LEADSHEETS

v0.1   2015/01/11

Typesetting leadsheets and songbooks

Clemens Niederberger

https://github.com/cgnieder/leadsheets/

contact@mychemistry.eu

Table of Contents

Part I.
About the Package

> I like beautiful melodies telling me terrible things.
>
> *Tom Waits*

1. License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3 or later (`http://www.latex-project.org/lppl.txt`). The software has the status "maintained."

LEADSHEETS requires the bundles l3kernel [The13a] and l3packages [The13b] to be available. It also needs the package translations [Nie13].

2. Background

Over the years I repeatedly wanted to typeset simple leadsheets of songs, *i. e.*, song lyrics and the corresponding chords.[1] This is not too hard with standard LaTeX commands and environments but it is not very convenient… so looking for existing packages is the logical next step and I found two very promising packages: songs [Ham12] and songbook [Rat10]. However, both were note *quite* what I wanted. Just a bit too inflexible in the wrong places, needing tweaking here and there, and so on. On the other hand I had quite some code lying on my hard drive with various attempts of typesetting leadsheets. This package is now the attempt to have a clean, documented and customisable version of my code.[2]

---

1. I also have had the need (or let's say: wish) to typeset leadsheets in the style of the *The Real Book* – but this is a task where other software than LaTeX usually is far easier.
2. Plus new things!

3. The Structure of the Package

LEADSHEETS is a modular package and consists of several libraries.[3] If you just say

```
1 \usepackage{leadsheets}
```

Package option:
`full`

then the songs library is loaded.[4] If you want to ensure that *every* library is loaded so you don't have to think about it any more you can use

```
1 \usepackage[full]{leadsheets}
```

Package option:
`minimal`

If you instead use

```
1 \usepackage[minimal]{leadsheets}
```

then *no* libraries are loaded. In this case you have to load the libraries yourself in order to use the package.

`\useleadsheetslibraries{`⟨*comma list of libraries*⟩`}`
  With this command one or several of LEADSHEETS' libraries can be loaded.

`\useleadsheetslibrary{`⟨*library*⟩`}`
  With this command one of LEADSHEETS' libraries can be loaded.

The libraries are divided into two parts: libraries to be loaded by users and auxiliary libraries loaded by other libraries but not to be loaded directly by users.
  The user-libraries are the following ones:

Package option:
`musicsymbols`

musicsymbols This library makes the music symbol font provided by MusiXTeX available as text font. It is described in part II.

Package option:
`chordnames`

chordnames This library defines a few macros for typesetting of chord symbols. It is described in part III. It also loads the musicsymbols library.

---

3. This did seem like a good idea for maintenance when I first came up with it. Now I'm not so sure any more but I am way too lazy to change it back. So here we are...

4. Which currently means that *all* libraries are loaded as the songs library needs them.

<div style="margin-left: 2em;">

Package option:
musejazz

musejazz extends the chordnames library to use MuseScore's "MuseJazz" font, see section 6. It loads the chordnames library.

Package option:
songs

songs This is the main library. It defines everything necessary for the typesetting of the leadsheets. It currently loads *all other libraries, i. e.*, user libraries *and* internal libraries. This library is described in part IV.

</div>

Those libraries can also be loaded as a package option. If you say

```
1 \usepackage[musicsymbols]{leadsheets}
```

then *only* the musicsymbols library is loaded. Saying

```
1 \usepackage[songs]{leadsheets}
```

is the same as using the package *without* any options. "Without any" also means that neither musicsymbols, chordnames nor songs has been given as class option!

There is also a number of auxiliary libraries which are all needed by the songs library. The libraries are usually not described in sections of their own but as part of the songs library. However, the following list gives hints to where some of the concepts are explained:

properties This is an auxiliary library not meant to be loaded directly by users. It defines the necessary macros for song properties. See section 8.4 for more details on the concept.

transposing This is an auxiliary library not meant to be loaded directly by users. It defines a transposing mechanism for chord symbols. See section 11 for further details.

chords This is an auxiliary library not meant to be loaded directly by users. It defines the necessary macros for printing the chords in songs as well as the mechanism of remembering and recalling chord sequences of verses.

templates This is an auxiliary library not meant to be loaded directly by users. It defines the necessary macros for LEADSHEETS' template mechanism. See section 14 for details on the concept.

translations This is an auxiliary library not meant to be loaded directly by users. It provides a few translations for a number of printed strings. See section 15 for more information.

Part II.
The `musicsymbols` Library

> Music expresses that which cannot be said and on which it is impossible to be silent.
>
> *Victor Hugo*

The `musicsymbols` library is a very small library that makes the music symbol font provided by MusiXTeX available as text font and then uses it to define a number of symbols. This redefines the macros `\sharp`, `\flat` and `\natural`. All defined symbols are listed in table 1.

If you want to use the library standalone then you can say:

```
1 \usepackage[minimal]{leadsheets}
2 \useleadsheetslibraries{musicsymbols}
```

`musicsymbols` defines three further commands, namely `\musix`, `\textmusix` – a font switch and a text font command – and `\musicsymbol`. Those commands are meant for internal use only which is why they're not explained here.

TABLE 1: Symbols defined by `musicsymbols`.

Command	Symbol	Command	Symbol
\sharp	♯	\flat	♭
\doublesharp	𝄪	\doubleflat	𝄫
\natural	♮		
\trebleclef	𝄞	\bassclef	𝄢
\altoclef	𝄡		
\allabreve	¢	\meterC	C
\wholerest	−	\halfrest	−
\quarterrest	⸸	\eighthrest	↱
\sixteenthrest	⸷		

`musicsymbols` also defines a number of macros for denoting bars. Those macros are listed in table 2.

There are three macros that can be redefined to adjust the appearance of those bars:

TABLE 2: Bar symbols.

Macro	Symbol	Macro	Symbol
\normalbar	|	\leftrepeat	‖:
\rightrepeat	:‖	\leftrightrepeat	:‖:
\doublebar	‖	\stopbar	▊

\normalbarwidth   Default: .02em
  The thickness of a bar rule as used in \normalbar.

\thickbarwidth   Default: .15em
  The thickness of the thick bar rules as used for example in \leftrepeat or \stopbar.

\interbarwidth   Default: .1em
  The distance between two bar rules as in \doublebar or \stopbar.

Part III.
The `chordnames` Library (and the `musejazz` Library)

> I never practice my guitar – from time to time I just open the case and throw in a piece of raw meat.
>
> *Wes Montgomery*

4. The \\`chordname` Command

chordnames provides the command `\chordname{⟨chord⟩}` for convenient typesetting of chords:

```
1 \chordname{Bb7(#9)} \chordname{Bbb6}
2 \chordname{C#7(b9)} \chordname{C##13}
```
$B\flat^{7(\sharp9)}$ $B\flat\flat^{6}$ $C\sharp^{7(\flat9)}$ $C\times^{13}$

`\chordname{⟨chord⟩}`
  Typesetting chords. Inside the argument every # will be replaced by $\sharp$ and every b is replaced with $\flat$. Numerals and parentheses are typeset as superscripts. Everything between parentheses is always a superscript: `\chordname{F#7(#11)}` $F\sharp^{7(\sharp11)}$.

There are several token lists that are treated specially inside `\chordname`:

`^` – This token is replaced by `\textsuperscript`.

`ma` – The symbol for major chords. Per default this is empty. `\chordname{Gma}` G.

`mi` – The symbol for minor chords. Per default this is m. `\chordname{Gmi}` Gm.

`o` – The symbol for diminished chords. Per default this is `\textsuperscript{o}`. `\chordname{Go}` G^{o}.

`+` – The symbol for augmented chords. Per default this is `\textsuperscript{+}`. `\chordname{G+}` G^{+}.

`/o` – The symbol for half diminished chords. Per default this is `\textsuperscript{\o{}}`. `\chordname{G/o}` $G^{\o}$.

`#` – The "sharp" symbol. Per default this is `\sharp`. `\chordname{F#}` $F\sharp$.

`##` – The "double sharp" symbol. Per default this is `\doublesharp`. `\chordname{F##}` $F\times$.

b   – The "flat" symbol. Per default this is `\flat`. `\chordname{Eb}` E♭.

bb   – The "double flat" symbol. Per default this is `\doubleflat`. `\chordname{Ebb}` E♭♭.

b#   – Cancelling flat/sharp combination: this is removed.

