LilyPond

Das Notensatzprogramm

Programmbenutzung

Das LilyPond-Entwicklerteam

Für LilyPond Version 2.13.27

Inhaltsverzeichnis

1 Installieren

Es gibt von Lilypond zwei verschiedene Typen von Versionen: stabile Versionen und instabile Entwicklerversionen. Stabile Versionen haben eine gerade Unter-Versionsnummer (z.B. 2.8, 2.10, 2.12, etc.). Entwicklerversionen haben hingegen ungerade Unter-Versionsnummern (z.B. 2.7, 2.9, 2.11, etc.).

LilyPond selbst zu kompilieren ist ein relativ komplizierter Vorgang, daher empfehlen wir **stark**, die vorkompilierten Binär-Pakete zu benutzen.

1.1 Vorkompilierte Binär-Pakete

Herunterladen

Aktuelle Informationen zu vorkompilierten Binär-Paketen für Ihre Plattform finden Sie unter `http://lilypond.org/web/install/`. Falls Ihr Betriebssystem auf dieser allgemeinen Seite nicht behandelt wird, sehen Sie bitte in der vollständigen Liste unter `http://download.linuxaudio.org/lilypond/binaries/` nach.

Momentan werden Binärpakete für folgende Plattformen erstellt:

```
darwin-ppc  - MacOS X powerpc
darwin-x86  - MacOS X intel
freebsd-64  - FreeBSD 6.x, x86_64
freebsd-x86 - FreeBSD 4.x, x86
linux-64    - Beliebige GNU/Linux Distribution, x86_64
linux-ppc   - Beliebige GNU/Linux Distribution, powerpc
linux-x86   - Beliebige GNU/Linux Distribution, x86
mingw       - Windows x86
```

Bekannte Probleme und Warnungen

Wenn Sie MacOS 10.3 oder 10.4 benutzen und Python-Skripte wie `convert-ly` und `lilypond-book` benutzen wollen, lesen Sie Abschnitt “Setup for MacOS X” in *Application Usage*.

1.2 Aus den Quellen übersetzen

LilyPond kann auch selbst direkt aus dem Quellcode des git-Depots kompiliert werden. Da jedoch für die Kompilierung definitiv Englisch-Kenntnisse vorhanden sein müssen, soll hier lediglich auf die englische Dokumentation verwiesen werden: Abschnitt “Compiling from source” in *Application Usage*.

2 Setup

In diesem Kapitel werden verschiedene Konfigurationsoptionen für LilyPond und andere Programme behandelt, die nach der Installation auszuführen sind. Dieses Kapitel kann als Referenz betrachtet werden, lesen Sie einen Abschnitt nur, wenn er auf Sie zutrifft.

2.1 Setup für bestimmte Betriebssysteme

Dieser Abschnitt erklärt zusätzliche Einstellungen für bestimmte Betriebssysteme.

2.1.1 Einrichtung für MacOS X

Benutzung von Python-Skripten unter MacOS 10.3 oder 10.4

Das LilyPond-Programmpaket für MacOS X stellt Python nicht zur Verfügung, aber Python 2.4 oder höher wird benötigt, um Skripte wie `convert-ly` zu benutzen. Wenn Sie MaxOS 10.3 oder 10.4 besitzen, müssen Sie eine neurere Python-Version von http://python.org/download/ installieren und dann die erste Zeile von `convert-ly` und `lilypond-book` auf folgende Weise ändern: Wenn sich das installierte Python-Programm in Ihrem *PFAD* befindet, muss die erste Zeile heißen:

```
#!/usr/bin/env python
```

andernfalls muss sie heißen:

```
#!/path/to/newly_installed/python
```

MacOS X auf der Kommandozeile

Die Skripte (etwa `lilypond-book`, `convert-ly`, `abc2ly` und sogar `lilypond`) befinden sich innerhalb der `.app`-Datei für MacOS X. Sie können auf der Kommandozeile direkt aufgerufen werden, etwa

```
Pfad/zu/LilyPond.app/Contents/Resources/bin/lilypond
```

Entsprechend funktionieren auch die anderen Skripte wie `lilypond-book`, `convert-ly`, `abc2ly` usw.

Sie können sich auch selber Skripte anlegen, die diesen Pfad automatisch hinzufügen. Erstellen Sie ein Verzeichnis, indem die Skripte gespeichert werden:

```
mkdir -p ~/bin
cd ~/bin
```

Erstellen Sie eine Datei mit dem Namen `lilypond` und dem Inhalt

```
exec path/to/LilyPond.app/Contents/Resources/bin/lilypond "$@"
```

Erstellen Sie entsprechende Dateien mit den Namen `lilypond-book`, `convert-ly` und den Namen der anderen Hilfsprogramme, die Sie benutzen (`abc2ly`, `midi2ly` usw.). Ersetzen Sie einfach den Teil `bin/lilypond` mit `bin/convert-ly` (oder einem anderen Programmnamen) in der entsprechenden Datei.

Machen Sie die Datei ausführbar:

```
chmod u+x lilypond
```

Jetzt müssen Sie dieses Verzeichnis noch zu Ihrem Pfad (PATH) hinzufügen. Verändern Sie die Datei `.profile` in Ihrem Benutzerverzeichnis (oder erstellen Sie sie), dass sie die Zeile

```
export PATH=$PATH:~/bin
```

beinhaltet. Die Datei muss mit einer Leerzeile enden.

Beachten Sie, dass *Pfad/zu* üblicherweise `/Applications/` ist.

2.2 Unterstützung von Texteditoren

Verschiedene Texteditoren haben Unterstützung für LilyPond.

2.2.1 Emacs-Modus

Emacs hat einen LilyPond-Modus (‘`lilypond-mode`’), eine Datei, die die Vervollständigung von Befehlen, Einrückungen, für LilyPond spezifische Klammerschließungen und die Markierung der Syntax beherrscht. Zusätzlich stehen noch praktische Tastaturkombinationen zum Programmaufruf und zum Nachschlagen in den Handbüchern zur Verfügung. Siehe unten, wenn die Datei ‘`lilypond-mode`’ sich nicht auf Ihrem Computer befindet.

Der Emacs-Modus zur Noteneingabe und zum Programmaufruf ist in den Emacs-Quellen im ‘`elisp`’-Verzeichnis enthalten. Mit dem Befehl `make install` können Sie es nach *elispdir* installieren. Die Datei ‘`lilypond-init.el`’ sollte in *load-path*‘`/site-start.d/`’ liegen oder Ihrem ‘`~/.emacs`’ or ‘`~/.emacs.el`’ hinzugefügt werden.

Als Benutzer können Sie Ihren Quellenpfad (etwa ‘`~/site-lisp/`’) ihrem *load-path* hinzufügen, indem Sie folgende Zeile zu Ihrer ‘`~/.emacs`’-Datei hinzufügen:

```
(setq load-path (append (list (expand-file-name "~/site-lisp")) load-path))
```

2.2.2 Vim-Modus

Für VIM wird ein ‘`vimrc`’ bereitgestellt, zusammen mit Werkzeugen zur Syntaxauszeichnung. Ein Vim-Modus zur Noteneingabe und zum Programmaufruf befindet sich im Quellarchiv im `$VIM`-Verzeichnis.

LilyPond-Dateien werden automatisch erkannt, wenn sich in der Datei ‘`~/.vim/filetype.vim`’ folgender Inhalt befindet:

```
if exists("did_load_filetypes")
  finish
endif
augroup filetypedetect
  au! BufNewFile,BufRead *.ly,*.ily           setf lilypond
augroup END
```

Fügen Sie den LilyPond-Pfad in ihre ‘`~/.vimrc`’-Datei ein, indem Sie ihre folgende Zeile hinzufügen:

```
set runtimepath+=/usr/local/share/lilypond/${LILYPOND_VERSION}/vim/
```

Dabei wird ${LILYPOND_VERSION}durch Ihre LilyPond-Version ersetzt. Sollten Sie LilyPond nicht nach ‘`/usr/local/`’ installiert haben, müssen Sie den Pfad entsprechend anpassen.

2.2.3 jEdit

Das Plugin für den jEdit-Texteditor, LilyPondTool genannt, ist das vielfältigste textbasierte Werkzeug, um LilyPond-Notationsdateien zu editieren. Zu den besonderen Eigenschaften gehört ein Dokument-Ersteller mit Liedtextunterstützung, der die Neuerstellung von Dateien erleichtert, und ein integriertes PDF-Anzeigeprogramm mit „point-and-click“-Unterstützung. Demos, Bildschirmphotos und Installationsanweisungen finden sich unter `http://lilypondtool.organum.hu`

2.2.4 TexShop

Der TexShop-Editor für MacOS X kann erweitert werden, um LilyPond, lilypond-book und convert-ly aus dem Editor heraus zu starten. Die Erweiterung findet sich unter `http://www.dimi.uniud.it/vitacolo/freesoftware.html`.

2.2.5 TextMate

Es gibt ein LilyPond-Paket für TextMate. Es kann installiert werden, indem man

```
mkdir -p /Library/Application\ Support/TextMate/Bundles
cd /Library/Application\ Support/TextMate/Bundles
svn co http://macromates.com/svn/Bundles/trunk/Bundles/Lilypond.tmbundle/
```

ausführt.

2.2.6 LilyKDE

LilyKDE ist eine Erweiterung für den Texteditor Kate für KDE. LilyKDE beinhaltet einen mächtigen Assistenten, mit dem sehr schnell neue LilyPond-Dokumente erstellt werden könne, sowie einen eingebetteten PDF-Betrachter.

LilyKDE benutzt auch Rumor, um Musik direkt von einem MIDI-Keyboard einzulesen.

Weitere Fähigkeiten von LilyKDE sind Silbentrennung für Liedtexte oder die Möglichkeit, LilyPond gleichzeitig mit mehreren Eingabedateien vom KDE Dateimanager aus aufzurufen.

2.3 Point and click

Point and click erlaubt es, die Noten in der Quelldatei zu finden, indem man sie im PDF anklickt. Das erleichtert es, fehlerhafte Stellen zu finden.

Wenn diese Funktionalität aktiv ist, fügt LilyPond Hyperlinks zum PDF hinzu. Diese werden dann bei einem Klick zum Browser geschickt, der wiederum einen Texteditor mit dem Cursor an der richtigen Stelle öffnet.

Damit diese Kettenreaktion funktioniert, müssen Sie ihrem PDF-Programm mit dem Skript '`lilypond-invoke-editor`' beibringen, Hyperlinks zu folgen.

Im Falle von Xpdf auf einer UNIX-Maschine sollte folgende Zeile in der Datei '`xpdfrc`'[1] stehen.

```
urlCommand     "lilypond-invoke-editor %s"
```

Das Programm '`lilypond-invoke-editor`' ist ein kleines Hilfsprogramm. Es ruft einen Editor auf für bestimmte `textedit`-URIs und startet einen Browser für die anderen. Die Umgebungsvariable `EDITOR` wird für folgende Zeichenketten:

`emacs` das startet

```
emacsclient --no-wait +line:column file
```

`vim` das startet

```
gvim --remote +:line:normchar file
```

`nedit` das startet

```
nc -noask +line file'
```

Die Umgebungsvariable `LYEDITOR` wird verwendet, um dieses zu überschreiben. Sie enthält den Befehl, um den Editor aufzurufen, wobei die Variablen `%(file)s`, `%(column)s`, `%(line)s` mit der Datei, Spalte und Zeile ersetzt werden. Die Einstellung

```
emacsclient --no-wait +%(line)s:%(column)s %(file)s
```

für `LYEDITOR` etwa entspricht dem von vornherein eingestellten Emacs-Aufruf.

Die point and click-Links vergrößern die Notationsdatei erheblich. Um die Größe der PDF- und PS-Dateien zu verringern, kann point and click ausgeschaltet werden, indem die Zeile

[1] Unter UNIX befindet sich diese Datei entweder in '`/etc/xpdfrc`' oder als '`.xpdfrc`' in Ihrem Heim-Verzeichnis.

```
\pointAndClickOff
```

in der '`.ly`'-Datei gesetzt wird. Point and Click kann auch explizit eingeschaltet werden durch die Zeile:

```
\pointAndClickOn
```

Alternativ können Sie Point and Click auch mit einer Kommandozeilenoption ausschalten:

```
lilypond -dno-point-and-click file.ly
```

> **Achtung:** Sie sollten Point and Click immer für Dateien ausschalten, die Sie an andere Personen weitergeben möchten. Anderenfalls werden Pfadinformationen Ihres Computers in die PDF-Datei kopiert, was ein potentielles Sicherheitsrisiko darstellt.

3 lilypond starten

Dieses Kapitel behandelt die technischen Details, wie Lilypond ausgeführt werden kann.

3.1 Übliche Programmbenutzung

Die meisten Benutzer führen LilyPond von einer graphischen Benutzeroberfläche aus. Siehe FIXME falls Sie dies nicht bereits getan haben.

3.2 Benutzung auf der Kommandozeile

Dieser Abschnitt enthält zusätzliche Informationen, wie Sie LilyPond von der Kommandozeile ausführen können. Dies kann erforderlich sein, um etwa zusätzliche Optionen an das Programm zu übergeben. Außerdem sind einige Zusatzprogramme (wie etwa `midi2ly`) nur von der Kommandozeile verfügbar.

Unter ‚Kommandozeile' verstehen wir die Kommandozeile des jeweiligen Betriebssystems. Windows Benutzern ist sie vielleicht eher unter den englischen Begriffen ‚DOS shell' oder ‚command shell' bekannt. MacOS X Benutzer kennen sie eher unter ‚Terminal' oder ‚Konsole'. Sie sollten auch den Abschnitt FIXME konsultieren.

Wie die Kommandozeile im jeweiligen Betriebssystem benutzt werden kann, soll in diesem Handbuch nicht näher beschrieben werden. Sehen Sie bitte im Handbuch Ihres Betriebssystems nach oder informieren Sie sich im Internet, wenn Sie mit der Kommandozeile nicht vertraut sind.

3.2.1 lilypond aufrufen

Das `lilypond` Programm kann folgendermaßen von der Kommandozeile aufgerufen werden.

```
lilypond [Option]... Dateiname...
```

Wird ein '`Dateiname`' ohne Erweiterung angegeben, so wird '`.ly`' als Standarderweiterung für LilyPond-Dateien benutzt. Um Daten von `stdin` einzulesen, benutzen Sie einfach einen Bindestrich (`-`) als *Dateiname*.

Wenn Lilypond die Datei '`Dateiname.ly`' verarbeitet, werden daraus die Dateien '`Dateiname.ps`' und '`Dateiname.pdf`' erzeugt. Es können an `lilypond` auch mehrere '`.ly`' Dateien übergeben werden, die dann einzeln und voneinander unabhängig abgearbeitet werden.[1]

Falls '`Dateiname.ly`' mehr als einen `\score`-Block enthält, werden die weiteren Stücke in durchnummerierte Dateien der Form '`Dateiname-1.pdf`' ausgegeben. Zusätzlich wird der Wert der Variable `output-suffix` zwischen den ursprünglichen Dateienamen und der Zahl eingefügt. Eine Lilypond-Datei *Dateiname.ly* mit dem Inhalt

```
#(define output-suffix "Geige")
\score { ... }
#(define output-suffix "Cello")
\score { ... }
```

erzeugt daher die Dateien *Dateiname*'`-Geige.pdf`' und *Dateiname*'`-Cello-1.pdf`'.

3.2.2 Optionen auf der Kommandozeile für lilypond

Die folgenden Kommandozeilenoptionen werden von `lilypond` unterstützt:

`-e,--evaluate=`*expr*

Wertet den Scheme-Ausdruck *expr* aus, bevor die '`.ly`' Dateien gelesen und interpretiert werden. Die `-e` Option kann auch mehrfach angegeben werden, die Ausdrücke werden nacheinander ausgewertet.

[1] Der Zustand von GUILE wird allerdings nicht nach jeder Datei zurückgesetzt, sodass Achtung geboten ist, wenn in einer Datei globale Änderungen von Scheme aus durchgeführt werden.

