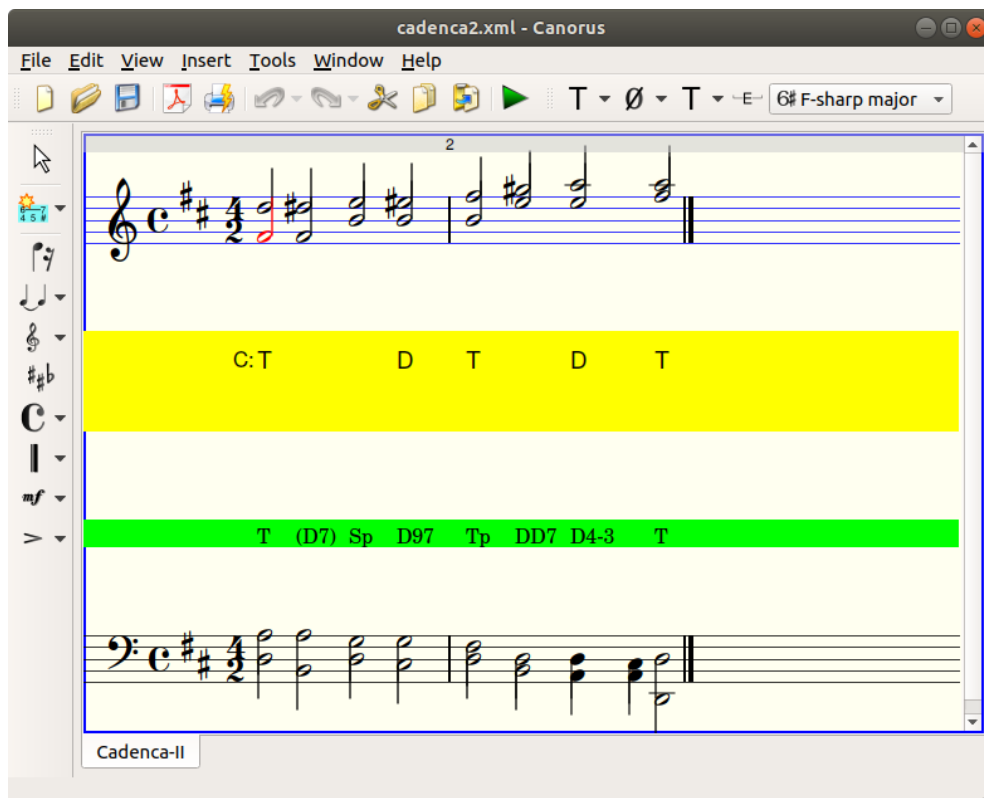
Die Arbeitsteilung zwischen *Canorus* und *LilyPond* hat zwei Konsequenzen: Zum einen reicht die Installation von *Canorus* allein nicht aus, um arbeitsfähig zu werden. Auch *LilyPond* muss systemisch bereitgestellt werden. Und zum anderen ist das, was man am Bildschirm sieht, letztlich nicht das, was man als druckfertige PDF-Datei erhält. Hier zunächst unsere Referenzkadenz, wie Canorus sie als PDF generiert. Wie kaum anders zu erwarten, ähnelt sie sehr der 'puren' *LilyPond*-Ausgabe:



Allerdings konnte der 4→3 Vorhalt mit einer  $\text{♩}$  gegen zwei gebundene  $\text{♩}$  im Bass offensichtlich nicht umgesetzt werden. Die Funktionssymbole sind dagegen erkennbar – ganz wie bei der Kadenz-I in der reinen *LilyPond*-Version<sup>219</sup> – als 'Liedtext' integriert worden. Hier wie da gilt: ohne unsere

kleine Zusatzbibliothek<sup>220</sup> können die Symbole der Funktionstheorie in einem *LilyPond*-Code nur in grober Form genutzt werden.

Dieser Ausgabe steht graphisch eine leicht anders gestaltete Eingabe gegenüber:



Man erkennt auf der linken Seite die Konzexte, die für Spezialeingaben zu aktivieren sind. Der Modus zur Eingabe von Noten wird über das Menu aktiviert. Oben in der

Notate (vgl. anon. [SourceForge]: Canorus Homepage, 2019, wp.). Der von uns getestete Releasekandidat 0.7.3 bot diese Option (noch) nicht (mehr).

<sup>219)</sup> → S. 43

<sup>220)</sup> → S. 45

### 3 Frontends: die (graphische) Eingabe

Liste erscheinen die konkreten Ausformungen. Letztlich klickt man mit der Maus dort in das Notensystem, wo die gewünschte Note erscheinen soll; Vorzeichen werden zuvor mit + oder – aktiviert. *Canorus* ist also recht intuitiv zu bedienen. Darum fällt es nur bedingt ins Gewicht, dass das Handbuch bei der manuellen Installation aus den Quellen heraus nicht mit kompiliert wird.<sup>221</sup>

Für die Eingabe von 'Liedtext' stellt *Canorus* einen besonderen Modus bereit (unten grün), den man ebenfalls im linken Randbereich aktiviert und dann über Maus und Tastatur bedient.

Außer dem Textmodus möchte *Canorus* noch einen Modus zur Harmonieanalyse bereitstellen (oben, gelb). In diesem Modus kann man – wie es unser Bild zeigt – in der oberen Programmleiste Funktionssymbole auswählen und unter bestimmte Noten einfügen. Tatsächlich versucht *Canorus* sogar, den entsprechenden Akkord zu analysieren.

Damit verfolgt *Canorus* einen vielversprechenden Ansatz, der über den anderer Notensatzsysteme hinausgeht. Gleichwohl ist das Ergebnis aus drei Gründen noch nicht produktiv nutzbar: Zum ersten werden *Canorus* eigenen Analysesymbole nicht mit gedruckt und exportiert. Zum zweiten löst die Nutzung dieses Modus noch viele Programmabstürze aus. Und drittens kann man in diesem Modus noch keine Funktionsparallelen, keine Gegenklänge und keine Doppeldominanten ausdrücken.<sup>222</sup> Darum ist dieses sehr innovative Verfahren für den heutigen Musikwissenschaftler praktisch nicht verwendbar.

Der *LilyPond*-Code, den *Canorus* exportiert, kann dagegen schon heute gut weiterverarbeitet werden. Er ist – sofern man auf die eigene und die *Canorus*-Harmonieanalyse verzichtet – gut strukturiert, lesbar und reproduzierbar<sup>223</sup> korrekt.<sup>224</sup> Ohne eine textuelle Nachbearbeitung in Sachen Harmonieanalyse wird er aber für den Musikwissenschaftler nur bedingt sinnvoll sein.

So geben wir dem Programm 3 von 5 Sternen: In Maßen kann es jetzt bereits als Frontend für *LilyPond* verwendet werden, es ist vielversprechend, wird aktuell gepflegt und erfüllt die Basisanforderungen. Als  $\beta$ -Version ist es aber – dem eigenen Anspruch gemäß – von der Funktionalität und von der Stabilität her ebenso noch begrenzt, wie vom Handling. Praktisch wird *Canorus* momentan allenfalls für den Musikwissenschaftler als *LilyPond*-Frontend in Frage kommen, der bereits mit *Canorus* vertraut ist.

#### 3.4.6 Denemo (★★★)

*LilyPond* bezeichnet *Denemo* als „graphischen Editor“ und bietet ihn – nach *Frescobaldi* – an zweiter Stelle seiner Liste von Programmen an, die das Editieren erleichtern sollen. Hervorgehoben wird hier, dass *Denemo* – neben der graphischen Manipulation des Notenbildes – auch das Editieren des *LilyPond*-Quelltextes



<sup>221</sup>) ... und bis dato auch nicht im Netz zu finden ist

<sup>222</sup>) Die angebotene Alternative der Zwischendominante funktioniert nicht zuverlässig.

<sup>223</sup>) → S.57

<sup>224</sup>) Aus Platzgründen verzichten wir auf den erneuten Abdruck des *LilyPond*-Codes

erlaube und zugleich das per *LilyPond* erzeugte PDF anzeige.<sup>225</sup> Damit wäre *Denemo* ein graphischer und ein semi-graphischer Editor.

*Denemo* selbst sagt von sich, dass die Musik via Tastatur, MIDI-Controller oder Mikrophon eingegeben werden könne und dass das eine Methode sei, „[...] to enter music in a musical, rather than mechanical, manner“.<sup>226</sup> Tatsächlich wird die Musik hier visuell und per Tastatur editiert: zuerst markiert man mit der Maus und dem Cursor eine Stelle, erhält damit einen Eingabemodus und sagt dann per Tastatur, was an dieser Stelle erscheinen resp. verändert werden solle.<sup>227</sup> Dies zu verstehen und den Cursor entsprechend zu nutzen, ist die eine Hürde, die man nehmen muss, wenn man *Denemo* produktiv einsetzen will.

In der Regel wird dieses Programm von gängigen Distributionen als fertig intergriertes Paket mitgeliefert.<sup>228</sup> Seine Homepage stellt jedoch auch eigenständige Versionen für GNU/Linux und Windows bereit.<sup>229</sup> Und als GPL lizenzierte freie Software werden auch seine Quellen in einem Repository gehostet und öffentlich weiterentwickelt.<sup>230</sup>

Ein Handbuch erläutert die Nutzung von *Denemo*. Es liegt in einer Online-Version<sup>231</sup> und in einer PDF-Version vor.<sup>232</sup> Beide sind allerdings schwer zu lesen und nicht unbedingt leicht zu verstehen – ersteres aber wenigstens leichter zu durchsuchen. Daneben bietet *Denemo* noch eine FAQ-Seite an.<sup>233</sup> Ohne sich diese Anleitungen wenigstens in Grundzügen anzueignen, wird man keine Freude an dem Programm haben: sie zu lesen, – und das ist die zweite Hürde – ist allerdings selbst kaum ein Genuss.

Die dritte Hürde ist die Integration des Programms in bestehende Desktopsysteme: *Denemo* erlaubt, Teile seines Menues 'abzureißen' und als Paletten in Form eigener Fenster zu behandeln. Unglücklicherweise geht damit oft der Fensterfokus verloren. Das *Denemo*-Menue überlagert dann noch das Fenster anderer Programme, wenn diese

<sup>225)</sup> vgl. *LilyPond Development Team*: Leichter Editieren; o.J. [2018] ⇒ <http://lilypond.org/easier-editing.de.html> – RDL: 2019-02-21, wp..

<sup>226)</sup> vgl. anon. [*Denemo Team*]: Denemo [Homepage]; o.J. [2019] ⇒ <http://www.denemo.org/> – RDL: 2019-02-22, wp..

<sup>227)</sup> vgl. Shann, Richard: Denemo User Manual; 2015 ⇒ <http://git.savannah.gnu.org/gitweb/?p=denemo.git;f=docs/denemomanual.pdf;hb=HEAD> – RDL: 2019-02-22, S. 1, 5, u. 47ff.

<sup>228)</sup> vgl. z.B. anon. [*Ubuntu*]: Denemo. [Das Ubuntu Package]; o.J. [2014] ⇒ <https://wiki.ubuntuusers.de/Denemo/> – RDL: 2019-02-23, wp..

<sup>229)</sup> vgl. anon. [*Denemo Team*]: Denemo [Download]; o.J. [2019] ⇒ <http://www.denemo.org/downloads-page/> – RDL: 2019-02-22, wp..

<sup>230)</sup> vgl. anon. [*Github*]: GNU Denemo. Free and Open Music Notation Editor. [Repository]; o.J. [2019] ⇒ <https://github.com/denemo/denemo> – RDL: 2019-02-23, wp.. Das Repository enthält unter dem namen 'COPYING' die GPL-3.0-Lizenz, die *Denemo* damit als echte freie Software ausweist. Darüber hinaus ist *Denemo* sogar Teil des GNU-Projektes ( → <https://www.gnu.org/software/> <https://www.gnu.org/software/>)

<sup>231)</sup> vgl. Shann, Richard: Denemo User Manual (online); 2015 ⇒ <http://www.denemo.org/~rshann/denemo-manual.html> – RDL: 2019-02-22, wp..

<sup>232)</sup> vgl. Shann: Denemo Manual, 2015, S. 2ff.

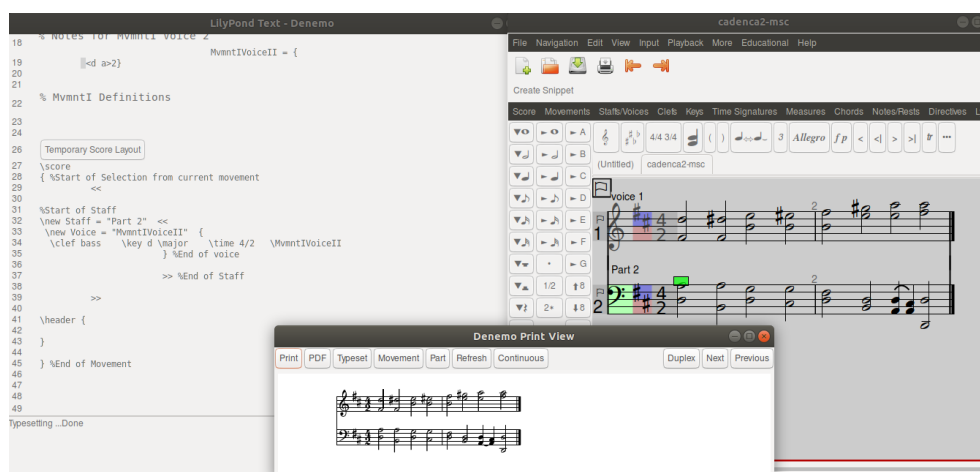
<sup>233)</sup> vgl. anon. [*Denemo Team*]: Denemo FAQ; o.J. [2019] ⇒ <http://www.denemo.org/faq/> – RDL: 2019-02-22, wp..

### 3 Frontends: die (graphische) Eingabe

längst den Fokus erhalten haben. Ein flüssiges gleichzeitiges Arbeiten mit z.B. einem L<sup>A</sup>T<sub>E</sub>X-Editor und mit *Denemo* ist – enttäuschenderweise – so nicht wirklich möglich.

*Denemo* verwendet *LilyPond* als Backend.<sup>234</sup> Trotzdem speichert es seine Dateien in erster Linie in einem eigenen XML-Format. Die entsprechenden Dateien tragen die Extension **.denemo**.<sup>235</sup> Über sein Menue bietet *Denemo* – wie es dort selbst anzeigt – einen 'begrenzten' Import von *LilyPond*- und *MIDI*-Dateien und einen uneingeschränkten Import von *MusicXML*-Dateien. Zum Export stehen – wie kaum anders zu erwarten – alle Formate bereit, die mit *LilyPond* generiert werden können, also die *LilyPond*-Dateien selbst und die entsprechende Graphik- und *MIDI*-Versionen. Außerdem bietet *Denemo* den Export als Audio-Datei an.

Tatsächlich ist es uns überhaupt nicht gelungen, die *LilyPond*-Version unser Referenzkadenz II nach *Denemo* zu importieren, die *Frescobaldi* problemlos hat nutzen können. Der Import der von *Frescobaldi* / *LilyPond* generierten *MIDI*-Datei war formal möglich, stimmte aber inhaltlich nicht. Erst der Import einer *MusicXML*-Datei ohne textuelle Analysesymbole gelang erwartungsgemäß.



Leider war es danach nicht möglich, die Symbole der Harmonyanalyse in den Notentext zu integrieren. Dass wir dazu unsere kleine Zusatzbibliothek über den *LilyPond*-Editor hätten nutzen können, hatten wir nach den ersten Erfahrungen realistischerweise schon nicht mehr angenommen. Dass dieser Editor in seinem spartanischen Design die *LilyPond*-Kodierung nur rudimentär unterstützt, hätte dabei kaum geholfen. Dass aber *Denemo* mit seinen eigenen Mitteln auch nicht dazu zu überreden war, eine vereinfachte Version der Harmonieanalyse in Form von 'Liedtext' zu erfassen, hatten wir nicht erwartet.

Insgesamt hat *Denemo* eine lange Geschichte. Das ist im Rahmen freier Software an sich ein Verdienst. Oft werden dann allerdings auch programmiertechnische Altlasten mitgeschleppt, die ein modernes Design und eine Integration in moderne Systeme behindern. Darunter scheint *Denemo* zu leiden.

<sup>234</sup> vgl. anon. [Denemo Team]: Denemo Homepage, 2019, wp..

<sup>235</sup> vgl. dazu anon. [Wikipedia]: Denemo; o.J. [2018] ⇒ <https://de.wikipedia.org/wiki/Denemo>, wp..

Für Musikwissenschaftler, die bereits mit dem Programm vertraut sind, kann es als Noteneditor dienen. Die Analysesymbole wird er aber nachgelagert einfügen müssen. Für Neueinsteiger scheinen Lern- und Arbeitsaufwand auf der einen Seiten und die Qualität der Ergebnisse – verglichen mit anderen freien Programmen – nicht (mehr) in einem angemessenen Verhältnis zu stehen. Wenn *Denemo* sich wirklich einmal zum „Ziel“ gesetzt hat, „[...] effektiv und in hoher Geschwindigkeit Notation für den LilyPond Music Engraver zu erstellen“<sup>236</sup>, dann muss man man ihm heute attestieren, dass es seinen Nutzern dabei hohe Hürden zu überwinden auferlegt hat. Wir sind daran gescheitert und geben *Denemo* deshalb drei Sterne: mehr als dem Uraltprogramm *NtEd* und weniger als *Elysium* oder gar *Frescobaldi* und *MuseScore*.

### 3.4.7 EasyABC (★★★★)

Seine Repositoryseite sagt von *EasyABC*, es sei in Python geschriebenes Notensatzprogramm, das für die graphische Darstellung auf das Kompatibilitätsframework *WxWidgets* zurückgreife.<sup>237</sup> Die letzte Version stammt vom Mai 2018.<sup>238</sup> Das Open Source Projekt pflegt außerdem eine Projekthomepage, die Installationsoptionen auflistet.<sup>239</sup> Neben diesen beiden Anlaufpunkten existiert noch die (veraltete) Homepage des ursprünglichen Programmierers.<sup>240</sup> Vom Typ her gehört *EasyABC* zu den semi-graphischen Editoren.



(Aktuelle) Distributionen werden *EasyABC* eher nicht mit anbieten.<sup>241</sup> Denn – laut *EasyABC* Installationsanleitung – gäbe es intendiert keine entsprechenden Installations- oder Binärpakete: *EasyABC* solle per Shell – über den Befehl `python easy_abc.py` – direkt aus dem heruntergeladenen Ordner mit den Softwarequellen gestartet werden, damit das Programm all seine Ressourcen finde.<sup>242</sup>

<sup>236)</sup> vgl. anon. [Wikipedia]: *Denemo*, 2018, wp..

<sup>237)</sup> vgl. anon. [SourceForge]: *EasyABC* Repository; 2017 ⇒ <https://sourceforge.net/projects/easyabc/> – RDL: 2019-02-04, wp.. Die zu startende Datei `easyabc.py` lizenziert das Programm im Header unter der GPL-3.0. Damit ist es freie Software.

<sup>238)</sup> vgl. anon. [SourceForge]: *EasyABC* Releases; 2017 ⇒ <https://sourceforge.net/projects/easyabc/files/EasyABC/> – RDL: 2019-02-04, wp..

<sup>239)</sup> vgl. anon. [SourceForge]: *EasyABC* Homepage; 2017 ⇒ <http://easyabc.sourceforge.net/> – RDL: 2019-02-04, wp..

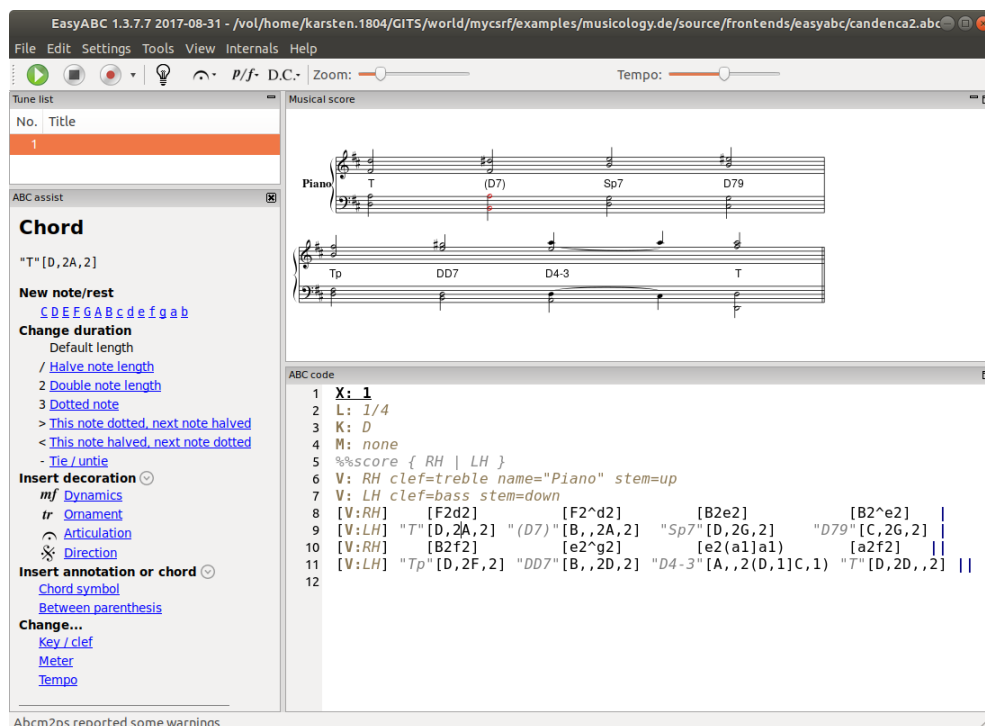
<sup>240)</sup> vgl. Liberg, Nils: *EasyABC*; o.J. [2015] ⇒ <https://www.nilsliberg.se/ksp/easyabc/> – RDL: 2019-02-04, wp..

<sup>241)</sup> Ubuntu 18.04 jedenfalls offeriert kein entsprechendes Paket.

<sup>242)</sup> Das Quellpaket enthält eine Linuxanleitung (`using_EasyABC_in_Linux.txt`) und eine Windowsanleitung (`using_EasyABC_in_Windows.txt`), die jeweils auflisten welche Zusatzpakete installiert sein müssen, bevor man *EasyABC* erfolgreich starten kann. Ubuntu 18.04 Nutzer müssen bei der Abarbeitung der Liste aufpassen: Python3 ist Distributionsstandard, während Python2.7 nur 'nebenbei' angeboten wird. Die letzte Version von *EasyABC* verlangt aber noch Python2.7 und das dazu passende wx-Paket. Deshalb gilt es bei Ubuntu 18.04, die Python3-Module so weit als möglich zu deinstallieren, bis der Befehl `python --version` eine 2.7-Version ausgibt. Danach reicht es, das Kommando `sudo apt-get install python-wxgtk-media3.0` abzusetzen. Anschließend kann man *EasyABC* erfolgreich starten.

### 3 Frontends: die (graphische) Eingabe

Nach dem Start bietet sich dem 'Notenschreiber' ein viergeteiltes Fenster an: Im unteren rechten Bereich notiert der Komponist, Arrangeur oder Musikwissenschaftler seine Noten im *ABC*-Format. Im oberen rechten Bereich wird daraus – on the fly – der entsprechende Notentext abgeleitet und angezeigt. Der obere linke Teil des Fensters listet die offenen 'Songs' auf. Und der untere linke Fensterteil enthält den Clou des Ganzen: einen – auf die Cursorposition im Notentext bezogenen – kontextsensitiven 'Next-Steps'-Bereich, in dem – direkt per Klick ausführbar – aufgelistet wird, was an der fraglichen Stelle wie modifiziert werden kann. Für unsere Referenzkadenz II, die *EasyABC* in den Grenzen des Backends erfolgreich darstellen kann, sähe das so aus:



Beim kontextsensitiven Editieren setzt man den Cursor im rechten unteren Editorfeld in einen 'Akkord' bzw. auf eine 'Note' – in unserem Beispiel Zeile 9, [D,2A,2] –, den oder die das System in der rechten oberen Visualisierung rot einfärbt. Dieser per Cursor ausgewählte Bereich wird im linken unteren Fensterteil analysiert und beschrieben, bevor darunter all die Modifikationen und Erweiterungen aufgelistet werden, die an dieser Stelle in diesem Kontext von der *ABC*-Syntax her möglich sind. *EasyABC* bietet damit eine zukunftsweisende und trickreiche 'Code-Completion' an, die die Tipparbeit wirklich zu vereinfachen vermag.

Schließlich bliebe zu erwähnen, dass man das, was man im Editorfenster eingegeben hat, als *ABC*-Code speichern oder als PDF-, MIDI, MusicXML- oder HTML-Datei exportieren kann. Außerdem bietet *EasyABC* bei gelungener MIDI-Integration die Möglichkeit, das Eingegebene auch zu hören.

In Verbindung mit den Konvertern bietet *EasyABC* den Musikwissenschaftlern als



textbasierter Editor mit automatisierter Visualisierung also ein sehr gut zu nutzendes Frontend für die *ABC*-Notationsmethode.