#b   – Cancelling sharp/flat combination: this is removed.

add   – This is superscripted: `\chordname{Gadd9}` G^{add9}.

sus   – This is superscripted: `\chordname{Gsus4}` G^{sus4}.

dim   – This is superscripted: `\chordname{Gdim5}` G^{dim5}.

maj7   – This is superscripted: `\chordname{Gmaj7}` G^{maj7}.

maj9   – This is superscripted: `\chordname{Gmaj9}` G^{maj9}.

How these token lists are treated depends on optional settings:

```
1 \setchordnames{
2   major-seven = $\Delta$ ,
3   major-nine  = $\Delta$\textsuperscript{9}
4 }
5 \chordname{Gmaj7} \chordname{Gmaj9}
6 \chordname{G^6} \chordname{G6}
7 \chordname{G7^#5}
```

    $G\Delta$ $G\Delta^9$ G^6 G^6 $G^{7\sharp5}$

If you want to use the library standalone then you can say:

```
1 \usepackage[minimal]{leadsheets}
2 \useleadsheetslibraries{chordnames}
```

This also loads the `musicsymbols` library.

5. Options

Options are set with the command

`\setchordnames{`⟨*options*⟩`}`
  where ⟨*options*⟩ is a comma separated list of keyval options.

  Actually there's a second possibility: options can also be set with the command `\setleadsheets` (see section 8.3) if they're preceded by chordnames/ (including the slash).
     The options allow detailed customization of how chords are printed. It doesn't change the input syntax.

`format` = {⟨*code*⟩} (initially empty)
  Code inserted before a chord within the same group. Can be used for special formatting of the chords, with `\sffamily`, say.

`sharp` = {⟨*code*⟩} Default: `\sharp`
  The sharp symbol.

`flat` = {⟨*code*⟩} Default: `\flat`
  The flat symbol.

`double-sharp` = {⟨*code*⟩} Default: `\doublesharp`
  The double sharp symbol.

`double-flat` = {⟨*code*⟩} Default: `\doubleflat`
  The double flat symbol.

`aug` = {⟨*code*⟩} Default: +
  The augmented symbol.

`half-dim` = {⟨*code*⟩} Default: `\o{}`
  The half-diminished symbol.

`full-dim` = {⟨*code*⟩} Default: o
  The diminished symbol.

`dim` = {⟨*code*⟩} Default: `\textsuperscript{dim}`
  The token list dim.

`add` = {⟨*code*⟩} Default: `\textsuperscript{add}`
  The token list add.

`sus` = {⟨*code*⟩} Default: `\textsuperscript{sus}`
  The token list sus.

`major` = {⟨*code*⟩} (initially empty)
  The token list ma.

`minor` = {⟨*code*⟩} Default: m
  The token list mi.

major-seven = {⟨*code*⟩}   Default: ^{maj7}
  The token list maj7.

major-nine = {⟨*code*⟩}   Default: ^{maj9}
  The token list maj9.

There are two further options which concern the different notations of the pitches B and B♭ in German (and a few other European countries) and English.

input-notation = literal|german|english   Default: literal
  If set to german B should be input as H and B flat as B. If set to american B should be input as B and B flat as Bb.

output-notation = literal|german|english   Default: literal
  If set to german B is output as H and B-flat as B.

Setting one option to a value other than literal implicitly sets the other one to english except the other option already has been set. This is shown in the example below. Setting on option to literal again also sets the other option to literal. This also resets the options. Earlier settings are forgotten. What does that mean? Here is an exercise: which options are active at each step and at the end?

```
1 1. \setchordnames{ input-notation = german }
2 2. \setchordnames{ output-notation = german , input-notation = english }
3 3. \setchordnames{ input-notation = german }
4 4. \setchordnames{ output-notation = english , input-notation = literal }
5 5. \setchordnames{ output-notation = german }
```

Here's the solution:

  1. input-notation = {german} and output-notation = {english}

  2. input-notation = {english} and output-notation = {german}

  3. input-notation = {german} and output-notation = {german}

  4. input-notation = {literal} and output-notation = {literal}

  5. input-notation = {german} and output-notation = {english}

The following example shows the effects of different input and output notations.

```
1 input and output literal:\par
2 \chordname{Bb} \chordname{B} \chordname{B#}\par
3 \chordname{B} \chordname{H} \chordname{H#}\par
4 \medskip
5
6 \setchordnames{input-notation=german}
7 input German and output English:\par
8 \chordname{Bb} \chordname{B} \chordname{B#}\par
9 \chordname{B} \chordname{H} \chordname{H#}\par
10 \medskip
11
12 \setchordnames{output-notation=german}
13 input and output German:\par
14 \chordname{Bb} \chordname{B} \chordname{B#}\par
15 \chordname{B} \chordname{H} \chordname{H#}\par
16
17 \medskip
18 \setchordnames{input-notation=english}
19 input English and output German:\par
20 \chordname{Bb} \chordname{B} \chordname{B#}\par
21 \chordname{B} \chordname{H} \chordname{H#}\par
```

input and output literal:
B♭ B B♯
B H H♯

input German and output English:
B♭♭ B♭ B
B♭ B B♯

input and output German:
B♭ B H
B H H♯

input English and output German:
B H H♯
H H H♯

6. The musejazz Library

The musejazz library extends the chordnames library to use MuseScore's[5] font "MuseJazz" for the chord symbols. The library contains the two lines which explains why it requires LuaLATEX or XƎLATEX. Well, and the font, obviously.

---

5. http://musescore.com/

```
1 \RequirePackage{fontspec}
2 \newfontfamily\musejazz{MuseJazz}
```

Here is a small example of it's usage and the outcome:

```
1 % compile with LuaLaTeX or XeLaTeX
2 \documentclass[margin=4mm]{standalone}
3 \usepackage{leadsheets}
4 \useleadsheetslibrary{musejazz}
5 \begin{document}
6 \Huge\chordname{Bb7(#9)}\space\chordname{F##9}
7 \end{document}
```
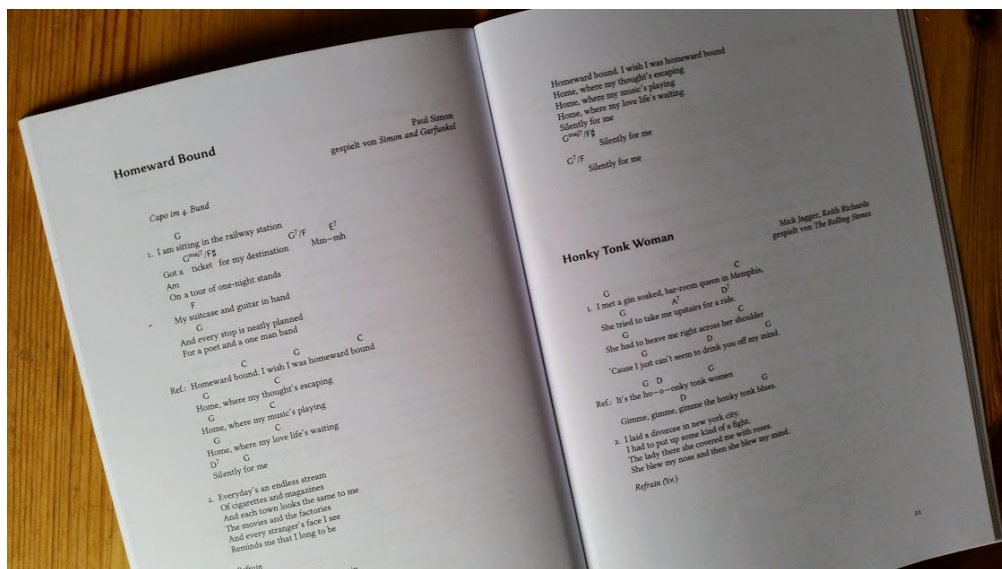
Bb7(#9) F×9

FIGURE 1: Even before officially publishing this bundle I used it for typesetting a small songbook!

Part IV.
The songs Library

> I don't care about the rules. If I
> don't break the rules at least ten
> times every song then I'm not do-
> ing my job.
>
> *Jeff Beck*

7. Background

The LEADSHEETS package allows for easy creation of leadsheets but it also can be used to create complete songbooks. The basic idea is that songs are typeset in the song environment. Each song gets a number of properties (see section 8.4) that determine how the title of the song is typeset. For the typesetting of the titles a template mechansim is used (see section 14.1). Songs can also be tagged. The tags then allow to typeset only songs matching a list of tags that is specified via an option.