Da der Ausdruck im `guile-user` Modul ausgewertet wird, ist bei der Definitionen innerhalb von *expr* folgendes Vorgehen nötig. An der Kommandozeile wird z.B. `a` im `guile-user` Modul definiert:

```
lilypond -e '(define-public a 42)'
```

Am Beginn der '`.ly`'-Datei muss dann das `guile-user` Modul noch geladen werden, bevor die Definition von *a* verfügbar ist:

```
#(use-modules (guile-user))
```

`-f,--format=`*Format*

Bestimmt das Ausgabeformat. Mögliche Werte von *Format* sind `svg`, `ps`, `pdf` und `png`.

Beispiel: `lilypond -fpng` *Dateiname*`.ly`

`-d,--define-default=`*Variable*`=`*Wert*

Damit wird die interne Programmoption *Variable* auf den Scheme-Wert *Wert* gesetzt. Wird kein *Wert* angegeben, so wird *#t* benutzt. Um eine Option auszuschalten, kann der Präfix `no-` dem Namen *Variable* der Variable vorangestellt werden. So ist etwa

```
-dno-point-and-click
```

dasselbe wie

```
-dpoint-and-click='#f'
```

Hier sind ein paar interessante Optionen:

'`help`' Die Ausführung von `lilypond -dhelp` zeigt alle verfügbaren `-d` Optionen.

'`paper-size`'

Setzt das Standard-Papierformat,

```
-dpaper-size=\"letter\"
```

Die Zeichenkette, die das Format angibt, muss in Anführungszeichen mit Backslash (`\"`) stehen.

'`safe`' Vertraut der `.ly` Datei nicht.

Wenn LilyPond über einen Webserver verfügbar gemacht wird, **MUSS** unbedingt eine die Optionen `--safe` oder `--jail` angegeben werden. Die `--safe` Option verhindert, dass in der .ly-Datei angegebener Scheme-Code das System gefährden kann, wie etwa in folgendem Beispiel:

```
#(system "rm -rf /")
{
  c4^#(ly:export (ly:gulp-file "/etc/passwd"))
}
```

Mit der `-dsafe` Option werden alle Scheme-Ausdrücke einem speziellen sicheren Modus ausgewertet. Dieser Modus ist vom GUILE '`safe-r5rs`' Modul abgeleitet und fügt noch zahlreiche weitere erlaubte Funktionen der LilyPond Programm-Schnittstelle hinzu. Diese Funktionen sind in '`scm/safe-lily.scm`' angegeben.

Zusätzliche verbietet der sichere Modus auch `\include` Befehle.

Im sicheren Modus ist es nicht möglich, LilyPond-Variablen nach Scheme zu exportieren.

`-dsafe` erkennt jedoch *KEINE* Überbeanspruchung der verfügbaren Ressourcen. In diesem Modus ist es also trotzdem möglich, dass LilyPond in einer Endlosschleife hängt, z.B. wenn zyklische Datenstrukturen an das Backend übergeben werden. Wenn LilyPond also auf einem öffentlich zugänglichen Webserver verfügbar gemacht wird, sollte der Prozess sowohl in der CPU- als auch in der Speichernutzung limitiert werden.

Der sichere Modus verhindert auch, dass zahlreiche nützliche Musikfragmente von LilyPond verarbeitet werden. Die `--jail` Option ist eine sicherere Alternative, benötigt allerdings auch mehr Aufwand zur Einrichtung.

'`backend`' Gibt an, welches Ausgabeformat das LilyPond Backend benutzt. Mögliche Werte für diese Option sind:

`ps` PostScript-Ausgabeformat.

Postscript-Dateien enthalten auch TTF-, Type1- und OTF-Schriften. Allerdings wird die gesamte Schriftart eingefügt und nicht nur die benötigten Zeichen. Vor allem wenn nichtwestliche Zeichensätze benutzt werden, kann dies zu sehr großen Dateien führen.

`eps` Erzeugt ‚encapsulated PostScript' (EPS). Jede Seite (oder jedes System) wird als eigene '`EPS`'-Datei ausgegeben, inklusive Schriftarten. Außerdem wird eine Datei mit allen Seiten (bzw. Systemen) und Schriftarten erzeugt.

Dies ist die Standardeinstellung von `lilypond-book`.

`svg`

SVG-Ausgabe (Scalable Vector Graphics).

Hiermit wird eine einzelne SVG-Datei ohne eingebundene Schriften für jede Seite der Partitur erstellt. Es wird empfohlen, Century Schoolbook-Schriftarten zu installieren, die auch in der LilyPond-Installation enthalten sind, um optimales Rendern zu erhalten. Unter UNIX können diese Schriftarten einfach aus dem LilyPond-Verzeichnis (normalerweise '`/usr/share/lilypond/VERSION/fonts/otf/`') nach '`~/.fonts`' kopiert werden. Die SVG-Ausgabe sollte mit allen SVG-Editoren oder Betrachtungsprogrammen kompatibel sein.

`scm` gibt die rohen Scheme-basierenden Zeichenbefehle aus, wie sie intern von LilyPond benutzt werden.

`null` Keine Partitur wird ausgegeben, hat gleichen Effekt wie `-dno-print-pages`.

Beispiel: `lilypond -dbackend=svg` *`Dateiname`*`.ly`

'`preview`' Erzeugt eine Ausgabedatei, die nur die Titelzeilen und das erste System enthält.

'`print-pages`'
Erzeugt vollständige Seiten (Standardeinstellung). `-dno-print-pages` ist in Verbindung mit `-dpreview` nützlich.

`-h,--help`
: Zeigt eine Zusammenfassung der Programmbenutzung und der Optionen.

`-H,--header=`*`FELD`*
: Gibt den Inhalt eines Feldes aus dem `\header`-Block in die Datei '`Dateiname.`*`FELD`*' aus.

`--include, -I=`*`Verzeichnis`*
: Fügt *Verzeichnis* zur Liste der Suchpfade hinzu.

`-i,--init=`*`Initialisierungsdatei`*
: Benutzt *Initialisierungsdatei* zur gesamten Programminitialisierung. Der Standardwert ist '`init.ly`'.

`-o,--output=`*`DATEI`*
: Schreibt das Ergebnis der Verarbeitung mit LilyPond in die Ausgabedatei *DATEI*. Die entsprechende Dateinamenserweiterung wird angehängt (z.B. `.pdf` für pdf).

`--ps` Erzeugt PostScript.

`--png` Erzeugt eine Grafik-Datei im PNG-Format von jeder Seite. Diese Option impliziert auch `--ps`. Die Auflösung in DPI der Grafik kann festgelegt werden durch

```
-dresolution=110
```

`--pdf` Erzeugt PDF-Dateien. Dies impliziert `--ps`.

`-j,--jail=`*`Benutzer,Gruppe,Jail-Verzeichnis,Arbeitsverzeichnis`*
: Führt `lilypond` in einem chroot-Jail aus.

  Die `--jail` Option ist eine flexiblere Alternative zu `--safe`, wenn LilyPond über das Internet verfügbar gemacht wird oder LilyPond-Quelldateien von Dritten automatisch vararbeitet werden.

  Sie funktioniert dergestalt, dass das Wurzelverzeichnis von `lilypond` auf *Jail-Verzeichnis* gesetzt wird, bevor die tatsächliche Kompilierung der .ly-Datei beginnt. Der Benutzer und die Gruppe werden auf die angegebenen Werte gesetzt und das aktuelle Arbeitsverzeichnis wird ebenfalls auf den angegebenen Wert *Arbeitsverzeichnis* gesetzt. Diese Einstellungen garantieren (zumindest in der Theorie), dass es nicht möglich ist, aus dem Jail auszubrechen. Damit `--jail` funktioniert, muss `lilypond` allerdings als root ausgeführt werden, was normalerweise auf sichere Art mit dem Kommando `sudo` erreicht werden kann.

  Das Jail-Verzeichnis zu erstellen ist etwas heikel, da LilyPond alle zur Ausführung nötigen Bibliotheken und Dateien *innerhalb des Jail-Verzeichnisses* finden muss. Ein typisches Setup besteht aus folgenden Punkten:

  Erstellung eines getrennten Dateisystems
  : Ein eigenes Dateisystem muss für LilyPond erstellt werden, sodass es mit sicheren Einstellungen wie `noexec`, `nodev` und `nosuid` eingebunden werden kann. Damit ist es unmöglich, Programme von diesem Dateisystem auszuführen oder direkt auf eine Hardware-Schnittstelle zuzugreifen. Wenn Sie keine eigene Partition erstellen möchten, können Sie auch eine Datei der entsprechenden Größe erstellen und sie als ‚loop'-Gerät einbinden. Ein getrenntes Dateisystem garantiert auch, dass LilyPond nicht mehr Festplattenspeicher benutzt als erlaubt.

  Erstellung eines eigenen Benutzerkontos
  : Es sollte ein eigener Benutzer und eine eigene Gruppe (z. B. `lily/lily`) mit geringen Rechten für die Ausführung von LilyPond innerhalb des Jails benutzt werden. Nur ein einziges Verzeichnis des Jails sollte für

den Benutzer schreibbar sein und als *Arbeitsverzeichnis* an `lilypond` übergeben werden.

Einrichtung des Jails
: LilyPond muss zahlreiche Dateien für die Ausführung einlesen. All diese Dateien müssen in das Jail-Verzeichnis kopiert werden (mit denselben Pfaden wie im tatsächlichen Wurzel-Dateisystem). Die gesamte LilyPond-Installation (typischerweise '`/usr/share/lilypond`') sollte kopiert werden.

  Falls Probleme auftreten, ist es am einfachsten, Lilypond mittels `strace` zu starten, wodurch Sie relativ leicht feststellen können, welche Dateien im Jail noch fehlen.

Ausführung von LilyPond
: In einem mit `noexec` eingebundenen Jail ist es nicht möglich, externe Programme auszuführen. Daher muss LilyPond auf eine Art gestartet werden, die keine weitere Ausführung von Programmen benötigt. Wie bereits erwähnt muss LilyPond mit Administrator-Rechten gestartet werden (die es allerdings sofort wieder abgibt), beispielsweise mittels `sudo`. Außerdem ist es eine gute Idee, die LilyPond zur Verfügung stehende CPU-Zeit zu limitieren (z. B. mit `ulimit -t`) und – falls das Betriebssystem dies unterstützt – auch den zur Verfügung stehenden Hauptspeicher.

`-v,--version`
: Gibt die Versionsnummer aus.

`-V,--verbose`
: Gibt ausführliche informative Meldungen aus: Zeigt die vollen Dateipfade aller gelesenen Dateien sowie Informationen über die Laufzeit.

`-w,--warranty`
: Zeigt die Garantiebedingungen an, unter denen GNU LilyPond steht. (Es besteht **KEINERLEI GARANTIE**!)

3.2.3 Umgebungsvariablen

`lilypond` erkennt und benützt die folgenden Umgebungsvariablen:

`LILYPOND_DATADIR`
: Diese Variable gibt das Verzeichnis an, wo Lilypond seine eigenen Dateien, Meldungen und Übersetzungen finden kann. Dieses Verzeichnis sollte Unterverzeichnisse '`ly/`', '`ps/`', '`tex/`', etc. beinhalten.

`LANG` Gibt die Sprache an, in der Warnungen und Fehlermeldungen ausgegeben werden.

`LILYPOND_GC_YIELD`
: Mit dieser Variable (mit Werten zwischen 0 und 100) kann die Feinabstimmung zwischen dem Bedarf an Hauptspeicher und Rechenleistung bei der Ausführung von LilyPond durchgeführt werden. Bei höheren Werten benutzt LilyPond mehr Hauptspeicher, benötigt aber weniger Prozessor-Leistung. Bei niedrigeren Werten wird mehr Prozessor-Zeit, dafür weniger Hauptspeicher benötigt. Voreinstellung ist ein Wert von `70`.

3.3 Fehlermeldungen

Während der Verarbeitung einer Dateien können diverse Meldungen an der Kommandozeile auftreten:

Warnung Irgendetwas ist verdächtig. Wenn Sie etwas Ungewöhnliches in Ihrer Datei durchführen, dann werden Sie die Meldung verstehen und können sie gegebenenfalls ignorieren. Im Normalfall jedoch bedeutet eine Warnung, dass mit Ihrer Datei etwas nicht stimmt, LilyPond jedoch trotzdem versucht, die Datei soweit wie möglich korrekt zu übersetzen.

Fehler Irgendetwas stimmt definitiv nicht. Der aktuelle Bearbeitungsschritt (Einlesen, Interpretieren oder Formatieren der Datei) wird noch fertig ausgeführt, danach bricht die Bearbeitung aber ab.

Fataler Fehler
Irgendetwas stimmt definitiv nicht und LilyPond kann nicht weiter ausgeführt werden. Dies ist nur sehr selten der Fall, meist sind die Schriftarten nicht korrekt installiert.

Scheme Fehler
Fehler, die während der Ausführung von Scheme-Code auftreten, werden vom Scheme-Interpreter aufgefangen und an der Kommandozeile ausgegeben. Wenn Sie LilyPond mit der `--verbose` Option (auch `-V`) ausführen, wird der sogenannte ‚Call trace' ausgegeben, der die aufgerufenen Funktionen zur Zeit des Fehlers angibt.

Programmierfehler
Eine interne Inkonsistenz ist aufgetreten. Diese Fehlermeldungen sollen den Programmierern die Fehlersuche erleichtern und können meistens einfach ignoriert werden. In manchen Fällen werden so viele Meldungen ausgegeben, dass die Lesbarkeit der restliche Ausgabe davon beeinträchtigt wird.

Abgebrochen (core dumped)
Dies bezeichnet einen ernsten Programmierfehler, der das Programm zum Absturz gebracht hat. Solche Fehler werden als kritisch angesehen. Falls daher einer auftritt, senden Sie bitte einen Bug-Report!

Wenn Warnungen oder Fehlermeldungen mit einer konkreten Stelle in der Eingabedatei verknüpft werden können, dann hat die Meldung die folgende Form:

```
Dateiname:Zeile:Spalte: Meldung
Fehlerhafte Eingabezeile
```

Ein Zeilenumbruch wird in der fehlerhaften Zeile an jener Stelle eingefügt, wo der Fehler aufgetreten ist. Zum Beispiel

```
test.ly:2:19: Fehler: keine gültige Dauer: 5
  { c'4 e'
             5 g' }
```

Diese Stellen sind LilyPonds Vermutung, wo die Warnung oder der Fehler aufgetreten ist, allerdings treten Warnungen und Fehler ja gerade in unerwarteten Fällen auf. Manchmal kann Lilypond auch eine fehlerhafte Stelle zwar noch problemlos verarbeiten, ein paar Zeilen später wirkt sich der Fehler aber dann doch noch aus. In solchen Fällen, wo Sie in der angegebenen Zeile keinen Fehler erkennen, sollten Sie auch die Zeilen oberhalb der angegebenen Stelle genauer unter die Lupe nehmen.

4 lilypond-book: Noten in den Text integrieren

Wenn Sie in ein Dokument Grafiken Ihres Musiksatzes einfügen möchten, so können Sie genauso machen, wie Sie andere Grafiken einfügen würden: Die Bilder werden getrennt vom Dokument im PostScript- oder PNG-Format erstellt und können dann in LaTeX oder HTML inkludiert werden.

`lilypond-book` automatisiert diesen Prozess: Dieses Programm extrahiert Musik-Schnipsel aus Ihrem Dokument, ruft `lilypond` auf und fügt die resultierenden Bilder in Ihr Dokument ein. Die Länge der Zeilen und die Schriftgroße werden dabei Ihrem Dokument angepasst.

`lilypond-book` ist ein eigenständiges Programm und wird üblicherweise von der Kommandozeile aufgerufen. Nähere Informationen hierzu finden sich in Abschnitt 3.2 [Benutzung auf der Kommandozeile], Seite 6. Wenn Sie MacOS 10.3 oder 10.4 benutzen und Probleme mit `lilypond-book` haben, lesen Sie Einrichtung für MacOS X

Dieses Vorgehen kann bei LaTeX, HTML, Texinfo oder DocBook Dokumenten angewendet werden.