### 3.4.8 Elysium (★★★★)

*Elysium* fungiert als Frontend für *LilyPond*, ohne ein eigenständiger Editor zu sein. Es wird vielmehr zusammen mit *Eclipse* genutzt, einer integrierten Entwicklungsumgebung, deren Funktionalität erst über die Integration von Plugins entsteht. Entwickelt und gepflegt wird diese IDE<sup>243</sup> von der *Eclipse Foundation*.<sup>244</sup> Sie bietet die verschiedensten vorkonfigurierten Pakete an, zusammengestellt für die unterschiedlichsten Zwecke und ausgelegt auf die gängigsten Betriebssysteme. In unserem Fall reicht das einfache Standardpaket „Eclipse IDE for Java Developers“.<sup>245</sup>



Ein Plugin, das zu installieren in unserem Kontext lohnt, wäre z.B. *TeXlipse*: es macht *Eclipse* zu einem exzellenten 'Editor' für *LaTeX*-Texte.<sup>246</sup>

Das für unsere Zwecke entscheidende Plugin ist *Elysium*. Es wird aus Eclipse heraus vom „Eclipse-Marketplace“ heruntergeladen und installiert.<sup>247</sup> Sein Schöpfer nennt seine Erweiterung die „LilyPond IDE für Eclipse“.<sup>248</sup> Der Name *Elysium* verweise auf *Eclipse* und *.ly*, der Extension von LilyPond-Dateien und stehe für eine „himmlische“ Verbindung: Schließlich sei beides Open-Source-Software, wobei „[...] writing complex scores with LilyPond inevitably requires a more agile, more managed approach than a simple command line and plain text editor“. Und eben das unterstütze *Eclipse* als bewährte Entwicklungsumgebung schon von sich aus.<sup>249</sup> Dem entsprechend ist *Elysium* als freie Software quelloffen unter der *Eclipse Public License* publiziert worden<sup>250</sup>. Das System werde – wie es heißt – in vier Schritten bereitgestellt: Sofern es die eigene Distribution nicht schon mit sich bringe, installiere man zuerst auf die gewohnte Weise *LilyPond*, dann *Eclipse* und von da aus das Plugin *Elysium*.<sup>251</sup> Alle Varianten laufen bei

<sup>243</sup>) Integrated Development Environment

<sup>244</sup>) vgl. anon. [*Eclipse Foundation*]: Eclipse Foundation. The Platform for Open Innovation and Collaboration; o.J. [2018] ⇒ <https://www.eclipse.org/> – RDL: 2019-02-09, wp..

<sup>245</sup>) vgl. anon. [*Eclipse Foundation*]: Eclipse Packages; o.J. [2018] ⇒ <https://www.eclipse.org/> – RDL: 2019-02-09, wp..

<sup>246</sup>) vgl. anon. [*SourceForge*]: TeXlipse Homepage; o.J. [2019] ⇒ <http://texlipse.sourceforge.net/> – RDL: 2019-02-09, wp..

<sup>247</sup>) vgl. Harmath, Dénes: Elysium [im Eclipse Marketplace]; 2019 ⇒ <https://marketplace.eclipse.org/content/elysium> – RDL: 2019-02-09, wp.. Die Paketbeschreibung sagt, *Elysium* werde unter der *Eclipse Public License* distribuiert, einer offiziellen Open Source Software Lizenz. ( → <https://opensource.org/licenses/EPL-2.0>)

<sup>248</sup>) vgl. Harmath, Dénes: Elysium. The LilyPond IDE for Eclipse. [Homepage]; 2019 ⇒ <http://elysium.thsoft.hu/> – RDL: 2019-02-09, wp..

<sup>249</sup>) vgl. Harmath, Dénes: Elysium. The LilyPond IDE for Eclipse. [Motivation]; 2019 ⇒ <http://elysium.thsoft.hu/about/> – RDL: 2019-02-09, wp..

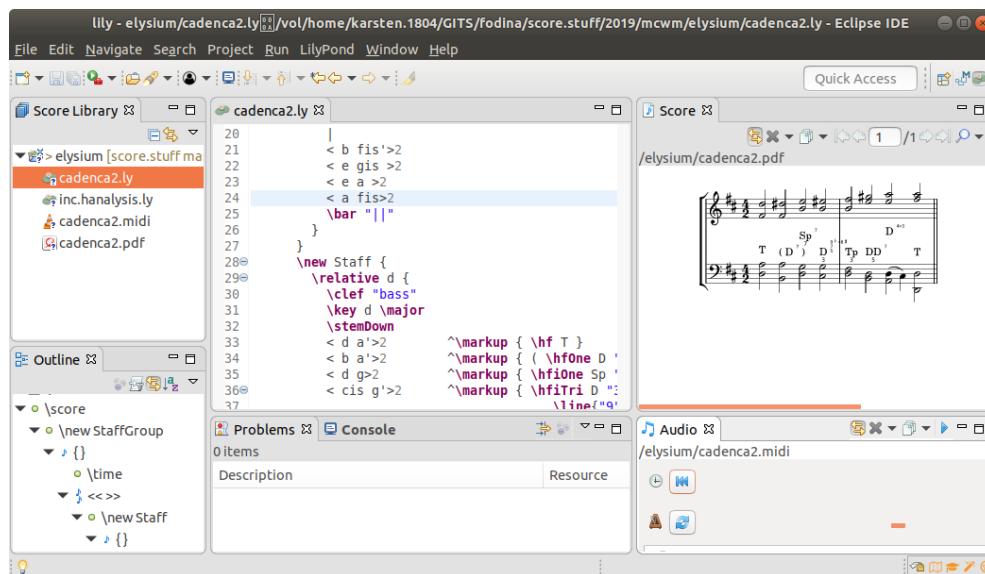
<sup>250</sup>) vgl. Harmath, Dénes (thsoft): Elysium. LilyPond IDE for Eclipse [Sources]; o.J. [2018] ⇒ <https://github.com/thSoft/elysium> – RDL: 2019-02-09, wp..

<sup>251</sup>) vgl. Harmath, Dénes: Elysium. The LilyPond IDE for Eclipse. [Installation]; 2019 ⇒ <http://elysium.thsoft.hu/> – RDL: 2019-02-09, wp..

### 3 Frontends: die (graphische) Eingabe

uns problemlos durch.

Von seinen Eigenschaften her ist *Elysium* ein semi-graphischer Editor: man gibt – Editor gestützt – den gewünschten *LilyPond*-Code ein und bei jeder Sicherung der Quelldatei werden die entsprechende MIDI- und die zugehörige PDF-Datei kompiliert und angezeigt.<sup>252</sup> Auf diese Weise kann *Elysium* auch mit unser Referenzkadenz II problemlos umgehen, als *LilyPond*-Frontend sogar inklusive unser kleinen Zusatzbibliothek:



Die *LilyPond*-Perspektive, die das *Elysium*-Plugin mitbringt, enthält in der oberen Mitte den eigentlichen *LilyPond*-Editor. Er nutzt Syntaxhighlighting und bietet – über die Eclipse-Menues – den bei *Eclipse* gewohnten Programmiersupport. Links davon findet sich – entsprechend – eine Liste der Projekte und ihrer Dateien. Und darunter wird die Struktur der Datei abgebildet, die gerade editiert wird. Auf der rechten oberen Seite erscheint dann nach jeder Sicherung des eingegebenen Codes der Notentext, der daraus erzeugt werden kann. Und der rechte untere Bereich bietet einen MIDI-Player, mit dem man sich seine generierte Musik anhören kann.

Zwei Kleinigkeiten können bei der Nutzung verwirren:

Zum ersten muss man das 'Noten'-Fenster und das *MIDI*-Player-Fenster an eine geöffnete Datei 'binden', wenn man den Notentext sehen und hören möchte. Dazu nutzt man die gelben Pfeile.

Außerdem erwartet *Elysium*, dass in der Projektpalte des Eclipsemenues die Option 'build automatically' aktiviert ist. Nur dann nämlich generiert *Elysium* unter Rückgriff auf das installierte *LilyPond*-Backend automatisch die zurhörige PDF-Datei und – sofern im Code mit `MIDI{}` aktiviert – die entsprechende MIDI-Datei, die dann im rechten oberen und unteren Bereich dargestellt werden.

<sup>252)</sup> vgl. Harmath, Dénes: Elysium. The LilyPond IDE for Eclipse. [Eigenschaften]; 2019 ⇒ <http://elysium.thsoft.hu/features> – RDL: 2019-02-09, wp..



Nun gibt es durchaus Gründe, nicht alle Projekte immer automatisch bilden zu lassen.<sup>253</sup> Will man solche Projekte mit manuellem Anstoß und die Lilypond-Projekte, die die automatische Abarbeitung fordern, gemeinsam offen halten, wird man Auto-Build-Funktionen je nach Bedarf an und abschalten müssen und die Fehlermeldungen souverän ignorieren, von denen man weiß, daß sie 'natürlich' sind. Alternativ kann man natürlich eine weitere Eclipse-Instanz mit anderem Workspace starten.

Insgesamt erhält man mit *Elysium* eine verlässlich solide Umgebung zur Entwicklung von *LilyPond* basierten Noten, die insbesondere denjenigen ans Herz wachsen wird, die eh schon mit *Eclipse* arbeiten.

### 3.4.9 Free Clef (★)

*Free Clef* bezeichnet sich selbst als „lightweight notation editor“, der es seinen Nutzern erlaube, Musik zu schreiben und im MusicXML-Format zu exportieren. Die letzte veröffentlichte Version stammt vom Juni 2008, es handelt sich allerdings noch um eine *Beta*-Version.<sup>254</sup>



Moderne Distributionen bieten keine Binärpakete für *Free Clef* an.<sup>255</sup> Der Sourcecode kann von der Projektseite heruntergeladen werden. Allerdings verlangt diese Software bei einer Installation aus den Quellen heraus die Kompatibilitätsbibliothek *wxwidgets* in der Version 2.8. Aktuell stellen besagte Distributionen aber nur die Version 3.x bereit, weil sie selbst ja bereits auf GNOME-3 und damit auf GTK-3 basieren.

Damit scheidet *Free Clef* als Notensatzsystem aktuell (noch) aus. Dass er in den Quellen vorliegt und das GNU-automake/autoconf-System benutzt, macht eine 'Wiederbelebung' nicht unmöglich. Dies ist uns – wieder einmal – wenigstens einen Stern wert.

### 3.4.10 Frescobaldi (★★★★★)

*LilyPond* 'bewirbt' *Frescobaldi* als „leichtgewichtigen“ und „mächtigen LilyPond Musik- und Texteditor mit vielen für die Arbeit mit LilyPond nützlichen Fähigkeiten“. Besonders hervorgehoben wird „die beidseitige Verknüpfung zwischen dem LilyPond Code und der dargestellten Musik“ über den „point-and-click per Maus“.<sup>256</sup> Daraus ergibt sich sofort, was *Frescobaldi* ist: ein Editor, bei dem der Komponist oder Musikwissenschaftler *LilyPond*-Code eingibt und als Feedback den visualisierten Notentext erhält: *Frescobaldi* gehört zu den semi-graphischen Frontends.



<sup>253</sup>) So schreiben wir z.B. den Text, den Sie gerade lesen, mit *Elcipse* und *Texlipse*. Gemeinsam wissen wir aber bereits, dass unser Text nicht standardmäßig prozessiert werden darf, weil ja *lilypond-book* immer zuerst ausgeführt werden muss und den eigentlichen L<sup>A</sup>T<sub>E</sub>X-Text erst generiert.

<sup>254</sup>) vgl. anon. [SourceForge]: Free Clef [Homepage]; o.J. [2008] ⇒ <https://sourceforge.net/projects/freeclef/> – RDL: 2019-02-12, wp.. Die Homepage sagt, dass Programm stünde unter der GPL-2.0 Lizenz. Damit wäre es echte freie Software.

<sup>255</sup>) Jedenfalls nicht Ubuntu 18.04.

<sup>256</sup>) vgl. *LilyPond Development Team*: LilyPond Leichteres Editieren, 2018, wp..

### 3 Frontends: die (graphische) Eingabe

Das Programm ist freie, unter GPL lizenzierte Software, dessen Code auf Github gehostet und zum Download bereitgestellt wird.<sup>257</sup> Als Applikation wird *Frescobaldi* auf einer umfangreichen Site vorgestellt<sup>258</sup>, die auch den Download und die Installation beschreibt<sup>259</sup> und ein ausführliches 'Manual' mitliefert.<sup>260</sup> Gängige Distributionen enthalten entsprechende Programmpakete, sodass die aufwendigere Installation aus den Quellen in der Regel nicht notwendig ist.<sup>261</sup>

*Frescobaldi* vermag *LilyPond*-Dateien direkt zu lesen und erlaubt den konvertierenden Import von *MusicXML*-, *MIDI*- und *ABC*-Dateien. Als Export bietet sein Menue auf den ersten Blick 'nur' die Sicherung als HTML-Seite an. Allerdings nutzt *Frescobaldi* ja das ganze *LilyPond*-Backend. Das heißt, dass aus einer *LilyPond*-Datei implizit 'immer' auch die korrespondierende *MIDI*-Datei und die entsprechende *PDF*-Datei erzeugt werden.<sup>262</sup> Initial muss das Generieren dieser Artefakte für ein geladenes oder editiertes Musikstück über das *LilyPond*-Icon angestoßen werden. Unter der Rubrik *Tools* kann man dann eine Midiplayer einblenden lassen, der das Stück – bei gelungener Soundaktivierung – hörbar macht.<sup>263</sup>

Seiner Natur nach kann *Frescobaldi* unsere Referenzkadenz II direkt aus als *LilyPond*-Datei einlesen und auch unmittelbar unsere kleine Zusatzbibliothek interpretieren, über die wir die Symbole der Harmonieanalyse in den Notentext integrieren. Sobald man einmal das *LilyPond*-Icon in der oberen Leiste angeklickt hat, werden die entsprechenden *Midi*- und die *PDF*-Dateien generiert und angezeigt:

<sup>257)</sup> vgl. anon. [Github]: *Frescobaldi* [Github Repository]; o.J. [2019] ⇒ <https://github.com/frescobaldi/frescobaldi> – RDL: 2019-02-21, wp.. Dem entsprechend enthält das Repository unter dem Dateinamen COPYING die GPL-2.0 Lizenz.

<sup>258)</sup> vgl. anon.: *Frescobaldi* [Homepage]; o.J. [2017] ⇒ <http://www.frescobaldi.org/index.html> – RDL: 2019-02-21, wp..

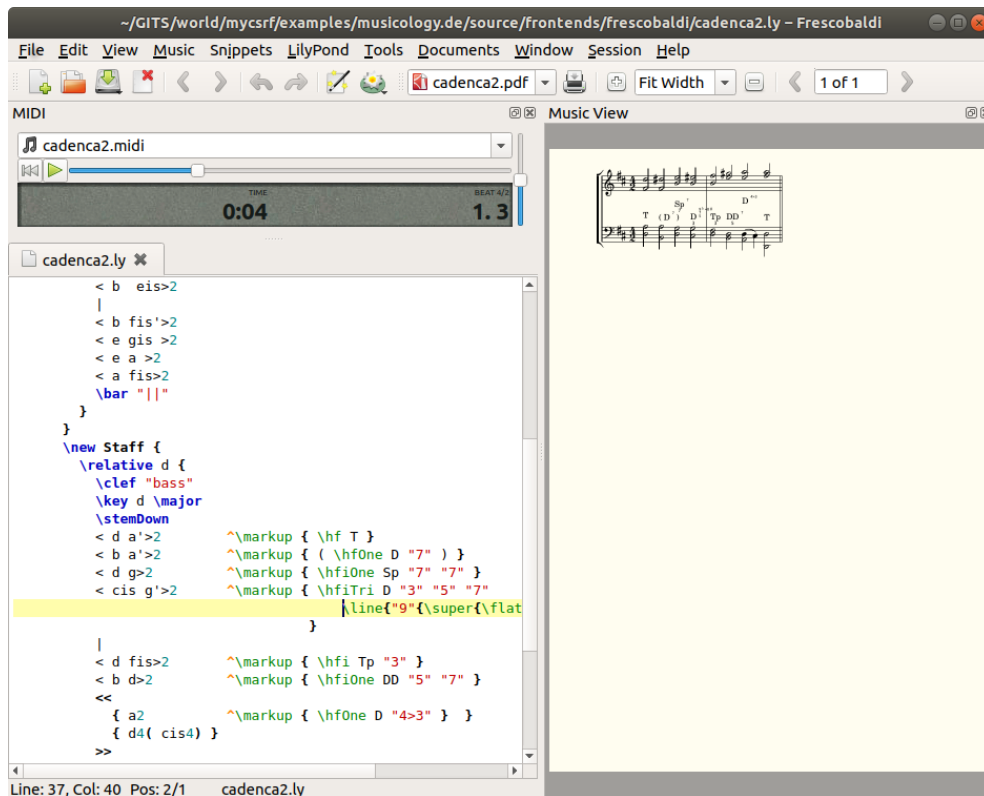
<sup>259)</sup> vgl. anon.: *Frescobaldi* Download; o.J. [2015] ⇒ <http://frescobaldi.org/download.html> – RDL: 2019-02-21, wp..

<sup>260)</sup> vgl. anon.: *Frescobaldi* Manual; o.J. [2012] ⇒ <http://frescobaldi.org/uguide.html> – RDL: 2019-02-21, wp..

<sup>261)</sup> vgl. anon. [Ubuntu]: *Frescobaldi*; o.J. [2016] ⇒ <https://wiki.ubuntuusers.de/Frescobaldi/> – RDL: 2019-02-21, wp..

<sup>262)</sup> Dies allerdings nur dann, wenn deren Bildung über die in den Code eingebetteten 'Kommandos' `\midi { }` bzw. `\layout { }` mit angestoßen werden. In diesem Fall werden die abgeleiteten Dateien unter gleichem Namenskern mit entsprechender Datei-Extension neben der `.ly`-Datei abgelegt.

<sup>263)</sup> In der Regel muss man *Frescobaldi* über den Dialog *Edit/Prefrences/Midi* noch mitteilen, über welches MIDI-Backendsystem es die Midi-Töne akkustisch generieren lassen soll. Unter GNU/Linux respektive Ubuntu kann man z.B. *timidity* über das Shell-Kommando `timidity -ia` als Server starten, dessen geöffnete Ports in dem *Frescobaldi*-Dialog angezeigt werden, wenn man *timidity* vor *Frescobaldi* gestartet hat.



Auf der linken unteren Seite editiert man den *LilyPond*-Text, unterstützt durch ein gelungenes Syntaxhighlighting und durch Menue-Befehle, die komplexere Eingaben zusammenfassen. Über diesem Eingabefenster erscheint das Frontend zum Midi-Player, wenn man diesen über die Rubrik 'Tools' aktiviert hat. Und rechts daneben wird das generierte PDF angezeigt.

Eine kleine 'Komplikation' könnte das Abspielen der Noten aus *Frescobaldi* heraus sein. Um sie aus dem Weg zu räumen, aktiviert man zuerst - wie schon beschrieben<sup>264</sup> - die Midi-Funktionalität seines Computers. Dann startet man den Midiserver mit `sudo timidity -iA -B2,8 -Os11 -s 44100`, öffnet die *Frescobaldi*-Einstellung, wechselt in den Bereich *Midi Settings*, drückt den *Refresh Button* und wählt als *Player Output* einen *TiMidity Port*.

Wenn wir *Elysium* mit vier Sternen ausgezeichnet haben, dann ist es nur fair, an *Frescobaldi* fünf Sterne zu vergeben: Beide Systeme gehören zur den semi-graphischen Editoren, beide nutzen dasselbe Backend und dasselbe Arbeitsprinzip. *Frescobaldi* unterstützt seinen Nutzer allerdings ein wenig eleganter und zielgerichteter.

### 3.4.11 Jniz (-)

Ein ganz spezielles Programm ist *Jniz*. Es bezeichnet sich selbst als „support tool“ für Komponisten, das es erlaube, mehrstimmige Stücke gemäß der klassischen Regeln (semi-

<sup>264</sup>) → S.57

### 3 Frontends: die (graphische) Eingabe

automatisch) zu harmonisieren.<sup>265</sup> Das zum Download angebotene Programm kann per `java -jar jnizpro.jar` gestartet werden. Die letzte wirklich lauffähige Version stammt aus dem Jahr 2016, die aktuellste von 2019.<sup>266</sup> Dazu gibt es ein Tutorial.<sup>267</sup>

Das Programm erlaubt keinen Import und lädt nur Dateien im eigenen Format. Kadenz-zen können in einfacher Form gebildet, allerdings nur sehr bedingt manuell gestaltet werden.<sup>268</sup> Der angebotene Export als *MusicXML*-, *LilyPond*-, *MIDI*- oder *PDF*-Datei ist darum nur begrenzt hilfreich.

Speziell ist dieses Programm nicht nur seines Ansatzes wegen, sondern auch ob seiner Lizenzierung. Hier verbietet der Autor, das Programm zu verkaufen. Und mehr noch, er untersagt auch, seine Quellen weiterzugeben: „You do not have the right to sell, distribute Jniz or use its sources under penalty of law.“<sup>269</sup> Damit unterläuft es seine 'Selbstklassifikation' als freie Software. Dass es dennoch unter Sourceforge gehostet wird, mutet merkwürdig an.<sup>270</sup> Noch bedenklicher aber ist die Tatsache, dass *Jniz* selbst unter der *GPL* lizenzierte *LilyPond*-Bibliotheken nutzt. Damit wird es zu einem abgeleiteten Werk und müsste – des *Copyleft-Effektes* wegen – ebenfalls unter der *GPL* veröffentlicht werden.<sup>271</sup>

So ist dieses Programm in unserem Kontext nicht nur funktional kaum nutzbar, seine Nutzung ist auch lizenztechnisch bedenklich. Das ist zu wenig für den einen 'Ehrenstern'.

#### 3.4.12 Laborejo (★)

*Laborejo* ist Teil einer Software Suite und sieht sich selbst als „MIDI sequencer based on classical music notation“. Es zielt hauptsächlich darauf, traditionelle Musik zu komponieren und zu produzieren. Anders als andere Notationssysteme sei *Laborejo* aber nicht dazu gedacht, Notenblätter zu erzeugen, sondern auf und mit dem Rechner zu musizieren.



Sein Autor meint, dass man das Programm aus den Quellen oder als Paket seiner je eigenen Distribution installieren könne.<sup>272</sup> Ubuntu 18.04 offeriert tatsächlich ein Paket

<sup>265)</sup> vgl. *Grandjean, Bruno*: Jniz [Homepage]; o.J. [2019] ⇒ <http://www.jniz.org/index.php/en/> – RDL: 2019-02-17, wp..

<sup>266)</sup> vgl. *anon.* [SourceForge]: Jniz [Downloadpage]; o.J. [2019] ⇒ <https://sourceforge.net/projects/jniz/files/> – RDL: 2019-02-17, wp..

<sup>267)</sup> vgl. *Grandjean, Bruno*: Jniz V2 Howto; o.J. [2019] ⇒ <http://www.jniz.org/doc/howtoV2.pdf> – RDL: 2019-02-17, wp..

<sup>268)</sup> Es ist uns nicht gelungen, die Referenzkadenz II einzugeben.

<sup>269)</sup> vgl. *Grandjean, Bruno*: Jniz [Lizenzseite]; o.J. [2019] ⇒ <http://www.jniz.org/index.php/en/license> – RDL: 2019-02-17, wp..

<sup>270)</sup> Immerhin bezeichnet sich Sourceforge als „Open Source community resource dedicated to helping open source projects“ (*rightarrow* <https://sourceforge.net/>).

<sup>271)</sup> Dies wird in den Reviews auf der Repositoriesite diskutiert (vgl. *anon.* [SourceForge]: Jniz [Projectsseite]; o.J. [2019] ⇒ <https://sourceforge.net/projects/jniz/> – RDL: 2019-02-17, wp.)

<sup>272)</sup> vgl. *Hilbricht, Nils*: Laborejo Homepage; 2019 ⇒ <https://www.laborejo.org/software.html> – RDL: 2019-02-11, wp..

*laborejo* in der Version 0.8; die vom Autor angegebene Downloadpage listet jedoch nur die anderen Pakete der Software Suite, nicht *laborejo* selbst.<sup>273</sup> Außerdem endet die spezifizierte Handbuchseite in einer 'Hello-World'-Version.<sup>274</sup>

Neben diesen *Laborejo*-Seiten gibt es noch ein *laborejo*-Github-Repository, das tatsächlich die Quellen der Software anbietet. Die entsprechende Readme-Seite sagt ganz offen, dass es im Moment kein Handbbuch und keine ordentliche Website gäbe, weil *Laborejo* ein normales Hobbyprojekt mit dem üblichen Problem sei, keine saubere Dokumentation erstellen zu können.<sup>275</sup>

Mag all dies tatsächlich einer der speziellen Situation geschuldet sein, so funktioniert doch auch die getestete Version nicht wirklich: Sie offeriert als Import nur das Format *Lisalo*, als Export u.a. die Formate *PDF*, *LilyPond* und *MIDI*. Die eigentliche Noteneingabe erfolgt menue- und tastaturbasiert, ist aber sehr unintuitiv arrangiert und ohne Handbuch nicht nutzbar. Jedenfalls ist es uns nicht gelungen, unsere Referenzkadenz II einzugeben – und eben auch nicht zu importieren. Die neuere Version von Github haben wir nicht einmal starten starten können.

Ohne dass das *Laborjo* weiter ausreift, bietet es zu dem, was wir suchen, keine Alternative, insbesondere für Musikwissenschaftler nicht: Zu flach ist die Lernkurve, zu viele 'Try-and-Error'-Erkundungen sind nötig. Mag die Grundidee auch jetzt schon anziehend sein, so muss die Umsetzung aber noch deutlich reifen. Die bloße Möglichkeit soll uns aber noch einen Stern wert sein.

### 3.4.13 MusEdit (-)

Die neuere Sichtung von Notensatzprogrammen listet *MusEdit* schon nicht mehr<sup>276</sup>, *MusicXML* erwähnt sie noch als „notation editor for Windows“, der schließlich freie Software geworden sei<sup>277</sup>. Die ursprüngliche Domain <http://www.musedit.com/> ist nicht mehr erreichbar. Die Software wird über eine Ersatzhomepage 'vertrieben'. Unten auf der Seite bekennt er Autor, dass er die Software seit 2011 nicht mehr pflegen könne. Und oben 'kündigt' er an, dass die Software 'bald' Open-Source-Software werde<sup>278</sup> – was aber offensichtlich (bisher) nicht geschehen ist.

Unabhängig davon, wie gut dieses Programm früher einmal mit *MusicXML*-Dateien umgehen konnte, heute ist es (für Musikwissenschaftler) keine Alternative mehr – und

<sup>273)</sup> vgl. Hilbricht, Nils: Laborejo Download; 2019 ⇒ <https://www.laborejo.org/downloads/> – RDL: 2019-02-11, wp..

<sup>274)</sup> vgl. Hilbricht, Nils: Laborejo Manual; 2019 ⇒ <https://www.laborejo.org/documentation/laborejo/> – RDL: 2019-02-11, wp..