8. The song Environment

\begin{song}[⟨*options*⟩]{⟨*properties*⟩}
   The main environment used to typeset songs. It has a mandatory argument where the song's

properties are set (*cf.* section 8.4). It also has an optional argument for setting options specific to the song.

8.1. A First Example

First let's take a look at an example:

```
1  \documentclass{article}
2  \usepackage{leadsheets}
3  \begin{document}
4
5  \begin{song}{title={Mary Had A Little Lamb}, music={Stevie Ray Vaughan},
6      lyrics={traditional}, tags={srv,blues}}
7  \begin{verse}
8    Mary had a little lamb \\
9    Its fleece was white as snow, yea \\
10   Everywhere the child went \\
11   The little lamb was sure to go, yea
12 \end{verse}
13 \begin{verse}
14   He followed her to school one day \\
15   And broke the teacher's rule \\
16   What a time did they have \\
17   That day at school
18 \end{verse}
19 \end{song}
20
21 \end{document}
```

Mary Had A Little Lamb

Mary had a little lamb
Its fleece was white as snow, yea
Everywhere the child went
The little lamb was sure to go, yea

He followed her to school one day
And broke the teacher's rule
What a time did they have
That day at school

Per default the songtitle simply is a \section* without any other additions. This is the songtitle template *minimal*, see section 14.1 for more details on those templates and how to create your own.

8.2. Using the song Environment

Inside the song environment a number of additional environments are used to specify the different parts of a song. They all are basically the same kind of environment, namely an itemize environment internally where the only \item has the name of the environment as option. The verse environment is a little bit different since verses can be numbered. If they are then each usage of verse inside song will step a vers number and print it (as option to the internal \item).

\begin{verse}[⟨*options*⟩]
  An environment for specifying the verses of a song.

\begin{chorus}[⟨*options*⟩]
  An environment for specifying the chorus of a song.
  This is the same as \begin{verse}[type=chorus,⟨*options*⟩].

\begin{intro}[⟨*options*⟩]
  An environment for specifying the intro of a song.
  This is the same as \begin{verse}[type=intro,⟨*options*⟩].

\begin{interlude}[⟨*options*⟩]
  An environment for specifying an interlude of a song.
  This is the same as \begin{verse}[type=interlude,⟨*options*⟩].

\begin{bridge}[⟨*bridge*⟩]
  An environment for specifying a bridge of a song.
  This is the same as \begin{verse}[type=bridge,⟨*options*⟩].

These environments and their options are described in more detail in sections 9 and 12.

8.3. Options

The options to the song environment are the same as the general options of LEADSHEETS. This means you can set the following options either local to a song or global for the whole document with this command:

\setleadsheets{⟨*options*⟩}
  Setup command for LEADSHEETS.

Although I used the word "global" above *all options are local to the current scope*!

title-template = {⟨*template name*⟩}   Default: minimal
  The songtitle template, see section 14.1 for details.

`chord-cs` = {⟨*cs*⟩}   Default: `\chordname`
  The command that is used to parse the chords. See section 10 for details. ⟨*cs*⟩ needs to be a command that takes a mandatory argument.

`song-format` = {⟨*TEX code*⟩}   (initially empty)
  ⟨*TEX code*⟩ is inserted *before* the song title at the beginning of the `song` environment.

`text-format` = {⟨*TEX code*⟩}   (initially empty)
  ⟨*TEX code*⟩ is inserted *after* the song title at the beginning of the `song` environment.

`print-tags` = {⟨*list of tags*⟩}
  A comma separated list of tags. When specified a song will only be printed if it is tagged with at least one of the tags in ⟨*list of tags*⟩.

`obey-lines` = `true`|`false`   Default: `false`
  An experimental option. Use at your own risk! If set to `true` then inside the `verse` like environments end-of-lines will be obeyed and start a new line. This comes with a price when using chords, see section 10.3.

`bar-shortcuts` = `true`|`false`   Default: `false`
  Makes the characters : and | active inside the `song` environment. See sections 8.5 and 13 for more details.

8.4. Song Properties

Songs can have a number of properties which basically are used in songtitle templates (see section 14.1). One specific property, `tags`, plays a different role, though.

`title` = {⟨*title*⟩}
  This is the main title of the song.

`subtitle` = {⟨*subtitle*⟩}
  A subtitle.

`short-title` = {⟨*short song title*⟩}
  A short title (may be useful in a template that writes the titles in `\section`s for a version to be used in the table of contents).

`sort-title` = {⟨*song title*⟩}
  If not set explicitly this property holds the same value as `title`.

`sort-short-title` = {⟨*short song title*⟩}
  If not set explicitly this property holds the same value as `short-title`.

`composer` = {⟨*composer*⟩}
  The composer of the song. As of now this accepts an arbitrary entry but maybe this will not be supported any more when indexing will be implemented. No promises.

`sort-composer` = {⟨*composer*⟩}
> If not set explicitly this property holds the same value as `composer`.

`lyrics` = {⟨*writer*⟩}
> Whoever wrote the lyrics if different from the composer. As of now this accepts an arbitrary entry but maybe this will not be supported any more when indexing will be implemented. No promises.

`sort-lyrics` = {⟨*writer*⟩}
> If not set explicitly this property holds the same value as `writer`.

`arr` = {⟨*arranger*⟩}
> Whoever arranged the song. As of now this accepts an arbitrary entry but maybe this will not be supported any more when indexing will be implemented. No promises.

`sort-arr` = {⟨*arranger*⟩}
> If not set explicitly this property holds the same value as `arr`.

`band` = {⟨*band*⟩}
> The band who plays or played the song.

`sort-band` = {⟨*band*⟩}
> If not set explicitly this property holds the same value as `band`.

`interpret` = {⟨*interpret*⟩}
> The interpret of the song. As of now this accepts an arbitrary entry but maybe this will not be supported any more when indexing will be implemented. No promises.

`sort-interpret` = {⟨*interpret*⟩}
> If not set explicitly this property holds the same value as `interpret`.

`genre` = {⟨*genre*⟩}
> The genre of the song.

`key` = {⟨*key*⟩}
> The key of the song. This property is used for transposing and must have a specific format then, see section 11.

`capo` = {⟨*fret*⟩}
> This property is used for transposing and for the \capo macro, see sections 8.6 and 11 .

`tempo` = {⟨*tempo*⟩}
> The tempo of the song.

`tags` = {⟨*tags*⟩}
> A comma separated list of tags. Those tags play a role for the option `print-tags`. When that option is used a song is only printed if it has at least one of the tags specified in the option.

There are three more properties, `counter`, `ID` and `height` that cannot be set but are set automatically for each song. The `counter` simply holds the number of the current song starting from 1 for the first song. The `ID` currently always ia song⟨*counter*⟩ where ⟨*counter*⟩ is the current `counter` value. The property `height` holds the height of the typeset song in pt. The height is determined by placing the body of the respective `song` environment in a vertical box and measuring the height and depth of the box. This is done in a measuring phase that can be tested in a songtitle template definition, see section 14.1 for details. *This is important since the property* `height` *is not available in the measuring phase but only afterwards!*

In principle all properties can get list of entries where items are separated with ␣and␣. Of course this doesn't make sense for each property – a song does only have one title. But a song can very well have more than one composer: think of the Beatles where most songs were written by Paul McCartney and John Lennon.[6]

It is possibly to define further such properties. For details see section 14.1.3.

8.5. Special Characters

Inside the `song` environment several characters don't have their usual catecory codes:

^   – category code 13 (active). It is a shortcut for the `\chord` command.

_   – category code 13 (active). It is a shortcut for the `\writechord` command.

| – category code 13 (active). Used for typesetting bars.

:   – category code 13 (active). Used for typesetting bars.

– category code 12 (other). Used fo chord names.

Actually the characters | and : are *not* changed per default. In order to do that you have to use the option `bar-shortcuts`.

For details on the usage of the characters | and : see section 13. The usage of chords is explained in section 10.

8.6. Capo Information

When you set the `capo` property the macro `\capo` writes

<div align="center">Capo: IV. fret</div>

What it writes *exactly* depends on a few settings: the `capo` property obviously, which determines the number that is printed. The translations for the "capo" and "fret" strings (see section 15 for details) and the setting of the following option:

`capo-nr-format` = `arabic|roman|Roman`   Default: `Roman`
  The format of the number printed by the `\capo` macro.