4.1 Ein musikwissenschaftlicher Text als Beispiel

Zahlreiche Texte enthalten Musikbeispiele: musikwissenschaftliche Abhandlungen, Liederbücher oder Handbücher wie dieses. Solche Texte können händisch erzeugt werden, indem einfach die Musikbeispiele als Grafik (PostScript, PNG, GIF, etc.) im Textverarbeitungsprogramm eingefügt werden. Für HTML, LaTeX, Texinfo und DocBook Dokumente existiert jedoch ein Weg, dies automatisiert durchzuführen.

Das Programm `lilypond-book` extrahiert die Musikfragmente aus dem Dokument, formatiert sie automatisiert in eine Grafik und fügt die resultierenden Notenbeispiele dann wieder in das Dokument ein. Dies soll hier an einem einfachen LaTeX-Beispiel verdeutlicht werden. Das Beispiel selbst enthält schon Erklärungen, sodass wir es hier nicht weiter diskutieren müssen.

Eingabe

```
\documentclass[a4paper]{article}

\begin{document}

Dokumente für \verb+lilypond-book+ können Musik und Text nach Belieben
kombinieren. Zum Beispiel:

\begin{lilypond}
\relative c' {
  c2 e2 \times 2/3 { f8 a b } a2 e4
}
\end{lilypond}

Optionen für \verb+lilypond+ werden dabei in eckige Klammern gesetzt.

\begin{lilypond}[fragment,quote,staffsize=26,verbatim]
  c'4 f16
\end{lilypond}

Größere Beispiele können auch in einer eigenständigen Datei gespeichert
und dann mit \verb+\lilypondfile+ eingebunden werden.
```

```
\lilypondfile[quote,noindent]{screech-boink.ly}

(Falls nötig kann screech-boink.ly durch eine beliebige andere .ly
Datei im selben Verzeichnis wie diese Datei ersetzt werden.)

\end{document}
```

Verarbeitung

Speichern Sie den obigen LaTeX Quellcode in eine Datei '`lilybook.lytex`' und führen Sie dann in der Kommandozeile folgende Befehle aus:

```
lilypond-book --output=out --pdf lilybook.lytex
lilypond-book (GNU LilyPond) 2.13.27

Reading lilybook.lytex...
..(viele Ausgabezeilen entfernt)..
Compiling lilybook.tex...
cd out
pdflatex lilybook
..(viele Ausgabezeilen entfernt)..
xpdf lilybook
(Ersetzen Sie xpdf durch Ihren PDF-Betrachter)
```

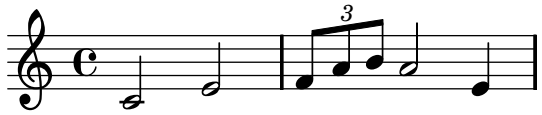
Die Ausführung von `lilypond-book` und `latex` erzeugt zahlreiche temporäre Dateien, die das Arbeitsverzeichnis unnötig vollstopfen würden. Daher empfiehlt sich die Benutzung der `--output=dir` Option, wodurch die Dateien im Unterverzeichnis '`dir`' erzeugt werden.

Das Endresultat des obigen LaTeX Beispiels ist im nächsten Abschnitt zu sehen.[1]

[1] Da dieses Handbuch mit Texinfo erzeugt wurde, kann sich das Aussehen des Beispiels leicht von dem mit LaTeX erzeugten unterscheiden.

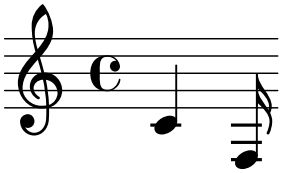
Ausgabe

Dokumente für `lilypond-book` können Musik und Text nach Belieben kombinieren. Zum Beispiel:

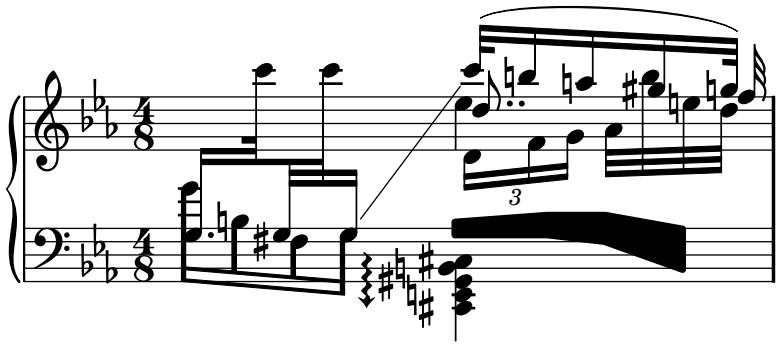


Optionen für `lilypond` werden dabei in eckige Klammern gesetzt.

```
c'4 f16
```



Größere Beispiele können auch in einer eigenständigen Datei gespeichert und dann mit `\lilypondfile` eingebunden werden.



4.2 Noten in Text integrieren

In diesem Abschnitt soll die Integration von LilyPond mit den verschiedenen Dateiformaten detailliert erläutert werden.

4.2.1 LaTeX

LaTeX ist der de-facto Standard zur Publikation von wissenschaftlichen Texten in Naturwissenschaft und Technik. Es basiert auf dem Schriftsetzer TeX, der die bestmögliche Typographie erzeugt.

Siehe die *LaTeX2e-Kurzbeschreibung* für eine Einführung in die Benutzung von LaTeX.

Musikbeispiele können eingegeben werden als

```
\begin{lilypond}[Optionen,kommen,hierhin]
  IHR LILYPOND QUELLCODE
\end{lilypond}
```

oder

```
\lilypondfile[Optionen,kommen,hier]{Dateiname}
```

oder

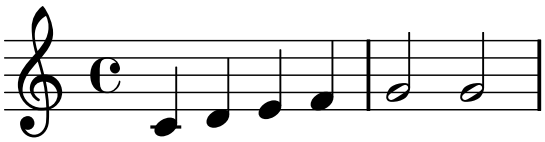
```
\lilypond{ IHR LILYPOND QUELLCODE }
```

Zusätzlich kann mit `\lilypondversion` die benutzte Versionsnummer von LilyPond angezeigt werden. Der Aufruf von `lilypond-book` liefert eine Datei, die dann mit LaTeX weiter verarbeitet werden kann.

Dies soll hier an einigen Beispielen gezeigt werden. Die `lilypond`-Umgebung

```
\begin{lilypond}[quote,fragment,staffsize=26]
  c' d' e' f' g'2 g'2
\end{lilypond}
```

erzeugt



Die Kurzversion

```
\lilypond[quote,fragment,staffsize=11]{<c' e' g'>}
```

erzeugt

Innerhalb des `\lilypond{}` Befehls dürfen keine geschwungenen Klammern `{` oder `}` vorkommen, weshalb dieser Befehl nur mit der `fragment` Option Sinn macht.

Die Standardzeilenlänge wird bestimmt, indem die Angaben in der Dokumentpräambel, also dem Teil der LaTeX Datei vor dem `\begin{document}`, analysiert werden. Der `lilypond-book` Befehl sendet diese Angaben an LaTeX, um herauszufinden, wie breit der Text tatsächlich ist. Die Breite der Notenzeilen wird dann an diese Textbreite angepasst. Ein derartig heuristischer Algorithmus kann natürlich auch versagen, wobei man in diesem Fall die Breite auch explizit durch die `line-width` Option des `\lilypond{}` oder `\begin{lilypond}` Befehls angeben kann.

Jedes Musikbeispiele ruft die folgenden Makros auf, wenn sie vom Benutzer definiert wurden:

- `\preLilyPondExample` – wird vor der Musik aufgerufen,

- `\postLilyPondExample` – wird nach der Musik aufgerufen,
- `\betweenLilyPondSystem[1]` – wird zwischen den einzelnen Systemen aufgerufen, wenn `lilypond-book` das Beispiel in verschiedene PostScript Dateien getrennt hat. Dieser LaTeX-Befehl muss so definiert werden, dass er genau ein Argument erhält, nämlich die Zahl der bereits in LaTeX eingefügten Dateien dieses Beispiels. Als Standard wird einfach ein `\linebreak` eingefügt.

Ausgewählte Schnipsel

Manchmal ist es nötig, Musikelemente wie Halte- oder Bindebögen so darzustellen, als ob sie am Ende des Musikausschnitten noch weitergehen würden. Eine solche Ausgabe kann erreicht werden, indem ein Zeilenumbruch in die Notenzeile eingefügt wird und die Ausgabe der folgenden Notenzeile unterdrückt wird.

In LaTeX wird dazu der Befehl `\betweenLilyPondSystem` einfach derartig programmiert, dass die Ausgabe der einzelnen Notensysteme abgebrochen wird, sobald die gewünschte Anzahl an Systemen erreicht ist. Da `\betweenLilyPondSystem` zum ersten Mal nach dem ersten System aufgerufen wird, ist die Ausgabe nur eines Systems trivial.

```
\def\betweenLilyPondSystem#1{\endinput}

\begin{lilypond}[fragment]
  c'1\( e'( c'~ \break c' d) e f\)
\end{lilypond}
```

Um eine größere Zahl an System nötig, dann muss dementsprechend eine TeX-Bedingung vor dem `\endinput` benutzt werden:

```
\def\betweenLilyPondSystem#1{
    \ifnum##1<2\else\endinput\fi
}
```

Dieses Beispiel bricht nach genau zwei ausgegebenen Notenzeilen ab. Für eine andere Anzahl braucht nur ‚2' durch die entsprechende Anzahl ersetzt werden.

Die Definition von `\betweenLilyPondSystem` bleibt gültig, bis TeX die aktuelle Umgebung in LaTeX verlässt oder der Befehl durch eine neue Definition überschrieben wird. Dies kann etwa folgendermaßen in der LaTeX-Datei geschehen:

```
\let\betweenLilyPondSystem\undefined
```

Obige Definition von `\betweenLilyPondSystem` kann durch die Definition eines TeX-Makros auch verallgemeinert werden,

```
\def\onlyFirstNSystems#1{
    \def\betweenLilyPondSystem##1{\ifnum##1<#1\else\endinput\fi}
}
```

wobei diesem Makro `\onlyFirstNSystems` einfach die Anzahl der gewünschten Systeme übergeben wird:

```
\onlyFirstNSystems{3}
\begin{lilypond}...\end{lilypond}
\onlyFirstNSystems{1}
\begin{lilypond}...\end{lilypond}
```

Siehe auch

`lilypond-book` stellt auch zahlreiche Kommandozeilen-Optionen zur Verfügung. Für eine Liste dieser Optionen und andere hilfreiche Details zur Verarbeitung von LaTeX-Dokumenten, siehe Abschnitt 4.4 [lilypond-book aufrufen], Seite 22.

4.2.2 Texinfo

Texinfo ist das Standard-Dokumentationsformat des GNU Projekts. Ein Beispiel für ein Dokument im Texinfo Format ist dieses Handbuch, wobei die HTML-, PDF- und Info-Versionen alle aus demselben Texinfo Dokument erzeugt werden.

In der Eingabedatei wir Musik eingegeben als

```
@lilypond[Optionen,kommen,hier]
  IHR LILYPOND QUELLCODE
@end lilypond
```

oder

```
@lilypond[Optionen,kommen,hier]{ IHR LILYPOND QUELLCODE }
```

oder

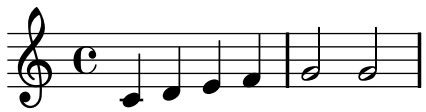
```
@lilypondfile[Optionen,kommen,hier]{Dateiname}
```

Zusätzlich kann mit `@lilypondversion` die aktuelle Versionsnummer von LilyPond angezeigt werden. Wenn `lilypond-book` eine derartige Datei verarbeitet, wird eine Texinfo-Datei mit der Erweiterung '`.texi`' erzeugt, die `@image` Befehle für die Ausgabe nach HTML, Info und PDF enthält. `lilypond-book` erzeugt die entsprechenden Grafiken der Musikbeispiele im EPS- und PDF-Format für die Ausgabe nach PDF und im PNG-Format für die Ausgabe nach HTML und Info.

Hier sollen zwei einfache Beispiele gezeigt werden. Eine `lilypond` Umgebung

```
@lilypond[fragment]
c' d' e' f' g'2 g'
@end lilypond
```

erzeugt



Die Kurzversion

```
@lilypond[fragment,staffsize=11]{<c' e' g'>}
```

erzeugt

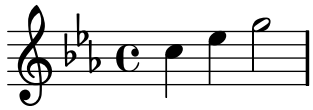
Im Gegensatz zu LaTeX erzeugt `@lilypond{...}` allerdings keine Grafik im Fließtext, sondern setzt sie immer in einen eigenen Absatz.

4.2.3 HTML

Musik wird eingegeben als

```
<lilypond fragment relative=2>
\key c \minor c4 es g2
</lilypond>
```

`lilypond-book` erzeugt dann daraus eine HTML-Datei mit den entsprechenden `<image>` Tags für die Musikbeispiele in jeweils einem eigenen Absatz.



Für Grafiken im Fließtext kann <lilypond ... /> benutzt werden, wobei die Optionen durch einen Doppelpunkt von der Musik getrennt angegeben werden.

```
Musik <lilypond relative=2: a b c/> in derselben Zeile.
```

Um Dateien mit Musik einzubinden, kann folgendermaßen vorgegangen werden:

```
<lilypondfile Option1 Option2 ...>Dateiname</lilypondfile>
```

Zusätzlich gibt <lilypondversion/> die aktuelle Versionsnummer von LilyPond aus.

4.2.4 DocBook

Bei der Einbindung von Musik im LilyPond-Format in DocBook soll die Konformität unseres DocBook Dokuments erhalten bleiben und damit die Bearbeiten mit DocBook-Editoren sowie die Validierung weiter möglich bleiben. Aus diesem Grund werden in DocBook keine eigenen Tags wie in HTML benutzt, sondern die von den vorhandenen DocBook-Elementen vorgegebenen Konventionen entsprechend benützt.

Definitionen

Für die Einbindung von LilyPond Code werden in allen Fällen die mediaobject und inlinemediaobject Elemente benutzt, die unsere Beispiele in einem eigenen Absatz oder im Fließtext einfügen. Die Optionen zur Formatierung mit LilyPond werden dabei in der role Eigenschaft des innersten Elements angegeben, wie im nächsten Abschnitt gezeigt wird. Die DocBook Datei, die dann von lilypond-book verarbeitet wird, sollte der Klarheit halber die Dateierweiterung '.lyxml' (jedenfalls nicht '.xml') besitzen.

Eine LilyPond-Datei einfügen

Dies ist der einfachste Fall: Die LilyPond-Datei besitzt die Erweiterung '.ly' und wird einfach als imageobject eingebettet:

```
<mediaobject>
  <imageobject>
    <imagedata fileref="music1.ly" role="printfilename" />
  </imageobject>
</mediaobject>
```

Für das äußerste Element kann je nach Bedarf mediaobject oder inlinemediaobject benutzt werden.

LilyPond-Code einfügen

Die Einbindung von LilyPond-Code direkt in der DocBook-Datei ist durch die Benutzung von programlisting möglich, wobei die Sprache auf lilypond gesetzt wird:

```
<inlinemediaobject>
  <textobject>
    <programlisting language="lilypond" role="fragment verbatim staffsize=16 ragged-rig
\context Staff \with {
  \remove Time_signature_engraver
  \remove Clef_engraver}
  { c4( fis) }
    </programlisting>
  </textobject>
</inlinemediaobject>
```

Das äußerste Element ist also mediaobject oder inlinemediaobject, welches ein textobject mit dem programlisting enthält.

Ein DocBook-Dokument übersetzen

Der Aufruf von `lilypond-book` mit der '`.lyxml`'-Datei erzeugt ein gültiges DocBook-Dokument mit der Erweiterung '`.xml`', welches normal weiterverarbeitet werden kann. Bei Benutzung von dblatex wird daraus automatisch eine PDF-Datei erzeugt. Für die Erzeugung von HTML (HTML Hilfe, JavaHelp, etc.) können die offiziellen DocBook XSL-Stylesheets benutzt werden. Eventuell müssen dafür allerdings kleinere Anpassungen vorgenommen werden.