<sup>275)</sup> vgl. anon. [Github]: Laborejo Github Page; o.J. [2018] ⇒ <https://github.com/diovu dau/Laborejo> – RDL: 2019-02-11, wp.. Das Repository enthält die GPL-3.0 unter dem Dateinamen LICENSE. *Laborejo* ist also freie Software.

<sup>276)</sup> vgl. anon. [Wikipedia]: Notensatzprogramme, 2019, wp..

<sup>277)</sup> vgl. anon. [MakeMusic]: MusicXML Software, 2018, wp..

<sup>278)</sup> vgl. Rogers, Doug: MusEdit [Homepage]; o.J. [2011] ⇒ <http://dougrrogers.info/websites/MusEdit/index.htm#StoryOfMusEdit> – RDL: 2019-02-03, wp..

### 3 Frontends: die (graphische) Eingabe

schon gar nicht für diejenigen, die Wert auf freie Software legen. Für den einen Ehrenstern hat uns das nicht gereicht.

#### 3.4.14 MuseScore (★★★★★)

*MuseScore* bezeichnet sich selbst als das „beliebteste Notensatzprogramm der Welt“ und verspricht „wunderschöne Notenblätter erzeugen, drucken und wiedergeben“ zu können.<sup>279</sup> Erst seit Dezember 2018 gibt es die Version 3.<sup>280</sup> LTS-Distributionen – wie Ubuntu 18.04 – werden also noch mehr oder minder lange die ältere Version 2 mitliefern.<sup>281</sup> Als GPL lizenziertes Open-Source-Software existiert selbstverständlich auch ein Repository, wo man den entsprechenden Quelltext herunterladen kann.<sup>282</sup> Für beide Releases steht je ein Online-Handbuch zur Verfügung, für *MuseScore2* ebenso<sup>283</sup>, wie für *MuseScore3*.<sup>284</sup> Die korrespondierenden PDF-Versionen stellt jedoch nicht *MuseScore* selbst zum Download bereit, sondern das *Open Source Lab* der *Oregon State University*.<sup>285</sup> Insofern ist es völlig korrekt, wenn die PDF-Version-2<sup>286</sup> und die PDF-Version-3<sup>287</sup> des *MuseScore*-Handbuches 'nur' über eine 'Fremd-URL' erreichbar sind. Neben diesen Handbüchern gibt es noch Video-Tutorials<sup>288</sup> und textuelle 'HowTos' zu speziellen Themen.<sup>289</sup> Insgesamt ist *MuseScore* damit wohl das am Besten dokumentierte freie Notensatzprogramm.



Vom Typ her gehört *MuseScore* zu den graphischen Frontends: hier setzt man Noten visuell, anstatt sie textuell zu editieren. Allerdings bringt *MuseScore* auch sein eigenes Backend mit, will sagen: seine eigene 'Notensatzmaschine'. Es hängt also nicht von

<sup>279</sup>) vgl. anon. [*MuseScore Team and Community*]: *MuseScore* [Homepage]; o.J. [2019] ⇒ <https://musescore.org/de> – RDL: 2019-02-22, wp..

<sup>280</sup>) → <https://musescore.org/en/handbook/developers-handbook/release-notes/release-notes-musescore-3>

<sup>281</sup>) vgl. anon. [*Ubuntu*]: *MuseScore*; o.J. [2018] ⇒ <https://wiki.ubuntuusers.de/MuseScore/> – RDL: 2019-02-22, wp..

<sup>282</sup>) vgl. anon. [*Github*]: *MuseScore*; o.J. [2019] ⇒ <https://github.com/musescore/MuseScore> – RDL: 2019-02-22, wp.. Das Repository enthält die GPL-2.0 unter dem Dateinamen LICENSE.GPL. Das Programm ist also freie Software.

<sup>283</sup>) vgl. anon. [*MuseScore Team and Community*]: *MuseScore* Handbuch 2; o.J. [2019] ⇒ <https://musescore.org/de/handbuch-2> – RDL: 2019-02-22, wp..

<sup>284</sup>) vgl. anon. [*MuseScore Team and Community*]: *MuseScore* Handbuch 3; o.J. [2019] ⇒ <https://musescore.org/de/handbook> – RDL: 2019-02-22, wp..

<sup>285</sup>) → <http://osuosl.org/>

<sup>286</sup>) vgl. anon. [*MuseScore Team and Community*]: *MuseScore* PDF-Handbuch 2; o.J. [2019] ⇒ <https://ftp.osuosl.org/pub/musescore/handbook/MuseScore-2.0/MuseScore-en.pdf> – RDL: 2019-02-22, wp..

<sup>287</sup>) vgl. anon. [*MuseScore Team and Community*]: *MuseScore* PDF-Handbuch 3; o.J. [2019] ⇒ <https://ftp.osuosl.org/pub/musescore-nightlies/handbook/MuseScore-3.0/MuseScore-de.pdf> – RDL: 2019-02-22, wp..

<sup>288</sup>) vgl. anon. [*MuseScore Team and Community*]: *MuseScore* Tutorials; o.J. [2019] ⇒ <https://musescore.org/de/tutorials> – RDL: 2019-02-22, wp..

<sup>289</sup>) vgl. anon. [*MuseScore Team and Community*]: *MuseScore* How To[s]; o.J. [2019] ⇒ <https://musescore.org/de/howto> – RDL: 2019-02-22, wp..



*ABC*, *LilyPond* oder *MusiX<sub>TEX</sub>* ab. Insofern darf man es in der Tat als ein integriertes Gesamtsystem bezeichnen.

Eine weitere Besonderheit bietet *MuseScore* in Sachen Download und Installation: Natürlich kann man auch die Version nutzen, die die je eigene Distribution mitbringt. Das wird von Zeit zu Zeit aber nicht die aktuellste Version sein. Diese aus den Quellen heraus zu kompilieren und zu installieren, ist ebenfalls möglich, wenn auch ungleich aufwendiger. *MuseScore* will hier eine Brücke zwischen gepflegtem System und Aktualität schlagen. Deshalb bietet es *AppImages* zum Download an.<sup>290</sup> Sie sind deutlich größer als normale Binärpakete, enthalten dafür aber in sich alle Module, Bibliotheken und Dateien, die benötigt werden. Solche *AppImages* können daher auch auf Sticks installiert und von Rechner zu Rechner getragen werden, ohne das Hostsystem erweitern zu müssen.<sup>291</sup> Das einzige, was dabei zu beachten ist, ist der Typ des Betriebssystems.<sup>292</sup>

*MuseScore* vermag u.a. *MusicXML*-, *MIDI*- und seine eigene *MSCZ*-Dateien einzulesen. Bei Fremdformaten geht damit implizit ein konvertierender Import einher. Umgekehrt kann *MuseScore* seine Musikstücke – außer in verschiedenen Audioformaten – als *MusicXML*-, *MIDI*- oder Graphikdatei sichern.<sup>293</sup> Der Im- oder Export von *ABC*-, *LilyPond*- oder *MusiX<sub>TEX</sub>*-Dateien wird nicht angeboten.<sup>294</sup> Damit scheint *MuseScore* auf den ersten Blick recht isoliert dazustehen. Es gibt allerdings eine Untersuchung vom 'Konkurrenzsysteem' *Denemo*, die die erfolgreiche Übertragung per *MIDI*- und *MusicXML*-Datei anhand eines sehr komplexen Notentextes nachweist.<sup>295</sup>

*MuseScore* kann den Musikanteil unser Referenzdatei II problemlos erfassen:

<sup>290)</sup> vgl. anon. [*MuseScore Team and Community*]: *MuseScore* [Download]; o.J. [2019] ⇒ <https://musescore.org/de/download> – RDL: 2019-02-22, wp..

<sup>291)</sup> vgl. dazu Prakash, Abhishek: *How To Use AppImage in Linux. Complete Guide*; 2018 ⇒ <https://itsfoss.com/use-appimage-linux/> – RDL: 2019-02-22, wp..

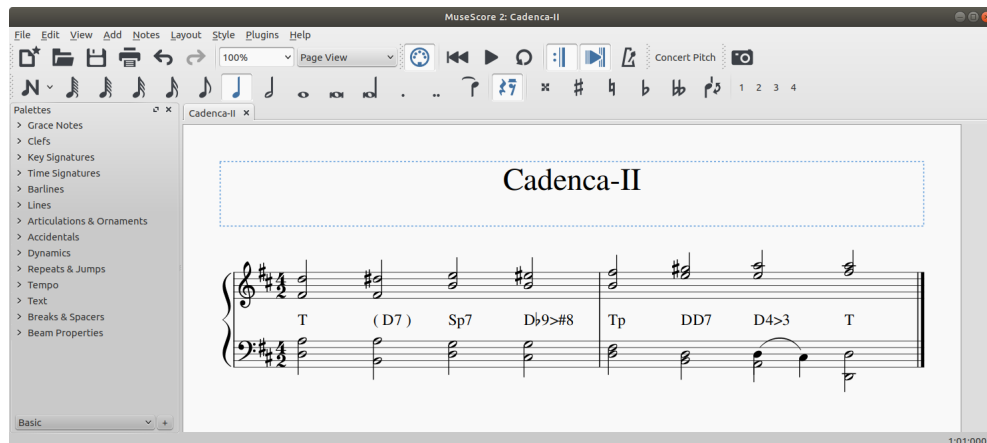
<sup>292)</sup> Die folgenden Analysen haben wir mit *MuseScore2* erstellt. Die Ergebnisse können direkt auf *MuseScore3* übertragen werden.

<sup>293)</sup> als *PDF*-, *PNG*- oder *SVG*-Datei

<sup>294)</sup> *MuseScore* bietet zwar ein *ABC Import plugin* an (→ <https://musescore.org/en/project/abc-import> RDL.: 2019-02-21). Dieses Plugin nutzt allerdings den *ABC2XML*-Onlinekonverter und liest dann 'nur' den extern generierten *MusicXML*-Text ein. Insofern handelt es sich dabei um ein Hybridsystem, das von einer funktionierenden Internetverbindung abhängig ist.

<sup>295)</sup> vgl. anon. [*Denemo Team*]: [Denemo und] *MuseScore* [Kompatibilität bei Im- und Export per *MusicXml*- bzw. *MIDI*-Datei]; o.J. [2019] ⇒ <http://denemo.org/musescore/> – RDL: 2019-02-22, wp..

### 3 Frontends: die (graphische) Eingabe



Für die Darstellung von Akkordsymbolen bietet das Programm ein eigens darauf ausgelegtes System: Man klickt zuerst eine Note in einem System an und tippt dann STR K. Daraufhin wird über der Note ein Textfeld geöffnet, in das man seine Symbole eingeben kann.  $b$  und  $bb$  bzw.  $\sharp$  und  $\sharp\sharp$  werden darin musikalisch als (doppelte) Alterationen interpretiert. Unglücklicherweise erlaubt dieses System jedoch nur die lineare Aneinanderreihung von Zeichen.<sup>296</sup> Eine zweidimensionale Darstellung komplexer Symbole, wie sie die Funktionstheorie erwartet, ist damit leider ausgeschlossen.

*MuseScore* bietet ein 'Wohlfühl'-Frontend der Extraklasse. Auch wenn man sich einarbeiten muss, findet man hier ohne große Anstrengungen, was man als Musiker sucht, nämlich die Möglichkeit, Noten visuell als Noten zu schreiben. Die optische Qualität des generierten Notentextes – am Bildschirm und in den PDF-Dateien – ist ebenfalls herausragend.

Für den Musikwissenschaftler kommt *MuseScore* trotzdem nicht als das alleinige 'Allheilmittel' in Frage, jedenfalls dann nicht, wenn er die Standardsymbole der Funktionstheorie in seiner Arbeit verwenden will. In diesem Fall wird seine Notenbeispiele zuletzt doch als *MusicXML*- oder *MIDI*-Datei exportieren und über *LilyPond* oder *MusiX<sub>TeX</sub>* um akzeptable Harmonieanalysen erweitern müssen, bevor er das Ergebnis in seine *L<sup>A</sup>T<sub>E</sub>X*-Texte einbettet. Darf er sich dagegen – aus welchen Gründen auch immer – mit den gegebenen Möglichkeiten zufriedengeben, kann er seine Ergebnisse auch als Graphiken sichern und über diese über die Standardmethode in seine Arbeiten integrieren.<sup>297</sup>

Insgesamt aber ist klar, dass *MuseScore* fünf Sterne verdient: ein ausgezeichnetes freies graphisches Notensatzprogramm.

#### 3.4.15 MuX2d (★)

*MuX2d* beschreibt sich als „WYSIWYM(ean) editor for MusiX<sub>TeX</sub>“, will sagen: als „macro package for typesetting music with TeX“, das unter der GPL veröffentlicht

<sup>296)</sup> vgl. anon. [*MuseScore Team and Community*]: *MuseScore Akkordsymbole*; o.J. [2019] ⇒ <https://musescore.org/de/handbook/akkordbezeichnungen> – RDL: 2019-02-22, wp..

<sup>297)</sup> → S. 49



werde.<sup>298</sup> Ein Open-Source-*MuSiX<sub>TE</sub>X*-Frontend weckt in unserem Kontext natürlich besonderes Interesse. Die letzte Version ist allerdings schon vor fast 20 Jahren als Release 0.2.4 erschienen.<sup>299</sup> Und sie wurde damals noch als „quite early“ bezeichnet.<sup>300</sup> Beides lässt Schlimmes erahnen.

Und in der Tat liefern gängige Distributionen keine *muX2d*-Pakete mehr. Die zu Download angebotene Binärversion verweigert mit dem Hinweis den Dienst, es könne die *libqt.so.2* nicht finden. Und die Sourcecodeversion kann nicht kompiliert werden, weil dabei eine veraltete Version der *qt*-Bibliothek gesucht wird.

Damit scheidet für Musikwissenschaftler auch *muX2D* als Editor aus, so gern wir ihn auch angeboten hätten. Man könnte dieses Programm wohl mit einigem Aufwand wiederbeleben. Angesichts der frühen Entwicklungsstadiums und des Alters mag sich das aber als nicht fruchtbringend herausstellen. Einen Stern ist uns die bloße Möglichkeit jedoch – wie immer – wert.

### 3.4.16 NoteEdit (★)

Wer nach freien Notensatzprogrammen sucht, dem wird sicher auch *NoteEdit*<sup>301</sup> vorgeschlagen. Ältere Sichtungen gehen gern auf das Programm ein<sup>302</sup>; manche mit affirmativem Ton<sup>303</sup>, manche ohne<sup>304</sup>. Neuere Sichtungen bezeichnen es als „obsolet“<sup>305</sup>. Der Quellcode des Programms ist allerdings auch heute noch erreichbar<sup>306</sup>, eine ausführbare Version wird in der Regel aber nicht mehr angeboten, weder im Netz, noch als Distributionspaket<sup>307</sup>. Gelegentlich wird erwähnt, dass spätere Entwickler von *Noteedit* schließlich zu *Canorus* gewechselt seien<sup>308</sup> und



<sup>298</sup>) vgl. anon. [SourceForge]: MuX2D [Homepage]; o.J. [2000] ⇒ <http://mux2d.sourceforge.net/> – RDL: 2019-02-14, wp..

<sup>299</sup>) vgl. anon. [SourceForge]: MuX2D [Projectpage]; o.J. [2000] ⇒ <https://sourceforge.net/projects/mux2d/> – RDL: 2019-02-14, wp.. Auch die Projektseite gibt an, dass *MuX2d* unter der GPL-2.0 distribuiert werde. Es ist also freie Software.

<sup>300</sup>) vgl. anon. [SourceForge]: MuX2D Homepage, 2000, wp..

<sup>301</sup>) vgl. Andres, Jörg: NoteEdit – Ein graphischer Noteneditor für Linux; in: LinuxUser, (2002), Nr. 02 ⇒ <http://www.linux-community.de/ausgaben/linuxuser/2002/04/noteedit-ein-graphischer-noteneditor-fuer-linux/> – RDL: 2019-01-23, wp..

<sup>302</sup>) vgl. Roitman, Alex: Editing music scores with free software; 2007 ⇒ <https://www.linux.com/news/editing-music-scores-free-software> – RDL: 2019-01-22, wp..

<sup>303</sup>) vgl. anon.: Sound & MIDI Software For Linux. Music Notation Editors; o.J. [2006] ⇒ <http://linux-sound.org/notation.html> – RDL: 2019-01-18, wp..

<sup>304</sup>) vgl. Brendel, Jens-Christoph: Notensatz unter Linux – eine Übersicht; in: Linux-Magazin, 9 (2005) ⇒ <http://www.linux-magazin.de/ausgaben/2005/09/digitale-notensteher/> – RDL: 2019-01-22, wp. – hier <http://www.linux-magazin.de/ausgaben/2005/09/digitale-notensteher/2/> und 3

<sup>305</sup>) vgl. anon. [Wikipedia]: Notensatzprogramme, 2019, wp..

<sup>306</sup>) vgl. anon. [SourceForge]: Noteedit – A Score Editor; o.J. [2014] ⇒ <https://sourceforge.net/projects/noteedit.berlios/> – RDL: 2019-01-23, wp.. Das Repository sagt, das Programm stünde unter der GPL-2.0 Lizenz. Damit wäre es freie Software.

<sup>307</sup>) jedenfalls nicht in Ubuntu 18.04.

<sup>308</sup>) → <https://wiki.ubuntuusers.de/Canorus/>

### 3 Frontends: die (graphische) Eingabe

dass *Canorus* nun der „offizielle Nachfolger“ von *Noteedit* sei<sup>309</sup>. Jedenfalls verlinkt auch sein ursprünglicher Autor *NoteEdit* in seiner Vita auf eine Homepage, die nicht mehr erreichbar ist<sup>310</sup>.

Dass der Quelltext noch zugänglich ist<sup>311</sup>, löst das Problem nicht. Zwar gehören zum Paket auch die Installationsskripte. Aber diese fordern Komponenten an, die heute ob ihres Alters nicht mehr (so einfach) zur Verfügung stehen. Eine direkte Installation aus den Quellen heraus misslingt also. So bleibt nur der Schluss, dass dieses Programm vorerst nicht mehr nutzbar ist<sup>312</sup>. Das ist insofern bedauerlich, als es eines der wenigen Programme (gewesen) sein soll, die ihren Inhalt auch als MusiX<sub>TEX</sub>-Datei exportieren konnten<sup>313</sup>.

Wir geben *NoteEdit* also 1 von 5 Sternen. Denn in der Vergangenheit hat das Programm wesentlich zum Komponieren und Arrangieren mit freier Software ermuntert.

#### 3.4.17 NtEd (★★)

Gegenwärtig trifft man im Netz unter dem Stichwort *Notensatzprogramm* auch noch auf *NtEd*. Die aktuellere Übersicht listet es ebenso<sup>314</sup>, wie die ältere: dort wird es unter dem Label „excellent music notation editor“ präsentiert.<sup>315</sup> Über Distributionen ist das Programm noch zugänglich.<sup>316</sup> Gleichwohl wird schon darauf hingewiesen, dass Sourcecode, Projektseite und Handbuch „seit November 2017“ nicht mehr erreichbar seien.<sup>317</sup> Dazu passt, dass selbst der Autor *NtEd* auf seiner 'Selbstvorstellungsseite'<sup>318</sup> auf eine *NtEd*-Homepage<sup>319</sup> verlinkt, die nicht (mehr) aufgerufen werden kann.

<sup>309)</sup> vgl. anon. [Wikipedia]: *Canorus*, 2019, wp..

<sup>310)</sup> Offensichtlich ist *Notedit* zunächst auf dem deutschen FOSS-Repository *Berlios* gehostet worden und nach dessen Einstellung nach Sourceforge migriert worden. An der Verlinkung kann man das noch erkennen. vgl. dazu *Andres, Jörg*: [Selbstbeschreibung des *NoteEdit*- und *NtEd*-Autors Jörg Andres]; o.J. ⇒ <https://vsr.informatik.tu-chemnitz.de/about/people/anders/> – RDL: 2019-01-18, wp..

<sup>311)</sup> vgl. anon. [SourceForge]: *Noteedit Repository*, 2014, wp..

<sup>312)</sup> Gleichwohl werden die Quellen samt der GNU-make-Files ausgeliefert. Damit steht einer Aktualisierung prinzipiell nichts im Wege. Es ist 'nur' eine Frage des Programmier- und Konfigurationsaufwandes, bis *NoteEdit* über die Aufrufe `autoconf automake configure --prefix=/dir make make install` wieder zur Verfügung stünde. Ob sich solch eine 'Wiederbelebung' lohnt, ist letztlich eine Frage der Lust am Programmieren, nicht der Lust am Musizieren.

<sup>313)</sup> vgl. *Roitman*: *Editing music scores with free software*, 2007, wp..

<sup>314)</sup> vgl. anon. [Wikipedia]: *Notensatzprogramme*, 2019, wp..

<sup>315)</sup> vgl. anon.: *Linux-Sound - Music Notation Editors*, wp..

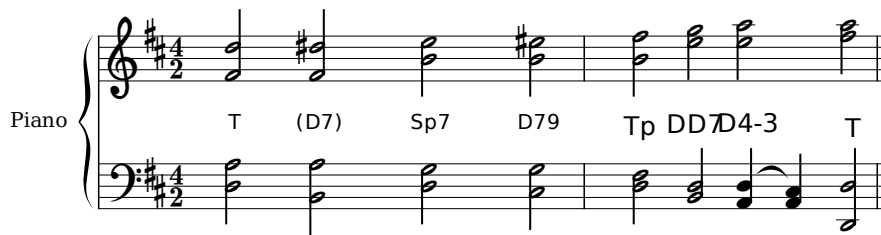
<sup>316)</sup> Unter Ubuntu 18.04 mit `sudo apt-get install nted`. Startet man das Program, bietet es einen Menüpunkt *Lizenz* an, unter dem gesagt wird, das Programm stünde unter der GPL-V2. Damit wäre es freie Software.

<sup>317)</sup> vgl. anon. [Ubuntu]: *NtEd*; o.J. [2016] ⇒ <https://wiki.ubuntuusers.de/NtEd/> – RDL: 2019-01-18, wp..

<sup>318)</sup> vgl. *Andres*: *NoteEdit und NtEd Autor*, wp..

<sup>319)</sup> vgl. *Andres, Jörg*: [NtEd Homepage]; o.J. ⇒ <https://vsr.informatik.tu-chemnitz.de/staff/jan/nted/nted.xhtml> – RDL: 2019-01-18, wp..

Unsere *Referenzkadenz II* kann mit *NtEd* durchaus erfasst werden. Das Handling ist gelegentlich sperrig<sup>320</sup> und 'buggy'.<sup>321</sup> Nicht alles, was man sehen möchte<sup>322</sup>, ist darstellbar. Das 'Druckbild' auf Bildschirm und Papier ist – verglichen mit *MusiX<sub>TeX</sub>* oder *LilyPond* – eher unschön. Den Noten kann Text in einem Lyrikmodus und einem Textmodus zugeordnet werden: Der Lyrikmodus vermag Wörter resp. Zeichen sogar vertikal auf die Noten auszurichten.<sup>323</sup> Beide Modus können in Grenzen dazu benutzt werden, Harmonieanalysesymbole zu erfassen; tiefer strukturierte Symbole sind aber nicht darstellbar.



*NtEd* erlaubt den Export im Bild<sup>324</sup>, Ton<sup>325</sup>- und *LilyPond*-Format. Der *LilyPond*-Code ist – sofern man auf textuelle Auszeichnung verzichtet – klar strukturiert, also zur Weiterverarbeitung geeignet. Unglücklicherweise scheitert die die Verifikation<sup>326</sup> der von *NtEd* exportierten *LilyPond*-Datei. Um sie prozessierbar zu machen, sie enthält mussten wir eine Klammer auskommentieren:

```
StaffAVoiceA = \relative c' {
  < fis d' >2 < fis dis' > < b e > < b f > | % 2
  < b fis' > < e g > < e a > < fis a >
  \bar "|."
}

StaffA = \new Staff \relative c' {\clef treble\key d \major \time 4/2
  <<
    \new Voice = "one" { \StaffAVoiceA }
  >>
}

StaffBVoiceA = \relative c' {
```

<sup>320)</sup> Um den Bassschlüssel zu aktivieren muss man 'entdecken', dass ein leerer Mini-Button durch die Alternativen scrollt.

<sup>321)</sup> Nach der Einstellung von 4/2 wurden die ersten Noten solange über die Schlüssel 'gedruckt', bis wir den Schlüssel noch einmal zu 4/4 und wieder zurück zu 4/2 gewechselt hatten.

<sup>322)</sup> z.B der Vorhalt in Takt 2 mit einer  $\text{♩}$  gegen 2  $\text{♩}$ .

<sup>323)</sup> Takt 1

<sup>324)</sup> PS, PNG, PDF, SVG

<sup>325)</sup> Midi

<sup>326)</sup> → S.57

### 3 Frontends: die (graphische) Eingabe

```
< d, a' >2 < b a' > < d g > < cis g' > | % 2
< d fis > < b d > < a d >4 ( < a cis > ) < d, d' >2
\bar "|."
}

StaffB = \new Staff \relative c' {\clef bass\key d \major \time 4/2
  <<
    \new Voice = "one" { \StaffBVoiceA }
  >>
}

\score {
  <<
    \new PianoStaff
    <<
      \StaffA
%    >> DIESE KLAMMER IST ZUVIEL
      \StaffB
    >>
  >>
  \layout { }
}
```

Aufs Ganze gesehen wird *NtEd* allenfalls für den Musikwissenschaftler als 'erleichterndes' *LilyPond*-Frontend in Frage kommt, der bereits mit *NtEd* arbeitet. Sich sein Handling neu anzueignen, ist wenig sinnvoll; Aufwand, Ergebnis und Wiederverwendbarkeit stehen angesichts des Entwicklungsstatus der Software nicht im Einklang. Will man als Musikwissenschaftler *NtEd* dennoch nutzen, kann man darin die Basisversion seines Notentextes editieren, diese als *accLilyPond*-Version exportieren und das Ergebnis anschließend 'manuell' mit einem Texteditor verbessern.