`capo-nr` = {⟨*code*⟩}   Default: `#1.`
  The code to print the number. In the code refer to the number with `#1`.

---

6. This is not quite true: most songs were mostly written either by Paul or John but legally usually both are the composers.

9. The verse Environment

\begin{verse}[⟨*options*⟩]

An environment for specifying the verses of a song.

\begin{verse*}[⟨*options*⟩]

The same as the verse environment but will always be unnumbered regardless of any option settings.

```
1  \documentclass{article}
2  \usepackage{leadsheets}
3  \setleadsheets{verse/numbered=true}
4  \begin{document}
5
6  \begin{song}{title=Foo}
7  \begin{verse}
8    Lorem ipsum dolor sit amet, consetetur sadipscing elitr,\\
9    sed diam nonumy eirmod tempor invidunt ut labore et dolore\\
10   magna aliquyam erat, sed diam voluptua.
11 \end{verse}
12 \begin{verse*}
13   Lorem ipsum dolor sit amet, consetetur sadipscing elitr,\\
14   sed diam nonumy eirmod tempor invidunt ut labore et dolore\\
15   magna aliquyam erat, sed diam voluptua.
16 \end{verse*}
17 \begin{verse}
18   Lorem ipsum dolor sit amet, consetetur sadipscing elitr,\\
19   sed diam nonumy eirmod tempor invidunt ut labore et dolore\\
20   magna aliquyam erat, sed diam voluptua.
21 \end{verse}
22 \end{song}
23
24 \end{document}
```

> **Foo**
>
> 1. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
>
>   Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
>
> 2. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

9.1. Options

The verse environment and all related environments have the following option:

format = {⟨*code*⟩}   (initially empty)
  ⟨*code*⟩ is inserted at the beginning of the environment and can thus be used to add formatting, *e. g.*, format = {\itshape}.

label-format = {⟨*code*⟩}   (initially empty)
  The same for the environment labels.

This can be used either locally, *i. e.*, as option to the corresponding environment, or for all environments of the same type using the setup command using the following syntax:

$$\setleadsheets{⟨\textit{env name}⟩/\texttt{format} = ⟨\textit{code}⟩}$$

```
1 \begin{verse}[format=\itshape]
2   Lorem ipsum dolor sit amet, consetetur sadipscing elitr,\\
3   sed diam nonumy eirmod tempor invidunt ut labore et dolore\\
4   magna aliquyam erat, sed diam voluptua.
5 \end{verse}
```

Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
sed diam nonumy eirmod tempor invidunt ut labore et dolore

> *magna aliquyam erat, sed diam voluptua.*

It is also possible so set the formatting for all related environments at once:

`verses-format` = {⟨*code*⟩}   (initially empty)
    Sets the format for all `verse` like environments.

`verses-label-format` = {⟨*code*⟩}   (initially empty)
    Sets the label format for all `verse` like environments.

Both options are overwritten if the options for a specific environment are set. That is, if you want all environments to have italic shape except for choruses, then you could do

```
1 \setleadsheets{
2   verses-format = \itshape ,
3   chorus/format =
4 }
```

10. Placing Chords

10.1. The Commands

Inside the `song` environment the characters ^ and _ are active characters.[7] ^ is a shortcut for the command `\chord`, _ is a shortcut for `\writechord`. Those commands have the following functions:

`\chord*-{`⟨*chord*⟩`}`⟨*text*⟩␣
    Places ⟨*chord*⟩ centered above ⟨*text*⟩. The starred version gobbles the trailing space while the unstarred version does not. Like the star the dash is optional. It sets the option `smash-next-chord` to `true`. ⟨*text*⟩ may be empty but the trailing space *must* be there. If ⟨*text*⟩ is empty then the chord is place above some horizontal space which can be set with the option `empty-chord-dim`.

`\writechord{`⟨*chord*⟩`}`
    This command transforms the chord according to the options `transpose` and `enharmonic` before printing it. This command can/should be used for placing chords inline or for typesetting the `key` property in a template. The command is used by `\chord` for the actual printing.

```
1 Text \chord{E7}text \chord*{B7}lon ger
    text
```
                                            E^7 B^7
                                      Text text longer text

---

7. There are more characters with a special function, see section 8.5

10.2. Usage

Note that per default the width of a chord is not ignored:

```
                                                 G♭m^7(♭5)
1 text \chord{Gbmi7(b5)}text text
                                          text   text   text
```

However, there is an option which sets the width of a chord to zero:

`smash-chords` = <u>true</u>|false   Default: `false`
  If set to true the width of the chords set with `\chord` is set to zero.

`smash-next-chord` = <u>true</u>|false   Default: `false`
  If set to true the width of the next chord set with `\chord` is set to zero.

```
                                                 G♭m^7(♭5)
                                              text text text
                                                 G♭m^7(♭5)
1 \setleadsheets{smash-next-chord=true}
                                              text   text   text
2 text \chord{Gbmi7(b5)}text text \par
                                                 G♭m^7(♭5)
3 text \chord{Gbmi7(b5)}text text \par
                                              text text text
4 \setleadsheets{smash-chords=true}
                                                 G♭m^7(♭5)
5 text \chord{Gbmi7(b5)}text text \par
                                              text text text
6 text \chord{Gbmi7(b5)}text text
```

Before we forget – there are more options:

`empty-chord-dim` = {⟨*dim*⟩}   Default: `1em`
  The horizontal space that is inserted if the ⟨*text*⟩ argument of `\chord` is empty.

`align-chords` = {⟨*col*⟩}   Default: `c`
  Determines how a chord is aligned with respect to the word it is placed above of. Valid input is
  any valid `tabular` column identifier.

`print-chords` = <u>true</u>|false   Default: `true`
  If set to false `\chord` won't print the chord but will do nothing instead. This options does not
  affect `\writechord`.

While `\chord` is available in the whole document the `^` syntax is – as mentioned before –
only available inside of the `song` environment.

```
1  \documentclass{article}
2  \usepackage{leadsheets}
3  \begin{document}
4
5  \begin{song}{title={Mary Had A Little Lamb}, music={Stevie Ray Vaughan},
6      lyrics={traditional}, tags={srv,blues}}
7  \begin{verse}
8    ^{E7} Mary had a little lamb ^{A7} \\
9    Its fleece was white as ^{E7}snow, yea \\
10   Everywhere the child ^{B7}went ^{A7} \\
11   The little lamb was sure to ^{E7}go, yea
12 \end{verse}
13 \begin{verse}
14   ^{E7} He followed her to school one day ^{A7} \\
15   And broke the teacher's ^{E7}rule \\
16   What a time did they ^{B7}have ^{A7} \\
17   That day at ^{E7}school {}
18 \end{verse}
19 \end{song}
20
21 \end{document}
```

Mary Had A Little Lamb

E⁷   A⁷

    Mary had a little lamb

                  E⁷

Its fleece was white as snow, yea

              B⁷ A⁷

Everywhere the child went

                   E⁷

The little lamb was sure to go, yea

E⁷   A⁷

    He followed her to school one day

                  E⁷

And broke the teacher's rule

              B⁷ A⁷

What a time did they have

     E⁷

You've probably noticed: chords are printed with `\chordname` in the default setting. You can change this with the option `chord-cs`. If you do then keep in mind that the input syntax will also change.

10.3. Caveat

If you use `obey-lines` = {true} you have to be careful when you place chords. If you place a chord over the last word in a line

```
1 ^{F#mi} You've been ^*{B}run nin', ^*{E}hid in' much too ^{A}long.
```

then the end of line that is used as the mandatory space argument for \chord won't be recognized as an end of line. Even worse: at the end of a verse like environment this may cause non-obvious errors. So in these cases you should always insert an explicit space by one of the following methods:

```
1 ^{F#mi} You've been ^*{B}run nin', ^*{E}hid in' much too ^{A}long. {}
2 ^{F#mi} You've been ^*{B}run nin', ^*{E}hid in' much too ^{A}long. \empty
3 ^{F#mi} You've been ^*{B}run nin', ^*{E}hid in' much too ^{A}long. \relax
```

10.4. Remembering Chords

LEADSHEETS has the option

remember-chords = <u>true</u>|false   Default: false

If set to true the chords in the *first* appearance of a verse like environment are remembered. In the next appearances of said environment the shortcut ^ has changed its meaning and inserts a chord automatically.