4.3 Die Musikfragment-Optionen

Im Folgenden meint ‚LilyPond-Kommando' ein beliebiges in den vorgehenden Abschnitten beschriebenes Kommando, welches ein LilyPond-Fragment in eine Datei einfügt und von `lilypond-book` verarbeitet wird. Der Einfachheit halber werden hier alle LilyPond-Kommandos in der Syntax von LaTeX dargestellt.

Zu beachten ist, dass die Optionen eines LilyPond-Kommandos von links nach rechts verarbeitet werden. Wenn eine Option also mehrfach angegeben wird, wird nur die letzte benutzt.

Die folgenden Optionen können für LilyPond-Kommandos benutzt werden:

`staffsize=`*ht*
: Setzt die Höhe einer Notenzeile auf *ht*, angegeben in Punkten.

`ragged-right`
: Erzeugt Notenzeilen im Flattersatz mit natürlichem Abstand der Noten. In anderen Worten: `ragged-right = ##t` wird in das Musikfragment eingefügt. Dies ist die Standardeinstellung für das `\lilypond{}` Kommando, wenn die Option `line-width` nicht angegeben wird. Ebenso ist dies die Standardeinstellung für die `lilypond`-Umgebung, wenn die Option `fragment`, aber keine Zeilenlänge explizit angegeben ist.

`noragged-right`
: Streckt Musikfragmente mit nur einer Notenzeile auf die volle Breite, es wird also `ragged-right = ##f` in das Musikfragment eingefügt.

`line-width`
`line-width=`*Breite*`\`*Einheit*
: Setzt die Breite der Notenzeilen auf *Breite*, gemessen in Vielfachen der *Einheit*. Als Einheit können die folgenden Zeichenfolgen angegeben werden: `cm`, `mm`, `in` oder `pt`. Diese Option hat nur Einfluss auf die Breite von Notenzeilen und Text im Musikfragment, nicht jedoch auf den restlichen Text des Dokuments.

  Wird diese Option ohne einen Wert angegeben, wird die Zeilenbreite auf einen Standardwert gesetzt, der durch einen heuristischen Algorithmus bestimmt wird.

  Wenn die `line-width` Option nicht angegeben wird, versucht `lilypond-book` einen geeigneten Standardwert für alle `lilypond`-Umgebungen zu finden, die die `ragged-right` Option nicht benutzen.

`notime`
: Verhindert die Ausgabe der Taktangabe am Anfang des Fragments und schaltet Taktstriche und alle Taktangaben im Fragment ab.

`fragment`
: Bewirkt, dass `lilypond-book` Standardcode um das Fragment herum einfügt, sodass z. B.

  ```
  c'4
  ```

  ohne `\layout`, `\score`, etc. eingegeben werden kann.

`nofragment`
: Verhindert das Einfügen von Standardcode um das Fragment herum. Dies ist die Standardeinstellung, insofern ist diese Option im Normalfall unnötig.

`indent=`*`Einzug`*`\`*`Einheit`*
: Setzt den Einzug des ersten Notensystems auf *Einzug*, gemessen in Vielfachen der *Einheit*. Als Einheit können die folgenden Zeichenfolgen angegeben werden: `cm`, `mm`, `in` oder `pt`. Diese Option hat nur Einfluss auf den Einzug von Notenzeilen und Text im Musikfragment, nicht jedoch auf den restlichen Text des Dokuments.

`noindent`
: Setzt den Einzug des ersten Notensystems auf 0. Diese Option hat nur Einfluss auf den Einzug von Notenzeilen und Text im Musikfragment, nicht jedoch auf den restlichen Text des Dokuments. Dies ist die Standardeinstellung, insofern ist diese Option im Normalfall unnötig.

`quote`
: Verringert die Zeilenlänge des Musikfragments um $2 * 0.4$ in und setzt das Fragment in einen Zitat-Block. Der Wert von ‚0.4 in‘ kann durch die `exampleindent` Option angepasst werden.

`exampleindent`
: Setzt den Betrag, um den das Fragment bei Benutzung der `quote` Option eingerückt wird.

`relative`
`relative=`*`n`*
: Benutzt relative Oktavenbezeichnungen. Standardmäßig werden Noten relativ zum mittleren C angegeben. Das optionale ganzzahlige Argument gibt die Oktave der ersten Note an, wobei die Standardeinstellung von `1` das mittlere C bedeutet. Die `relative` Option macht nur Sinn in Verbindung mit der `fragment` Option, weshalb `fragment` automatisch durch die Angabe der `relative` Option impliziert wird. Eine explizite Angabe der `(no)fragment` Option hat keinen Effekt.

LilyPond benutzt zur Erzeugung seiner eigenen Dokumentation ebenfalls `lilypond-book`. Zu diesem Zweck stehen noch zahlreiche spezialisierte Optionen zur Verfügung:

`verbatim`
: Der LilyPond-Code im LilyPond-Kommando wird zum einen benutzt, um das Musikfragment in eine Grafik mit schönem Notensatz zu konvertieren, andererseits aber auch wörtlich in das Dokument eingefügt. Dies geschieht in einem ‚verbatim‘-Block, gefolgt vom Text einer möglicherweise angegebenen `intertext` Option[1] und der Grafik des tatsächlichen Notensatzes. Diese Option funktioniert nur fehlerhaft, wenn `\lilypond{}` im Fließtext benutzt wird.

  Wird `verbatim` in Verbindung mit einem `lilypondfile`-Kommando benutzt, so ist es auch möglich, nur ein Teil der Datei wörtlich einfügen zu lassen: Wenn die eingebundene LilyPond-Datei ein Kommentar mit dem Inhalt ‘`begin verbatim`’ (ohne Anführungszeichen) enthält, wird nur der Dateiinhalt ab dieser Position eingefügt. Enthält die Datei mehrere solche Kommentare, wirkt nur das letzte. Analog wird nur der Dateiinhalt bis zu einem etwaigen Kommentar mit dem Inhalt ‘`end verbatim`’ eingefügt. Im folgenden Beispiel wird das gesamte Musik für die Erzeugung der Grafik im relativen Oktavenmodus interpretiert, der wörtlich in das Dokument kopierte LilyPond-Code zeigt den `relative`-Befehl jedoch nicht.

  ```
  \relative c' { % begin verbatim
    c4 e2 g4
    f2 e % end verbatim
  }
  ```

  erzeugt ein Zitat der Form

  ```
    c4 e2 g4
    f2 e
  ```

[1] Die `intertext` Option ist noch nicht implementiert.

Wenn Kommentare und Variablen im Zitat, aber nicht im Quelltext übersetzt werden sollen, kann die Umgebungsvariable LYDOC_LOCALEDIR auf einen Verzeichnispfad gesetzt werden. Das Verzeichnis sollte einen Baum an '.mo'-Nachrichtenkatalogen beinhalten mit lilypond-doc als Domain.

addversion

(Nur innerhalb von Texinfo-Dateien.) Stellt \version @w{"@version{}"} an den Beginn des Fragments der Ausgabe mit verbatim.

texidoc (Nur innerhalb von Texinfo-Dateien.) Wird lilypond mit der Kommandozeilenoption '--header=texidoc' für eine Datei 'foo.ly' und enthält die Datei ein texidoc-Feld im \header-Block, so wird dessen Inhalt in die Datei 'foo.texidoc' ausgegeben. Die texidoc Option veranlasst lilypond-book, den Inhalt dieser '.texidoc' Dateien innerhalb eines Dokumentationsblocks direkt vor dem Musikfragment in das aktuelle Dokument einzufügen.

Enthält also die Datei 'foo.ly' etwa den LilyPond-Code

```
\header {
  texidoc = "Dieses Beispiel zeigt eine einzelne Note."
}
{ c'4 }
```

und das Texinfo-Dokument 'text.texinfo'

```
@lilypondfile[texidoc]{foo.ly}
```

so liefert der folgende Aufruf von lilypond-book das gewünschte Ergebnis:

```
lilypond-book --pdf --process="lilypond \
  -dbackend=eps --header=texidoc" test.texinfo
```

Die meisten Test-Dateien (im 'input/' Verzeichnis von LilyPond) sind kleine '.ly' Dateien von genau dieser Form.

Auch die Übersetzung dieser zusätzlichen Kommentare ist möglich: Dazu muss das Texinfo-Dokument den Befehl @documentlanguage *LANG* und der \header Block in der Datei 'foo.ly' die Übersetzung im Feld texidoc*LANG* enthalten. Wird nun lilypond mit der Option '--header=texidoc*LANG*' aufgerufen, so wird der Inhalt der Datei 'foo.texidoc*LANG*' anstelle von 'foo.texidoc' eingefügt.

lilyquote

(Nur innerhalb von Texinfo-Dateien.) Diese Option wirkt wie die quote Option, außer dass nur das Musikfragment (und natürlich optional der gesamte LilyPond-Code bei Benutzung von verbatim) in den Zitatblock eingefügt wird. Dies ist nützlich, wenn Sie ein Musikfragment zitieren möchten, nicht aber dessen texidoc Dokumentationsblock.

doctitle (Nur innerhalb von Texinfo-Dateien.) Diese Option wirkt ähnlich wie die texidoc Option: Wenn lilypond mit der Option '--header=doctitle' aufgerufen wird und die Eingabedatei 'foo.ly' ein Feld doctitle im \header-Block enthält, wird dessen Wert in die Datei 'foo.doctitle' geschrieben. Wird die doctitle Option für ein Musikfragment benutzt, so wird der Inhalt dieser Datei, der eine einzelne Textzeile sein sollte, im Texinfo-Dokument als @lydoctitle *Text* eingefügt. @lydoctitle muss allerdings in Ihrem Texinfo-Dokument als Makro selbst definiert werden. Die Übersetzung funktioniert völlig analog zu texidoc.

nogettext

Nur für Texinfo-Ausgabe: Kommentare und Variablenbezeichnungen im zitierten Quelltext des Schnipsel werden nicht übersetzt.

`printfilename`
: Wenn eine LilyPond-Datei mittels `\lilypondfile` und dieser Option eingebunden wird, wird der Dateiname (ohne die Pfadangabe) unmittelbar vor dem Musikfragment ausgegeben. In HTML-Dateien ist er außerdem ein Link auf die LilyPond-Datei. Nur der eigentliche Name der Datei wird ausgegeben, der Pfad wird also nicht mit angezeigt.

4.4 `lilypond-book` aufrufen

`lilypond-book` erzeugt abhängig vom Ausgabeformat eine Datei mit einer der folgenden Dateierweiterungen: '`.tex`', '`.texi`', '`.html`' oder '`.xml`'. Alle '`.tex`', '`.texi`' und '`.xml`' Dateien müssen noch mit den entsprechenden Programmen (LaTeX, DocBook, etc.) weiter verarbeitet werden, um druckfähige Dateien zu erhalten.

Formatabhängige Anweisungen

LaTeX

Es existieren zwei Methoden, Ihr LaTeX-Dokument weiter zu verarbeiten, um zu einer druck- oder publikationsfähigen Datei zu gelangen: Zum einen die direkte Erzeugung einer PDF-Datei mit PDFLaTeX, zum anderen die Erzeugung einer DVI daraus einer PostScript-Datei mittels LaTeX und einem DVI-nach-PostScript Konverters wie `dvips`. Die erste Methode ist einfacher und daher empfehlenswert.[2] Welche Methode auch immer benutzt wird, die Konvertierung zwischen PostScript und PDF kann leicht mit Hilfsprogrammen wie `ps2pdf` und `pdf2ps` (aus dem Ghostscript Paket) erfolgen.

Um eine PDF-Datei mittels PDFLaTeX zu erzeugen, kann folgendermaßen vorgegangen werden:

```
lilypond-book --pdf Ihre_Datei.pdftex
pdflatex Ihre_Datei.tex
```

Um eine PDF-Datei mittels LaTeX/`dvips`/`ps2pdf` zu erhalten, sind folgende Befehle nötig:

```
lilypond-book Ihre_Datei.lytex
latex Ihre_Datei.tex
dvips -Ppdf Ihre_Datei.dvi
ps2pdf Ihre_Datei.ps
```

Die '`.dvi`'-Datei, die beim Aufruf von `latex` erzeugt wird, scheint keine Notenköpfe zu enthalten, was in Ordnung ist. Wenn Sie die Datei wie beschrieben weiter verarbeiten, erscheinen die Notenköpfe korrekt in den '`.ps`' und '`.pdf`' Dateien.

Der Aufruf von `dvips` kann einige Warnungen über fehlende Schriftarten ausgeben. Auch dies ist in Ordnung und kann ignoriert werden.

Wenn Sie in der LaTeX-Datei das Papierformat auf Querformat eingestellt haben, vergessen Sie nicht auf die `-t landscape` Option beim Aufruf von `dvips`.

Texinfo

Um ein von `lilypond-book` erzeugtes Texinfo-Dokument zu verarbeiten, gehen Sie wie für alle anderen Texinfo-Dokumente vor: Rufen Sie – abhängig vom gewünschten Ausgabeformat – eines der Programme `texi2pdf`, `texi2dvi`, `makeinfo` oder `texi2html` auf.

Die Dokumentation von Texinfo liefert dazu nähere Informationen.

[2] Manchmal kann eine Datei entweder von PDFLaTeX oder von LaTeX nicht korrekt verarbeitet werden, weshalb hier beide Methoden beschrieben werden.

Optionen auf der Kommandozeile

lilypond-book unterstützt die folgenden Kommandozeilenoptionen:

-f *Format*
--format=*Format*
: Gibt das Format des Eingabedokuments an: html, latex, texi (Standardeinstellung), texi-html oder docbook. Ist diese Option nicht angegeben, versucht lilypond-book das Format anhand des Dateinamens zu bestimmen. Im Moment bedeutet texi praktisch dasselbe wie texi-html.

-F *Filter*
--filter=*Filter*
: Leitet die Musikfragmente durch das Programm *filter* anstatt sie mit Lilypond zu einer Grafik zu verarbeiten. '--filter' und '--process' kann nicht gleichzeigt benutzt werden. Beispielaufruf:

```
lilypond-book --filter='convert-ly --from=2.0.0 -' Mein-Buch.tely
```

-h
--help
: Gibt eine kurze Hilfemeldung aus.

-I *Pfad*
--include=*Pfad*
: Fügt *Pfad* zu den Include-Pfaden hinzu. lilypond-book sucht auch in allen Include-Pfaden nach bereits erstellten Grafiken für die Musikfragmente. Wird das aktuelle Fragment gefunden und hat sich seit der letzten Erstellung nicht geändert, wird es nicht erneut erzeugt. Bei manchen der Programme zur Weiterverarbeitung wie etwa makeinfo oder latex muss dieselbe -I *Pfad* Option angegeben werden, damit das entsprechende Programm die Grafiken ebenso findet.

-o *Verzeichnis*
--output=*Verzeichnis*
: Erzeugt die Ausgabedateien in *Verzeichnis*. Der Aufruf von lilypond-book erzeugt zahlreiche kleine Dateien, die von LilyPond, latex, makeinfo etc. dann weiter benützt werden. Um zu vermeiden, dass das Quellenverzeichnis durch diese Dateien unübersichtlich wird, kann die '--output' Option benutzt werden. Vor dem Aufruf von latex oder makeinfo sollten Sie in dieses Verzeichnis wechseln.

```
lilypond-book --output=out IhreDatei.lytex
cd out
...
```

--skip-lily-check
: Nicht mit einer Fehlermeldung abbrechen, wenn keine Ausgabe von LilyPond gefunden wird. Dies wird benutzt für Dokumentation ohne Grafiken.

--skip-png-check
: Nicht mit einer Fehlermeldung abbrechen, wenn für die EPS-Dateien keine PNG-Grafiken gefunden werden. Dies wird benutzt für Dokumentation ohne Grafiken.

--lily-output-dir=*Verzeichnis*
: Schreibt 'lily-XXX' Dateien nach *Verzeichnis* und erzeugt im mit --output angegebenen Verzeichnis Verknüpfungen darauf. Diese Option ist nützlich, um

Zeit zu sparen, wenn Dokumente in verschiedenen Verzeichnissen viele identische Musikfragmente enthalten.

`--info-images-dir=`*Verzeichnis*
: Formatiert die Texinfo-Ausgabe dergestalt, dass Info in *Verzeichnis* nach den Grafiken zu den Musikfragmenten sucht.