Wir geben dem Programm 2 von 5 Sternen, weil es historisch gesehen durchaus Verdienste hat und weil es auch heute noch funktioniert.

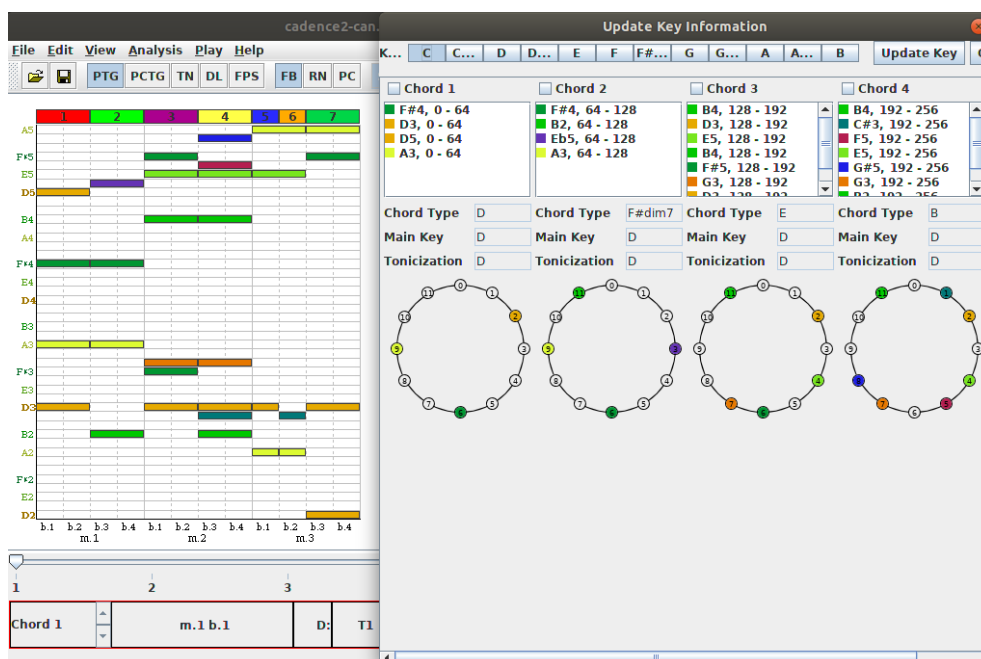
#### 3.4.18 Ptolemaic (★)

*Ptolemaic* wird auf seiner Homepage – sternenreich – als ein Programm zur Visualisierung und Untersuchung tonaler Musik offeriert, die bei der Analyse „of all types of Western music“ helfen könne und dazu etablierte analytische Techniken nutze, „[...] including tonal functional analysis (Harrison 1994) [...]“.<sup>327</sup> Das Java-Programm wird als Paket zum Download angeboten, das darin enthaltene *jar*-File kann mit einem simplen *java*

<sup>327</sup>) vgl. anon. [*Lander University*]: *Ptolemaic: A Computer Application for Music and Visualization and Analysis*. [Homepage]; o.J. [2016] ⇒ <http://ptolemaic.lander.edu/> – RDL: 2019-02-16, wp..

-jar `Ptolemaic.jar` gestartet werden.<sup>328</sup> Es liest *MusicXML*-Dateien ein und exportiert *seq*-Files.

Unsere Referenzkadenz II vermag *Ptolemaic* in der *MusicXML*-Version erfolgreich zu laden, die *Canorus* exportiert, nicht ohne allerdings das angebliche Fehlen einer „tonicization“ anzumehmen. Die *MusicXML*-Version der Referenzkadenz II, die wir deshalb aus *MuseScore* heraus 'ersatzweise' exportiert haben, kann *Ptolemaic* überhaupt nicht mehr öffnen: es scheint dabei in eine Schleife zu geraten. Nach dem Einlesen der Referenzkadenz II bietet *Ptolemaic* zwei Fenster an. Das eine enthält eine Visualisierung der Töne in Form von Balken, das andere eine daraus abgeleitete harmonische Analyse:



Auch wenn *Ptolemaic* über das Menü noch andere Analysen anbietet, generiert es keine 'Noten', die man in einem  $\text{\LaTeX}$ -Text einbetten könnte, nicht einmal als Bild. Außerdem erlaubt es vorderhand nicht, seine Analyseergebnisse textuell zu exportieren. Von daher bietet *Ptolemaic* hier wirklich keine Unterstützung – weder so, noch so.<sup>329</sup>

<sup>328)</sup> vgl. anon. [Lander University]: Ptolemaic: A Computer Application for Music and Visualization and Analysis. [Download]; o.J. [2016]  $\Rightarrow$  <http://ptolemaic.lander.edu/download-RDL:2019-02-16,wp..> Unglücklicherweise ist die Lizenzierung unklar.

<sup>329)</sup> Dass wir *Ptolemaic* deshalb nur einen Stern geben, ist in gewisser Hinsicht ungerecht: Man kann diesem Programm kaum wirklich vorwerfen, unsere Ziele nur mangelhaft umgesetzt zu haben. Es ist ja nur dadurch in unseren Blick geraten, dass es in der MusicXML-Softwareseite gelistet wird (vgl. anon. [MakeMusic]: MusicXML Software, 2018, wp.). *Ptolemaic* selbst ist gar nicht angetreten, den Notensatz zu unterstützen. Und da es das tut, was es wirklich tun will, wären 3 Sterne eigentlich angemessen. Wir belassen es trotzdem bei einem, weil es gelegentlich abstürzt, Funktionen nicht ausführt und nicht bekennet, ob es nun Open-Source-Software ist oder nicht.

### 3.4.19 Rosegarden (★★★)

*Rosegarden* versteht sich als Kompositionsumgebung, die um einen 'MIDI Sequencer' herum aufgebaut worden sei und dabei auch als Notensatz- und digitales Audiosystem fungiere.<sup>330</sup> Als „MIDI and audio sequencer“ und „musical notation editor“ wolle es „das Tool der Wahl“ für jene sein, die es vorziehen, mittels Noten zu arbeiten.<sup>331</sup>



*„Rosegarden allows you to record, arrange, and compose music, in the shape of traditional score or MIDI data, or of audio files either imported or recorded from a microphone, guitar or whatever audio source you care to specify.“*<sup>332</sup>

Man könne mit *Rosegarden* Musik schreiben, editieren oder komponieren, diese synthetisieren, mit Effekten anreichern oder abmischen, um sie schließlich auf CD zu brennen. Und nicht zuletzt biete *Rosegarden* eben den „[...] well-rounded notation editing support for high quality printed output via LilyPond“.<sup>333</sup>

Stellt man dem die These zur Seite, dass der „Kern eines Sequenzers [...] die Speicherung und Übermittlung einer Partitur an einen Tonerzeuger (sei)“, die „[...] in einem maschinenlesbaren Format (vorliege)“, und dass diese Partitur „[...] Tonhöhe, Tondauer und ggf. weitere Aspekte der wiederzugebenden Noten einer oder mehrerer Stimmen in ihrer zeitlichen Reihenfolge an ein Gerät (weitergäbe), das entsprechende Töne (erzeuge)“<sup>334</sup>, dann gewinnt man damit eine recht genaue Vorstellung von dem, was *Rosegarden* leisten will: Es erlaubt den Import von *MIDI*-Aufnahmen, bietet die Möglichkeit, diese zu korrigieren, zu modifizieren, zu arrangieren und wieder abzuspielen<sup>335</sup>. Ein Seiteneffekt des Angebots ist, dass diese Modifikationen auch visuell über einen Noteneditor erfolgen können.<sup>336</sup>

Als Open-Source-Software wird der Quelltext von *Rosegarden* öffentlich gehostet und weiterentwickelt.<sup>337</sup> Die Dokumentation wird als Wiki gepflegt<sup>338</sup> und auch in themati-

<sup>330)</sup> vgl. [Rosegarden Development Team]: Rosegarden [Homepage]; o.J. [2019] ⇒ <https://www.rosegardenmusic.com/> – RDL: 2019-02-24, wp..

<sup>331)</sup> Im Original: „to serve as the sequencer of choice for users who prefer to work with music notation“ (vgl. Cannam, Chris et al.: The Rosegarden Manual; o.J. [2019] ⇒ <https://www.rosegardenmusic.com/wiki/doc:manual-en> – RDL: 2019-02-24, wp.)

<sup>332)</sup> vgl. ds., ebda.

<sup>333)</sup> vgl. ds., ebda.

<sup>334)</sup> vgl. [Wikipedia]: Sequencer; o.J. [2018] ⇒ [https://de.wikipedia.org/wiki/Sequencer\\_\(Musik\)](https://de.wikipedia.org/wiki/Sequencer_(Musik)) – RDL: 2019-02-24, wp..

<sup>335)</sup> vgl. Cannam et al.: Rosegarden Manual, 2019, wp..

<sup>336)</sup> vgl. Cannam, Chris et al.: The Rosegarden Manual. Notationeditor; o.J. [2019] ⇒ <https://www.rosegardenmusic.com/wiki/doc:notation-en> – RDL: 2019-02-24, wp..

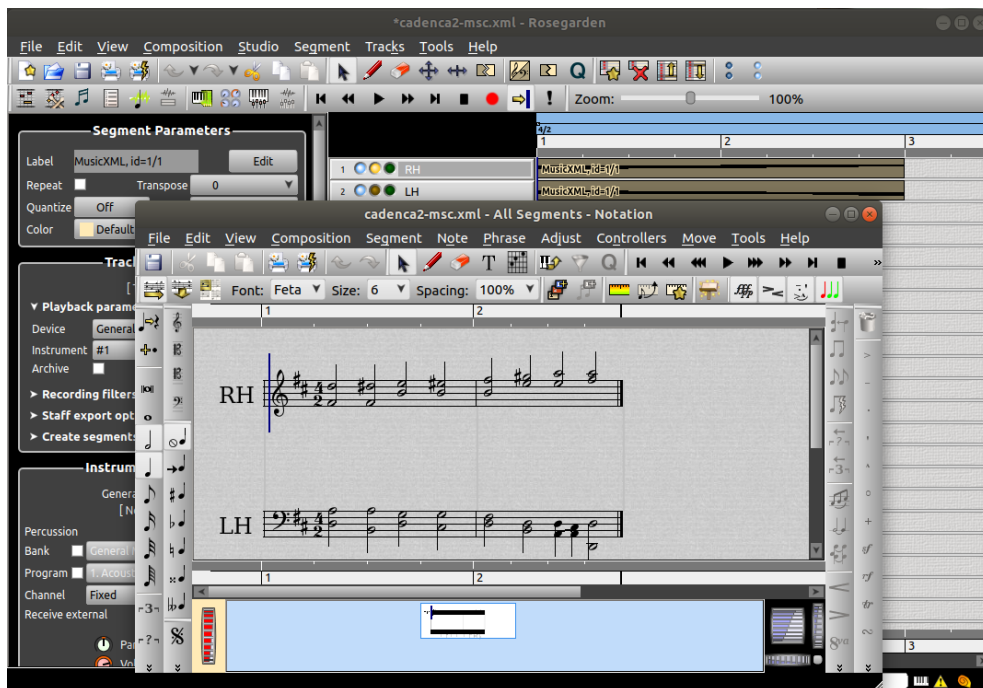
<sup>337)</sup> vgl. [SourceForge]: Rosegarden [Repository]; o.J. [2019] ⇒ <https://sourceforge.net/projects/rosegarden/> – RDL: 2019-02-24, wp.. Die Projektseite gibt an, das Programm werde unter der GPL-2.0 Lizenz distribuiert. Damit ist *LilyPond* freie Software.

<sup>338)</sup> vgl. Cannam et al.: Rosegarden Manual, 2019, wp..

schen Teilbereichen bereitgestellt<sup>339</sup>. Daneben gibt es noch spezielle Tutorials<sup>340</sup>, die ausgehend von einem bestimmten Aspekt in die Nutzung von *Rosegarden* einführen.<sup>341</sup>

Vom Format her erlaubt es *Rosegarden*, außer dem eigenen Dateityp auch *MIDI*- und *MusicXML*-Dateien zu öffnen bzw. zu importieren. Beim Export wird u.a. noch das *LilyPond*-Format angeboten. Startet man einen *MIDI*-Server – etwa *timidity*<sup>342</sup> –, bevor man *Rosegarden* aufruft, können die geladenen Noten außerdem erfolgreich abgespielt werden.

Unsere Referenzkadenz II vermag *Rosegarden* problemlos als *MusicXML*-Datei zu lesen, sofern diese keinen Text, also keine Harmonisierungssymbole enthält. Selektiert man beide Tracks und ruft über das Menue die Notenvisualisierung auf<sup>343</sup>, wird der Notentext auf dem Bildschirm angezeigt:



Das Editieren des Notentextes ist gewöhnungsbedürftig, aber nicht unmöglich. *LilyPond*-Code kann nicht direkt eingegeben werden. Damit steht unsere kleine Zusatzbibliothek für die Harmonieanalyse nicht zu Verfügung, allenfalls die Umfunktionierung der normalen Liedtextfunktionalität. Wenn man unsere Referenzkadenz als *LilyPond*-Code exportiert, kann man das Ergebnis ohne Abstriche z.B. in *Frescobaldi* wieder einlesen.

<sup>339)</sup> vgl. Cannam et al.: *Rosegarden Notationeditor*, 2019, wp..

<sup>340)</sup> vgl. [Rosegarden Development Team]: *Learning Rosegarden*. Rosegarden Tutorials; o.J. [2019] ⇒ <https://www.rosegardenmusic.com/> – RDL: 2019-02-24, wp..

<sup>341)</sup> vgl. McIntyre, D. Michael: *Learning Rosegarden. Piano Notation by Example*; 2008 ⇒ <https://www.rosegardenmusic.com/tutorials/supplemental/piano/index.html> – RDL: 2019-02-24, wp..

<sup>342)</sup> über eine Shell per `timidity -iA`

<sup>343)</sup> `Select/Edit With/Open in Notation Editor`



### 3 Frontends: die (graphische) Eingabe

Für die diejenigen, die ihre Musik eher über *MIDI*-Kanäle eingeben wollen, ist *Rosegarden* ein hervorragendes Frontend, ist doch seine optische Erscheinung auf die spurenorientierte Bearbeitung von Musik ausgelegt. Die Funktion zum Notensatz wirkt da etwas aufgesetzt. Für einen eher notenorientierten Musikwissenschaftler wird *Rosegarden* nicht wirklich das Mittel der Wahl darstellen, auch wenn es das, was es leisten will, ausgezeichnet tut. Insofern geben wir *Rosegarden* – nur aus unserem Kontext heraus – drei Sterne.

## 3.5 Konverter II

Vor der großen Verknüpfung vom Backend und Frontend müssen wir nun noch eruieren, welche der üblicherweise genannt Konverter wir wirklich woher beziehen können:

### 3.5.1 *abc2ly*

*abc2ly* wird von *Lilypond* im Paket als Tool mitgeliefert und kann nach dessen Installation über die Kommandozeile aufgerufen werden.<sup>344</sup>

### 3.5.2 *abc2mtex*

*abc2mtex* ist 'nur' noch als CTAN-Paket erhältlich<sup>345</sup>. Der auf der Paketseite integrierte Link auf die eigentliche Homepage ist nicht mehr aktuell, anders der Link auf die Quellen und der Link auf das Handbuch: Folgt man ersterem, erhält man Zugriff auf das *tar.gz*-Quellpaket; letzterer macht den *Userguide* zugänglich. Ein Binärpaket gibt es allerdings nicht (mehr). Das downloadbare Quellenpaket enthält jedoch ein Makefile, über das das eigentliche Program in Verbindung mit *gcc* und *make* umstandslos erzeugt werden kann.

### 3.5.3 *ly2abc*

Der Konverter *ly2abc* wird als Github-Projekt entwickelt und distribuiert. Auf der entsprechenden Homepage wird allerdings gesagt, es werde daran noch stark gearbeitet und es sei kein Support für Mehrstimmigkeit geplant.<sup>346</sup> Damit darf dieser Konverter getrost ignoriert werden: Zur Verwendung von Harmonieanalysen in und mit  $\text{\LaTeX}$  kann er – unabhängig von seiner sonstigen Relevanz – nichts beitragen.

---

<sup>344</sup>) → <http://lilypond.org/doc/v2.18/Documentation/usage/invoking-abc2ly>

<sup>345</sup>) vgl. *[CTAN]*: *abc2mtex* – Notate tunes stored in ABC notation; o.J. [2019] ⇒ <https://ctan.org/pkg/abc2mtex?lang=de> – RDL: 2019-10-01, wp..

<sup>346</sup>) vgl. *anon.*: *ly2abc* [Projektseite]; o.J. [2019] ⇒ <https://github.com/aekiss/ly2abc> – RDL: 2019-10-03, wp..



### 3.5.4 musicxml2ly

*musicxml2ly* wird von *Lilypond* im Paket als Tool mitgeliefert und kann nach dessen Installation über die Kommandozeile aufgerufen werden.<sup>347</sup>

### 3.5.5 mxml2abc

Sucht man nach *mxml2abc*, führt das Netz einen zunächst in Sackgassen. Es sagt, der Konverter *mxml2abc* sei als unabhängiges Projekt angelegt; das Tool solle ein Java-Programm sein.<sup>348</sup> Und es zeigt noch alte Updateankündigungen an.<sup>349</sup> Allerdings ist die angegebene Homepage von *mxml2abc*, selbst nicht mehr erreichbar.<sup>350</sup>

Schließlich führt es den hartnäckigen Sucher aber auch zu dem Konverter *mxml2abc* von Stephen Merrony<sup>351</sup>. Das dort angebotene Java-Programm kann per `java -jar mxml2abc-stephen-merrony.jar musicxml.xml` aufgerufen werden und schreibt den konvertierten Inhalt der Datei *musicxml.xml* auf die Konsole zurück.

### 3.5.6 xml2abc

Der Konverter *xml2ab* wird mit *EasyABC* mitgeliefert. Zu starten ist er im Sourcecode-Ordner per `python xml2abc musicxml.xml`, nimmt dann die auf der Kommandozeile übergebene MusicXML-Datei und konvertiert deren Inhalt in *abc*-Code.

### 3.5.7 xml2ly

Der Konverter *xml2ly* wird gesondert bereitgestellt, und zwar in Form einer *xsl*-Datei.<sup>352</sup> Um die als Konverter zu nutzen, muss man einen zusätzlichen xslt-Prozessor installiert haben. Unglücklicherweise verweist der Link auf der Homepage, der in den Downloadbereich führen soll, in einen leeren Raum. Auch der Link, der das ausgewiesene Downloadpaket direkt zugänglich machen soll, funktioniert nicht. Hier scheint einiges veraltet zu sein. Das passt zum Erscheinungsjahr (2002) und zur Releasenummer (0.0.34) der 'letzten' Veröffentlichung.<sup>353</sup>

Damit kann dieser Konverter nicht ausprobiert und also nicht wirklich als Option in Rechnung gestellt werden.

<sup>347)</sup> → <http://lilypond.org/doc/v2.18/Documentation/usage/invoking-musicxml2ly>

<sup>348)</sup> → <http://freshmeat.sourceforge.net/projects/mxml2abc> RDL 2019-04-11

<sup>349)</sup> → <https://musescore.org/en/node/15698> RDL 2019-04-11

<sup>350)</sup> → <https://code.google.com/p/mxml2abc/> RDL 2019-04-11

<sup>351)</sup> → <https://www.softpaz.com/software/download-mxml2abc-windows-42792.htm>

<sup>352)</sup> vgl. *anon.*: *xml2ly* [Homepage]; o.J. [2019] ⇒ <http://www.nongnu.org/xml2ly/> – RDL: 2019-10-03, wp..

<sup>353)</sup> Allerdings scheint der *xml2ly*-Code – der Idee von Opensource-Software entsprechend – später in einer Bibliothek weiterverwendet worden sein. → <https://github.com/grame-cncm/libmusicxml/releases>

### 3.5.8 xml2pmx

Der Konverter *xml2pmx* wird als Paket über das Icking-Music-Archive bereitgestellt<sup>354</sup>, das auch auf seiner Softwareseite selbst auf den Konverter verweist.<sup>355</sup> Dieser kann als Windows- oder Linuxpaket heruntergeladen werden. Es enthält dann die entsprechende Binärversion und eine Aufrufskript, das ein Beispiel konvertiert. Die eigenen Umformungen können dann entsprechend umgesetzt werden.

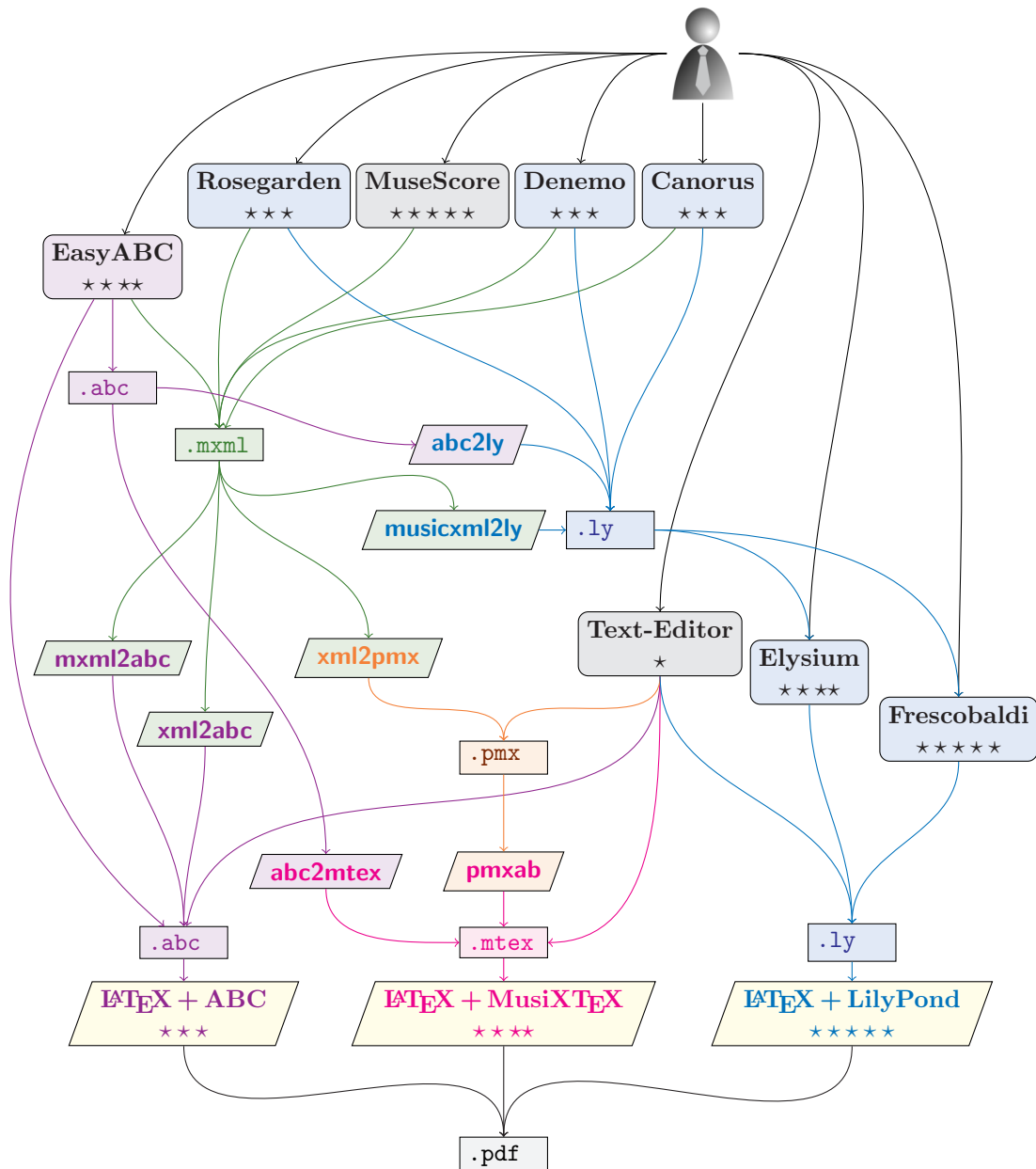
---

<sup>354)</sup> vgl. *anon.*: xml2pmx Download Area; o.J. [2019] ⇒ <https://www.icking-music-archive.org/software/xml2pmx/> – RDL: 2019-10-03, wp..

<sup>355)</sup> → <https://icking-music-archive.org/software/htdocs/index.html>

## 4 Vom Frontend nach $\text{\LaTeX}$ : die Toolketten

Verknüpfen wir die bisher gewonnenen Informationen, gewinnen wir das folgende Bild:



Wir kennen demnach vier echte graphische Frontends – *Rosegarden*, *MuseScore*, *Denemo* und *Canorus* –, drei semi-graphische Frontends – *EasyABC*, *Elysium* und *Frescobaldi* –,

#### 4 Vom Frontend nach L<sup>A</sup>T<sub>E</sub>X: die Toolketten

drei Backendsysteme – L<sup>A</sup>T<sub>E</sub>X + ABC, L<sup>A</sup>T<sub>E</sub>X + MusiX<sub>T</sub>E<sub>X</sub> und L<sup>A</sup>T<sub>E</sub>X + LilyPond – und einige Konverter, mit denen wir die einen mit den anderen verbinden können.

Wollten wir die Möglichkeiten systematisch evaluieren, müssten wir viele Wege ausprobieren und bewerten, und zwar unabhängig davon, dass zuletzt nur wenige davon wirklich hervorstechen werden. Glücklicherweise können wir den Aufwand heuristisch reduzieren:

Als Musikwissenschaftler möchten wir Harmonieanalysen in der vollen Komplexität der Funktionstheorie in unsere Notentexte und mit diesen in unsere L<sup>A</sup>T<sub>E</sub>X-Texte einbetten. Das war und ist unser Antrieb. Und die Möglichkeit zu solch einer Darstellung bieten uns nur L<sup>A</sup>T<sub>E</sub>X + MusiX<sub>T</sub>E<sub>X</sub> + harmony und L<sup>A</sup>T<sub>E</sub>X + LilyPond + harmonyli.ly.