Let's take at look at an example to see what this means:

```
1 \definesongtitletemplate{empty}{}
2 \begin{song}[verse/numbered,remember-chords,title-template=empty]{title=foobar}
3 \begin{verse}
4  ^{G}Lorem ipsum ^{C}dolor sit ^{E7}amet, consetetur ^{Bb7(b5)}sadipscing
5 \end{verse}
6 \begin{verse}
7  ^Lorem ipsum ^dolor sit ^amet, consetetur ^sadipscing
8 \end{verse}
9 \end{song}
```

<div style="text-align:center">

     G   C E^7 $B\flat^{7(\flat5)}$

1. Lorem ipsum dolor sit amet, consetetur sadipscing

     G   C E^7 $B\flat^{7(\flat5)}$

2. Lorem ipsum dolor sit amet, consetetur sadipscing

</div>

In this example the chords used in the first verse environment have been remembered and in the second verse environment the ^ shortcut inserted the corresponding chords in the order they had been specified the first time. It is important when using this that you don't use the ^ shortcut in subsequent environments more often than the first time. It will produce an error otherwise. You can use it less, of course.

The ^ shortcut still has the *the same* syntax as \chord with one exception: it lacks the mandatory argument ⟨*chord*⟩ (since it's obviously not needed). It has the optional ∗ and -, though, as well as the mandatory space (␣)!

Each verse like environment (see section 12 for more information) is treated uniquely by this mechansim:

```
1  \definesongtitletemplate{empty}{}
2  \begin{song}[verse/numbered,remember-chords,title-template=empty]{title=foobar}
3  \begin{verse}
4    ^{G}Lorem ipsum ^{C}dolor sit ^{E7}amet, consetetur ^{Bb7(b5)}sadipscing
5  \end{verse}
6  \begin{chorus}
7    ^{F}Lorem ipsum ^{Gmi}dolor sit amet, consetetur ^{C7}sadipscing
8  \end{chorus}
9  \begin{verse}
10   ^Lorem ipsum ^dolor sit ^amet, consetetur ^sadipscing
11 \end{verse}
12 \begin{chorus}
13   ^Lorem ipsum ^dolor sit amet, consetetur ^sadipscing
14 \end{chorus}
15 \end{song}
```

       G   C   E^7 B♭$^{7(♭5)}$

1. Lorem ipsum dolor sit amet, consetetur sadipscing

       F   Gm   C^7

Chorus: Lorem ipsum dolor sit amet, consetetur sadipscing

       G   C   E^7 B♭$^{7(♭5)}$

2. Lorem ipsum dolor sit amet, consetetur sadipscing

       F   Gm   C^7

Chorus: Lorem ipsum dolor sit amet, consetetur sadipscing

This is important: verse and verse∗ are treated as two different environments, the same holds for all starred verse like environments! If you want to recall the chords of a different

type of environment, then you can use the option `recall-chords`:

```
1  \definesongtitletemplate{empty}{}
2  \begin{song}[verse/numbered,remember-chords,title-template=empty]{title=foobar}
3  \begin{verse}
4    ^{G}Lorem ipsum ^{C}dolor sit ^{E7}amet, consetetur ^{Bb7(b5)}sadipscing
5  \end{verse}
6  \begin{chorus}
7    ^{F}Lorem ipsum ^{Gmi}dolor sit amet, consetetur ^{C7}sadipscing
8  \end{chorus}
9  \begin{verse}
10   ^Lorem ipsum ^dolor sit ^amet, consetetur ^sadipscing
11 \end{verse}
12 \begin{chorus}[recall-chords=verse]
13   ^Lorem ipsum ^dolor sit amet, consetetur ^sadipscing
14 \end{chorus}
15 \end{song}
```

        G   C E^7 $B\flat^{7(\flat5)}$

1. Lorem ipsum dolor sit amet, consetetur sadipscing

        F   Gm C^7

Chorus: Lorem ipsum dolor sit amet, consetetur sadipscing

        G   C E^7 $B\flat^{7(\flat5)}$

2. Lorem ipsum dolor sit amet, consetetur sadipscing

        G   C E^7

Chorus: Lorem ipsum dolor sit amet, consetetur sadipscing

11. Transposing

Provided a song has the property `key` *and* the key is given as one of the twelve "usual" keys, *i.e.*, one of the keys given in table 3, the chords of a song can be transposed.

`transpose` = {⟨*number*⟩}
  Transposes the chords of a song by ⟨*number*⟩ of semitones. ⟨*number*⟩ can be a negative number, then the chords are transposed down.

`enharmonic` = `sharp|flat`
  Suppose you transpose a song in the key of E down a semitone. LEADSHEETS will then transpose

to the key of E♭. It always chooses the key whose signature has less accidentals. You can force a variant, though, by using this option. With enharmonic = {sharp} LEADSHEETS would have chosen D♯ instead of E♭.

transpose-capo = <u>true</u>|false

When this is set to true chords are transposed down on semitone per capo fret.

The transposing mechanism relies on the chordnames input syntax which means that if you change chord-cs horrible things may happen. *So don't change chord-cs and use transpose at the same time!*

TABLE 3: Allowed keys for the key property.

Key	Input	Key	Input	Key	Input	Key	Input	Key	Input	Key	Input
C	C			C	Cma			Am	Ami		
G	G	F	F	G	Gma	F	Fma	Em	Emi	Dm	Dmi
D	D	B♭	Bb	D	Dma	B♭	Bbma	Bm	Bmi	Gm	Gmi
A	A	E♭	Eb	A	Ama	E♭	Ebma	F♯m	F#mi	Cm	Cmi
E	E	A♭	Ab	E	Ema	A♭	Abma	C♯m	C#mi	Fm	Fmi
B	B	D♭	Db	B	Bma	D♭	Dbma	G♯m	G#mi	F♭m	Fbmi
F♯	F#	G♭	Gb	F♯	F#ma	G♭	Gbma	D♯m	D#mi	E♭m	Ebmi

```
1  \documentclass{article}
2  \usepackage{leadsheets}
3  \begin{document}
4
5  \begin{song}[transpose=2]{title={Mary Had A Little Lamb}, music={Stevie Ray
6      Vaughan}, lyrics={traditional}, tags={srv,blues},key=E}
7  \begin{verse}
8    ^{E7} Mary had a little lamb ^{A7} \\
9    Its fleece was white as ^{E7}snow, yea \\
10   Everywhere the child ^{B7}went ^{A7} \\
11   The little lamb was sure to ^{E7}go, yea
12 \end{verse}
13 \begin{verse}
14   ^{E7} He followed her to school one day ^{A7} \\
15   And broke the teacher's ^{E7}rule \\
16   What a time did they ^{B7}have ^{A7} \\
17   That day at ^{E7}school {}
18 \end{verse}
19 \end{song}
20
21 \end{document}
```

Mary Had A Little Lamb

F♯⁷ B⁷
: Mary had a little lamb
  F♯⁷
Its fleece was white as snow, yea
  C♯⁷ B⁷
Everywhere the child went
  F♯⁷
The little lamb was sure to go, yea

F♯⁷ B⁷
: He followed her to school one day
  F♯⁷
And broke the teacher's rule
  C♯⁷ B⁷
What a time did they have
  F♯⁷

12. Other verse-like Environments

Songs can have lots of different kinds of parts: verses, choruses, bridges, intros, outros, and so on. Typographically they're all more or less the same, at least for the purpose of this package. This means we'd ideally have environments for all of these parts with a distinct name in order to get a clean source. At the same time these environments should all behave basically the same. This is what the environments described in the following sections are for.

12.1. Available Environments

\begin{chorus}[⟨*options*⟩]
  An environment for specifying the chorus of a song.

\begin{chorus*}[⟨*options*⟩]
  The same as chorus but does not display the label.

\begin{intro}[⟨*options*⟩]
  An environment for specifying the intro of a song.

\begin{intro*}[⟨*options*⟩]
  The same as intro but does not display the label.

\begin{interlude}[⟨*options*⟩]
  An environment for specifying an interlude of a song.

\begin{bridge}[⟨*bridge*⟩]
  An environment for specifying a bridge of a song.

\begin{info}[⟨*options*⟩]
  An environment for specifying arbitrary information. This environment has no label.

\begin{solo}[⟨*options*⟩]
   An environment for specifying a solo to a song.

\begin{solo*}[⟨*options*⟩]
   The same as solo but does not display the label.