`--latex-program=`*Programm*
: Führt `Programm` anstelle von `latex` aus. Dies ist nützlichen, wenn das Dokument mit einer anderen LaTeX-Variante wie etwa `xelatex` verarbeitet werden soll.

`--left-padding=`*Einrückung*
: Fügt *Einrückung* als zusätzlichen Einzug in die EPS-Box ein. *Einrückung* wird in Millimetern angegeben, die Standardeinstellung is 3.0 Millimeter. Diese Option kann benutzt werden, wenn die Notenzeilen über den Rand des Dokuments hinausstehen.

  Die Breite eines eng ausgeschnittenen Notensystems kann variieren aufgrund von Notationselementen, die über den linken Rand hinausstehen, wie etwa Taktzahlen und Bezeichnungen der Instrumente. Diese Option verkürzt die Notenzeile und verschiebt sie um denselben Betrag nach rechts.

`-P` *Kommando*
`--process=`*Kommando*
: Verarbeitet LilyPond-Fragmente mit *Kommando* anstelle des Standardbefehls `lilypond`. `--filter` und `--process` können nicht gleichzeitig angegeben werden.

`--pdf` Erzeugt PDF-Dateien mit PDFLaTeX.

`-V`
`--verbose`
: Gibt ausführliche informative Meldungen aus.

`-v`
`--version`
: Gibt die Versionsnummer aus.

Bekannte Probleme und Warnungen

Der Texinfo-Befehl `@pagesizes` wird ignoriert. Ebenso werden LaTeX-Befehle ignoriert, die den Seitenrand oder die Zeilenlänge nach der Dokumentpräambel verändern.

Nur der erste `\score`-Block eines LilyPond-Fragments wird verarbeitet.

4.5 Dateiendungen

Für die Eingabedatei kann zwar jede beliebige Dateinamenserweiterung benutzt werden, allerdings muss bei Verwendung einer nicht bekannten Erweiterung das Ausgabeformat explizit an `lilypond-book` angegeben werden. Details dazu finden sich im Abschnitt Abschnitt 4.4 [lilypond-book aufrufen], Seite 22. Wird eine bekannte Erweiterung benutzt, wählt `lilypond-book` automatisch das richtige Ausgabeformat basierend auf der Erweiterung der Eingabedatei:

Erweiterung	**Ausgabeformat**
'`.html`'	HTML
'`.itely`'	Texinfo
'`.latex`'	LaTeX
'`.lytex`'	LaTeX
'`.lyxml`'	DocBook
'`.tely`'	Texinfo
'`.tex`'	LaTeX
'`.texi`'	Texinfo

'`.texinfo`'	Texinfo
'`.xml`'	HTML

Wird dieselbe Erweiterung für die Eingabedatei wie für die Ausgabedatei benutzt und befindet sich die Eingabedatei im aktuellen Arbeitsverzeichnis von `lilypond-book`, muss die `--output` Option für `lilypond-book` benutzt werden. Anderenfalls würde `lilypond-book` ja die Eingabedatei überschreiben, weshalb das Programm in diesem Fall mit einer Fehlermeldung wie „Fehler: Ausgabe würde Eingabedatei überschreiben; verwenden Sie –output.“ abbricht.

4.6 Alternative Methoden, Text mit Musik zu kombinieren

Dieser Abschnitt stellt Methoden vor, wie Text und Musik auf andere Weise kombiniert werden können als dies durch `lilypond-book` automatisiert geschieht.

4.6.1 Viele Zitate aus einer langen Partitur

Wenn aus einer großen Partitur viele kleine Fragmente eingefügt werden sollen, kann dazu das ‚clip systems‘ Feature benutzt werden. Siehe Abschnitt “Notationsfragmente extrahieren” in *Notationsreferenz*.

4.6.2 LilyPond-Noten in OpenOffice.org integrieren

Musik im LilyPond-Format kann in OpenOffice.org eingefügt werden mittels OOoLilyPond.

4.6.3 LilyPond-Noten in andere Programme integrieren

Im die Ausgabe von LilyPond in anderen Programmen einzufügen, sollte `lilypond` benutzt werden. Jedes Beispiel muss getrennt manuell erzeugt und ins Dokument eingefügt werden; für letzteres schlagen Sie bitte im Handbuch Ihrer Textverarbeitungs-Software nach. Die meisten Programme unterstützen das Einfügen von Grafiken im '`PNG`'-, '`EPS`'- oder '`PDF`'-Format.

Um den leeren Rand um die Notenzeilen zu verringern, können folgende Einstellungen benutzt werden:

```
\paper{
  indent=0\mm
  line-width=120\mm
  oddFooterMarkup=##f
  oddHeaderMarkup=##f
  bookTitleMarkup = ##f
  scoreTitleMarkup = ##f
}

{ c1 }
```

Eine '`EPS`'-Datei kann mit folgendem Befehl erzeugt werden:

```
lilypond -dbackend=eps -dno-gs-load-fonts -dinclude-eps-fonts   Dateiname.ly
```

eine '`PNG`'-Datei mittels:

```
lilypond -dbackend=eps -dno-gs-load-fonts -dinclude-eps-fonts --png Dateiname.ly
```

5 Von anderen Formaten konvertieren

LilyPond kann auch Musik aus diversen anderen Formaten importieren. Dieses Kapitel beschreibt die dazu mit LilyPond mitgelieferten Hilfsprogramme. Daneben existieren natürlich auch noch weitere Programme, die Dateien für LilyPond erstellen können, wie etwa graphische Sequenzierprogramme und XML-Konverter. Näheres dazu findet sich auf der Homepage von LilyPond.

Die im Folgenden beschriebenen Programme sind eigenständige Hilfsprogramme und werden üblicherweise von der Kommandozeile aufgerufen. Siehe Abschnitt 3.2 [Benutzung auf der Kommandozeile], Seite 6 für weitere Informationen. Wenn Sie MacOS 10.3 oder 10.4 benutzen und Probleme mit diesen Skripten (z. B. `convert-ly`) haben, lesen Sie Einrichtung für MacOS X.

Bekannte Probleme und Warnungen

Leider haben wir nicht ausreichend viele Entwickler, um all die folgenden Hilfsprogramme ständig zu warten. Wir stellen den aktuellen Stand zur Verfügung, können aber leider Fehlerberichte nur selten bearbeiten. Selbstverständlich sind Patches von Ihnen sehr willkommen!

5.1 `midi2ly` aufrufen

`midi2ly` übersetzt eine Typ 1 MIDI-Datei in eine Eingabedatei für LilyPond.

MIDI (Music Instrument Digital Interface) ist ein internationaler Standard für digitale Instrumente: Es spezifiziert die Verkabelung, ein serielles Protokoll und ein Dateiformat. Das MIDI-Dateiformat ist der de-facto Standard um Musik von vielen Programmen zu exportieren. Allerdings fehlen in den MIDI-Dateien viele Ausdrucks- und Artikulationszeichen. Dennoch kann MIDI vielfach nützlich sein, um Musik von einem Programm zu importieren, für das kein spezielles Hilfsprogramm den direkten Import nach LilyPond unterstützt.

`midi2ly` konvertiert die MIDI-Spuren nach Abschnitt “Staff” in *Referenz der Interna* und MIDI-Kanäle in Abschnitt “Voice” in *Referenz der Interna* Kontexte. Tonhöhen werden relativ angegeben, Tondauern nur wenn nötig.

MIDI-Dateien können auch direkt von einem digitalen Keyboard aufgenommen und dann in eine ‘`.ly`’-Datei konvertiert werden. Allerdings sind Musikinterpretationen von Menschen (aus gutem Grund!) rhythmisch nicht exakt genug um die Konvertierung von MIDI nach LY trivial zu gestalten. Wenn `midi2ly` mit Quantisierung (`-s` und `-d` Kommandozeilenoptionen) aufgerufen wird, versucht es diese Unschärfen im Zeitablauf zu korrigieren, ist allerdings nicht sonderlich gut darin. Daher können wir diese Technik leider nicht für die Konvertierung von MIDI-Aufnahmen empfehlen.

`midi2ly` wird von der Kommandozeile folgendermaßen aufgerufen:

```
midi2ly [Optionen]... MIDI-Datei
```

Unter ‚Kommandozeile‘ verstehen wir dabei die Kommandozeile des jeweiligen Betriebssystems. Für nähere Informationen hierzu siehe Kapitel 5 [Von anderen Formaten konvertieren], Seite 26.

Die folgenden Kommandozeilenoptionen werden von `midi2ly` unterstützt:

`-a, --absolute-pitches`
: Gibt absolute Tonhöhen aus.

`-d, --duration-quant=`*LÄNGE*
: Quantisiert Tondauern zu Vielfachen von *LÄNGE*.

`-e, --explicit-durations`
: Gibt alle Tondauern explizit an.

`-h,--help`
: Zeigt eine Zusammenfassung der Programmbenutzung und der Optionen.

`-k, --key=`*acc*`[:`*Moll*`]`
: Setzt die Standard-Tonart. *acc* > 0 gibt die Anzahl der Kreuze an, *acc* < 0 gibt die Anzahl der Bs der Tonart an. Eine Moll-Tonart wird durch `:1` angegeben.

`-o, --output=`*Datei*
: Die Ausgabe wird in die Datei *Datei*‘`.ly`’ geschrieben.

`-s, --start-quant=`*LÄNGE*
: Quantisiert den Beginn der Noten zu Vielfachen von *LÄNGE*.

`-t, --allow-tuplet=`*DUR*`*`*NUM*`/`*DEN*
: Erlaubt Tuplet-Dauern der Form *DUR***NUM*/*DEN*.

`-V, --verbose`
: Gibt ausführliche informative Meldungen während der Konvertierung aus.

`-v, --version`
: Gibt die Versionsnummer aus.

`-w, --warranty`
: Zeigt die Lizenzbedingungen und Urheberrechtshinweise.

`-x, --text-lyrics`
: Interpretiert alle Texte als Liedtexte.

Bekannte Probleme und Warnungen

Überlappende Noten in einem Arpeggio werden nicht korrekt dargestellt. Nur die erste Note wird eingelesen und konvertiert, die restlichen werden ignoriert. Als Abhilfe können Sie alle Noten auf dieselbe Tonlänge setzen und Phrasierungszeichen oder Pedalindikatoren hinzufügen.

5.2 `muscxml2ly` aufrufen

MusicXML ist ein XML-Dialekt zur Darstellung von Musiknotation.

`musicxml2ly` wandelt eine MusicXML-Datei nach LilyPond um, wobei sowohl die Noten, Artikulationszeichen, Struktur der Partitur, Liedtexte etc. einer MusicXML-Datei (im ‚partwise‘-Format) in eine .ly-Datei um.

`musicxml2ly` wird von der Kommandozeile folgendermaßen aufgerufen:

```
musicxml2ly [Optionen]... XML-Datei
```

Unter ‚Kommandozeile‘ verstehen wir dabei die Kommandozeile des jeweiligen Betriebssystems. Für nähere Informationen hierzu siehe Kapitel 5 [Von anderen Formaten konvertieren], Seite 26.

Wenn als Dateiname ‘`-`’ angegeben wird, liest `musicxml2ly` Daten direkt von der Kommandozeile ein.

Die folgenden Kommandozeilenoptionen werden von `musicxml2ly` unterstützt:

`-a, --absolute`
: Konvertiert in absolute Tonhöhen.

`-h,--help`
: Zeigt eine Zusammenfassung der Programmbenutzung und der Optionen.

`-l, --language=LANG`
: Inkludiert die Sprachdatei *LANG*‘`.ly`’ in der Ausgabe. Wird z.B. ’deutsch’ angegeben, wird ‘`deutsch.ly`’ in die .ly-Datei eingebunden und deutsche Notenbezeichnungen benutzt.

`--lxml` Benutzt das lxml.etree Python-Paket für die Verarbeitung von XML (benötigt weniger Speicher und Rechenleistung)

`--nd --no-articulation-directions`
Konvertiert keine Richtungsangaben (`^`, `_` oder `-`) von Artikulations- und Lautstärkebezeichnungen.

`--no-beaming`
Konvertiert keine Informationen über die Balkensetzung aus der MusicXML-Datei. Stattdessen wird dies LilyPond überlassen.

`-o,--output=`*Dateiname*
Die Ausgabe wird in die Datei *Dateiname*‘`.ly`’ geschrieben. Wird als *Dateiname* nur ‘`-`’ angegeben, wird das Ergebnis der Konvertierung an der Kommandozeile ausgegeben. Wird diese Option nicht angegeben, so erfolgt die Ausgabe in die Datei *XML-Datei*‘`.ly`’.

`-r,--relative`
Konvertiert in relative Tonhöhen. (Standardeinstellung)

`-v, --verbose`
Gibt ausführliche informative Meldungen während der Konvertierung aus.

`--version`
Gibt die Versionsnummer aus.

`-z,--compressed`
Die Eingabedatei wird als komprimierte MusicXML-Datei eingelesen. Dies ist die Standardeinstellung für Dateien mit der Erweiterung ‘`.mxl`’.

5.3 `abc2ly` aufrufen

ABC ist ein relativ einfaches ASCII-basierendes Musikformat und ist dokumentiert auf der ABC-Homepage:

http://www.walshaw.plus.com/abc/learn.html.

`abc2ly` konvertiert ABC-Dateien nach LilyPond und wird von der Kommandozeile folgendermaßen aufgerufen:

`abc2ly [`*Optionen*`]...` *ABC-Datei*

Unter ‚Kommandozeile‘ verstehen wir dabei die Kommandozeile des jeweiligen Betriebssystems. Für nähere Informationen hierzu siehe Kapitel 5 [Von anderen Formaten konvertieren], Seite 26.

Die folgenden Kommandozeilenoptionen werden von `abc2ly` unterstützt:

`-b,--beams=None`
Die Balkensetzung aus der ABC-Datei erhalten.

`-h,--help`
Zeigt eine Zusammenfassung der Programmbenutzung und der Optionen.

`-o,--output=`*Dateiname*
Die Ausgabe wird in die Datei *Dateiname*‘`.ly`’ geschrieben.

`-s,--strict`
Strenge Auslegung einer erfolgreichen Konvertierung.

`-v,--version`
Gibt die Versionsnummer aus.

Es existiert außerdem die Möglichkeit, LilyPond-Befehle für die Konvertierung mit `abc2ly` gleich in der ABC-Datei anzugeben. Wenn sich etwa in der ABC-Datei das Kommentar

```
%%LY voices \set autoBeaming = ##f
```

befindet, so wird der Text nach dem Schlüsselwort ‚voices' direkt in die aktuelle Stimme in der LilyPond-Datei eingefügt.

Ebenso bewirkt

```
%%LY slyrics more words
```

dass alles nach dem ‚slyrics' Schlüsselwort an der aktuellen Stelle im Liedtext eingefügt wird.

Bekannte Probleme und Warnungen

Der ABC-Standard ist eigentlich kein wirklich vollständiger Standard. Für komplexere Notation wie etwa Polyphonie existieren verschiedene Konventionen.

Mehrere Lieder in einer Datei können nicht konvertiert werden.

ABC synchronisiert den Liedtext am Anfang jeder Zeile mit den Noten, `abc2ly` macht dies nicht.

`abc2ly` ignoriert die Balkensetzung in der ABC-Datei.

5.4 `etf2ly` aufrufen

ETF (Enigma Transport Format) ist ein Dateiformat, das Coda Music Technology in älteren Versionen des Programms Finale benutzt hat.

`etf2ly` konvertiert Teile einer ETF-Datei nach LilyPond und wird von der Kommandozeile folgendermaßen aufgerufen:

```
etf2ly [Optionen]... ETF-Datei
```

Unter ‚Kommandozeile' verstehen wir dabei die Kommandozeile des jeweiligen Betriebssystems. Für nähere Informationen hierzu siehe Kapitel 5 [Von anderen Formaten konvertieren], Seite 26.

Die folgenden Kommandozeilenoptionen werden von `etf2ly` unterstützt:

`-h,--help`
: Zeigt eine Zusammenfassung der Programmbenutzung und der Optionen.

`-o,--output=`*Dateiname*
: Die Ausgabe wird in die Datei *Dateiname*`.ly` geschrieben.