Für LilyPond + harmonyli.ly steht uns Frescobaldi als bester semi-graphischer Editor zur Verfügung. Daneben gibt es das gleichermaßen nutzbare Eclipseplugin Elysium. Den Weg von Frescobaldi nach LilyPond hatten wir implizit bereits erfolgreich getestet, als wir Frescobaldi vorgestellt haben, nutzt er doch direkt das Backend LilyPond, um die Eingaben zu visualisieren.

Den direkten Weg von Elysium nach LilyPond haben wir bei der Erstellung dieses Dokuments erfolgreich verifiziert. Auf eine explizite Dokumentation verzichten wir, weil auch Elysium als Editor zur Visualisierung dieselbe Technik wie Frescobaldi nutzt, nämlich die Bordmittel von LilyPond.

Obwohl wir das Backendsystem ABC von seiner eigenen Funktionalität her für unsere Zwecke ausschließen dürfen – es erlaubt ja nur die Eingabe sehr einfach strukturierter Funktionssymbole –, könnte der semi-graphische Editor EasyABC trotzdem noch eine Rolle spielen: Zum einen gibt es den Konverter abc2ly, der das native ABC-Format von EasyABC konvertieren können will. Und zum anderen erlaubt EasyABC den Export in das Format musicxml. Damit könnte der Konverter musicxml2ly den Weg in die LilyPond-Welt öffnen.

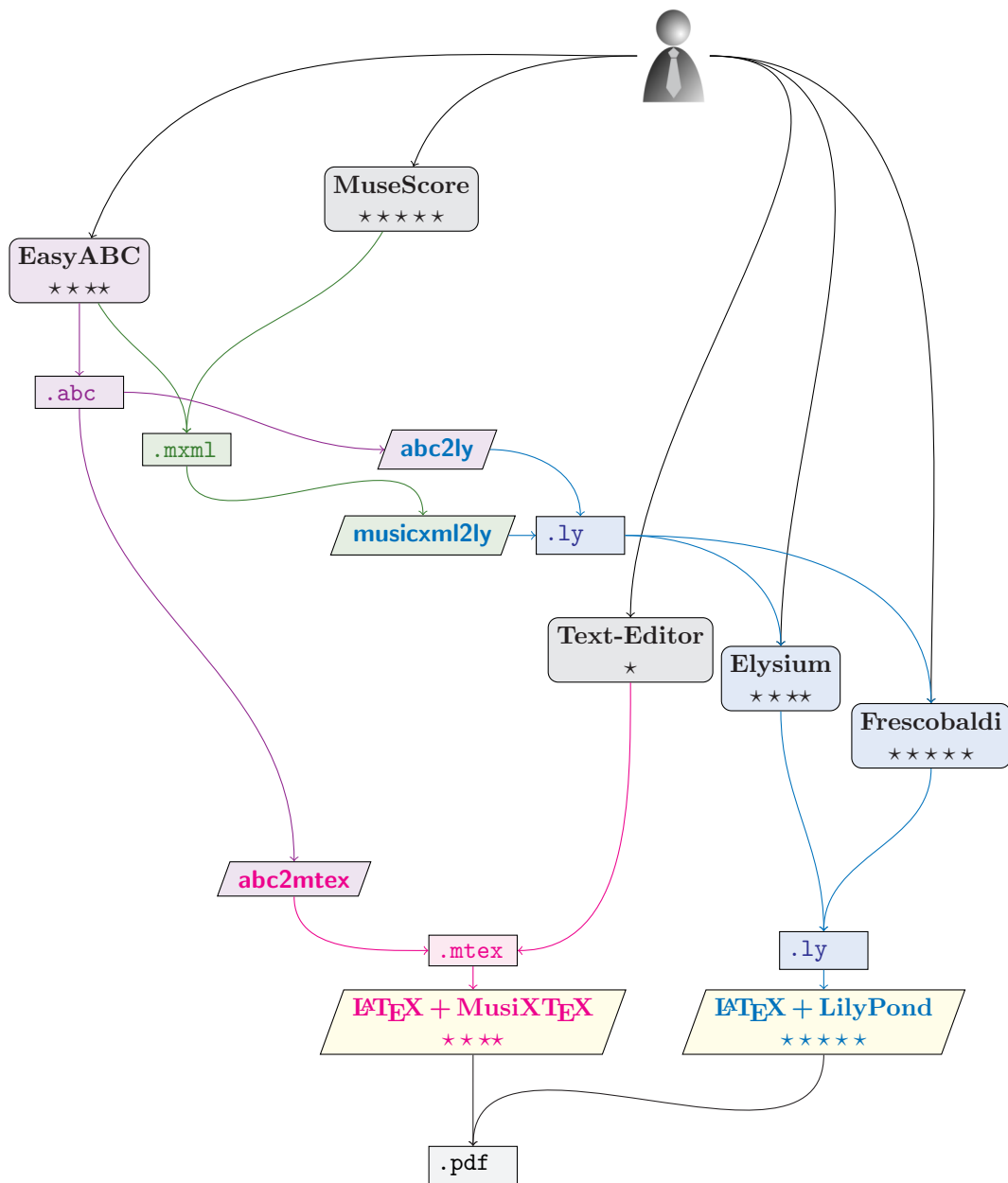
Wollten wir LilyPond stattdessen 'mit' echten einem graphischen Editor benutzen, müssten wir überprüfen, ob der MusicXML-Export von MuseScore mittels des Konverters musicxml2ly für Frescobaldi respektive Elysium 'importierbar' wird.<sup>356</sup>

Für MusiX<sub>T</sub>E<sub>X</sub> + harmony stehen uns genuin nur (L<sup>A</sup>T<sub>E</sub>X)-Editoren zur Verfügung. Wollten wir also dieses Backendsystem 'mit' einem (semi-)graphischen Editor knüpfen, zeigt uns die Wegekarte, dass nur EasyABC als Frontend in Frage käme und dass wir dafür testen müssten, ob der Output von EasyABC über den Konverter abcmex zu L<sup>A</sup>T<sub>E</sub>X-kompatiblen MusiX<sub>T</sub>E<sub>X</sub>-Code wird.

Diese Überlegungen reduzieren unsere 'Karte der Möglichkeiten' deutlich:

---

<sup>356)</sup> Dass wir Frescobaldi bzw. Elysium immer noch zwischenschalten müssen, folgt aus der Tatsache, dass MuseScore uns nur die Möglichkeit bietet, Basissymbole in den Notentext einzubetten (etwa über Add/Text/ChordSymbol oder Add/Text/Lyrics), nicht aber komplexe Harmoniesymbole mit hoch und tief gestellten Zahlen.



Wir können die Liste der damit anstehenden Evaluationen insofern noch vereinfachen, als wir bereits wissen, dass *Elysium* und *Frescobaldi* ihre *LilyPond-Input-Datei* jeweils mit den *LilyPond* genuinen Techniken auswerten. Was das eine Frontend zu laden vermag, vermag also auch das andere zu verstehen. Damit entsteht folgende Aufgabentabelle<sup>357</sup>:

<sup>357)</sup> Wir werden die Wege nach folgender Systematik evaluieren: '++' möge für eine reibungslose Nutzung stehen, '+' für eine Nutzung mit zusätzlichem Integrationsaufwand und '-' für gescheiterte Nutzungsversuche. Für die direkten Wege *Frescobaldi* resp. *Elysium* → *LilyPond* können wir das Ergebnis '++' gleich so eintragen, weil wir deren Nutzung ja schon vorgeführt resp. sinnfällig gemacht haben.

#### 4 Vom Frontend nach $\text{\LaTeX}$ : die Toolketten

Weg	VON		ÜBER	NACH		=
	Frontend	★	Konvertierung	$\text{\LaTeX}$ +	★	
01	EasyABC	4	$\rightarrow \text{abc2mtex} \rightarrow$	Musix $\text{\TeX}$	4	
02/03	EasyABC	4	$\rightarrow \text{abc2ly} \rightarrow \text{Frescobaldi/Elysium} \rightarrow$	LilyPond	5	
04/05	EasyABC	4	$\rightarrow \text{musicxml2ly} \rightarrow \text{Frescobaldi/Elysium} \rightarrow$	LilyPond	5	
06/07	MuseScore	5	$\rightarrow \text{musicxml2ly} \rightarrow \text{Frescobaldi/Elysium} \rightarrow$	LilyPond	5	
08	Frescobaldi	5	$\rightarrow$	LilyPond	5	++
09	Elysium	4	$\rightarrow$	LilyPond	5	++

Wir haben unter [...]/[musicology.de/chain-evaluation/](https://musicology.de/chain-evaluation/) in den Quellen zu diesem Dokument<sup>358</sup> für jede Variante einen wegspezifischen Ordner angelegt, in denen die Inputdatei für das jeweilige Frontend und alle abgeleiteten Zwischendateien beieinander abgelegt sind. Zudem enthält der Ordner [...]/[musicology.de/chain-evaluation/](https://musicology.de/chain-evaluation/) einen Subordner `tools`, in dem zwecks Absicherung der Reproduzierbarkeit die Quellen aller Konverter als Paket abgelegt sein.

### 4.1 w01: EasyABC $\rightarrow$ abc2mtex $\rightarrow$ $\text{\LaTeX}$ +Musix $\text{\TeX}$

Entpackt man das Paket `abc2mtex1.6.1.tar.gz` aus dem genannten Toolverzeichnis, kann man – mit entsprechender Pfadangabe – mittels `$PATH/abc2mtex` den Konverter mit der nativen ABC-Datei aufrufen, die wir mit *EasyABC* generiert und unter dem weg-spezifischen Subordner `w01-easyabc-abc2mtex` abgelegt haben:

```
$PATHX/abc2mtex $PATHY/easyabc-cadenca3.abc
```

Leider bricht das Programm mit einer Fehlermeldung ab:

```
error in input file easyabc-cadenca3.abc: line no. 10 - V field
not allowed in tune body
```

Das ist 'leicht' erklärbar: Das ABC-Format war ursprünglich ein Notationsverfahren für einstimmige 'Lieder'.<sup>359</sup> Es ist erst später zu einem 'patitur-fähigen' Format erweitert worden. Der Konverter *abc2mtex* versteht nun offensichtlich nur das alte Format.

Daraus folgt nun leider, dass uns offensichtlich kein Verfahren zur Verfügung steht, um Notentexte im *Music $\text{\TeX}$* -Format über ein (semi-) graphischs Frontend zu erfassen, und zwar auch dann nicht, wenn das Zielformat erst über einen Konverter erzeugt werden darf.

<sup>358)</sup> vgl. [Reincke: musicology.de](https://musicology.de), 2019, wp.

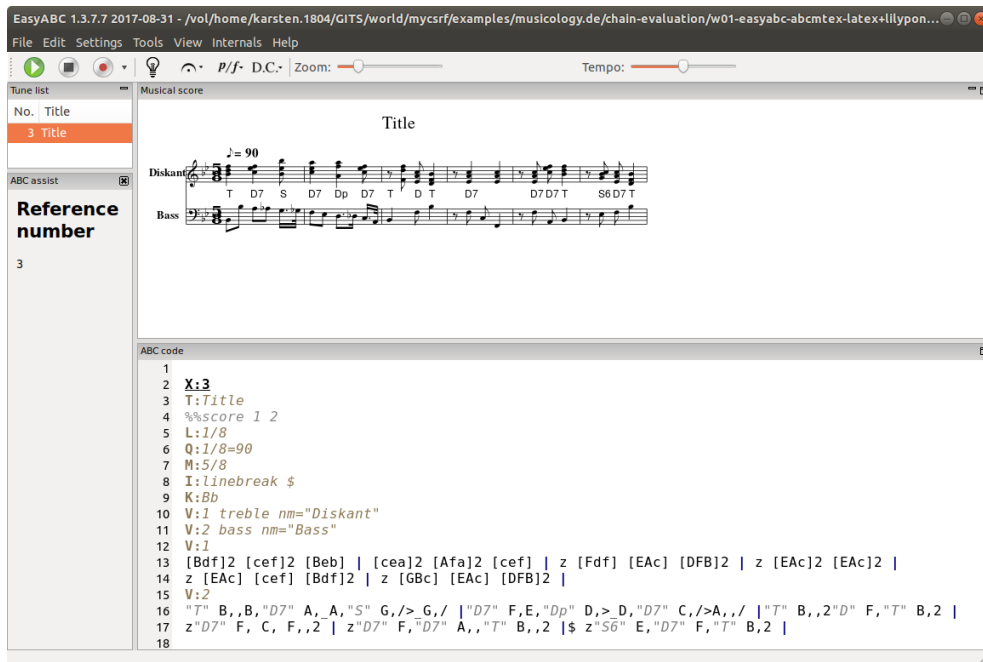
<sup>359)</sup>  $\rightarrow$  S. 18

4.2 w02/w03: *EasyABC* → *abc2ly* → *Frescobaldi/Elysium* → ...

## 4.2 w02/w03: *EasyABC* → *abc2ly* → *Frescobaldi/Elysium* →

...

In dem weg-spezifischen Subordner `w02w03-easyabc-abc2ly` haben wir dieselbe Ausgangsdatei `easyabc-cadenca3.abc` abgelegt, wie wir sie in und mit *EasyABC* generiert hatten:

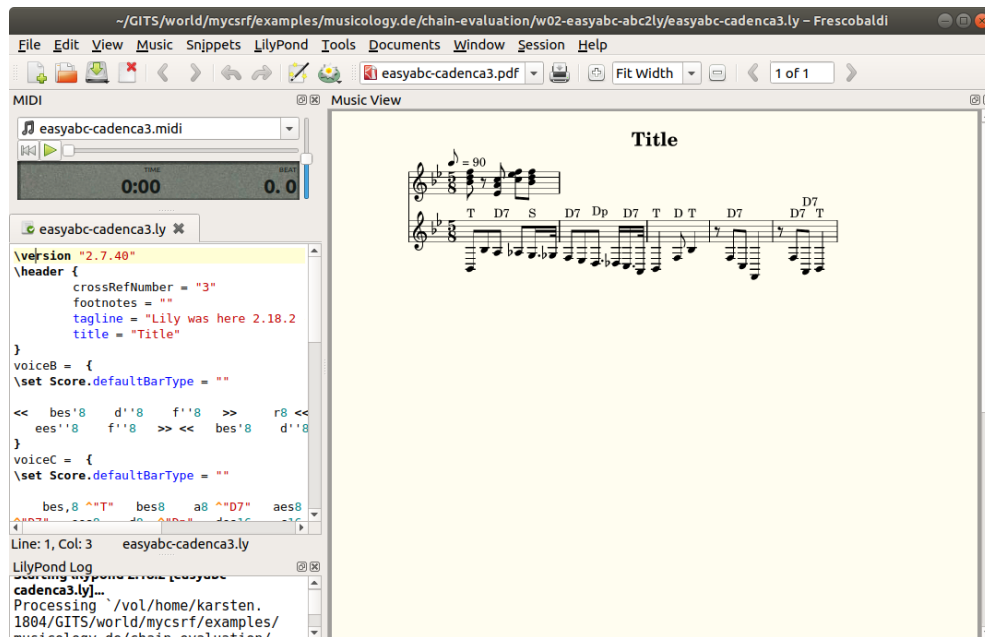


Mit dem folgenden Befehl kann man diese *ABC*-Outputdatei in eine *LilyPond*-Datei konvertieren:

```
abc2ly -o easyabc-cadenca3.ly easyabc-cadenca3.abc
```

Betrachtet man das Ergebnis in *Frescobaldi*, sieht man, dass die Konvertierung nur sehr bedingt funktioniert hat:

## 4 Vom Frontend nach L<sup>A</sup>T<sub>E</sub>X: die Toolketten



Daran ändert sich auch nichts, wenn man aus der *EasyABC* die 'Texte' entfernt und nur die Noten erfasst.

### 4.3 w04/w05: EasyABC → musicxml2ly → Frescobaldi/Elysium → ...

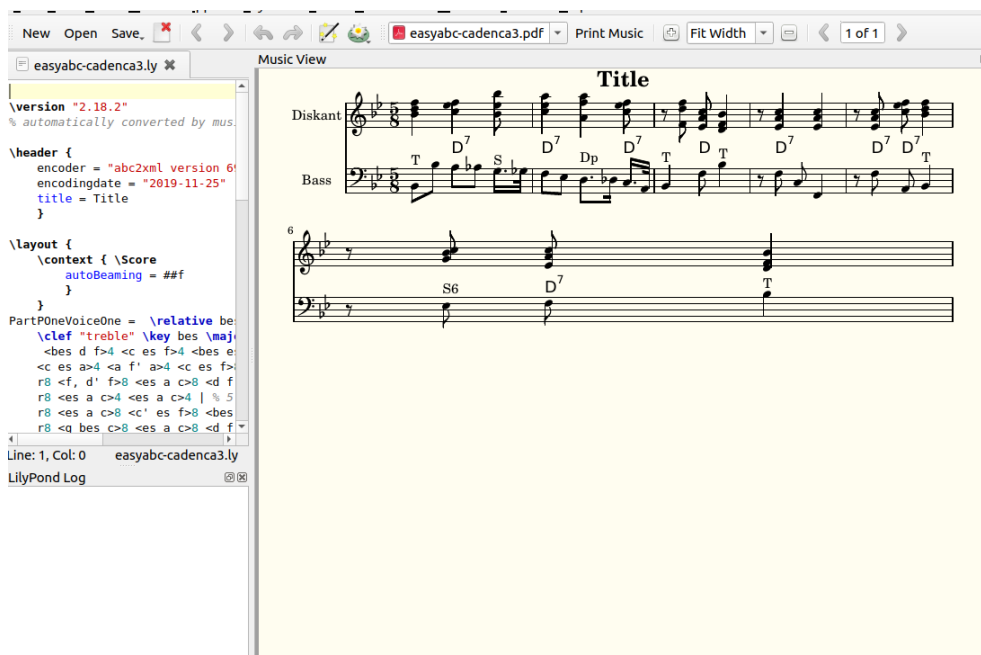
*EasyABC* erlaubt es nicht nur, seine Daten im *ABC*-Format zu speichern, sondern auch, sie als *Musicxml*-Datei zu exportieren. Damit wäre zu überprüfen, ob der von *LilyPond*-Konverter *musicxml2ly* die angelegten Daten so umformt, dass sie das erwartete Ergebnis in *Frescobaldi* und *Elysium* bzw. *LilyPond* generieren. Dazu haben wir im weg-spezifischen Subordner *w04w05-easyabc-musicxml2ly* – neben der bisher schon genutzten Ausgangsdatei *easyabc-cadenca3.abc* – auch die Exportdatei *easyabc-cadenca3.xml* abgelegt. Daraus erzeugt man mit folgendem Befehl die entsprechende *LilyPond*-Datei generieren:

```
musicxml2ly easyabc-cadenca3.abc
```

Und wenn man diese Datei mit *Frescobaldi* öffnet, erhält man eine perfekte Umformung:



#### 4.4 w06/07: MuseScore → musicxml2ly → Frescobaldi / Elysium → ...



#### 4.4 w06/07: MuseScore → musicxml2ly → Frescobaldi / Elysium → ...

Schließlich haben wir in dem Ordner [...]musicology/chain-evaluation/ den weg-spezifischen Subordner w0607-musescore-musicxml2ly erzeugt und darin die Kandenz-III als *MuseScore*-Datei (cadenca3.mscz) und den korrespondierenden MusicXML-Output (cadenca3.xml)abgelegt:

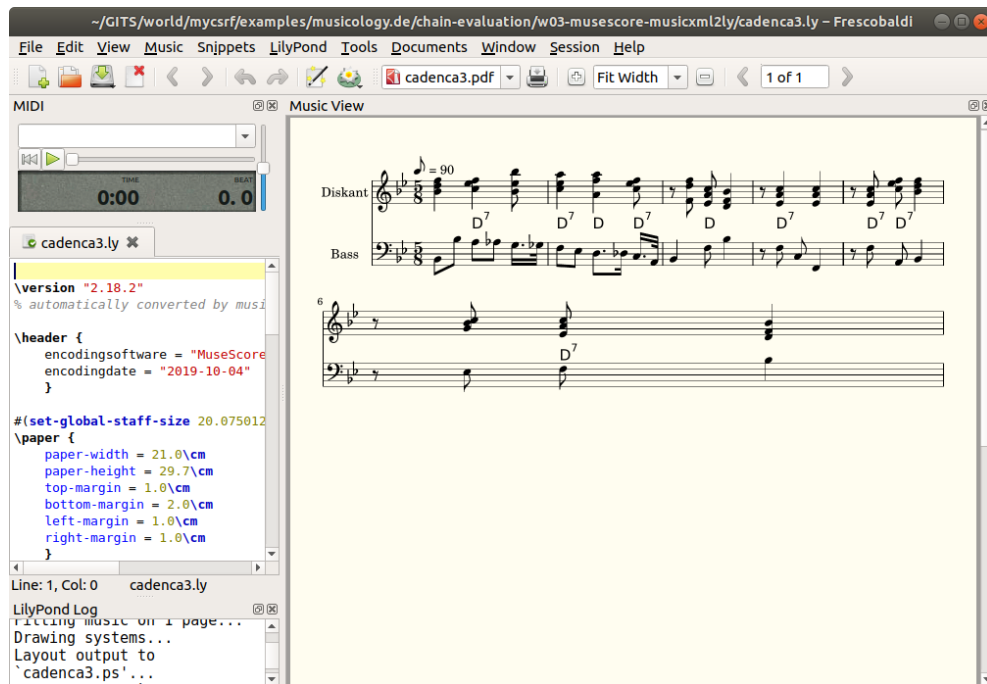


Damit sollte sich auch die XML-Datei in eine *LilyPond*-Datei konvertieren lassen:

```
musicxml2ly -o cadenca3.ly cadenca3.xml
```

Technisch läuft die Konvertierung problemlos durch. Das Ergebnis sieht in *Frescobaldi* nicht so schlecht aus:

## 4 Vom Frontend nach L<sup>A</sup>T<sub>E</sub>X: die Toolketten



Einzig sind einige Harmoniesymbole unter den Tisch gefallen, die eh durch die äquivalenten Ausdrücke über unser Zusatzbibliothek *harmonyli* ersetzt werden sollen.

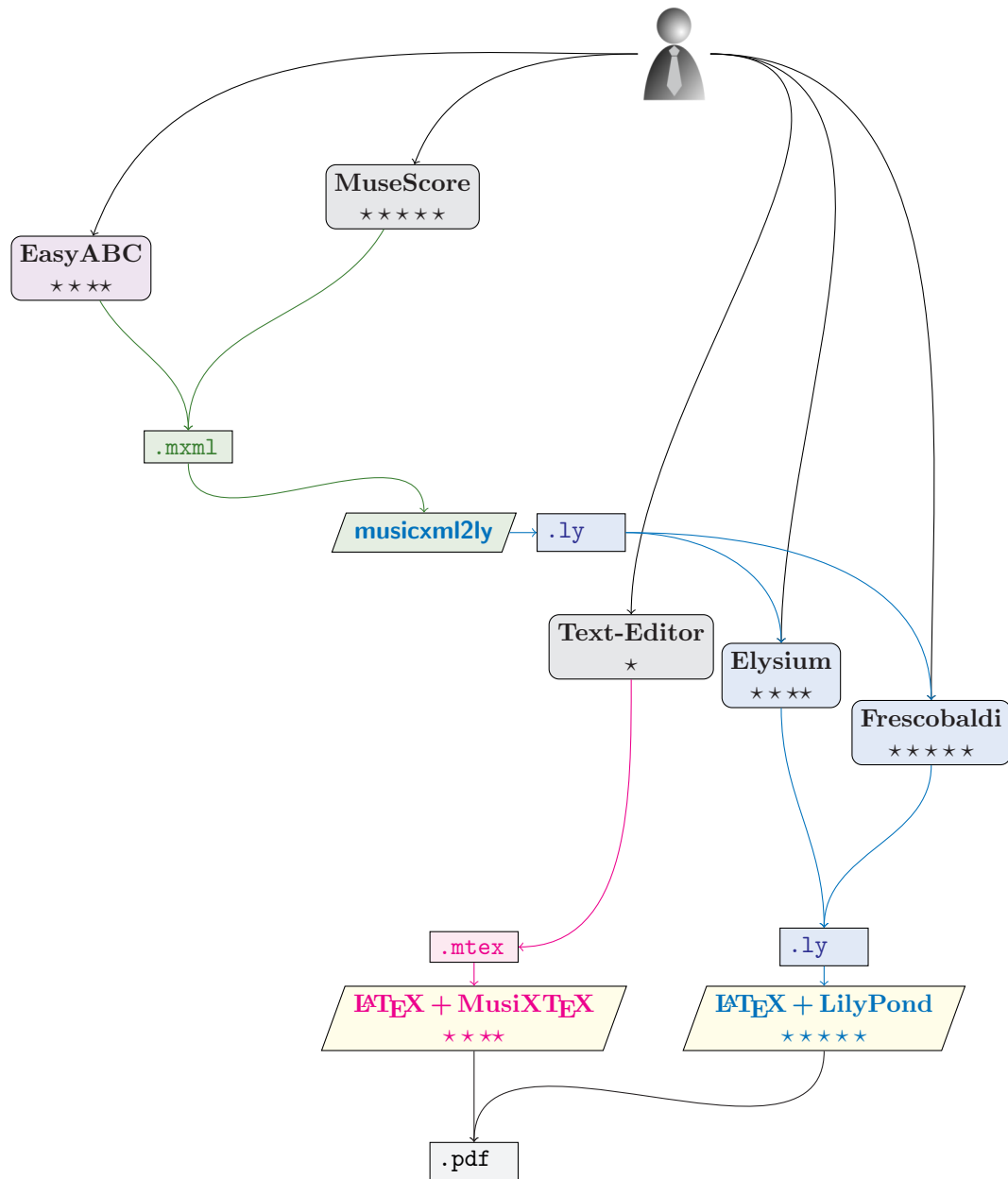
Damit dürfen wir sagen, dass man *LilyPond* auch mit einem echten graphischen Frontend verwenden kann, nämlich mit *MuseScore*, sofern man auch den von *LilyPond* gepflegten Konverter *musicxml2ly* verwendet. Dass das Frontend *MuseScore* von sich aus keine adäquaten Harmonieanalysesymbole mitbringt, ist kein Nachteil: man lässt sie ja eh weg, um später über die semi-graphische Editoren *Frescobaldi* oder *Elysium* die Syntagmen nachzutragen, die die Bibliothek *harmonyli.ly* in die Gesamtlung einbringt.