12.2. Own verse-like Environments

All environments mentioned in the previous section were defined with this command:

\newversetype*{⟨*environment name*⟩}[⟨*default options*⟩]
   This defines an environment ⟨*environment name*⟩. This environment uses the verse-type template *itemize* (see section 14.2) unless specified differently in the ⟨*default options*⟩. The starred version defines an environment which hides the displayed name. More precisely: the macro \verselabel that is used in verse-type template definitions prints nothing in an environment defined with \newversetype*.

   This also defines a translation string (see section 15) leadsheets/⟨*environment name*⟩ with both an empty translation fallback and English translation unless specified differently with the name option.

   At last this defines a new counter ⟨*environment name*⟩ and sets \the⟨*environment name*⟩ to \arabic{environment name}..

   As mentioned before in section 9.1 all environments defined this way have the options format and label-format. They have more options. Here's a complete list:

format = {⟨*code*⟩}   (initially empty)
   ⟨*code*⟩ is inserted at the beginning of the environment and can thus be used to add formatting, *e. g.*, format = {\itshape}.

label-format = {⟨*code*⟩}   (initially empty)
   The same for the environment labels.

after-label = {⟨*code*⟩}   Default: :
   ⟨*code*⟩ is inserted in the label after the label text.

name = {⟨*name*⟩}   (initially empty)
   The translation fallback and English translation of the environment. This should only be used with \newversetype. Later changes should be made with \DeclareTranslation (see section 15).

template = {⟨*template*⟩}   Default: itemize
   The verse-type template used for the environment, see section 14.2 for details.

numbered = <u>true</u>|false   Default: false
   If set to true \verselabel (used in verse-type template definitions, see section 14.2) adds a number after the name.

named = <u>true</u>|false   Default: `true`

> If set to `true` \verselabel (used in verse-type template definitions, see section 14.2) prints the name of the current environment (as determined by the translation of the corresponding string, see also section 15).

recall-chords = {⟨*environment*⟩}

> An option to be used with the `remember-chords` mechanism, see section 10.4 for an example.

There are also a number of general options for setting the defaults of some options for all environments:

verses-format = {⟨*code*⟩}   (initially empty)

> Sets the format for all `verse` like environments.

verses-label-format = {⟨*code*⟩}   (initially empty)

> Sets the label format for all `verse` like environments.

verses-after-label = {⟨*code*⟩}   Default: :

> Defeault ⟨*code*⟩ that is inserted in the label after the label text of `verse` like environments.

This options only determine the formatting of an environment if the corresponding options of the environment hasn't been set.

Let's summarize: the label text of these environments is built of three items in the following order:

1. The ⟨*code*⟩ set with the corresponding `label-format` option.

2. The label text as defined as second argument to \newversetype or as declared through the corresponding translation.

3. The ⟨*code*⟩ set with the corresponding `after-label` option.

```
1 \newversetype{foo}{Foo}
2 \setleadsheets{
3   foo/label-format = \bfseries ,
4   foo/after-label  = ~$\Rightarrow$
5 }
6 \begin{foo}
7   Lorem ipsum dolor sit amet, consetetur sadipscing elitr,\\
8   sed diam nonumy eirmod tempor invidunt ut labore et dolore\\
9   magna aliquyam erat, sed diam voluptua.
10 \end{foo}
```

Foo

⇒ Lorem ipsum dolor sit amet, consetetur sadipscing elitr,

> sed diam nonumy eirmod tempor invidunt ut labore et dolore
> magna aliquyam erat, sed diam voluptua.

And just to give you some more examples here is how some of the existing environments are defined:

```
1 \newversetype{verse}[ name=Verse, named=false, after-label= ]
2 \newversetype*{verse*}
3 \newversetype{chorus}[ name=Chorus ]
4 \newversetype*{chorus*}
```

13. Typesetting Bars

Sometimes it can be useful to typeset the chord scheme of a song. Then one should be able to indicate start and beginnings of bars, maybe indicate repeats and so on. While this is obviously possible with the macros provided by the musicsymbols package listed in table 2 it may be more convenient to have a shorter syntax. This is why inside the song environment some characters can be made (or are) active (see section 8.5). For the typesetting of bars this are the characters : nor |. Per default they are not active, though. If you want to use the shortcut syntax you have to use the option bar-shortcuts. Here's a short example that emulates the behaviour by setting the characters active explicitly:

```
1 \catcode'|=\active
2 \catcode':=\active
3 |: repeat | this :|: and | this :| \par
4 | this | part | ends | here || \par
5 | the | song | is over | now |||
```

‖: repeat │ this :‖: and │ this :‖
│ this │ part │ ends │ here ‖
│ the │ song │ is over │ now ‖

All possibly combinations that have a special definition are shown in the example above. The replacements that are done internally are these:

| – \normalbar\space (the space is there because otherwise it eats following spaces which would be annoying)

31

```
|:   – \leftrepeat
```

```
:|   – \rightrepeat
```

```
:|:  – \leftrightrepeat
```

```
||   – \doublebar
```

```
|||  – \stopbar
```

14. Templates

14.1. Title Templates

14.1.1. Background

The titles of songs set with the song environment are displayed according to the chosen title template. It is chosen through the option `title-template` which can be set with `\setleadsheets` or as option to a specific song environment. LEADSHEETS provides few predefined templates and an easy mechansim to define own templates.

14.1.2. Existing Templates

Currently LEADSHEETS provides two templates:

minimal This only typesettes the song title in a `\section*`.

tabular This typesets the song title in a `\section` and prints some song properties in a tabular below it. This template needs the array [MC08] package loaded.

14.1.3. Own Templates

The principle is pretty straight forward: templates are defined with the following command:

`\definesongtitletemplate{⟨name⟩}{⟨code⟩}`
 This defines the template ⟨*name*⟩.

 Inside of ⟨*code*⟩ any code can be used. The idea is that you use the commands presented below to insert song properties where you want them.
  First there are two commands related to defining new properties:

`\definesongproperty{⟨property⟩}`
 Defines a new property ⟨*property*⟩. All existing properties have been defined this way. The command can only be used in the preamble.

`\copysongproperty{⟨from⟩}{⟨to⟩}`
 Copies the values of property ⟨*from*⟩ to property ⟨*to*⟩ if property ⟨*to*⟩ has not been set but property ⟨*from*⟩ has been. For example all sort-⟨*property*⟩ properties have been treated this way so they have the ⟨*property*⟩ value as fallback. The command can only be used in the preamble.

Then there are a number of commands related to retrieving and using the values of properties. All these commands only make sense inside a title template definition (see section 14.1). Some of the commands are expandable which means they can be used in an \edef like context, *i. e.*, they are also suitable for writing the property values to the table of contents or other auxiliary files.

* \songproperty{⟨*property*⟩}
  Retrieves property ⟨*property*⟩.

\printsongpropertylist[⟨*code*⟩]{⟨*property*⟩}{⟨*between two*⟩}{⟨*between more*⟩}{⟨*between last two*⟩}   Default: \@firstofone
  Prints a property list ⟨*property*⟩ separated with ⟨*between two*⟩ if the list contains only two items and separated with ⟨*between more*⟩ and ⟨*between last two*⟩ if the list contains more than two items. ⟨*code*⟩ is placed directly in front of each item and items are surrounded with braces which means that the last token in ⟨*code*⟩ may be a macro with a mandatory argument.

\usesongpropertylist[⟨*code*⟩]{⟨*property*⟩}{⟨*between*⟩}   Default: \@firstofone
  Like \printsongpropertylist but separates items with ⟨*between*⟩ regardless of the length of the list.

* \forsongpropertylist{⟨*property*⟩}{⟨*code*⟩}
  Places all items of the property list ⟨*property*⟩ in the input stream, each item preceded with ⟨*code*⟩. Items are surrounded with braces which means that the last token in ⟨*code*⟩ may be a macro with a mandatory argument.

* \ifsongproperty{⟨*property*⟩}{⟨*true*⟩}{⟨*false*⟩}
  Checks if property ⟨*property*⟩ has been set.

\ifsongpropertiesequal{⟨*property 1*⟩}{⟨*property 2*⟩}{⟨*true*⟩}{⟨*false*⟩}
  Checks if properties ⟨*property 1*⟩ and ⟨*property 2*⟩ have been set to the same value.