`--version`
: Gibt die Versionsnummer aus.

Bekannte Probleme und Warnungen

Die Liste der Artikulationszeichen is unvollständig. Leere Takte verwirren `etf2ly`. Mehrfache Vorschlagnoten werden falsch beendet.

5.5 LilyPond-Dateien erstellen

LilyPond selbst wird nur mit den oben genannten Konvertierprogrammen ausgeliefert und unterstützt keine weiteren Formate. Es existieren allerdings externe Hilfsprogramme, die auch LilyPond-Dateien erzeugen können. Diese umfassen unter anderem:

- Denemo, ein grafisches Programm zum Musiksatz.
- Rumor, ein MIDI-nach-LilyPond Konverter, der monophone MIDI-Eigaben in Echtzeit konvertiert.
- lyqi, ein Emacs-Modus.

- xml2ly, ein Konverter für MusicXML, der auf XML-Transformationen mittels XSLT basiert.
- NoteEdit, ein grafisches Programm zum Notensatz, das auch MusicXML importieren kann.
- Rosegarden, das MIDI importieren kann.
- FOMUS, eine LISP-Bibliothek, um Musiknotation zu erzeugen.
- `http://vsr.informatik.tu-chemnitz.de/staff/jan/nted/nted.xhtml`, ein Notensatzprogramm mit experimentellem Export nach LilyPond.
- TuxGuitar, ein Programm zur Erzeugung von Gitarren-Tabulaturen, unterstützt den Export nach LilyPond.
- MuseScore unterstützt ebenfalls den Export nach LilyPond.

6 An LilyPond-Projekten arbeiten

Dieses Kapitel erklärt, wie bestimmte häufige Probleme zu lösen oder ganz zu vermeiden sind. Wenn Sie schon Programmiererfahrung mitbringen, erscheinen diese Hinweise vielleicht überflüssig, aber es wird dennoch empfohlen, dieses Kapitel zu lesen.

6.1 Vorschläge, wie LilyPond-Eingabe-Dateien geschrieben werden sollen

Jetzt sind Sie so weit, größere Stücke mit LilyPond zu schreiben – nicht nur die kleinen Beispiele aus der Übung, sondern ganze Stücke. Aber wie geht man das am besten an?

Solange LilyPond Ihre Dateien versteht und die Noten so setzt, wie Sie das wollen, spielt es eigentlich keine Rolle, wie Ihre Dateien aussehen. Es gibt aber trotzdem ein paar Dinge, die man beim Schreiben von LilyPond-Code berücksichtigen sollte.

- Was ist, wenn Sie einen Fehler machen? Die Struktur einer LilyPond-Datei kann es erleichtern (oder erschweren), bestimmte Fehler zu finden.
- Was ist, wenn Sie Ihre Dateien mit jemandem austauschen wollen? Oder Ihre Dateien nach einige Jahren noch einmal überarbeiten wollen? Manche LilyPond-Dateien versteht man auf den ersten Blick, über anderen muss man eine Stunde grübeln, um die Struktur zu ahnen.
- Was ist, wenn sie Ihre Dateien auf eine neuere LilyPond-Version aktualisieren wollen? Die Syntax der Eingabesprache verändert sich allmählich mit Verbesserungen im Programm. Die meisten Veränderungen können automatisch durch `convert-ly` gelöst werden, aber bestimmte Änderungen brauchen Handarbeit. LilyPond-Dateien können strukturiert werden, damit sie einfacher aktualisierbar sind.

6.1.1 Allgemeine Vorschläge

Hier einige Vorschläge, wie Sie Probleme vermeiden oder lösen können:

- **Schreiben Sie immer mit `\version` die Versionsnummer in jede Datei**. Beachten Sie, dass in allen Vorlagen die Versionsnummer `\version "2.12.0"` eingetragen ist. Es empfiehlt sich, in alle Dateien, unabhängig von ihrer Größe, den `\version`-Befehl einzufügen. Persönliche Erfahrung hat gezeigt, dass es ziemlich frustrierend sein kann zu erinnern, welche Programmversion man etwa vor einem Jahr verwendet hat. Auch `convert-ly` benötigt die Versionsnummer.
- **Benutzen Sie Überprüfungen**: Abschnitt “Oktavenüberprüfung” in *Notationsreferenz*, und Abschnitt “Takt- und Taktzahlüberprüfung” in *Notationsreferenz*. Wenn Sie hier und da diese Überprüfungen einfügen, finden Sie einen möglichen Fehler weit schneller. Wie oft aber ist „hier und da“? Das hängt von der Komplexität der Musik ab. ei einfachen Stücken reicht es vielleicht ein- oder zweimal, in sehr komplexer Musik sollte man sie vielleicht in jeden Takt einfügen.
- **Ein Takt pro Textzeile**. Wenn irgendetwas kompliziertes vorkommt, entweder in der Musik selber oder in der Anpassung der Ausgabe, empfiehlt es sich oft, nur einen Takt pro Zeile zu schreiben. Bildschirmplatz zu sparen, indem Sie acht Takte in eine Zeile zwängen, hilft nicht weiter, wenn Sie ihre Datei „debuggen“ müssen.
- **Kommentieren Sie ihre Dateien**. Benutzen Sie entweder Taktnummern (in regelmäßigen Abständen) oder Verweise auf musikalische Themen („Zweites Thema in den Geigen“, „vierte Variation“ usw.). Sie brauchen diese Kommentare vielleicht noch nicht, wenn Sie das Stück notieren, aber spätestens wenn Sie nach ein paar Jahren etwas verändern wollen oder Sie den Quelltext an einen Freund weitergeben wollen, ist es weitaus komplizierter, die Dateistruktur ohne Kommentare zu verstehen, als wenn Sie sie rechtzeitig eingefügt hätten.
- **Schreiben Sie Klammern mit Einrückung**. Viele Probleme entstehen durch ungerade Anzahl von `{` and `}`-Klammern.

- **Schreiben Sie Tondauerangaben** am Anfang von Abschnitten und Bezeichnern. Wenn Sie beispielsweise `c4 d e` am Anfang eines Abschnittes schreiben, ersparen Sie sich viele Probleme, wenn Sie ihre Musik eines Tages umarrangieren wollen.
- **Trennen Sie Einstellungen** von den eigentlichen Noten. Siehe auch Abschnitt 6.1.4 [Tipparbeit sparen durch Bezeichner und Funktionen], Seite 33 und Abschnitt 6.1.5 [Stil-Dateien], Seite 35.

6.1.2 Das Kopieren von bereits vorhandener Musik

Wenn Sie Musik aus einer fertigen Partitur kopieren (z. B. die LilyPond-Eingabe einer gedruckten Partitur):

- Schreiben Sie ein System ihrer Quelle nach dem anderen (aber trotzdem nur einen Takt pro Textzeile) und überprüfen Sie jedes System, nachdem Sie es fertig kopiert haben. Mit dem `showLastLength`- oder `showFirstLenght`-Befehl können Sie den Übersetzungsprozess beschleunigen. Siehe auch Abschnitt “Korrigierte Musik überspringen” in *Notationsreferenz*.
- Definieren Sie `mBreak = { \break }` und schreiben Sie `\mBreak` in der Quelldatei immer dann, wenn im Manuskript ein Zeilenumbruch vorkommt. Das macht es einfacher, die gesetzte Zeile mit den ursprünglichen Noten zu vergleichen. Wenn Sie die Partitur fertig gestellt haben, könne Sie `mBreak = { }`, also leer definieren, um diese manuellen Zeilenumbrüche zu entfernen. Damit kann dann LilyPond selber entscheiden, wohin es passende Zeilenumbrüche platziert.
- Wenn Sie eine Stimme für ein transponierendes Instrument als eine Variable notieren, wird empfohlen, dass die Noten von

  ```
  \transpose c klingende-Tonhöhe {...}
  ```

  eingefasst werden (wobei `klingende-Tonhöhe` die klingende Tonhöhe des Instruments ist), sodass die Noten innerhalb der Variable für klingendes C geschrieben sind. Sie können die Variable zurücktransponieren, wenn es nötig ist, aber Sie müssen es nicht tun. Fehler in Transpositionen sind treten seltener auf, wenn alle Noten in den Variablen für die gleiche Ausgangstonhöhe geschrieben werden.

  Denken Sie auch daran, dass Sie nur von/nach C transponieren. Das heißt, dass die einzigen anderen Tonhöhen, die Sie in Transpositionen benutzen, die Tonhöhen der Instrumente sind, für die Sie schreiben: `bes` für eine B-Trompete oder `aes` für eine As-Klarinette usw.

6.1.3 Große Projekte

Besonders wenn Sie an größeren Projekten arbeiten, ist es unumgänglich, dass Sie ihre LilyPond-Dateien klar strukturieren.

- **Verwenden Sie Variablen für jede Stimme**, innerhalb der Definition sollte so wenig Struktur wie möglich sein. Die Struktur des `\score`-Abschnittes verändert sich am ehesten, während die `violine`-Definition sich wahrscheinlich mit einer neuen Programmversion nicht verändern wird.

  ```
  violine = \relative c'' {
  g4 c'8. e16
  }
  ...
  \score {
    \new GrandStaff {
      \new Staff {
        \violine
      }
    }
  ```

```
}
```

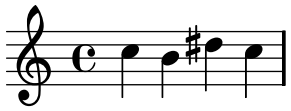
- **Trennen Sie Einstellungen von den Noten**. Diese Empfehlung wurde schon im Abschnitt Abschnitt 6.1.1 [Allgemeine Vorschläge], Seite 31 gegeben, aber für große Projekte ist es unumgänglich. Muss z. B. die Definition für `fdannp` verändert werden, so braucht man es nur einmal vorzunehmen und die Noten in der Geigenstimme, `violin`, bleiben unberührt.

```
fdannp = _\markup{
  \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violin = \relative c'' {
g4\fdannp c'8. e16
}
```

6.1.4 Tipparbeit sparen durch Bezeichner und Funktionen

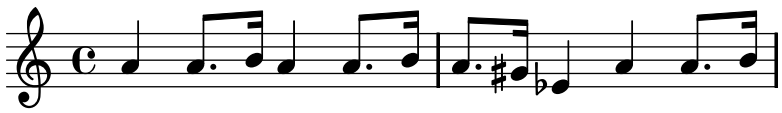
Bis jetzt haben Sie immer etwa solche Noten gesehen:

```
hornNotes = \relative c'' { c4 b dis c }
\score {
  {
    \hornNotes
  }
}
```



Das könnte auch nützlich in Minimal-Music sein:

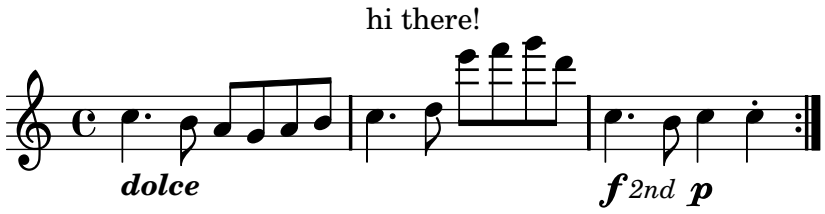
```
fragmentA = \relative c'' { a4 a8. b16 }
fragmentB = \relative c'' { a8. gis16 ees4 }
violin = \new Staff { \fragmentA \fragmentA \fragmentB \fragmentA }
\score {
  {
    \violin
  }
}
```



Sie können diese Bezeichner oder Variablen aber auch für (eigene) Einstellungen verwenden:

```
dolce = \markup{ \italic \bold dolce }
padText = { \once \override TextScript #'padding = #5.0 }
fthenp=_\markup{ \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violin = \relative c'' {
  \repeat volta 2 {
    c4._\dolce b8 a8 g a b |
    \padText
    c4.^"hi there!" d8 e' f g d |
    c,4.\fthenp b8 c4 c-. |
  }
}
\score {
```

```
  {
    \violin
  }
\layout{ragged-right=##t}
}
```



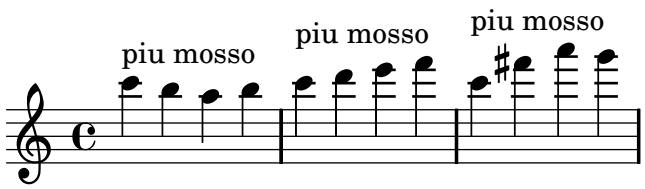
Die Variablen haben in diesem Beispiel deutlich die Tipparbeit erleichtert. Aber es lohnt sich, sie zu einzusetzen, auch wenn man sie nur einmal anwendet, denn sie vereinfachen die Struktur. Hier ist das vorangegangene Beispiel ohne Variablen. Es ist sehr viel komplizierter zu lesen, besonders die letzte Zeile.

```
violin = \relative c'' {
  \repeat volta 2 {
    c4._\markup{ \italic \bold dolce } b8 a8 g a b |
    \once \override TextScript #'padding = #5.0
    c4.^"hi there!" d8 e' f g d |
    c,4.\markup{ \dynamic f \italic \small { 2nd }
      \hspace #0.1 \dynamic p } b8 c4 c-. |
  }
}
```

Bis jetzt wurde nur statische Substitution vorgestellt – wenn LilyPond den Befehl `\padText` findet, wird er ersetzt durch durch unsere vorherige Definition (alles, was nach dem `padtext =` kommt).

LilyPond kennt aber auch nicht-statische Substitutionen (man kann sie sich als Funktionen vorstellen).

```
padText =
#(define-music-function (parser location padding) (number?)
  #{
    \once \override TextScript #'padding = #$padding
  #})

\relative c''' {
  c4^"piu mosso" b a b
  \padText #1.8
  c4^"piu mosso" d e f
  \padText #2.6
  c4^"piu mosso" fis a g
}
```



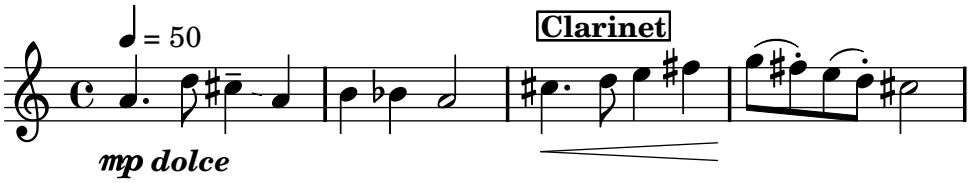
Die Benutzung von Variablen hilft auch, viele Schreibarbeit zu vermeiden, wenn die Eingabesyntax von LilyPond sich verändert (siehe auch Abschnitt 6.2.1 [Alte Dateien aktualisieren], Seite 38). Wenn nur eine einzige Definition (etwa `\dolce`) für alle Dateien verwendet wird (vgl. Abschnitt 6.1.5 [Stil-Dateien], Seite 35), muss nur diese einzige Definition verändert werden, wenn sich die Syntax ändert. Alle Verwendungen des Befehles beziehen sich dann auf die neue Definition.

6.1.5 Stil-Dateien

Die Ausgabe, die LilyPond erstellt, kann sehr stark modifiziert werden, siehe ⟨undefined⟩ [Die Ausgabe verändern], Seite ⟨undefined⟩ für Einzelheiten. Aber wie kann man diese Änderungen auf eine ganze Serie von Dateien anwenden? Oder die Einstellungen von den Noten trennen? Das Verfahren ist ziemlich einfach.