## 4.5 Vergleich

So können wir die Ergebnisse in Sachen einer Verknüpfung von Frontend und Backend zusammentragen:

Weg	VON		ÜBER		NACH		=
	Frontend	★	Konvertierung		L <sup>A</sup> T <sub>E</sub> X +	★	
01	EasyABC	4	→ abc2mtex →		MusixT <sub>E</sub> X	4	-
02/03	EasyABC	4	→ abc2ly → Frescobaldi/Elysium →		LilyPond	5	-
04/05	EasyABC	4	→ musicxml2ly → Frescobaldi/Elysium →		LilyPond	5	++
06/07	MuseScore	5	→ musicxml2ly → Frescobaldi/Elysium →		LilyPond	5	++
08	Frescobaldi	5	→		LilyPond	5	++
09	Elysium	4	→		LilyPond	5	++

Und damit reduziert sich unsere Wegskizze noch einmal:



## 5 Fazit

Damit haben wir das Ende unserer Untersuchung erreicht. Fassen wir das Unterfangen zusammen:

Aus Anlass einer anstehenden, größeren musikwissenschaftlichen Arbeit mussten wir in  $\text{\LaTeX}$ -Texte Notenbeispiele einbetten können, und zwar samt der ausgefeilten Symbolkomplexe, wie sie in der funktionalen Harmonieanalyse üblich sind. Eine einfache Anleitung dazu gab bisher nicht. Die Suche im Netz verwies stattdessen auf eine Fülle von Tools und Techniken, deren Nutzen und Nutzbarkeit und Kombinierbarkeit unklar blieb. So haben wir dieses Feld ausgeleuchtet, um den von der Qualität besten und von der Handlichkeit einfachsten Weg zu finden:

Notenbeispiele können auf verschiedenen Wegen in  $\text{\LaTeX}$ -Texten eingebettet werden. Wir haben drei Backendsysteme gefunden, nämlich  $\text{\LaTeX} + \text{ABC}$ ,  $\text{\LaTeX} + \text{MusiXTEX}$  und  $\text{\LaTeX} + \text{LilyPond}$ .

Letztlich kann man aus den Tools in und um  $\text{\LaTeX} + \text{ABC}$  kein akzeptables Editiersystem für Musikwissenschaftler zusammenstellen. Wenn man unbedingt will, steht der *EasyABC*-Editor und sein Export nach *musicxml* bereit, um von dort aus über den Konverter *musicxml2ly* in die *LilyPond*-Welt hinüberzuleiten. Dass der Musikwissenschaftler dann zusätzlich immer auch *LilyPond* kennen muss, ergibt sich aus der Tatsache, dass die wirklich komplexen Harmonieanalysesymbole erst über die *LilyPond*-Zusatzbibliothek *harmonyli.ly* in das Notenbild eingebracht werden können. Ob und in wie weit es Sinn macht, dafür zwei Repräsentationssprachen – also *ABC* und *LilyPond* – zu lernen, möge jeder für sich selbst entscheiden.

Deutlich klarer sieht die Lage bei  $\text{\LaTeX} + \text{MusiXTEX}$  und  $\text{\LaTeX} + \text{LilyPond}$  aus:

$\text{\LaTeX} + \text{MusiXTEX}$  – im Verbund mit dem  $\text{\LaTeX}$ -Tool *harmony* – und  $\text{\LaTeX} + \text{LilyPond}$  – in Kombination mit der *LilyPond* Bibliothek *harmonyli.ly* – bieten von sich aus die Option, hinreichend ausdrucksstarke Harmonieanalysesymbole in die Notenbeispiele zu integrieren. Das Druckergebnis ist exzellent und ganz auf der Höhe, die ein  $\text{\LaTeX}$ -Nutzer erwartet.

Der **Vorteil** des  $\text{\LaTeX} + \text{MusiXTEX}$ -Ansatzes liegt in der bruchlosen Integration in das gewohnte  $\text{\LaTeX}$ -Handling und in der Tatsache, dass  $\text{\LaTeX}$ -Syntagmen in *MusiXTEX*-Bereichen verwendet werden können.

Der **Nachteil** der  $\text{\LaTeX} + \text{MusiXTEX}$ -Methode liegt in der Komplexität und Unhandlichkeit der Auszeichnungssprache *MusiXTEX*. Leider gibt es kein graphisches oder semi-graphisches Frontend für dieses Backend. Die Idee, ein existierendes Frontend über einen Konverter dafür nutzbar zu machen, scheitert entweder daran, dass es keinen entsprechenden Konverter gibt (*ly2musictex*) oder dass die existierenden Konverter nicht adäquat arbeiten (*abc2ly* resp. *abc2mtex*).

Der **Nachteil** des  $\text{\LaTeX}$  + *LilyPond*-Ansatzes besteht dagegen darin, dass die Notenbeispiele nicht nativ in den  $\text{\LaTeX}$ -Text eingebettet werden, sondern 'nur' als vorab erzeugte Graphik. Damit muss man die Skalierungsfragen ebenso gesondert bedenken, wie man seine Make-Prozedur auf die Vorabnutzung von *lilypond-book* umstellen muss. Und dann bleibt immer noch, dass man  $\text{\LaTeX}$ -Konstrukte nicht in *LilyPond*-Umgebungen verwenden kann.

Die **Vorteile** der  $\text{\LaTeX}$  + *LilyPond*-Methode machen sie jedoch zum Mittel der Wahl:

- Zum ersten steht mit *LilyPond* eine im Vergleich zu *MusixTeX* deutlich einfachere Auszeichnungssprache zur Verfügung.
- Zum zweiten bietet sich mit *Frescobaldi* ein ausgezeichneteter semi-graphischen Editor an, der auch die eingebundene Bibliothek. *harmonyli.ly* und die daraus genutzten Funktionen korrekt auswertet.
- Neben diesem Standardeditor kann man auch auf das gute Eclipse-Plugin *elysium* zurückgreifen, wenn man *LilyPond* erstellen will, und zwar insbesondere dann, wenn man auch sein  $\text{\LaTeX}$ -Texte per *TeXlipse*-Eclipseplugin erzeugt.
- Außerdem darf man hoffen, dass mit *Canorus* noch ein dritter Kandidat entsteht, der wenigstens zukünftig gute Dienste leisten kann.
- Und schließlich kann man für *LilyPond* auch den graphischen Editor *MuseScore* als Frontend verwenden, sofern man in Kauf nimmt, vor der eigentlichen Arbeit auf  $\text{\LaTeX}$ -*LilyPond*-Ebene den Konverter *musicxml2ly* auf den *MuseScore*-XML-Export anzuwenden, das Ergebnis in *Frescobaldi* zu laden und dort die Harmonieanalysensymbole in einem zweiten Schritt einzugeben.

Damit dürfte auch klar sein, welche Methode wir selbst anwenden werden, nämlich  $\text{\LaTeX}$  + *LilyPond* + *harmonyli.ly*. Das wäre nicht möglich gewesen, wenn es sich hier nicht um Freie Open Source Software gehandelt hätte. Erst das gab uns die Möglichkeit, die Dinge selbst zu ergänzen, die wir für ein graphisch und inhaltlich adäquates Ergebnis noch benötigten.

Insgesamt sind wir froh, den Dschungel der Möglichkeiten, der sich nach der reinen Internetrecherche abgezeichnet hatte, gelichtet und die wirklich gangbaren Wege gefunden zu haben. Jetzt wissen wir, woran wir sind. Wir wünschten allerdings, wir hätten diesen Text schon eingangs von jemand anderem ausgehändigt bekommen, anstatt ihn selbst geschrieben haben zu müssen. Das hätte uns sehr viel Zeit gespart.

Möge also unser Text anderen den Aufwand für eine solche Tool-Evaluation sparen. Und möge er anderen als Ausgang für Updates, Verfeinerungen und Zusatzarbeiten dienen. Deshalb sei er auch als CC-BY-SA lizensierter Text veröffentlicht.

Frankfurt 2019-11-26 Karsten Reincke

## 6 Nachrede: Das 'vergessene Kapitel'?

Vielleicht mag sich der eine oder andere Leser nun wünschen, den idealen Weg einmal im Detail vorgeführt zu bekommen. Diesen sei gesagt, dass sie das nicht mehr nötig haben: Was zur Nutzung von *LilyPond* unter und mit L<sup>A</sup>T<sub>E</sub>X zu sagen war, haben wir im Kapitel über das Backend *LilyPond* ausgeführt.<sup>360</sup> Und was zu einer guten Nutzung der Bibliothek *harmonyli.ly*<sup>361</sup> noch zu sagen wäre, bietet bereits dessen Tutorial.<sup>362</sup> Insofern dürfen Sie jetzt direkt loslegen.

---

<sup>360</sup>) → [39](#)

<sup>361</sup>) vgl. *Blum, Karsten Reincke; Hans*: *harmonyli.ly*; 2019 ⇒ <https://github.com/kreincke/harmonyli.ly> – RDL: 2019-11-22, wp..

<sup>362</sup>) vgl. *Reincke*: *Harmonyli.ly Tutorial*, 2019.

## Abkürzungen

a.a.O. ....	am angegebenen Ort
anon. ....	Anonymous
ds. ....	kollektiv für ders., dies., ...
ebda. ....	ebenda
et al. ....	et aliter = u.a.
HP. ....	Homepage
RDL ....	Referenzdownload
u.a.O. ....	und andere Orte
u.a. ....	und Andere
ubdpw. ....	unbekannt, da nach 'Wiederabdruck' als Webpage zitiert
vgl. ....	vergleiche
wp. ....	webpage = Webdokument ohne innere Seitennummerierung

## Literaturverzeichnis

Albrecht, Mirko: Digitaler Notenschlüssel. Notensatzsoftware im Test; in: Linux User, 10 (2009), S.46–51 ⇒ [https://musescore.org/sites/musescore.org/files/MuseScore\\_LinuxUser\\_de\\_2009\\_10.pdf](https://musescore.org/sites/musescore.org/files/MuseScore_LinuxUser_de_2009_10.pdf) – RDL: 2019-02-22, FreeWeb / PDF

*Ältere Sichtung von Notensatzprogrammen, die die üblichen Kandidaten LilyPond, Frescobaldi, NoteEdit und MuseScore bespricht. Das damalige Fazit dürfte auch damals schon ungerecht gewesen sein, sagt es doch – ohne Frontend und Backendsysteme sorgsam auseinander zu halten –, die Open Source Welt habe Schwierigkeiten, ein Nischenprodukt wie ein Notensatzprogramm zu entwickeln.*

Andres, Jörg: NoteEdit – Ein graphischer Noteneditor für Linux; in: LinuxUser, (2002), Nr. 02, S.ubdwp. ⇒ <http://www.linux-community.de/ausgaben/linuxuser/2002/04/noteedit-ein-graphischer-noteneditor-fuer-linux/> – RDL: 2019-01-23, FreeWeb / HTML

*Eine frühe Beschreibung des Notensatzprogrammes durch den initialen Autor.*

Andres, Jörg: [NtEd Homepage]; o.J., FreeWeb / HTML ⇒ <https://vsr.informatik.tu-chemnitz.de/staff/jan/nted/nted.xhtml> – RDL: 2019-01-18

*Im Netz vielfach erwähnte Homepage des Notensatzprogramms NtEd - leider ist sie heute nicht mehr zugänglich.*

Andres, Jörg: [Selbstbeschreibung des NoteEdit- und NtEd-Autors Jörg Andres]; o.J., FreeWeb / HTML ⇒ <https://vsr.informatik.tu-chemnitz.de/about/people/anders/> – RDL: 2019-01-18

*Eine Selbstbeschreibungsseite, auf der sich Jörg Andres als Autor von NoteEdit und NtEd bezeichnet, wobei er beide Tools auf nicht mehr erreichbare Seiten verlinkt.*

anon.: Sound & MIDI Software For Linux. Music Notation Editors; o.J. [2006], FreeWeb / HTML ⇒ <http://linux-sound.org/notation.html> – RDL: 2019-01-18

*Eine deutliche ältere Site, die Sound-Software für Linux auflistet - darunter eben auch Notationsprogramme. Ob ihrer Fülle und thematischen Breite vermag diese Site auch heute zu helfen, selbst wenn Links und Pakete gelegentlich nicht mehr ganz aktuell sind.*

anon.: Frescobaldi Manual; o.J. [2012], FreeWeb / HTML ⇒ <http://frescobaldi.org/uguide.html> – RDL: 2019-02-21

*Die Online-Version des Frescobaldi-Handbuchs - trotz des 'älteren' Datums beschreibt es die wesentlichen Möglichkeiten.*

anon.: Frescobaldi Download; o.J. [2015], FreeWeb / HTML ⇒ <http://frescobaldi.org/download.html> – RDL: 2019-02-21

*Die Download- und Installationsseite aus der Frescobaldi-Site*

anon.: Frescobaldi [Homepage]; o.J. [2017], FreeWeb / HTML ⇒ <http://www.frescobaldi.org/index.html> – RDL: 2019-02-21



- Der Einstiegspunkt in die Frescobaldi-Site, der das Tool direkt als 'LilyPond sheet music text editor' bezeichnet.*
- anon.: ly2abc [Projektseite]; o.J. [2019], Freeweb / HTML  $\Rightarrow$  <https://github.com/aelkiss/ly2abc> – RDL: 2019-10-03  
*Die Github-Site des ly2abc-Projektes*
- anon.: xml2ly [Homepage]; o.J. [2019], Freeweb / HTML  $\Rightarrow$  <http://www.nongnu.org/xml2ly/> – RDL: 2019-10-03  
*Die Homepage des xml2ly-Konverters*
- anon.: xml2pmx Download Area; o.J. [2019], FreeWeb / HTML  $\Rightarrow$  <https://www.icking-music-archive.org/software/xml2pmx/> – RDL: 2019-10-03  
*Der Bereich im Icking-Software-Bereich, über den xml2pmx als Tool heruntergeladen werden kann.*
- anon. [CTAN]: Running TeXmuse; 2005, FreeWeb / PDF  $\Rightarrow$  <http://ctan.space-pro.be/tex-archive/macros/texmuse/Doc/pdf/running.pdf> – RDL: 2018-12-27  
*Die Installationsanleitung des TeXmuse-L<sup>A</sup>T<sub>E</sub>X-Paketes und seiner Schriften.*
- anon. [CTAN]: wasysym - L<sup>A</sup>T<sub>E</sub>X support file to use the WASY2 fonts; o.J. [2003], FreeWeb / HTML  $\Rightarrow$  <https://ctan.org/pkg/wasysym> – RDL: 2018-12-21  
*Die Homepage des wasysym-L<sup>A</sup>T<sub>E</sub>X-Paketes, das zusätzliche Sonderzeichen bereitstellt, darunter kleine Notensymbole.*
- anon. [CTAN]: harmony – Typeset harmony symbols, etc., for musicology; o.J. [2005], FreeWeb / HTML  $\Rightarrow$  <https://ctan.org/pkg/harmony> – RDL: 2018-12-21  
*Die Homepage des Harmony-L<sup>A</sup>T<sub>E</sub>X-Paketes, eines freien L<sup>A</sup>T<sub>E</sub>X-Addons, das die Zeichen für die Harmonieanalyse bereitstellt.*
- anon. [CTAN]: TeXmuse – Music typesetting system using TeX and METAFONT; o.J. [2005], FreeWeb / HTML  $\Rightarrow$  <https://ctan.org/pkg/texmuse> – RDL: 2018-12-27  
*Die Homepage TeXmuse-L<sup>A</sup>T<sub>E</sub>X-Paketes, das nur in einer ersten 'Interimsversion' vorgelegt worden ist.*
- anon. [CTAN]: abc – Support ABC music notation in L<sup>A</sup>T<sub>E</sub>X; o.J. [2016], FreeWeb / HTML  $\Rightarrow$  <https://ctan.org/pkg/abc> – RDL: 2018-12-25  
*Die Homepage des abc-L<sup>A</sup>T<sub>E</sub>X-Paketes, dass das text basierte Musiknotationssystem zur Formulierung einzeliger Notate für L<sup>A</sup>T<sub>E</sub>X bereitstellt.*
- anon. [CTAN]: M-Tx – A preprocessor for pmx; o.J. [2018], FreeWeb / HTML  $\Rightarrow$  <https://ctan.org/pkg/m-tx> – RDL: 2019-01-04  
*Die Homepage des M-Tx-Paketes, das einen freien Präprozessor für M-Tx anbietet, der – wie bei Liedern nötig – die Verbindung von Text und PMX encodierten Noten vereinfachen soll.*
- anon. [CTAN]: musicography – Accessing symbols for music writing with pdfL<sup>A</sup>T<sub>E</sub>X; o.J. [2018], FreeWeb / HTML  $\Rightarrow$  <https://ctan.org/pkg/musicography> – RDL: 2018-12-23  
*Die Homepage des musicography-L<sup>A</sup>T<sub>E</sub>X-Paketes mit zusätzlichen Sonderzeichen.*
- anon. [CTAN]: MusiX<sub>T</sub>E<sub>X</sub> – Sophisticated music typesetting; o.J. [2018], FreeWeb / HTML  $\Rightarrow$  <https://ctan.org/pkg/musixtex> – RDL: 2018-12-21  
*Die Homepage des MusiX<sub>T</sub>E<sub>X</sub>-Paketes, eines freien T<sub>E</sub>X basierten Systems zum Druck*

## Literaturverzeichnis

*von typographisch exzellent aufbereiteten Noten(blättern): Wer L<sup>A</sup>T<sub>E</sub>X schätzt, wird den Output von MusiX<sub>T</sub>E<sub>X</sub> lieben.*

anon. [CTAN]: pmx – Preprocessor for MusiX<sub>T</sub>E<sub>X</sub>; o.J. [2018], FreeWeb / HTML ⇒ <https://ctan.org/pkg/pmx> – RDL: 2018-12-21

*Die Homepage des PMX-Paketes, das einen freien Präprozessor für MusiX<sub>T</sub>E<sub>X</sub> anbietet, der die Eingabe von Noten in Form von Text – verglichen mit der von MusiX<sub>T</sub>E<sub>X</sub> – signifikant vereinfacht, auch wenn damit immer noch kein WYSIWYG-Tool bereitgestellt ist.*

anon. [Denemo Team]: Denemo [Download]; o.J. [2019], FreeWeb / HTML ⇒ <http://www.denemo.org/downloads-page/> – RDL: 2019-02-22

*Die Denemo-Download-Seite, die auch Pakete für Windows anbietet, die keine Installation benötigen.*

anon. [Denemo Team]: Denemo FAQ; o.J. [2019], FreeWeb / HTML ⇒ <http://www.denemo.org/faq/> – RDL: 2019-02-22

*Die Denemo-FAQ Seite, die u.a. auch eine Erklärung für den Namen 'Denemo' bietet: er sei eine 'Verballhornung' des französischen Wortes 'dénouement'*

anon. [Denemo Team]: Denemo [Homepage]; o.J. [2019], FreeWeb / HTML ⇒ <http://www.denemo.org/> – RDL: 2019-02-22

*Die Denemo-Hompage, die sagt, Denemo habe eine einzigartige Methode 'to enter music in a musical, rather than mechanical, manner'.*

anon. [Denemo Team]: [Denemo und] MuseScore [Kompatibilität bei Im- und Export per MusicXml- bzw. MIDI-Datei]; o.J. [2019], FreeWeb / HTML ⇒ <http://denemo.org/musescore/> – RDL: 2019-02-22

*Ein Beweis der Kompatibilität von MuseScore und Denemo in Sachen Im- und Export via MusicXml resp. MIDI.*

anon. [Eclipse Foundation]: Eclipse Foundation. The Platform for Open Innovation and Collaboration; o.J. [2018], FreeWeb / HTML ⇒ <https://www.eclipse.org/> – RDL: 2019-02-09

*Die Homepage der 'Eclipse-Foundation', die sich als Plattform der offenen Innovation und Zusammenarbeit darbietet und weit mehr sein möchte, als bloßer Distributor der integrierten Entwicklungsumgebung 'Eclipse'.*

anon. [Eclipse Foundation]: Eclipse Packages; o.J. [2018], FreeWeb / HTML ⇒ <https://www.eclipse.org/> – RDL: 2019-02-09

*Die Downloadpage der verschiedenen auf bestimmte Zwecke vorbereiteten Eclipse-Pakete. Für LilyPond und Elysium reicht ein einfaches Standardpaket.*

anon. [Github]: BRAHMS-SystemML/brahms, a modular execution middleware; o.J. [2018], FreeWeb / HTML ⇒ <https://github.com/BRAHMS-SystemML/brahms> – RDL: 2019-02-12

*Das offizielle Nachfolgerprojekt, per Abzweigen entstanden aus dem SourceForge-Projekt 'Brahms'. Hier bezeichnet sich die Software direkt als 'simulation execution engine'.*

anon. [Github]: Laborejo Github Page; o.J. [2018], FreeWeb / HTML ⇒ <https://github.com/diovudau/Laborejo> – RDL: 2019-02-11

*Die github-Homepage von Laborejo, die von den anderen 'Homepages' sagt, dass es keine richtige Website und kein sauberes Handbuch gäbe, weil Laborejo ein 'Hobbyprojekt' sei, das (aus Ressourcengründen) keine richtige Dokumentation erzeugen könne.*

anon. [Github]: Frescobaldi [Github Repository]; o.J. [2019], FreeWeb / HTML ⇒ <https://github.com/frescobaldi/frescobaldi> – RDL: 2019-02-21

*Das Github-Repository für die Quelltexte des Frescobaldi-Programms, eines in Python geschriebenen und unter der GPL veröffentlichten LilyPond-Frontends*

anon. [Github]: GNU Denemo. Free and Open Music Notation Editor. [Repository]; o.J. [2019], FreeWeb / HTML ⇒ <https://github.com/denemo/denemo> – RDL: 2019-02-23

*Das Denemo Repository mit dem Sourcecode des Programms*

anon. [Github]: MuseScore; o.J. [2019], FreeWeb / HTML ⇒ <https://github.com/musescore/MuseScore> – RDL: 2019-02-22

*Die Quelldateien für MuseScore.*

anon. [Lander University]: Ptolemaic: A Computer Application for Music and Visualization and Analysis. [Download]; o.J. [2016], FreeWeb / HTML ⇒ <http://ptolemaic.lander.edu/download> – RDL: 2019-02-16

*Die Downloadpage von Ptolemaic, eines Javaprogramms zur Visualisierung und Analyse vom Musik, das einfach gestartet werden kann.*

anon. [Lander University]: Ptolemaic: A Computer Application for Music and Visualization and Analysis. [Homepage]; o.J. [2016], FreeWeb / HTML ⇒ <http://ptolemaic.lander.edu/> – RDL: 2019-02-16

*Die Homepage von Ptolemaic, eines Computerprogramms zur Visualisierung und Analyse vom Musik gemäß verschiedener theoretischer Vorgaben, wie etwa der tonalen Funktionsanalyse von Harrison oder älteren Ansätzen. Trotz der innovativen Methoden kann es zum Notensatz nichts beitragen.*

anon. [MakeMusic]: MusicXML Homepage; o.J. [2018], FreeWeb / HTML ⇒ <https://www.musicxml.com/> – RDL: 2018-12-21

*Eine Seite aus der MusicXML-Site listet eine Fülle von Software auf, mit der MusicXML encodierte Dateien generiert resp. ver- und bearbeitet werden können.*

anon. [MakeMusic]: MusicXML Software; o.J. [2018], FreeWeb / HTML ⇒ <https://www.musicxml.com/software/> – RDL: 2019-01-17

*Die Homepage von MusicXML, dem per DTD und Xml-Schema über das W3C offiziell standardisierten freien Dateiformat, das die Notendatei wenigstens grundsätzlich vom verarbeitenden Programm entkoppelt und so den Datenaustausch zwischen Musikprogrammen 'gewährleistet'. Insofern ist es geschickt, seine Noten direkt in diesem Format zu speichern.*

anon. [MuseScore Team and Community]: MuseScore Akkordsymbole; o.J. [2019], FreeWeb / HTML ⇒ <https://musescore.org/de/handbook/akkordbezeichnungen> – RDL: 2019-02-22

*Eine Beschreibung möglicher Akkordsymbole und deren Integration in einen MuseScore-Notentext*