* \ifsongmeasuring{⟨*true*⟩}{⟨*false*⟩}
  LEADSHEETS measures the height of a song body before it typesets it and it can be necessary in a template to know if the measuring phase is active or not. For example the song property height should only be used if *not* in the measuring phase: it's value get's determined there and is not yet available.

\expandcode{⟨*code*⟩}
  Exhaustively expands ⟨*code*⟩. Experienced users won't need this. It is essentially

  \begingroup\edef\x{\endgroup⟨*code*⟩}\x.

  (More precisely it is a wrapper for the expl3 function \use:x.) This means that any # needs to doubled. Inside the argument of this commands non-robust macros that should not be expanded need to be prefixed with \noexpand.

With the right template definition you can index composers, interprets, song titles, … You can write tables of contents for properties such as song titles, and so on, and so on. LEADSHEETS does not do this for you and it may require some experience to create templates which do all this.

14.1.4. Examples

In order to give you an idea on how to use songtitle templates I'll show you how the existing ones are defined and one new definition.

The *minimal* template   This is quite short and self-explaining.

```
1 \definesongtitletemplate{minimal}{\section*{\songproperty{title}}}
```

A custom template   Now let's see an example for a newly defined template. It's nearly as simple as the *minimal* template.

```
1 \documentclass{article}
2 \usepackage{leadsheets}
3 \definesongtitletemplate{custom}{
4   \ifsongmeasuring
5     {\section*}
6     {\section}{%
7     \songproperty{title}%
8     \ifsongproperty{music}
9       { (music by \printsongpropertylist{music}{ \& }{, }{ \& })}
10      {}%
11  }
12 }
13 \setleadsheets{title-template = custom}
14 \begin{document}
15
16 \begin{song}{title={Mary Had A Little Lamb}, music={Stevie Ray Vaughan},
17    lyrics={traditional}, tags={srv,blues}}
18 \begin{verse}
19   Mary had a little lamb \\
20   Its fleece was white as snow, yea \\
21   Everywhere the child went \\
22   The little lamb was sure to go, yea
23 \end{verse}
24 \begin{verse}
25   He followed her to school one day \\
26   And broke the teacher's rule \\
27   What a time did they have \\
28   That day at school
29 \end{verse}
30 \end{song}
31
```

```
32 \end{document}
```

> ## 1 Mary Had A Little Lamb (music by Stevie Ray Vaughan)
>
> Mary had a little lamb
> Its fleece was white as snow, yea
> Everywhere the child went
> The little lamb was sure to go, yea
>
> He followed her to school one day
> And broke the teacher's rule
> What a time did they have
> That day at school

The *tabular* template   This one is a lot more advanced and demonstrates various of the available commands.

```
1  \definesongtitletemplate{tabular}{
2    \ifsongmeasuring
3      {\section*}
4      {\section}%
5      {\songproperty{title}}
6    \begingroup\footnotesize
7    \begin{tabular}{
8        @{}
9        >{\raggedright\arraybackslash}p{.5\linewidth}
10       @{}
11       >{\raggedleft\arraybackslash}p{.5\linewidth}
12       @{}
13    }
14    \ifsongproperty{interpret}
15      {\GetTranslation{leadsheets/interpret}}
16      {}%
17    \ifsongproperty{composer}
18      {%
19        &
20        \GetTranslation{leadsheets/composer}: %
21        \printsongpropertylist{composer}{ \& }{, }{ \& }
22        \ifsongproperty{lyrics}
23          {
24            \\ &
```

```
25        \GetTranslation{leadsheets/lyrics}: %
26        \printsongpropertylist{lyrics}{ \& }{, }{ \& }
27      }
28      {}%
29    }
30    {}%
31    \ifsongproperty{interpret}{\\}{\ifsongproperty{composer}{\\}{}}%
32    \ifsongproperty{genre}
33      {& Genre: \songproperty{genre} \\}
34      {}%
35    \ifsongproperty{tempo}
36      {& Tempo: \songproperty{tempo} \\}
37      {}%
38    \ifsongproperty{key}
39      {%
40        & \setchordnames{
41            major = -\GetTranslation{leadsheets/major} ,
42            minor = -\GetTranslation{leadsheets/minor}
43          }%
44        \GetTranslation{leadsheets/key}: %
45        \expandcode{\writechord{\songproperty{key}}} \\%
46      }
47      {}%
48    \end{tabular}
49    \par\endgroup
50 }
```

A song using the *tabular* template:

```
1 \documentclass{article}
2 \usepackage{leadsheets}
3 \usepackage{array}
4 \setleadsheets{title-template = tabular}
5 \begin{document}
6
7 \begin{song}{title={Mary Had A Little Lamb}, interpret={Stevie Ray Vaughan},
8     genre={blues}, tags={srv,blues},key=E}
9 \begin{verse}
10   ^{E7} Mary had a little lamb ^{A7} \\
11   Its fleece was white as ^{E7}snow, yea \\
12   Everywhere the child ^{B7}went ^{A7} \\
13   The little lamb was sure to ^{E7}go, yea
14 \end{verse}
15 \begin{verse}
16   ^{E7} He followed her to school one day ^{A7} \\
```

```
17    And broke the teacher's ^{E7}rule \\
18    What a time did they ^{B7}have ^{A7} \\
19    That day at ^{E7}school {}
20  \end{verse}
21  \end{song}
22
23  \end{document}
```

1   Mary Had A Little Lamb

as interpreted by Stevie Ray Vaughan

Genre: blues
Key: E

E^7 A^7
    Mary had a little lamb
                E^7
Its fleece was white as snow, yea
           B^7 A^7
Everywhere the child went
              E^7
The little lamb was sure to go, yea

14.2. Verse-type Templates

14.2.1. Background

Similar to the songtitles also the `verse` like environments are typeset using templates. Defining them is just as easy as for the song titles.

14.2.2. Existing Templates

Currently LEADSHEETS provides only one template:

itemize Uses an `itemize` environment for typesetting the corresponding environment.

14.2.3. Own Templates

Own templates can be defined using these commands:

`\defineversetypetemplate{`⟨*name*⟩`}{`⟨*begin code*⟩`}{`⟨*end code*⟩`}`
  This defines the template ⟨*name*⟩.

`\verselabel`
  Used inside `\defineversetypetemplate`. This determines where the label of the environment using the template will be displayed.

`\verselabelformat`
  Used inside `\defineversetypetemplate`. The format of the current environment as set with the corresponding `format` option.

`\verseafterlabel`
  Used inside `\defineversetypetemplate`. The format of the current environment as set with the corresponding `after-label` option.

`\versename`
  Used inside `\defineversetypetemplate`. This prints the name of the current environment.

`\versenumber`
  Used inside `\defineversetypetemplate`. Expands to the `\the`⟨*environment*⟩ command for the current environment.

`\ifversestarred{`⟨*true*⟩`}{`⟨*false*⟩`}`
  Can be used inside `\defineversetypetemplate` for checking if the current environment was defined by the starred version of `\newversetype`.

`\ifversenumbered{`⟨*true*⟩`}{`⟨*false*⟩`}`
  Can be used inside `\defineversetypetemplate` for checking if for the current environment the option `numbered` is true or false.

`\ifversenamed{`⟨*true*⟩`}{`⟨*false*⟩`}`
  Can be used inside `\defineversetypetemplate` for checking if for the current environment the option `named` is true or false.

`\ifobeylines{`⟨*true*⟩`}{`⟨*false*⟩`}`
  Checks if for the current song the option `obey-lines` is true or false.

Since with `\defineversetypetemplate` you define a template for an environment it has two argument for code: one for code at the beginning of the environment and one for code at the end. The command `\verselabel` internally uses the conditionals. Its definition is equivalent to the following:

```
1  \newcommand*\verselabel{%
2    \ifversestarred
3      {}
4      {%
5        \verselabelformat
6        \ifversenamed
7          {%
8            \versename
9            \ifversenumbered{ }{}%
10         }
11         {}%
```

```
12          \ifversenumbered
13            {\versenumber}
14            {}%
15          \verseafterlabel
16        }%
17 }
```

14.2.4. Examples

In order to give you an idea on how to use verse-type templates I'll show you how the existing ones are defined and a few new definitions.