Hier ist ein Beispiel. Es ist nicht schlimm, wenn Sie nicht auf Anhieb die Abschnitte mit den ganzen `#()` verstehen. Das wird im Kapitel ⟨undefined⟩ [Fortgeschrittene Optimierungen mit Scheme], Seite ⟨undefined⟩ erklärt.

```
mpdolce = #(make-dynamic-script (markup #:hspace 0 #:translate '(5 . 0)
  #:line(#:dynamic "mp" #:text #:italic "dolce" )))

inst = #(define-music-function (parser location string) (string?)
  (make-music
    'TextScriptEvent
    'direction UP
    'text (markup #:bold (#:box string))))

\relative c'' {
  \tempo 4=50
  a4.\mpdolce d8 cis4--\glissando a | b4 bes a2
  \inst "Clarinet"
  cis4.\< d8 e4 fis | g8(\! fis)-. e( d)-. cis2
}
```



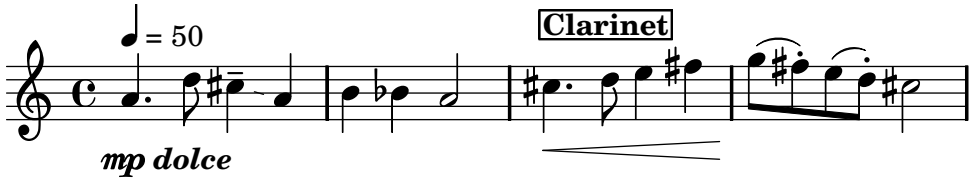
Es treten einige Probleme mit überlappenden Symbolen auf. Sie werden beseitigt mit den Tricks aus dem Kapitel ⟨undefined⟩ [Verschieben von Objekten], Seite ⟨undefined⟩. Aber auch die `mpdolce` und `inst`-Definitionen können verbessert werden. Sie produzieren das Ergebnis, das gewünscht ist, aber es wäre schön, sie auch in anderen Stücken verwenden zu können. Man könnte sie natürlich einfach kopieren und in die anderen Dateien einfügen, aber das ist lästig. Die Definitionen verbleiben auch in der Notendatei und diese `#()` sehen nicht wirklich schön aus. Sie sollen in einer anderen Datei versteckt werden:

```
%%% speichern in einer Datei "definitions.ily"
mpdolce = #(make-dynamic-script (markup #:hspace 0 #:translate '(5 . 0)
  #:line(#:dynamic "mp" #:text #:italic "dolce" )))

inst = #(define-music-function (parser location string) (string?)
  (make-music
    'TextScriptEvent
```

```
    'direction UP
    'text (markup #:bold (#:box string))))
```

Auf diese Datei kann dann später mit dem `\include`-Befehl im oberen Teil der LilyPond-Datei zurückgegriffen werden. (Die Erweiterung `.ily` wird benutzt, um diese eingefügte Datei, die nicht alleine kompiliert werden kann, von der Hauptdatei zu unterscheiden.) Jetzt muss natürlich noch die Notendatei angepasst werden (gespeichert unter dem Namen '`"music.ly"`').

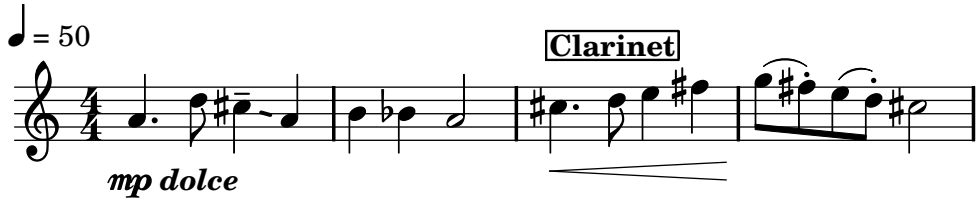
```
\include "definitions.ily"

\relative c'' {
  \tempo 4=50
  a4.\mpdolce d8 cis4--\glissando a | b4 bes a2
  \inst "Clarinet"
  cis4.\< d8 e4 fis | g8(\! fis)-. e( d)-. cis2
}
```



Das sieht schon besser aus, aber es sind noch einige Verbesserungen möglich. Das Glissando ist schwer zu sehen, also soll es etwas dicker erscheinen und dichter an den Notenköpfen gesetzt werden. Das Metronom-Zeichen soll über dem Schlüssel erscheinen, nicht über der ersten Note. Und schließlich kann unser Kompositionsprofessor „C“-Taktangaben überhaupt nicht leiden, also müssen sie in „4/4“ verändert werden.

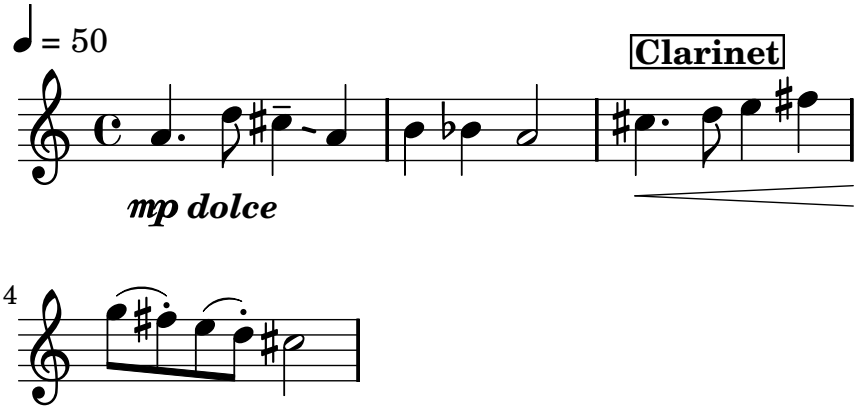
Diese Veränderungen sollten Sie aber nicht in der '`music.ly`'-Datei vornehmen. Ersetzen Sie die '`definitions.ily`'-Datei hiermit:

```
%%%  definitions.ily
mpdolce = #(make-dynamic-script (markup #:hspace 0 #:translate '(5 . 0)
  #:line( #:dynamic "mp" #:text #:italic "dolce" )))

inst = #(define-music-function (parser location string) (string?)
  (make-music
    'TextScriptEvent
    'direction UP
    'text (markup #:bold (#:box string))))

\layout{
  \context { \Score
    \override MetronomeMark #'extra-offset = #'(-9 . 0)
    \override MetronomeMark #'padding = #'3
  }
  \context { \Staff
    \override TimeSignature #'style = #'numbered
  }
  \context { \Voice
    \override Glissando #'thickness = #3
    \override Glissando #'gap = #0.1
  }
}
```



Das sieht schon besser aus! Aber angenommen Sie möchten dieses Stück jetzt veröffentlichen. Ihr Kompositionsprofessor mag die „C“-Taktangaben nicht, aber Sie finden sie irgendwie schöner. Also kopieren Sie die Datei ‘`definitions.ily`’ nach ‘`web-publish.ily`’ und verändern diese. Weil die Noten in einer PDF-Datei auf dem Bildschirm angezeigt werden sollen, bietet es sich auch an, die gesamte Ausgabe zu vergrößern.

```
%%%  definitions.ily
mpdolce = #(make-dynamic-script (markup #:hspace 0 #:translate '(5 . 0)
  #:line( #:dynamic "mp" #:text #:italic "dolce" )))

inst = #(define-music-function (parser location string) (string?)
  (make-music
    'TextScriptEvent
    'direction UP
    'text (markup #:bold (#:box string))))

#(set-global-staff-size 23)
\layout{
  \context { \Score
    \override MetronomeMark #'extra-offset = #'(-9 . 0)
    \override MetronomeMark #'padding = #'3
  }
  \context { \Staff
  }
  \context { \Voice
    \override Glissando #'thickness = #3
    \override Glissando #'gap = #0.1
  }
}
```



In der Notendatei muss jetzt nur noch `\include "definitions.ily"` durch `\include "web-publish.ily"` ausgetauscht werden. Das könnte man natürlich noch weiter vereinfachen. Also eine Datei ‘`definitions.ily`’, die nur die Definitionen von `mpdolce` und `inst` enthält, eine Datei ‘`web-publish.ily`’, die alle die Änderungen für den `\layout`-Abschnitt enthält und eine Datei ‘`university.ily`’ für eine Ausgabe, die den Wünschen des Professors entspricht. Der Anfang der ‘`music.ly`’-Datei würde dann so aussehen:

```
\include "definitions.ily"

%%%  Nur eine der beiden Zeilen auskommentieren!
\include "web-publish.ily"
%\include "university.ily"
```

Durch diese Herangehensweise kann auch bei der Erstellung von nur einer Ausgabeversion Arbeit gespart werden. Ich benutze ein halbes Dutzend verschiedener Stilvorlagen für meine Projekte. Jede Notationsdatei fängt an mit `\include "../global.ily"`, welches folgenden Inhalt hat:

```
%%%   global.ly
\version "2.12.0"
#(ly:set-option 'point-and-click #f)
\include "../init/init-defs.ily"
\include "../init/init-layout.ily"
\include "../init/init-headers.ily"
\include "../init/init-paper.ily"
```

6.2 Wenn etwas nicht funktioniert

6.2.1 Alte Dateien aktualisieren

Die Syntax von LilyPond verändert sich ab und zu. Wenn LilyPond besser wird, muss auch die Syntax (Eingabesprache) entsprechend angepasst werden. Teilweise machen diese Veränderungen die Eingabesprache einfacher lesbar, teilweise dienen sie dazu, neue Eigenschaften des Programmes benutzbar zu machen.

LilyPond stellt ein Programm bereit, das Aktualisierungen vereinfacht: `convert-ly`. Einzelheiten zur Programmbenutzung finden sich in Abschnitt “Dateien mit convert-ly aktualisieren” in *Anwendungsbenutzung*.

Leider kann `convert-ly` nicht alle Veränderungen der Syntax berücksichtigen. Hier werden einfache „Suchen und Ersetzen“-Veränderungen vorgenommen (wie etwa `raggedright` zu `ragged-right`), aber einige Veränderungen sind zu kompliziert. Die Syntax-Veränderungen, die das Programm nicht berücksichtigt, sind im Kapitel Abschnitt “Dateien mit convert-ly aktualisieren” in *Anwendungsbenutzung* aufgelistet.

Zum Beispiel wurden in LilyPond 2.4 und früheren Versionen Akzente und Umlaute mit LaTeX-Befehlen eingegeben, ein „No\"el“ etwa ergäbe das französische Wort für Weihnachten. In LilyPond 2.6 und höher müssen diese Sonderzeichen direkt als utf-8-Zeichen eingegeben werden, in diesem Fall also „ë“. `convert-ly` kann nicht alle dieser LaTeX-Befehle verändern, das muss manuell vorgenommen werden.

6.2.2 Fehlersuche (alles auseinandernehmen)

Früher oder später werden Sie in die Lage kommen, dass LilyPond Ihre Datei nicht kompilieren will. Die Information, die LilyPond während der Übersetzung gibt, können Ihnen helfen, den Fehler zu finden, aber in vielen Fällen müssen Sie nach der Fehlerquelle auf die Suche gehen.

Die besten Hilfsmittel sind in diesem Fall das Zeilen- und Blockkommentar (angezeigt durch `%` bzw. `%{ ... %}`). Wenn Sie nicht bestimmen können, wo sich das Problem befindet, beginnen Sie damit, große Teile des Quelltextes auszukommentieren. Nachdem Sie einen Teil auskommentiert haben, versuchen Sie, die Datei erneut zu übersetzen. Wenn es jetzt funktioniert, muss sich das Problem innerhalb der Kommentare befinden. Wenn es nicht funktioniert, müssen Sie weitere Teile auskommentieren bis sie eine Version haben, die funktioniert.

In Extremfällen bleibt nur noch solch ein Beispiel übrig:

```
\score {
  <<
    % \melody
    % \harmony
    % \bass
  >>
  \layout{}
}
```

(also eine Datei ohne Noten).

Geben Sie nicht auf, wenn das vorkommen sollte. Nehmen Sie das Kommentarzeichen von einem Teil wieder weg, sagen wir der Bassstimme, und schauen Sie, ob es funktioniert. Wenn nicht, dann kommentieren Sie die gesamte Bassstimme aus, aber nicht den `\bass`-Befehl in dem `\score`-Abschnitt:

```
bass = \relative c' {
%{
  c4 c c c
  d d d d
%}
}
```

Jetzt beginnen Sie damit, langsam Stück für Stück der Bassstimme wieder hineinzunehmen, bis Sie die problematische Zeile finden.

Eine andere nützliche Technik zur Problemlösung ist es, Abschnitt 6.2.3 [Minimalbeispiele], Seite 39 zu konstruieren.

6.2.3 Minimalbeispiele

Ein Minimalbeispiel ist eine Beispieldatei, die so klein wie möglich ist. Diese Beispiele sind sehr viel einfacher zu verstehen als die langen Originaldateien. Minimalbeispiele werden benutzt, um

- Fehlerberichte zu erstellen,
- eine Hilfeanfrage an die E-Mail-Liste zu schicken,
- Ein Beispiel zur LilyPond Schnipselsammlung hinzuzufügen.

Um ein Beispiel zu konstruieren, das so klein wie möglich ist, gibt es eine einfache Regel: Alles nicht Notwendige entfernen. Wenn Sie unnötige Teile einer Datei entfernen, bietet es sich an, sie auszukommentieren und nicht gleich zu löschen. Auf diese Weise können Sie eine Zeile leicht wieder mit aufnehmen, sollten Sie sie doch brauchen, anstatt sie von Anfang an neu zu schreiben.

Es gibt zwei Ausnahmen dieser „So klein wie möglich“-Regel:

- Fügen Sie immer einen `\version`Befehl ein.
- Wenn es möglich ist, benutzen Sie `\paper{ ragged-right = ##t }` am Beginn des Beispiels.

Der Sinn der Minimalbeispiele ist, dass sie einfach lesbar sind:

- Vermeiden Sie es, komplizierte Noten, Schlüssel oder Taktangaben zu verwenden, es sei denn, Sie wollen genau an diesen Elementen etwas demonstrieren.
- Benutzen Sie keine `\override`-Befehle, wenn sie nicht der Zweck des Beispieles sind.

6.3 Partituren und Stimmen

Orchesternoten werden alle zweimal gesetzt. Erstens als Stimmen für die Musiker, und dann als große Partitur für den Dirigenten. Mit Variablen kann hier doppelte Arbeit erspart werden. Die Musik muss nur einmal eingegeben werden und wird in einer Variable abgelegt. Der Inhalt dieser Variable wird dann benutzt, um sowohl die Stimme als auch die Partitur zu erstellen.

Es bietet sich an, die Noten in eigenen Dateien zu speichern. Sagen wir beispielsweise, dass in der Datei ‘Horn-Noten.ly’ die folgenden Noten eines Duetts für Horn und Fagott gespeichert sind:

```
HornNoten = \relative c {
  \time 2/4
  r4 f8 a cis4 f e d
}
```

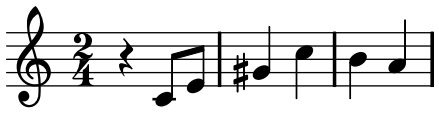
Daraus wird dann eine eigene Stimme gemacht, indem folgende Datei erstellt wird:

```
\include "Horn-Noten.ly"
\header {
  instrument = "Horn in F"
}

{
 \transpose f c' \HornNoten
}
```

Die Zeile

```
\include "Horn-Noten.ly"
```

setzt den Inhalt der Datei ‘Horn-Noten.ly’ an die Stelle des Befehls in die aktuelle Datei. Damit besteht also eine Definition für HornNoten, so dass die Variable verwendet werden kann. Der Befehl \transpose f c' zeigt an, dass das Argument, also \HornNoten, um eine Quinte nach oben transponiert wird. Klingendes ‚f‘ wird also als c' notiert. Das entspricht der Notation eines Waldhorns in F. Die Transposition zeigt die folgende Ausgabe:



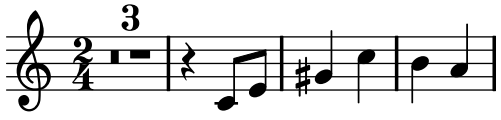
In der Musik für mehrere Instrumente kommt es oft vor, dass eine Stimme für mehrere Takte nicht spielt. Das wird mit einer besonderen Pause angezeigt, dem Pausenzeichen für mehrere Takte (engl. multi-measure rest). Sie wird mit dem *großen* Buchstaben ‘R’ eingegeben, gefolgt von einer Dauer (1 für eine Ganze, 2 für eine Halbe usw.). Indem man die Dauer multipliziert, können längere Pausen erstellt werden. Z. B. dauert diese Pause drei Takte eines 2/4-Taktes:

```
R2*3
```

Wenn die Stimme gedruckt wird, müssen diese Pausen zusammengezogen werden. Das wird durch eine Variable erreicht:

```
\set Score.skipBars = ##t
```

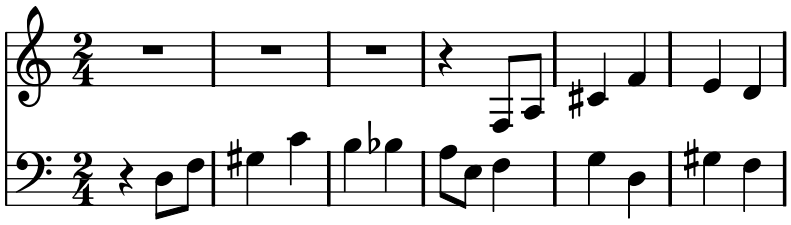
Dieser Befehl setzt die Eigenschaft des skipBars („überspringe Takte“) auf wahr (##t). Wenn diese Option und die Pause zu der Musik des Beispiels gesetzt wird, erhält man folgendes Ergebnis:



Die Partitur wird erstellt, indem alle Noten zusammengesetzt werden. Angenommen, die andere Stimme trägt den Namen FagottNoten und ist in der Datei ‘Fagott-Noten.ly’ gespeichert. Die Partitur sieht dann folgendermaßen aus:

```
\include "Fagott-Noten.ly"
\include "Horn-Noten.ly"

<<
  \new Staff \HornNoten
  \new Staff \FagottNoten
>>
```

Und mit LilyPond übersetzt:



6.4 Make und Makefiles

Fast alle Betriebssysteme, auf denen LilyPond benutzt werden kann, unterstützen ein Programm mit dem Namen `make`. Dieses Programm liest eine besondere Datei mit der Bezeichnung `Makefile`, die definiert, welche Dateien von welchen anderen Dateien abhängen und welche Befehle für das Betriebssystem nötig sind, um eine Datei aus einer anderen zu erstellen. Ein Makefile könnte etwa erklären, wie `ballad.pdf` und `ballad.midi` aus `ballad.ly` erstellt werden können, indem LilyPond aufgerufen wird.