## Literaturverzeichnis

- anon. [MuseScore Team and Community]: MuseScore [Download]; o.J. [2019], FreeWeb / HTML ⇒ <https://musescore.org/de/download> – RDL: 2019-02-22  
*Die Downloadseite des Programms 'MuseScore'. Aktuell ist seit 2019 die Version 3.x; gängige LTS Distributionen bringen deshalb gelegentlich noch eine 2.x Version mit.*
- anon. [MuseScore Team and Community]: MuseScore Handbuch 2; o.J. [2019], FreeWeb / HTML ⇒ <https://musescore.org/de/handbuch-2> – RDL: 2019-02-22  
*Die Onlineversion des MuseScore-Handbuchs, ausgelegt auf die Version 2*
- anon. [MuseScore Team and Community]: MuseScore Handbuch 3; o.J. [2019], FreeWeb / HTML ⇒ <https://musescore.org/de/handbook> – RDL: 2019-02-22  
*Die Onlineversion des MuseScore-Handbuchs, ausgelegt auf die Version 3*
- anon. [MuseScore Team and Community]: MuseScore [Homepage]; o.J. [2019], FreeWeb / HTML ⇒ <https://musescore.org/de> – RDL: 2019-02-22  
*Die Homepage des Programms 'MuseScore', das sich selbst als das 'beliebteste Notensatzprogramm der Welt' bezeichnet und das 'verspricht', wunderschöne Notenblätter erzeugen, drucken und wiedergeben zu können*
- anon. [MuseScore Team and Community]: MuseScore How To[s]; o.J. [2019], FreeWeb / HTML ⇒ <https://musescore.org/de/howto> – RDL: 2019-02-22  
*Eine Suchmaske, über die sich die verschiedensten Anleitungen zu Spezialthemen finden lassen, gefiltert nach der je angegebenen Version.*
- anon. [MuseScore Team and Community]: MuseScore PDF-Handbuch 2; o.J. [2019], FreeWeb / PDF ⇒ <https://ftp.osuosl.org/pub/musescore/handbook/MuseScore-2.0/MuseScore-en.pdf> – RDL: 2019-02-22  
*Die PDF-Version des MuseScore-Handbuchs, ausgelegt auf die Version 2. Die PDF-Versionen werden vom Open Source Lab der Oregon State University bereitgestellt, deshalb der veränderte Link.*
- anon. [MuseScore Team and Community]: MuseScore PDF-Handbuch 3; o.J. [2019], FreeWeb / PDF ⇒ <https://ftp.osuosl.org/pub/musescore-nightlies/handbook/MuseScore-3.0/MuseScore-de.pdf> – RDL: 2019-02-22  
*Die PDF-Version des MuseScore-Handbuchs, ausgelegt auf die Version 3. Die PDF-Versionen werden vom Open Source Lab der Oregon State University bereitgestellt, deshalb der veränderte Link.*
- anon. [MuseScore Team and Community]: MuseScore Tutorials; o.J. [2019], FreeWeb / HTML ⇒ <https://musescore.org/de/tutorials> – RDL: 2019-02-22  
*Eine Seite mit verschiedenen Videoanleitung zur Nutzung von MuseScore*
- anon. [Repology]: Versions for canorus; o.J. [2019], FreeWeb / HTML ⇒ <https://repology.org/metapackage/canorus/versions> – RDL: 2019-01-27  
*Seite eines Services, der aktuelle Pakete von Opensource-Software listet, in diesem Fall die, die von Canorus erhältlich sind.*
- anon. [SourceForge]: EasyABC Homepage; 2017, FreeWeb / HTML ⇒ <http://easyabc.sourceforge.net/> – RDL: 2019-02-04  
*Die neuere EasyABC-Hompage, die über Installationsoptionen informiert.*
- anon. [SourceForge]: EasyABC Releases; 2017, FreeWeb / HTML ⇒ <https://sourceforge.net/projects/easyabc/files/EasyABC/> – RDL: 2019-02-04  
*Die neuere EasyABC-Versionsseite*

- anon. [SourceForge]: EasyABC Repository; 2017, FreeWeb / HTML  $\Rightarrow$  <https://sourceforge.net/projects/easyabc/> – RDL: 2019-02-04  
*Die neuere EasyABC-Projektseite, die sehr grundsätzlich über das Programm informiert und den Download offeriert.*
- anon. [SourceForge]: MuX2D [Homepage]; o.J. [2000]  $\Rightarrow$  <http://mux2d.sourceforge.net/> – RDL: 2019-02-14  
*Die Homepage eines richtig alten 'WYSIWYM(ean) Editors für MusiX<sub>TE</sub>X'*
- anon. [SourceForge]: MuX2D [Projectpage]; o.J. [2000]  $\Rightarrow$  <https://sourceforge.net/projects/mux2d/> – RDL: 2019-02-14  
*Die Projektsite des 'WYSIWYM(ean) Editors für MusiX<sub>TE</sub>X' einschließlich der Möglichkeit zum Download der Source- und der Binärversion. Gleichwohl können diese auf aktuellen Systemen nicht mehr gestartet bzw. kompiliert werden.*
- anon. [SourceForge]: Audimus Download; o.J. [2008], FreeWeb / HTML  $\Rightarrow$  <https://sourceforge.net/projects/audimus/files/> – RDL: 2019-01-31  
*Die Audimus-Downloadpage. Das 1.0 Release stammt vom 09/2007, ein anschließendes Testrelease von 08/2008.*
- anon. [SourceForge]: Audimus Homepage; o.J. [2008], FreeWeb / HTML  $\Rightarrow$  <https://sourceforge.net/projects/audimus/> – RDL: 2019-01-31  
*Die Audimus-Homage eines älteren Java basierten Notensatzprogrammes, das 'nur' MusicXML liest und schreibt und das heute nur noch begrenzt lauffähig ist.*
- anon. [SourceForge]: Free Clef [Homepage]; o.J. [2008]  $\Rightarrow$  <https://sourceforge.net/projects/freeclef/> – RDL: 2019-02-12  
*Die Free Clef Homage, des leichtgewichtigen und ausgelaufenen Notensatzsystems, das seine Eingaben im Format MusicXML abzuspeichern versprach.*
- anon. [SourceForge]: Brahms Documentation: What is Brahms; o.J. [2013], FreeWeb / HTML  $\Rightarrow$  <http://brahms.sourceforge.net/docs/What%20is%20BRAHMS.html> – RDL: 2019-02-12  
*Die Dokumentation, die Aufschluss darüber gibt, was 'Brahms' wirklich ist, nämlich, grob gesagt, eine Prozessmaschine, und keine Notationssoftware.*
- anon. [SourceForge]: Brahms Homepage; o.J. [2013], FreeWeb / HTML  $\Rightarrow$  <http://brahms.sourceforge.net/home/> – RDL: 2019-02-12  
*Die scheinbare Homepage eines angeblichen Sequencers names 'Brahms'. Nur hat das, was das als Programm beschrieben und angeboten wird, wenig bis gar nichts mit Musik zu tun, jedenfalls nichts mit Notensatz.*
- anon. [SourceForge]: Noteedit - A Score Editor; o.J. [2014], FreeWeb / HTML  $\Rightarrow$  <https://sourceforge.net/projects/noteedit.berlios/> – RDL: 2019-01-23  
*Das SourceForge Repository zum Download der Noteedit Quellen.*
- anon. [SourceForge]: Aria Maestosa; o.J. [2017], FreeWeb / HTML  $\Rightarrow$  <http://ariamaestosa.sourceforge.net/> – RDL: 2019-02-03  
*Die Homepage eines MIDI-Sequencers und Editors, das die Noteneingabe graphisch über die Simulation von Instrumenten anbietet und das 'nur' insofern ein Notensatzprogramm ist, als es auch den Ausdruck der erarbeiteten Noten erlaubt.*
- anon. [SourceForge]: Aria Maestosa Installation; o.J. [2017], FreeWeb / HTML  $\Rightarrow$  <http://ariamaestosa.sourceforge.net/building.html> – RDL: 2019-02-03



*Die Anleitung zur Installation eines MIDI-Sequencers und Editors, auch aus den Quellen heraus. Unglücklicherweise erwartet die Version von 2017 heute veraltete Bibliotheken.*

anon. [SourceForge]: Aria Maestosa Manual; o.J. [2017], FreeWeb / HTML ⇒ <http://ariamaestosa.sourceforge.net/man.html> – RDL: 2019-02-03

*Die Anleitung zur Nutzung des MIDI-Sequencers und Editors, die auch die Eingabe der Noten per Simulation von Instrumenten beschreibt.*

anon. [SourceForge]: Canorus - music score editor. [Files]; o.J. [2019], FreeWeb / HTML ⇒ <https://sourceforge.net/projects/canorus/files/> – RDL: 2019-01-27

*Die Download- und Releasesseite von Canorus*

anon. [SourceForge]: Canorus - music score editor. [Homepage]; o.J. [2019], FreeWeb / HTML ⇒ <https://sourceforge.net/projects/canorus/> – RDL: 2019-01-27

*Die Homepage von Canorus und zugleich der Einstiegspunkt in die Projektsite - sie listet Eigenschaften auf, bietet Beispiele und zeigt die letzten Aktivitäten an.*

anon. [SourceForge]: Jniz [Downloadpage]; o.J. [2019], FreeWeb / HTML ⇒ <https://sourceforge.net/projects/jniz/files/> – RDL: 2019-02-17

*Die Downloadpage im Jniz-Repository*

anon. [SourceForge]: Jniz [Projectsseite]; o.J. [2019], FreeWeb / HTML ⇒ <https://sourceforge.net/projects/jniz/> – RDL: 2019-02-17

*Das Interface zum Jniz-Repository*

anon. [SourceForge]: TeXlipse Homepage; o.J. [2019], FreeWeb / HTML ⇒ <http://texlipse.sourceforge.net/> – RDL: 2019-02-09

*TeXlipse, das Plugin, das die Eclipse IDE um L<sup>A</sup>T<sub>E</sub>X Unterstützung erweitert.*

anon. [Ubuntu]: Canorus; o.J. [2014], FreeWeb / HTML ⇒ <https://wiki.ubuntuusers.de/Canorus/> – RDL: 2019-01-27

*Die ältere Ubuntu-Seite für Canorus, der man auch entnehmen kann, dass die Entwicklung auch schon vor 2015 unterbrochen worden ist.*

anon. [Ubuntu]: Denemo. [Das Ubuntu Package]; o.J. [2014], FreeWeb / HTML ⇒ <https://wiki.ubuntuusers.de/Denemo/> – RDL: 2019-02-23

*Eine ältere Ubuntu-Seite für das Denemopakete, die aber immer noch hilfreiche Tipps zur Nutzung bietet.*

anon. [Ubuntu]: Frescobaldi; o.J. [2016], FreeWeb / HTML ⇒ <https://wiki.ubuntuusers.de/Frescobaldi/> – RDL: 2019-02-21

*Die Ubuntu-Seite von Frescobaldi, die kurz auch auf die Einbindung des Sounds über den Midi-Server 'timidity' eingeht.*

anon. [Ubuntu]: NtEd; o.J. [2016], FreeWeb / HTML ⇒ <https://wiki.ubuntuusers.de/NtEd/> – RDL: 2019-01-18

*Eine Ubuntu-Seite zu NtEd, in der Jörg Andres als Autor und die mittlerweile nicht mehr erreichbare Homepage genannt wird, nicht ohne allerdings darauf zu verweisen, dass die NtEd-Hompage seit 11/2017 nicht mehr zugänglich ist. Das bedeutet auch, dass man die Quellen und Dokumentation nur noch auf Umwegen erreicht.*

anon. [Ubuntu]: MuseScore; o.J. [2018], FreeWeb / HTML ⇒ <https://wiki.ubuntuusers.de/MuseScore/> – RDL: 2019-02-22

*Die Beschreibung des MuseScore-Paketes für Ubuntu 2018-04*

- anon. [Wikipedia]: Denemo; o.J. [2018], <https://de.wikipedia.org/wiki/Denemo> ⇒ <https://de.wikipedia.org/wiki/Denemo>  
*Deutsche Denemo-Seite in Wikipedia, die deutlich auf die verschiedenen Dateitypen verweist.*
- anon. [Wikipedia]: MusicXML; o.J. [2018], FreeWeb / HTML ⇒ <https://de.wikipedia.org/wiki/MusicXML> – RDL: 2018-12-21  
*Einige ergänzende Informationen zur Entstehung des Formates und zur Lizenzierung als freier Standard: die Freigabe zur Nutzung macht das Format erst wirklich praktikabel.*
- anon. [Wikipedia]: Baum (Graphentheorie); o.J. [2019], FreeWeb / HTML ⇒ [https://de.wikipedia.org/wiki/Baum\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Baum_(Graphentheorie)) – RDL: 2019-01-11  
*Eine erste Orientierung in Sachen graphentheoretischer Definition 'Baum'.*
- anon. [Wikipedia]: Canorus; o.J. [2019], FreeWeb / HTML ⇒ <https://de.wikipedia.org/wiki/Canorus> – RDL: 2019-01-23  
*Generelle Einordnung von Canorus als Notensatzprogramm.*
- anon. [Wikipedia]: GNU Guile; o.J. [2019], FreeWeb / HTML ⇒ [https://de.wikipedia.org/wiki/GNU\\_Guile](https://de.wikipedia.org/wiki/GNU_Guile) – RDL: 2019-01-14  
*Als GNU-Projekt nutzt LilyPond für seine Markup-Language auch die 'GNU Ubiquitous Intelligent Language for Extensions' und macht sich so erweiterbar. Gleichwohl muss man bei seiner Nutzung die LilyPond spezifischen Evaluationsfunktionen berücksichtigen.*
- anon. [Wikipedia]: Liste von Notensatzprogrammen; o.J. [2019], FreeWeb / HTML ⇒ [https://de.wikipedia.org/wiki/Liste\\_von\\_Notensatzprogrammen](https://de.wikipedia.org/wiki/Liste_von_Notensatzprogrammen) – RDL: 2019-01-17  
*Eine umfangreiche Liste von Notensatzprogrammen, klassifiziert nach den Kriterien 'proprietär' versus 'open source', 'visuell' versus 'markup' und 'aktuell' versus 'obsolet'.*
- anon. [Wikipedia]: Musical Instrument Digital Interface; o.J. [2019], FreeWeb / HTML ⇒ [de.wikipedia.org/wiki/Musical\\_Instrument\\_Digital\\_Interface](https://de.wikipedia.org/wiki/Musical_Instrument_Digital_Interface) – RDL: 2019-02-02  
*Grundsätzliche Informationen über das 'Wesen' von MIDI als 'Standard für den Austausch musikalischer Steuerinformationen zwischen elektronischen Instrumenten'.*
- Arkkra Enterprises: [Homepage of] Mup [, the] music publication program; o.J. [2017ff], FreeWeb / HTML ⇒ <http://www.arkkra.com/> – RDL: 2019-01-26  
*Die Homepage des 'music publication program's, beschrieben als ein weiteres textbasiertes Musiknotationssystem, mit dem man schließlich hochwertige Postskript-dateien erzeugen können soll.*
- Arkkra Enterprises: Mup Licence; o.J. [2017ff], FreeWeb / HTML ⇒ <http://www.arkkra.com/doc/license.html> – RDL: 2019-01-26  
*Die Mup Lizenz, eine Instantiierung der BSD-3-Clause Lizenz.*
- Arkkra Enterprises: The Mup Music Publication program [Featurelist]; o.J. [2017ff], FreeWeb / HTML ⇒ <http://www.arkkra.com/doc/mupfeat.html> – RDL: 2019-01-26  
*Eigenschaften der Mup Software.*

## Literaturverzeichnis

- Arkra Enterprises: Obtaining a copy of Mup Version 6.6; o.J. [2017ff], FreeWeb / HTML  $\Rightarrow$  <http://www.arkkra.com/doc/obtain.html> – RDL: 2019-01-26  
*Verschiedene Wege, Mup als Software zu bekommen und zu installieren. Nicht alle Distributionen bringen die Software als Paket mit.*
- Blum, Karsten Reincke; Hans: harmonyli.ly; 2019, FreeWeb / HTML  $\Rightarrow$  <https://github.com/kreincke/harmonyli.ly> – RDL: 2019-11-22  
*Die freie LilyPond Bibliothek 'harmonyli.ly' für Musikwissenschaftler samt Beispielen und umfangreichem Tutorial*
- Brendel, Jens-Christoph: Notensatz unter Linux – eine Übersicht; in: Linux-Magazin, 9 (2005), S. ubdwp.  $\Rightarrow$  <http://www.linux-magazin.de/ausgaben/2005/09/digitale-notensteher/> – RDL: 2019-01-22, FreeWeb / HTML  
*Eine ältere 'Sichtung': bespricht MUP, ABC PLUS, LilyPond, Noteedit, Rosegarden und KGuitar.*
- Callon, Gordon J.: Music Notation Software; o.J. [2009], FreeWeb / HTML  $\Rightarrow$  <http://www.acadiau.ca/~gcallon/www/others.htm> – RDL: 2019-02-12  
*Eine recht alte alphabetische Auflistung von Notensatzsystemen und -programmen, die noch gut zu gebrauchen ist, wenn es darum geht, ältere Programme wiederzufinden.*
- Cannam, Chris et al.: The Rosegarden Manual; o.J. [2019], FreeWeb / HTML  $\Rightarrow$  <https://www.rosegardenmusic.com/wiki/doc:manual-en> – RDL: 2019-02-24  
*Die Rosegardendokumentation, bereitgestellt als Wiki und ausgelegt als durchgehendes Dokument, bei dem nur einige Teile ausgelagert und verlinkt sind.*
- Cannam, Chris et al.: The Rosegarden Manual. Notationeditor; o.J. [2019], FreeWeb / HTML  $\Rightarrow$  <https://www.rosegardenmusic.com/wiki/doc:notation-en> – RDL: 2019-02-24  
*Die ausgelagerte Dokumentation zum Notation-Editor, über den leere und per MIDI importierte Stimmen visuell per Notentext (um)komponiert werden können.*
- Cashner, Andrew A.: The musicography Package: Symbols for Music Writing with pdflatex; 2018, FreeWeb / PDF  $\Rightarrow$  <http://ctan.space-pro.be/tex-archive/macros/latex/contrib/musicography/musicography.pdf> – RDL: 2018-12-23  
*Ein Paket, dass Irritationen bei der Erzeugung druckfähiger PDFs per pdflatex mit einem neuen Zeichensatz lösen will, für den auch die entsprechenden PDF Fonts bereitgestellt werden.*
- Chambers, John: ABC Music Notation; o.J. [2009], FreeWeb / HTML  $\Rightarrow$  <http://trillian.mit.edu/~jc/music/abc/doc/ABCTutorial.html> – RDL: 2018-12-27  
*Eine weitere Online-Anleitung zur Nutzung der ABC-Notation.*
- [CTAN]: abc2mtex – Notate tunes stored in ABC notation; o.J. [2019], FreeWeb / HTML  $\Rightarrow$  <https://ctan.org/pkg/abc2mtex?lang=de> – RDL: 2019-10-01  
*Paketpage für das Tool, dass ABC-Dateien nach MusiXTeX konvertiert*
- Dorok, Sebastian: Notenprogramme 2014; 2014, FreeWeb / PDF  $\Rightarrow$  <https://www.dorok.info/notationsprogramme-2014/> – RDL: 2019-02-22  
*Der Wertungsseite zur allgemeineren Sichtung, auf der MuseScore zum 'Gewinner' erklärt wird.*
- Dorok, Sebastian: Notenprogramme im Überblick; 2014, FreeWeb / PDF  $\Rightarrow$  <https://www.herrdorok.de/notenprogramme-im-ueberblick/> – RDL: 2019-02-22



- Sehr kursorische Sichtung von Open Source und Freeware Notensatzprogrammen, die nicht zwischen Backend und Frontend unterscheidet und so – grosso modi – den freien Programmen, ausgenommen von MuseScore, eine gringe Qualität zuspricht.*
- Emms, Steve: 11 Excellent Free Scorewriters – Compose, arrange, print, and publish music; 2018, FreeWeb / HTML  $\Rightarrow$  <https://www.linuxlinks.com/excellent-scorewriters-compose-great-music/>
- Eine jüngere Auflistung freier Notensatzprogramme, nämlich u.a. LilyPond, Rosegarden, MuseScore, Denemo, Frescobaldi und Canorus. Die Besprechungen der einzelnen Systeme sind verlinkt, beschränken sich aber ihrerseits im Wesentlichen auf eine Auflistung der jeweiligen Eigenschaften.*
- Enzenhofer, Michael: L<sup>A</sup>T<sub>E</sub>X für Musiker; 2016, FreeWeb / PDF  $\Rightarrow$  [http://www.michael-enzenhofer.at/LaTeXfuerMusiker/Latex4Musiker\\_1.pdf](http://www.michael-enzenhofer.at/LaTeXfuerMusiker/Latex4Musiker_1.pdf) – RDL: 2019-01-11
- Eine eher grundsätzliche Einführung in Sachen Musik und L<sup>A</sup>T<sub>E</sub>X: Sie bietet einen ersten Überblick über die L<sup>A</sup>T<sub>E</sub>X- und LilyPond-Codierung.*
- Felix, Ted: Ted's Linux MIDI Guide; 2018, FreeWeb / HTML  $\Rightarrow$  <http://tedfelix.com/linux/linux-midi.html> – RDL: 2019-02-10
- Eine sehr detaillierte und kenntnisreiche Erläuterung, wie man sein Linux-System wirklich sauber MIDI-fähig macht. Für gelegentliche Arbeiten braucht man aber vielleicht nicht das ganze Instrumentarium.*
- Free Software Foundation: Freie Software. Was ist das? Übersetzung aus dem Amerikanischen; o.J. [2016], FreeWeb / HTML  $\Rightarrow$  <https://www.gnu.org/philosophy/free-sw.de.html> – RDL: 2018-12-21
- Kurze Erklärung des Wesens von freier Software, insbesondere der vier Freiheiten, die Software erst zu freier Software machen.*
- Garcia, Federico: T<sub>E</sub>Xmuse's Main Loop; 2005, FreeWeb / HTML  $\Rightarrow$  <http://ftp.fau.de/ctan/macros/texmuse/Doc/pdf/mainloop.pdf> – RDL: 2018-12-27
- Die Anleitung zur Nutzung von T<sub>E</sub>Xmuse.*
- Gardner, Matthew u. Sara Springfeld: Musikwissenschaftliches Arbeiten. Eine Einführung; 2. Auflage. Kassel: Bärenreiter-Verlag, 2018 (= Bärenreiter Studienbücher Musik), Print: ISBN 978-3-7618-2249-4
- Die aktuellste Einführung in die Musikwissenschaft*
- [Github]: abcm2ps [Repository]; o.J. [2019], FreeWeb / HTML  $\Rightarrow$  <https://github.com/leesavide/abcm2ps/> – RDL: 2019-02-27
- Das abcm2ps repository, das die Quellen des GPL lizenzierten Programms öffentlich zugänglich macht.*
- Gonzato, Guido: Musik setzen mit ABC Plus. Einführung in die ABCPLUS Notation und Applikation; ins Deutsche übers. v. Michael Krause; 2003, FreeWeb / PDF  $\Rightarrow$  <http://abcplus.sourceforge.net/ABCPlus.pdf> – RDL: 2018-12-28
- Deutsches Tutorial zur Nutzung der ABC-Notationsmethode inklusive der Erweiterungen zur Erfassung von mehrersystemigen Noten.*
- Gonzato, Guido: Making Music with Abc 2; A practical guide to the Abc notation; [formerly: 'Making Music with Abc Plus']; 2018, FreeWeb / PDF  $\Rightarrow$  [http://abcplus.sourceforge.net/abcplus\\_en.html](http://abcplus.sourceforge.net/abcplus_en.html) – RDL: 2018-12-28

## Literaturverzeichnis

- Tutorial zur Nutzung der ABC-Notationsmethode inklusive der Erweiterungen zur Erfassung von mehrersystemigen Noten.*
- Gonzato, Guido: The ABC Plus Project; o.J. [2018], FreeWeb / HTML ⇒ <http://abcplus.sourceforge.net/> – RDL: 2018-12-28  
*Die Homepage des ABC-Plus-Projektes, das die ABC-Notationsmethodik auf Mehrstimmigkeit in mehreren Systemen ausweitete.*
- Grabner, Hermann: Allgemeine Musiklehre; mit einem Nachtrag v. Diether de la Motte; 11. Auflage. Kassel, Basel [... u.a.O.]: Bärenreiter Verlag, 1974, Print: ISBN 3-7618-0061-4  
*Das Standardwerk der Musikanalyse, das nicht erst heute mit hoher Auflagennummer erscheint, sondern auch schon vor 50 Jahren so erschienen ist - also im wahrsten Sinne des Wortes: ein Jahrhundertwerk.*
- Grandjean, Bruno: Jniz [Homepage]; o.J. [2019], FreeWeb / HTML ⇒ <http://www.jniz.org/index.php/en/> – RDL: 2019-02-17  
*Die Homepage von Jniz, einem Programm zur (automatisierten) Komposition mehrstimmiger Sätze gemäß der Regeln der klassischen Harmonisierung*
- Grandjean, Bruno: Jniz [Lizenzseite]; o.J. [2019], FreeWeb / HTML ⇒ <http://www.jniz.org/index.php/en/license> – RDL: 2019-02-17  
*Die Lizenzseite von Jniz, die das Programm unter eine sehr eigenwillige Lizenz stellt, sodass es sicher keine Opensource-Software ist.*
- Grandjean, Bruno: Jniz V2 Howto; o.J. [2019], FreeWeb / PDF ⇒ <http://www.jniz.org/doc/howtoV2.pdf> – RDL: 2019-02-17  
*Das kurze Jniz-Howto.*
- Gregorio, Enrico: The abc package; 2016, FreeWeb / PDF ⇒ <https://ctan.org/tex-archive/macros/latex/contrib/abc/abc.pdf> – RDL: 2018-12-25  
*Erläuterung des ABC L<sup>A</sup>T<sub>E</sub>X-Paketes.*
- Guepewi: Aria Meastosa; 2017, FreeWeb / HTML ⇒ <https://www.paules-pc-forum.de/forum/thread/177505-aria-maestosa/> – RDL: 2019-02-03  
*Eine Beschreibung des Programms aus einem Windowskatalog, der auch das Exportformat 'aiff' erwähnt.*
- Harmath, Dénes: Elysium [im Eclipse Marketplace]; 2019, FreeWeb / HTML ⇒ <https://marketplace.eclipse.org/content/elysium> – RDL: 2019-02-09  
*Die Elysium-Package-Seite im Eclipse Marketplace*
- Harmath, Dénes: Elysium. The LilyPond IDE for Eclipse. [Eigenschaften]; 2019, FreeWeb / HTML ⇒ <http://elysium.thsoft.hu/features> – RDL: 2019-02-09  
*Die Seite, auf der der Autor die Eigenschaften von Elysium skizziert.*
- Harmath, Dénes: Elysium. The LilyPond IDE for Eclipse. [Homepage]; 2019, FreeWeb / HTML ⇒ <http://elysium.thsoft.hu/> – RDL: 2019-02-09  
*Die Homepage von Elysium, die das Plugin als 'Zen Garten' von LilyPond bezeichnet und mit dem Slogan 'Text-based music notation made easy' beginnt.*
- Harmath, Dénes: Elysium. The LilyPond IDE for Eclipse. [Installation]; 2019, FreeWeb / HTML ⇒ <http://elysium.thsoft.hu/> – RDL: 2019-02-09  
*Die Seite, auf der die Installation erklärt wird: man möge LilyPond installieren, dann Eclipse und Elysium - und fertig.*