The *itemize* template   This is how the *itemize* is defined.

```
1 \makeatletter
2 \defineversetypetemplate{itemize}
3   {%
4     \itemize
5       \@itemdepth=0
6       \ifobeylines
7         {%
8           \setlength{\parskip}{0pt}%
9           \setleadsheets{ obey-lines-parskip = \parsep }%
10        }
11        {}%
12      \item[{\verselabel}]%
13    }
14  {\enditemize}
15 \makeatother
```

The most interesting part is probably the `\ifobeylines` part. When the option `obey-lines` is set to true an end of a line inserts a `\par` token. So in order not to get a vertical skip after every line the template sets `\parskip` to zero. With `obey-lines` = {true} an empty line also inserts a `\par` token but it also inserts a vertical space according to the value set with `obey-lines-parskip`. This option can only be set in a verse-type template definition (which is why it isn't documented elsewhere). All `verse` like environments initialize the length to the current value of `\parskip` *before* the template code is inserted.

A *flushleft* template   An example for a template *flushleft* that typesets the label in the margin:

```
1 \defineversetypetemplate{flushleft}
2   {%
3     \noindent\llap{\verselabel\space}%
4     \flushleft
5     \unskip
6     \vspace*{-\baselineskip}
7     \ifobeylines{\setlength\parskip{0pt}}{\vspace*{-\parskip}}%
8   }
9   {\endflushleft}
10 \begin{chorus}[template=flushleft]
11   Lorem ipsum dolor sit amet, consetetur sadipscing elitr, \\
12   sed diam nonumy eirmod tempor invidunt ut labore et dolore \\
13   magna aliquyam erat, sed diam voluptua.
14
15   Lorem ipsum dolor sit amet, consetetur sadipscing elitr, \\
16   sed diam nonumy eirmod tempor invidunt ut labore et dolore \\
17   magna aliquyam erat, sed diam voluptua.
18 \end{chorus}
```

Chorus: Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
sed diam nonumy eirmod tempor invidunt ut labore et dolore
magna aliquyam erat, sed diam voluptua.
Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
sed diam nonumy eirmod tempor invidunt ut labore et dolore
magna aliquyam erat, sed diam voluptua.

A *flushright* template   An example for a template *flushright* that typesets the label in the margin:

```
1 \defineversetypetemplate{flushright}
2   {%
3     \noindent\llap{\verselabel\space}%
4     \flushright
5     \unskip
6     \vspace*{-\baselineskip}
7     \ifobeylines{\setlength\parskip{0pt}}{\vspace*{-\parskip}}%
8   }
9   {\endflushright}
10 \begin{chorus}[template=flushright]
11   Lorem ipsum dolor sit amet, consetetur sadipscing elitr, \\
12   sed diam nonumy eirmod tempor invidunt ut labore et dolore \\
13   magna aliquyam erat, sed diam voluptua.
```

```
14
15    Lorem ipsum dolor sit amet, consetetur sadipscing elitr, \\
16    sed diam nonumy eirmod tempor invidunt ut labore et dolore \\
17    magna aliquyam erat, sed diam voluptua.
18 \end{chorus}
```

Chorus:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
sed diam nonumy eirmod tempor invidunt ut labore et dolore
magna aliquyam erat, sed diam voluptua.
Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
sed diam nonumy eirmod tempor invidunt ut labore et dolore
magna aliquyam erat, sed diam voluptua.

As you can see it's not entirely easy to define a template that suits both songs with and without obey-lines = {true}. Personally I would forget about that option and not care about it at all in my templates.

A *framed* template   Last but not least an example using the mdframed package [Dan13] – just to show you that everything is possible. The example adapts one of the examples of mdframed's manual.

```
1 \defineversetypetemplate{framed}
2   {%
3     \expandcode{%
4       \noexpand\mdframed[
5         \ifversestarred{}{%
6           frametitle={%
7             \noexpand\tikz[baseline=(current bounding box.east),outer sep=0pt]
8               \noexpand\node[anchor=east,rectangle,fill=blue!20]
9                 {\noexpand\strut\noexpand\verselabel};
10          }%
11        },
12        roundcorner = 5pt ,
13        linecolor = blue!20 ,
14        linewidth = 2pt,
15        topline = true,
16        frametitleaboveskip = \dimexpr-\ht\strutbox\relax ,
17      ]%
18    }%
19    \ifobeylines{\setlength\parskip{0pt}}{}%
20  }
21  {%
22    \endmdframed
```

```
23      \addvspace{\baselineskip}%
24    }
25  \begin{chorus}[template=framed]
26    Lorem ipsum dolor sit amet, consetetur sadipscing elitr, \\
27    sed diam nonumy eirmod tempor invidunt ut labore et dolore \\
28    magna aliquyam erat, sed diam voluptua.
29
30    Lorem ipsum dolor sit amet, consetetur sadipscing elitr, \\
31    sed diam nonumy eirmod tempor invidunt ut labore et dolore \\
32    magna aliquyam erat, sed diam voluptua.
33  \end{chorus}
```

Chorus:

Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
sed diam nonumy eirmod tempor invidunt ut labore et dolore
magna aliquyam erat, sed diam voluptua.
  Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
sed diam nonumy eirmod tempor invidunt ut labore et dolore
magna aliquyam erat, sed diam voluptua.

15. **Internationalization**

Th environments described in sections 9 and 12 as well as a few other words used in LEADSHEETS are translated with the help of the translations package [Nie13]. All predefined and available translation strings are listed in table 4. You can change those translations or add translations for other languages with this command:

\DeclareTranslation{⟨*language*⟩}{⟨*string*⟩}{⟨*translation*⟩}
The command provided by the translations package for translating strings.

Those translations can be used for example in song title templates (see section 14.1). One of the strings listed in table 4 is a little different: the string leadsheets/interpret is declared as

```
1  \DeclareTranslation{English}{leadsheets/interpret}
2    {as interpreted by \printsongpropertylist{interpret}{ \& }{, }{ \& }}
3  \DeclareTranslation{German}{leadsheets/interpret}
4    {wie von \printsongpropertylist{interpret}{ \& }{, }{ \& } interpretiert}
```

which means it uses the song property interpret. As a consequence it only really can be used inside a song environment. In other cases as for example in table 4 the property part

expands to nothing (but the spaces around it are of course there). Also keep in mind that
`\printsongpropertylist` is not expandable.

TABLE 4: Predefined translation strings.

String	English	German
leadsheets/major	major	Dur
leadsheets/minor	minor	Moll
leadsheets/chorus	Chorus	Refrain
leadsheets/verse	Verse	Strophe
leadsheets/composer	Composer	Komponist
leadsheets/lyrics	Lyrics	Text
leadsheets/key	Key	Tonart
leadsheets/capo	Capo	Kapo
leadsheets/fret	fret	Bund
leadsheets/interpret	as interpreted by	wie von interpretiert
leadsheets/intro	Intro	Intro
leadsheets/interlude	Interlude	Interlude
leadsheets/bridge	Bridge	Bridge
leadsheets/solo	Solo	Solo

Part V.
Appendix

A. References

[Dan13] Marco DANIEL. mdframed. version 1.9b, July 1, 2013.
URL: http://mirror.ctan.org/macros/latex/contrib/mdframed/.

[Ham12] Kevin W. HAMLEN. songs. version 2.14, Mar. 17, 2012.
URL: http://mirror.ctan.org/macros/latex/contrib/songs/.

[MC08] Frank MITTELBACH and David CARLISLE. array. version 2.4c, Sept. 9, 2008.
URL: http://mirror.ctan.org/macros/latex/required/tools/.

[Nie13] Clemens NIEDERBERGER. translations. version 1.1a, Sept. 30, 2013.
URL: http://mirror.ctan.org/macros/latex/contrib/translations/.

[Rat10] Christopher RATH. songbook. version 4.5, Apr. 30, 2010.
URL: http://mirror.ctan.org/macros/latex/contrib/songbook/.

[The13a] THE LATEX3 PROJECT TEAM. l3kernel. version SVN 4582, July 28, 2013.
URL: http://mirror.ctan.org/macros/latex/contrib/l3kernel/.

[The13b] THE LATEX3 PROJECT TEAM. l3packages. version SVN 4582, July 28, 2013.
URL: http://mirror.ctan.org/macros/latex/contrib/l3packages/.

[Var00] VARIOUS. *The Real Book. C Edition*. 6th edition. Vol. I.
Hal Leonard Publishing Corporation, Jan. 1, 2000. ISBN: 978-0634060380.

B. Index