Es gibt Fällen, wenn es sich sehr stark empfiehlt, ein `Makefile` für das aktuelle Projekt zu erstellen, entweder zur eigenen Bequemlichkeit, oder aber auch als Hilfe für andere, die vielleicht einmal die Quelldateien lesen und verstehen wollen. Insbesondere bei großen Projekten mit vielen eingefügten Dateien und unterschiedlichen Ausgabeoptionen (etwa Partitur, einzelne Stimmen, Dirigierpartitur, Klavierauszug usw.), aber auch bei Projekten, die komplizierte Programmaufrufe zur Verarbeitung erfordern (wenn man etwa mit `lilypond-book` arbeitet), lohnt sich die Erstellung einer Make-Datei. Diese Dateien können sehr unterschiedliche ausfallen, und ihre Komplexität und Flexibilität kann den Bedürfnissen aber auch Kenntnissen des Schreibers angepasst werden. Das Programm GNU Make ist auf GNU/Linux Distributionen und MacOS X installiert, aber es ist auch für Windows erhältlich.

Das **GNU Make Manual** gibt eine vollständige Anleitung, wie `make` benutzt werden kann. Hier sollen nur einige kleine Blicke auf die vielfältigen Möglichkeiten geworfen werden.

Die Befehle, um Regeln in einer Make-Datei zu erstellen, unterscheidet sich zwischen den Betriebssystemen. Die verschiedenen Linuxe und MacOS X benutzen `bash`, während unter Windows `cmd` eingesetzt wird. Unter MacOS X muss man das System so konfigurieren, dass die Kommandozeile benutzt wird. Hier einige Beispiele für Make-Dateien, mit Versionen für Linux/MacOS und Windows.

Das erste Beispiel ist für ein Orchesterstück in vier Stätzen unt mit der folgenden Dateistruktur:

```
Symphony/
|-- MIDI/
|-- Makefile
|-- Notes/
|   |-- cello.ily
|   |-- figures.ily
|   |-- horn.ily
```

```
|   |-- oboe.ily
|   |-- trioString.ily
|   |-- viola.ily
|   |-- violinOne.ily
|   `-- violinTwo.ily
|-- PDF/
|-- Parts/
|   |-- symphony-cello.ly
|   |-- symphony-horn.ly
|   |-- symphony-oboes.ly
|   |-- symphony-viola.ly
|   |-- symphony-violinOne.ly
|   `-- symphony-violinTwo.ly
|-- Scores/
|   |-- symphony.ly
|   |-- symphonyI.ly
|   |-- symphonyII.ly
|   |-- symphonyIII.ly
|   `-- symphonyIV.ly
`-- symphonyDefs.ily
```

Die `.ly`-Dateien un den Verzeichnissen `Scores` und `Parts` erhalten ihrere Noten aus `.ily`-Dateien, die sich im `Notes`-Verzeichnis befinden:

```
%%% Kopfzeile der Datei "symphony-cello.ly"
\include ../definitions.ily
\include ../Notes/cello.ily
```

Die Make-Datei hat die Ziele `score` (das gesamte Stück als große Partitur), `movements` (die einzelnen Sätze als große Partitur) und `parts` (die einzelnen Stimmen für die Spieler). Es gibt auch das Ziel `archive`, welches ein Tar-Archiv der Quelldateien erstellt, etwa um die Quellen über das Internet oder per E-Mail zu verteilen. Hier die Make-Datei für GNU/Linux oder MacOS X. Sie sollte unter dem Namen `Makefile` im obersten Verzeichnis des Projektes gespeichert werden:

> **Achtung:** Wenn ein Ziel oder eine Musterregel definiert ist, müssen die folgenden Zeilen mit Tabulatoren, nicht mit Leerzeichen beginnen.

```
# Namensstamm der Ausgabedateien
piece = symphony
# finde heraus, wieviele Prozessoren vorhanden sind
CPU_CORES=`cat /proc/cpuinfo | grep -m1 "cpu cores" | sed s/".*: "//`
# Der Befehl, um lilypond aufzurufen
LILY_CMD = lilypond -ddelete-intermediate-files \
                    -dno-point-and-click -djob-count=$(CPU_CORES)

# Die Endungen, die im Makefile benutzt werden
.SUFFIXES: .ly .ily .pdf .midi

# Eingabe- und Ausgabedateien werden in den Verzeichnissen durchsucht,
# die sich in der VPATH-Variable befinden.  Alle sind Unterverzeichnisse
# des aktuellen Verzeichnisses (angegeben durch die GNU make-Variable
# `CURDIR').
VPATH = \
```

```
  $(CURDIR)/Scores \
  $(CURDIR)/PDF \
  $(CURDIR)/Parts \
  $(CURDIR)/Notes

# Die Musterregel, um PDF und MIDI-Dateien aus der LY-Eingabedatei
# zu erstellen.  Die .pdf-Ausgabedateien werden in das
# `PDF'-Unterverzeichnis abgelegt, die .midi-Dateien in das
# `MIDI'-Unterverzeichnis.
%.pdf %.midi: %.ly
        $(LILY_CMD) $<; \           # this line begins with a tab
        if test -f "$*.pdf"; then \
            mv "$*.pdf" PDF/; \
        fi; \
        if test -f "$*.midi"; then \
            mv "$*.midi" MIDI/; \
        fi

notes = \
  cello.ily \
  horn.ily \
  oboe.ily \
  viola.ily \
  violinOne.ily \
  violinTwo.ily

# Abhängigkeiten der einzelnen Sätze.
$(piece)I.pdf: $(piece)I.ly $(notes)
$(piece)II.pdf: $(piece)II.ly $(notes)
$(piece)III.pdf: $(piece)III.ly $(notes)
$(piece)IV.pdf: $(piece)IV.ly $(notes)

# Abhängigkeiten der großen Partitur.
$(piece).pdf: $(piece).ly $(notes)

# Abhängigkeiten der Stimmen.
$(piece)-cello.pdf: $(piece)-cello.ly cello.ily
$(piece)-horn.pdf: $(piece)-horn.ly horn.ily
$(piece)-oboes.pdf: $(piece)-oboes.ly oboe.ily
$(piece)-viola.pdf: $(piece)-viola.ly viola.ily
$(piece)-violinOne.pdf: $(piece)-violinOne.ly violinOne.ily
$(piece)-violinTwo.pdf: $(piece)-violinTwo.ly violinTwo.ily

# `make score' eintippen, um die große Partitur mit allen vier
# Sätzen als eine Datei zu erstellen.
.PHONY: score
score: $(piece).pdf

# `make parts' tippen, um alle Stimmen zu erstellen.
# `make foo.pdf' tippen, um die Stimme für das Instrument `foo' zu erstellen.
# Beispiel: `make symphony-cello.pdf'.
.PHONY: parts
```

```
parts: $(piece)-cello.pdf \
       $(piece)-violinOne.pdf \
       $(piece)-violinTwo.pdf \
       $(piece)-viola.pdf \
       $(piece)-oboes.pdf \
       $(piece)-horn.pdf

# `make movements' tippen um Dateien für die vier Sätze einzeln zu erstellen.
.PHONY: movements
movements: $(piece)I.pdf \
           $(piece)II.pdf \
           $(piece)III.pdf \
           $(piece)IV.pdf

all: score parts movements

archive:
        tar -cvvf stamitz.tar \       # this line begins with a tab
        --exclude=*pdf --exclude=*~ \
        --exclude=*midi --exclude=*.tar \
        ../Stamitz/*
```

Unter Windows ergeben sich bestimmte Komplikationen. Nachdem man GNU Make für Windows heruntergeladen und installiert hat, muss man den richtigen Pfad in den Umgebungsvariablen des Systems setzen, damit die DOS-Kommandozeile das Make-Programm finden kann. Um das vorzunehmen, kann mit der rechten Maustaste auf "Arbeitsplatz" klicken, dann `Eigenschaften` und `Erweitert` geklickt werden. Hier wählt man `Umgebungsvariablen`. In der Liste `Systemvariablen` wählt man `Path` und mit einem Klick auf `Bearbeiten` kann man den Pfad zu der `.exe`-Datei von GNU Make hinzufügen, der etwa wie folgt aussieht:

```
C:\Program Files\GnuWin32\bin
```

Die Make-Datei selber muss auch angepasst werden, um unterschiedliche Shell-Befehle zu verwenden und mit Leerzeichen umgehen zu können, die sich in einigen Standardverzeichnissen unter Windows befinden. Das `archive`-Ziel wird entfernt, da Windows den `tar`-Befehl nicht kennt, und Windows benutzt auch eine andere Dateiendung für midi-Dateien.

```
## WINDOWS VERSION
##
piece = symphony
LILY_CMD = lilypond -ddelete-intermediate-files \
                    -dno-point-and-click \
                    -djob-count=$(NUMBER_OF_PROCESSORS)

# 8.3 Bezeichnung für CURDIR erhalten (Workaround wg. Leerzeichen in PATH)
workdir = $(shell for /f "tokens=*" %%b in ("$(CURDIR)") \
          do @echo %%~sb)

.SUFFIXES: .ly .ily .pdf .mid

VPATH = \
  $(workdir)/Scores \
  $(workdir)/PDF \
  $(workdir)/Parts \
  $(workdir)/Notes
```

```
%.pdf %.mid: %.ly
        $(LILY_CMD) $<      # diese Zeile beginnt mit Tabulator
        if exist "$*.pdf"  move /Y "$*.pdf"  PDF/ # begin with tab
        if exist "$*.mid" move /Y "$*.mid" MIDI/  # begin with tab

notes = \
  cello.ily \
  figures.ily \
  horn.ily \
  oboe.ily \
  trioString.ily \
  viola.ily \
  violinOne.ily \
  violinTwo.ily

$(piece)I.pdf: $(piece)I.ly $(notes)
$(piece)II.pdf: $(piece)II.ly $(notes)
$(piece)III.pdf: $(piece)III.ly $(notes)
$(piece)IV.pdf: $(piece)IV.ly $(notes)

$(piece).pdf: $(piece).ly $(notes)

$(piece)-cello.pdf: $(piece)-cello.ly cello.ily
$(piece)-horn.pdf: $(piece)-horn.ly horn.ily
$(piece)-oboes.pdf: $(piece)-oboes.ly oboe.ily
$(piece)-viola.pdf: $(piece)-viola.ly viola.ily
$(piece)-violinOne.pdf: $(piece)-violinOne.ly violinOne.ily
$(piece)-violinTwo.pdf: $(piece)-violinTwo.ly violinTwo.ily

.PHONY: score
score: $(piece).pdf

.PHONY: parts
parts: $(piece)-cello.pdf \
       $(piece)-violinOne.pdf \
       $(piece)-violinTwo.pdf \
       $(piece)-viola.pdf \
       $(piece)-oboes.pdf \
       $(piece)-horn.pdf

.PHONY: movements
movements: $(piece)I.pdf \
           $(piece)II.pdf \
           $(piece)III.pdf \
           $(piece)IV.pdf

all: score parts movements
```

Die nächste Make-Datei ist für ein `lilypond-book`-Dokument, das in LaTeX gesetzt wird. Das Projekt hat einen Index, welcher erfordert, dass der Befehl `latex` zweimal aufgerufen wird,

um die Verweise zu aktualisieren. Ausgabedateien werden in einem `out`-Verzeichnis für die .pdf-Dateien gespeichert und in `htmlout` für die html-Dateien.

```
SHELL=/bin/sh
FILE=myproject
OUTDIR=out
WEBDIR=htmlout
VIEWER=acroread
BROWSER=firefox
LILYBOOK_PDF=lilypond-book --output=$(OUTDIR) --pdf $(FILE).lytex
LILYBOOK_HTML=lilypond-book --output=$(WEBDIR) $(FILE).lytex
PDF=cd $(OUTDIR) && pdflatex $(FILE)
HTML=cd $(WEBDIR) && latex2html $(FILE)
INDEX=cd $(OUTDIR) && makeindex $(FILE)
PREVIEW=$(VIEWER) $(OUTDIR)/$(FILE).pdf &

all: pdf web keep

pdf:
        $(LILYBOOK_PDF)  # begin with tab
        $(PDF)           # begin with tab
        $(INDEX)         # begin with tab
        $(PDF)           # begin with tab
        $(PREVIEW)       # begin with tab

web:
        $(LILYBOOK_HTML) # begin with tab
        $(HTML)          # begin with tab
        cp -R $(WEBDIR)/$(FILE)/ ./  # begin with tab
        $(BROWSER) $(FILE)/$(FILE).html &  # begin with tab

keep: pdf
        cp $(OUTDIR)/$(FILE).pdf $(FILE).pdf  # begin with tab

clean:
        rm -rf $(OUTDIR) # begin with tab

web-clean:
        rm -rf $(WEBDIR) # begin with tab

archive:
        tar -cvvf myproject.tar \ # begin this line with tab
        --exclude=out/* \
        --exclude=htmlout/* \
        --exclude=myproject/* \
        --exclude=*midi \
        --exclude=*pdf \
        --exclude=*~ \
        ../MyProject/*
```

TODO: soll auch unter Windows funktionieren

Die vorige Make-Datei funktioniert nicht unter Windows. Als Alternative für Windows-Benutzer könnte man eine einfache batch-Datei erstellen, welche die erforderlichen Befehl

enthält. Sie kümmert sich nicht um Abhängigkeiten, wie es eine Make-Datei kann, aber wenigstens wird die Kompilation auf einen einzigen Befehl beschränkt. Das folgende kann als Datei `build.bat` oder `build.cmd` gespeichert werden. Die Batch-Datei kann auf der Kommandozeile aufgerufen werden oder einfach doppelt angeklickt werden.

```
lilypond-book --output=out --pdf myproject.lytex
cd out
pdflatex myproject
makeindex myproject
pdflatex myproject
cd ..
copy out\myproject.pdf MyProject.pdf
```

Siehe auch

Programmbenutzung: Abschnitt “Einrichtung für MacOS X” in *Anwendungsbenutzung*, Abschnitt “Benutzung auf der Kommandozeile” in *Anwendungsbenutzung*, Abschnitt “LilyPond-book” in *Anwendungsbenutzung*

7 Vorschläge, wie man Dateien schreibt

7.1 General suggestions

7.2 Typesetting existing music

7.3 Large projects

7.4 Troubleshooting

7.5 Make and Makefiles

Anhang A GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both

covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Anhang B LilyPond-Index

V

W