- Harmath, Dénes: Elysium. The LilyPond IDE for Eclipse. [Motivation]; 2019, FreeWeb / HTML  $\Rightarrow$  <http://elysium.thsoft.hu/about/> – RDL: 2019-02-09  
*Die Seite, auf der der Autor erläutert, warum er Elysium als Eclipse-Plugin programmiert habe.*
- Harmath, Dénes (thsoft): Elisium. LilyPond IDE for Eclipse [Sources]; o.J. [2018], FreeWeb / HTML  $\Rightarrow$  <https://github.com/thSoft/elysium> – RDL: 2019-02-09  
*Die auf Github gehosteten, unter der EPL lizenzierten Quellen des Eclipse-Plugins 'Elysium'.*
- Hilbricht, Nils: Laborejo Download; 2019  $\Rightarrow$  <https://www.laborejo.org/downloads/> – RDL: 2019-02-11  
*Die Downloadpage von Laborejo. Unglücklicherweise werden die Quellen Stand 01/2019 dort gar nicht angeboten, obwohl seine Homepage auf diese Seite verlinkt..*
- Hilbricht, Nils: Laborejo Homepage; 2019, FreeWeb / HTML  $\Rightarrow$  <https://www.laborejo.org/software.html> – RDL: 2019-02-11  
*Die Homepage von Laborejo, einem MIDI sequencer, der auf klassischer Notation zu beruhen vorgibt.*
- Hilbricht, Nils: Laborejo Manual; 2019, FreeWeb / HTML  $\Rightarrow$  <https://www.laborejo.org/documentation/laborejo/> – RDL: 2019-02-11  
*Die Manualpage von Laborejo. Unglücklicherweise besteht es – Stand 01/2019 – nur aus einem Helloworld.*
- Janssen, Thomas: MuseScore - Schnelleinstieg; 2010, FreeWeb / PDF  $\Rightarrow$  <http://jansofranso.de/texte/MUSESCORE%20%20Schnelleinstieg.pdf> – RDL: 2019-02-22  
*Ein älterer 'Crash-Kurs' in Sachen 'MuseScore', den abzuarbeiten (und an den richtigen Stellen die aktuellen Zugriffe zu finden) immer noch hilfreich ist.*
- Kenlon, Seth: Make sweet music with digital audio workstation Rosegarden; 2018, FreeWeb / HTML  $\Rightarrow$  <https://opensource.com/article/18/3/make-sweet-music-digital-audio-workstation-rosegarden> – RDL: 2019-02-24  
*Ein jüngerer Crash-Kurs in Sachen 'Rosegarden'*
- Kielhorn, Axel: The wasysym macro package for L<sup>A</sup>T<sub>E</sub>X2<sub>ε</sub>; 2003, FreeWeb / PDF  $\Rightarrow$  <http://texdoc.net/texmf-dist/doc/latex/wasysym/wasysym.pdf> – RDL: 2018-12-22  
*Übersicht über den Zeichenbestand des L<sup>A</sup>T<sub>E</sub>X-Paketes wasysym.*
- Kreussel, Peter: Neue Software; in: EasyLinux, 03 (2015), S.ubdpw.  $\Rightarrow$  <http://www.linux-community.de/ausgaben/easylinux/2015/03/neue-software-teil-1-2/2/> – RDL: 2019-01-27, FreeWeb / HTML  
*'Wiederabdruck' eines Reviews der vorletzten Canorus-Version: es sei eine leichtgewichtige Alternative zu MuseScore und erhält – trotz des sehr vorläufigen Status – 4 von Sternen.*
- Krämer, Thomas: Harmonielehre im Selbststudium; 5. Auflage. Wiesbaden, Leipzig u. Paris: Breitkopf & Härtel, 2009, Print: ISBN 978-3-7651-0261-5  
*Das 'Übungsbuch' zum neueren 'Lehrbuch der harmonischen Analyse' - sozusagen.*
- Krämer, Thomas: Lehrbuch der harmonischen Analyse. 2., verbesserte Aufl.; Wiesbaden, Leipzig u. Paris: Breitkopf & Härtel, 2012, Print: ISBN 978-3-7651-0305-6

## Literaturverzeichnis

*Ein neueres Lehrbuch, das die Zusammenhänge der Tonalität zugänglich macht.*

Kuhn, Markus: UTF-8 and Unicode FAQ for Unix/Linux; o.J. [2009], FreeWeb / HTML  
⇒ <https://www.cl.cam.ac.uk/~mgk25/unicode.html> – RDL: 2019-01-22

*Eine ausführliche Darstellung der Zusammenhänge von UCS, Unicode und UTF8:  
von der Zeichentabelle bis zur Encodierungsmethode.*

Köllerwirth, Micha: UTF-8-Codetabelle mit Unicode-Zeichen; o.J. [2015], FreeWeb  
/ HTML ⇒ <https://www.utf8-zeichentabelle.de/unicode-utf8-table.pl> –  
RDL: 2019-01-14

*Seite zum Anzeigen der Unicode-Zeichen: Die UTF-8-Musikzeichen sind zwischen  
U+1D100 bis U+1D1FF encodiert.*

Laurie, Dirk: M-Tx: Music from Text; 2016, FreeWeb / PDF ⇒ <http://ctan.space-pro.be/tex-archive/support/m-tx/doc/mtxdoc.pdf> – RDL: 2019-01-04  
*Das M-Tx-Handbuch und -Tutorial.*

Liberg, Nils: EasyABC; o.J. [2015], FreeWeb / HTML ⇒ <https://www.nilsliberg.se/ksp/easyabc/> – RDL: 2019-02-04

*Die ältere Homepage EasyABC, einem Open Source ABC Editor für Windows, OSX  
und Linux. Diese Tutorialseite erläutert, wie man ihn benutzt und woher man ihn  
bekommt. Daneben gibt es aber noch eine neuere SourceForge-Projekt-Seite.*

LilyPond Development Team: Aufsatz über den automatischen Musiksatz; o.J. [2012],  
FreeWeb / HTML ⇒ <http://lilypond.org/doc/v2.18/Documentation/essay.de.pdf> – RDL: 2019-01-08

*Eine Arbeit, in der das LilyPond-Team darstellt, was aus seiner Sicht guter und  
schlechter Notensatz ist und welche Aufgaben damit für LilyPond entstehen.*

LilyPond Development Team: LilyPond. Learning Manual; o.J. [2012], FreeWeb  
/ PDF ⇒ <http://lilypond.org/doc/v2.18/Documentation/learning.de.pdf> –  
RDL: 2019-01-05

*Das LilyPond Tutorial, das einen gelungenen Einstieg gewährleistet. Es ist auch  
als html-Seite erreichbar (<http://lilypond.org/doc/v2.18/Documentation/learning-big-page.de.html>).*

LilyPond Development Team: LilyPond. Notationsreferenz; o.J. [2012], FreeWeb  
/ PDF ⇒ <http://lilypond.org/doc/v2.18/Documentation/notation.de.pdf> –  
RDL: 2019-01-05

*Die LilyPond Notationsreferenz: sehr, sehr lang (800+x Seiten), also nicht ausdrück-  
bar, aber sehr, sehr wichtig für die Details. Deshalb gibt es diese Referenz auch als  
HTML Version (<http://lilypond.org/doc/v2.18/Documentation/notation/>).*

LilyPond Development Team: LilyPond. Allgemeine Information; o.J. [2015], FreeWeb  
/ PDF ⇒ <http://lilypond.org/doc/v2.19/Documentation/web.de.pdf> – RDL:  
2019-01-17

*Eine PDF-Datei, die den Internetauftritt von LilyPond dokumentiert und mit Seiten-  
zahlen zitierfähig macht.*

LilyPond Development Team: Leichteres Editieren; o.J. [2018], FreeWeb / HTML ⇒  
<http://lilypond.org/easier-editing.de.html> – RDL: 2019-02-21

*Eine Auflistung von LilyPond Frontends, die das Editieren von LilyPond-Dateien  
erleichtern. Besprochen werden u.A. Frescobaldi, Denemo und Elysium*

- LilyPond Development Team*: LilyPond. Notensatz für Jedermann; o.J. [2018], FreeWeb / PDF  $\Rightarrow$  <http://lilypond.org/index.de.html> – RDL: 2019-01-05  
*Die Homepage von LilyPond, einer freien Alternative zu MusixTEX, die also auch zur Erstellung und zum Druck von typographisch exzellent aufbereiteten Partituren dient.*
- LilyPond Development Team*: LilyPond. Usage; o.J. [2018], FreeWeb / PDF  $\Rightarrow$  <http://lilypond.org/doc/v2.18/Documentation/usage.de.pdf> – RDL: 2019-01-09  
*Die LilyPond Nutzungreferenz 'Usage': die kürzere Anleitung zur Nutzung von LilyPond.*
- Mansfield, Steve*: How to interpret abc music notation; 2016, FreeWeb / HTML  $\Rightarrow$  [http://www.lesession.co.uk/abc/abc\\_notation.htm](http://www.lesession.co.uk/abc/abc_notation.htm) – RDL: 2018-12-25  
*Eine Anleitung zur Interpretation der ABC-Notation.*
- Mantel, Gerhard*: Intonation; Spielräume für Streicher; Mainz, London, [... u.a.O.]: Schott, 2005 (= Studienbuch Musik), Print: ISBN 3-7957-8729-7  
*Ein Buch, das die Grenzen der systematischen Tonalität anhand der Intonation bei Streichinstrumenten verdeutlicht: um sauber zu spielen, muss man an der richtigen Stellen auf die richtige Weise falsch spielen.*
- McIntyre, D. Michael*: Learning Rosegarden. Piano Notation by Example; 2008, FreeWeb / HTML  $\Rightarrow$  <https://www.rosegardenmusic.com/tutorials/supplemental/piano/index.html> – RDL: 2019-02-24  
*Ein Rosegarden-Tutorial, das anhand eines öffentlichen MIDI-Files schrittweise alle Aspekte vom Import bis zur Notenproduktion hin erläutert: ein guter, wenn auch älterer Einstieg*
- Michels, Ulrich*: dtv-Atlas zur Musik. Tafeln und Texte; Systematischer Teil [u.] Historischer Teil: Von den Anfängen bis zur Renaissance; 5. Auflage. München, Kassel [... u.a.O.]: DTV & Bärenreiter Verlag, 1980, Print: ISBN 3-423-03022-4  
*Auch ein Standardwerk für alles, was zur systematischen Musiktheorie gehört.*
- Mittelbach, Frank u. Michel Goossens*: Der L<sup>A</sup>T<sub>E</sub>X-Begleiter; unter Mitarbeit v. J. Braams, D. Carlisle u. C. Rowley; mit Beitr. v. Christine Detig u. Joachim Schrod; 2., überarb. u. erw. Aufl.; München, Boston [... u.a.O.], 2005, Print: ISBN 3-8273-7166-X  
*Ein Standardnachschlagewerk, das auch viele 'abseitige' L<sup>A</sup>T<sub>E</sub>X-Themen ausführlich und fundiert erläutert.*
- Moine, Jean-Francois*: Jef's page. ABC music notation software; o.J. [2018], FreeWeb / HTML  $\Rightarrow$  <http://moinejf.free.fr/> – RDL: 2019-02-27  
*Die Homepage des Extrahierungs- und Konvertierungstools 'abcm2ps': Es wird Kommandozeilenprogramm bezeichnet, das ABC-Notate in das Postscript- oder SVG-Format umwandele.*
- Motte, Diether de la*: Harmonielehre; 16. Auflage. München, Kassel [... u.a.O.]: Bärenreiter Verlag & DTV, 2011, Print: ISBN 978-3-7618-2115-2  
*Nach Grabner das nächste, kommende Jahrhundertwerk zur Harmonieanalyse, das mit seiner Verlagerung des Schwerpunktes in die Funktionstheorie einen wesentlichen Fortschritt gegenüber älteren Lehrwerken bedeutete.*
- Noack, Cornelius C.*: Typesetting music with PMX; 2013, FreeWeb / PDF  $\Rightarrow$  <http://icking-music-archive.org/software/pmx/pmxccn.pdf> – RDL: 2018-12-31



## Literaturverzeichnis

*Das PMX-Handbuch, das ausführlich alle Eigenschaften und Eigenarten von PMX bespricht und ehrlicherweise auf die begrenzte Verwendbarkeit für L<sup>A</sup>T<sub>E</sub>X-Nutzer hinweist - nicht ohne allerdings auch auf einige Brücken einzugehen.*

Phillips, Dave: Music Notation Software for Linux: a Progress Report, Part 1; in: Linux Journal, (2009), S.ubdpw. ⇒ <https://www.linuxjournal.com/content/music-notation-software-linux-progress-report-part-2> – RDL: 2019-01-27, FreeWeb / HTML

*Eine ältere Sichtung von Notensatzprogrammen. Bespricht im zweiten Teil MuseScore, NtEd, NoteFlight und Outro*

Phillips, Dave: Music Notation Software for Linux: a Progress Report, Part 2; in: Linux Journal, (2009), S.ubdpw. ⇒ <https://www.linuxjournal.com/content/music-notation-software-linux-progress-report-part-1> – RDL: 2019-01-27, FreeWeb / HTML

*Eine ältere Sichtung von Notensatzprogrammen. Bespricht im ersten Teile die 'LilyPond-Connection', nämlich Denemo, Frescobaldi und Canorus*

Prakash, Abihishek: How To Use AppImage in Linux. Complete Guide; 2018, FreeWeb / HTML ⇒ <https://itsfoss.com/use-appimage-linux/> – RDL: 2019-02-22

*Eine Erläuterung des Downloadformates 'AppImage', wie es von MuseScore benutzt und angeboten wird.*

Reincke, Karsten: Dienst am Leser, Dienst am Scholaren. Über Anmerkungsapparate in Fußnoten - aber richtig: Release 2.0; (Geistes-) Wissenschaftliche Texte mit jurabib; 2018, FreeWeb / PDF ⇒ <http://kreincke.github.io/mycsrf/examples/scholar-fono-de.pdf> – RDL: 2018-12-21

*Eine selbstreferentielles Dokument, das erläutert, warum und wie man gute geisteswissenschaftliche Arbeiten mit L<sup>A</sup>T<sub>E</sub>X und mycsrf erzeugt.*

Reincke, Karsten: mycsrf; mind your Classical Scholar Research Framework; 2018, FreeWeb / HTML ⇒ <https://fodina.de/mycsrf/> – RDL: 2018-12-21

*Ein freies Framework zur Erstellung L<sup>A</sup>T<sub>E</sub>X basierter geisteswissenschaftlicher Texte, das den altphilologischen Arbeitsstil möglichst kompromisslos in die Computerwelt überträgt.*

Reincke, Karsten: harmonyli.ly . Harmonical Analysis Symbols in LilyPond Scores. Tutorial; 2019, FreeWeb / PDF ⇒ <https://github.com/kreincke/harmonyli.ly/doc/tutorial.pdf> – RDL: 2019-11-22

*Umfangreiche Einführung in die Nutzung der LilyPond Bibliothek 'harmonyli.ly' für Musikwissenschaftler*

Reincke, Karsten: Musikwissenschaft mit LaTeX. Wie man Musikbeispiele mit Open-Source-Tools in seine wissenschaftliche Texte integriert. Eine selbstreferentielle Anleitung; Eine Sichtung von Tools und Techniken um musikwissenschaftliche Texte mit LaTeX zu erzeugen. Ein Blick in die Quellen zeigt, wie man das macht, was dieses Buch vorführt und erklärt.; 2019, FreeWeb / TXT ⇒ <http://kreincke.github.io/mycsrf/examples/latex-musicology.pdf> – RDL: 2019-01-02

*Alle Quelldateien zur mycsrf-basierten Erzeugung der Tooluntersuchung Musikwissenschaft mit LaTeX.*

- Reincke, Karsten: Quellen zu Musikwissenschaft mit LaTeX; 2019, FreeWeb / TXT  
 ⇒ <https://github.com/kreincke/mycsrf/tree/master/examples/musicology.de> – RDL: 2019-01-02  
*Alle Quelldateien zur mycsrf-basierten Erzeugung der Tooluntersuchung 'Musikwissenschaft mit L<sup>A</sup>T<sub>E</sub>X'.*
- Rogers, Doug: MusEdit [Homepage]; o.J. [2011], FreeWeb / HTML ⇒ <http://dougrogers.info/websites/MusEdit/index.htm#StoryOfMusEdit> – RDL: 2019-02-03  
*Die 'neue' Homepage von MusEdit. Die Domain zu 'musedit.com' ist jedenfalls nicht mehr aktiv. Auch der Inhalt scheint alt: am Ende verkündet der Autor, dass er keine Zeit mehr habe, das Programm zu pflegen. Und am Anfang kündigt er an, die Quellen als Open-Source-Software freizugeben - was (bisher wohl) nicht geschehen ist.*
- Roitman, Alex: Editing music scores with free software; 2007, FreeWeb / HTML ⇒ <https://www.linux.com/news/editing-music-scores-free-software> – RDL: 2019-01-22  
*Eine weitere ältere Sichtung von Opensource-Notensatzsystemen: es werden LilyPond, NoteEdit, Rosegarden und Denemo erwähnt.*
- [Rosegarden Development Team]: [Rosegarden Tutorial] Notation; o.J. [2008], FreeWeb / HTML ⇒ <http://rosegarden.sourceforge.net/tutorial/en/chapter-8.html> – RDL: 2019-02-24  
*Das 8. Kapitel aus der älteren Anleitung, die eingangs von sich selbst sagt, sie sei teilweise Handbuch, teilweise Tutorial. Das 8. Kapitel ist auch heute noch hilfreich, wenn es um Spezialherausforderungen geht, wie etwa dem gleichzeitigen Öffnen mehrerer Tracks als eine Partitur.*
- [Rosegarden Development Team]: Learning Rosegarden. Rosegarden Tutorials; o.J. [2019], FreeWeb / HTML ⇒ <https://www.rosegardenmusic.com/> – RDL: 2019-02-24  
*Eine Liste von Tutorials zu ganz verschiedenen Rosegardenaspekten, die die Nutzung von Rosegarden von einer spezifischen Aufgabe her erklären.*
- [Rosegarden Development Team]: Rosegarden [Homepage]; o.J. [2019], FreeWeb / HTML ⇒ <https://www.rosegardenmusic.com/> – RDL: 2019-02-24  
*Der Einstiegspunkt in die Rosegarden-Site, auf der das Programm nicht zu Unrecht als 'music composition and editing environment based around a MIDI sequencer' bezeichnet wird, 'that features a rich understanding of music notation and includes basic support for digital audio.'*
- Rosen, Paul: ABC Notation [Wordpress Plugin]; o.J. [2018], FreeWeb / HTML ⇒ <https://de.wordpress.org/plugins/abc-notation/> – RDL: 2018-12-29  
*Als Beispiel für die Weiterverwendbarkeit der zu lernenden Notationstechnik: sie kann z.B. über ein Wordpressplugin gleichermaßen auch in einer Wordpresssite verwendet werden.*
- Schlosser, Joachim: Wissenschaftliche Arbeiten schreiben mit L<sup>A</sup>T<sub>E</sub>X. Leitfaden für Einsteiger. 6., überarb. Aufl.; Frechen: mitp Verlag, 2016, Print: ISBN 978-3-95845-289-3  
*Eine sehr praxisnahe Einführung in die Nutzung von L<sup>A</sup>T<sub>E</sub>X als Werkzeug zum Schrei-*

## Literaturverzeichnis

*ben (wissenschaftlicher) Texte. Die durchaus spannende Frage, wie und warum das Wissenschaftliche das Schreiben wissenschaftlicher Texte prägt, bleibt ungestellt.*

Shann, Richard: Denemo User Manual; 2015, FreeWeb / PDF  $\Rightarrow$  <http://git.savannah.gnu.org/gitweb/?p=denemo.git;f=docs/denemomanual.pdf;hb=HEAD> – RDL: 2019-02-22

*Das Denemo-Handbuch: schwierig zu lesen, schwer zu verstehen.*

Shann, Richard: Denemo User Manual (online); 2015, FreeWeb / HTML  $\Rightarrow$  <http://www.denemo.org/~rshann/denemo-manual.html> – RDL: 2019-02-22

*Das Denemo-Handbuch als Online-Version: ebenfalls schwierig zu lesen, aber leichter zu durchsuchen.*

Simons, Don: PMX – a Preprocessor for MusiX $\TeX$ ; [Handbuch]; 2017, FreeWeb / PDF  $\Rightarrow$  <http://ftp.uni-erlangen.de/ctan/support/pmx/doc/pmx284.pdf> – RDL: 2019-01-04

*Das PMX-Handbuch des PMX-Paketes.*

[SourceForge]: Rosegarden [Repository]; o.J. [2019], FreeWeb / HTML  $\Rightarrow$  <https://sourceforge.net/projects/rosegarden/> – RDL: 2019-02-24

*Das Rosegarden-SourceForge-Repository*

Spencer-Jowett, Steve: The ABCJ Page; o.J., FreeWeb / HTML  $\Rightarrow$  <http://abcj.ganderband.com/> – RDL: 2019-02-04

*Die ABCJ Homepage. Sie stellt alles auf einer Seite bereit: eine Beschreibung, den Download etc. etc. Aussehen und Funktionalität der Homepage lassen darauf schließen, dass ABCJ selbst ein älteres Programm ist,*

Tennent, Bob: MusiX $\TeX$  and Related Software; o.J., FreeWeb / HTML  $\Rightarrow$  [https://icking-music-archive.org/software/htdocs/More\\_Related\\_Software.html](https://icking-music-archive.org/software/htdocs/More_Related_Software.html) – RDL: 2019-01-17

*Tools zur Editierung und Auswertung von MusiX $\TeX$ -Musiknotationen.*

Tennent, Bob: MusiX $\TeX$  and More Related Software; o.J., FreeWeb / HTML  $\Rightarrow$  <https://icking-music-archive.org/software/htdocs/> – RDL: 2019-01-17

*Eine generelle Einführung in die Nutzung von MusiX $\TeX$  und verwandte Software.*

Thoma, Martin: How to write music with LaTeX; 2012, FreeWeb / PDF  $\Rightarrow$  <https://martin-thoma.com/how-to-write-music-with-latex/> – RDL: 2018-12-19

*Ein kleiner Artikel, der verschiedene Möglichkeiten erläutert, Noten in L $\TeX$ -Dokumente zu integrieren.*

Vogel, Oliver *et al.*: MusiX $\TeX$ . Using  $\TeX$  to write polyphonic or instrumental music; Version 1.28. [Revised File]; 2018, FreeWeb / PDF  $\Rightarrow$  <http://texdoc.net/texmf-dist/doc/generic/musixtex/musixdoc.pdf> – RDL: 2018-12-08

*Das Standardhandbuch zum Erlernen von MusiX $\TeX$ : Es zeigt, wie man MusiX $\TeX$ -Dokumenten mit Texteditoren erstellt. Dem Kenner macht es auch klar, wie man Ergebnisse in die L $\TeX$ -Welt überträgt. Dem Musiker, der kein Computerexperte ist, verlangt es aber einiges ab.*

Voß, Herbert: Einführung in L $\TeX$  unter Berücksichtigung von pdfL $\TeX$ , XeL $\TeX$  und LuaL $\TeX$ ; Berlin: Lehmanns Media, 2012 (= dante), Print: ISBN 978-3-86541-462-5

*Eine gründliche Einführung von einem echten Kenner der Materie.*



Vofß, Herbert: Die wissenschaftliche Arbeit mit L<sup>A</sup>T<sub>E</sub>X; unter Verwendung von LuaT<sub>E</sub>X, KOMA-Script und Biber/BibL<sup>A</sup>T<sub>E</sub>X; Berlin: Lehmanns Media, 2018 (= dante), Print: ISBN 978-3-86541-946-0

*Eine sehr praktische Einführung in L<sup>A</sup>T<sub>E</sub>X, die sich auf die Erstellung wissenschaftlicher Texte konzentriert. Einziger kleiner Wermutstropfen ist die eher mathematisch orientierte Vorgabe, wie wissenschaftliche Texte auszusehen hätten. Die Geisteswissenschaft ist – wie so oft – irgendwie nur 'mitgemeint'.*

Walshaw, Chris: abc notation home page; o.J. [2018], FreeWeb / HTML ⇒ <http://abcnotation.com/> – RDL: 2018-12-25

*Homepage der abc Notation - das text basierte Musiknotationssystem, das von sich beansprucht, der de-facto-Standard für Folk- und Volksmusik zu sein - erlaubt produktiv allerdings nur einzeilige Notate.*

Walshaw, Chris: abc software packages; o.J. [2018], FreeWeb / HTML ⇒ <http://abcnotation.com/software> – RDL: 2018-12-25

*Eine Liste von Tools zur Editierung und Auswertung von abc-Musiknotationen: Die Fülle der Verwertungstools für die unterschiedlichsten Zwecke und Systeme erzeugt ein komplexes Musikverwertungssystem.*

Wegner, Dagny u. Arnim Wegner: harmony.sty; 2007, FreeWeb / PDF ⇒ <ftp://ftp.dante.de/pub/tex/macros/latex/contrib/harmony/harmony.pdf> – RDL: 2018-12-21

*Das Handbuch zur Nutzung des freien Paketes 'harmony', mit dem dem L<sup>A</sup>T<sub>E</sub>X-Autoren die Zeichen für eine adäquaten Harmonieanalyse bereitgestellt werden.*

[Wikipedia]: Sequencer; o.J. [2018], FreeWeb / HTML ⇒ [https://de.wikipedia.org/wiki/Sequencer\\_\(Musik\)](https://de.wikipedia.org/wiki/Sequencer_(Musik)) – RDL: 2019-02-24

*Eine kurze und klare Erläuterung, was Sequencer ausmacht und in welchem Zusammenhang sie zu Notensatzprogrammen stehen.*

[Wikipedia]: Rosegarden; o.J. [2019], FreeWeb / HTML ⇒ <https://de.wikipedia.org/wiki/Rosegarden> – RDL: 2019-02-24

*Die deutsche Wikipediaseite von Rosegarden, die auch einige Details über die sehr lange Entwicklungsgeschichte von Rosegarden enthält*